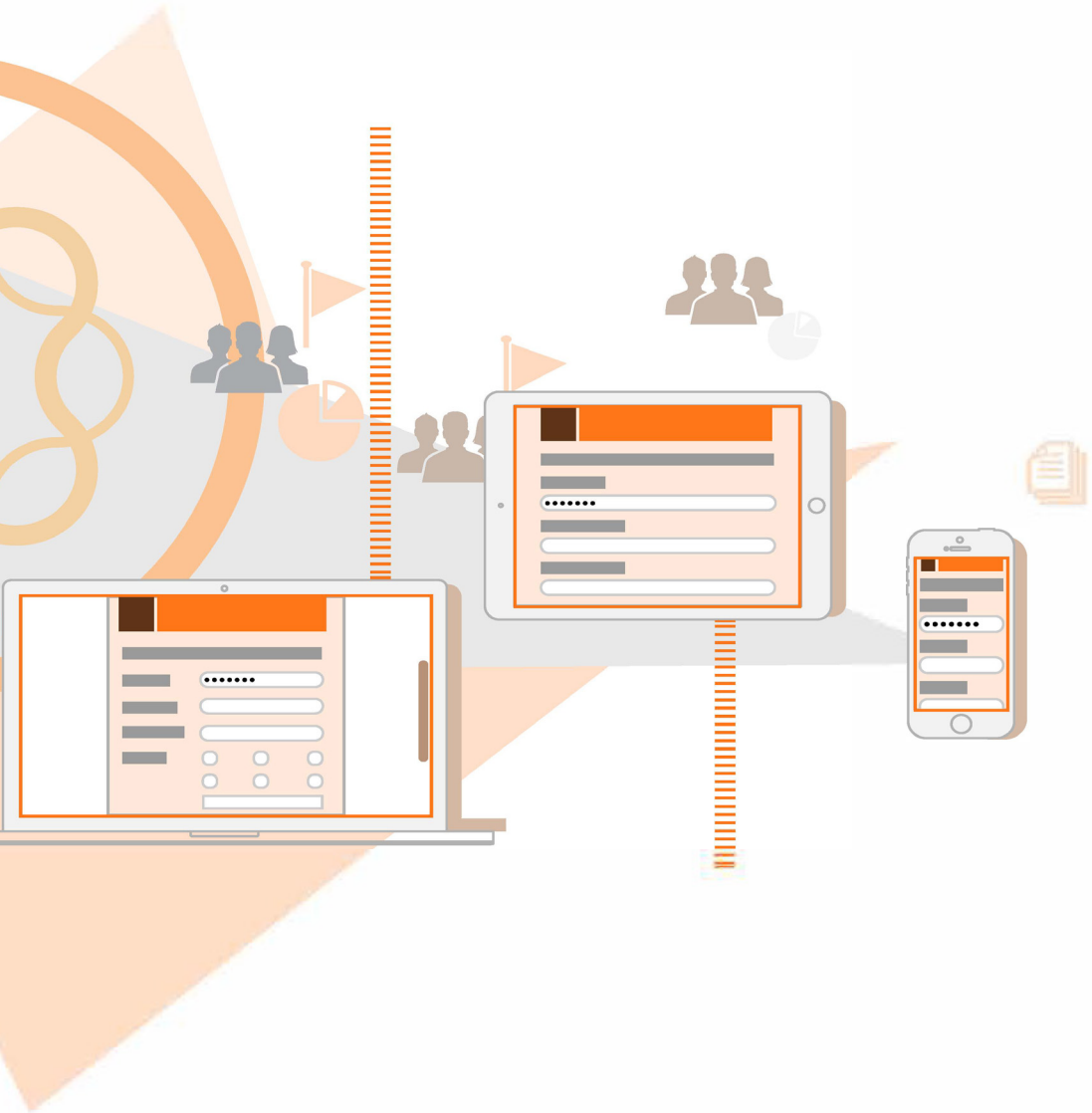
Skriptgrundlagen für Designer



AEM 6.2 Forms

Rechtliche Hinweise

Rechtliche Hinweise finden Sie unter https://helpx.adobe.com/de/legal/legal-notices.html.

Inhaltsverzeichnis

1. Informationen zu diesem Dokument

Willkommen zur Dokumentation „Grundlagen zum Erstellen von Skripten mit Designer“ *Grundlagen zum Erstellen von Skripten* liefert Ihnen einen Überblick darüber, wie Sie Designer-Berechnungen und -Skripte für die Entwicklung und Erweiterung von in Designer erstellten Formularen einsetzen.

Sie können Berechnungen und Skripts für die Ausführung der folgenden Aktionstypen verwenden:

- Änderung von Verhalten und Erscheinungsbild von Objekten zur Laufzeit
- Steuerung der Darstellung von Feldwerten
- Interaktion mit Formularausfüllern über Dialogfelder und visuelle Hinweise
- Automatisierung der Formularausfüllung
- Steuerung der Host-Umgebung
- Interaktion mit Webdiensten
- Interaktion mit Datenbanken und Ausfüllen von Formularen mit Daten aus Datenquellen

HINWEIS: *In diesem Dokument haben die Begriffe Adobe Experience Manager Forms, AEM Forms, AEM Forms on JEE und LiveCycle dieselbe Bedeutung und sind untereinander austauschbar.*

1.1. Zielsetzung dieser Einleitung

Grundlagen zum Erstellen von Skripten ist für Formularverfasser und -entwickler konzipiert, die Berechnungen und Skripten zur Erweiterung ihrer Designer-Formulare einsetzen möchten. Es wird angenommen, dass Sie Kenntnisse über Skriptsprachen haben, besonders JavaScript™sowie Objektmodelle. Sie sollten mit Adobe® Acrobat® Professional- oder Acrobat Standard vertraut sein und sich mit dem Arbeiten in einer strukturierten XML-Umgebung auskennen.

Grundlagen zum Erstellen von Skripten geben Sie die folgenden Informationen an:

- Einführung in die Verwendung von Designer-Berechnungen und -Skripten zur Formularerweiterung
- Verständliche Anweisungen und Beispiele zu den Designer-Berechnungs- und -Skriptfunktionen
- Verweise auf weiterführende Informationsquellen zu Designer-Skriptfunktionen und ähnlichen Technologien

  Nach dem Lesen dieser Einleitung sollten Sie über ausreichende Kenntnisse zur Anwendung der Designer-Berechnungen und -Skripten verfügen. Bei der Entwicklung von Formularen verfügen Sie anhand der in dieser Einleitung enthaltenen Erläuterungen und Beispiele Sie über entsprechende Anweisungen und Informationen zur erfolgreichen Fertigstellung Ihrer Projekte.

1.2. Zusätzliche Informationen

Adobe bietet zahlreiche Informationsquellen zur Designer-Skripterstellung, sowohl für Formularverfasser als auch für Entwickler von Formularen.

1.2.1. Designer-Hilfe

Die Designer-Hilfe enthält detaillierte Informationen zur Produktverwendung, darunter auch Hinweise zur Verwendung von Berechnungen und Skripten, und dient als erste Informationsquelle bei der Suche nach Informationen zu Themen im Zusammenhang mit Designer. Sie können auf folgende Ressourcen zugreifen *Designer-Hilfe* über das Hilfe-Menü oder online unter Designer-Hilfe.

1.2.2. Grundlagen zum Erstellen von Skripten

Diese Einleitung liefert einen Überblick über das Erstellen von Berechnungen und Skripten mit Designer. Sie enthält ausführliche Anleitungen zur Erstellung von Berechnungen und Skripten mit FormCalc und JavaScript.

1.2.3. Skriptreferenz

Die Designer-Skriptreferenz ist eine umfangreiche Informationsquelle zu Modellen, Objekten, Eigenschaften und Methoden, die in verwendet werden können. Das PDF-Dokument dient nur zum Nachschlagen einzelner Informationen und enthält keinerlei Anleitungen zum Erstellen von Berechnungen und Skripten.

Siehe *Skriptreferenz*.

1.2.4. Benutzerforen

Das Designer-Forum dient dem Austausch von Experten zu Themen im Zusammenhang mit Designer. Sie können Fragen anderer Leser beantworten, Produktprobleme melden oder eigene Fragen an andere Formularverfasser oder Adobe-Experten stellen. Weitere Informationen finden Sie unter www.adobeforums.com.

1.2.5. Beispielskripten

Beispielskripten sind funktionstüchtige Formulare oder Pakete mit Anweisungen zur Erstellung des Beispiels und mit den zum Erstellen und Anzeigen des Formulars verwendeten Beispieldaten. Adobe-Experten und andere Unternehmen fügen regelmäßig neue Beispiele hinzu. Siehe Developer Center.

2. Skripterstellung mit Designer

Bei der Gestaltung von Formularen kann der Entwickler Berechnungen und Skripten verwenden, um den Gebrauchswert für den Benutzer zu erhöhen. Sie können die meisten Formularfelder und -objekte durch Berechnungen und Skripten ergänzen. Das folgende JavaScript-Skript multipliziert beispielsweise die Werte von zwei numerischen Feldern und zeigt das Ergebnis in einem dritten numerischen Feld an:

```
    NumericField3.rawValue = NumericField1.rawValue * Numeric-
Field2.rawValue;
```

Auf einem anspruchsvolleren Niveau können Sie dann eigene Funktionen erstellen, die auf Ihre individuellen Anforderungen hinsichtlich der Verarbeitung benutzerdefinierter Formulare zugeschnitten sind.

Designer unterstützt zwei Skriptsprachen, die jeweils auf die Anforderungen einer bestimmten Gruppe von Formularentwicklern abgestimmt sind. FormCalc ist eine unkomplizierte, einfach anzuwendende Berechnungssprache, deren Funktionalität sich an gebräuchlicher Tabellenkalkulations-Software orientiert. Diese Skriptsprache enthält eine Vielzahl integrierter Funktionen, mit deren Hilfe Sie Formularentwürfe im Handumdrehen erstellen können. JavaScript ist eine leistungsfähige Skriptsprache, die viel Flexibilität bei der Erstellung von Skripten bietet, wobei Sie bereits vorhandene Kenntnisse über diese Sprache nutzen können.

Beachten Sie, dass es Ihnen völlig freigestellt ist, ob Sie in einem Formular Skripterstellung verwenden. Sie können sich zwar für die Vorteile der Skripterstellung entscheiden, um den Gebrauchswert des Formulars für den Benutzer zu erhöhen, viele der leistungsstarken Funktionen stehen in Designer aber auch ohne die Anwendung von Skripten zur Verfügung. Durch Skripterstellung können Sie jedoch fast alle Aspekte der Formulargestaltung steuern.

HINWEIS: *Sie können gängige interaktive Funktionen in Formularen mit flexiblem Layout auch über das Dialogfeld „Aktionsgenerator“ im Menü „Extras“ erstellen, ohne Skripten zu schreiben.*

2.1. Funktionsweise von Skripten

Designer-Skripterstellung funktioniert auf der Grundlage eines ereignisbasierten Modells, das es Ihnen erlaubt, verschiedene Aspekte von Objekten in einem Formular zur Laufzeit zu ändern. Sie fügen als Formularentwickler Skripten zu Objekten in Abhängigkeit vom gewünschten Ausführungszeitpunkt des Skripts hinzu. Sie platzieren beispielsweise das folgende Skript auf das `click` -Ereignis eines Schaltflächenobjekts, sodass zur Laufzeit, wenn ein Benutzer auf die Schaltfläche klickt, ein Feld mit einer Meldung angezeigt wird:

```
    xfa.host.messageBox("This is a message for a form filler.",
"Benutzer-Feedback", 3);
```

Einem bestimmten Ereignis zugeordnete Skripten werden immer dann ausgeführt, wenn das jeweilige Ereignis stattfindet. Einige Ereignisse können mehrmals innerhalb einer Formularausfüllsitzung stattfinden. Das folgende Skript erhöht beispielsweise den aktuellen Wert eines numerischen Felds um den Wert 1:

```
NumericField1.rawValue = NumericField1.rawValue + 1;
```

Wenn Sie dieses Skript zum `calculate` -Ereignis für `NumericField1` hinzufügen, wenn Sie das Formular das erste Mal öffnen, zeigt `NumericField1` den Wert `2` an. Dies zeigt an, dass das `calculate` -Ereignis in der Reihenfolge der Ereignisse beim Öffnen des Formulars zweimal stattgefunden hat.

VERKNPFTE LINKS:

Ereignisse

Objekte, die Berechnungen und Skripte unterstützen

Beziehungen zwischen Objekten in der Objektbibliothek

2.2. Objekte, die Berechnungen und Skripte unterstützen

Die folgende Tabelle gibt Ihnen einen Überblick über die Skriptunterstützung für die Standardobjekte, die in Designer auf der Palette „Objektbibliothek“ enthalten sind.

Objekte, die Berechnungen und Skripte unterstützen	Objekte, die keine Berechnungen und Skripten unterstützen
Barcodes	Kreis
Schaltfläche	Inhaltsbereich
Kontrollkästchen	Linie
Datums-/Uhrzeitfeld	Rechteck
Dezimalfeld	Bild
Unterschriftsfeld	Teilformularsätze
Dropdown-Liste	Tabellenabschnitte
E-Mail-Senden-Schaltfläche	Text
HTTP-Senden-Schaltfläche	
Bildfeld	
Listenfeld	

Objekte, die Berechnungen und Skripte unterstützen	Objekte, die keine Berechnungen und Skripten unterstützen
Numerisches Feld	
Papierformular-Barcode	
Kennwortfeld	
Drucken-Schaltfläche	
Optionsfeld	
Zurücksetzen-Schaltfläche	
Teilformular	
Tabelle (einschließlich Textzeilen, Kopf- und Fußzeilen)	
Textfeld	

VERKNPFTE LINKS:

Beziehungen zwischen Objekten in der Objektbibliothek

2.3. Beziehungen zwischen Objekten in der Objektbibliothek

Beim Erstellen von Berechnungen und Skripten in Designer sollten Sie daran denken, dass die Objekte, für die Sie Skripten hinzufügen, in der zugrunde liegenden XML-Formulararchitektur als XML-Objekte definiert sind. Die Registerkarte „Standard“ der Palette „Objektbibliothek“ enthält zwar eine breite Vielfalt von Objekten, jedoch werden viele dieser Objekte durch dasselbe XML-Objekt definiert. Die verschiedenen verfügbaren Skripteigenschaften und -methoden basieren daher auf der Definition des XML-Objekts und nicht auf dem Objekt in der Palette „Objektbibliothek“.

Auf der Registerkarte „Standard“ in der Palette „Objektbibliothek“ verfügbare Objekte, die auf derselben XML-Objektdefinition basieren, verwenden einen Satz gemeinsamer Eigenschaften und Methoden. Wenn Sie den Abschnitt Skriptobjekte lesen, können Sie sich mit den verfügbaren Eigenschaften und Methoden vertraut machen. Jede zugrunde liegende XML-Objektdefinition enthält wiederum ein untergeordnetes Objekt, das das Erscheinungsbild des Designer-Objekts steuert.

Beispiel: Für das Durchsuchen der Eigenschaften und Methoden, die in Designer für das Objekt Datum/Uhrzeit-Feld zur Verfügung stehen, beginnen Sie mit dem `field` -Objekt. Falls Sie sich das entsprechende XML-Objekt ansehen möchten, das das Erscheinungsbild des Datums-/Uhrzeitfelds steuert, zeigen Sie das `dateTimeEdit` -Objekt an.

In der nachstehenden Tabelle wird die Zuordnung der auf der Registerkarte „Standard“ in der Palette „Objektbibliothek“ von Designer angezeigten Objekte zu dem entsprechenden XML-Formulararchitektur-Objekt veranschaulicht.

Objektbibliothek-Standardobjekt	XML-Formulararchitektur-Objekt (Basisobjekt)	XML-Formulararchitektur-Objekt (UI)
Barcodes	field	barcode
Schaltfläche	field	Schaltfläche
Kontrollkästchen	field	checkButton
Datums-/Uhrzeitfeld	field	dateTime
Dezimalfeld	field	numericEdit
Unterschriftsfeld	field	signature
Dropdown-Liste	field	choiceList
E-Mail-Senden-Schaltfläche	field	Schaltfläche
HTTP-Senden-Schaltfläche	field	Schaltfläche
Bildfeld	field	imageEdit
Listenfeld	field	choiceList
Numerisches Feld	field	numericEdit
Papierformular-Barcode	field	barcode
Kennwortfeld	field	passwordEdit
Drucken-Schaltfläche	field	Schaltfläche
Optionsfeld	field	checkButton
Schaltfläche „Zurücksetzen“	field	Schaltfläche
Teilformular	subform	Nicht zutreffend
Tabelle (einschließlich Textzeilen, Kopf- und Fußzeilen)	subform	Nicht zutreffend
Textfeld	field	textEdit

VERKNPFTE LINKS:

Objekte, die Berechnungen und Skripte unterstützen

2.4. Skript-Editor

Der Skript-Editor ist die Komponente, in der Sie die Berechnungen und Skripten für ein bestimmtes Formular erstellen, modifizieren und anzeigen. Mit dem Skript-Editor können Sie beispielsweise eine einfache Berechnung schreiben, die die Summe der Werte zweier numerischer Felder berechnet, oder auch komplexere Skripten erstellen, die die Darstellung des Formulars in Abhängigkeit von den Aktionen des Endbenutzers anpasst. Designer unterstützt die Skripterstellung in der anwendungseigenen Programmiersprache FormCalc oder in JavaScript.

Der Skript-Editor wird standardmäßig oben im Designer-Arbeitsbereich angezeigt, kann aber auch an jeder beliebigen anderen Stelle verankert werden. Er verfügt sowohl über eine einzeilige als auch über eine mehrzeilige Ansicht, zwischen denen Sie je nach Anforderung jederzeit wechseln können. Die einzeilige Ansicht soll dafür sorgen, dass möglichst viel Platz für den Layout-Editor und andere Paletten zur Verfügung steht, wogegen die mehrzeilige Ansicht möglichst viel Platz für die Skripterstellung bereitstellen soll.

Anzeigen

Listet alle Ereignisse des Formularentwurfs auf, die benutzerdefinierte Skripterstellung unterstützen. Alle Ereignisse, die für ein bestimmtes Objekt nicht gültig sind, werden grau dargestellt. Enthält ein Ereignis eine Berechnung oder ein Skript, steht neben dem Namen dieses Ereignisses ein Sternchen (*).

Ereignisse für untergeordnete Objekte anzeigen

Zeigt das Ereignis an, das gegenwärtig unter „Anzeigen“ für das aktuelle Objekt und alle untergeordneten Objekte ausgewählt ist. Wenn Sie das oberste Objekt in der Palette „Hierarchie“ auswählen, zeigt diese Option das Ereignis an, das Sie derzeit in der Liste „Anzeigen“ für alle Objekte auf Ihrem Formular ausgewählt haben.

Funktionen

Zeigt eine Liste der verfügbaren integrierten FormCalc- bzw. JavaScript-Funktionen an, je nachdem, welche Skriptsprache Sie aktuell in der Liste „Sprache“ ausgewählt haben.

Um eine Funktion in das Skript-Bearbeitungsfeld einzufügen, wählen Sie eine Funktion in der Liste aus und drücken die Eingabetaste.

Skriptsyntax prüfen

Prüft alle Skripten eines Formulars auf korrekte Syntax und erstellt einen in der Palette „Bericht“ auf der Registerkarte „Warnungen“ einsehbaren Fehlerbericht.

Sprache

Legt fest, welche Skriptsprache Sie für die aktuelle Berechnung bzw. das aktuelle Skript verwenden möchten. Die folgenden Optionen stehen zur Auswahl:

- **FormCalc** FormCalc ist eine anwendungseigene Adobe-Berechnungssprache, die gewöhnlich für kürzere Skripten wie etwa einfache Berechnungen verwendet wird.
- **JavaScript** JavaScript ist die Standard-Skriptsprache für neue Formulare. (Siehe So legen Sie die Standard-Skriptsprache für neue Formulare fest lautete.)

Die in der Liste „Sprache“ angezeigte Skriptsprache stimmt mit der Skriptsprache überein, die Sie im Dialogfeld „Optionen“ im Bereich „Arbeitsbereich“ als Standardsprache für neue Formulare ausgewählt haben. Wenn Sie jedoch im Dialogfeld „Formulareigenschaften“ auf der Registerkarte „Standard“ die Einstellung für die Skriptsprache für das aktuelle Formular ändern, wird diese Änderung in die Liste „Sprache“ übernommen und für alle neuen Skripten in Verbindung mit neuen Ereignissen verwendet. Die Änderung der Option für die Skriptsprache im Dialogfeld „Formulareigenschaften“ hat keine Änderung der Skriptsprache für vorhandene Skripten zur Folge. Wenn ein Ereignis bereits Skript enthält und dieses Skript gelöscht wird, verwendet der Skript-Editor für die Dauer der Designer-Sitzung weiterhin dieselbe Skriptsprache.

Ausführen am

Gibt an, wo die Berechnung bzw. das Skript ausgeführt werden soll. Es stehen drei Optionen zur Auswahl:

- **Client** Berechnungen und Skripte werden ausgeführt, während die Client-Anwendung (z. B. Acrobat, Adobe® Reader® oder ein Webbrowser) das Formular verarbeitet.
- **Server** Berechnungen und Skripte werden ausgeführt, während das Formular in der entsprechenden Serveranwendung (z. B. Forms Generator) verarbeitet wird.
- **Client und Server** -Berechnungen und Skripten werden ausgeführt, während während das Formular in der entsprechenden Serveranwendung (z. B. Forms) verarbeitet wird, es sei denn, die HTML-Client-Anwendung unterstützt clientseitige Skripterstellung. Beispiel: Ein Skript, das auf eine Datenbank zugreift, um automatisch Daten in das Formular einzutragen.

Ereignisübertragung

Um das Kontrollkästchen „Ereignisübertragung“ anzuzeigen, gehen Sie zu „Extras“ > „Optionen“ und aktivieren Sie auf der Registerkarte „Arbeitsbereich“ das Kontrollkästchen zur Anzeige der Option für die Ereignisübertragung.

Durch Aktivierung der Ereignisübertragung im Skript-Editor werden die Skripte global. Mithilfe der Einstellung werden Formularereignisse an vorherige Container weitergeleitet. Die Ereignisübertragung kann die Anzahl der Skripte in einem Formular reduzieren. Sie können beispielsweise ein globales Skript zum Steuern der Anzeige von ungültigen Feldern, Teilformularen oder Ausschlussgruppen erstellen. Hier sind einige Beispiele für globale Ereignisse:

- Ein Ereignis „enter/exit/mouseEnter/mouseExit“, das das aktive Feld farblich darstellt.
- Ein change-Ereignis, das Tastenanschläge für eine Formularsitzung verfolgt.

3. Konfigurieren von Designer für die Skripterstellung

3.1. So zeigen Sie den Skript-Editor an

1) Wählen Sie „Fenster“ > „Skript-Editor“.

HINWEIS: *Wird der Skript-Editor im Designer-Arbeitsbereich angezeigt, können Sie ihn mit Hilfe der Schaltfläche „Erweitern“ schnell andocken oder lösen.*

3.2. So wechseln Sie von der einzeiligen zur mehrzeiligen Ansicht

1) Ziehen Sie die Palettenleiste des Skript-Editors, bis die Palette die erforderliche Größe hat.

HINWEIS: *Die mehrzeilige Ansicht erweitert die Liste „Anzeigen“ um die Optionen „Alle Ereignisse“ und „Ereignisse mit Skripten“. Die Option „Alle Ereignisse“ zeigt alle Ereignisse für ein bestimmtes Formularentwurfsobjekt an, und zwar auch dann, wenn die Ereignisse keine Berechnungen oder Skripten enthalten. Die Option „Ereignisse mit Skripten“ zeigt nur diejenigen Ereignisse eines bestimmten Objekts an, die Berechnungen oder Skripten enthalten.*

3.3. So legen Sie die Standard-Skriptsprache für neue Formulare fest

1) Wählen Sie „Extras“ > „Optionen“.
2) Klicken Sie auf „Arbeitsbereich“.
3) Wählen Sie in der Liste „Standardsprache für neue Formulare“ die Standard-Skriptsprache für neue Formulare aus.

3.4. So legen Sie die Standard-Skriptsprache für das aktuelle Formular fest

1) Wählen Sie Datei > Formulareigenschaften.
2) Klicken Sie auf die Registerkarte „Standard“.
3) Wählen Sie in der Liste „Standardsprache“ die Standard-Skriptsprache für das aktuell angezeigte Formular aus.

3.5. So legen Sie die Standard-Skriptsprache für das aktuelle Formular fest

1) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.
2) Klicken Sie auf die Registerkarte „Standard“.
3) Wählen Sie in der Liste „Standardsprache“ die Standard-Skriptsprache für das aktuell angezeigte Formular aus.

3.6. So legen Sie die Standard-Skriptsprache für eine Formularvorlage fest

1) Erstellen Sie einen neuen Formularentwurf.
2) Wählen Sie Datei > Formulareigenschaften.
3) Klicken Sie auf die Registerkarte „Standard“.
4) Wählen Sie in der Liste „Standardsprache“ die standardmäßige Sprache für die Skripterstellung aus.
5) Erstellen Sie eine Sicherungskopie der ursprünglichen Vorlagendatei im Ordner „Templates“, der sich im Installationsverzeichnis von Designer befindet.
6) Speichern Sie den neuen Formularentwurf als TDS-Datei und überschreiben Sie die dazugehörige Formularvorlage. Beispiel: Speichern Sie die Datei unter dem Namen „Letter.tds“ und überschreiben Sie die Datei „Letter.tds“ im Ordner „Templates\Blank“.

3.7. So legen Sie die Standard-Skriptsprache für eine Formularvorlage fest

1) Erstellen Sie einen neuen Formularentwurf.
2) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.
3) Klicken Sie auf die Registerkarte „Standard“.
4) Wählen Sie in der Liste „Standardsprache“ die standardmäßige Sprache für die Skripterstellung aus.
5) Erstellen Sie eine Sicherungskopie der ursprünglichen Vorlagendatei im Ordner „Templates“, der sich im Installationsverzeichnis von Designer befindet.
6) Speichern Sie den neuen Formularentwurf als TDS-Datei und überschreiben Sie die dazugehörige Formularvorlage. Beispiel: Speichern Sie die Datei unter dem Namen „Letter.tds“ und überschreiben Sie die Datei „Letter.tds“ im Ordner „Templates\Blank“.

3.8. So legen Sie die Standard-Anwendung für die Verarbeitung fest

1) Wählen Sie Datei > Formulareigenschaften.
2) Klicken Sie auf die Registerkarte „Standard“.
3) Wählen Sie aus der Liste „Standardausführung am“ die gewünschte Standardanwendung für die Verarbeitung.

   HINWEIS: *Durch diesen Vorgang wird der Wert der Standardanwendung für die Verarbeitung nur für die aktuelle Instanz des Formulars festgelegt.*

   Damit Sie die Standardanwendung für die Verarbeitung beim Erstellen eines Formulars nicht immer neu festlegen müssen, ändern Sie die zugehörige Vorlagendatei, die zum Erstellen von neuen Formularentwürfen verwendet wird.

3.9. So legen Sie die Standard-Anwendung für die Verarbeitung fest

1) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.
2) Klicken Sie auf die Registerkarte „Standard“.
3) Wählen Sie aus der Liste „Standardausführung am“ die gewünschte Standardanwendung für die Verarbeitung.

   HINWEIS: *Durch diesen Vorgang wird der Wert der Standardanwendung für die Verarbeitung nur für die aktuelle Instanz des Formulars festgelegt.*

   Damit Sie die Standardanwendung für die Verarbeitung beim Erstellen eines Formulars nicht immer neu festlegen müssen, ändern Sie die zugehörige Vorlagendatei, die zum Erstellen von neuen Formularentwürfen verwendet wird.

3.10. So ändern Sie die Standardanwendung für die Verarbeitung für eine Formularvorlage

1) Erstellen Sie einen neuen Formularentwurf.
2) Wählen Sie Datei > Formulareigenschaften.
3) Klicken Sie auf die Registerkarte „Standard“.
4) Wählen Sie aus der Liste „Standardausführung am“ die gewünschte Standardanwendung für die Verarbeitung.
5) Erstellen Sie eine Sicherungskopie der ursprünglichen Vorlagendatei im Ordner „Templates“, der sich im Installationsverzeichnis von Designer befindet.
6) Speichern Sie den neuen Formularentwurf als TDS-Datei und überschreiben Sie die dazugehörige Formularvorlage. Beispiel: Speichern Sie die Datei unter dem Namen „Letter.tds“ und überschreiben Sie die Datei „Letter.tds“ im Ordner „Templates\Blank“.

3.11. So ändern Sie die Standardanwendung für die Verarbeitung für eine Formularvorlage

1) Erstellen Sie einen neuen Formularentwurf.
2) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.
3) Klicken Sie auf die Registerkarte „Standard“.
4) Wählen Sie aus der Liste „Standardausführung am“ die gewünschte Standardanwendung für die Verarbeitung.
5) Erstellen Sie eine Sicherungskopie der ursprünglichen Vorlagendatei im Ordner „Templates“, der sich im Installationsverzeichnis von Designer befindet.
6) Speichern Sie den neuen Formularentwurf als TDS-Datei und überschreiben Sie die dazugehörige Formularvorlage. Beispiel: Speichern Sie die Datei unter dem Namen „Letter.tds“ und überschreiben Sie die Datei „Letter.tds“ im Ordner „Templates\Blank“.

3.12. So zeigen Sie arabische, hebräische, thailändische und vietnamesische Zeichen an

Zum Anzeigen von Zeichen in Arabisch, Hebräisch, Thai oder Vietnamesisch im Skript-Editor oder auf der Registerkarte „XML-Quelle“ müssen Sie die Schrifteinstellungen ändern, die Designer auf diesen Registerkarten verwendet. Andernfalls werden an Stelle der landesspezifischen Zeichen in Designer nur kleine Kästchen angezeigt.

1) Wählen Sie „Extras“ > „Optionen“ und dann links im Fenster den Eintrag „Arbeitsbereich“.

2) Wählen Sie eine der folgenden Optionen aus:

   - „FormCalc-Syntaxformatierung“, um die Schrift im Skript-Editor festzulegen, wenn FormCalc verwendet wird
   - „JavaScript-Syntaxformatierung“, um die Schrift im Skript-Editor festzulegen, wenn JavaScript verwendet wird
   - „Syntaxformatierung der XML-Quelle“, um die Schrift auf der Registerkarte „XML-Quelle“ festzulegen

3) Wählen Sie unter „Schrift“ eine Schrift aus, die Ihre Sprache unterstützt. Adobe Arabic unterstützt beispielsweise Arabisch, Adobe Hebrew unterstützt Hebräisch, Adobe Thai unterstützt Thai und Myriad® Pro und Minion® Pro unterstützt Vietnamesisch. Falls die erforderlichen Schriften noch nicht auf dem System installiert sind, können Sie sie aus dem Internet herunterladen.

4) Klicken Sie auf OK.

5) Klicken Sie auf „OK“, um das Dialogfeld „Optionen“ zu schließen.

3.13. Berechnungen und Skripten mit dem Arbeitsbereich debuggen

Zum Debugging von Berechnungen und Skripten stehen im Designer-Arbeitsbereich verschiedene Möglichkeiten zur Auswahl.

Die folgende Tabelle zeigt, wo Sie nützliche Debugging-Informationen in den verschiedenen Designer-Paletten und -Registerkarten finden und wie Sie diese nutzen können.

Arbeitsbereich	Zweck
Registerkarte „Warnungen“ der Palette „Bericht“	Zeigt Ziel und Warnmeldungen sowie Syntaxfehler für JavaScript oder FormCalc bei Verwendung des Befehls „Skriptsyntax prüfen“ über das Menü „Extras“ oder durch Klicken auf die Schaltfläche „Skriptsyntax prüfen“ in der Werkzeugleiste an. Weitere Informationen unter So prüfen Sie die Skriptsyntaxändern. Durch Doppelklicken auf eine Syntaxwarnung auf der Registerkarte „Warnungen“ wird das fehlerhafte Skript in den Skript-Editor geladen und die Zeile mit dem Fehler hervorgehoben. Durch Doppelklicken auf eine Warnmeldung wird das entsprechende Objekt in der Designansicht und auf der Palette „Hierarchie“ markiert. Durch anschließendes Drücken der Taste F1 erhalten Sie Informationen zum Beheben des jeweiligen Problems. Eine Prüfung auf JavaScript-Laufzeitfehler können Sie duch Aktivieren der JavaScript-Konsole durchführen. Weitere Informationen unter JavaScript-Debuggingändern.
Registerkarte „Bindung“ der Palette „Bericht“	Wenn ein Formularentwurf an eine Datenquelle gebundene Felder enthält, können Sie mit Hilfe der Registerkarte „Bindungen“ abhängig von der definierten Datenbindung Felderlisten anzeigen. Auf diese Weise können Sie z. B. nur Felder mit globaler Datenbindung oder nur Felder ohne definierte Datenbindung anzeigen. Dies ist vor allem bei Formularen mit einer großen Anzahl von Feldern mit Datenbindungen nützlich.
Registerkarte „Protokoll“ der Palette „Bericht“	Auf dieser Registerkarte werden Prüfungsmeldungen, Skriptausführungsfehler für JavaScript oder FormCalc sowie beim Importieren oder Speichern von Formularen oder bei Verwendung der Registerkarte „PDF-Vorschau“ von Designer erstellte Wiedergabefehler angezeigt.
„Hierarchie“, Palette	Mit der Palette „Hierarchie“ können Sie die Position eines Formularobjekts für eine Referenz-Syntax ermitteln. Bei der Palette „Hierarchie“ handelt es sich um eine grafische Darstellung der Struktur eines Formulars. Sie zeigt die Inhalte der Registerkarten „Masterseiten“ und „Designansicht“ an. In der Palette „Hierarchie“ werden auch referenzierte Objekte unter der Node „Referenzierte Objekte“ angezeigt. A *Referenzobjekt* ist ein Objekt, das nur bei Bedarf einem Formular hinzugefügt wird. Immer, wenn Daten über mehrere Seiten oder Inhaltsbereiche fließen, werden die Teilformulare für den Überlaufkopfbereich und den Überlauffußbereich an den entsprechenden Stellen in das Formular eingefügt.
Registerkarte „Bindung“ der Palette „Objekt“	Für jedes Designer-Objekt, das an eine Datenquelle gebunden werden kann, gibt es in der Palette „Objekt“ eine Registerkarte „Bindung“. Wenn Sie ein Objekt in Ihrem Formularentwurf an eine bestimmte Daten-Node aus Ihrer Datenverbindung binden, wird in der Liste „Datenbindung (Öffnen, Speichern, Absenden)“ eine gültige FormCalc-Referenzsyntax für den Zugriff auf diese Daten-Node angezeigt. Die FormCalc-Referenz-Syntax lässt sich auch zum Testen für andere Berechnungen oder Skripten verwenden.

Arbeitsbereich	Zweck
XML-Quelle, Registerkarte	Die Registerkarte „XML-Quelle“ enthält den XML-Code des Formularentwurfs. Der XML-Quellcode definiert alle Aspekte des Formulars. Auf der Registerkarte „XML-Quelle“ können Sie sich die XML-Formobjektmodellstruktur eines Formularentwurfs sowie die Beziehungen zwischen Objekten und Eigenschaften genauer ansehen. In der XML-Quelle entsprechen die XML-Elementnamen den Objektnamen im XML Form Object Model und die Attribute entsprechen Eigenschaften. Wenn Sie in der Palette „Hierarchie“ ein Objekt auswählen und dann auf die Registerkarte „XML-Quelle“ klicken, wird die erste Zeile des entsprechenden Elements hervorgehoben. Der Objektname in Designer, wie in der Palette „Hierarchie“ wird zum Wert des `name` Attributs in der XML-Quelle. In dem über „Extras“ > „Optionen“ aufgerufenen Dialogfeld stehen verschiedene Optionen zur Anzeige der Quelle auf der Registerkarte „XML-Quelle“ zur Auswahl. Dazu gehören beispielsweise das Ein- oder Ausblenden der Zeilennummern und die Einstellung der Syntaxfarbe. XML-Quellcode sollte nicht direkt bearbeitet werden.

Zum Debugging von Berechnungen und Skripten kann es außerdem hilfreich sein, die Standardoptionen für den Skript-Editor zu ändern. Diese Optionen finden Sie im Dialogfeld „Optionen“ im Bereich „Arbeitsbereich“. Wählen Sie „Extras“ > „Optionen“ und anschließend links in der Liste den Eintrag „Arbeitsbereich“. Sie können beispielsweise festlegen, dass Zeilennummern im Skript-Editor angezeigt werden. Außerdem können Sie die Formatierung der FormCalc- oder JavaScript-Syntax ändern.

4. Berechnungen und Skripten erstellen

Designer bietet für das Erstellen von Berechnungen und Skripten eine Vielzahl von Funktionen, mit deren Hilfe sich zahlreiche Aufgaben ausführen lassen. Das folgende Skript ändert beispielsweise die Farbe eines Textfeldrandes und die Schriftgröße des Textfeldwertes:

```
    TextField1.border.edge.color.value = "255,0,0";
TextField1.font.typeface = "Courier New";
```

Komplexere Formulare können mit Hilfe von Skripten zur Laufzeit Datenquellverbindungen vornehmen und Daten bearbeiten. Beispiele für gängige Aufgaben bei der Skripterstellung finden Sie unter Beispiele für gängige Aufgaben bei der Skripterstellungändern.

Für das Erstellen von Berechnungen und Skripten in Designer gibt es einen allgemeinen Prozess, der jedes Mal ausgeführt werden muss, wenn Sie eine Berechnung oder ein Skript an ein Objekt anhängen. Nicht alle Teile dieses Prozesses sind jedes Mal erforderlich, wenn Sie eine Berechnung oder ein Skript erstellen. Dennoch ist es empfehlenswert, diesen Prozess umzusetzen, um potenzielle Fehler und unerwartete Ergebnisse zu vermeiden.

Beim Erstellen einer Berechnung oder eines Skripts müssen generell die folgenden Schritte ausgeführt werden:

- Wählen Sie das Objekt aus, an das Sie eine Berechnung oder ein Skript anhängen möchten. Sie können zwar Berechnungen und Skripten erstellen, mit denen fast jedes Objekt im Formularentwurf bearbeitet werden kann, jedoch werden Formularereignisse nicht von allen Objekten des Formularentwurfs unterstützt. Eine Liste mit den Standardobjekten, die sich in der Palette „Objektbibliothek“ in Designer befindet, die die Skripterstellung unterstützen, finden Sie unter Objekte, die Berechnungen und Skripte unterstützen.
- Erstellen Sie die Berechnung bzw. das Skript im Skript-Bearbeitungsfeld des Skript-Editors.
- Unterziehen Sie die Berechnung oder das Skript entweder auf der Registerkarte „PDF-Vorschau“ oder in Ihrer Testumgebung einer gründlichen Überprüfung.

4.1. Namenskonventionen für Formularentwurfsobjekte und Variablen

Wenn Sie Berechnungen oder Skripten erstellen, um ein Formular anspruchsvoller zu gestalten, müssen Sie die im Formular verwendeten Namen für das Formularentwurfsobjekt und die Variablen sorgfältig auswählen. Die Namen von XML Form Object Model-Eigenschaften, -Methoden und -Objekten sollten nach Möglichkeit nicht für Formularentwurfsobjekte und Variablen verwendet werden. Andernfalls werden Berechnungen und Skripten möglicherweise nicht ordnungsgemäß ausgeführt.

Wenn Sie beispielsweise ein neues Textfeld mit dem Namen x in einem Teilformularobjekt mit dem Namen `Subform1` erstellen, greifen Sie auf das Textfeldobjekt mit Hilfe der folgenden Syntax zu:

```
Subform1.x.[Ausdruck]
```

Teilformularobjekte besitzen jedoch bereits eine XML-Formobjektmodell-Eigenschaft mit dem Namen `x`, das die horizontale Position des Teilformulars im Formularentwurf darstellt.

Um Namenskonflikte zu vermeiden, müssen Sie Feldbenennungskonventionen auswählen, die sich von denen des XML-Formularobjektmodells unterscheiden. Sie können folgende Feldnamen für das Textfeld in dem Beispiel oben verwenden:

- horizontalValue
- x_value
- xLetter
- hValue

Weitere Informationen sowie eine Liste mit den Namen der Eigenschaften, Methoden und Objekten für das XML-Formobjektmodell finden Sie unter Skriptreferenz.

VERKNPFTE LINKS:

Variablen benennen

Berechnungen und Skripten erstellen

4.2. Skriptsprache wählen

Designer unterstützt die Skripterstellung sowohl mit FormCalc als auch mit JavaScript. Jede Skriptsprache bietet ihre eigenen Vorteile, die Sie kennen sollten, bevor Sie mit der Erstellung von Skripten für Ihr Formular beginnen.

FormCalc ist eine Berechnungssprache mit einer umfangreichen Palette von integrierten Funktionen zur Vereinfachung der gebräuchlichsten Formularfunktionen. Beispielsweise können Sie mit den Finanzfunktionen von FormCalc anhand des Kreditbetrags, des Zinssatzes und der Anzahl der Zahlungsperioden die Höhe der pro Periode fälligen Zahlungsbeträge berechnen.

JavaScript ist eine leistungsstärkere und vielseitigere Skriptsprache, die Ihnen mehr Flexibilität bietet und die Möglichkeit eröffnet, Ihre bereits vorhandenen Skripterstellungskenntnisse zu nutzen. Beispielsweise können Sie vorhandene JavaScript-Funktionen in Designer wiederverwerten und sich so die Arbeit bei der Skripterstellung etwas erleichtern.

HINWEIS: *Designer unterstützt JavaScript, Version 1.6 und älter.*

Sie können die für neue Formulare verwendete Skriptsprache im Bereich „Arbeitsbereich“ des Dialogfelds „Optionen“ auswählen. Wählen Sie die Skriptsprache für das aktuelle Formular auf der Registerkarte „Standard“ des Dialogfelds „Formulareigenschaften“ aus.

Die in der Liste „Sprache“ im Skript-Editor angezeigte Skriptsprache stimmt mit der Skriptsprache überein, die Sie als Standardsprache für neue Formulare ausgewählt haben. Wenn Sie jedoch die Einstellung für die Skriptsprache für das aktuelle Formular ändern, wird diese Änderung in die Liste „Sprache“ übernommen und für alle neuen Skripten in Verbindung mit neuen Ereignissen verwendet. Die Änderung der Option für die Skriptsprache im Dialogfeld „Formulareigenschaften“ hat keine Änderung der Skriptsprache für vorhandene Skripten zur Folge. Wenn ein Ereignis bereits Skript enthält und dieses Skript gelöscht wird, verwendet der Skript-Editor für die Dauer der Designer-Sitzung weiterhin dieselbe Skriptsprache.

HINWEIS: *Ab dem 10. März 2012 stellt Adobe die Unterstützung der Leitfaden-Funktion in Adobe® LiveCycle® ES ein. Die Guides-Funktion steht dann nur noch im Rahmen von Produkt-Upgrades zur Verfügung und wird nach den nächsten zwei Hauptversionen vollständig entfernt.*

In der folgenden Tabelle sind einige der Hauptunterschiede zwischen FormCalc und JavaScript zusammengestellt.

FormCalc	JavaScript
Programmeigene Berechnungssprache von Adobe, gültig in Designer und Forms	Standard-Skriptsprache, kommt in zahlreichen verbreiteten Software-Anwendungen zum Einsatz
Kürzere Skripten (üblicherweise nur eine Zeile) Unterstützt Skriptschleifen	Möglichkeit, bei Bedarf längere Skripten mit Schleifenlogik zu verwenden
In Formularleitfäden nicht unterstützt (veraltet)	In Formularleitfäden unterstützt (veraltet)
Enthält eine Vielzahl von nützlichen integrierten Funktionen, durch die für allgemeine Aufgaben beim Formularentwurf weniger Skripterstellungsaufwand erforderlich ist	Bietet Zugriff auf das Acrobat-Objektmodell und die JavaScript-Funktionalität von Acrobat
Unterstützung für internationale Datums-, Uhrzeit-, Währungs- und Zahlenformate	Debugging mit dem JavaScript-Debugger in Acrobat möglich
Integrierte URL-Funktionen für `Post`, `Put` und `Get` ermöglichen webbasierte Interaktion	Erstellung benutzerdefinierter Funktionen für Ihre spezifischen Anforderungen
Kompatibel mit allen Plattformen, die von Designer und Forms unterstützt werden	Kompatibel mit allen Plattformen, die von Designer und Forms unterstützt werden

VERKNPFTE LINKS:

FormCalc verwenden

JavaScript verwenden

JavaScript-Funktionen erstellen und wiederverwenden

4.3. So erstellen Sie eine Berechnung oder ein Skript

1) Wählen Sie im Formularentwurf ein Objekt aus, das Ereignisse unterstützt. Fügen Sie einem neuen, leeren Formular eine Schaltfläche hinzu.

2) Wählen Sie im Skript-Editor aus der Liste „Anzeigen“ eines der Ereignisse aus, die für das betreffende Objekt gültig sind. Das gewählte Ereignis bestimmt, wann das Skript ausgeführt wird. Wenn Sie eine Berechnung oder ein Skript erstellen, die/das sich auf ein Objekt auswirkt, das keine Ereignisse unterstützt, müssen Sie Ihre Berechnung bzw. Ihr Skript einem Formularentwurfsobjekt hinzufügen, das Formularereignisse unterstützt. Wählen Sie für das neue Schaltflächenobjekt beispielsweise das `click` -Ereignis in der Liste „Anzeigen“.

3) Wählen Sie in der Liste „Sprache“ eine Skriptsprache aus. Wählen Sie für das neue Schaltflächenobjekt „JavaScript“ aus.

4) Wählen Sie in der Liste "Ausführen am", wo das Skript ausgeführt werden soll. Wählen Sie für das neue Schaltflächenobjekt beispielsweise "Client".

   Zur Auswahl stehen die clientbasierte Anwendung (z. B. Acrobat oder ein Webbrowser) und der serverbasierte Prozess.

   Wenn „Client“ festgelegt wurde, wird die Verarbeitung von Berechnungen und Skripten erst nach der Wiedergabe des Formulars ausgelöst. Wenn „Server“ festgelegt wurde, erfolgt die Verarbeitung von Berechnungen und Skripten bereits während des Wiedergabevorgangs. Durch die Anzeige des Formulars in der Vorschau mit Hilfe der Registerkarte „PDF-Vorschau“ wird das Öffnen des Formulars in Acrobat simuliert. Daher werden Skripten ausgeführt, für die die Ausführung auf dem Client bzw. auf dem Client und dem Server festgelegt wurde.

   HINWEIS: *Wenn Sie in der Liste „Ausführen am“ die Option „Client und Server“ wählen, wird das Skript entweder in der Client- oder in der Serveranwendung ausgeführt. Dies ist davon abhängig, welche Anwendung zur Verarbeitung des Formulars verwendet wird.*

5) Geben Sie in das Feld "Skriptquelle" eine FormCalc-Berechnung bzw. ein JavaScript-Skript ein. Mit Hilfe der Anweisungsende-Funktionalität von Designer können Sie Referenz-Syntaxen für eine Berechnung oder ein Skript erstellen. Fügen Sie beispielsweise das folgende JavaScript-Skript zum neuen Schaltflächenobjekt hinzu:

```
   xfa.host.messageBox("Hello World!", "Creating a new
script", 3);
```

6) Nach Fertigstellung des Formularentwurfs sollten Sie Ihre Berechnungen und Skripten vor dem eigentlichen Einsatz prüfen. Beispielsweise können Sie sich das neue Schaltflächenobjekt in der PDF-Version des Formulars auf der Registerkarte "PDF-Vorschau" ansehen. Klicken Sie auf das Schaltflächenobjekt, um die in Schritt 5 angegebene Meldung anzuzeigen.

   Weitere Informationen über die Designer-Objekte, die Skripterstellung unterstützen, finden Sie unter Objekte, die Berechnungen und Skripte unterstützen.

4.4. So erstellen Sie eine Berechnung oder ein Skript

1) Wählen Sie im Formularentwurf ein Objekt aus, das Ereignisse unterstützt. Fügen Sie einem neuen, leeren Formular eine Schaltfläche hinzu.

2) Wählen Sie im Skript-Editor aus der Liste „Anzeigen“ eines der Ereignisse aus, die für das betreffende Objekt gültig sind. Das gewählte Ereignis bestimmt, wann das Skript ausgeführt wird. Wenn Sie eine Berechnung oder ein Skript erstellen, die/das sich auf ein Objekt auswirkt, das keine Ereignisse unterstützt, müssen Sie Ihre Berechnung bzw. Ihr Skript einem Formularentwurfsobjekt hinzufügen, das Formularereignisse unterstützt. Wählen Sie für das neue Schaltflächenobjekt beispielsweise das `click` -Ereignis in der Liste „Anzeigen“.

3) Wählen Sie in der Liste „Sprache“ eine Skriptsprache aus. Wählen Sie für das neue Schaltflächenobjekt „JavaScript“ aus.

4) Wählen Sie in der Liste "Ausführen am", wo das Skript ausgeführt werden soll. Wählen Sie für das neue Schaltflächenobjekt beispielsweise "Client".

   Zur Auswahl stehen die clientbasierte Anwendung (z. B. Acrobat oder ein Webbrowser) und der serverbasierte Prozess (z. B. Adobe Document Services).

   Wenn „Client“ festgelegt wurde, wird die Verarbeitung von Berechnungen und Skripten erst nach der Wiedergabe des Formulars ausgelöst. Wenn „Server“ festgelegt wurde, erfolgt die Verarbeitung von Berechnungen und Skripten bereits während des Wiedergabevorgangs. Durch die Anzeige des Formulars in der Vorschau mit Hilfe der Registerkarte „PDF-Vorschau“ wird das Öffnen des Formulars in Acrobat simuliert. Daher werden Skripten ausgeführt, für die die Ausführung auf dem Client bzw. auf dem Client und dem Server festgelegt wurde.

   HINWEIS: *Wenn Sie in der Liste „Ausführen am“ die Option „Client und Server“ wählen, wird das Skript entweder in der Client- oder in der Serveranwendung ausgeführt. Dies ist davon abhängig, welche Anwendung zur Verarbeitung des Formulars verwendet wird.*

5) Geben Sie in das Feld "Skriptquelle" eine FormCalc-Berechnung bzw. ein JavaScript-Skript ein. Mit Hilfe der Anweisungsende-Funktionalität von Designer können Sie Referenz-Syntaxen für eine Berechnung oder ein Skript erstellen. Fügen Sie beispielsweise das folgende JavaScript-Skript zum neuen Schaltflächenobjekt hinzu:

```
   xfa.host.messageBox("Hello World!", "Creating a new
script", 3);
```

6) Nach Fertigstellung des Formularentwurfs sollten Sie Ihre Berechnungen und Skripten vor dem eigentlichen Einsatz prüfen. Beispielsweise können Sie sich das neue Schaltflächenobjekt in der PDF-Version des Formulars auf der Registerkarte "PDF-Vorschau" ansehen. Klicken Sie auf das Schaltflächenobjekt, um die in Schritt 5 angegebene Meldung anzuzeigen.

   Weitere Informationen über die Designer-Objekte, die Skripterstellung unterstützen, finden Sie unter Objekte, die Berechnungen und Skripte unterstützen.

VERKNPFTE LINKS:

FormCalc verwenden

JavaScript verwenden

Ereignisse

So erstellen Sie Berechnungen und Skripten mit dem Anweisungsende

Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll

So zeigen Sie Skriptereignisse und Skripten an

Festlegen, wo eine Berechnung oder ein Skript ausgeführt werden soll

Berechnungen und Skripten testen und debuggen

4.5. So suchen Sie nach Text oder anderen Objekten

Auf der Registerkarte "XML-Quelle" oder im Skript-Editor können Sie schnell nach jedem Exemplar eines Wortes oder Ausdrucks suchen.

1) Wählen Sie auf der Registerkarte „XML-Quelle“ oder im Skript-Editor den Befehl „Bearbeiten“ > „Suchen“ oder klicken Sie mit der rechten Maustaste, um das Kontextmenü aufzurufen.

2) Geben Sie im Feld „Suchen nach“ den zu suchenden Text ein.

3) Wählen Sie nach Wunsch andere Optionen aus.

4) Klicken Sie auf „Weitersuchen“.

   Zum Abbrechen eines laufenden Suchvorgangs drücken Sie die Esc-Taste oder klicken auf die Schaltfläche "Abbrechen".

   WICHTIG: *Sie können den XML-Quellcode zwar direkt auf der Registerkarte „XML-Quelle“ bearbeiten, jedoch wird empfohlen, nur dann Änderungen am Quellcode vorzunehmen, wenn Sie mit der Adobe-XML-Formulararchitektur vertraut sind. Weitere Informationen zur XML-Form-Architektur erhalten Sie im Developer Center.*

VERKNPFTE LINKS:

Berechnungen und Skripten erstellen
So erstellen Sie eine Berechnung oder ein Skript

So ersetzen Sie Text oder andere Objekte

So erstellen Sie Berechnungen und Skripten mit dem Anweisungsende

Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll

So zeigen Sie Skriptereignisse und Skripten an

Festlegen, wo eine Berechnung oder ein Skript ausgeführt werden soll

Berechnungen und Skripten testen und debuggen

4.6. So ersetzen Sie Text oder andere Objekte

Text kann automatisch ersetzt werden. Sie können beispielsweise *Corp.* durch *Corporation* ersetzen.

1) Wählen Sie im Skript-Editor „Bearbeiten“ > „Ersetzen“.
2) Geben Sie im Feld „Suchen nach“ den zu suchenden Text ein.
3) Geben Sie im Feld „Ersetzen durch“ den Ersatztext ein.
4) Wählen Sie nach Wunsch andere Optionen aus.
5) Klicken Sie auf „Weitersuchen“, „Ersetzen“ oder „Alle ersetzen“.
6) Zum Abbrechen eines laufenden Suchvorgangs drücken Sie die Esc-Taste oder klicken auf die Schaltfläche "Abbrechen".

   Zum Ersetzen von Text in Skripten, die an mehrere Objekte im Formular angehängt sind, wählen Sie das Stammteilformular Ihres Formulars (Standard: `form1`) und anschließend „Ereignisse für untergeordnete Objekte anzeigen“ aus. Führen Sie dann den oben beschriebenen Vorgang durch.

   WICHTIG: *Sie können den XML-Quellcode zwar direkt auf der Registerkarte „XML-Quelle“ bearbeiten, jedoch wird empfohlen, nur dann Änderungen am Quellcode vorzunehmen, wenn Sie mit der Adobe-XML-Formulararchitektur vertraut sind. Weitere Informationen zur XML-Form-Architektur erhalten Sie im Developer Center.*

VERKNPFTE LINKS:

Berechnungen und Skripten erstellen

So suchen Sie nach Text oder anderen Objekten

4.7. So erstellen Sie Berechnungen und Skripten mit dem Anweisungsende

Die Anweisungsende-Funktion im Skript-Editor gibt Ihnen die Möglichkeit, Ihre Berechnungen und Skripten interaktiv zu erstellen.

Wenn Sie eine Berechnung oder ein Skript erstellen, zeigt die Anweisungsende-Funktion jedes Mal, wenn Sie einen Punkt (.) unmittelbar hinter dem Namen eines Formularobjekts oder einer Formulareigenschaft eingeben, eine Liste der verfügbaren Methoden und Eigenschaften an. Falls die Anweisungsende-Liste nicht angezeigt wird, prüfen Sie, ob Sie den Objekt- oder Eigenschaftennamen richtig eingegeben haben und ob sich das Objekt innerhalb des Objektbereichs befindet, in dem Sie das Skript erstellen. Weitere Informationen zum Referenzieren von Objekten bei Berechnungen und Skripten finden Sie unter Objekte in Berechnungen und Skripten referenzieren.

1) Geben Sie den Namen eines Formularentwurfsobjekts oder einer Eigenschaft bzw. einen gültigen FormCalc-Kurzbefehl ein und gleich darauf einen Punkt.

2) Wählen Sie die Methode oder Eigenschaft aus, die Sie für das Formularentwurfsobjekt anwenden möchten, und fahren Sie dann mit dem Erstellen des Skripts fort. Um die Anweisungsende-Liste zu schließen, ohne eine Funktion auszuwählen, drücken Sie die Esc-Taste.

   Die Liste der verfügbaren Eigenschaften des XML Form Object Model ändert sich in Abhängigkeit von dem Formularentwurfsobjekt bzw. der Eigenschaft unmittelbar vor dem Punkt.

   HINWEIS: *Die Anweisungsende-Liste wird nur angezeigt, wenn auf Objekte, Eigenschaften und Methoden im XML Form Object Model zugegriffen wird. Bei Standard-JavaScript-Objekten oder -Methoden wird sie nicht angezeigt.*

VERKNPFTE LINKS:

Objekteigenschaften und -werte referenzieren

4.8. So fügen Sie eine Objektreferenz-Syntax automatisch ein

Anstatt eine Objektreferenz-Syntax manuell mit der Anweisungsende-Liste zu erstellen, können Sie mit der Funktion zum Einfügen einer Objektreferenz-Syntax Ihren Berechnungen oder Skripten auch automatisch eine Referenz-Syntax hinzufügen. Diese Funktion gibt für das Objekt, das Sie im Zeichnungsbereich für das Feld „Skriptquelle“ des Skript-Editors ausgewählt haben, eine abgekürzte Referenz-Syntax ein. Berechnungen und Skripten lassen sich auf diese Weise schneller erstellen und es ist gewährleistet, dass die Referenz-Syntax fehlerfrei ist.

1) Stellen Sie sicher, dass das Feld „Skriptquelle“ im Skript-Editor aktiv ist und dass der Cursor dort positioniert ist, wo Sie die Objektreferenz einfügen wollen.

2) Klicken Sie im Formular bei gedrückter Strg-Taste auf das zu referenzierende Objekt. Der Cursor nimmt die Form an, um Ihnen die Objektauswahl zu erleichtern.

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

4.9. Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll

Beim Erstellen von Berechnungen und Skripten müssen Sie jeden Eintrag einem bestimmten Formularereignis zuweisen. Jedes Formularereignis stellt eine Änderung des Formularstatus dar, die zu einem bestimmten Zeitpunkt eintritt.

Der Formularstatus kann sich während der Formularwiedergabe auf dem Server durch Forms, während der Formularwiedergabe auf dem Client durch Acrobat oder Adobe Reader sowie während des Ausfüllens des Formulars durch einen Benutzer ändern.

Wenn eine Änderung des Formularstatus eintritt, werden alle mit dem Ereignis verbundenen Berechnungen oder Skripten automatisch verarbeitet.

Das Ereignis, das Sie beim Erstellen einer Berechnung oder eines Skriptes verwenden, bestimmt in gewissem Maß, was Sie bei der Berechnung oder im Skript berücksichtigen müssen. Beispielsweise kann die auf einem Formular verfügbare Datenmenge und -art je nach dem gewählten Zeitpunkt des Ereignisses anders ausfallen. Die Ergebnisse einer Berechnung oder eines Skriptes, die bzw. das den Wert eines Feldes abruft, fallen daher möglicherweise unterschiedlich aus, je nachdem, ob sie bzw. es ausgeführt wird, bevor oder nachdem ein Benutzer beim Ausfüllen des Formulars bestimmte Aktionen ausführt. Weitere Informationen zu Ereignissen finden Sie unter Ereignisse.

Je nach dem erstellten Formulartyp treten einige Ereignisse möglicherweise niemals ein. Angenommen, ein Formular enthält ein festes Layout und keine interaktiven Objekte. In diesem Fall treten interaktive Ereignisse, die mit Benutzeraktionen verknüpft sind, wahrscheinlich nie ein und die mit diesen Ereignissen verbundenen Skripten werden folglich nicht ausgeführt.

Designer unterstützt eine Vielzahl von Formularereignissen. Zahlreiche gängige Berechnungs- und Skriptaufgaben lassen sich aber auch mit Hilfe einiger weniger Ereignisse durchführen, die bei wichtigen Änderungen des Formularstatus auftreten. Dazu gehören unter anderem:

docReady

Wird sofort nach dem Öffnen des Formulars in Acrobat oder Adobe Reader® ausgelöst sowie unmittelbar bevor der Benutzer mit Formularobjekten arbeiten kann. Dieses Ereignis wird als letztes ausgelöst, bevor der Benutzer die Steuerung des Formulars übernimmt.

enter

Wird ausgelöst, wenn ein Benutzer beim Ausfüllen den Fokus auf ein bestimmtes Feld, auf eine Schaltfläche oder auf ein Teilformular verlagert.

exit

Wird ausgelöst, wenn ein Benutzer beim Ausfüllen den Fokus von einem bestimmten Feld, einer Schaltfläche oder einem Teilformular auf ein anderes Objekt verlagert.

change

Wird ausgelöst, wenn ein Benutzer beim Ausfüllen einen Feldwert ändert. Dieses Ereignis wird am häufigsten bei Dropdown-Listen oder Listenfelder verwendet; wenn ein Benutzer den aktuellen Wert ändert, wird ein Skript ausgeführt.

click

Wird ausgelöst, wenn ein Benutzer beim Ausfüllen auf ein Feld oder eine Schaltfläche klickt. Dieses Ereignis wird häufig mit Schaltflächen verwendet, um ein Skript auszuführen, wenn der Benutzer beim Ausfüllen des Formulars auf die Schaltfläche klickt.

VERKNPFTE LINKS:

So zeigen Sie Skriptereignisse und Skripten an

Ereignisse

Festlegen, wo eine Berechnung oder ein Skript ausgeführt werden soll

4.10. So zeigen Sie Skriptereignisse und Skripten an

Im Skript-Editor gibt es verschiedene Möglichkeiten zur Anzeige der Skriptereignisse für Objekte in einem Formular. Welche Möglichkeiten verfügbar sind, hängt von der Art der ausgewählten Objekte und der Anzahl der anzuzeigenden Ereignisse ab.

Führen Sie zuerst die folgenden Schritte durch:

- Wenn der Skript-Editor nicht auf dem Bildschirm angezeigt wird, wählen Sie „Fenster“ > „Skript-Editor“.
- Wenn der Skript-Editor zum Anzeigen von mehr als einer Zeile eines Skripts nicht ausreicht, ziehen Sie die untere Linie des Skript-Editors nach unten, um das Anzeigefenster zu vergrößern.

4.10.1. So zeigen Sie im Skript-Editor ein Skriptereignis für ein einzelnes Objekt an

1) Wählen Sie im Formular ein Objekt aus.
2) Wählen Sie in der Liste „Anzeigen“ ein gültiges Skriptereignis aus.

4.10.2. So zeigen Sie im Skript-Editor ein Skriptereignis für ein Container-Objekt und dessen untergeordnete Objekte an

1) Ändern Sie bei Bedarf die Größe des Skript-Editors, damit mehrere Skriptzeilen angezeigt werden können, und vergewissern Sie sich, dass die Option „Ereignisse für untergeordnete Objekte anzeigen“ ausgewählt ist.
2) Wählen Sie ein Container-Objekt, wie z. B. ein Teilformular, aus.

3) Wählen Sie in der Liste „Anzeigen“ ein gültiges Skriptereignis aus.

   Die Ereignisse werden im Skript-Bearbeitungsfeld des Skript-Editors jeweils durch die Referenz-Syntax der einzelnen Ereignisse getrennt angezeigt. Bestimmte Ereignisse sind nur für bestimmte Objekttypen vorgesehen. Wenn Sie ein Skriptereignis auswählen, werden im Skript-Bearbeitungsfeld des Skript-Editors nur die gültigen Instanzen des Ereignisses aufgeführt. Angenommen, Sie wählen ein Teilformular aus, das eine Dropdown-Liste enthält, und das `preOpen` -Ereignis wählen, zeigt der Skript-Editor einen Eintrag für die Dropdown-Liste an. Dies liegt daran, dass das `preOpen` -Ereignis ist nur für Dropdown-Listen gilt. Alternativ dazu zeigt das `enter` -Ereignis zwei Einträge an, d. h. einen Eintrag für die Dropdown-Liste und einen zweiten Eintrag für das Teilformular.

   HINWEIS: *In der Liste „Anzeigen“ steht hinter den Namen von Ereignissen, die Skripten enthalten, jeweils ein Sternchen (*). Wenn ein Ereignis ein Skript enthält und Sie dieses Ereignis in der Liste „Anzeigen“ auswählen, wird die Quelle im Skript-Bearbeitungsfeld des Skript-Editors angezeigt.*

4.10.3. So zeigen Sie im Skript-Editor alle Skriptereignisse für ein einzelnes Objekt an

1) Wählen Sie im Formular ein Objekt aus.

2) Wählen Sie in der Liste „Anzeigen“ die Option „Alle Ereignisse“ aus.

   Die Ereignisse werden im Skript-Bearbeitungsfeld des Skript-Editors jeweils durch die Referenz-Syntax der einzelnen Ereignisse getrennt angezeigt.

4.10.4. So zeigen Sie im Skript-Editor alle Skriptereignisse für ein Container-Objekt und dessen untergeordnete Objekte an

1) Ändern Sie bei Bedarf die Größe des Skript-Editors, damit mehrere Skriptzeilen angezeigt werden können, und vergewissern Sie sich, dass die Option „Ereignisse für untergeordnete Objekte anzeigen“ ausgewählt ist.

2) Wählen Sie ein Container-Objekt, wie z. B. ein Teilformular, aus.

3) Wählen Sie in der Liste „Anzeigen“ die Option „Alle Ereignisse“ aus.

   Die Ereignisse werden im Skript-Bearbeitungsfeld des Skript-Editors jeweils durch die Referenz-Syntax der einzelnen Ereignisse getrennt angezeigt.

4.10.5. So zeigen Sie im Skript-Editor alle Skripten für ein einzelnes Objekt an

1) Wählen Sie ein Objekt, an das Skripten angehängt sind.
2) Wählen Sie in der Liste "Anzeigen" die Option "Ereignisse mit Skripten" aus.

   Die Skripten werden im Skript-Bearbeitungsfeld des Skript-Editors jeweils durch die Referenz-Syntax der einzelnen Ereignisse getrennt angezeigt.

4.10.6. So zeigen Sie im Skript-Editor alle Skripten für ein Container-Objekt und dessen untergeordnete Objekte an

1) Ändern Sie bei Bedarf die Größe des Skript-Editors, damit mehrere Skriptzeilen angezeigt werden können, und vergewissern Sie sich, dass die Option „Ereignisse für untergeordnete Objekte anzeigen“ ausgewählt ist.
2) Wählen Sie ein Container-Objekt, wie z. B. ein Teilformular, aus. Alle Ereignisse für das Container-Objekt und dessen untergeordnete Objekte werden im Skript-Editor angezeigt.
3) Wählen Sie in der Liste „Anzeigen“ die Option „Alle Ereignisse“ aus.

   Die Skripten werden im Skript-Bearbeitungsfeld des Skript-Editors jeweils durch die Referenz-Syntax der einzelnen Ereignisse getrennt angezeigt.

4.11. Festlegen, wo eine Berechnung oder ein Skript ausgeführt werden soll

Für jede in Designer erstellte Berechnung und jedes Skript müssen Sie die Position angeben, an der die Berechnung bzw. das Skript ausgeführt werden soll.

Sofern Sie keine serverbasierte Verarbeitung wie Forms verwenden, müssen Sie sicherstellen, dass alle Ihre Berechnungen und Skripten für die Ausführung auf der Client-Anwendung (z. B. auf Acrobat, einem Webbrowser oder der Mobile Workspace-App) ausgelegt sind.

HINWEIS: *FormCalc-Berechnungen und -Skripten sind nicht auf HTML-Formulare anwendbar und werden bei der Formularausfüllung übergangen.*

Bei Verwendung einer serverbasierten Verarbeitung können Sie wählen, ob Berechnungen in der Client-Anwendung oder auf dem Server ausgeführt werden sollen. Wenn Sie sich entscheiden, Berechnungen und Skripten auf dem Server ausführen zu lassen, legen Sie damit fest, dass die Skripten zu einem bestimmten Zeitpunkt während des Formularwiedergabeprozesses ausgeführt werden.

Weitere Informationen unter Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll.

Wenn Sie in der Liste "Ausführen am" die Option "Client und Server" wählen, steht die Berechnung bzw. das Skript sowohl client- als auch serverbasierten Anwendungen zur Verfügung. Diese Option ist beispielsweise dann sinnvoll, wenn Sie nicht wissen, ob den Benutzern, die Ihr Formular nutzen möchten, Client- oder Serveranwendungen zur Verfügung stehen. Die Option ist auch dann von Nutzen, wenn sich bestimmte Formularobjekte gegenüber einer Client-Anwendung und einer serverbasierten Anwendung unterschiedlich verhalten.

VERKNPFTE LINKS:

Ereignisse

Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll

So zeigen Sie Skriptereignisse und Skripten an

4.12. Berechnungen und Skripten testen und debuggen

Nach dem Erstellen von Berechnungen oder Skripten und dem Testen Ihres Formulars können Skriptfehler oder andere unerwartete Feldwerte aufgrund von Skriptfehlern auftreten.

In Designer gibt es drei grundlegende Methoden zum Testen und Debugging Ihrer Berechnungen und Skripten:

- Mit Hilfe der Paletten im Arbeitsbereich von „Paletten“. Weitere Informationen unter Berechnungen und Skripten mit dem Arbeitsbereich debuggen .
- Testen Ihrer Skripten mit Hilfe von JavaScript-Debugger (nur für JavaScript). Weitere Informationen zur Verwendung des Debuggers finden Sie unter JavaScript-Debugging.
- Verwenden der Hostmodell- und Ereignismodell-Eigenschaften und -Methoden zur Behebung der Fehler im Formular.

Mit Hilfe der Hostmodell- und Ereignismodell-Funktionalität können Sie entweder mit der Host-Anwendung oder mit den verschiedenen Formularereignissen arbeiten. Diese Modelle können hilfreiche Informationen zum Debugging von Berechnungen und Skripten zurückgeben.

Das folgende Skript gibt z. B. zur Laufzeit eine Meldung zurück, die den Namen des Ereignisses angibt, auf dem das Skript platziert wurde. Das bedeutet, dass ein bestimmtes Ereignis ausgelöst wurde:

```
    xfa.host.messageBox(xfa.event.name) // FormCalc
xfa.host.messageBox(xfa.event.name); // JavaScript
```

Ein weiteres Beispiel für den Einsatz der Hostmodell- und Ereignismodell-Methoden ist das Abrufen des Wertes eines Feldes in einem interaktiven Formular, bevor es vom Benutzer manuell bearbeitet wird. Dadurch lässt sich feststellen, wie die Objekte in Ihrem Formularentwurf auf die vom Benutzer eingegebenen Daten reagieren:

```
    xfa.host.messageBox(xfa.event.prevText) // FormCalc
xfa.host.messageBox(xfa.event.prevText); // JavaScript
```

VERKNPFTE LINKS:

Berechnungen und Skripten mit dem Arbeitsbereich debuggen

Mit Host-Anwendungen arbeiten

Mit dem Ereignismodell arbeiten

4.13. So prüfen Sie die Skriptsyntax

Bei der Arbeit an einem Formularentwurf können Sie alle JavaScript- oder FormCalc-Skripten auf eventuelle Syntaxfehler prüfen, um vor der Formularverteilung sicherzustellen, dass alle Formularfunktionen einwandfrei funktionieren. Im Formular gefundene Skriptsyntaxfehler werden auf der Registerkarte "Warnungen" der Palette "Bericht" angezeigt. Auf der Registerkarte "Warnungen" der Palette "Bericht" wird jeder Fehler in einer separaten, nummerierten Zeile mitsamt dem Ereignis oder dem Objektnamen und einer entsprechenden Beschreibung angegeben. Bei mehreren Ereignissen beginnt die Zeilennummerierung für jedes Ereignis mit 1.

Durch Klicken auf einen Skriptfehler in der Liste wird das entsprechende Skript mit einer Markierung der fehlerhaften Zeile und einer Einfügemarke am Zeilenanfang angezeigt. Skriptsyntaxfehler werden auch beim Speichern eines Formularentwurfs oder beim Verwenden der PDF-Vorschau auf der Registerkarte "Warnungen" angegeben.

HINWEIS: *Sie können das gewünschte Ereignis auch unter Verwendung des Dialogfelds "Gehe zu Zeile" anzeigen. Die Dropdown-Liste der Skriptereignisse enthält den Ausdruck „SOM“ (System Object Model), wie in den Header-Zeilen angezeigt, für jedes aktuell im Skript-Editor sichtbare Ereignis.*

1) Wählen Sie im Skript-Editor „Extras“ > „Skriptsyntax prüfen“.

VERKNPFTE LINKS:

FormCalc verwenden

JavaScript verwenden

Objekte, die Berechnungen und Skripte unterstützen

4.14. Sicherheitseinschränkungen umgehen

Werden das sourceSet-Modell oder dessen untergeordnete Elemente von Skripten verändert, führt dies zu einer ungültigen, nicht länger vertrauenswürdigen Formularzertifizierung. Da ein Formular jederzeit zertifiziert werden kann, sollten unbedingt Skripttechniken verwendet werden, die nicht zur Ungültigkeit von Zertifikaten führen.

Bei Verwendung von Skripten, mit denen Änderungen am sourceSet-Modell oder dessen untergeordneten Elementen vorgenommen werden, sollten Sie mit Modellklonen anstelle von Originalmodellen arbeiten. Dadurch werden ungültige Formulare bei Änderung eines Datenmodells verhindert. Bei Formularen zum Ausführen allgemeiner Aufgaben wie die Anzeige von Datenbankeinträgen ist die Änderung der Datenverbindungs-Nodes im sourceSet-Modell erforderlich.

Zum Klonen des sourceSet-Modells müssen Sie eine Methode für das Skript erstellen, mit der die im sourceSet-Modell zu ändernde Datenverbindung bestimmt wird. Stellen Sie zudem sicher, dass das Skript ausschließlich den Klon anstelle der Originaldefinition verwendet.

Nachfolgend wird ein Skript einer Daten-Dropdown-Liste angezeigt. Das Skript füllt die Liste mit Daten einer Datenquelle.

```
...
var oDB = xfa.sourceSet.nodes.item(nIndex);
...
// Suchknoten mit dem Klassennamen "command"
var nDBIndex = 0;
while(oDB.nodes.item(nDBIndex).className != "command")
nDBIndex++;

oDB.nodes.item(nDBIndex).query.recordSet.setAttribute("stayBOF", "bofAction");
oDB.nodes.item(nDBIndex).query.recordSet.setAttribute("stayEOF", "eofAction");
```

Zum Klonen des sourceSet-Modells müssen Sie die Zeile zum Zugriff auf das Modell durch Anfügen der clone (1)-Methode am Ende der Anweisung entsprechend ändern:

```
var oDB = xfa.sourceSet.nodes.item(nIndex).clone(1);
```

HINWEIS: *Sie können die geklonte Datenverbindungs-Node in einer Variablen oder in einem variablendefinierten Skriptobjekt speichern.*

5. Ereignisse

Alle Berechnungen und Skripten, die einem Formularobjekt hinzugefügt werden, sind jeweils einem bestimmten Ereignis zugeordnet. Unter einem *Ereignis* versteht man ein bestimmtes Vorkommnis oder eine Aktion, das bzw. die den Zustand eines Formulars ändern kann. Wenn eine Änderung des Zustands eintritt, wird automatisch eine mit dem Ereignis verbundene Berechnung oder ein Skript aufgerufen. Ereignisse können zu beliebigen Zeitpunkten auftreten – vom Beginn des Formularwiedergabeprozesses, wenn Daten mit einem Formularentwurf zusammengeführt werden, bis hin zum Ausfüllen des Formulars, wenn ein Benutzer die Formularobjekte in einer Client-Anwendung verwendet. Durch die Verknüpfung von Berechnungen und Skripten mit bestimmten Ereignissen können Sie präzise steuern, wie Formularobjekte und -daten dargestellt werden und wie die Objekte und Daten auf die Benutzerinteraktionen beim Ausfüllen des Formulars reagieren.

Eine einzelne Zustandsänderung oder Aktion beim Ausfüllen des Formulars kann mehrere Ereignisse auslösen. Wenn Sie beispielsweise mit der Tabulatortaste vom aktuellen Feld zum nächsten Feld navigieren, wird das `exit` Ereignis für das aktuelle Feld und das `enter` -Ereignis für das nächste Feld ausgelöst. Wenn sich das aktuelle und das nächste Feld in unterschiedlichen Teilformularen befinden, werden insgesamt vier Ereignisse ausgelöst, und zwar das `exit` -Ereignisse für das aktuelle Feld und das Teilformular und das `enter` -Ereignis für das nächste Feld und Teilformular. In der Regel gilt für alle Kategorien der Formularereignisse eine vorhersehbare Reihenfolge.

5.1. Ereignistypen

Die Formularereignisse werden in vier Kategorien unterteilt:

Prozessereignisse

Dieser Ereignistyp wird durch einen internen Prozess bzw. eine interne Aktion im Zusammenhang mit Objekten in einem Formular automatisch ausgelöst. Angenommen, ein Benutzer klickt beim Ausfüllen des Formulars auf eine Schaltfläche, durch die dem Formular eine neue Seite hinzugefügt wird, werden die `initialize`, `calculate`, `validate` und `layout:ready` -Prozessereignisse automatisch für die neue Seite initiiert.

Interaktive Ereignisse

Dieser Ereignistyp wird direkt durch Aktionen beim Ausfüllen des Formulars ausgelöst. Wenn ein Benutzer z. B. den Zeiger über ein Feld in einem Formular bewegt, wird `mouseEnter` als Antwort auf die Aktiion initiiert.

Anwendungsereignisse

Dieser Ereignistyp wird durch die Aktionen ausgelöst, die eine Client-Anwendung oder eine Serveranwendung ausführt. Beispiel: Sie können eine Berechnung oder ein Skript erstellen, um unmittelbar nach dem Speichern des Formulars eine Aufgabe auszuführen. Verwenden Sie hierzu das `postPrint` -Ereignis.

VERKNPFTE LINKS:

Prozessereignisse

Interaktive Ereignisse

Anwendungsereignisse

5.2. Prozessereignisse

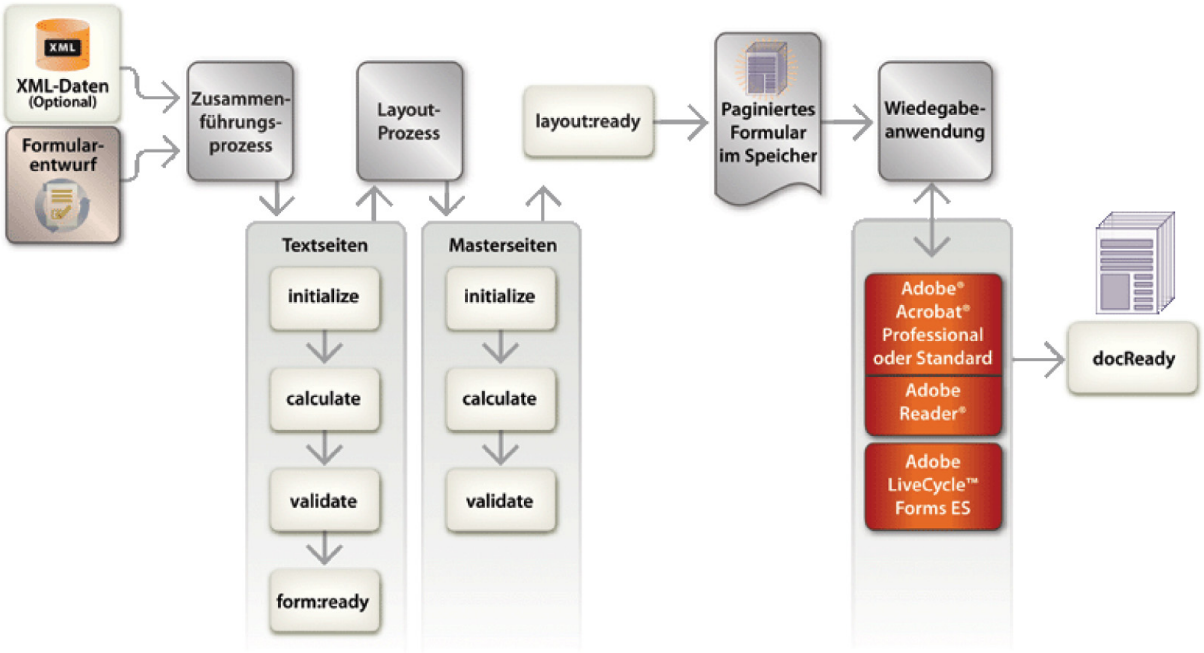
Prozessereignisse werden durch einen internen Prozess bzw. eine interne Aktion im Zusammenhang mit einem Formular oder Objekten in einem Formular automatisch ausgelöst. Sie werden unmittelbar nach bedeutsamen Formularänderungen initiiert, z. B. nach dem Zusammenführen eines Formularentwurfs mit Daten oder nach dem Abschluss der Formularpaginierung. Prozessereignisse werden außerdem unmittelbar nach dem Initiieren interaktiver Ereignisse ausgelöst. Beispielsweise wird direkt nach dem Initiieren eines beliebigen interaktiven Ereignisses das `calculate` Ereignis initiiert, gefolgt vom `validate` -Ereignis.

Die folgenden Prozessereignisse sind im Skript-Editor unter „Anzeigen“ verfügbar:

- `calculate`
- `form:ready`
- `indexChange`
- `initialize`
- `layout:ready`
- `validate`

  Prozessereignisse können aufgrund von Abhängigkeiten viele Male initiiert werden. Unter Abhängigkeiten versteht man Aktionen, die einem einzelnen Ereignis zugeordnet sind, das letztlich ein oder mehrere zusätzliche Ereignisse auslöst. Angenommen, ein Benutzer klickt beim Ausfüllen eines Formulars auf eine Schaltfläche, um einen zuvor ausgeblendeten Bereich des Formulars einzublenden. Durch Klicken auf die Schaltfläche wird nicht nur eine Reihe interaktiver und Prozessereignisse für die Schaltfläche selbst ausgelöst, sondern auch eine Reihe von Prozessereignissen für das neue Teilformular.

  In der folgenden Abbildung wird die normale Ereignisabfolge bis zum Öffnen eines PDF-Formulars in Acrobat oder Acrobat Reader veranschaulicht.



Nach dem Öffnen des Formulars in Acrobat oder Acrobat Reader werden diese Prozessereignisse unter Umständen aufgrund von Formularänderungen erneut ausgelöst. Beispielsweise werden die `calculate`, `validate` und `layout:ready` für ein Objekt unmittelbar nach dem Eintreten beliebiger interaktiver Ereignisse ausgelöst. Die mit den Prozessereignissen verknüpften Berechnungen und Skripten werden daher mehrmals ausgeführt.

VERKNPFTE LINKS:

Interaktive Ereignisse

Anwendungsereignisse

calculate-Ereignis

docReady-Ereignis

form:ready-Ereignis

indexChange-Ereignis

initialize-Ereignis

layout:ready-Ereignis

validate-Ereignis

Festlegen, wann eine Berechnung oder ein Skript ausgeführt werden soll

5.3. Interaktive Ereignisse

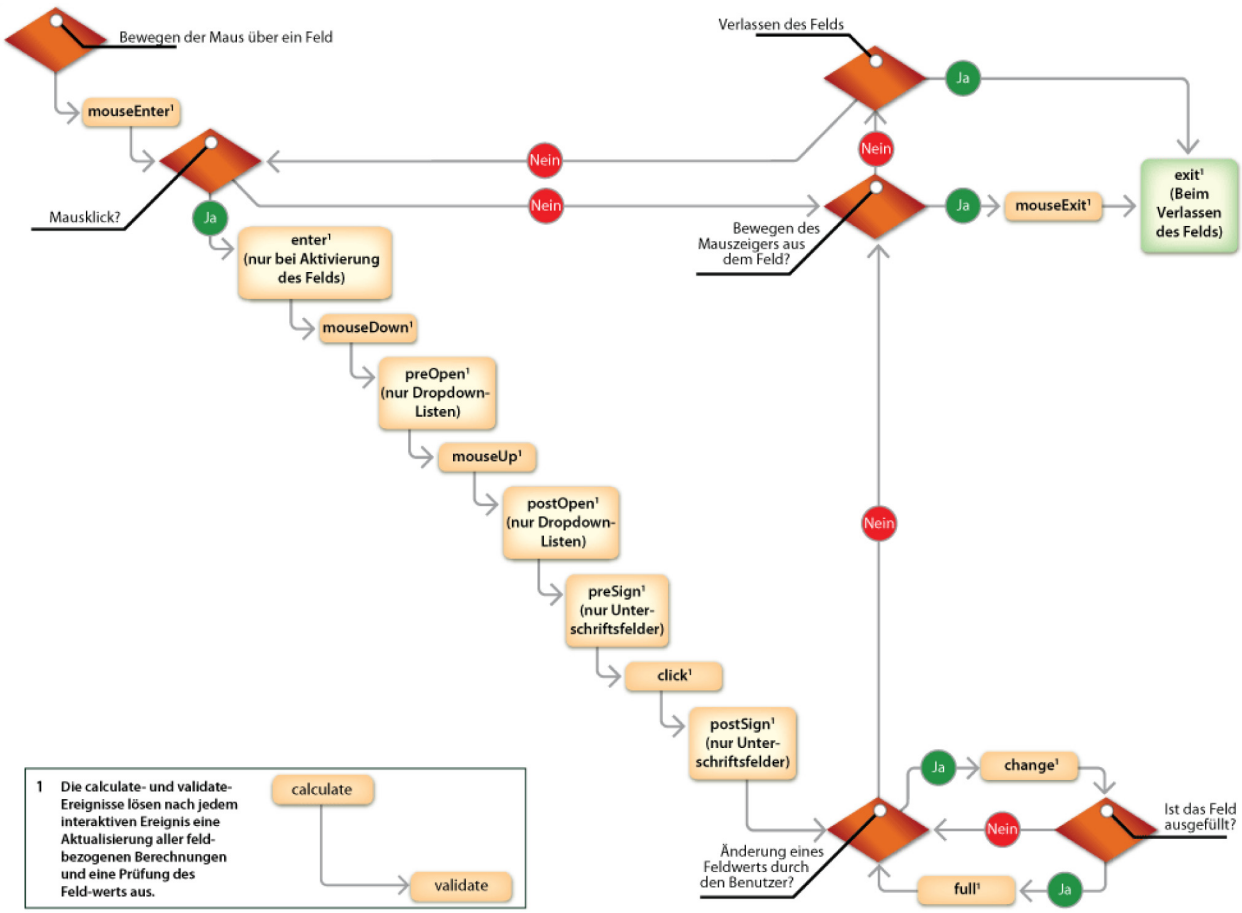
Interaktive Ereignisse werden direkt durch Benutzeraktionen beim Ausfüllen des Formulars ausgelöst. Daher sind sie für eine Vielzahl von Berechnungs- und Skriptaufgaben nützlich. So können Sie den zum Beispiel ein Skrip zum `mouseEnter` -Ereignis für ein Textfeld hinzufügen, das die Rahmenfarbe des Felds in Blau ändert, und ein Skript zum `mouseExit` -Ereignis, das die Rahmenfarbe wiederherstellt. Durch diese Aktion wird das Feld visuell hervorgehoben, wenn ein Benutzer beim Ausfüllen des Formulars den Zeiger über das Feld bewegt. Interaktive Ereignisse sind auch nützlich, wenn Formulardaten in Abhängigkeit von einer Benutzerauswahl geändert werden sollen. So können Sie den zum Beispiel ein Skrip zum `change` -Ereignis für eine Dropdown-Liste hinzufügen, das die Datenwerte in mehreren Feldern je nach dem Wert, der der Formularbenutzer in der Dropdown-Liste auswählt.

Die folgenden interaktiven Ereignisse sind im Skript-Editor unter „Anzeigen“ verfügbar:

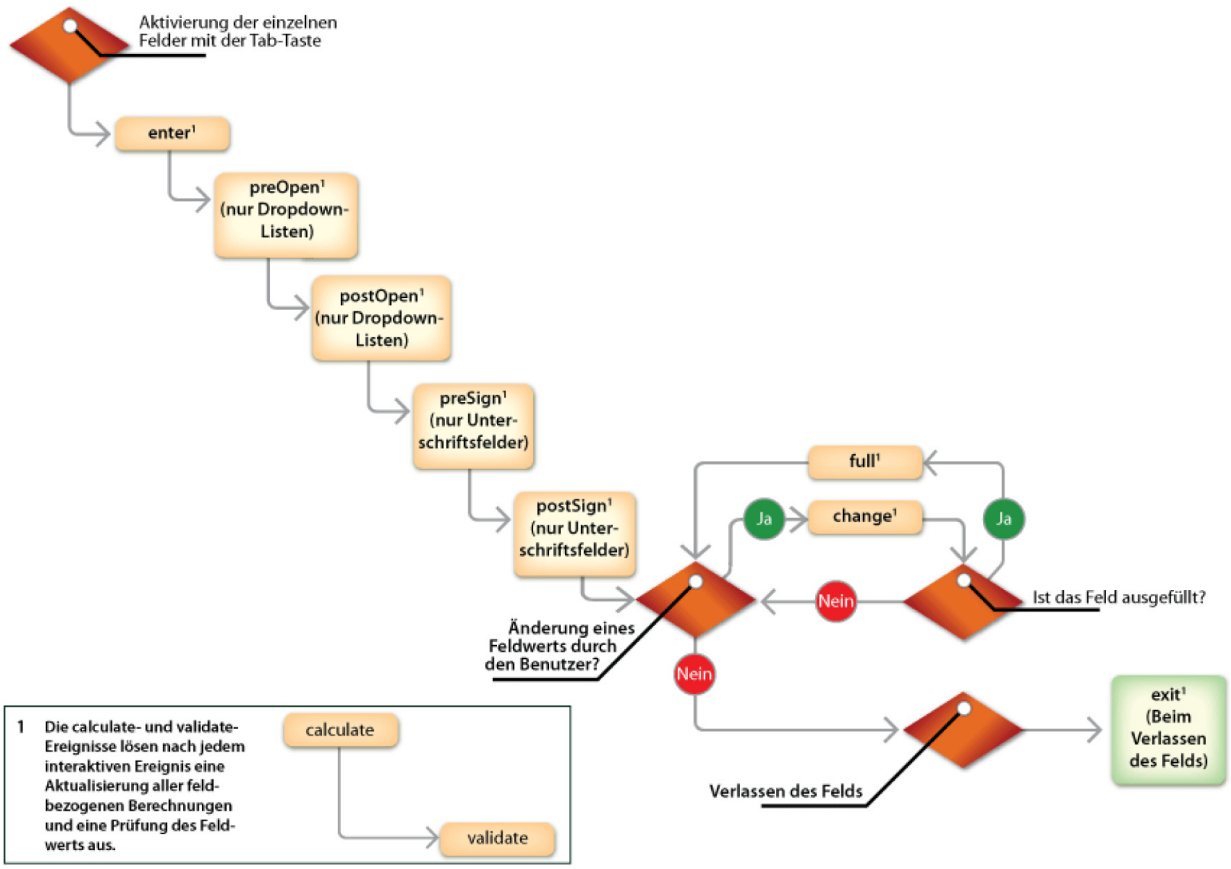
- `change`
- `click`
- `enter`
- `exit`
- `mouseDown`
- `mouseEnter`
- `mouseExit`
- `mouseUp`
- `postOpen`
- `postSign`
- `preOpen`
- `preSign`

In der folgenden Abbildung wird die normale Ereignisabfolge beim Ausfüllen von Formularen veranschaulicht, wenn der Benutzer mit der Maus ein Objekt auswählt und dessen Wert ändert.

HINWEIS: *Die Abbildung veranschaulicht eine normale Ereignisabfolge. Diese Abfolge kann aber von bestimmten Aktionen des Formularbenutzers und von bestimmten Formularobjekten geändert werden. Wenn beispielsweise ein Formularbenutzer in der Dropdown-Liste einen Wert auswählt, tritt das `mouseExit` -Ereignis tritt nach dem `click` -Ereignis auf, aber vor den `change` oder `full` Ereignissen. Wenn ein Benutzer ein Feld auswählt, die Maustaste drückt und dann das Feld schließt, während die Maustaste noch gedrückt wird, tritt das `mouseUp` -Ereignis anders auf, als in dieser Abbildung beschriebenen.*



In der folgenden Abbildung wird die normale Ereignisabfolge beim Ausfüllen von Formularen veranschaulicht, wenn der Benutzer über die Tastatur ein Objekt auswählt und dessen Wert ändert.



VERKNPFTE LINKS:

change-Ereignis

click-Ereignis

enter-Ereignis

exit-Ereignis

full-Ereignis

mouseDown-Ereignis

mouseEnter-Ereignis

mouseExit-Ereignis

mouseUp-Ereignis

postOpen-Ereignis

postSign-Ereignis

preOpen-Ereignis

preSign-Ereignis

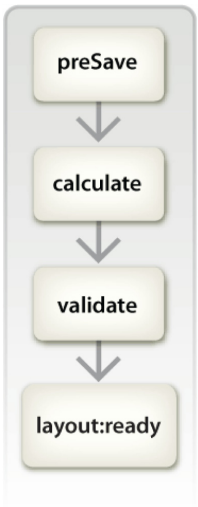
5.4. Anwendungsereignisse

Anwendungsereignisse werden durch Aktionen ausgelöst, die eine Client-Anwendung oder eine Serveranwendung infolge einer Benutzeraktion beim Ausfüllen des Formulars oder eines automatisierten Prozesses ausführt. Anwendungsereignisse kommen in normalen Ereignisabfolgen nicht vor. Sie sind vielmehr Einzelereignisse im Zusammenhang mit Aktionen, welche die Client- oder Serveranwendung ausführt.

Die folgenden Anwendungsereignisse sind im Skript-Editor unter „Anzeigen“ verfügbar:

- `docClose`
- `docReady`
- `postPrint`
- `postSave`
- `postSubmit`
- `prePrint`
- `preSave`
- `preSubmit`

  Beispielsweise wird in der folgenden Abbildung die normale Ereignisabfolge für das `preSave` -Ereignis veranschaulicht.



  Wenn ein Formularbenutzer das Formular in Acrobat oder Adobe Reader speichert, wird das `preSave` -Ereignis unmittelbar vor dem Speichern-Vorgang initiiert, gefolgt von den `calculate`, `validate` und `layout:ready` Ereignissen in dieser Reihenfolge für alle Objekte im Formular. Die gleiche Ereignisabfolge wird initiiert, wenn das Formular ein Skript enthält, welches das Formular programmatisch speichert.

  Eine ähnliche Ereignisabfolge gilt für alle anderen oben aufgeführten Anwendungsereignisse.

VERKNPFTE LINKS:

docClose-Ereignis

postPrint-Ereignis

postSave-Ereignis

postSubmit-Ereignis

prePrint-Ereignis

preSave-Ereignis

preSubmit-Ereignis

5.5. calculate-Ereignis

5.5.1. Beschreibung

Wird unter den folgenden Umständen initiiert:

- Wenn der Formularentwurf und die Daten zum fertigen Formular zusammengeführt werden.
- Wenn sich einer der Werte ändert, von denen die Berechnung abhängig ist, z. B. der Wert eines bestimmten Felds, es sei denn, der berechnete Wert wurde beim Ausfüllen des Formulars manuell vom Benutzer überschrieben. Als Ergebnis zeigt das Objekt den Rückgabewert des Ereignisses an. Die Eigenschaften für manuell überschriebene Felder befinden sich auf der Registerkarte „Wert“ der Palette „Objekt“.
- Wenn ein Feld den Fokus verliert, z. B. wenn ein Benutzer klickt oder die Tabulatortaste drückt, um ein Feld zu verlassen.

Bei der Verwendung des `calculate` -Ereignisses zum Durchführen von Berechnungen oder Skripte, sind folgende potenzielle Probleme zu beachten:

- Berechnungen und Skripte im `calculate` -Ereignis dürfen keine Änderungen an der Formularstruktur vornehmen; davon ausgenommen sind die Formularfeld- und Datenwerte.
- Eingefügter Inhalt durch das `calculate` -Ereignis müssen den zugehörigen Validierungen für das Objekt entsprechen; anderenfalls werden Validierungsfehler gemeldet.
- Berechnungen und Skripten dürfen keine Endlosschleifen enthalten, weil diese dazu führen, dass das Formular den Wert kontinuierlich aktualisiert. Beispielsweise könnte ein Skript, das den Wert eines Felds im Rahmen eines Schleifenausdrucks inkrementiert, z. B. eine `while` oder `for` Schleife, eine Endlosschleife erstellen.
- Der zuletzt evaluierte Ausdruck im `calculate` -Ereignis wird verwendet, um den Wert des aktuellen Formularobjekts auszufüllen. Wenn das `calculate` -Ereignis den Wert des aktuellen Felds zunächst auf 500 festlegt und anschließend den Wert eines weiteren Felds auf 1000 festlegt, zeigen beide Felder zur Laufzeit den Wert 1000 an. Daher müssen Sie die Skripte, die Sie dem `calculate` -Ereignis hinzufügen auf die Skripte beschränken, die sich speziell mit dem Festlegen des Werts des aktuellen Felds beschäftigen.

5.5.2. Tippen Sie

Prozessereignis

5.5.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.5.4. Version

XFA 2.1

5.5.5. Beispiel

Verwenden Sie das `calculate` -Ereignis zur Aktualisierung von Zahlenwerten in Feldern, weil es unmittelbar nach den meisten anderen Ereignissen initiiert wird. In einem Bestellformular können Sie beispielsweise das `calculate` -Ereignis für ein Feld verwenden, um den Prozentwert der zu bestimmen. Die Berechnung wird jedes Mal ausgeführt, wenn die Werte in den Formularfeldern geändert werden. Auf diese Weise wird gewährleistet, dass der für die Umsatzsteuer angezeigte Wert immer korrekt ist.

Da das `calculate` -Ereignis aber viele Male initiiert werden kann, müssen Sie sicherstellen, dass die Berechnung oder das Skript, die bzw. das Sie dem Ereignis hinzufügen, nicht zu einer unnötigen Inkrementierung von Datenwerten führt. Wenn z. B. im Rahmen der Umsatzsteuerberechnung der Wert der Umsatzsteuer den Gesamtkosten bei jedem `calculate` -Ereignis hinzugefügt wird, ist möglicherweise der resultierende Gesamtkostenwert im Formular zu groß.

Beispiele für die Verwendung des `calculate` -Ereignisses finden Sie unter Feldsummen berechnen.

VERKNPFTE LINKS:

Ereignisse

Prozessereignisse

5.6. change-Ereignis

5.6.1. Beschreibung

Wird ausgelöst, wenn ein Benutzer beim Ausfüllen des Formulars den Inhalt eines Feldes durch eine der folgenden Aktionen ändert:

- Tasteneingabe, während das Feld den Tastaturfokus besitzt
- Dateneingabe in das Feld
- Auswahl aus einem Listenfeld oder einer Dropdown-Liste
- Aktivieren oder Deaktivieren eines Kontrollkästchens
- Änderung der Einstellungen einer Optionsfeldgruppe

  Dieses Ereignis wird nicht ausgelöst, wenn die Objektwerte durch Berechnungen oder Skripten bzw. durch die Zusammenführung des Formularentwurfs mit Daten geändert werden.

5.6.2. Tippen Sie

Interaktives Ereignis

5.6.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes (Nur für Dropdown-Listen)

5.6.4. Version

XFA 2.1

5.6.5. Beispiel

Dieses Ereignis eignet sich für Berechnungen oder Skripten, die ausgelöst werden müssen, wenn ein Benutzer beim Ausfüllen des Formulars den Wert eines Feldes ändert. Beispielsweise können Sie das `change` Ereignis für eine Dropdown-Liste verwenden, um bestimmte Zeilen in einer Tabelle hervorzuheben. Jedes Mal, wenn der Benutzer einen Wert in der Dropdown-Liste auswählt, wird dann die entsprechende Tabellenzeile hervorgehoben.

HINWEIS: *Skripterstellung für ein Objekt `'this.rawValue'` funktioniert nicht. Verwenden Sie stattdessen die Ereignismodell-Eigenschaft `$event.fullText` des aktuellen Werts des Objekts.*

Beispiele für die Verwendung des `change` -Ereignisses finden Sie unter Aktuellen oder vorherigen Wert einer Dropdown-Liste abrufen.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.7. click-Ereignis

5.7.1. Beschreibung

Wird initiiert, wenn innerhalb des Bereichs ein Mausklick erfolgt. Wenn ein `click` -Ereignis für einen Text- oder ein numerisches Feld initiiert wird, werden Berechnungen oder Skripten sofort ausgeführt. Der Wert des Feldes wird aber erst dann aufgrund von Berechnungen und Skripten geändert, wenn das Feld den Fokus verliert.

HINWEIS: *Sie können keine Berechnung und kein Skript auf das `click` Ereignis der Senden-Schaltfläche platzieren, weil die Berechnung oder das Skript die Übergabeaktion außer Kraft setzt. Berechnungen und Skripte stattdessen im `preSubmit` -Ereignis einer Senden-Schaltfläche platzieren. Weitere Informationen zu Aktionen zum Senden von Formularen erhalten Sie in der Designer-Hilfe.*

5.7.2. Tippen Sie

Interaktives Ereignis

5.7.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.7.4. Version

XFA 2.1

5.7.5. Beispiel

Dieses Ereignis eignet sich für die Ausführung einer Aktion, nachdem ein Benutzer im Formular auf eine Schaltfläche geklickt oder ein Optionsfeld bzw. ein Kontrollkästchen aktiviert hat. Beispielsweise können Sie das `click` -Ereignis für ein Kontrollkästchen verwenden, um ein Formularfeld ein- und auszublenden.

Beispiele für die Verwendung des `click` -Ereignisses finden Sie unter Visuelle Eigenschaften von Objekten im Client ändern.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

preSubmit-Ereignis

5.8. docClose-Ereignis

5.8.1. Beschreibung

Wird ganz am Ende der Verarbeitung eines Formulars ausgeführt, sofern sämtliche Formularüberprüfungen fehlerfrei durchgeführt wurden.

5.8.2. Tippen Sie

Anwendungsereignis

5.8.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.8.4. Version

XFA 2.1

5.8.5. Beispiel

Dieses Ereignis wird erst spät ausgelöst und führt keine Änderungen am gespeicherten Formular durch. Es soll vielmehr die Möglichkeit bieten, einen Beenden-Status oder eine Fertig-Meldung zu erzeugen. Beispielsweise können Sie das `docClose` -Ereignis verwenden, um eine Meldung mit dem Hinweis einzublenden, dass das Formular fertig ausgefüllt ist.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

Prozessereignisse

5.9. docReady-Ereignis

5.9.1. Beschreibung

Wird sofort nach dem Öffnen des Formulars in Acrobat oder Adobe Reader ausgelöst.

5.9.2. Tippen Sie

Anwendungsereignis

5.9.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.9.4. Version

XFA 2.1

5.9.5. Beispiel

Dies ist das erste Ereignis, das nach dem Öffnen eines Formulars in Acrobat oder Adobe Reader ausgelöst wird. Dieses Ereignis sollte für alle Berechnungs- oder Skriptaufgaben verwendet werden, für welche das vollständige Formular gebraucht wird oder die nur einmal beim ersten Öffnen des Formulars ausgeführt werden sollen. Beispielsweise können Sie mit dem `docReady` -Ereignis die Version von Acrobat oder Adobe Reader prüfen und eine Meldung einblenden, wenn vor dem Ausfüllen des Formulars eine Anwendungsaktualisierung erforderlich ist.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.10. enter-Ereignis

5.10.1. Beschreibung

Wird ausgelöst, wenn ein Feld oder Teilformular den Tastaturfokus erhält, und zwar unabhängig davon, ob dies durch eine Benutzeraktion verursacht wird (Wechsel in ein Feld per Tabulatortaste oder Mausklick) oder durch ein Skript, welches den Fokus programmatisch setzt.

5.10.2. Tippen Sie

Interaktives Ereignis

5.10.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.10.4. Version

XFA 2.1

5.10.5. Beispiel

Mit diesem Ereignis können Sie Hilfetext oder andere Meldungen bereitstellen, welche dem Benutzer das Ausfüllen des aktuellen Feldes oder Teilformulars erleichtern. Angenommen, in ein Feld muss ein Wert in einem bestimmten Format eingegeben werden oder beim Ausfüllen eines Feldes sind spezifische Anweisungen zu beachten. In diesem Fall können Sie mit diesem Ereignis eine Meldung einblenden, welche den Benutzer über die besonderen Anforderungen informiert.

Beispiele für die Verwendung des `enter` -Ereignisses finden Sie unter Felder als Reaktion auf Benutzeraktionen hervorheben.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.11. exit-Ereignis

5.11.1. Beschreibung

Wird ausgelöst, wenn das Feld oder Teilformular den Tastaturfokus verliert, und zwar unabhängig davon, ob dies durch eine Benutzeraktion verursacht wird (Wechsel in ein anderes Feld per Tabulatortaste oder Mausklick außerhalb des Feldes) oder durch ein Skript, welches den Fokus programmatisch entfernt.

HINWEIS: *Wenn das Skript den Wert des aktuellen Felds bearbeiten soll, müssen Sie das Skript eventuell an das* `calculate` *-Ereignis anhängen.*

5.11.2. Tippen Sie

Interaktives Ereignis

5.11.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.11.4. Version

XFA 2.1

5.11.5. Beispiel

Dieses Ereignis eignet sich zur Überprüfung von Felddaten, wenn der Benutzer den Fokus von einem Feld entfernt. Angenommen, in ein Feld muss ein Wert eingegeben werden. In diesem Fall können Sie mit diesem Ereignis eine Meldung bereitstellen, welche den Benutzer darauf aufmerksam macht, dass das Formular nur gesendet werden kann, wenn in dieses Feld Daten eingegeben wurden.

Beispiele für die Verwendung des `exit` -Ereignisses finden Sie unter Felder als Reaktion auf Benutzeraktionen hervorheben.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.12. form:ready-Ereignis

5.12.1. Beschreibung

Wird initiiert, wenn Formularentwurf und Daten zusammengeführt wurden, das Formular im Speicher vorhanden ist und die `initialize`, `calculate` und `validate` -Ereignisse abgeschlossen sind.

HINWEIS: *Das form:ready-Ereignis ist nur für Objekte in der Designansicht und nicht für Objekte auf Masterseiten vorgesehen (siehe Prozessereignisse).*

5.12.2. Tippen Sie

Prozessereignis

5.12.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.12.4. Version

XFA 2.1

5.12.5. Beispiel

Dieses Ereignis eignet sich zur Ausführung von Aufgaben, nachdem Formularentwurf und Daten zusammengeführt wurden, aber bevor das Layout fertig ist. Beispielsweise können Sie mit diesem Ereignis die Anordnung oder Platzierung von Teilformularen im Formular anpassen, bevor das Formular paginiert und wiedergegeben wird.

VERKNPFTE LINKS:
Ereignisse

5.13. full-Ereignis

5.13.1. Beschreibung

Wird ausgelöst, wenn beim Ausfüllen von Formularen mehr als die maximal zulässige Menge an Inhalt in ein Feld eingegeben wird. Angenommen, die Eigenschaft für die Längenbegrenzung für ein Feld wurde auf `5` eingestellt und ein Benutzer versucht, die `abcdef` Zeichenfolge einzugeben, wird das `full` -Ereignis initiiert, sobald der Benutzer den Buchstaben `f` eingeben.

HINWEIS: *Die Eigenschaft „Länge begrenzen“ für ein Feld befindet sich auf der Registerkarte „Feld“ der Palette „Objekt“.*

5.13.2. Tippen Sie

Interaktives Ereignis

5.13.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.13.4. Version

XFA 2.1

5.13.5. Beispiel

Verwenden Sie dieses Ereignis, um einen Benutzer darauf hinzuweisen, dass die maximal zulässige Datenmenge in ein Feld eingegeben wurde. Beispielsweise können Sie eine Meldung mit dem Hinweis einblenden, dass das Feld voll ist, und Möglichkeiten zur Problemlösung vorschlagen.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.14. indexChange-Ereignis

5.14.1. Beschreibung

Wird ausgelöst, nachdem ein Teilformular durch Zusammenführen neuer Daten mit dem Formular oder mit Hilfe von Skripten eingefügt, verschoben oder aus dem Formular entfernt wurde.

Beachten Sie, dass das indexChange-Ereignis nicht ausgelöst wird, wenn die letzte Zeile einer Tabelle gelöscht wird.

HINWEIS: *Dieses Ereignis wird nur von den Teilformularinstanzen empfangen, die vom Instanzmanager verwaltet werden; bei Teilformularsätzen wird es ignoriert.*

5.14.2. Tippen Sie

Prozessereignis

5.14.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.14.4. Version

XFA 2.5

5.14.5. Beispiel

Mit diesem Ereignis können Sie anhand des Instanzwerts eines bestimmten Objekts Eigenschaften festlegen. Beispielsweise lässt sich damit die abwechselnde Zeilenschattierung in einer Tabelle steuern.

VERKNPFTE LINKS:

Ereignisse

Prozessereignisse

5.15. initialize-Ereignis

5.15.1. Beschreibung

Wird für alle Objekte initiiert, nachdem der Formularentwurf mit Daten zusammengeführt wurde.

5.15.2. Tippen Sie

Prozessereignis

5.15.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.15.4. Version

XFA 2.1

5.15.5. Beispiel

Mit diesem Ereignis können Sie beim Erstellen eines Objekts Aktionen ausführen, die entweder durch eine Benutzeraktion beim Ausfüllen des Formulars oder im Rahmen des Formularerstellungsprozesses verursacht werden. Beispielsweise können Sie die Einstellungen für neue Instanzen eines Teilformularobjekts steuern, welche der Benutzer dem Formular durch Klicken auf eine Schaltfläche hinzufügt.

VERKNPFTE LINKS:

Ereignisse

Prozessereignisse

5.16. layout:ready-Ereignis

5.16.1. Beschreibung

Wird initiiert, wenn Formularentwurf und Daten zusammengeführt wurden, das Formular vorhanden ist und das Layout des Formulars angewendet wurde. Zu diesem Zeitpunkt wurde das fertige Formular noch nicht wiedergegeben; eine Berechnung oder ein Skript, die bzw. das bei diesem Ereignis ausgeführt werden soll, könnte das Layout vor der Wiedergabe also modifizieren. Dieses Ereignis tritt auch nach der Formularwiedergabe auf, wenn eine Berechnung oder ein Skript die Daten ändert bzw. eine Änderung des Formulars in Acrobat oder Adobe Reader bewirkt.

HINWEIS: *Skripten mit „layout:ready" sollten das Layout des Formulars nicht ändern. Dies beinhaltet das Vergrößern oder Verkleinern von Teilformularen oder Tabellen, das dynamische Hinzufügen von Fragmenten zur Laufzeit, das Hinzufügen oder Löschen von Teilformularinstanzen und das Hin- und Herschalten zwischen den Funktionen „sichtbares Objekt" und „ausgeblendetes Objekt".*

Felder interaktiver Formulare mit dem „layout:ready"-Ereignis werden in Acrobat ab Version 7.0.5 unterstützt.

5.16.2. Tippen Sie

Prozessereignis

5.16.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.16.4. Version

XFA 2.1

5.16.5. Beispiel

Mit diesem Ereignis können Sie unmittelbar nach der Festlegung des Formularlayouts bestimmte Aufgaben ausführen. Beispielsweise können Sie die Anzahl der Seiten bestimmen, welche das Formular enthält.

VERKNPFTE LINKS:

Ereignisse

Prozessereignisse

5.17. mouseDown-Ereignis

5.17.1. Beschreibung

Wird initiiert, wenn ein Benutzer beim Ausfüllen des Formulars die Maustaste drückt, während sich der Zeiger in einem Feld befindet.

HINWEIS: *Wenn ein* `mouseDown` *-Ereignis für ein Text- oder numerisches Feld ausgelöst wird, werden Berechnungen oder Skripte sofort ausgeführt. Der Wert des Feldes wird aber erst dann aufgrund von Berechnungen und Skripten geändert, wenn das Feld den Fokus verliert. Wenn ein* `mouseDown` *-Ereignis für ein Unterschriftsfeld initiiert wird, wird das Ereignis vor Beginn des Unterschriftsprozesses ausgelöst.*

5.17.2. Tippen Sie

Interaktives Ereignis

5.17.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.17.4. Version

XFA 2.1

5.17.5. Beispiel

Dieses Ereignis eignet sich für die Ausführung einer Aktion, nachdem ein Benutzer im Formular auf eine Schaltfläche geklickt oder ein Optionsfeld bzw. ein Kontrollkästchen aktiviert hat. Beispielsweise können Sie mit dem `mouseDown` -Ereignis für ein Kontrollkästchen verwenden, um ein Formularfeld ein- und auszublenden. Dieses Ereignis ist konzeptionell gesehen ähnlich dem `click` Ereignis und erfüllt einen ähnlichen Zweck.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

click-Ereignis

5.18. mouseEnter-Ereignis

5.18.1. Beschreibung

Dieses Ereignis wird ausgelöst, wenn der Benutzer beim Ausfüllen des Formulars den Mauszeiger in den Feldbereich bewegt; dabei muss nicht notwendigerweise die Maustaste gedrückt werden. Es wird nicht ausgelöst, wenn der Mauszeiger aus einem anderen Grund in das Feld gelangt, z. B. weil ein darüber liegendes Fenster geschlossen wurde.

5.18.2. Tippen Sie

Interaktives Ereignis

5.18.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.18.4. Version

XFA 2.1

5.18.5. Beispiel

Sie können dieses Ereignis verwenden, um dem Benutzer beim Ausfüllen des Formulars visuelles Feedback zu liefern mit dem `mouseExit` -Ereignis. Beispielsweise können Sie mit diesem Ereignis die Rahmen- oder Hintergrundfarbe eines Objekts ändern, damit der Benutzer beim Ausfüllen visuell erkennen kann, dass er sich zurzeit in einem bestimmten Feld befindet.

Beispiele für die Verwendung des `mouseEnter` -Ereignisses finden Sie unter Felder als Reaktion auf Benutzeraktionen hervorheben.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.19. mouseExit-Ereignis

5.19.1. Beschreibung

Dieses Ereignis wird ausgelöst, wenn der Benutzer beim Ausfüllen des Formulars den Mauszeiger aus dem Feld herausbewegt, auch wenn er dabei die Maustaste gedrückt hält. Es wird nicht ausgelöst, wenn der Mauszeiger aus einem anderen Grund aus dem Feld bewegt wird, z. B. weil ein darüber liegendes Fenster geöffnet wurde.

5.19.2. Tippen Sie

Interaktives Ereignis

5.19.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.19.4. Version

XFA 2.1

5.19.5. Beispiel

Sie können dieses Ereignis verwenden, um dem Benutzer beim Ausfüllen des Formulars visuelles Feedback zu liefern mit dem `mouseEnter` -Ereignis. Beispielsweise können Sie mit diesem Ereignis die Rahmen- oder Hintergrundfarbe eines Objekts auf die ursprüngliche Einstellung zurücksetzen, damit der Benutzer beim Ausfüllen visuell erkennen kann, dass er sich nicht mehr in einem bestimmten Feld befindet.

Beispiele für die Verwendung des `mouseExit` -Ereignisses finden Sie unter Felder als Reaktion auf Benutzeraktionen hervorheben.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.20. mouseUp-Ereignis

5.20.1. Beschreibung

Wird initiiert, wenn ein Benutzer beim Ausfüllen des Formulars die Maustaste loslässt, während sich der Zeiger in einem Feld befindet.

HINWEIS: *Wenn ein* `mouseUp` *-Ereignis für ein Text- oder numerisches Feld eintritt, werden Berechnungen oder Skripten sofort ausgeführt. Der Wert des Feldes wird aber erst dann aufgrund von Berechnungen und Skripten geändert, wenn das Feld den Fokus verliert.*

5.20.2. Tippen Sie

Interaktives Ereignis

5.20.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.20.4. Version

XFA 2.1

5.20.5. Beispiel

Dieses Ereignis eignet sich für die Ausführung von Aktionen, nachdem ein Benutzer im Formular auf eine Schaltfläche geklickt oder ein Optionsfeld bzw. ein Kontrollkästchen aktiviert hat. Beispielsweise können Sie mit dem `mouseUp` -Ereignis für ein Kontrollkästchen verwenden, um ein Formularfeld ein- und auszublenden. Dieses Ereignis ist konzeptionell gesehen ähnlich dem `click` Ereignis und erfüllt einen ähnlichen Zweck.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.21. postOpen-Ereignis

5.21.1. Beschreibung

Wird ausgeführt, unmittelbar nachdem ein Benutzer beim Ausfüllen des Formulars eine Aktion ausführt, mit welcher die Daten in einer Dropdown-Liste angezeigt werden. Dies geschieht beispielsweise, wenn der Benutzer auf das Pfeilsymbol neben der Dropdown-Liste klickt oder wenn er mit der Tabulatortaste zur Dropdown-Liste wechselt und anschließend das Pfeilsymbol verwendet. Das Ereignis wird ausgelöst, nachdem der Inhalt der Dropdown-Liste angezeigt wird.

HINWEIS: *Dieses Ereignis ist nur für Dropdown-Listenobjekte vorgesehen.*

5.21.2. Tippen Sie

Interaktives Ereignis

5.21.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.21.4. Version

XFA 2.8

5.21.5. Beispiel

Mit diesem Ereignis können Sie Fehler oder unerwartete Ergebnisse behandeln, die bei der Verarbeitung des Öffnens der Dropdown-Liste auftreten. Wenn das `preOpen` -Ereignis von einem Skript anstelle einer Benutzerinteraktion gesendet wird oder wenn das Öffnen der Dropdown-Listendaten aufgrund eines Fehlers nicht eintritt, wird das `postOpen` -Ereignis trotzdem gesendet, damit Fehlerbearbeitungsskripte ausgeführt werden können.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.22. postPrint-Ereignis

5.22.1. Beschreibung

Wird sofort initiiert, sobald das wiedergegebene Formular zum Drucker, Spooler oder Ausgabeziel gesendet wurde.

5.22.2. Tippen Sie

Anwendungsereignis

5.22.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.22.4. Version

XFA 2.1

5.22.5. Beispiel

Mit diesem Ereignis können Sie Meldungen für den Benutzer anzeigen, nachdem das Formular gedruckt wurde. Sie können beispielsweise ein Skript auf dem `postPrint` -Ereignisses erstellen, um die Formularbenutzer daran zu erinnern, wie sie das Formular manuell einreichen können.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.23. postSave-Ereignis

5.23.1. Beschreibung

Wird sofort ausgeführt, nachdem ein Benutzer ein Formular im PDF- oder XDP-Format gespeichert hat. Dieses Ereignis wird nicht ausgeführt, wenn Sie eine Teilmenge des Formulars (z. B. nur Formulardaten) in das XDP-Format exportieren.

5.23.2. Tippen Sie

Anwendungsereignis

5.23.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.23.4. Version

XFA 2.1

5.23.5. Beispiel

Mit diesem Ereignis können Sie Meldungen für den Benutzer anzeigen, nachdem die Formulardaten gespeichert wurden. Sie können beispielsweise ein Skript auf dem `postSave` -Ereignis erstellen, um den Formularbenutzer daran zu erinnern, wie viel Zeit bleibt, um das Formular auszufüllen und zu senden.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.24. postSign-Ereignis

5.24.1. Beschreibung

Wird ausgelöst, unmittelbar nachdem ein Benutzer beim Ausfüllen des Formulars eine Aktion ausführt, mit welcher dem Formular eine digitale Unterschrift hinzugefügt wird.

HINWEIS: *Dieses Ereignis ist nur für das Unterschriftsfeld-Objekt vorgesehen.*

5.24.2. Tippen Sie

Interaktives Ereignis

5.24.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.24.4. Version

XFA 2.8

5.24.5. Beispiel

Mit diesem Ereignis können Sie einen Benutzer über Einschränkungen informieren, die nach der digitalen Unterzeichnung des Formulars gelten.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.25. postSubmit-Ereignis

5.25.1. Beschreibung

Wird ausgelöst, unmittelbar nachdem ein Formular über das HTTP-Protokoll Daten an den Host übergibt.

HINWEIS: *Dieses Ereignis unterscheidet nicht zwischen Übergaben, die durch Instanzen angeklickter Schaltflächen ausgelöst werden oder die für unterschiedliche URLs vorgesehen sind. Wenn ein Skript Unterscheidungen dieser Art benötigt, muss es entsprechenden Skriptcode enthalten, damit ermittelt werden kann, auf welche Schaltfläche geklickt wurde. Im Allgemeinen ist das* `postSubmit` *-Ereignis ist konzeptionell gesehen ähnlich dem* `postSave` *-Ereignis und dient einem ähnlichen Zweck.*

5.25.2. Tippen Sie

Anwendungsereignis

5.25.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes (Nur für Senden-Schaltflächen)

5.25.4. Version

XFA 2.8

5.25.5. Beispiel

Mit diesem Ereignis können Sie unmittelbar nach dem Senden der Formulardaten bestimmte Aktionen ausführen. Sie können beispielsweise ein Skript auf dem `postSubmit` -Ereignis erstellen, um eine Bestätigung anzuzeigen, dass die Datenübergabe erfolgreich durchgeführt wurde.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.26. preOpen-Ereignis

5.26.1. Beschreibung

Wird ausgeführt, wenn ein Benutzer beim Ausfüllen des Formulars eine Aktion ausführt, mit welcher die Dropdown-Liste angezeigt wird. Dies geschieht beispielsweise, wenn der Benutzer auf das Pfeilsymbol neben der Dropdown-Liste klickt oder wenn er mit der Tabulatortaste zur Dropdown-Liste wechselt und anschließend das Pfeilsymbol verwendet. Das Ereignis wird ausgelöst, bevor der Inhalt der Dropdown-Liste angezeigt wird.

HINWEIS: *Dieses Ereignis ist nur für Dropdown-Listenobjekte vorgesehen.*

5.26.2. Tippen Sie

Interaktives Ereignis

5.26.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.26.4. Version

XFA 2.4

5.26.5. Beispiel

Mit diesem Ereignis können Sie das Laden großer Mengen von Listenelementen steuern. Beispielsweise können Sie eine feste Anzahl von Datensätzen aus einer Datenquelle in eine Dropdown-Liste laden. Dadurch wird die Reaktionsgeschwindigkeit des Formulars beim Ausfüllen verbessert.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.27. prePrint-Ereignis

5.27.1. Beschreibung

Wird unmittelbar vor Beginn der Wiedergabe eines Formulars zum Drucken ausgelöst. Sie können den Druckvorgang mit diesem Ereignis nicht abbrechen.

WICHTIG: *Sie sollten dieses Ereignis nicht zum Ausblenden oder Anzeigen von Formularobjekten verwenden. Beispiel: Wenn der Benutzer das Formular beim Ausfüllen bereits digital unterschrieben hat, wirkt sich die Verwendung dieses Ereignisses zum Ausblenden aller Schaltflächenobjekte vor dem Drucken auf den Status der Unterschrift aus.*

5.27.2. Tippen Sie

Anwendungsereignis

5.27.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.27.4. Version

XFA 2.1

5.27.5. Beispiel

Mit diesem Ereignis können Sie die Präsenz eines Objekts verändern, um zu verhindern, dass es gedruckt wird. Beispielsweise können Sie Texte oder Anweisungen ausblenden, die lediglich beim Ausfüllen des Formulars sichtbar sein sollen.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.28. preSave-Ereignis

5.28.1. Beschreibung

Wird unmittelbar vor dem Speichern von Formulardaten im PDF- oder XDP-Format ausgelöst. Dieses Ereignis wird nicht ausgelöst, wenn Sie die Formulardaten oder eine andere Teilmenge des Formulars in das XDP-Format exportieren.

5.28.2. Tippen Sie

Anwendungsereignis

5.28.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.28.4. Version

XFA 2.1

5.28.5. Beispiel

Mit diesem Ereignis können Sie Formulardaten unmittelbar vor dem Speichern der Daten ändern. Sie können beispielsweise ein Skript auf dem `preSave` -Ereignis erstellen, um die Daten zu scannen und um den Benutzern eine Erinnerungsmeldung anzuzeigen, falls bestimmte erforderliche Felder noch leer sind.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

5.29. preSign-Ereignis

5.29.1. Beschreibung

Wird ausgelöst, unmittelbar bevor ein Benutzer beim Ausfüllen des Formulars eine Aktion ausführt, mit welcher dem Formular eine digitale Unterschrift hinzugefügt wird.

HINWEIS: *Dieses Ereignis ist nur für das Unterschriftsfeld-Objekt vorgesehen.*

5.29.2. Tippen Sie

Interaktives Ereignis

5.29.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	Nein

5.29.4. Version

XFA 2.8

5.29.5. Beispiel

Mit diesem Ereignis können Sie beispielsweise die Daten überprüfen, für welche die digitale Unterschrift gilt, oder einem Benutzer Informationen bereitstellen, bevor die digitale Unterschrift angewendet wird.

VERKNPFTE LINKS:

Ereignisse

Interaktive Ereignisse

5.30. preSubmit-Ereignis

5.30.1. Beschreibung

Wird ausgelöst, wenn ein Formular über das HTTP-Protokoll Daten an den Host übergibt. Zu diesem Zeitpunkt sind die Daten bereits in einem Datensatz angeordnet, wurden aber noch nicht an den Host gesendet. Diesem Ereignis zugeordnete Berechnungen und Skripten können die Daten vor der Übergabe des Formulars prüfen und verändern. Wenn die Berechnung oder das Skript für die Ausführung auf dem Server konfiguriert ist, sendet das Formular die Daten mit dem Hinweis zum Server, dass er die Berechnung oder das Skript ausführen soll, bevor eine weitere Verarbeitung der Daten erfolgt.

HINWEIS: *Dieses Ereignis unterscheidet nicht zwischen Übergaben, die durch Instanzen angeklickter Schaltflächen ausgelöst werden oder die für unterschiedliche URLs vorgesehen sind. Wenn ein Skript Unterscheidungen dieser Art benötigt, muss es entsprechenden Code enthalten, damit ermittelt werden kann, auf welche Schaltfläche geklickt wurde. Im Allgemeinen ist das* `preSubmit` *-Ereignis ist konzeptionell gesehen ähnlich dem* `preSave` *-Ereignis und dient einem ähnlichen Zweck.*

5.30.2. Tippen Sie

Anwendungsereignis

5.30.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes (Nur für Senden-Schaltflächen)

5.30.4. Version

XFA 2.1

5.30.5. Beispiel

Mit diesem Ereignis können Sie Formulardaten unmittelbar vor dem Senden der Daten ändern. Sie können beispielsweise ein Skript auf dem `preSubmit` Ereignis erstellen, um die Datenmenge zu scannen und den Benutzern eine Meldung anzuzeigen, die einschätzt, wie lange die Datenübergabe dauern kann.

VERKNPFTE LINKS:

Ereignisse

Anwendungsereignisse

preSave-Ereignis

5.31. validate-Ereignis

5.31.1. Beschreibung

Wird initiiert, wenn Formularentwurf und Daten zum Formular zusammengeführt werden und wenn ein Feld den Fokus verliert, z. B. wenn ein Benutzer klickt oder die Tabulatortaste drückt, um ein Feld zu verlassen. Dieses Ereignis wird jedes Mal ausgelöst, wenn sich der Wert eines Feldes ändert. Mit Berechnungen und Skripten, die auf das `validate` -Ereignis platziert werden, lassen sich Überprüfungen durchführen, die spezifischeren Charakter haben als die über die Registerkarte „Wert“ der Palette „Objekt“ verfügbaren Überprüfungen.

Berechnungen und Skripte im `validate` -Ereignis sind erforderlich, um `true` oder `false` (in einem Format entsprechend der Skriptsprache) entsprechend einer Validierung, die erfolgreich ist oder fehlschlägt, zurückzugeben, und darf nicht die Gesamtformularstruktur von Formularwerten beeinträchtigen. Außerdem dürfen Berechnungen und Skripten nicht versuchen, dem Formularbenutzer Rückmeldungen zu liefern, da der Benutzer das Formular möglicherweise nicht in einer Client-Anwendung wie Acrobat verwendet.

HINWEIS: *Da Überprüfungen am Inhalt des Formulars vorgenommen werden, kann damit die durch Feldmuster vorgegebene Formatierung der Darstellung nicht überprüft werden.*

5.31.2. Tippen Sie

Prozessereignis

5.31.3. Unterstützung

Client-Anwendung	Verfügbarkeit
Acrobat und Adobe Reader	yes
HTML-Browser	yes

5.31.4. Version

XFA 2.1

5.31.5. Beispiel

Mit diesem Ereignis können Sie Objektwerte überprüfen. Es eignet sich insbesondere für Situationen, in denen Objektdaten mit bestimmten Regeln übereinstimmen müssen. Sie können beispielsweise ein Skript auf dem `validate` -Ereignis erstellen, um sich zu vergewissern, dass das Feld für die Gesamtkosten in einem Bestellformular keinen negativen Wert enthält.

Beispiele für die Verwendung des `validate` -Ereignisses finden Sie unter Felder zur Laufzeit als „Erforderlich“ festlegeneingeben.

VERKNPFTE LINKS:

Ereignisse

Prozessereignisse

6. Skripterstellung mit FormCalc und JavaScript

FormCalc und JavaScript richten sich an zwei unterschiedliche Anwendergruppen, es besteht aber eine gewisse Überschneidung zwischen den jeweiligen integrierten Funktionen. In der folgenden Tabelle sind alle verfügbaren FormCalc-Funktionen zusammengestellt; dabei ist jeweils angegeben, ob JavaScript eine vergleichbare Funktion enthält.

Weitere Informationen zu FormCalc-Funktionen und deren Parametern finden Sie unter Integrierte Funktionssyntax.

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Abs(n1)`	Gibt den Betragswert eines Zahlenwerts oder Ausdrucks zurück.	`Math.abs(n1)`
`Apr(n1, n2, n3)`	Gibt die jährliche Gesamtbelastung für einen Kredit zurück.	Kein
`At(s1, s2)`	Findet die Anfangs-Zeichenposition einer Zeichenfolge innerhalb einer anderen Zeichenfolge.	`String.search(s1)`
`Avg(n1 [, n2... ] )`	Wertet einen Satz von Zahlenwerten und/oder Ausdrücken aus und gibt den Mittelwert der von null verschiedenen Elemente innerhalb dieses Satzes zurück.	Kein
`Ceil(n1)`	Gibt die ganze Zahl zurück, die größer oder gleich einer angegebenen Zahl ist.	`Math.ceil(n1)`
`Choose(n1, s1 [, s2... ] )`	Wählt einen Wert aus einem gegebenen Satz von Parametern.	Kein
`Concat(s1 [, s2... ] )`	Gibt die Verkettung von zwei oder mehr Zeichenfolgen zurück.	`String.concat(s1, s2 [, s3 ... ])`

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Count(n1 [, n2...])`	Wertet einen Satz von Werten und/oder Ausdrücken aus und gibt die Anzahl der von null verschiedenen Elemente innerhalb dieses Satzes zurück.	Kein
`CTerm(n1, n2, n3)`	Gibt die Anzahl der Perioden zurück, die erforderlich sind, damit eine Anlage mit einer festen Verzinsung mit Zinseszins auf einen Endwert anwächst.	Kein
`Datum()`	Gibt das aktuelle Systemdatum als Anzahl von Tagen seit Beginn der Epoche zurück.	`Date.getDate()` Das JavaScript-Datumsobjekt verwendet die Epoche nicht als Referenzpunkt.
`Date2Num(d1 [, f1 [, k1 ] ] )`	Gibt für eine angegebene Datums-Zeichenfolge die Anzahl der Tage seit Beginn der Epoche zurück.	Das JavaScript-Datumsobjekt verwendet die Epoche nicht als Referenzpunkt.
`DateFmt([ n1 [, k1 ] ] )`	Gibt für einen angegebenen Datumsformat-Stil eine Datumsformat-Zeichenfolge zurück.	Kein
`Decode(s1 [, s2 ] )`	Gibt die dekodierte Version einer angegebenen Zeichenfolge zurück.	Zum Teil unterstützt JavaScript unterstützt nur URL-kodierte Werte ohne Escape-Zeichen.
`Encode(s1 [, s2 ] )`	Gibt die kodierte Version einer angegebenen Zeichenfolge zurück.	Zum Teil unterstützt JavaScript unterstützt nur URL-kodierte Werte ohne Escape-Zeichen.
`Eval()`	Gibt den Wert einer angegebenen Formularberechnung zurück.	`eval(s1)`
`Exists(v1)`	Bestimmt, ob der angegebene Parameter eine gültige Referenz-Syntax zu einem vorhandenen Objekt ist.	Kein

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Floor(n1)`	Gibt die größte Ganzzahl zurück, die kleiner oder gleich dem angegebenen Wert ist.	`Math.floor(n1)`
`Format(s1, s2)`	Formatiert die angegebenen Daten gemäß der angegebenen Musterformat-Zeichenfolge.	Kein
`FV(n1, n2, n3)`	Gibt den Endwert von Anlagebeträgen zurück, die regelmäßig und in gleich bleibender Höhe bei einem konstanten Zinssatz eingezahlt werden.	Kein
`Get(s1)`	Lädt den Inhalt der angegebenen URL herunter.	Kein
`HasValue(v1)`	Ermittelt, ob der angegebene Parameter eine gültige Referenz-Syntax mit einem von null verschiedenen, nicht leeren oder vom Leerzeichen verschiedenen Wert ist.	Kein
`IPmt(n1, n2, n3, n4, n5)`	Gibt den Zinsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.	Kein
`IsoDate2Num(d1)`	Gibt bei Angabe einer gültigen Datums-Zeichenfolge die Anzahl der Tage seit Beginn der Epoche zurück.	Kein
`IsoTime2Num(d1)`	Gibt für eine angegebene gültige Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.	Kein
`Left(s1, n1)`	Extrahiert eine angegebene Anzahl von Zeichen aus einer Zeichenfolge, beginnend beim ersten Zeichen auf der linken Seite.	`String.substring(n1, n2)`

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Len(s1)`	Gibt die Anzahl der Zeichen in einer angegebenen Zeichenfolge zurück.	`String.length`
`LocalDateFmt( [ n1 [, k1 ] ] )`	Gibt eine lokalisierte Datumsformat-Zeichenfolge mit dem angegebenen Datumsformat-Stil zurück.	Kein
`LocalTimeFmt( [ n1 [, k1 ] ] )`	Gibt eine lokalisierte Uhrzeitformat-Zeichenfolge mit dem angegebenen Uhrzeitformat-Stil zurück.	Kein
`Lower(s1 [, k1 ] )`	Wandelt alle Großbuchstaben in einer angegebenen Zeichenfolge in Kleinbuchstaben um.	`String.toLowerCase(s1)`
`Ltrim(s1)`	Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Anfang entfernt wurden.	Kein Sie können normale JavaScript-Ausdrücke verwenden, um diesen Vorgang auszuführen.
`Max(n1 [, n2... ] )`	Gibt den Maximalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.	`Math.max(n1, n2)`
`Min(n1 [, n2... ] )`	Gibt den Minimalwert der von null verschiedenen Elemente im angegebenen Satz von Zahlen zurück.	`Math.min(n1, n2)`
`Mod(n1, n2)`	Gibt den Modulus der Division einer Zahl durch eine andere zurück.	Verwenden Sie den `modulo (%)`-Operator.
`NPV(n1, n2 [, ... ] )`	Gibt den Kapitalwert einer Anlage auf der Grundlage eines Diskontsatzes und einer Folge von zukünftigen periodischen Cashflows zurück.	Kein
`Num2Date(n1[, f1 [, k1 ] ] )`	Gibt für eine angegebene Anzahl von Tagen seit Beginn der Epoche eine Datums-Zeichenfolge zurück.	Kein

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Num2GMTime(n1 [,f1 [, k1 ] ] )`	Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine GMT-Uhrzeit-Zeichenfolge zurück.	Kein
`Num2Time(n1 [,f1 [, k1 ] ] )`	Gibt für eine angegebene Anzahl von Millisekunden seit Beginn der Epoche eine Uhrzeit-Zeichenfolge zurück.	Kein
`Oneof(s1, s2 [, s3... ] )`	Gibt TRUE (1) zurück, wenn sich ein Wert in einem angegebenen Satz befindet, bzw. FALSE (0), wenn dies nicht der Fall ist.	Kein Diese Funktion ähnelt der `String.search(`*`s1`*`)` -Methode und dem `String.match(`*`-Ausdruck`*`)` -Methode eingestellt.
`Parse(s1, s2)`	Analysiert die angegebenen Daten gemäß dem angegebenen Musterformat.	Kein
`Pmt(n1, n2, n3)`	Gibt die Höhe des Rückzahlungsbetrags für einen Kredit bei konstanten Zahlungsbeträgen und konstantem Zinssatz zurück.	Kein
`Post(s1, s2 [, s3 [, s4 [, s5 ] ] ] )`	Sendet die angegebenen Daten an die genannte URL.	Kein
`PPmt(n1, n2, n3, n4, n5)`	Gibt den Tilgungsbetrag zurück, der in einer bestimmten Zeitspanne für einen Kredit gezahlt wurde.	Kein
`Put(s1, s2 [, s3 ] )`	Lädt die angegebenen Daten zu der genannten URL hoch.	Kein
`PV(n1, n2, n3)`	Gibt den Gegenwartswert einer Anlage mit regelmäßigen konstanten Einzahlungen und konstantem Zinssatz zurück.	Kein

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Rate(n1, n2, n3)`	Gibt den Zinssatz pro Verzinsungsperiode zurück, der benötigt wird, damit eine Anlage mit Zinseszins in einem gegebenen Zeitraum von einem gegebenen Gegenwartswert auf einen Endwert anwächst.	Kein
`Ref()`	Gibt eine Referenz auf ein vorhandenes Objekt zurück.	Kein
`Replace(s1, s2 [, s3 ] )`	Ersetzt innerhalb einer angegebenen Zeichenfolge alle Fundstellen einer Zeichenfolge durch eine andere.	`String.replace(s1, s2)`
`Right(s1, n1)`	Extrahiert mehrere Zeichen aus einer angegebenen Zeichenfolge, beginnend beim letzten Zeichen auf der rechten Seite.	`String.substring(n1, n2)`
`Round(n1 [, n2 ] )`	Wertet einen angegebenen Zahlenwert oder Ausdruck aus und gibt eine auf die angegebene Anzahl von Dezimalstellen gerundete Zahl zurück. Verwenden Sie für genauere Ergebnisse eine ältere Markierung in xfa.xci. Um zum Standardverhalten zu wechseln, entfernen Sie die ältere Markierung.	`Math.round(n1)`
`Rtrim(s1)`	Gibt eine Zeichenfolge zurück, bei der alle Leerraum-Zeichen am Ende entfernt wurden.	Kein Sie können normale JavaScript-Ausdrücke verwenden, um diesen Vorgang auszuführen.
`Space(n1)`	Gibt eine Zeichenfolge zurück, die aus einer angegebenen Anzahl von Leerzeichen besteht.	Kein

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`Str(n1 [, n2 [, n3 ] ] )`	Wandelt eine Zahl in eine Zeichenfolge um. FormCalc formatiert das Ergebnis auf die angegebene Breite und rundet auf die angegebene Zahl von Dezimalstellen.	`Zeichenfolge(n1)` oder `Number.toString(radix)`
`Stuff(s1, n1, n2 [, s2 ] )`	Fügt eine Zeichenfolge in eine andere Zeichenfolge ein.	Kein
`Substr(s1, n1, n2)`	Extrahiert einen Abschnitt aus einer angegebenen Zeichenfolge.	`String.substring(n1, n2)`
`Sum(n1 [, n2... ] )`	Gibt die Summe der von null verschiedenen Elemente eines gegebenen Satzes von Zahlen zurück.	Kein
`Term(n1, n2, n3)`	Gibt die Anzahl der Perioden zurück, die erforderlich sind, um mit konstanten periodischen Einzahlungen auf ein verzinstes Konto einen angegebenen Endwert zu erzielen.	Kein
`Time()`	Gibt die aktuelle Systemuhrzeit als Anzahl von Millisekunden seit Beginn der Epoche zurück.	`Date.getTime()` Das JavaScript-Datumsobjekt verwendet die Epoche nicht als Referenzpunkt.
`Time2Num(d1 [, f1 [, k1 ] ] )`	Gibt für eine angegebene Uhrzeit-Zeichenfolge die Anzahl der Millisekunden seit Beginn der Epoche zurück.	Kein
`TimeFmt([n1 [, k1 ] ] )`	Gibt ein Uhrzeitformat in einem angegebenen Uhrzeitformat-Stil zurück.	Kein
`UnitType(s1)`	Gibt die Einheit einer Maßangabe zurück. Eine *unitspan* ist eine Zeichenfolge, die aus einer Zahl und einer nachfolgenden Einheitenbezeichnung besteht.	Kein

FormCalc-Funktion	Beschreibung	Entsprechende JavaScript-Methode
`UnitValue(s1 [, s2 ] )`	Gibt den Zahlenwert einer Messung mit ihrer zugeordneten Maßangabe nach einer optionalen Einheitenumwandlung zurück.	Kein
`Upper(s1 [, k1 ] )`	Wandelt alle Kleinbuchstaben in einer angegebenen Zeichenfolge in Großbuchstaben um.	`String.toUpperCase()`
`Uuid(n1)`	Gibt eine UUID-Zeichenfolge (Universally Unique Identifier) zurück, die als Kennzeichnungsmethode verwendet werden soll.	Kein
`Within(s1, s2, s3)`	Gibt TRUE (1) zurück, wenn der geprüfte Wert innerhalb eines angegebenen Bereichs liegt, bzw. FALSE (0), wenn dies nicht der Fall ist.	`String.search(s1)`
`WordNum(n1 [, n2 [, k1 ] ] )`	Gibt die englischsprachige Textentsprechung einer angegebenen Zahl zurück.	Kein

VERKNPFTE LINKS:

FormCalc verwenden

6.1. FormCalc verwenden

FormCalc ist eine einfache, aber leistungsfähige Berechnungssprache, die nach dem Vorbild allgemein gebräuchlicher Tabellenkalkulationssoftware entwickelt wurde. Diese Sprache soll die schnelle, effiziente Gestaltung von Formularen ohne Kenntnis konventioneller Skripterstellungstechniken oder Skriptsprachen ermöglichen. Mit Hilfe von wenigen integrierten Funktionen können Benutzer, die FormCalc noch nicht kennen, in kurzer Zeit Formulare erstellen, welche den Endbenutzern zeitaufwendige Berechnungen, Kontrollen und andere Überprüfungsmaßnahmen abnehmen. Auf diese Weise haben Sie die Möglichkeit, für den Formularentwurf einen Satz Grundregeln aufzustellen, anhand dessen das resultierende Formular auf die Daten reagieren kann, mit denen es in Berührung kommt.

In Designer ist FormCalc die beim Erstellen von Skripten generell verwendete Standard-Skriptsprache. Alternativ wird JavaScript eingesetzt. Informationen zur Festlegung der Standard-Skriptsprache finden Sie unter Konfigurieren von Designer für die Skripterstellung.

WICHTIG: *Wenn Sie Formulare für die Nutzung in einem serverbasierten Prozess erstellen (z. B. mit Foms) und diese in HTML wiedergeben möchten, sollten Sie Ihre Berechnungen und Skripten in JavaScript entwickeln. FormCalc-Berechnungen sind in HTML-Browsern nicht gültig und werden vor der Wiedergabe in HTML aus dem Formular entfernt.*

FormCalc behandelt jede neue Zeile im Skript-Editor als neu auszuwertenden Ausdruck.

VERKNPFTE LINKS:

JavaScript verwenden

6.2. Integrierte Funktionen verwenden

Die integrierten Funktionen, aus denen sich FormCalc zusammensetzt, decken eine umfangreiche Palette von Bereichen ab: Mathematik, Datum und Uhrzeit, Zeichenketten, Finanzen, Logik und Internet. Diese Bereiche entsprechen den Funktionalitätsarten, die in Formularen üblicherweise genutzt werden. Die Funktionen sollen eine schnelle und einfache Bearbeitung von Formulardaten auf zweckmäßige Weise ermöglichen.

Auf der elementarsten Ebene kann eine Berechnung aus einer einzelnen FormCalc-Funktion bestehen. Diese einzelne FormCalc-Funktion kann aber weitere FormCalc-Funktionen als Parameter nutzen.

6.2.1. So fügen Sie einem Objekt eine FormCalc-Funktion hinzu

Sie können eine FormCalc-Funktion jedem Formularentwurfsobjekt hinzufügen, das Berechnungen und Skripten zulässt, mit Ausnahme des Skriptobjekts.

1) Achten Sie darauf, dass im Designer-Arbeitsbereich die mehrzeilige Version des Skript-Editors angezeigt wird.
2) Wählen Sie ein Feld in Ihrem Formular aus.
3) Wählen Sie in der Liste „Anzeigen“ das Ereignis „calculate“.
4) Klicken Sie auf das Symbol „Funktionen“ oder drücken Sie die Taste F10 zum Einblenden einer Liste der FormCalc-Funktionen.
5) Wählen Sie die entsprechende Funktion und drücken Sie die Eingabetaste.
6) Ersetzen Sie die Standardnotation der Funktionssyntax durch Ihre eigenen Werte.
7) Zum Testen des Formulars klicken Sie auf die Registerkarte „PDF-Vorschau“.

6.2.2. Integrierte Funktionssyntax

Jede FormCalc-Funktion verwendet eine bestimmte Syntax-Notation, die Sie zur korrekten Ausführung der Funktion einhalten müssen. In dieser Tabelle werden die Elemente der Syntax-Notation in allgemeiner Form beschrieben.

Syntax-Notation	Ersetzungswerte
`d`	Eine gültige Datums-Zeichenfolge (Beispiel: 03/15/1996)
`f`	Eine gültige Datumsformat-Zeichenfolge (Beispiel: MM/TT/JJJJ)
`k`	Eine gültige Gebietsschema-Kennung (Beispiel: fr_FR)
`n`	Ein gültiger Zahlenwert. Beachten Sie, dass der Bereich der gültigen Werte je nach Funktion unterschiedlich ist.
`S`	Eine gültige Maßeinheit (Beispiel: „mm“ für „Millimeter“).
`v`	Eine gültige Referenz-Syntax.
`n1, n2, n3`	Alle Werte sind obligatorisch.
`[ [ n [, k ] ] ]`	Keiner der Werte ist obligatorisch, Sie können aber fakultativ entweder nur n oder sowohl n als auch k angeben.
`n1 [, n2 ... ]`	n1 ist obligatorisch, Sie können aber fakultativ beliebig viele weitere Werte angeben.
`d [, f [, k ] ]`	d ist obligatorisch, Sie können aber fakultativ entweder f oder sowohl f als auch k angeben.

VERKNPFTE LINKS:

Basisberechnungen erstellen

FormCalc verwenden

6.3. Basisberechnungen erstellen

6.3.1. Grundlagen zu Basisberechnungen

Einfache Ausdrücke sind die grundlegenden Elemente in der Skripterstellung. Diese Ausdrücke verwenden keine integrierten Funktionen von FormCalc und sind niemals länger als eine Zeile. Fügen Sie dem calculate-Ereignis eines bestimmten Feldes oder Objekts einfache Ausdrücke hinzu, damit der Wert des Ausdrucks auf dem Formular ausgegeben wird.

6.3.2. Beispiele für Basisberechnungen

Die folgenden Beispiele zeigen einfache Ausdrücke:

```
2 "abc" 2 - 3 * 10 / 2 + 7
```

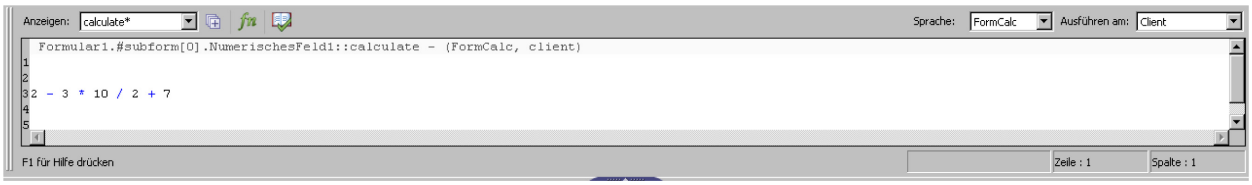
Jeder einfache Ausdruck wird zu einem einzelnen Wert ausgewertet. Dabei gilt die konventionelle Reihenfolge der Operationen, auch wenn die Reihenfolge aus der Syntax der Ausdrücke nicht immer offensichtlich ist. Beispielsweise liefern die folgenden Gruppen von Ausdrücken jeweils das gleiche Ergebnis.

Ausdruck	Entspricht	Ergebnis
`"abc"`	`"abc"`	`abc`
`2 - 3 * 10 / 2 + 7`	`2 - (3 * 10 / 2) + 7`	`-6`
`(10 + 2) * (5 + 4)`	`(10 + 2) * (5 + 4)`	`108`
`0 und 1 oder 2 > 1`	`(0 und 1) oder (2 >1)`	`1 ( true)`
`(2 < 3) nicht (1 == 1)`	`(2 < 3) nicht (1 == 1)`	`0 (false)`

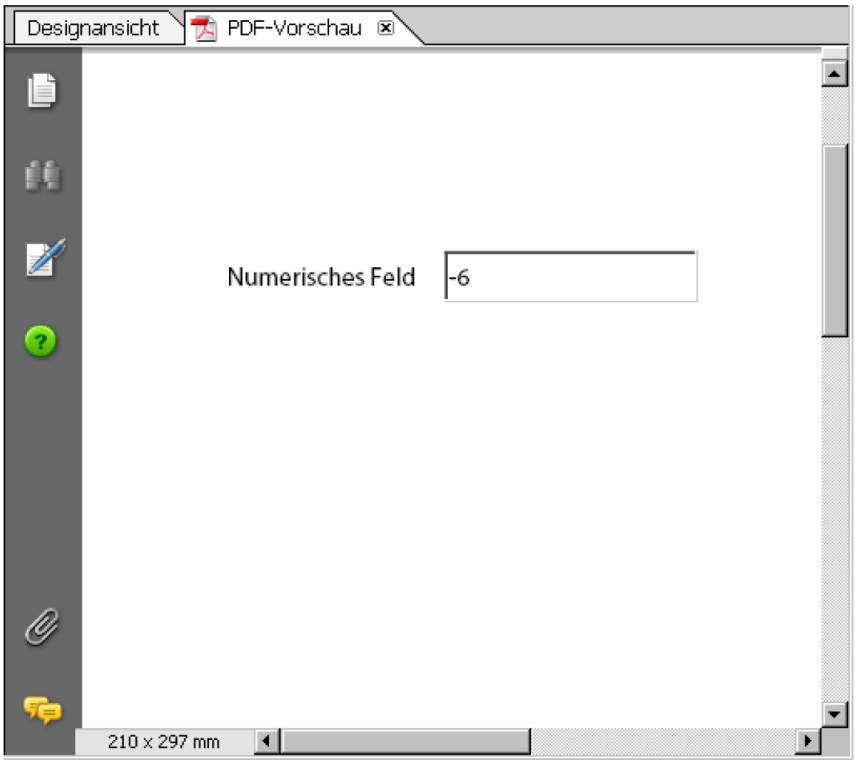
Wie aus der obigen Tabelle hervorgeht, besitzen alle FormCalc-Operatoren innerhalb von Ausdrücken eine bestimmte Priorität. Die folgende Tabelle zeigt die Hierarchie der Operatoren im Überblick.

Priorität	Operator
Höchste	=
	(Unär) - , + , not
	* , /
	+ , -
	< , <= , > , >= , lt , le , gt , ge
	== , <> , eq , ne
	& , and
Niedrigste	| , oder

Alle vorherigen Beispiele sind gültige einfache Ausdrücke, die Sie in ein Formularfeld oder Objekt einfügen können, das Berechnungen und Skripten akzeptiert. Angenommen, Sie erstellen in Designer ein Formular mit einem einzelnen numerischen Feld und fügen dem Ereignis „calculate“ im Skript-Editor die folgende Berechnung hinzu.



Wenn Sie anschließend die Registerkarte „PDF-Vorschau“ zum Anzeigen des ausgefüllten Formulars wählen, erscheint in dem Textfeld der Wert des einfachen Ausdrucks.



Falls der Wert in der PDF-Vorschau nicht angezeigt wird, vergewissern Sie sich, dass der einfache Ausdruck im calculate-Ereignis des Formularentwurfsobjekts erscheint. Sie müssen außerdem sicherstellen, dass Designer und Acrobat ordnungsgemäß installiert sind.

VERKNPFTE LINKS:

FormCalc verwenden

JavaScript verwenden

6.4. JavaScript verwenden

Um Formulardesignern mehr Flexibilität und Skripterstellungsmöglichkeiten bieten zu können, wird in Designer die Verwendung von JavaScript (Version 1.6 oder älter) für jedwede Aspekte der Skripterstellung unterstützt.

Mit JavaScript vertraute Formulardesigner können ihre Kenntnisse in Designer anwenden. Designer stellt zahlreiche Eigenschaften und Methoden zur Verfügung, die JavaScript so erweitern, dass Sie damit auf Feld- und Objektwerte zugreifen können. Diese Eigenschaften und Methoden ermöglichen Ihnen in Verbindung mit der Referenzsyntax von Designer eine einfache Manipulation von Formularwerten und -daten.

HINWEIS: *Im Skript-Editor gibt es für Skripten, die mit JavaScript erstellt wurden, keine Möglichkeit zur Überprüfung der Syntax auf Fehler. Es werden keine Anweisungsende-Optionen für standardmäßige JavaScript-Objekte oder -Methoden angezeigt.*

VERKNPFTE LINKS:

Skripten mit JavaScript erstellen

6.5. Skripten mit JavaScript erstellen

Die Skripterstellung in Designer mit JavaScript ist der JavaScript-Erstellung in anderen Anwendungen ähnlich. Sie können Ihre Vorkenntnisse zu JavaScript-Konzepten anwenden sowie im Designer-Skriptobjekt JavaScript-Funktionen einsetzen und die JavaScript-Sprachfunktionalität nutzen.

Dabei ist zu beachten, dass sich JavaScript-Vorkenntnisse zwar übertragen lassen, Sie jedoch auch wissen müssen, wie eine Referenzsyntax in Designer aufgebaut wird, damit Sie JavaScript in Ihren Formularentwürfen effektiv anwenden können. Insbesondere müssen Sie wissen, wie Sie die XML-Formularobjektmodell-Referenzsyntax korrekt einsetzen, um im Formularentwurf auf Objekte zuzugreifen.

In der folgenden Tabelle werden die Hauptkonzepte zum Entwickeln von Skripten in JavaScript für Designer erläutert. Darüber hinaus erfahren Sie, wo Sie weitere Informationen zu den einzelnen Konzepten in der *Designer-Hilfe* finden können.

Hauptkonzept	Weitere Informationsquellen
Referenzen zu Objekteigenschaften und -werten, einschließlich der Verwendung der `resolveNode` -Methode erstellen.	Objekteigenschaften und -werte referenzieren resolveNode So erstellen Sie Berechnungen und Skripten mit dem Anweisungsende
Verwenden der Host- und Ereignismodelle zum Testen und Debugging von Formularen	Berechnungen und Skripten testen und debuggen Objekteigenschaften und -werte referenzieren
Verwenden des Skriptobjekts zur Wiederverwendung existierender JavaScript-Funktionen	JavaScript-Funktionen erstellen und wiederverwenden

Zusätzlich zu den Ressourcen in der *Designer-Hilfe* enthält das Developer Center ausführliche Skripterstellungsressourcen und Dokumentation.

VERKNPFTE LINKS:

Strikte Scoping-Regeln in JavaScript erzwingen

So fügen Sie einem Objekt ein JavaScript-Skript hinzu

JavaScript verwenden

6.6. Strikte Scoping-Regeln in JavaScript erzwingen

Bei der Arbeit mit JavaScript in Formularen ist es wichtig, Objekte und Variablen stets im beabsichtigten Scope (Gültigkeitsbereich) zu deklarieren. Werden Objekte oder Variablen unnötigerweise global deklariert, kann dies zu Beeinträchtigungen der Leistung führen.

In Designer 8.1 wurden strikte Scoping-Regeln eingeführt, um die Laufzeit und Speichernutzung von Formularen zu optimieren. In Designer sind strikte Scoping-Regeln für neue Formulare standardmäßig aktiviert. Für alte Formulare ist eine Option zum Aktivieren strikter Scoping-Regeln verfügbar.

6.6.1. Funktionsweise von Scopes in JavaScript

Scopes wirken nach außen hin. Dies bedeutet, dass der gesamte in geschweifte Klammern ({}) eingeschlossene Ausdruck über die geschweiften Klammern hinaus blicken kann. Ausdrücke außerhalb der geschweiften Klammern können jedoch nicht auf den in Klammern eingeschlossenen Ausdruck zugreifen.

Im folgenden Beispiel öffnet die erste geschweifte Klammer den Scope der Funktion, während die zweite ihn schließt. Sämtliche Angaben zwischen den geschweiften Klammern befinden sich im Scope von foo ().

```
        nOutsideVar befindet sich außerhalb
                    des Scopes von foo()  -> var nOutsideVar = 2;
                                             function foo()
  Hiermit wird der Scope der Funktion geöffnet. -> {
                                                       // Sämtliche Angaben zwischen den beiden
                                                       // geschweiften Klammern befinden sich im
                                                       // Scope von foo().
                                                       var nFooVar = 4;
Hiermit wird der Scope der Funktion geschlossen. -> }
```

Der Bereich im folgenden Beispiel ist gültig, weil `var nFooVar = nOutsideVar` in eckigen Klammern `var nOutsideVar = 2` außerhalb der geschweiften Klammern erkennt.

```
var nOutsideVar = 2;
function foo()
{
  var nFooVar = nOutsideVar; // Das ist richtig.
                             // Alle Ausdrücke innerhalb der
                             // Klammern können über die Klammern
                             // hinausblicken.
}
```

Im Gegensatz dazu zeigt folgendes Beispiel einen ungültigen Bereich, da `var nOutsideVar = nFooVar` nicht auf `var nFooVar =4` innerhalb der geschweiften Klammern zugreifen kann.

```
function foo()
{
  var nFooVar = 4;
}
var nOutsideVar = nFooVar; // Das ist falsch.
                           // OutsideVar kann nicht auf Angaben
                           // zugreifen, die im Scope von foo()
                           // deklariert sind.
```

Bei der Skripterstellung beschreiben Scopes Skriptelemente, die auf andere Elemente zugreifen können. Bei diesen Skriptelementen kann es sich um Variablen oder Funktionen handeln.

6.6.2. XML-Scope

In Formularentwürfen geht es bei Scopes um die Hierarchie. Um zum Beispiel auf das Teilformular `inside` innerhalb der folgenden XML-Quelle zuzugreifen, müssen Sie *outside.inside* schreiben.

```
<subform name="outside">
<subform name="inside">
…
</subform> </subform>
```

Sie schreiben nicht *inside.outside*, da Sie zuerst auf das äußere Teilformular und erst dann auf untergeordnete Ebenen zugreifen müssen.

6.6.3. SOM-Ausdrücke und Scopes

In Formularen, die für Acrobat Adobe Reader 8.1 bestimmt sind, muss das Scoping für SOM-Ausdrücke wie im nachstehenden Beispiel erfolgen:

```
<subform name="a">

<subform name="b"/>
```

In Formularen, die für Acrobat oder Adobe Reader 8.0 bestimmt sind, gibt der SOM-Ausdruck `a.b.a` das Teilformular `a`zurück. In Formularen, die für Acrobat oder Adobe Reader 8.1 bestimmt sind, gibt der SOM-Ausdruck `a.b.` a `null zurück`, da das Teilformular `b` kein untergeordnetes Element mit dem Namen `a`hat. In Acrobat oder Adobe Reader 9.0 oder höher gibt der Ausdruck einen Fehler zurück, da `a` kein untergeordnetes Element von `b`ist.

In Acrobat oder Adobe Reader 8.1 werden Funktionen und Variablen im Skript einer Node nicht zu globalen Funktionen oder Variablen (Skriptobjekte stellen eine Ausnahme dar). Beispiel:

```
<field name="field1">

event activity="initialize">

<script contentType="application/x-javascript">

// Funktionsbalken() ist Bereich von field1.initialize; nicht außerhalb <event
activity="initialize"> Bereich kann hier erkennen (in 8.1)

function bar()

{

return "bar";

}

</script>

</event>

/field>

field name="field2">

<event activity="click">

<script contentType="application/x-javascript">

field1.bar();

</script>

</event>
```

```
</field>
```

Bei Auswahl von `field 2` in einem Formular, das für Acrobat oder Adobe Reader 8.0 vorgesehen ist, führt die Funktion `bar()` aus.

Bei Auswahl von `field 2` in einem Formular, das für Acrobat oder Adobe Reader 8.1 vorgesehen ist, wird die Funktion `bar()` nicht ausgeführt. Der Grund dafür ist, dass die Funktion `bar()` nur innerhalb des initialisierten Skripts von `feld 1`verfügbar ist.

6.6.4. Scoping und Skriptobjekte

Skriptobjekte weisen einen globalen Scope auf. Daher kann der Zugriff darauf von einer beliebigen Position aus durch einen beliebigen Benutzer erfolgen. Wenn eine Methode auf `field1.initialize` und `field2.click` zugreifen soll, müssen Sie die Methode in ein Skriptobjekt platzieren. Bei striktem Scoping können Sie nicht `bar()` von überall überall in einem Formular aufrufen. Es wird außerdem ein Laufzeitfehler angezeigt, der anzeigt, dass die Funktion `bar()` nicht gelöst werden konnte. Das Skriptmodul hat nach `bar()` innerhalb des Bereichs gesucht, auf den Sie Zugriff haben, und wurde nicht gefunden.

6.6.5. Scoping und Zielversion

Bei Verwendung des strikten Scopings erzielen Sie Verbesserungen der Leistung in Formularen, die für Acrobat oder Adobe Reader 8.1 und höher bestimmt sind. Vermeiden Sie striktes Scoping in Formularen, die für ältere Versionen von Acrobat oder Adobe Reader bestimmt sind. Andernfalls variiert die Funktionalität der Skripten in den Formularen. Erstellen Sie eine Sicherungskopie von vorhandenen Formularen, bevor Sie das strikte Scoping aktivieren. Überprüfen Sie das Skript anschließend. Wenn Sie das strikte Scoping aktivieren und danach die Zielversion auf eine Version vor Acrobat oder Adobe Reader 8.1 ändern, werden Warnungen angezeigt.

6.6.6. Verwendungsbereich des Scopings

Wurde bei Formularen für Acrobat oder Adobe Reader ab Version 8.1 das strikte Scoping aktiviert, werden deklarierte JavaScript-Variablen nach der Ausführung des jeweiligen Skripts freigegeben. Bei Formularen für Acrobat oder Adobe Reader ab Version 9.0 gibt das strikte Scoping nicht alle JavaScript-Variablen frei. Eine Ausnahme bildet das erneute Zusammenführen oder Importieren von neuen Daten.

Die Verbesserung der Leistung in Verbindung mit strikten Scoping-Regeln trifft für Formulare zu, die für Acrobat oder Adobe Reader 8.1 und höher bestimmt sind. Bei Formularen für ältere Versionen als Acrobat oder Adobe Reader 8 sollte von der Anwendung strikter Scoping-Regeln abgesehen werden. Die Skripten verhalten sich möglicherweise anders oder sind nicht funktionsfähig.

6.6.7. So aktivieren Sie das strikte Scoping

1) Wählen Sie „Datei“ > „Formulareigenschaften“ aus und klicken Sie auf die Registerkarte „Runtime“.

2) Wählen Sie „Strikte Scoping-Regeln in JavaScript erzwingen“, wenn die Option zur Verfügung steht, und klicken Sie dann auf „OK“.

   HINWEIS: *Wenn die Option zum Erzwingen strikter Scoping-Regeln auf der Registerkarte „Runtime“ nicht verfügbar ist, dann ist striktes Scoping bereits aktiviert.*

VERKNPFTE LINKS:

JavaScript verwenden

So fügen Sie einem Objekt ein JavaScript-Skript hinzu

6.7. So fügen Sie einem Objekt ein JavaScript-Skript hinzu

Sie können jedem Formularentwurfsobjekt, das Berechnungen und Skripten zulässt, ein JavaScript-Skript hinzufügen. Dies gilt auch für das Skriptobjekt.

1) Achten Sie darauf, dass im Designer-Arbeitsbereich die mehrzeilige Version des Skript-Editors angezeigt wird.

2) Wählen Sie ein Feld in Ihrem Formular aus. Fügen Sie dem Formularentwurf beispielsweise ein neues Textfeld hinzu.

3) Wählen Sie in der Liste „Anzeigen“ ein gültiges Ereignis aus. Beispiel: Wählen Sie für das neue Textfeld das `docReady` -Ereignis auswählen.

4) Wählen Sie in der Liste "Ausführen am", wo das Skript ausgeführt werden soll. Beispiel: Wählen Sie für das neue Textfeld „Client“ aus.

5) Klicken Sie auf das Symbol „Funktionen“ oder drücken Sie die Taste F10 zum Einblenden einer Liste der JavaScript-Funktionen.

6) Wählen Sie die gewünschte Funktion und drücken Sie die Eingabetaste.

7) Ersetzen Sie die Standardnotation der Funktionssyntax durch Ihre eigenen Werte. Alternativ können Sie im Skript-Editor im Feld „Skriptquelle“ ein Skript manuell eingeben. Fügen Sie in dem neuen Textfeld beispielsweise das folgende JavaScript zum Feld „Skriptquelle“ hinzu:

   ```
   this.border.fill.color.value = "255,0,0";
   ```

8) Zum Testen des Formulars klicken Sie auf die Registerkarte „PDF-Vorschau“. Bei Anzeige des Formulars auf der Registerkarte „PDF-Vorschau“ sollte das Textfeld für das neue Schaltflächenobjekt rot dargestellt sein.

VERKNPFTE LINKS:

JavaScript verwenden

7. Variablen

Sie können in Designer Formularvariablen definieren, um spezifische Informationen an einer zentralen, zugänglichen Position zu speichern. Eine *Variable* dient üblicherweise als Platzhalter für Text, den Sie zu einem späteren Zeitpunkt eventuell ändern müssen. Formularvariablen in Designer sind immer vom Typ „String“. Beispielsweise kann in einer Variablen der Titel eines Meldungsdialogfelds gespeichert werden. Wenn der Text geändert werden muss, müssen Sie lediglich das betreffende Formular bzw. die Vorlage öffnen und den Text mit Hilfe der Variablendefinition einmalig aktualisieren. Designer fügt den neuen Text automatisch an allen Stellen ein, an denen die eingefügte Variable steht.

Denken Sie daran, dass Formularvariablen außerhalb des Skript-Editors definiert werden und anders als Skriptvariablen, die Sie in einem bestimmten FormCalc- oder JavaScript-Skript erstellen, für Skripten in allen Objekten eines Formulars zugänglich sind.

Sie können Variablen ohne Eingabe von Skript-Code erstellen, anzeigen und löschen. Sie müssen allerdings Skript-Code eingeben, um auf die von Variablen gespeicherten Werte zuzugreifen und diese zu bearbeiten oder um die Werte auf Objekte in Ihrem Formular anzuwenden.

HINWEIS: *Die Werte von Formularvariablen werden jedes Mal zurückgesetzt, wenn Sie ein Formular öffnen.*

Bevor Sie eine Variable erstellen, legen Sie den Namen der Variablen fest sowie den Text, den sie enthalten soll. Variablendefinitionen werden mit dem Formular bzw. mit der Vorlage gespeichert.

7.1. Variablen benennen

Zur Laufzeit treten Namenskonflikte auf, wenn die Namen von Variablen mit den Namen von Eigenschaften oder Methoden des XML Form Object Model bzw. mit Feldnamen des Formularentwurfs identisch sind. Diese Konflikte können dazu führen, dass Skripten unerwartete Werte zurückgeben. Daher ist es wichtig, jeder Variablen einen eindeutigen Namen zuzuweisen. Dazu einige Beispiele:

- Verwenden Sie den Variablennamen `fieldWidth` und `fieldHeight` anstatt `x` und `y`.
- Verwenden Sie den Formentwurfsobjektnamen `clientName` anstatt `name`.

  HINWEIS: *Bei Variablennamen ist die korrekte Groß-/Kleinschreibung wichtig; sie sollten außerdem keine Leerzeichen enthalten.*

7.2. So definieren Sie Textvariablen

1) Wählen Sie „Datei“ > „Formulareigenschaften“.

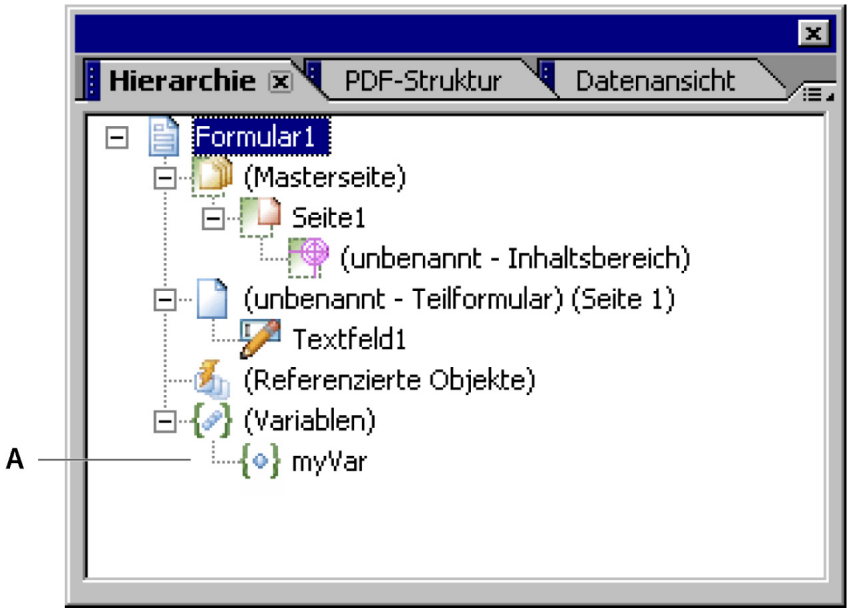
2) Klicken Sie auf der Registerkarte „Variablen“ auf „Neu (Einfügen)“ .

3) Geben Sie in der Liste „Variablen“ einen eindeutigen Namen für die Variable ein und drücken Sie die Eingabetaste. Bei Variablennamen ist die korrekte Groß-/Kleinschreibung wichtig; sie sollten außerdem keine Leerzeichen enthalten.

4) Klicken Sie einmal in das Feld auf der rechten Seite und geben Sie den Text ein, den Sie der Variablen zuweisen möchten.

   Die Variable erscheint in der Palette „Hierarchie“ auf der Formularebene.

A. *Neue Formularvariable*



7.3. So definieren Sie Textvariablen

1) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.

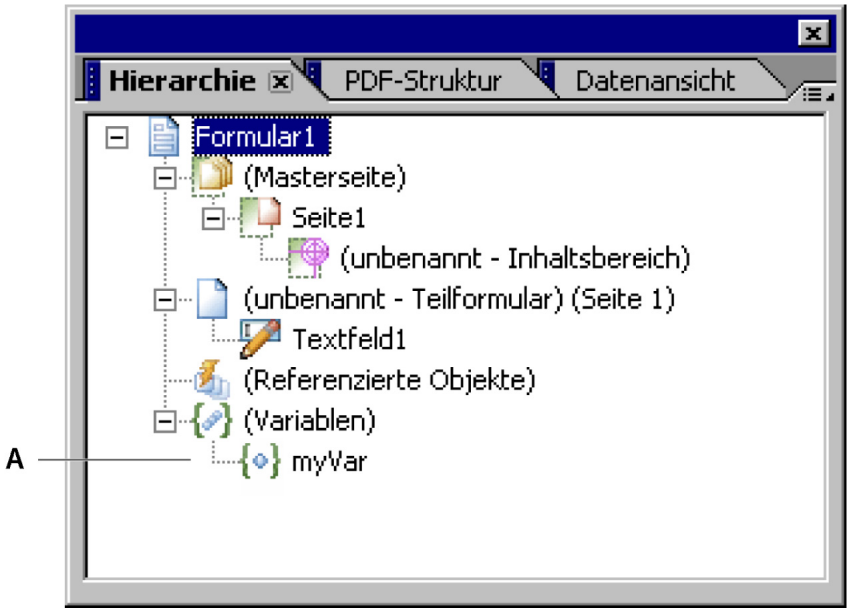
2) Klicken Sie auf der Registerkarte „Variablen“ auf „Neu (Einfügen)“ .

3) Geben Sie in der Liste „Variablen“ einen eindeutigen Namen für die Variable ein und drücken Sie die Eingabetaste. Bei Variablennamen ist die korrekte Groß-/Kleinschreibung wichtig; sie sollten außerdem keine Leerzeichen enthalten.

4) Klicken Sie einmal in das Feld auf der rechten Seite und geben Sie den Text ein, den Sie der Variablen zuweisen möchten.

Die Variable erscheint in der Palette „Hierarchie“ auf der Formularebene.

A. *Neue Formularvariable*



7.4. So zeigen Sie die Definition einer Textvariablen an

1) Wählen Sie „Datei“ > „Formulareigenschaften“.

2) Klicken Sie auf die Registerkarte „Variablen“ und wählen Sie die Variable in der Variablenliste aus. Der zugehörige Text wird in dem Feld auf der rechten Seite angezeigt.

7.5. So zeigen Sie die Definition einer Textvariablen an

1) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.

2) Klicken Sie auf die Registerkarte „Variablen“ und wählen Sie die Variable in der Variablenliste aus. Der zugehörige Text wird in dem Feld auf der rechten Seite angezeigt.

7.6. So löschen Sie Textvariablen

1) Wählen Sie „Datei“ > „Formulareigenschaften“.

2) Wählen Sie auf der Registerkarte „Variablen“ die Variable und klicken Sie auf „Löschen“ .

7.7. So löschen Sie Textvariablen

1) Wählen Sie „Bearbeiten“ > „Formulareigenschaften“.

2) Wählen Sie auf der Registerkarte „Variablen“ die Variable und klicken Sie auf „Löschen“ .

7.8. Variablen in Berechnungen und Skripten verwenden

Nach dem Erstellen von Formularvariablen genügt es, den Variablennamen in Ihren Berechnungen und Skripten zu referenzieren, um den Wert der Variablen zu erhalten.

WICHTIG: *Bei der Benennung von Variablen sollten Sie Namen vermeiden, die mit den Namen von Eigenschaften, Methoden oder Objekten des XML Form Object Model identisch sind.*

Weitere Informationen zu den Eigenschaften, Methoden und Objekten des XML Form Object Model finden Sie in der Skriptreferenz.

Erstellen Sie beispielsweise die folgenden Definitionen für Formularvariablen.

Variablenname	Wert
firstName	Tony
lastName	Blue
age	32

In FormCalc können Sie auf die Variablenwerte auf die gleiche Weise zugreifen wie auf Feld- und Objektwerte. In diesem Beispiel werden die Werte drei getrennten Feldern zugewiesen:

```
TextField1 = firstName
TextField2 = lastName
NumericField1 = age
```

Sie können Variablen auf die gleiche Weise auch in FormCalc-Funktionen verwenden, wie dieses Beispiel zeigt.

```
Concat( "Dear ", firstName, lastName )
```

In JavaScript referenzieren Sie Variablenwerte mit der `.value` Eigenschaft anstatt der `.rawValue` -Eigenschaft, die für Feld- und Objektwerte vorgesehen ist. Dazu ein Beispiel:

```
TextField1.rawValue = firstName.value;
```

HINWEIS: *Wenn Sie Formularvariablen mit Skripts in XFA-Formularen verwenden und ändern, zeigt die Dokumentmeldungsleiste in Acrobat und Adobe Reader eventuell eine Warnung bezüglich des Signaturvalidierungsstatus an. Diese weist darauf hin, dass die Gültigkeit der Signatur aufgrund nachfolgender Änderungen am Dokument unbekannt ist.*

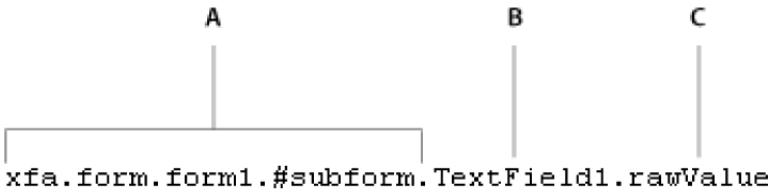
8. Objekte in Berechnungen und Skripten referenzieren

FormCalc-Berechnungen und JavaScript-Skripten unterliegen bei der Strukturierung von Code zwar jeweils spezifischen Regeln, aber wenn es um den Zugriff auf Formularobjekteigenschaften und -werte geht, stützen sie sich auf die gleiche Referenz-Syntax. Das XML-Formularobjektmodell bietet über eine Namenskonvention, bei der alle Objekte, Eigenschaften und Methoden jeweils durch ein Punkt (.) Zeichen voneinander getrennt sind, eine strukturierte Möglichkeit, auf Objekteigenschaften und -werte zuzugreifen.

Jede Referenzsyntax weist in der Regel eine in folgende Abschnitte unterteilte Struktur auf:

- Die Namen der übergeordneten Objekte in der Formularhierarchie, die zur Navigation zu einem bestimmten Feld oder Objekt dient. Mit den beiden Paletten „Hierarchie“ und „Datenansicht“ können Sie die Position eines Objekts im Verhältnis zu anderen Objekten im Formular und in zugehörigen Daten ermitteln.
- Der Name des Objekts, das Sie referenzieren möchten.
- Der Name der Eigenschaft oder Methode, auf die Sie zugreifen möchten. Dieser Abschnitt enthält möglicherweise auch Objekte des XML-Formularobjektmodells, die in der Struktur vor der Eigenschaft oder Methode auftreten, aber in der Palette „Hierarchie“ nicht als Objekte aufgeführt werden.

In der folgenden Abbildung sehen Sie die Referenz-Syntax für den Zugriff auf den Wert eines Textfelds in einem Formularentwurf. Dabei gelten die folgenden Konventionen für die Objektbenennung:



A.

Formularhierarchie-Objekte

B.

Objektname

C.

Name der Eigenschaft oder Methode

HINWEIS: *Das Teilformularobjekt, welches die erste Seite eines neuen Formulars darstellt, ist standardmäßig unbenannt. In der obigen Referenzsyntax wird das unbenannte Teilformular als* `#subform` *bezeichnet.*

Die Notationsstruktur der Referenz-Syntax hängt immer von der spezifischen Situation ab. Beispielsweise ist eine vollständig qualifizierte Referenz-Syntax für alle Situationen geeignet. In einigen Fällen können Sie die Syntax aber durch eine verkürzte Referenz-Syntax oder einen Referenz-Syntax-Kurzbefehl etwas übersichtlicher gestalten.

8.1. Objekteigenschaften und -werte referenzieren

Für die Referenz-Syntax, über die Sie auf Objekteigenschaften und -werte zugreifen bzw. diese ändern, stehen zwei Varianten zur Auswahl:

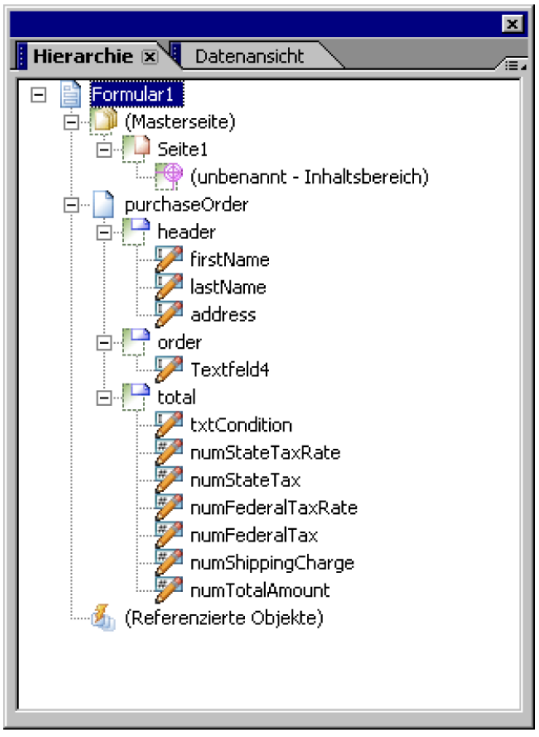
Vollständig qualifiziert

Die Referenzsyntax enthält die gesamte Objekthierarchie, angefangen mit dem `xfa` Stammknoten. Die vollständig qualifizierte Syntax greift präzise auf die Eigenschaft oder den Wert eines Objekts zu, unabhängig davon, wo sich die Berechnung oder das Skript befindet, die bzw. das die Referenz-Syntax enthält.

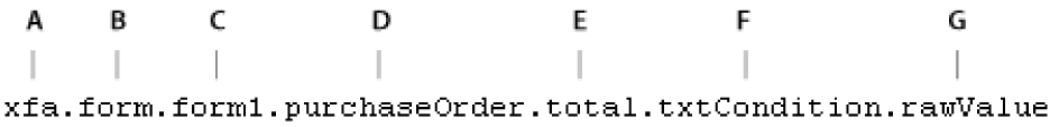
Abgekürzt

Die Referenz-Syntax ist entweder wegen der relativen Position der Berechnung oder des Skripts, die bzw. das die Referenz-Syntax und die Objektsyntaxreferenzen enthält, abgekürzt, oder weil Kurzbefehle verwendet werden. Eine abgekürzte Referenz-Syntax lässt sich zwar schneller erstellen, sie hat aber auch den Nachteil, dass sie nur so lange funktioniert, wie die Position der Objekte im Verhältnis zueinander unverändert bleibt.

In der folgenden Abbildung wird die Hierarchie eines Musterbestellformulars dargestellt.



Die Abbildung zeigt eine vollständig qualifizierte Referenzsyntax für FormCalc und JavaScript, die zum Zugriff auf den Wert des `txtCondition` Felds dient. Diese Referenz-Syntax könnte als Teil einer Berechnung oder eines Skriptes für ein beliebiges Objekt des Formulars eingesetzt werden.



A.

Stamm-Node

B.

Modell

C.

Stamm-Node des Formularentwurfs

D.

Seitenobjekt

E.

Name des Teilformulars

F.

Objektname

G.

Name der Eigenschaft oder Methode

HINWEIS: *Obwohl die Referenz-Syntax sowohl von FormCalc als auch von JavaScript verwendet wird, müssen Sie die für die beiden Skriptsprachen geltenden Konventionen beachten. Beispielsweise kann die im obigen Beispiel angegebene Referenzsyntax ohne Änderungen in FormCalc eingesetzt werden. Für JavaScript müssten Sie jedoch ein Semikolon (;) am Ende einfügen.*

Wenn sich zwei Objekte im selben Container befinden, z.B. in einem Teilformular, haben sie denselben Kontext. Wenn Objekte in demselben Kontext vorkommen, können Sie eine abgekürzte Referenz-Syntax verwenden, die sich nur aus dem Namen des Objekts gefolgt von der Eigenschaft oder Methode, auf die Sie zugreifen möchten, zusammensetzt. Beim obigen Beispiel würde die folgende abgekürzte Referenzsyntax auf den Wert des `txtCondition` Felds in allen Feldern im `total` Unterformular zugreifen:

```
txtCondition.rawValue
```

Wenn sich zwei Objekte in unterschiedlichen Containern befinden, haben sie nicht denselben Kontext. In diesem Fall können Sie zwar ein abgekürzte Referenz-Syntax verwenden, aber die Syntax muss mit dem Namen des obersten Container-Objekts beginnen, welches die beiden Objekte nicht gemein haben. Bei der obigen Hierarchie würde die folgende abgekürzte Referenzsyntax auf den Wert des `address` Felds vom `txtCondition` Feld aus zugreifen:

```
header.address.rawValue
```

Aufgrund der Struktur des XML Form Object Model treten einige Objekteigenschaften und -methoden bei untergeordneten Objekten der Objekte im Formular auf. Diese untergeordneten Objekte kommen nur als Teil des XML-Formularobjektmodells und nicht in den beiden Paletten „Hierarchie“ und „Datenansicht“ vor. Wenn Sie auf diese Eigenschaften und Methoden zugreifen möchten, müssen Sie die untergeordneten Objekte in die Referenz-Syntax einschließen. Beispielsweise wird über die folgende Referenzsyntax die QuickInfo für das `txtCondition` Feld festgelegt:

```
txtCondition.assist.toolTip.value = "Conditions of purchase." // FormCalc
txtCondition.assist.toolTip.value = "Conditions of purchase."; // JavaScript
```

Weitere Informationen zu den Formularobjektmodellen und ihrer Struktur finden Sie unter Skriptreferenz bezeichnet.

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

Unbenannte und wiederholte Objekte referenzieren

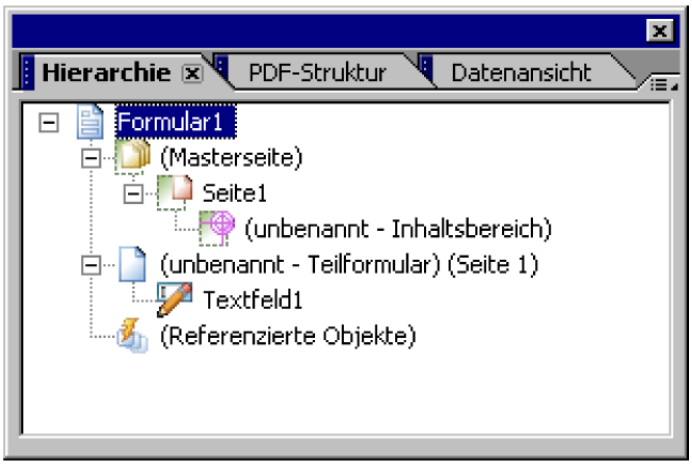
Aktuelles Objekt referenzieren

Referenz-Syntax-Kurzbefehle für FormCalc

8.2. Unbenannte und wiederholte Objekte referenzieren

In Designer können Sie sowohl unbenannte Objekte als auch mehrere gleichnamige Objekte erstellen. Ferner können Sie Berechnungen und Skripte erstellen, um auf Eigenschaften und Werte unbenannter Objekte zuzugreifen und diese zu ändern, indem Sie das Nummernzeichen (#) und Objektwerte für das Vorkommen mit den eckigen Klammern (`[ ]`) verwenden. FormCalc interpretieert das Nummernzeichen (#) und Zeichen in eckigen Klammern `[ ]`) korrekt, aber JavaScript kann das nicht. Um auf den Wert eines Textfelds in einer Situation zuzugreifen, in der das Nummernzeichen (#) oder die eckigen Klammern (`[ ]`) mit JavaScript vorkommen, müssen Sie unter Verwendung von JavaScript die `resolveNode` -Methode in Kombination mit einer vollständig qualifizierte Referenzsyntax oder einer abgekürzten Referenzsyntax verwenden.

Wenn Sie beispielsweise ein neues leeres Formular erstellen, ist der Name des Teilformulars, welches die Seite des Formulars darstellt, standardmäßig ein unbenanntes Teilformular mit dem Vorkommenswert von `0`. In der folgenden Abbildung sehen Sie die Formularhierarchie bei einem neuen Formular mit standardmäßiger Objektbenennung.



Das unbenannte Teilformular, welches die erste Seite des Formulars darstellt, hat die Vorkommensnummer `0`. In dieser Situation greifen die beiden folgenden Referenz-Syntaxen auf den Wert des Textfelds zu, das in der Formularhierarchie über einem neuen Formular liegt und für das standardmäßige Namenskonventionen gelten:

```
xfa.form.form1.#subform.TextField1.rawValue
xfa.form.form1.#subform[0].TextField1.rawValue
```

HINWEIS: *Wenn für ein Objekt kein Vorkommenswert angegeben wurde, greift die Referenz-Syntax standardmäßig auf das erste Vorkommen dieses Objekts zu.*

FormCalc erkennt die oben angegebene vollständig qualifizierte Referenz-Syntax und wertet sie korrekt aus. Wenn Sie mit JavaScript auf denselben Wert zugreifen möchten, müssen Sie eine der folgenden Varianten der `resolveNode` -Skriptmethode verwenden:

```
xfa.resolveNode("xfa.form.form1.#subform.TextField1").rawValue;
xfa.resolveNode("xfa.form.form1.#subform[0].TextField1").rawValue;
```

Wenn Sie einem Formular eine neue Seite hinzufügen, ist das Teilformular, welches die neue Seite darstellt, standardmäßig unbenannt. Der Vorkommenswert des neuen Teilformulars wird allerdings auf `1`eingestellt. Sie können das neue unbenannte Teilformular mit einer ähnlichen Referenz-Syntax wie der obigen angeben:

```
xfa.form.form1.#subform[1].TextField1.rawValue // FormCalc
xfa.resolveNode("xfa.form.form1.#subform[1].TextField1").rawValue;
// JavaScript
```

HINWEIS: *Zu den im Skript-Editor verfügbaren Anweisungsende-Optionen gehören unbenannte Objekte am Anfang der Liste. Objekte mit mehreren Vorkommenswerten werden in der Liste nur einmal aufgeführt. Der Listeneintrag bezieht sich auf das erste Vorkommen des Objekts. Wenn Sie anstelle des ersten Vorkommenswerts einen anderen Vorkommenswert abrufen möchten, müssen Sie diesen Wert manuell der Referenz-Syntax hinzufügen.*

Sie können die `resolveNode` -Methode verwenden, um Objekte innerhalb anderer Referenz-syntax-Anweisungen zureferenzieren. Dadurch kann der Skripterstellungsaufwand, der zum Referenzieren eines bestimmten Objekts, einer bestimmten Eigenschaft oder einer bestimmten Methode erforderlich ist, erheblich reduziert werden. Beispielsweise ließe sich die Referenz-Syntax, die auf ein Textfeld auf der zweiten Formularseite verweist, auf die folgende Anweisung kürzen:

```
    xfa.form.form1.resolveNode("#subform[1].TextField1").rawValue;
// JavaScript
```

Weitere Informationen zur `resolveNode` -Methode finden Sie unter resolveNode.

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

Objekteigenschaften und -werte referenzieren

Aktuelles Objekt referenzieren

Referenz-Syntax-Kurzbefehle für FormCalc

8.3. Aktuelles Objekt referenzieren

Wenn Sie die Eigenschaften oder Werte des aktuellen Objekts mit Hilfe von Berechnungen oder Skripten ändern möchten, die an das Objekt selbst angehängt sind, bieten sowohl FormCalc als auch JavaScript eindeutige Kurzbefehle, mit denen sich der Umfang der Referenz-Syntax verringern lässt. In FormCalc wird das Nummernzeichen (`$`) verwendet, um das aktuelle Objekt zu kennzeichnen und JavaScript verwendet hierzu das Schlüsselwort `this`.

Beispielsweise gibt die folgende Referenz-Syntax den Wert des aktuellen Objekts zurück:

```
$ // FormCalc
this.rawValue // JavaScript
```

Sie können die Verknüpfung für das Dollar-Zeichen (`$`) verwenden und den Suchbegriff `this`, um den Namen des aktuellen Objekts zu ersetzen, wenn auf Objekteigenschaften in Berechnungen und Skripten zugegriffen wird. Beispielsweise wird über die folgende Referenz-Syntax die mit dem aktuellen Objekt verbundene QuickInfo ersetzt:

```
$.assist.toolTip.value = „Das ist ein QuickInfo-Text.“ // FormCalc
this.assist.toolTip.value = „Das ist ein QuickInfo-Text.“; // JavaScript
```

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

Objekteigenschaften und -werte referenzieren

Unbenannte und wiederholte Objekte referenzieren

Referenz-Syntax-Kurzbefehle für FormCalc

8.4. Referenz-Syntax-Kurzbefehle für FormCalc

Mit Hilfe einer Reihe von Kurzbefehlen lassen sich Referenzen in FormCalc bequemer erstellen, wodurch der Zugriff auf Objekteigenschaften und -werte erleichtert wird. Dieser Abschnitt beschreibt die Referenzsyntax-Kurzbefehle für FormCalc.

8.4.1. Aktuelles Feld oder Objekt

Bezeichnet das aktuelle Feld oder Objekt

Notation

```
$
```

Beispiel

```
$ = "Tony Blue"
```

Das obige Beispiel setzt den Wert des aktuellen Feldes oder Objekts auf `Tony Blue`.

8.4.2. Stamm-Node des Datenmodells xfa.datasets.data

Bezeichnet die Stamm-Node des Datenmodells `xfa.datasets.data`.

Notation

```
$data
```

Beispiel

```
$data.purchaseOrder.total
```

entspricht

```
xfa.datasets.data.purchaseOrder.total
```

8.4.3. Formularobjekt-Ereignis

Bezeichnet das aktuelle Formularobjekt-Ereignis.

Notation

```
$event
```

Beispiel

```
$event.name
```

entspricht

```
xfa.event.name
```

Weitere Informationen finden Sie in Mit dem Ereignismodell arbeiten.

8.4.4. Stamm-Node des Formularmodells

Bezeichnet die Stamm-Node des Formularmodells `xfa.form`.

Notation

```
$form
```

Beispiel

```
$form.purchaseOrder.tax
```

entspricht

```
xfa.form.purchaseOrder.tax
```

8.4.5. Host-Objekt

Bezeichnet das Host-Objekt.

Notation

```
$Host
```

Beispiel

```
$Host.messageBox(„Hallo Welt“)
```

entspricht

```
xfa.host.messageBox(„Hallo Welt“)
```

Weitere Informationen unter Mit Host-Anwendungen arbeiten.

8.4.6. Stamm-Node des Layoutmodells

Bezeichnet den Stammknoten des Layoutmodells`xfa.layout`.

Notation

```
$layout
```

Beispiel

```
$layout.ready
```

entspricht

```
xfa.layout.ready
```

8.4.7. Datensatz aus einer Zusammenstellung von Daten

Bezeichnet den aktuellen Datensatz aus einer Zusammenstellung von Daten, z. B. aus einer XML-Datei.

Notation

```
$record
```

Beispiel

```
$record.header.txtOrderedByCity
```

verweist auf den `txtOrderedByCity` -Knoten im Kopfzeilenknoten der aktuellen XML-Daten.

8.4.8. Stamm-Node des Vorlagenmodells

Bezeichnet die Stamm-Node des Vorlagenmodells xfa.template.

Notation

```
$template
```

Beispiel

```
$template.purchaseOrder.item
```

entspricht

```
xfa.template.purchaseOrder.item
```

8.4.9. Stamm-Node des Datenmodells xfa.datasets

Bezeichnet die Stammknoten des Datenmodells`xfa.datasets`.

Notation

```
!
```

Beispiel

```
!data
```

entspricht

```
xfa.datasets.data
```

8.4.10. Alle Formularobjekte auswählen

Wählt alle Formularobjekte innerhalb eines gegebenen Containers, z. B. in einem Teilformular, unabhängig von ihrem Namen bzw. alle Objekte mit einem ähnlichen Namen aus.

Sie können die Syntax „*“ (Sternchen) mit JavaScript verwenden, wenn es mit der resolveNode -Methode verwendet wird.

Notation

```
*
```

Beispiel

Beispielsweise wählt der folgende Ausdruck alle Objekte mit dem Namen `item` in einem Formular:

```
xfa.form.form1.item[*]
```

8.4.11. Nach Objekten suchen, die Teil eines Unter-Containers sind

Sie können an jeder Stelle in Ihrer Referenzsyntax zwei Punkte einsetzen, um nach Objekten zu suchen, die Teil eines Unter-Containers des aktuellen Container-Objekts (z. B. eines Teilformulars) sind.

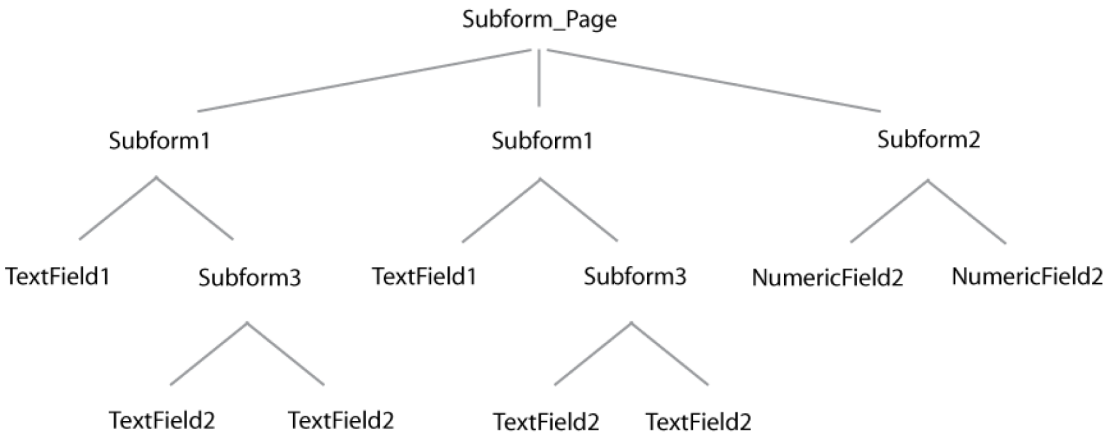
Sie können die Syntax '..' (zwei Punkte) mit JavaScript verwenden, wenn es mit der resolveNode -Methode verwendet wird.

Notation

```
..
```

Beispiel

Der Ausdruck `Subform_Page..Subform2` heißt, den Knoten `Subform_Page` (wie üblich) zu suchen und ein untergeordnetes Element von `Subform_Page` mit dem Namen `Subform2` aufzurufen.



Aus dem obigen Beispiel-Baumdiagramm ergibt sich:

```
Subform_Page..TextField2
```

entspricht

```
Subform_Page.Subform1[0].Subform3.TextField2[0],
```

because `TextField2[0]` ist im ersten `Subform1` -Knoten, auf den FormCalc bei der Suche trifft. Ein zweites Beispiel:

```
Subform_Page..Subform3[*]
```

gibt alle vier `TextField2` Objekte zurück.

8.4.12. Unbenannte Objekte bezeichnen oder Eigenschaften angeben

Das Nummernzeichen (#) dient zur Bezeichnung eines der folgenden Elemente in einer Referenzsyntax:

- Ein unbenanntes Objekt
- Geben Sie in einer Referenzsyntax eine Eigenschaft an, wenn eine Eigenschaft und ein Objekt den gleichen Namen tragen

Sie können die Syntax mit Raute (#) mit JavaScript verwenden, wenn die resolveNode -Methode verwendet wird.

Notation

```
#
```

Beispiel

Beispielsweise greift die folgende Referenzsyntax auf ein unbenanntes Teilformular zu:

```
xfa.form.form1.#subform
```

Die folgende Referenzsyntax greift auf die `name` -Eigenschaft eines Teilformulars zu, wenn das Teilformular auch ein Feld mit dem Namen `name` enthält:

```
xfa.form.form1.#Teilformular.#name
```

8.4.13. Wert für das Vorkommen eines Objekts

Die eckige Klammer (`[ ]`) gibt den Wert für das Vorkommen eines Objekts an.

In sprachspezifischen Formularen für Arabisch, Hebräisch, Thailändisch und Vietnamesisch befindet sich die Referenzsyntax grundsätzlich rechts (auch bei Sprachen, die von rechts nach links geschrieben werden).

Notation

```
[ ]
```

Beispiel

Um eine Referenz mit einem solchen Wert zu erstellen, platzieren Sie eckicke Klammern (`[ ]`) hinter einen Objektnamen und schließen Sie in den eckigen Klammern einen der folgenden Werte ein:

- `[ n ]`, wobeineine absolute Vorkommens-Indexnummer ist (der Index beginnt bei 0). Wenn eine Vorkommensnummer außerhalb des zulässigen Bereichs liegt, wird kein Wert zurückgegeben. Beispiel,

  ```
  xfa.form.form1.#subform.Quantity[3]
  ```

  referenziert das vierte Vorkommen des Objekts „Quantity“.

- `[ +/- n ]`, wobei `n` ein Vorkommen relativ zum Vorkommen des Objekts ist, von welchem die Referenz ausgeht. Positive Werte liefern höhere Vorkommensnummern und negative Werte liefern niedrigere. Beispiel,

  ```
  xfa.form.form1.#subform.Quantity[+2]
  ```

  Diese Referenz liefert das Vorkommen von „Quantity“, dessen Vorkommensnummer um 2 höher ist als die Vorkommensnummer des Containers, von dem die Referenz ausgeht. Stünde diese Referenz beispielsweise mit dem Objekt „Quantity[2]“ in Verbindung, so wäre sie gleichbedeutend mit:

  ```
  xfa.template.Quantity[4]
  ```

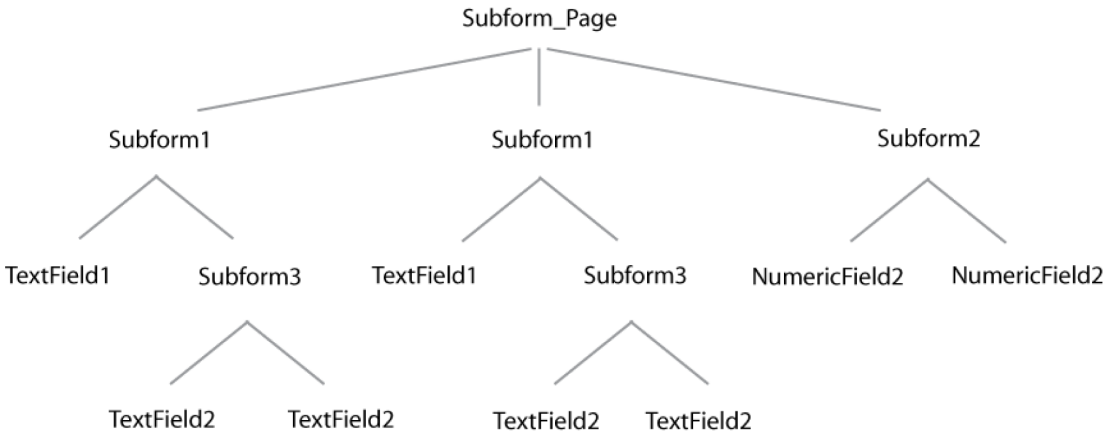
  Wenn die berechnete Indexnummer außerhalb des zulässigen Bereichs liegt, gibt die Referenz einen Fehler zurück.

  Die häufigste Anwendung dieser Syntax ist die Ansteuerung des vorigen oder nächsten Vorkommens eines bestimmten Objekts. Beispielsweise könnte man bei jedem Vorkommen des Objekts „Quantity“ (außer beim ersten) mit „Quantity[-1]“ den Wert des vorigen „Quantity“-Objekts abrufen.

- `[*]` gibt mehrere Vorkommen eines Objekts an. Das erste benannte Objekt wird gefunden und die mit dem ersten Objekt verwandten Objekte desselben Namens werden zurückgegeben. Beachten Sie, dass bei dieser Notation eine Zusammenstellung von mehreren Objekten zurückgegeben wird. Beispiel,

  ```
  xfa.form.form1.#subform.Quantity[*]
  ```

- Dieser Ausdruck bezieht sich auf alle Objekte mit dem Namen `Menge` die gleichgestellte Elemente des ersten Vorkommens von `Menge ist` gefundenen von der Referenz.



In dem gezeigten Baumdiagramm geben diese Ausdrücke jeweils die folgenden Objekte zurück:

- `Subform_Page.Subform1[*]` gibt beide `Subform1` -Objekte zurück.
- `Subform_Page.Subform1.Subform3.TextField2[*]` gibt beide `TextField2` -Objekte zurück. `Subform_Page.Subform1` wird in das erste `Subform1` -Objekt auf der linken Seite aufgelöst, und `TextField2[*]` wird relativ zum `Subform3` -Objekt evaluiert.
- `Subform_Page.Subform1[*].TextField1` gibt beide `TextField1` Instanzen zurück. `Subform_Page.Subform1[*]` löst beide `Subform1` -Objekte auf, und `TextField1` evaluiert relativ zu den `Subform1` -Objekten.
- `Subform_Page.Subform1[*].Subform3.TextField2[1]` gibt das zweite und vierte `TextField2` -Objekt von links zurück. `Subform_Page.Subform1[*]` löst beide `Subform1` -Objekte auf, und `TextField2[1]` evaluiert relativ zu den `Subform3` -Objekten.
- `Subform_Page.Subform1[*].Subform3[*]` gibt beide Instanzen des `Subform3` -Objekts zurück.
- `Subform_Page.*` gibt beide `Subform1` Objekte und das `Subform2` -Objekt zurück.
- `Subform_Page.Subform2.*` gibt die beiden Instanzen des `NumericField2` -Objekts zurück.
- Sie können die `[ ]` '(mit eckigen Klammern) Syntax in Verbindung mit JavaScript verwenden, wenn die resolveNode -Methode verwendet wird.
-

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

9. JavaScript-Funktionen erstellen und wiederverwenden

Das *Skriptobjekt* ist ein Objekt, in dem Sie JavaScript-Funktionen und -Werte getrennt von konkreten Formularobjekten speichern können. Üblicherweise verwenden Sie das Skriptobjekt zur Erstellung von benutzerdefinierten Funktionen und Methoden, die Sie als Teil von JavaScript-Skripten an mehreren Stellen in Ihrem Formular verwenden möchten. Dadurch wird der Arbeitsaufwand, der bei der Skripterstellung für sich wiederholende Aktionen normalerweise anfällt, erheblich reduziert.

Das Skriptobjekt unterstützt nur Skripten, die in JavaScript geschrieben wurden. Hinsichtlich der Stellen, an welchen die Skripten ausgeführt werden, gibt es jedoch keine Einschränkungen. Es muss lediglich gewährleistet sein, dass die Skriptsprache für das Ereignis, welches das Skriptobjekt auslöst, auf JavaScript eingestellt ist.

Im Allgemeinen verarbeiten Acrobat und Forms die Skripterstellung eines Skriptobjekts auf die gleiche Weise, es gibt jedoch auch Unterschiede.

Nur Skripten, die für die Ausführung auf dem Client konfiguriert sind, können Skriptobjekte nutzen, die für die Ausführung auf dem Client konfiguriert sind, und umgekehrt.

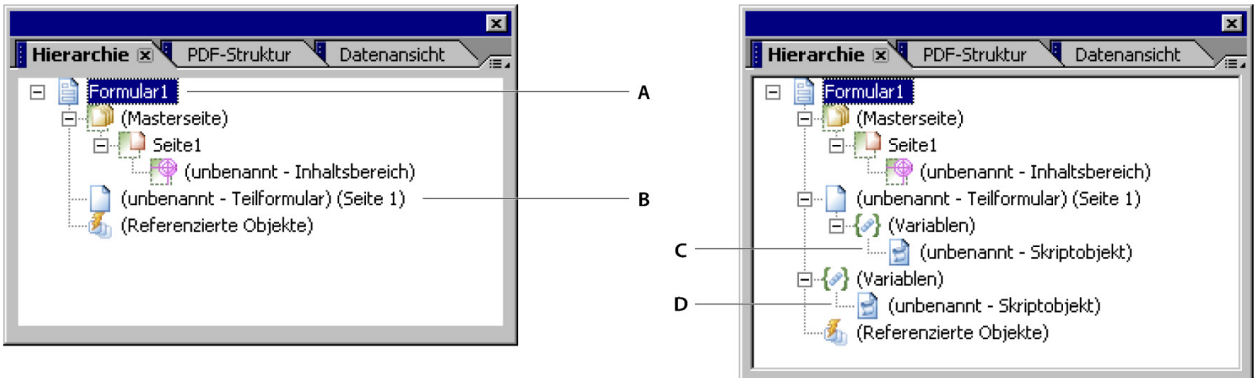
9.1. So erstellen Sie ein Skriptobjekt

Die Erstellung eines Skriptobjekts erfolgt in zwei Phasen. Die erste Phase besteht darin, das Objekt selbst dem Formularentwurf hinzuzufügen; die zweite Phase ist die eigentliche Erstellung des Skripts, das Sie im Skriptobjekt speichern möchten.

1) Erstellen Sie ein neues Formular oder öffnen Sie ein vorhandenes.

2) Klicken Sie in der Palette „Hierarchie“ mit der rechten Maustaste auf ein Objekt auf Formularebene bzw. auf Teilformularebene und wählen Sie „Skriptobjekt einfügen“.

A. Formularebenenobjekt **B.** Teilformularebenenobjekt **C.** Teilformularebenenskriptobjekt **D.** *Formularebenenskriptobjekt*



3) (Optional) Klicken Sie mit der rechten Maustaste auf das Skriptobjekt und wählen Sie „Objekt umbenennen“.

9.2. So fügen Sie einem Skriptobjekt Skripten hinzu

Nachdem Sie Ihr Formular mit einem Skriptobjekt versehen haben, können Sie mit dem Skript-Editor Skripten hinzufügen.

1) Wählen Sie in der Palette „Hierarchie“ das Skriptobjekt „“ aus.

   Der Skript-Editor wird angezeigt. Die Liste „Anzeigen“ enthält den Wert „Skriptobjekt“ und die Liste „Sprache“ den Wert „JavaScript“. Sie können keinen dieser Werte ändern.

2) Geben Sie im Feld "Skriptquelle" Ihr Skript ein.

3) Zum Testen des Formulars klicken Sie auf die Registerkarte „PDF-Vorschau“.

9.2.1. Beispiel

Erstellen Sie z. B. ein Skriptobjekt namens `Feedback`, die folgende Funktion enthält:

```
function emptyCheck(oField) {

if ((oField.rawValue == null) || (oField.rawValue == "")) { xfa.host.messa-
geBox("You must input a value for this field.", "Error Message", 3);
}
}
```

9.3. So referenzieren Sie JavaScript-Funktionen in einem Skriptobjekt

Nachdem Sie einem Skriptobjekt Skripten hinzugefügt haben, können Sie das Skriptobjekt von jedem Ereignis aus referenzieren, das JavaScript-Skripten unterstützt.

1) Wählen Sie ein Objekt im Formular und anschließend unter „Anzeigen“ ein Ereignis aus.

2) Erstellen Sie eine Referenz zum Skriptobjekt und zu beliebigen Funktionen im Skriptobjekt. In der folgenden generischen Syntax wird davon ausgegangen, dass sich das Objekt, von dem aus das Skriptobjekt referenziert wird, in der Formularhierarchie auf derselben Ebene wie das Skriptobjekt befindet bzw. dass sich das Skriptobjekt in der Formularhierarchie auf der obersten Ebene befindet.

   ```
   script_object.function_name(parameter1, ...);
   ```

3) Wenden Sie das neue Skript auf das Formularobjekt an und testen Sie es durch Anzeigen einer Formularvorschau auf der Registerkarte „PDF-Vorschau“.

Ähnlich wie beim Referenzieren anderer Objekte auf einem Formular müssen Sie beim Referenzieren des Skriptobjekts eine gültige Syntax angeben, die auch die Position des Objekts innerhalb der Formularhierarchie beinhaltet. Weitere Informationen zum Referenzieren von Objekten bei der Skripterstellung finden Sie unter Objekteigenschaften und -werte referenzieren.

9.3.1. Beispiel

Verwenden Sie das Skriptobjektbeispiel aus Um ein Skript zu einem Skriptobjekt hinzuzufügen, platzieren Sie das folgende JavaScript-Skript auf das `exit` -Ereignis für ein Textfeld. Testen Sie das Formular durch Anzeige auf der Registerkarte „PDF-Vorschau“.

10. Skriptfragmente verwenden

Ein Skriptfragment enthält grundsätzlich ein Skriptobjekt. Ein Skriptobjekt enthält wiederverwendbare JavaScript-Funktionen oder Werte, die unabhängig von einem bestimmten Formularobjekt, z.B. einem Datumsparser oder einem Webdienstaufruf, gespeichert werden. Normalerweise verwenden Sie Skriptobjekte zur Erstellung von benutzerdefinierten Funktionen und Methoden, die Sie an mehreren Stellen in Ihrem Formular verwenden möchten. Dadurch wird der Arbeitsaufwand, der bei der Skripterstellung für sich wiederholende Aktionen normalerweise anfällt, erheblich reduziert.

Skriptfragmente enthalten nur Skriptobjekte, die als untergeordnete Objekte von Variablen auf der Palette „Hierarchie“ angezeigt werden. Fragmente können keine Skripten enthalten, die mit anderen Formularobjekten verbunden sind. Dies gilt z. B. für Ereignisskripten wie „validate“, „calculate“ oder „initialize“.

Sie können Skriptfragmente über die Palette „Hierarchie“ erstellen.

Skriptfragmente werden genau wie herkömmliche Fragmente bearbeitet.

10.1. Eigenschaften von Skriptfragmenten

Bei Auswahl eines Skriptfragments werden auf der Registerkarte „Skriptobjekt“ der Palette „Objekt“ die Eigenschaften des Skriptfragments angezeigt.

10.1.1. Quelldatei

Legt die Quelldatei für den Fragmentverweis fest. Diese Eigenschaft ist nur sichtbar, wenn das ausgewählte Objekt eine Fragmentverweis ist.

10.1.2. Fragmentname

Legt den Namen des Fragments fest. Sie können auf die Schaltfläche „Fragmentinformationen“ klicken, um die Fragmentinformationen anzuzeigen.

Diese Eigenschaft ist sichtbar, wenn ein Fragmentverweis bzw. ein in einer Quelldatei definiertes Fragment ausgewählt ist. Wenn es sich bei dem ausgewählten Objekt um einen Fragmentverweis handelt und keine Quelldatei angegeben wurde, wird diese Eigenschaft nicht angezeigt. Unter „Fragmentname“ werden alle in der angegebenen Quelldatei festgelegten Fragmente aufgeführt. Die Option „Benutzerdefiniert“ bietet eine direkte Unterstützung für das Festlegen eines SOM-Ausdrucks oder eines ID-Werts als Fragmentverweis sowie für die Implementierung in die XML Forms Architecture.

10.2. So erstellen Sie ein Skriptfragment

Sie können ein in mehreren Formularen wiederverwendbares Skriptfragment mit allgemeinen Funktionen erstellen. Zum Erstellen eines Skriptfragments erstellen Sie ein Skriptobjekt mit den entsprechenden wiederverwendbaren Funktionen. Ein Skriptfragment kann auch nur aus einem Skriptobjekt bestehen.

1) Erstellen Sie ein Skriptobjekt.

2) Klicken Sie in der Palette „Hierarchie“ mit der rechten Maustaste auf das Skriptobjekt und wählen Sie „Fragmente“ und dann „Fragment erstellen“.

   HINWEIS: *Sie können Skriptfragmente auch durch Ziehen des Skriptobjekts aus der Palette „Hierarchie“ in die Palette „Fragmentbibliothek“ erstellen.*

3) Zum Verwenden eines anderen Fragmentnamens geben Sie im Feld „Name“ den gewünschten Namen ein.

4) (Optional) Geben Sie im Feld „Beschreibung“ eine Beschreibung des Fragments ein.

5) Wählen Sie eine Methode zum Erstellen des Fragments:

   - Wenn Sie das Fragment in einer separaten XDP-Datei definieren möchten, die in der Fragmentbibliothek gespeichert wird, wählen Sie die Option „Neues Fragment in Fragmentbibliothek erstellen“. Wählen Sie in der Liste „Fragmentbibliothek“ die Fragmentbibliothek aus, in der Sie die Fragmentdatei speichern möchten. Wenn Sie einen anderen Dateinamen verwenden möchten, geben Sie im Feld „Dateiname“ den gewünschten Dateinamen für das Fragment ein. Wenn Sie die Auswahl nicht durch das neue Fragment ersetzen möchten, deaktivieren Sie das Kontrollkästchen „Auswahl durch Verweis auf neues Formularfragment ersetzen“.
   - Um das Fragment in der aktuellen Datei zu definieren, wählen Sie „Neues Fragment in aktuellem Dokument erstellen“.

6) Klicken Sie auf OK.

10.3. So fügen Sie ein Skriptfragment ein

Skriptfragmente bieten Ihnen die Möglichkeit, JavaScript-Funktionen in mehreren Formularen wiederzuverwenden. Beim Erstellen eines neuen Formularentwurfs fügen Sie einen Verweis auf ein vorhandenes Skriptfragment ein. Das Fragment wird dann im Formularentwurf angezeigt.

Fragmente können nicht in XFAF-Dokumente eingefügt werden.

HINWEIS: *Aktivieren Sie im Menü der Palette „Fragmentbibliothek“ die Option „Vorschaufenster einblenden“, um eine Vorschau der Fragmente in der Palette anzuzeigen.*

10.3.1. So fügen Sie ein Skriptfragment über die Palette „Fragmentbibliothek“ ein

1) Wählen Sie in der Fragmentbibliothek das Skriptfragment aus.
2) Ziehen Sie das Fragment auf ein Teilformular- oder Variablen-Objekt in der Palette „Hierarchie“.

10.3.2. So fügen Sie ein Skriptfragment über das Menü „Einfügen“ ein

1) Wählen Sie „Einfügen“ „Fragment“.
2) Rufen Sie die Datei mit dem Fragment auf.
3) Wählen Sie die Datei aus und klicken Sie auf „OK“. Das Fragment wird im Stammteilformular als dem Variablen-Objekt untergeordnetes Objekt angezeigt.

VERKNPFTE LINKS:

JavaScript-Funktionen erstellen und wiederverwenden

So erstellen Sie ein Skriptobjekt

Um ein Skript zu einem Skriptobjekt hinzuzufügen

11. Debugging von Berechnungen und Skripten

In Designer stehen gemäß der gewählten Skriptsprache verschiedene Funktionen und Techniken zum Debugging von Berechnungen und Skripten zur Verfügung.

Wenn Sie JavaScript-Sprachskripte debuggen möchten, können Sie den Befehl `alert` oder die `messageBox` -Methode verwenden, um Debugging-Feedback bereitzustellen. Ein Nachteil dieser Methode besteht darin, dass Sie viele Meldungsfelder schließen müssen. Die Anzeige eines Meldungsfelds kann zudem Unterschiede im Verhalten des Formulars zur Folge haben. Dies gilt insbesondere, wenn Sie ein Skript debuggen, das den Fokus auf ein Objekt im Formular festlegt. Am besten verwenden Sie `console.println`, um Text von Acrobat auf die JavaScript-Konsole auszugeben, um ein Formular zu debuggen.

11.1. Warn- und Prüfmeldungen in der Palette „Bericht“ von Designer

In der Palette „Bericht“ werden Warn- und Prüfmeldungen angezeigt, die Sie im Rahmen des Formularentwurfs beim Debugging unterstützen. Auf der Registerkarte „Warnungen“ finden Sie Fehler oder Meldungen, die von Designer während des Formularentwurfs erzeugt wurden. Folgende Fehler und Meldungen werden auf der Registerkarte „Protokoll“ angezeigt:

- Prüfmeldungen
- JavaScript- oder FormCalc-Skriptausführungsfehler
- Wiedergabefehler beim Entwurf, die beim Importieren oder Speichern eines Formulars oder bei der Vorschau eines Formulars auf der Registerkarte „PDF-Vorschau“ generiert werden

Weitere Informationen zur Verwendung der Palette „Bericht“ finden Sie unter Berechnungen und Skripte mit dem Arbeitsbereich debuggen.

11.2. Debugging-Feedback mit der messageBox-Methode bereitstellen

Mit der XML-Fomularobjektmodell `messageBox` -Methode können Sie die aus einem interaktiven Formular stammenden Informationen zur Laufzeit in einem Dialogfeld ausgeben. Sie können die XML-Fomularobjektmodell `messageBox` -Methode verwenden, um Meldungen oder Feldwerte zur Laufzeit anzuzeigen. Wenn initiiert, zeigt die `messageBox` -Methode in der Client-Anwendung in einem neuen Dialogfeld einen Zeichenfolgenwert an. Bei diesem Zeichenfolgenwert kann es sich um eine Textmeldung handeln, die Sie zu Debugging-Zwecken erstellen, oder um den Zeichenfolgenwert von Feldern oder Ausdrücken.

Angenommen, ein einfacher Formularentwurf enthält ein einzelnes numerisches Feld (NumericField1) und eine Schaltfläche (Button1). In diesem Fall geben die folgende FormCalc-Berechnung und das folgende JavaScript-Skript jeweils eine Meldung aus, die sich aus Text und dem im numerischen Feld gegenwärtig angezeigten Wert zusammensetzt. Durch das Hinzufügen der Berechnung oder des Skripts zum `click` -Ereignis des Schaltflächenobjekts, wird der Wert des numerischen Felds interaktiv in einem neuen Dialogfeld angezeigt, indem Sie auf die Schaltfläche klicken.

11.3. FormCalc

```
xfa.host.messageBox(Concat("Der Wert des NumericField1 lautet: ",
NumericField1), "Debugging", 3)
```

11.4. JavaScript

```
xfa.host.messageBox("Das Wert von NumericField1 lautet: " +
NumericField1.rawValue, "Debugging", 3);
```

WICHTIG: *Die `messageBox` -Methode gibt im Meldungsdialogfeld einen Ganzzahlwert für die Schaltfläche zurück, auf die der Benutzer beim Ausfüllen des Formulars klickt. Wenn Sie die `messageBox` -Methode an das `calculate` -Ereignis eines Feldobjekts anhängen und die `messageBox` -Methode die letzte Skriptzeile ist, gibt das Feld den Rückgabewert der `messageBox` -Methode zur Laufzeit an.*

Weitere Informationen zur Verwendung von `messageBox` finden Sie unter messageBox

11.5. Informationen in ein Textfeld ausgeben

Sie haben die Möglichkeit, Informationen wie Feldwerte oder Meldungen in ein Textfeld des Formularentwurfs auszugeben. Beispielsweise können Sie neue Meldungen oder Werte an den Wert eines Textfeldes anhängen und so ein Protokoll für die künftige Verwendung erstellen.

11.6. JavaScript-Debugging

Wenn Sie JavaScript-Sprache für ein Skript verwenden, können Sie die `console.println("string")` -Funktion verwenden, um Informationen auf auf der JavaScript-Konsole in Acrobat Professional auszugeben. Alternativ dazu können Sie die `alert` -Methode vom Acrobat-JavaScript-Objektmodell zum Debuggen von JavaScript verwenden.

11.6.1. JavaScript Debugger in Acrobat Professional

Testen Sie JavaScript-Skripten mit dem JavaScript-Debugger von Acrobat Professional. Der Debugger enthält die JavaScript-Konsole, mit der Sie auf der Registerkarte „PDF-Vorschau“ Teile des JavaScript-Codes testen können. Die JavaScript-Konsole bietet eine interaktive und bequeme Schnittstelle, mit der Sie Teile des JavaScript-Codes testen und mit anderen Objekteigenschaften und -methoden experimentieren können. Dank dieser interaktiven Funktionalität verhält sich die JavaScript-Konsole wie ein Editor, welcher die Ausführung einzelner Code-Zeilen oder Code-Blöcke unterstützt.

Wenn Sie den JavaScript-Debugger für Designer aktivieren und den Code von der JavaScript-Konsole ausführen möchten, aktivieren Sie in Acrobat Professional JavaScript und den JavaScript-Debugger.

HINWEIS: *Wenn Sie Acrobat Reader DC-Erweiterungen installiert haben, können Sie den JavaScript-Debugger in Adobe Reader aktivieren. Zum Aktivieren des JavaScript-Debuggers in Adobe Reader muss die Datei „debugger.js“ geöffnet und in der Microsoft Windows-Registrierung bearbeitet werden. Informationen zur Aktivierung des JavaScript-Debuggers in Adobe Reader erhalten Sie unter Entwickeln von Acrobat-Anwendungen mithilfe von JavaScript (Nur Englisch).*

11.6.2. So aktivieren Sie den JavaScript-Debugger für Designer

1) Starten Sie Designer.
2) Starten Sie Acrobat Professional.
3) In Acrobat Professional wählen Sie „Bearbeiten“ > „Voreinstellungen“.
4) Wählen Sie aus der linken Liste "JavaScript".
5) Sofern nicht bereits aktiviert, wählen Sie „Acrobat JavaScript aktivieren“.

6) Aktivieren Sie unter „JavaScript-Debugger“ die Option „JavaScript-Debugger nach dem Neustart von Acrobat aktivieren“.

7) Wählen Sie "Interaktive Konsole aktivieren". Mit dieser Option können Sie den in der JavaScript-Konsole geschriebenen Code auswerten.

8) Wählen Sie "Konsole bei Fehlern und Meldungen anzeigen". Mit dieser Option wird sichergestellt, dass die JavaScript-Konsole bei Falscheingaben hilfreiche Informationen anzeigt.

9) Klicken Sie auf „OK“, um das Dialogfeld „Grundeinstellungen“ zu schließen.

10) Beenden Sie Acrobat Professional.

11) Klicken Sie in Designer auf die Registerkarte „PDF-Vorschau“.

12) Drücken Sie die Tasten Strg+J, um den JavaScript-Debugger zu öffnen.

11.6.3. So verhindern Sie, dass der JavaScript-Debugger in Designer ausgeblendet wird

Wenn der JavaScript-Debugger von Acrobat aktiviert wurde und ausgeblendet wird, sobald Sie in Designer auf „Komponenten“ klicken, beenden Sie den Acrobat.exe-Prozess im Microsoft Windows Task-Manager. Der Acrobat.exe-Prozess wird auch nach Beendigung von Acrobat weiter ausgeführt, damit Acrobat bei einem Neustart schneller angezeigt werden kann. Durch Beenden des Prozesses wird auch die Verknüpfung zwischen dem JavaScript-Debugger und der Acrobat Professional-Sitzung beendet. Der JavaScript-Debugger kann dann in Designer eingesetzt werden.

1) Klicken Sie im Windows Task-Manager auf die Registerkarte "Prozesse".

2) Klicken Sie in der Spalte „Name“ mit der rechten Maustaste auf Acrobat.exe und wählen Sie „Prozess beenden“.

11.6.4. Code mit der JavaScript-Konsole auswerten

Sie haben drei Möglichkeiten, einzelne oder mehrere Code-Zeilen mit der JavaScript-Konsole von Acrobat auszuwerten.

11.6.5. So werten Sie einen Teil einer Code-Zeile aus

1) Markieren Sie den gewünschten Teil im Konsolenfenster und drücken Sie entweder die Eingabetaste auf der Zehnertastatur oder `Strg+Enter` auf der normalen Tastatur.

11.6.6. So werten Sie eine einzelne Code-Zeile aus

1) Platzieren Sie den Cursor in die gewünschten Zeile im Konsolenfenster und drücken Sie `Enter` auf der Zehnertastatur `Strg+Enter` auf der normalen Tastatur.

11.6.7. So werten Sie mehrere Code-Zeilen aus

1) Markieren Sie die Zeilen im Konsolenfenster und drücken Sie entweder die Eingabetaste auf der Zehnertastatur oder `Strg+Enter` auf der normalen Tastatur.

11.6.8. So löschen Sie in der JavaScript-Konsole angezeigte Inhalte

1) Klicken Sie im Konsolenfenster auf die Option zum Löschen.

   Das Ergebnis des zuletzt ausgewerteten JavaScript-Skriptes wird im Konsolenfenster angezeigt.

   Nach der Auswertung jedes JavaScript-Skripts druckt das Konsolenfenster `undefined` aus, welches der Wiedergabewert der Anweisung ist. Beachten Sie, dass das Ergebnis einer Anweisung nicht dem Wert eines Ausdrucks innerhalb der Anweisung entspricht. Der Rückgabewert `undefined` bedeutet nicht, dass der Wert des Skriptes nicht definiert („undefined“) ist, sondern dass der Rückgabewert der JavaScript-Anweisung nicht definiert ist.

11.6.9. Debugging-Feedback für die JavaScript-Konsole bereitstellen

Wenn Sie Skripte mit JavaScript erstellen, können Sie Meldungen von Acrobat an die JavaScript-Konsole zur Laufzeit ausgeben, indem Sie die `console.println` -Methode verwenden, die sich im JavaSkript-Objektmodell von Acrobat befindet. Wenn initiiert, zeigt die `console.println` einen Zeichenfolgenwert in der JavaScript-Konsole an. Bei diesem Zeichenfolgenwert kann es sich um eine Textmeldung handeln, die Sie zu Debugging-Zwecken erstellen, oder um den Zeichenfolgenwert von Feldern oder Ausdrücken.

Angenommen, ein einfacher Formularentwurf enthält ein einzelnes numerisches Feld (Numeric-Field1) und eine Schaltfläche (Button1). In diesem Fall gibt das folgende JavaScript-Skript eine Meldung aus, die sich aus Text und dem im numerischen Feld gegenwärtig angezeigten Wert zusammensetzt. Durch das Hinzufügen der Berechnung oder des Skripts zum `click` -Ereignis des Schaltflächenobjekts, wird der Wert des numerischen Felds interaktiv in einem neuen Dialogfeld angezeigt, indem Sie auf die Schaltfläche klicken.

```
console.println("Der Wert lautet " + NumericField1.rawValue);
```

Weitere Informationen zur `console.println` -Methode und dem Javascript-Objektmodell von Acrobat finden Sie unter Entwickeln von Acrobat-Anwendungen mithilfe von JavaScript (Nur Englisch).

Informationen zur JavaScript-Konsole sowie zum JavaScript-Debugger erhalten Sie unter Entwickeln von Acrobat-Anwendungen mithilfe von JavaScript (Nur Englisch).

11.6.10. Debugging-Feedback mit der alert-Methode bereitstellen

Wenn Sie beispielsweise ein Meldungsfeld während eines `calculate` -Ereignis zurückgeben wollen, können Sie die `alert` -Methode vom Javascript-Objektmodell von Acrobat nutzen. Das folgende Skript gibt zum Beispiel den Wert eines Textfelds zurück:

```
    var oField = xfa.resolveNode("TextField1").rawValue;
app.alert(oField);
```

Weitere Informationen zur `alert` -Methode und dem Javascript-Objektmodell von Acrobat finden Sie unter Entwickeln von Acrobat-Anwendungen mithilfe von JavaScript (Nur Englisch).

VERKNPFTE LINKS:

Berechnungen und Skripte mit dem Arbeitsbereich debuggen

11.7. Tipps zum Debugging

Beachten Sie beim Debugging von Berechnungen und Skripten die folgenden Tipps.

11.7.1. Musterdaten

Denken Sie daran, im Dialogfeld „Formulareigenschaften“ eine Vorschaudatendatei anzugeben. Hierdurch werden die Daten jedoch nicht in der endgültigen PDF-Datei gespeichert.

11.7.2. Masterseiten

Legen Sie für das Debugging von Masterseiten auf jeder Masterseite ein anderes Objekt ab, um festzustellen, welches Objekt angegeben wurde.

11.7.3. Erste Seite eines Formulars

Designer orientiert sich am Stammteilformular, um festzustellen, auf welcher Seite das Formular beginnt. Geht die erste Seite nicht aus dem Stammteilformular hervor, wird standardmäßig die erste Masterseite verwendet.

11.7.4. Inkrementelles Debugging

Beginnen Sie beim Debugging eines Formularentwurfs damit, nacheinander einzelne Teile des Formulars zu entfernen. Führen Sie den Vorgang so lange durch, bis das Problem nicht mehr reproduziert werden kann. Versuchen Sie die Quelle des Problems einzugrenzen, nachdem Sie alle Skript- und Objekteigenschaften überprüft haben. Um Teilformulare zu debuggen, können Sie das jeweilige Teilformular mit einer dicken farbigen Umrandung versehen. Sie haben auch die Möglichkeit, eine Füllfarbe zu verwenden. Mit Hilfe von Farben bzw. Füllfarben können Sie verdeutlichen, welches Teilformular verwendet wird und bis wohin das Teilformular reicht. Diese Vorgehensweise empfiehlt sich, wenn Sie die Begrenzungen eines Objekts ermitteln und feststellen möchten, warum das Objekt an einem bestimmten Ort platziert ist.

11.7.5. Hierarchieansicht

Zeigen Sie den Formularentwurf in der Hierarchieansicht an, um eine bessere Einsicht zu erhalten. Die Reihenfolge der Objekte in der Hierarchie entspricht der Reihenfolge ihrer Platzierung auf der Seite. Einige Objekte sind nicht anklickbar, wenn sie untereinander angeordnet sind.

11.7.6. Skriptfehlermeldungen

In Designer werden Skriptfehlermeldungen bei der Vorschau des Formulars auf der Registerkarte „Protokoll“ der Palette „Bericht“ angezeigt. Wenn der Formularentwurf FormCalc-Skripten enthält und der Fehler auf dem Server auftritt, werden Warnungen auf der Registerkarte „Protokoll“ angezeigt. Tritt der FormCalc-Skriptfehler auf dem Client auf, erfolgt die Anzeige der Meldung in Adobe Reader oder Acrobat.

Das Vorhandensein eines Fehlers in einem FormCalc-Skript verhindert die Ausführung des Skripts.

Bei Vorhandensein eines Fehlers in einem JavaScript wird das Skript bis zur Fehlerposition ausgeführt.

11.7.7. Syntaxfehler in FormCalc

Syntaxfehler in FormCalc sind bisweilen schwer zu beheben. Wenn die Meldung ""Syntax error near token '%1' on line %2, column %3"angezeigt wird, `enthält %1` im Allgemeinen den Token (Wort), der dem Fehler am nächsten ist. Das Token ist jedoch möglicherweise korrekt und die Meldung steht lediglich durch die Nähe zur Fehlerposition in Zusammenhang mit dem Fehler. Das folgende Skript generiert beispielsweise den Fehler 7008: „Syntaxfehler bei Token then in Zeile x, Spalte y.“

```
var b = abc(1)
if (b ne 1) dann
//comment
```

Das Problem ist, dass ein `endif` Token im Skript fehlt. Der letzte korrekte Token ist `then` (Kommentare zählen nicht als Token). Das Hinzufügen einer `endif` -Anweisung am Ende des Skripts behebt das Problem.

11.7.8. In einem Skriptobjekt definierte Funktionen

Sie können nur Funktionen aufrufen, die mit einem JavaScript-Skript in einem Skriptobjekt definiert sind. Ändern Sie daher die Skriptsprache im Skript-Editor in JavaScript. Anderenfalls werden Sie eventuell in einer Meldung informiert, dass Designer das Skriptobjekt nicht auflösen kann. Derselbe Fehler kann auftreten, wenn ein Syntaxproblem im Skriptobjekt auftritt.

11.7.9. Webdienstaufrufe

Verwenden Sie beim Erstellen von Webdienstaufrufen das postExecute-Ereignis, um die Rückgabewerte sowie etwaige Fehlermeldungen des Webdienstes anzuzeigen.

11.7.10. Lange SOM-Ausdrücke

Drücken Sie bei der Eingabe langer SOM-Ausdrücke mit mehreren Ebenen die Strg-Taste und klicken Sie im Zeichenbereich auf das Objekt. Der Befehl fügt den SOM-Ausdruck des Objekts in das Skript ein. Der SOM-Ausdruck ist relativ zu dem Objekt, in dem das Skript enthalten ist. Um den absoluten SOM-Ausdruck einzufügen, drücken Sie die Tastenkombination Strg+Umschalt und klicken Sie auf das Objekt. Diese Befehle sind nur aktiv, wenn Sie in der Entwurfsansicht auf Objekte klicken. In der Hierarchieansicht sind sie deaktiviert.

11.7.11. SOM-Ausdrücke testen

Tritt ein Fehler bei einem langen SOM-Ausdruck auf, beginnen Sie beim Stamm des Ausdrucks und testen Sie die einzelnen Punkte mit `className` bis Sie zum Problem gelangen. Beispiel, Testen `a.b.c.d` indem Sie am Stamm beginnen:

- console.println(`a.className`)
- console.println(`a.b.className`)
- console.println(`a.b.c.className`)
- console.println(`a.b.c.d.className`)

11.7.12. Skriptobjekte für das Debuggen von Formularentwürfen verwenden

Verwenden Sie ein Skriptobjekt (z. B. ein Fragment) als Hilfe beim Debuggen von Formularentwürfen:

- Geben Sie den Dump einer Node-Hierarchie unter einer Node aus.
- Geben Sie den Wert einer Eigenschaft oder das Attribut einer Node aus.
- Geben Sie aus, ob für die Node eine Eigenschaft oder ein Attribut angegeben wurde.
- Geben Sie den SOM-Ausdruck einer Node aus.
- Entfernt die `xml src` eines bestimmten Knotens.

Beispiel eines Skriptobjekts mit verschiedenen Debugging-Funktionen:

```
<script contentType="application/x-javascript" name="XFADEBUG">
//Das Skriptobjekt stellt verschiedene Nachfolgefunktionen bereit, um das
Debuggen eines Formulardesigns zu unterstützen
//Platzieren Sie die Knotenhierarchie in die Konsole.println()
function printNode(node) {... }
//Platzieren Sie den SOM-Ausdruck in die Konsole.println() function
printSOM(node) {... }
//Platzieren Sie die Eigenschaft- oder Attributwertfunktionprint Value(node,
attrOrPropertyName) {...}
function printXMLSource(node) { ....}
function printHasPropertySpecified(node, prop) {...}\\
</script>
```

11.7.13. Beim Formularentwurf zu vermeidende Fehler

- Bei einem Aufruf von `xfa.layout.relayout()`. im `docReady-Ereignis` verursacht oftmals Probleme, da das `docReady` Ereignis jeweils bei Fertigstellung des Layouts ausgelöst wird.
- Das Platzieren eines Containers mit Textfluss in einem positionierten Container verursacht Probleme mit Seitenumbrüchen, überlagerten Objekten und sich wiederholenden Teilformularen. Das Stammteilformular ist ein Container mit Textfluss. Nutzen Sie diesen Vorteil und platzieren Sie die Container mit Textfluss im Stammteilformular. Entfernen Sie die Teilformulare der Seite nach Fertigstellung des Layouts. Stellen Sie alternativ dazu die Teilformulare der Seite auf „Textfluss“ ein.
- Problem mit leeren Seiten (Acrobat 7.1 oder früher). Beim Entwurf des Formulars wird eine leere Seite angezeigt, wenn das Teilformular nicht in die Begrenzung des Inhaltsbereichs passt. Um das Problem mit der leeren Seite zu beheben, passen Sie entweder die Größe des Teilformulars an oder lassen Sie Umbrüche zwischen den Seiten zu. Verwendet der Benutzer Acrobat 7.1 oder früher, wird das Teilformular der zweiten Ebene auf einer anderen Seite angezeigt.

12. Mit Host-Anwendungen arbeiten

Unter einer Host-Anwendung versteht man die Anwendung, in der ein Formular zu einem bestimmten Zeitpunkt vorhanden ist.

Wenn Sie beispielsweise ein Formular mit Forms im HTML-Format wiedergeben, ist Forms in der Phase vor dem Wiedergabeprozess die Host-Anwendung.

Sobald Sie ein Formular wiedergeben und in einer Client-Anwendung wie Acrobat, Adobe Reader oder einem HTML-Browser anzeigen, wird die Client-Anwendung zur Host-Anwendung.

In Designer steht ein Skriptmodell mit Eigenschaften und Methoden zur Herstellung einer direkten Verbindung zur Host-Anwendung zur Verfügung. Beispielsweise können Sie mit den Eigenschaften und Methoden des Host-Skriptmodells in Acrobat oder Adobe Reader Navigationsaktionen für PDF-Seiten bereitstellen sowie mit der `importData` -Methode Daten in ein Formular laden

Sie können die Host-Skriptmodellsyntax bei jedem gültigen Skriptereignis für Formularentwurfsobjekte mit der folgenden Syntax in FormCalc und JavaScript referenzieren:

```
xfa.host.property_or_method
```

12.1. Eigenschaften und Methoden des Host-Skriptmodells

Mit Hilfe der Eigenschaften und Methoden des Host-Skriptmodells können Sie Informationen abrufen und Aktionen ausführen, die normalerweise über Berechnungen und Skripten nicht zugänglich sind. Sie können etwa den Namen der Host-Anwendung abrufen (z. B. Acrobat) oder bei einem interaktiven Formular von der aktuellen Seite aus weiterblättern. In der folgenden Tabelle werden die für das Host-Skriptmodell verfügbaren Eigenschaften und Methoden aufgeführt.

Eigenschaften	Methoden
appType	beep
calculationsEnabled	exportData
currentPage	gotoURL
language	importData
name	messageBox
numPages	pageDown
platform	pageUp

Eigenschaften	Methoden
title	print
validationsEnabled	resetData
variation	response
version	setFocus

Weitere Informationen zu den Eigenschaften und Methoden des Host-Skriptmodells finden Sie im Developer Center.

12.2. Die Funktionalität des Host-Skriptmodells im Vergleich

In der folgenden Tabelle werden die Eigenschaften und Methoden des Host-Skriptmodells von Designer aufgeführt und den entsprechenden Ausdrücken im JavaScript-Objektmodell von Acrobat gegenübergestellt.

Weitere Informationen zu den Eigenschaften und Methoden des Host-Skriptmodells finden Sie in der *Designer-Hilfe* oder unter Skriptreferenz.

Eigenschaften und Methoden des Host-Skriptmodells	Acrobat-Entsprechungen im JavaScript-Objektmodell
xfa.host.appType	app.viewerType
xfa.host.beep([INTEGER param])	app.beep([nType])
xfa.host.currentPage	doc.pageNum
xfa.host.exportData([STRING param1 [, BOOLEAN param2]])	doc.exportXFAData(cPath [, bXDP])
xfa.host.gotoURL(STRING param1)	doc.getURL(cURL, [bAppend]) oder app.launchURL(URL);
xfa.host.importData([STRING param])	doc.importXFAData(cPath)
xfa.host.language	app.language
xfa.host.messageBox(STRING param1 [, STRING param2 [, INTEGER param3 [, INTEGER param4]]])	app.alert(cMsg [, nIcon [, nType [, cTitle]]])
xfa.host.name	Keine

Eigenschaften und Methoden des Host-Skriptmodells	Acrobat-Entsprechungen im JavaScript-Objektmodell
`xfa.host.numPages`	`doc.numPages`
`xfa.host.pageDown()`	`doc.pageNum++`
`xfa.host.pageUp()`	`doc.pageNum--`
`xfa.host.platform`	`app.platform`
`xfa.host.print(BOOLEAN param1, INTEGER param2, INTEGER param3, BOOLEAN param4, BOOLEAN param5, BOOLEAN param6, BOOLEAN param7, BOOLEAN param8)`	`doc.print([ bUI [, nStart [, nEnd [, bSilent [, bShrinkToFit [, bPrintAsImage [, bReverse [, bAnnotations ] ] ] ] ] ] ] ])`
`xfa.host.resetData( [ STRING param ] )`	`doc.resetForm([ aFields ])`
`xfa.host.response( STRING param1 [, STRING param2 [, STRING param3 [, BOOLEAN param4] ] ])`	`app.response(cQuestion [, cTitle [, cDefault [, bPassword ] ] ])`
`xfa.host.setFocus( STRING param )`	`field.setFocus()` (Nicht weiter unterstützt)
`xfa.host.title`	`doc.title`
`xfa.host.variation`	`app.viewerVariation`
`xfa.host.version`	`app.viewerVersion`

VERKNPFTE LINKS:

Objekte in Berechnungen und Skripten referenzieren

13. Mit dem Ereignismodell arbeiten

Im Ereignismodell werden Eigenschaften von Objektereignissen gespeichert. Diese Eigenschaften sind dann nützlich, wenn Sie auf Werte zugreifen möchten, die außerhalb des Gültigkeitsbereichs der Ereignisse liegen, die im Skript-Editor in der Liste „Anzeigen“ aufgeführt sind.

Das Ereignismodell steuert die Änderungen in einem Formular, die vor, während und nach Aktionen ausgeführt werden. Zu diesen Aktionen zählen dynamische Formularereignisse, wie z. B. der Zeitpunkt, zu dem die Daten und der Formularentwurf zusammengeführt werden, jedoch bevor eine Paginierung angewendet wird, sowie interaktive Formularereignisse, wie z. B. wenn ein Benutzer den Wert eines Felds aktualisiert.

13.1. Eigenschaften und Methoden des Ereignismodells

Mit Hilfe der Eigenschaften und Methoden des Ereignismodells können Sie Informationen abrufen und Aktionen ausführen, die normalerweise nicht über Berechnungen und Skripten zugänglich sind. Beispielsweise können Sie den vollständigen Wert eines Feldes abrufen, bei dem andernfalls ein Teil der Daten wegfallen würde, da sie zu lang oder auf andere Weise ungültig sind. Dies ist dann von Nutzen, wenn Sie eine umfassende Fehlerprüfung durchführen müssen.

Eigenschaften	Methoden
change	emit
className	reset
commitKey	
fullText	
keyDown	
modifier	
name	
newContentType	
newText	
prevContentType	
prevText	

Eigenschaften	Methoden
reenter	
selEnd	
selStart	
shift	
soapFaultCode	
soapFaultString	
Ziel	

Weitere Informationen zu den Eigenschaften und Methoden des Ereignis-Skriptmodells finden Sie im Developer Center.

14. Von der Skripterstellung in Acrobat zu Designer wechseln

In Designer steht eine breite Palette von Funktionen zur Skripterstellung zur Auswahl. Dazu gehört auch die Unterstützung der gängigsten JavaScript-Objekte aus Acrobat. Wenn Sie ein Acrobat-Formular in Designer konvertieren, funktionieren die meisten JavaScript-Skripten weiterhin in der gewohnten Weise, ohne dass Änderungen erforderlich sind. Einige JavaScript-Skripten müssen jedoch manuell aus Acrobat konvertiert werden, um zu gewährleisten, dass die Funktionalität des Acrobat-Formulars erhalten bleibt.

Beim Konvertieren von Skripten in einem Acrobat-Formular sind einige Unterschiede zwischen der Skripterstellung in Designer und der Skripterstellung in Acrobat zu beachten:

Designer-Arbeitsbereich

Sie können im Designer-Arbeitsbereich die Eigenschaften und Verhaltensweisen von Formularobjekten ändern, ohne Skripten zu erstellen.

Skriptsprachen

Designer unterstützt neben JavaScript auch FormCalc, eine einfache Berechnungssprache. Mit den in FormCalc integrierten Funktionen können zahlreiche nützliche Vorgänge ausgeführt werden, für die anderenfalls ein erheblicher Skripterstellungsaufwand anfallen würde.

Referenzieren von Objekten, Eigenschaften und Methoden

Designer-Formulare sind hochgradig strukturiert. Daher müssen Sie zum Referenzieren bestimmter Objekte, Eigenschaften oder Methoden die entsprechende Referenzsyntax in Ihr Skript einbauen. Die Anweisungsende-Optionen im Skript-Editor erleichtern die Erstellung der Referenz-Syntax.

JavaScript-Objekte, -Eigenschaften und -Methoden aus Acrobat können in Designer weiter verwendet werden. Allerdings sollten Sie JavaScript aus Acrobat nur für Aufgaben einsetzen, die Sie nicht mit dem XML-Formularobjektmodell in Designer ausführen können. Beispielsweise können Sie JavaScript aus Acrobat verwenden, um Anhänge, Lesezeichen oder Anmerkungen hinzuzufügen, ein Formular zu durchsuchen, die Rechtschreibung in einem Formular zu prüfen, Berichte zu erstellen oder Metadaten abzurufen bzw. zu bearbeiten. JavaScript aus Acrobat ist für Vorgänge wie das Festlegen von Feldwerten, das Hinzufügen neuer Felder zu einem Formular sowie das Löschen von Seiten aus einem Formular ungeeignet.

HINWEIS: *Einem Designer-Formular können in Acrobat keine JavaScript-Skripten hinzugefügt werden. Dies gilt auch für Acrobat-Formulare, die mit Designer konvertiert wurden. Bei der Anzeige eines Designer-Formulars in Acrobat sind keine JavaScript-Werkzeuge verfügbar.*

Weitere Informationen über das Konvertieren von Acrobat in Designer finden Sie im Artikel Konvertieren von Acrobat-JavaScript für die Verwendung in Designer Forms im Developer Center.

14.1. Acrobat-Formulare mit Skripten konvertieren

Einer der ersten Schritte bei der Konvertierung eines Formulars aus Acrobat in Designer besteht darin festzustellen, welcher Teil der Acrobat-Skripten von Designer unterstützt wird und welcher Teil konvertiert werden muss.

In der Regel sollten Sie alle Acrobat-Skripten in entsprechende Designer-Skripten konvertieren. In Designer werden der Entwurf und die Implementierung von Formularlösungen durch die hochgradige Strukturierung der Designer-Formulare sowie durch nützliche formularspezifische Funktionen beschleunigt und erleichtert.

Zu den Acrobat-Skripten, die Sie beibehalten sollten, gehören die Skripten, die für die Umgebung und die Peripherievorgänge des Formulars zuständig sind. Dazu gehören das Hinzufügen von Anhängen oder Multimedia, die Durchführung von Suchvorgängen, die Berichterstellung sowie der Umgang mit Dokument-Metadaten.

Weitere Informationen über das Konvertieren von Acrobat in Designer finden Sie im Artikel Konvertieren von Acrobat-JavaScript für die Verwendung in Designer Forms im Developer Center.

VERKNPFTE LINKS:

Von der Skripterstellung in Acrobat zu Designer wechseln

JavaScript-Objekte aus Acrobat in Designer verwenden

In Designer unterstützte JavaScript-Objekte aus Acrobat

14.2. JavaScript-Objekte aus Acrobat in Designer verwenden

In Designer können Sie mit Hilfe der Acrobat-Skriptsyntax Skripten für bestimmte JavaScript-Objekte in Acrobat erstellen. Sie können daher die Eigenschaften und Methoden dieser Objekte in Ihren Formularen verwenden. Wenn Sie beispielsweise eine Meldung in der JavaScript-Konsole von Acrobat anzeigen möchten, können Sie dem Ereignis eines Formularentwurfsobjekts in Designer das folgende Skript hinzufügen:

```
    console.println(„Diese Meldung wird in der JavaScript-Konsole
angezeigt.“);
```

Sie können auch veranlassen, dass das Formular per E-Mail versendet wird. Fügen Sie dazu dem `click` -Ereignis einer Schaltfläche Folgendes hinzu:

```
var myDoc = event.target;
myDoc.mailDoc(true);
```

HINWEIS: *In Designer müssen Sie dafür sorgen, dass die Skriptsprache für das Ereignis auf JavaScript eingestellt ist, damit das Skript zur Laufzeit korrekt ausgeführt wird.*

Sie können auch Verweise auf die JavaScript-Objekte in Acrobat in die Referenz-Syntax einbauen. Beispielsweise ruft das folgende Skript den Unterschriftsstatus eines Unterschriftsfelds ab und führt eine vom Status abhängige Aktion aus:

```
// Fahren Sie fort, wenn das aktuelle Feld nicht signiert wurde.
var oState =
event.target.getField("form1[0].#subform[0].SignatureField1[0]")
.signatureValidate(); //Rufen Sie den Signaturstatus des Felds auf.

if (oState == 0) {
...
}
```

HINWEIS: *In diesem Beispiel wird für den Verweis auf den Text eine vollständig qualifizierte Referenzsyntax verwendet. Weitere Informationen zum Referenzieren von Formularentwurfsobjekten finden Sie unter Objekteigenschaften und -werte referenzieren.*

Bei Verwendung von JavaScript aus Acrobat in Designer sind die folgenden Punkte zu beachten:

- Verwenden Sie in Designer `event.target,` um auf das `Doc` JavaScript-Objekte aus Acrobat zuzugreifen. In Acrobat wird das `this` -Objekt zum Referenzieren von `Doc` -Objekten verwendet; aber in Designer bezieht sich das `this` -Objekt auf das Formularentwurfsobjekt, an welches das Skript angehängt ist.
- Im Skript-Editor stehen keine Anweisungsende-Optionen für JavaScript-Objekte aus Acrobat zur Verfügung. Siehe JavaScript für Acrobat API-Referenz.
- Die Doc-Methode `event.target.importTextData("file.txt")` wird für dynamische XFA-Formulare, die zertifiziert wurden, nicht unterstützt.

  Weitere Informationen über das Konvertieren von Acrobat in Designer finden Sie im Artikel Konvertieren von Acrobat-JavaScript für die Verwendung in Designer Forms im Developer Center.

VERKNPFTE LINKS:

Von der Skripterstellung in Acrobat zu Designer wechseln

Acrobat-Formulare mit Skripten konvertieren

In Designer unterstützte JavaScript-Objekte aus Acrobat

14.3. In Designer unterstützte JavaScript-Objekte aus Acrobat

In der folgenden Tabelle finden Sie eine Übersicht über die Verfügbarkeit der am häufigsten verwendeten Acrobat-Objekte, -Eigenschaften und -Methoden in Designer und deren Entsprechungen in Designer. Einige gängige -Objekte, -Eigenschaften und -Methoden fehlen in dieser Liste. Beispielsweise sind keine Multimedia-Objekte aufgeführt, weil sie in Formularen nur sehr selten vorkommen.

Ist keine entsprechende Designer-Funktionalität aufgeführt, gibt es keine Designer-Eigenschaft oder -Methode, mit welcher das Acrobat-Verhalten reproduziert werden kann. Sie können aber trotzdem eigene Funktionen oder Skripten erstellen, welche die Acrobat-Funktionalität reproduzieren.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
Annot-**Objekteigenschaften und -methoden**			
Alle Eigenschaften und Methoden	Ja	Kein	Nur Formulare mit festem Layout unterstützen die Anmerkungsebene.
App**Objekteigenschaften**			
`calculate`	Nein	Kein	Designer umfasst die `execCalculate` -Methode, die das `calculate` -Ereignis initiiert. execCalculate
`language`	Ja	`xfa.host.language`	Siehe `language` zugreifen. language
`monitors`	Ja	Kein	
`platform`	Ja	`xfa.host.platform`	Siehe `platform` zugreifen. platform
`plugins`	Ja	Kein	
`toolbar`	Ja	Kein	
`viewerType`	Ja	`xfa.host.appType`	Siehe `appType` zugreifen. appType
`viewerVariation`	Ja	`xfa.host.variation`	Siehe `variation` zugreifen. variation
`viewerVersion`	Ja	`xfa.host.version`	Siehe `version` zugreifen. version
App**Objektmethoden**			
`addMenuItem`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
addSubMenu	Ja	Kein	
addToolButton	Ja	Kein	
alert	Ja	xfa.host.messageBox()	Siehe messageBox -Methode erstellen. messageBox
beep	Ja	xfa.host.beep()	Siehe beep -Methode erstellen. beep
browseForDoc	Ja	Kein	
clearInterval	Ja	Kein	
clearTimeOut	Ja	Kein	
execDialog	Ja	Kein	
execMenuItem	Ja	Kein	Führt den angegebenen Menübefehl aus. Verwenden Sie diese Methode in Designer für Datei-Menübefehle.
getNthPluginName	Ja	Kein	
getPath	Ja	Kein	
goBack	Ja	Kein	
goForward	Ja	Kein	
hideMenuItem	Ja	Kein	
hideToolbarButton	Ja	Kein	
launchURL	Ja	Kein	Designer umfasst die gotoURL -Methode zum Laden einer angegebenen URL in die Client-Anwendung wie z. B. Acrobat oder Adobe Reader. gotoURL
listMenuItems	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`listToolbarButtons`	Ja	Kein	
`mailGetAddrs`	Ja	Kein	
`mailMsg`	Ja	Kein	
`newDoc`	Ja	Kein	Diese Methode kann nur während Stapelverarbeitungs-, Konsolen- oder Menüereignissen ausgeführt werden.
`newFDF`	Nein	Kein	
`openDoc`	Ja	Kein	
`openFDF`	Nein	Kein	
`popUpMenuEx`	Ja	Kein	
`popUpMenu`	Ja	Kein	
`removeToolButton`	Ja	Kein	
`response`	Ja	`xfa.host.response()`	Siehe `response` -Methode erstellen. response
`setInterval`	Ja	Kein	
`setTimeOut`	Ja	Kein	
`trustedFunction`	Ja	Kein	
`trustPropagatorFunction`	Ja	Kein	Diese Methode ist nur während der Stapelverarbeitungs-, Konsolen- und Anwendungsinitialisierung verfügbar.
Lesezeichen - Objekteigenschaften und -methoden			
Alle Eigenschaften und Methoden	Ja	Kein	
`doc`**Objekteigenschaften**			
`author`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`baseURL`	Ja	Kein	
`bookmarkRoot`	Ja	Kein	
`calculate`	Nein	Kein	
`dataObjects`	Ja	Kein	
`delay`	Nein	Kein	
`dirty`	Ja	Kein	Dieses JavaScript-Skript für Designer speichert eine Kopie eines Formulars und prüft, ob sich das Formular geändert hat. `var sOrigXML = xfa.data.saveXML; if (sOrigXML != xfa.data.saveXML) {...}`
`disclosed`	Ja	Kein	
`documentFileName`	Ja	Kein	
`dynamicXFAForm`	Ja	Kein	
`extern`	Ja	Kein	
`filesize`	Ja	Kein	
`Ausgeblendet`	Ja	Kein	
`icons`	Ja	Kein	
`keywords`	Ja	Kein	
`layout`	Ja	Kein	
`media`	Ja	Kein	
`Metadaten`	Ja	`xfa.form.desc`	Siehe `desc`Objekt. desc
`modDate`	Ja	Kein	
`mouseX mouseY`	Ja	Kein	
`noautocomplete`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`nocache`	Ja	Kein	
`numFields`	Ja	`xfa.layout.pageContent()`	Die `pageContent` -Methode gibt eine Liste sämtlicher Objekte zurück, die einem bestimmten Typ angehören. Allerdings müssen Sie die Methode für Designansichten und Masterseiten ausführen, damit das gesamte Formular durchsucht wird. pageContent

`numPages`	Ja	`xfa.host.numPages`
		oder
		`xfa.layout.absPageCount()`
		`xfa.layout.pageCount()`
			Die `numPages` -Eigenschaft gibt die Seitenanzahl des im Client wiedergegebenen Formulars zurück. Weitere Informationen über die Verarbeitung von Anforderungen durch die ID und das Festlegen von IDs finden Sie im Abschnitt `absPageCount` und `pageCount` Methoden.
			numPages
			absPageCount
			pageCount

`pageNum`	Ja	`xfa.host.currentPage`	Siehe `currentPage` zugreifen. currentPage

`pageNum--`	Ja	`xfa.host.currentPage--`
		oder
		`xfa.host.pageUp()`
			Siehe `currentPage` Eigenschaft oder `pageUp` -Methode.
			currentPage
			pageUp

`pageNum++`	Ja	`xfa.host.currentPage++`
		oder
		`xfa.host.pageDown()`
			Siehe `currentPage` Eigenschaft oder `pageDown` -Methode.
			currentPage
			pageDown

`path`	Ja	Kein	
`securityHandler`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`templates`	Nein	Kein	Verwenden Sie Teilformularobjekte in Designer. Verwenden Sie Eigenschaften und Methoden, um Teilformularinstanzen hinzuzufügen, zu entfernen, zu verschieben und einzurichten. Teilformularinstanzen mit Hilfe von Skripten hinzufügen und entfernen
`title`	Ja	`xfa.host.title`	Siehe title.
`doc`**Objektmethoden**			
`addAnnot`	Ja	Kein	
`addField`	Nein	Kein	Sie müssen Formulare mit einem festen Layout in Designer verwenden und dann das i `instanceManager` Objekt verwenden, um die Anzahl der Instanzen eines bestimmten Objekts zu entfernen, hinzuzufügen und festzulegen. instanceManager Weitere Informationen unter Teilformularinstanzen mit Hilfe von Skripten hinzufügen und entfernen.
`addIcon`	Ja	Kein	
`addLink`	Nein	Kein	
`addRecipientListCryptFilter`	Ja	Kein	
`addScript`	Ja	Kein	
`addThumbnails`	Nein	Kein	
`addWatermarkFromFile`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`addWatermarkFromText`	Ja	Kein	
`addWeblinks`	Ja	Kein	
`appRightsSign`	Ja	Kein	
`appRightsValidate`	Ja	Kein	
`bringToFront`	Ja	Kein	
`calculateNow`	Nein	`xfa.form.recalculate(1);` oder `execCalculate()`	recalculate Die `recalculate` -Methode erzwingt die Initiierung eines angegebenen Satzes von Skripten in den `calculate` -Ereignissen. Der boolesche Wert gibt an, ob `True` (Standard) - alle Berechnungsskripte werden initiiert; oder `False` - nur anstehende Berechnungsskripten werden initiiert). Das Designer `calculate` -Objekt steuert, ob ein Benutzer beim Ausfüllen den berechneten Wert eines Feldes überschreiben darf. execCalculate Alternativ können Sie die `execCalculate` -Methode für jedes Objekt verwenden, für das Sie eine Neuberechnung erzwingen möchten.
`closeDoc`	Ja	Kein	
`createDataObject`	Ja	Kein	
`createTemplate`	Nein	Kein	Bei Designer-Formularen gibt es keine Entsprechung für Acrobat-Vorlagen. In Designer müssen Sie Teilformularobjekte verwenden.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`deletePages`	Nein	Kein	instanceManager In Designer können Sie das `instanceManager` -Objekt verwenden, um das Teilformularobjekt zu entfernen, das eine Seite im Formular darstellt. Weitere Informationen unter Teilformularinstanzen mit Hilfe von Skripten hinzufügen und entfernen.
`embedDocAsDataObject`	Ja	Kein	
`encryptForRecipients`	Ja	Kein	
`encryptUsingPolicy`	Ja	Kein	
`exportAsText`	Ja	Kein	Diese Methode ist nur in der JavaScript-Konsole des JavaScript-Debuggers in Acrobat bzw. während der Stapelverarbeitung verfügbar.
`exportAsFDF`	Nein	`xfa.host.exportData()`	exportData Die `exportData` -Methode exportiert anstelle einer FDF-Datei eine XML- oder XDP-Datei.
`exportAsXFDF`	Nein	`xfa.host.exportData()`	exportData Die `exportData` -Methode exportiert anstelle einer FDF-Datei eine XML- oder XDP-Datei.
`exportDataObject`	Ja	Kein	
`exportXFAData`	Nein	`xfa.host.exportData()`	exportData Die `exportData` -Methode exportiert anstelle einer FDF-Datei eine XML- oder XDP-Datei.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`extractPages`	Nein	Kein	
`flattenPages`	Nein	Kein	
`getAnnot`	Ja	Kein	
`getAnnots`	Ja	Kein	
`getDataObjectContents`	Ja	Kein	
`getField("`*`Feldname`*`")`	Ja	`xfa.resolveNode ("`*`Feldname`*`")`	resolveNode Die `resolveNode` -Methode greift auf das angegebene Objekt in der XML-Quelle des Formulars zu.
`getLegalWarnings`	Ja	Kein	
`getLinks`	Nein	Kein	
`getNthFieldName`	Ja	Sie müssen alle Objekte mit einem ähnlichen Klassennamen durchlaufen, bis das`n` . -Vorkommen erreicht ist.	className Siehe `className` zugreifen.
`getNthTemplate`	Nein	Kein	
`getOCGs`	Ja	Kein	
`getOCGOrder`	Ja	Kein	
`getPageBox`	Ja	Kein	
`getPageLabel`	Ja	Kein	
`getPageNthWord`	Ja	Kein	
`getPageNthWordQuads`	Ja	Kein	
`getPageNumWords`	Ja	Kein	
`getPageRotation`	Ja	Kein	
`getPrintParams`	Ja	Kein	
`getTemplate`	Nein	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`getURL`	Ja	`xfa.host.gotoURL( "http://www.adobe.com");`	Siehe `gotoURL` -Methode. gotoURL
`gotoNamedDest`	Nein	Kein	
`importAnFDF`	Nein	Kein	
`importAnXFDF`	Ja	Kein	
`importDataObject`	Ja	Kein	
`importIcon`	Ja	Kein	
`importTextData`	Ja	Kein	
`importXFAData`	Nein	`xfa.host.importData ("filename.xdp" );`	Siehe `importData` -Methode. importData
`insertPages`	Nein	Kein	
`mailDoc`	Ja	Kein	
`mailForm`	Nein	Kein	
`movePage`	Nein	Kein	
`newPage`	Nein	Kein	
`openDataObject`	Ja	Kein	
`print`	Ja	`xfa.host.print( );`	Siehe `print` -Methode. print
`removeDataObject`	Ja	Kein	
`removeField`	Nein	Kein	
`removeIcon`	Ja	Kein	
`removeLinks`	Nein	Kein	
`removeScript`	Ja	Kein	
`removeTemplate`	Nein	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`removeThumbnails`	Nein	Kein	
`removeWeblinks`	Ja	Kein	
`replacePages`	Nein	Kein	
`resetForm`	Nein	`xfa.host.resetData()` oder `xfa.event.reset()`	Die `resetData`Methode setzt alle Feldwerte in einem Formular auf die Standardwerte zurück. Die `reset` -Methode setzt alle Eigenschaften im Ereignismodell zurück. resetData reset
`saveAs`	Ja	Kein	In Designer muss die Datei auf Anwendungsebene gespeichert werden. Die folgenden Skripten sind Beispiele für die Speicherung auf Anwendungsebene: `app.executeMenuItem ("SaveAs");` oder `var myDoc = event.target; myDoc.saveAs();`
`spawnPageFromTemplate`	Nein	Kein	
`setAction`	Nein	Kein	
`setPageLabel`	Ja	Kein	
`setPageRotation`	Nein	Kein	
`setPageTabOrder`	Nein	Kein	Wählen Sie in Designer „Bearbeiten“ > „Tab-Reihenfolge“, um die Tab-Reihenfolge festzulegen.
`setScript`	Nein	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`submitForm`	Ja	Verwenden Sie eines der Senden-Schaltfläche-Objekte in Designer.	
`Ereignis`**Objekteigenschaften**			
`change`	Ja	`xfa.event.change`	change Siehe `change` zugreifen.
`targetName`	Ja	`xfa.event.target`	Ziel Siehe `Ziel` zugreifen.
`field`**Objekteigenschaften**			
`comb`	Nein	Kein	
`charLimit`	Nein	`this.value.#text.maxChars`	Bei Formularen mit festem Layout kann die maximale Zeichenanzahl im Designer-Arbeitsbereich festgelegt werden. Sie können Felder in Formularen einstellen, deren Layout an die Datenmenge angepasst wird. maxChars
`display = display.noView`	Nein	Siehe Präsenz von Formularentwurfsobjekten ändern.	presence Sie können auch die `presence` -Eigenschaft im Designer-Arbeitsbereich festlegen. Sie können das `prePrint` -Ereignis nicht verwenden, um die Präsenz eines Objekts vor dem Drucken zu ändern.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`display = display.noPrint`	Nein	Siehe Präsenz von Formularentwurfsobjekten ändern.	presence Sie können auch die `presence` -Eigenschaft im Designer-Arbeitsbereich festlegen. Sie können das `prePrint` -Ereignis nicht verwenden, um die Präsenz eines Objekts vor dem Drucken zu ändern.
`defaultValue`	Nein	Kein	Legt den Standard-Feldwert im Designer-Arbeitsbereich fest.
`exportValues`	Nein	Kein	Legt den Exportwert im Designer-Arbeitsbereich fest.
`fillColor`	Nein	`xfa.form.Form1.NumericField1.fillColor`	fillColor Siehe `fillColor` zugreifen.
`Ausgeblendet`	Nein	`this.presence = "invisible"` `this. presence = "visible"`	presence Sie können auch die `presence` -Eigenschaft im Designer-Arbeitsbereich festlegen.
`multiLine`	Nein	`this.ui.textEdit.multiLine = "1";`	multiLine Siehe `multiLine` zugreifen.
`password`	Nein	Kein	Designer enthält ein Kennwortfeld, das Sie für Kennwörter in Formularen verwenden können.
`page`	Nein	Kein	Bei Designer-Formularen nicht zutreffend.
`print`	Nein	`this.relevant = "-print";`	relevant Siehe `relevant` zugreifen.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`radiosInUnison`	Nein	Kein	Gruppierte Optionsfelder in Designer schließen sich standardmäßig gegenseitig aus.
`rect`	Ja	Sie können die Höhe und Breite eines Designer-Formularfelds mit Hilfe der folgenden Referenzsyntax abrufen: `this.h; this.w;` Alternativ können Sie die x- und y-Koordinaten eines Objekts mit der folgenden Referenz-Syntax abrufen: `this.x; this.y;`	h, x, y Siehe `h`, `w`, `x` und `y` properties.
`Erforderlich`	Nein	`this.mandatory = "error";` oder `this.validate.nullTest = "error";`	mandatory, nullTest Siehe `mandatory` und `nullTest` properties.
`textColor`	Nein	`this.fontColor`	fontColor Siehe `fontColor` zugreifen.
`textSize`	Nein	`this.font.size`	size Siehe `size` zugreifen.
`textFont`	Nein	`this.font.typeface`	typeface Siehe `typeface` zugreifen.
`value`	Nein	`this.rawValue`	rawValue Siehe `rawValue` zugreifen. value Designer-Felder haben eine `value` -Funktion; Sie ist nicht das Entsprechung der Acrobat `value` -Eigenschaft.

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`field`**Objektmethoden**			
`clearItems`	Nein	`DropDownList1.clearItems();`	clearItems Die `clearItems` -Methode gilt nur für Dropdown-Listen- und Listenfeldobjekte in Designer.
`deleteItemAt`	Nein	Kein	
`getItemAt`	Nein	Kein	
`insertItemAt`	Nein	`DropDownList1.addItem .....)`	addItem Siehe `addItem` -Methode.
`isBoxChecked`	Nein	`if(CheckBox1.rawValue == 1)....`	rawValue Siehe `rawValue` -Eigenschaft.
`isDefaultChecked`	Nein	Kein	
`setAction`	Nein	Kein	Bei Designer-Formularen nicht zutreffend.
`setFocus`	Ja	`xfa.host.setFocus ("TextField1.somExpression")`	setFocus Die `setFocus` -Methode setzt voraus, dass das angegebene Objekt einen eindeutigen Namen hat und nicht mit anderen Formularobjekten verwechselt werden kann.
`setItems`	Nein	Kein	
`setLock`	Ja	Kein	
`signatureGetModifications`	Ja	Kein	
`signatureGetSeedValue`	Ja	Kein	
`signatureInfo`	Ja	Kein	
`signatureSetSeedValue`	Ja	Kein	
`signatureSign`	Ja	Kein	

JavaScript in Acrobat	Designer-Unterstützung	JavaScript-Entsprechung in Designer	Kommentare
`signatureValidate`	Ja	Kein	
`search`**Objektmethode**			
`search.query("<your text>");`	Ja	Keine	Das ".." der Kurzbefehlsyntax (zwei Punkte, (..)) von FormCalc können Sie das XML-Formularobjektmodell nach Objekten durchsuchen. Weitere Informationen unter Referenz-Syntax-Kurzbefehle für FormCalc.
`SOAP`**Objektmethode**			
Alle Eigenschaften und Methoden	Ja	Kein	

VERKNPFTE LINKS:

Mit dem Ereignismodell arbeiten

15. Beispiele für gängige Aufgaben bei der Skripterstellung

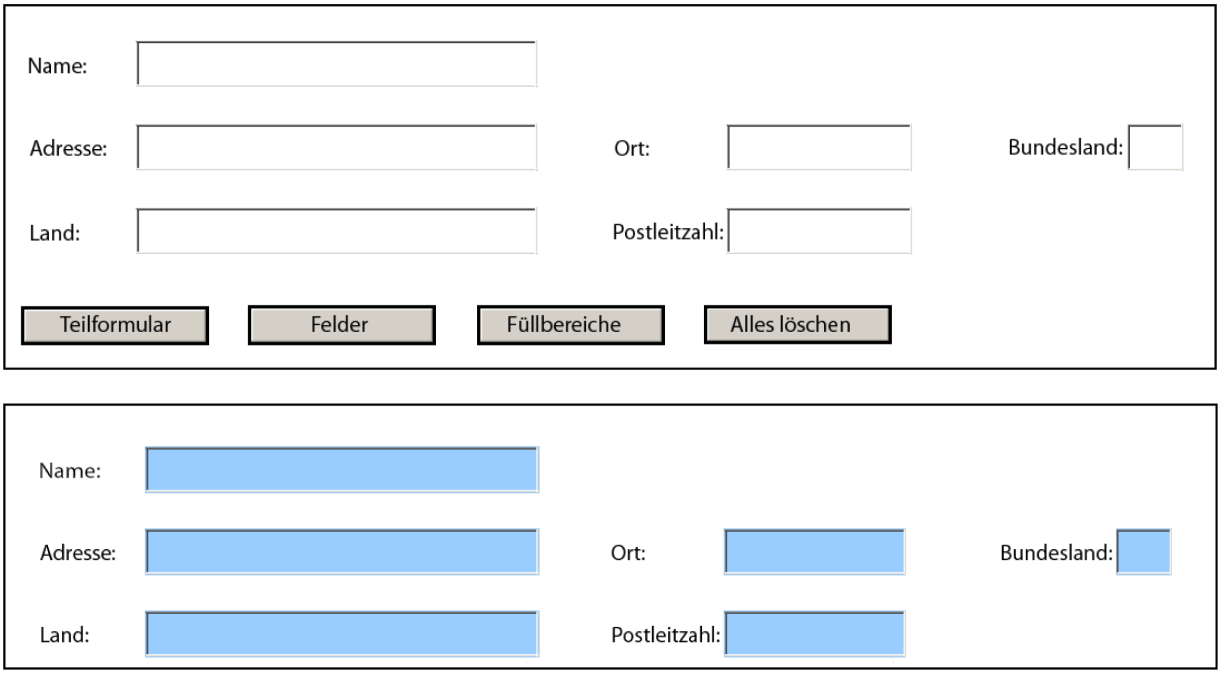
Die Beispiele zur Skripterstellung veranschaulichen schnelle und einfache Techniken, die Sie auf Ihre eigenen Projekte übertragen können.

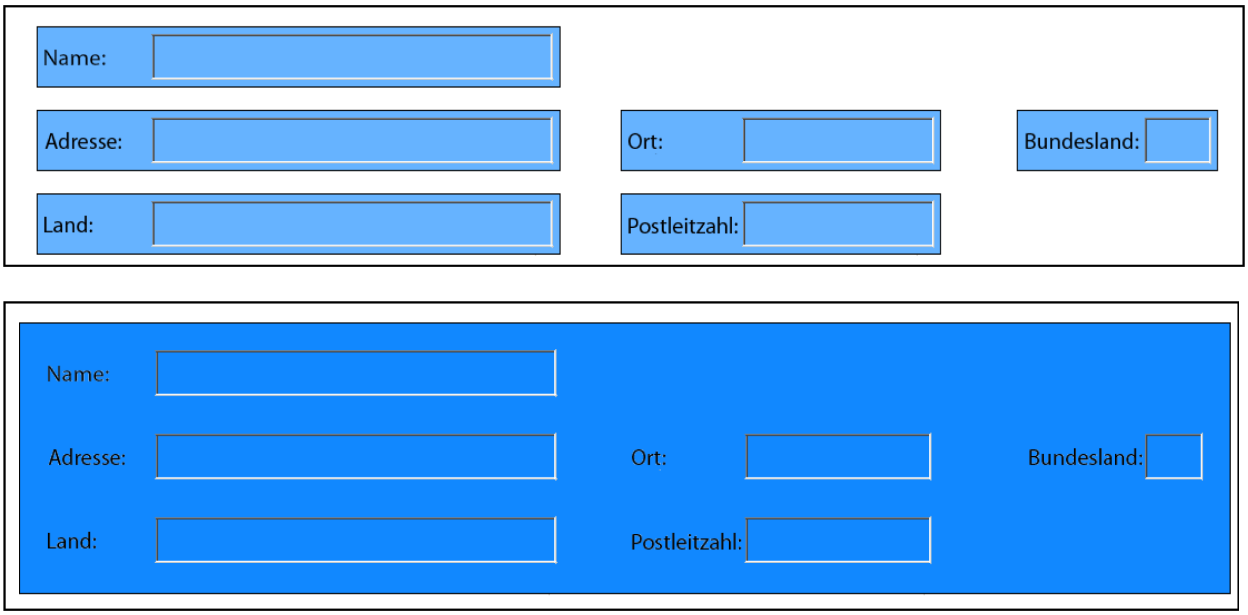
Weitere Beispiele und Anregungen finden Sie unter Developer Center.

15.1. Hintergrundfarben von Feldern, Füllbereichen und Teilformularen ändern

Dieses Beispiel zeigt, wie Sie die Hintergrundfarbe von Teilformularen, Feldern und Füllbereichen in einem Formular als Reaktion auf Benutzeraktionen zur Laufzeit ändern.

In diesem Beispiel ändert sich die Hintergrundfarbe eines bestimmten Objekts, wenn der Benutzer auf eine Schaltfläche klickt.





HINWEIS: *Um die Hintergrundfarbe von Objekten zur Laufzeit zu ändern, müssen Sie Ihr Formular im Format "Acrobat (Dynamisch) XML-Formular" speichern.*

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.1.1. Skripten für die Hintergrundfarben von Teilformularen und Textfeldern

Die Hintergrundfarben von Teilformularen und Textfeldern werden mit der `fillColor` -Methode eingestellt. Beispielsweise stellt die folgende Zeile das Skript für das Teilformular dar:

```
Subform1.fillColor = "17,136,255";
```

Die folgenden Zeilen bilden das Skript für die Hintergrundfarbe der Textfelder:

```
Subform1.Name.fillColor = "102,179,255";
Subform1.Address.fillColor = "102,179,255";
Subform1.City.fillColor = "102,179,255";
Subform1.State.fillColor = "102,179,255";
Subform1.ZipCode.fillColor = "102,179,255";
Subform1.Country.fillColor = "102,179,255";
```

15.1.2. Skripterstellung für die Hintergrundfarbe von Füllbereichen

Beim Festlegen der Hintergrundfarbe oder des Füllbereichs für die einzelnen Textfelder müssen die Skripten auf Eigenschaften zugreifen, die einen Referenz-Syntax-Ausdruck mit dem Nummernzeichen (#) erfordern. Da JavaScript das Nummernzeichen (#) in Referenz-Syntax-Ausdrücken nicht richtig interpretiert, verwendet das Skript die `resolveNode` -Methode, um den Ausdruck zu lösen.

```
xfa.resolveNode("Subform1.Name.ui.#textEdit.border.fill.color").value =
"153,204,255";
xfa.resolveNode("Subform1.Address.ui.#textEdit.border.fill.color").value =
"153,204,255";
xfa.resolveNode("Subform1.City.ui.#textEdit.border.fill.color").value =
"153,204,255";
xfa.resolveNode("Subform1.State.ui.#textEdit.border.fill.color").value =
"153,204,255";
xfa.resolveNode("Subform1.ZipCode.ui.#textEdit.border.fill.color").value =
"153,204,255";
xfa.resolveNode("Subform1.Country.ui.#textEdit.border.fill.color").value =
"153,204,255";
```

15.1.3. Skripterstellung für die Schaltfläche „Alles löschen“

Das Skript für die Schaltfläche „Alles löschen“ verwendet die `remerge` Methode und führen Sie den Formularentwurf und die Formulardaten erneut zusammen. In diesem Beispiel stellt die Methode die Felder, Füllbereiche und Teilformulare in ihrem Originalzustand wieder her.

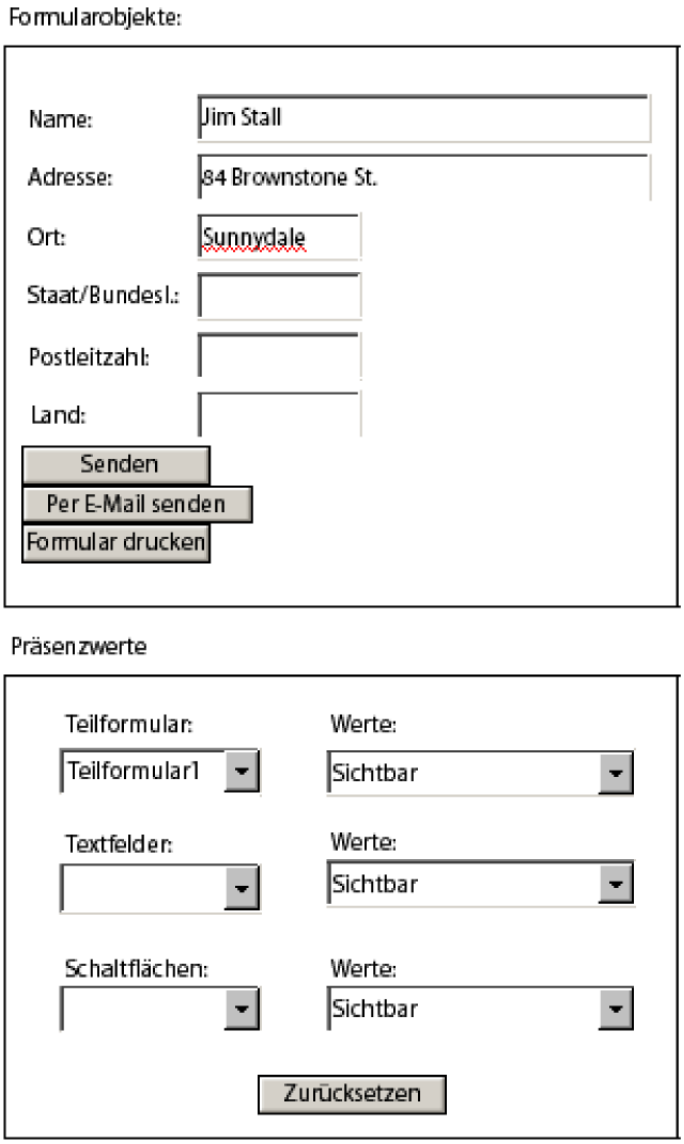
```
xfa.form.remerge();
```

15.2. Objekte ein- und ausblenden

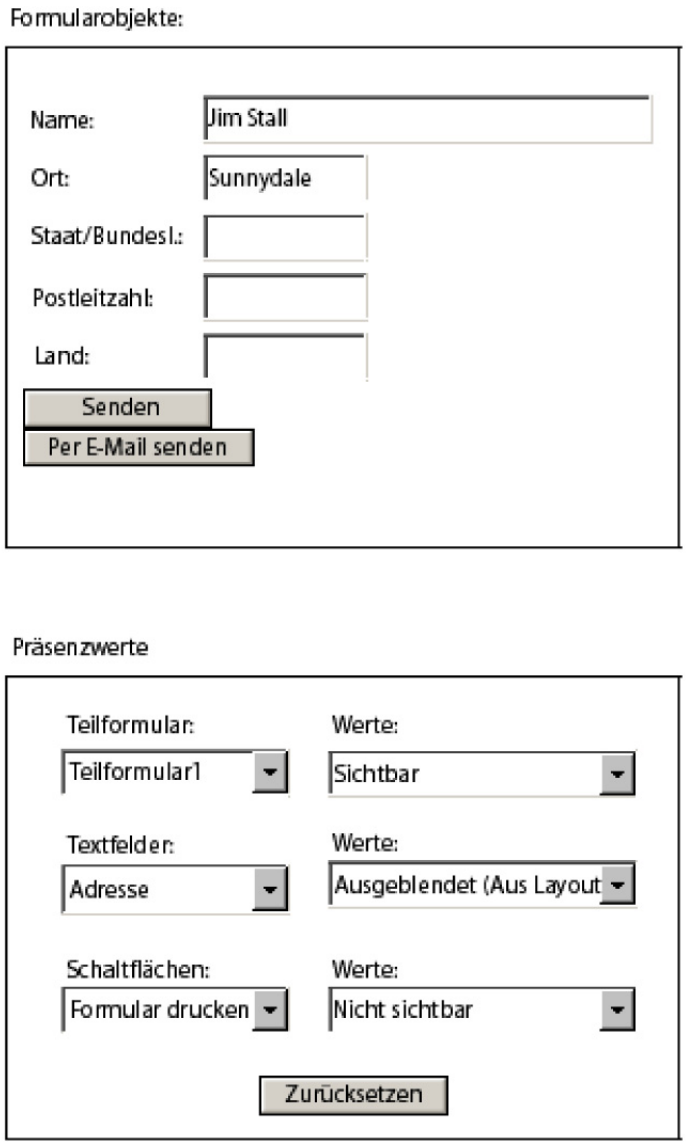
Dieses Beispiel zeigt, wie Sie Schaltflächen beim Drucken eines Formulars ausblenden und wie Sie Objekte durch Ändern der Präsenzwerte zur Laufzeit ein- und ausblenden.

HINWEIS: *Sie können Objekte in Formularen mit flexiblem Layout auch über das Dialogfeld „Action Builder“ im Menü „Extras“ ein- und ausblenden, ohne Skripten zu erstellen. Weitere Informationen finden Sie unter Erstellen von Aktionen in Formularen*

In diesem Beispiel sind alle Formularobjekte im Formular sichtbar.



Der Formularbenutzer kann mit den Dropdown-Listen im Bereich "Präsenzwerte" Objekte ein- oder ausblenden. In der folgenden Abbildung ist das Feld "Adresse" ausgeblendet und das Formularlayout entsprechend angepasst. Die Schaltfläche "Formular drucken" ist ebenfalls nicht sichtbar.



HINWEIS: *Um Objekte zur Laufzeit ein- und auszublenden, müssen Sie Ihr Formular im Format "Acrobat (Dynamisch) XML-Formular" speichern.*

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.2.1. Skripterstellung für die Präsenzwerte der Teilformulare

Das Skript für die Präsenzwerte der Teilformulare enthält eine switch-Anweisung zur Steuerung der drei Präsenzoptionen, die ein Formularbenutzer dem Teilformularobjekt zuweisen kann:

```
switch(xfa.event.newText) {
case 'Invisible':
Subform1.presence = "invisible";
break;
```

```
case 'Hidden (Exclude from Layout)':
Subform1.presence = "hidden";
break;
default: S
ubform1.presence = "visible";
break;
}
```

15.2.2. Skripten für die Präsenzwerte der Textfelder

Für das Skript für die Präsenzwerte der Textfelder sind zwei Variablen erforderlich. Die erste Variable speichert die in Subform1 enthaltene Anzahl von Objekten:

```
var nSubLength = Subform1.nodes.length;
```

Die zweite Variable speichert den Namen des Textfelds, welches der Formularbenutzer in der Dropdown-Liste "Textfelder" auswählt:

```
var sSelectField = fieldList.rawValue;
```

Das folgende Skript verwendet die `replace` -Methode, um alle Leerstellen aus dem Namen des Felds zu entfernen, die in der `sSelectField` Variable gespeichert sind, wodurch der Wert der Dropdown-Liste mit dem Namen des Objekt in der Palette „Verlauf“ übereinstimmt:

```
sSelectField = sSelectField.replace(' ', '');
```

Dieses Skript verwendet eine `for` Schleife, um zwischen allen Objekten, die in Subform1 enthalten sind, zu wechseln:

```
für (var nCount = 0; nCount < nSubLength; nCount++) {
```

Wenn das aktuelle Objekt in Subform1 vom Typ `field` ist und das aktuelle Objekt denselben Namen wie das Objekt hat, das der Formularbenutzer ausgewählt hat, wird Folgendes ausgeführt:

```
if ((Subform1.nodes.item(nCount).className == "field") &
(Subform1.nodes.item(nCount).name == sSelectField)) {
```

Das folgende Skript enthält eine switch-Anweisung zur Steuerung der drei Präsenzoptionen, die ein Formularbenutzer den Textfeldobjekten zuweisen kann:

```
switch(xfa.event.newText) {
case 'Invisible':
Subform1.nodes.item(nCount).presence = "invisible";
break;
case 'Hidden (Exclude from Layout)':
Subform1.nodes.item(nCount).presence = "hidden";
break;
default:
Subform1.nodes.item(nCount).presence = "visible";
break;
}
}
}
```

15.2.3. Skripten für die Präsenzwerte der Schaltflächen

Für das Skript für die Präsenzwerte der Schaltflächen sind zwei Variablen erforderlich. Diese Variable speichert die in Subform1 enthaltene Anzahl von Objekten:

```
var nSubLength = Subform1.nodes.length;
```

Diese Variable speichert den Namen der Schaltfläche, welche der Formularbenutzer in der Dropdown-Liste "Schaltflächen" auswählt:

```
var sSelectButton = buttonList.rawValue;
```

Das folgende Skript verwendet die `replace` -Methode, um alle Leerstellen aus dem Namen der Schaltfläche zu entfernen, die in der `sSelectField` -Variablen gespeichert sind, wodurch der Wert der Dropdown-Liste mit dem Namen des Objekt in der Palette „Verlauf“ übereinstimmt:

```
sSelectButton = sSelecButton.replace(/\s/g, '');
```

Dieses Skript verwendet eine `for` Schleife, um zwischen allen Objekten, die in Subform1 enthalten sind, zu wechseln:

```
für (var nCount = 0; nCount < nSubLength; nCount++) {
```

Wenn das aktuelle Objekt in Subform1 vom Typ `field` ist und denselben Namen wie das Objekt hat, das der Formularbenutzer ausgewählt hat, wird Folgendes ausgeführt:

```
if ((Subform1.nodes.item(nCount).className == "field") &
Subform1.nodes.item(nCount).name == sSelectButton)) {
```

Dieses Skript verwendet eine `switch` -Anweisung zur Steuerung der fünf Präsenzwerte, die ein Formularbenutzer den Schaltflächenobjekten zuweisen kann.

HINWEIS: *Die relevante Eigenschaft gibt jeweils an, ob ein Objekt beim Drucken des Formulars sichtbar sein soll.*

```
switch(xfa.event.newText) {
case 'Invisible':
Subform1.nodes.item(nCount).presence = "invisible";
break;
case 'Hidden (Exclude from Layout)':
Subform1.nodes.item(nCount).presence = "hidden";
break;
case 'Visible (but Don\'t Print)':
Subform1.nodes.item(nCount).presence = "visible";
Subform1.nodes.item(nCount).relevant = "-print";
break;
case 'Invisible (but Print Anyway)':
Subform1.nodes.item(nCount).presence = "invisible";
Subform1.nodes.item(nCount).relevant = "+print";
break;
default:
Subform1.nodes.item(nCount).presence = "visible"; break;
}
}
}
```

15.2.4. Skripterstellung für die Zurücksetzung der Dropdown-Listen

Verwenden Sie die `resetData` -Methode, damit alle Dropdownlisten auf ihre Standardwerte zurückgesetzt werden:

```
xfa.host.resetData();
```

Verwenden Sie die `remerge` Methode und führen Sie den Formularentwurf und die Formulardaten erneut zusammen. In diesem Beispiel stellt die Methode die Objekte im Bereich "Formularobjekte" in ihrem Originalzustand wieder her.

```
xfa.form.remerge();
```

15.3. Objekte aus der Tab-Reihenfolge ausschließen

Dieses Beispiel demonstriert, wie Sie ein Objekt aus der standardmäßigen Tab-Reihenfolge ausschließen. In diesem Beispiel beginnt der Benutzer im TextField1 und navigiert anschließend mit der Tabulatortaste zu TextField2 und TextField3. Das Dropdown-Listenobjekt DropDownList1 wird angezeigt, wenn der Cursor in TextField2 gesetzt wird.

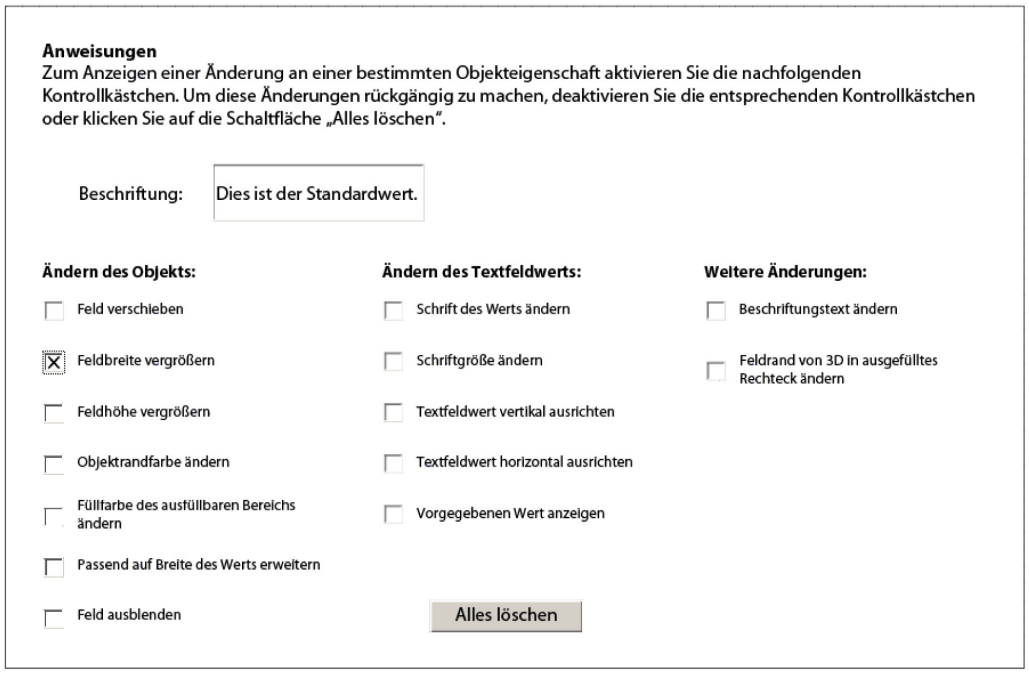
In diesem Fall navigiert der Benutzer standardmäßig in der folgenden Reihenfolge durch das Formular:

Fügen Sie die folgenden Skripten zum Objekt TextField2 hinzu, um das Objekt DropDownList1 aus der Tab-Reihenfolge auszuschließen:

Ereignis	Skript
enter	`// Diese bedingte Anweisung zeigt DropDownList3 dem Benutzer // und legt den Fokus auf die Client-Anwendung TextField2. if (DropDownList3.presence != "visible") { DropDownList3.presence = "visible"; xfa.host.setFocus(this); }`
exit	`// Diese bedingte Anweisung überprüft, ob der Benutzer während dem // Drücken der Tabulatortaste die Umschalttaste drückt. Wird die Umschalttaste // gedrückt, legt die Kundenanwendung den Fokus auf // TextField1 zurück, ansonsten wird der Fokus auf TextField3 gelegt. Der // Benutzer erkennt an diesen Vorgang, dass die DropDownList3 kein // Teil der Tabulatorreihenfolge ist. var isShiftDown = xfa.event.shift; if (isShiftDown) { xfa.host.setFocus(TextField1); } else { xfa.host.setFocus(textField3); }`

15.4. Visuelle Eigenschaften von Objekten im Client ändern

Dieses Beispiel zeigt, wie Sie die visuellen Eigenschaften eines Objekts (in diesem Fall ein Textfeld) ändern. Wenn ein Benutzer beispielsweise das Kontrollkästchen "Feldbreite vergrößern" aktiviert, wird der Füllbereich des Textfelds auf vier Zoll erweitert.



HINWEIS: *Um die visuellen Eigenschaften von Objekten auf dem Client zu ändern, müssen Sie Ihr Formular im Format „Acrobat (Dynamisch) XML-Formular“ speichern.*

In diesem Beispiel haben die Kontrollkästchen keine eindeutigen Objektnamen. Designer weist daher einen Instanzwert zu, damit ein Objekt referenziert werden kann. Das Skript für Kontrollkästchen enthält eine `if-else` -Anweisung, um die Aktivierung und Deaktivierung zu ermöglichen.

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.4.1. Skripten für das Kontrollkästchen "Feld verschieben"

Wenn das Kontrollkästchen aktiviert wird, wird das Feld den x- und y-Einstellungen entsprechend verschoben. Wenn das Kontrollkästchen deaktiviert wird, wird das Feld an seine ursprüngliche Position zurückgesetzt.

```
if (CheckBox1.rawValue == true) {
TextField.x = "3.0in";
TextField.y = "3.5in";
}
else
{
TextField.x = "1in";
TextField.y = "3in";
}
```

15.4.2. Skripten für das Kontrollkästchen "Feldbreite vergrößern"

Wenn das Kontrollkästchen aktiviert wird, wird die Feldbreite auf 4 Zoll erhöht. Wenn das Kontrollkästchen deaktiviert wird, wird die Feldbreite auf 2,5 Zoll verringert.

```
if (CheckBox2.rawValue == true)
TextField.w = "4in";
else
TextField.w = "2.5in";
```

15.4.3. Skripterstellung für das Kontrollkästchen „Feldhöhe vergrößern“

Wenn das Kontrollkästchen aktiviert wird, wird die Feldhöhe auf 1,5 Zoll gesteigert. Wenn das Kontrollkästchen deaktiviert wird, wird die Feldhöhe auf 0,5 Zoll verringert.

```
if (CheckBox3.rawValue == true)
TextField.h = "1.5in";
else
TextField.h = "0.5in";
```

15.4.4. Skripten für das Kontrollkästchen "Objektrandfarbe ändern"

Wenn das Kontrollkästchen aktiviert wird, wird dem Feldrand die Farbe Rot zugewiesen. Wenn das Kontrollkästchen deaktiviert wird, wird dem Feldrand die Farbe Weiß zugewiesen.

```
if (CheckBox4.rawValue == true)
TextField.border.edge.color.value = "255,0,0";
else
TextField.border.edge.color.value = "255,255,255";
```

15.4.5. Skripten für das Kontrollkästchen "Füllfarbe des ausfüllbaren Bereichs ändern"

Wenn das Kontrollkästchen aktiviert wird, wird dem Füllbereich des Textfelds die Farbe Grün zugewiesen. Wenn das Kontrollkästchen deaktiviert wird, wird dem Füllbereich des Textfelds die Farbe Weiß zugewiesen.

```
if (CheckBox5.rawValue == true) {
xfa.resolveNode("TextField.ui.#textEdit.border.fill.color").value = "0,255,0";
}
else {
xfa.resolveNode("TextField.ui.#textEdit.border.fill.color").value =
"255,255,255";
}
```

15.4.6. Skripterstellung für das Kontrollkästchen „Passend auf Breite des Werts erweitern“

Wenn das Kontrollkästchen aktiviert wird, wird der Füllbereich des Textfelds an den Wert angepasst. Wenn das Kontrollkästchen deaktiviert wird, wird der Füllbereich des Textfelds nicht an den Wert angepasst.

```
if (CheckBox6.rawValue == true)
TextField.minW = "0.25in";
else
TextField.maxW = "2.5in";
```

15.4.7. Skripterstellung für das Kontrollkästchen „Feld ausblenden“

Wenn das Kontrollkästchen aktiviert wird, wird das Feld ausgeblendet. Wenn das Kontrollkästchen deaktiviert wird, ist das Feld sichtbar.

```
if (CheckBox7.rawValue == true)
TextField.presence = "hidden";
else
TextField.presence = "visible";
```

15.4.8. Skripten für das Kontrollkästchen "Schrift des Werts ändern"

Wenn das Kontrollkästchen aktiviert wird, wird dem Wert die Schrift Courier New zugewiesen. Wenn das Kontrollkästchen deaktiviert wird, wird dem Wert die Schrift Myriad Pro zugewiesen.

```
if (CheckBox8.rawValue == true)
TextField.font.typeface = "Courier New";
else
TextField.font.typeface = "Myriad Pro";
```

15.4.9. Skripterstellung für das Kontrollkästchen „Schriftgröße ändern“

Wenn das Kontrollkästchen aktiviert wird, wird die Schriftgröße auf 14 Pt eingestellt. Wenn das Kontrollkästchen deaktiviert wird, wird die Schriftgröße auf 10 Pt eingestellt.

```
if (CheckBox9.rawValue == true)
TextField.font.size = "14pt";
else
TextField.font.size = "10pt";
```

15.4.10. Skripterstellung für das Kontrollkästchen „Textfeldwert vertikal ausrichten“

Wenn das Kontrollkästchen aktiviert wird, wird der Textfeldwert an der oberen Kante ausgerichtet. Wenn das Kontrollkästchen deaktiviert wird, wird der Textfeldwert an der Mitte ausgerichtet.

```
if (CheckBox10.rawValue == true)
TextField.para.vAlign = "top";
else
TextField.para.vAlign = "middle";
```

15.4.11. Skripterstellung für das Kontrollkästchen „Textfeldwert horizontal ausrichten“

Wenn das Kontrollkästchen aktiviert wird, wird der Textfeldwert an der Mitte ausgerichtet. Wenn das Kontrollkästchen deaktiviert wird, wird der Textfeldwert an der linken Kante ausgerichtet.

```
if (CheckBox11.rawValue == true)
TextField.para.hAlign = "center";
else
TextField.para.hAlign = "left";
```

15.4.12. Skripterstellung für das Kontrollkästchen „Vorgegebenen Wert anzeigen“

Wenn das Kontrollkästchen aktiviert wird, wird im Textfeld ein durch ein Skript definierter Wert angezeigt. Wenn das Kontrollkästchen deaktiviert wird, wird im Textfeld der (ebenfalls durch ein Skript definierte) Standardwert angezeigt.

```
if (CheckBox12.rawValue == true)
TextField.rawValue = "This is a value set using a script.";
else
TextField.rawValue = "Dies ist ein Standardwert.";
```

15.4.13. Skripten für das Kontrollkästchen "Beschriftungstext ändern"

Wenn das Kontrollkästchen aktiviert wird, wird der durch ein Skript definierte, alternative Beschriftungstext angezeigt. Wenn das Kontrollkästchen deaktiviert wird, wird die (ebenfalls durch ein Skript definierte) Standardbeschriftung angezeigt.

```
if (CheckBox13.rawValue == true)
xfa.resolveNode("TextField.caption.value.#text").value = "Alternate Caption:";
else
xfa.resolveNode("TextField.caption.value.#text").value = "Caption:";
```

15.4.14. Skripterstellung für das Kontrollkästchen „Feldrand von 3D in ausgefülltes Rechteck ändern“

Wenn das Kontrollkästchen aktiviert wird, wird der Feldrand in ein ausgefülltes Rechteck geändert. Wenn das Kontrollkästchen deaktiviert wird, wird dem Feldrand ein 3D-Stil zugewiesen.

```
if (CheckBox14.rawValue == true)
xfa.resolveNode("TextField.ui.#textEdit.border.edge").stroke = "solid";
else
xfa.resolveNode("TextField.ui.#textEdit.border.edge").stroke = "lowered";
```

15.4.15. Skripten für die Schaltfläche "Alle Kontrollkästchen deaktivieren"

Verwenden Sie die `resetData` -Methode, um sämtliche Kontrollkästchen auf ihren Standardwert (Aus) zurücksetzen.

```
xfa.host.resetData();
```

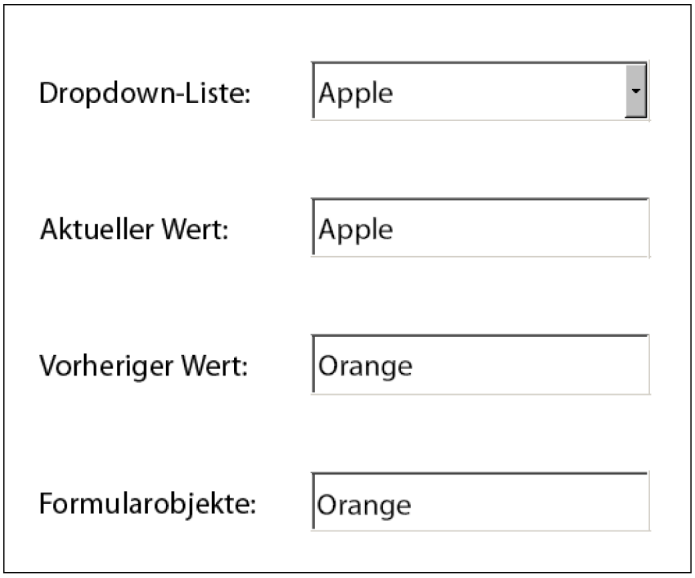
Verwenden Sie die `remerge` Methode und führen Sie den Formularentwurf und die Formulardaten erneut zusammen. In diesem Fall stellt die Methode das Textfeld in seinem Originalzustand wieder her.

```
xfa.form.remerge();
```

15.5. Aktuellen oder vorherigen Wert einer Dropdown-Liste abrufen

Dieses Beispiel zeigt, wie Sie den aktuellen Wert einer Dropdown-Liste abrufen und welche Möglichkeiten Sie haben, auf den vorherigen Wert einer Dropdown-Liste in einem Formular zuzugreifen. Abgesehen von den eigentlichen Skripten, durch die die aktuellen und vorherigen Werte festgelegt werden, ist auch wichtig, dass sich die Skripten im `change` Ereignis für die Dropdownliste befinden.

Im folgenden Beispiel wird, wenn ein Formularbenutzer in der Dropdown-Liste einen Wert auswählt, der ausgewählte Wert im Feld "Aktueller Wert" angezeigt. Wenn der Formularbenutzer dann einen anderen Wert in der Dropdown-Liste auswählt, wird dieser neue Wert in der Liste "Aktueller Wert" angezeigt und der vorherige Wert im Feld "Vorheriger Wert 1".



HINWEIS: *Die verschiedenen Methoden zum Abrufen des vorherigen Wertes einer Dropdown-Liste beruhen jeweils auf einem anderen Skript. Das Textfeld „Vorheriger Wert 1“ wird durch eine direkte Referenz auf die `rawValue` -Eigenschaft der Dropdown-Liste befüllt, wobei das Textfeld „Vorheriger Wert 2“ hingegen mit der `prevText` Eigenschaft befüllt wird. Damit Sie konsistente Ergebnisse erzielen, sollten Sie auf den vorherigen Wert über die Eigenschaft `prevText` zugreifen.*

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.5.1. Skripterstellung zum Ausfüllen des Textfelds „Aktueller Wert“

Wenn Sie das Textfeld „Aktueller Wert“ befüllen möchten, verwenden Sie die `newText` -Eigenschaft:

```
CurrentValue.rawValue = xfa.event.newText;
```

15.5.2. Skripten zum Ausfüllen des Textfelds "Vorheriger Wert 1"

Wenn Sie das Textfeld „Vorheriger Wert 1“ befüllen möchten, verwenden Sie `rawValue` Dropdownliste aus:

```
PreviousValue1.rawValue = DropDownList.rawValue;
```

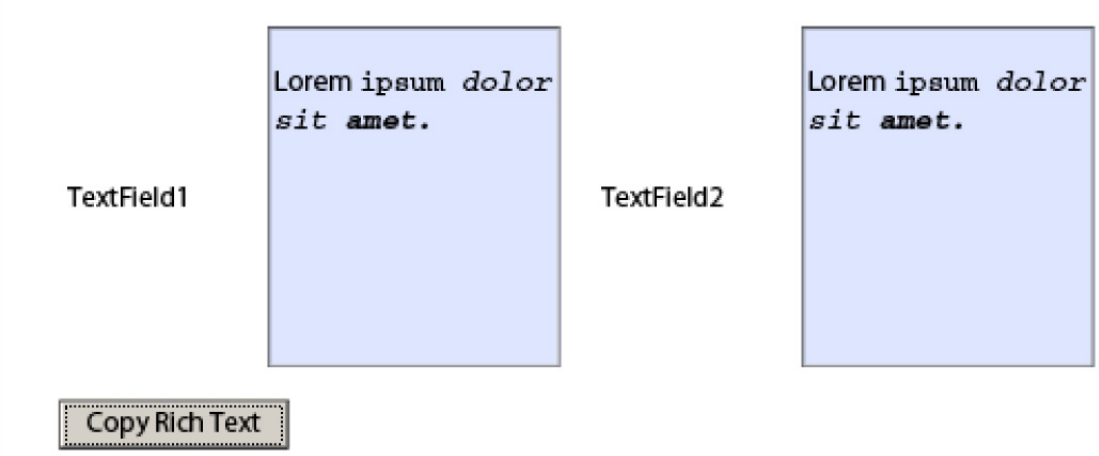
15.5.3. Skripten zum Ausfüllen des Textfelds "Vorheriger Wert 2"

Wenn Sie das Textfeld „Vorheriger Wert 2“ ausfüllen möchten, verwenden Sie die `prevText` -Eigenschaft:

```
PreviousValue2.rawValue = xfa.event.prevText;
```

15.6. Beim Kopieren von Feldwerten die Rich-Text-Formatierung beibehalten

Dieses Beispiel demonstriert, wie Sie die Rich-Text-Formatierung von Felddaten beim Kopieren von Werten zwischen Feldern beibehalten.



TextField1 und TextField2 sind so konfiguriert, dass mehrere Zeilen zugelassen werden. Außerdem zeigen die Felder die Rich-Text-Formatierung an.

Die Schaltfläche „Rich-Text kopieren“ kopiert den Wert von TextField1, einschließlich der Rich-Text-Formatierung, und fügt ihn in TextField2 ein.

15.6.1. Skripten für die Schaltfläche „Rich-Text kopieren“

Rich-Text-Feldwerte werden im XML-Format in einem untergeordneten Objekt des Felds gespeichert, das den Wert enthält. Das folgende Skript im click-Ereignis der Schaltfläche „Rich-Text kopieren“ verwendet die saveXML-Methode zum Speichern der XML-Definition des Rich-Text-Werts. Anschließend werden die XML-Daten in das entsprechende untergeordnete Objekt von TextField2 geladen.

```
    var richText = TextField1.value.exData.saveXML();
TextField2.value.exData.loadXML(richText,1,1);
```

In diesem Beispiel überschreibt der Rich-Text-Wert den bestehenden Wert von TextField2. Wenn Sie das Skript wie folgt anpassen, werden die Rich-Text-Daten an den aktuellen Wert von TextField2 angehängt:

```
var richText = TextField1.value.exData.saveXML();
TextField2.value.exData.loadXML(richText,1,0);
```

15.7. Höhe eines Feldes zur Laufzeit anpassen

Dieses Beispiel zeigt, wie Sie ein Feld erweitern, um es an die Höhe des Inhalts in einem anderen Feld anzupassen.

In diesem Beispiel wird, wenn der Formularbenutzer in TextField1 mehrere Zeilen eingibt und dann auf die Schaltfläche "Erweitern" klickt, die Höhe von TextField2 vergrößert und somit an die Höhe von TextField1 angepasst.

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.7.1. Skripten für die Schaltfläche "Erweitern"

Das folgende Skript wurde für die Schaltfläche "Erweitern" erstellt:

```
var newHeight = xfa.layout.h(TextField1, "in");
TextField2.h = newHeight + "in";
```

15.8. Felder zur Laufzeit als „Erforderlich“ festlegen

Dieses Beispiel zeigt, wie Sie ein Feld zur Laufzeit als "Erforderlich" festlegen.

In diesem Beispiel wird eine entsprechende Fehlermeldung eingeblendet, wenn auf die Schaltfläche "Als 'Erforderlich' festlegen" geklickt wurde und ein Formularbenutzer versucht, ein Formular zu senden, ohne vorher das Feld "TextField1" auszufüllen.

Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.8.1. Skripten für die Schaltfläche "Als 'Erforderlich' festlegen"

Das folgende Skript wurde für die Schaltfläche "Als 'Erforderlich' festlegen" erstellt:

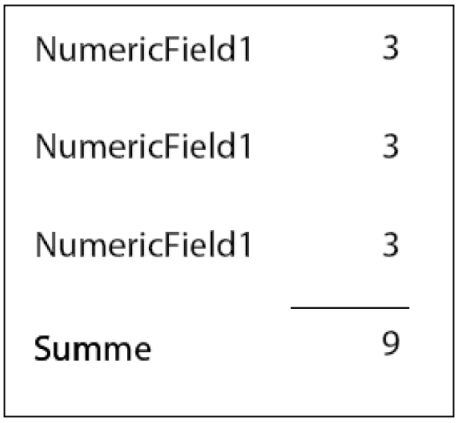
```
TextField1.validate.nullTest = "error";
```

Alternativ können Sie auch eines der beiden folgenden Skripten verwenden:

```
TextField1.mandatory = "error"
TextField1.mandatoryMessage = "this field is mandatory!"
```

15.9. Feldsummen berechnen

Dieses Beispiel zeigt, wie Sie die Summen von Feldern auf unterschiedlichen Ebenen der Formularhierarchie berechnen, wenn der Formularbenutzer das Formular in einer Client-Anwendung wie Acrobat Professional, Adobe Reader oder einem HTML-Client öffnet.



Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.9.1. Skripten zur Berechnung der Summe sich wiederholender Felder in einem Formular

Um die Summe sich wiederholender Felder in einem Formular zu berechnen, fügen Sie das `calculate` -Ereignis in das Summenfeld hinzu:

```
var fields = xfa.resolveNodes("NumericField1[*]");

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
total = total + fields.item(i).rawValue;
}

this.rawValue = total;
```

15.9.2. Skripten zur Berechnung der Summe sich wiederholender Felder

Zur Berechnung der Summe sich wiederholender Felder fügen Sie das `calculate` -Ereignis in das Summenfeld hinzu:

```
var fields = xfa.resolveNodes("detail[*].NumericField1");

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
total = total + fields.item(i).rawValue;
}

this.rawValue = total;
```

15.9.3. Skripten zur Berechnung der Summe der Felder auf der Seite

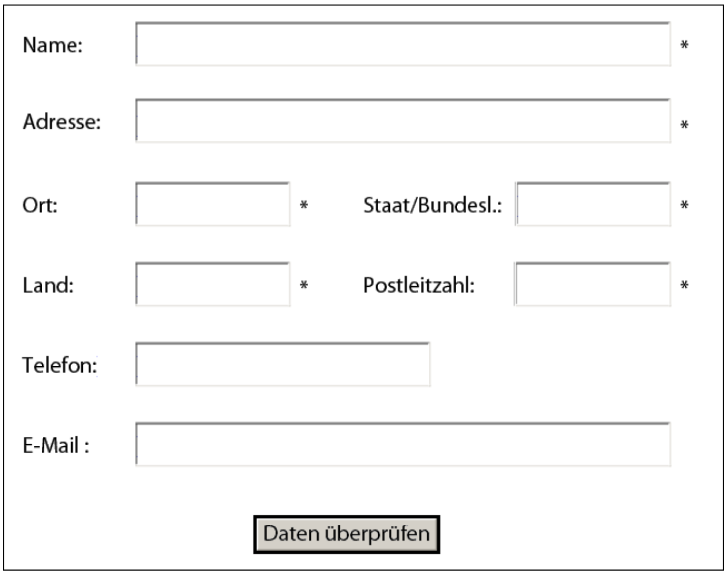
Um die Summe der Felder auf der Seite zu berechnen, fügen Sie ein `calculate` -Ereignis in das Summenfeld hinzu:

```
var fields = xfa.layout.pageContent(0 , "field", 0);

var total = 0;
for (var i=0; i <= fields.length-1; i++) {
if (fields.item(i).name == "NumericField1") {
total = total + fields.item(i).rawValue;
}
}

this.rawValue = total;
```

15.10. Felder als Reaktion auf Benutzeraktionen hervorheben

Dieses Beispiel zeigt, wie Sie das aktuelle Feld hervorheben, das ein Formularbenutzer gerade bearbeitet, wie Sie die Felder hervorheben, die Formularbenutzer ausfüllen müssen, und wie Sie den Formularbenutzern Feedback in Form von Meldungsfeldern geben können.

In diesem Beispiel erscheint rechts neben den erforderlichen Feldern ein Sternchen (*). Bei Auswahl eines Felds wird dem Feldrand die Farbe Blau zugewiesen. Wenn ein Formularbenutzer auf die Schaltfläche "Daten überprüfen" klickt, aber nicht alle erforderlichen Felder ausgefüllt hat, wird eine Meldung angezeigt und das entsprechende Feld rot markiert. Wenn alle erforderlichen Felder ausgefüllt wurden und der Formularbenutzer auf die Schaltfläche "Daten überprüfen" klickt, wird eine Bestätigungsmeldung eingeblendet.



Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.10.1. Skripten zum Kennzeichnen eines ausgewählten Feldes mit einem blauen Rand

Zum Kennzeichnen des ausgewählten Feldes durch einen blauen Rand fügen Sie allen Textfeldern die folgenden Skripten hinzu:

Geben Sie beispielsweise ein `enter` -Ereignis in das Feld „Name“ ein:

```
Name.border.edge.color.value = "0,0,255";
```

Geben Sie beispielsweise ein `exit` -Ereignis in das Feld „Name“ ein:

```
Name.border.edge.color.value = "255,255,255";
```

Fügen Sie beispielsweise ein `mouseEnter` -Ereignis in das Feld „Name“ ein:

```
Name.border.edge.color.value = "0,0,255";
```

Fügen Sie beispielsweise ein `mouseExit` -Ereignis in das Feld „Name“ ein:

```
Name.border.edge.color.value = "255,255,255";
```

15.10.2. Skripten für die Schaltfläche "Daten überprüfen"

Das folgende Skript, das speziell für die Schaltfläche "Daten überprüfen" konzipiert wurde, stellt anhand einiger Überprüfungsschritte fest, ob die erforderlichen Felder Daten enthalten. Dabei wird jedes Feld einzeln überprüft und festgestellt, ob der Feldwert nicht null oder eine leere Zeichenfolge ist. Ist der Feldwert null oder eine leere Zeichenfolge, wird ein Hinweis eingeblendet, welcher den Benutzer daran erinnert, dass dieses Feld ausgefüllt werden muss. Gleichzeitig wird die Hintergrundfarbe des Füllbereichs in Rot geändert.

Verwenden Sie diese Variable, um anzugeben, ob ein Feld keine Daten enthält:

```
var iVar = 0;

if ((Name.rawValue == null) || (Name.rawValue == "")) {
xfa.host.messageBox("Geben Sie einen Namenin das Feld „Name“ ein.);
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode("Name.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("Name.ui.#textEdit.border.fill.color").value = "255,100,50";

// Legen Sie die Variable fest, um anzuzeigen, dass dieses Feld keine Daten
enthält. iVar = 1;
}
else
{
// Setzen Sie den füllbaren Bereich des Textfelds zurück.
xfa.resolveNode("Name.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("Name.ui.#textEdit.border.fill.color").value = "255,255,255";
}
```

```
if ((Address.rawValue == null) || (Address.rawValue == "")) {
xfa.host.messageBox("Please enter a value in the Address field.");
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode ("Address.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("Address.ui.#textEdit.border.fill.color").value = "255,100,50";
```

Dieses Skript stellt die Variable ein, die angibt, dass dieses Feld keine Daten enthält:

```
iVar = 1;
}
else {
```

Dieses Skript setzt den Füllbereich des Textfelds zurück:

```
xfa.resolveNode ("Address.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("Address.ui.#textEdit.border.fill.color").value =
"255,255,255";
}

if ((City.rawValue == null) || (City.rawValue == "")) {
xfa.host.messageBox("Please enter a value in the City field.");
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode("City.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("City.ui.#textEdit.border.fill.color").value = "255,100,50";
```

Dieses Skript stellt die Variable ein, die angibt, dass dieses Feld keine Daten enthält:

```
iVar = 1;
}
else {
```

Dieses Skript setzt den Füllbereich des Textfelds zurück:

```
xfa.resolveNode("City.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("City.ui.#textEdit.border.fill.color").value = "255,255,255";
}

if ((State.rawValue == null) || (State.rawValue == "")) {
xfa.host.messageBox("Please enter a value in the State field.");
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode("State.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("State.ui.#textEdit.border.fill.color").value = "255,100,50";
```

Dieses Skript stellt die Variable ein, die angibt, dass dieses Feld keine Daten enthält:

```
iVar = 1;
}
else {
```

Dieses Skript setzt den Füllbereich des Textfelds zurück:

```
xfa.resolveNode("State.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("State.ui.#textEdit.border.fill.color").value = "255,255,255";
}

if ((ZipCode.rawValue == null) || (ZipCode.rawValue == "")) {
xfa.host.messageBox("PGeben Sie einen Wert in das Feld für die Postleitzahl
ein.");
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode("ZipCode.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("ZipCode.ui.#textEdit.border.fill.color").value = "255,100,50";
```

Dieses Skript stellt die Variable ein, die angibt, dass dieses Feld keine Daten enthält:

```
iVar = 1;
}
else {
```

Dieses Skript setzt den Füllbereich des Textfelds zurück:

```
xfa.resolveNode("ZipCode.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("ZipCode.ui.#textEdit.border.fill.color").value =
"255,255,255";
}

if ((Country.rawValue == null) || (Country.rawValue == "")) {
xfa.host.messageBox("Geben Sie einen Wert in das Feld für das Land ein.");
```

Dieses Skript ändert die Farbe des Füllbereichs des Textfelds:

```
xfa.resolveNode("Country.ui.#textEdit.border.edge").stroke = "solid";
xfa.resolveNode("Country.ui.#textEdit.border.fill.color").value = "255,100,50";
```

Dieses Skript stellt die Variable ein, die angibt, dass dieses Feld keine Daten enthält.

```
iVar = 1;
}
else {
```

Dieses Skript setzt den Füllbereich des Textfelds zurück:

```
xfa.resolveNode("Country.ui.#textEdit.border.edge").stroke = "lowered";
xfa.resolveNode("Country.ui.#textEdit.border.fill.color").value =
"255,255,255"; }
```

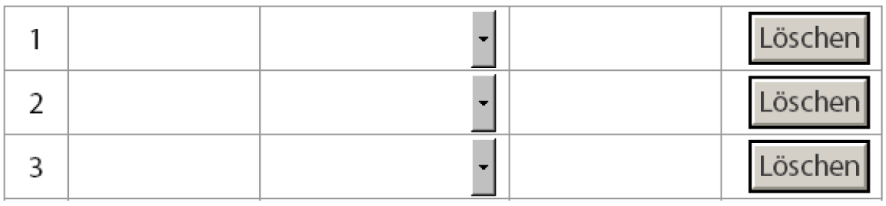
Wenn alle erforderlichen Felder Daten enthalten, wird die `iVar` Variable auf null gesetzt und eine Bestätigungsmeldung angezeigt:

```
if (iVar == 0) {
xfa.host.messageBox("Vielen Dank für die Eingabe Ihrer Informationen.");
}
```

15.11. Die Werte des aktuellen Teilformulars zurücksetzen

Dieses Beispiel zeigt, wie Sie die Werte eines bestimmten Satzes von Feldern zurücksetzen, nicht aber die Werte des gesamten Formulars. Dazu setzen Sie nur die Felder im erforderlichen Teilformularobjekt zurück.

In diesem Beispiel kann der Formularbenutzer die Feldwerte durch Klicken auf die Schaltfläche "Löschen" zurücksetzen.



Zum Anzeigen dieser Beispieldatei und anderer rufen Sie das Developer Center.

15.11.1. Skripten für die Werte in der linken Spalte

Geben Sie dieses Skript für die Werte ein, die in der linken Spalte angezeigt werden:

```
this.rawValue = this.parent.index + 1;
```

Um die Standardwerte zurückzusetzen, fügen Sie ein `click` -Ereignis zur Schaltfläche „Löschen“ hinzu. Sie benötigen einen dynamischen Referenz-Syntax-Ausdruck, weil es sich hier um ein sich wiederholendes Teilformular handelt, was im Referenz-Syntax-Ausdruck berücksichtigt werden muss. In diesem Fall ist es einfacher, die `resetData` Parameter einzeln zu erstellen.

```
var f1 = this.parent.somExpression + ".TextField2" + ",";
var f2 = f1 + this.parent.somExpression + ".DropDownList1" + ",";
var f3 = f2 + this.parent.somExpression + ".NumericField1";

// ...und geben Sie die Variable als einen Parameter weiter.
xfa.host.resetData(f3);
```

15.12. Präsenz von Formularentwurfsobjekten ändern

Designer bietet auf verschiedenen Registerkarten der Palette „Objekt“ die folgenden Präsenzeinstellungen für die Objekte in einem Formular. Die Einstellungen "Unsichtbar" und "Ausgeblendet (Aus Layout ausschließen)" sind nicht für Gruppen, Inhaltsbereiche, Masterseiten, Seitensätze und Objekte von Teilformularsätzen verfügbar.

HINWEIS: *Um die Präsenzeinstellung eines Objekts mit Hilfe eines Skripts zu ändern, müssen Sie den Wert der zwei zugrunde liegenden Eigenschaften des XML-Formularobjektmodells* `presence` *und* `relevant` *ändern.*

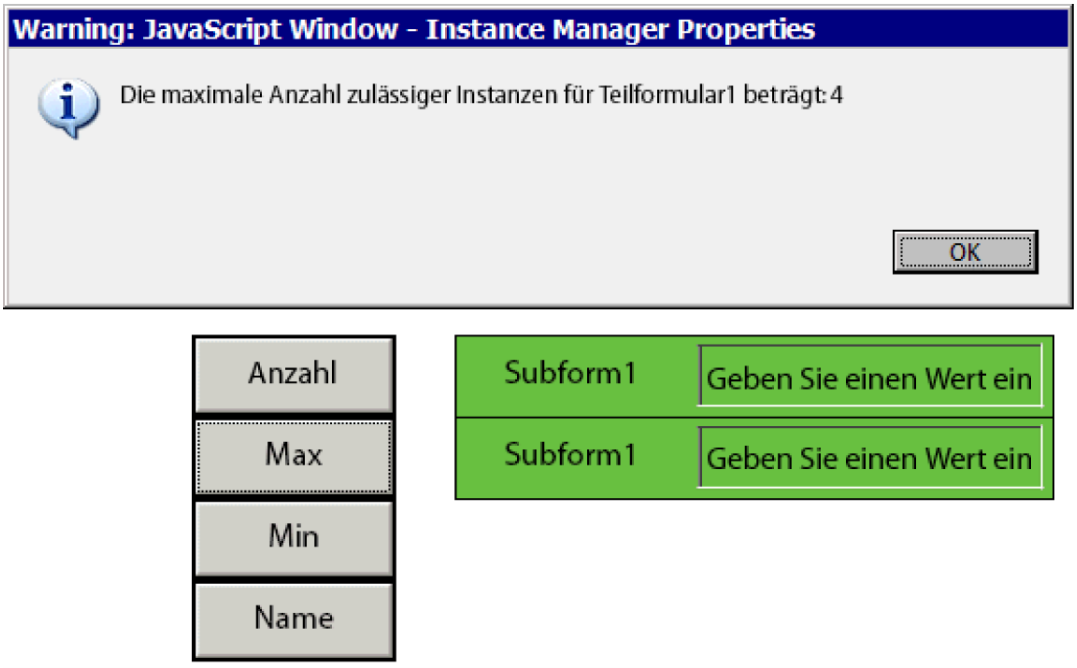
In der folgenden Tabelle finden Sie eine Übersicht über die Präsenzeinstellungen und die zugehörige Referenz-Syntax.

Präsenzeinstellung	Referenz-Syntax
Sichtbar	FormCalc *ObjectName*.presence = "visible" JavaScript *ObjectName*.presence = "visible";
Sichtbar (nur Bildschirm)	FormCalc *ObjectName*.presence = "visible" *ObjectName*.relevant = "-print" JavaScript *ObjectName*.presence = "visible"; *ObjectName*.relevant = "-print";
Sichtbar (nur drucken)	FormCalc *ObjectName*.presence = "visible" *ObjectName*.relevant = "+print" JavaScript *ObjectName*.presence = "visible"; *ObjectName*.relevant = "+print";
Unsichtbar	FormCalc *ObjectName*.presence = "invisible" JavaScript *ObjectName*.presence = "invisible";
Ausgeblendet (Aus Layout ausschließen)	FormCalc *ObjectName*.presence = "hidden" JavaScript *ObjectName*.presence = "hidden";
Nur einseitiger Druck	FormCalc *ObjectName*.presence = "simplex" JavaScript *ObjectName*.presence = "simplex";
Nur zweiseitiger Druck	FormCalc *ObjectName*.presence = "duplex" JavaScript *ObjectName*.presence = "duplex";

15.13. Teilformulare mit Hilfe der Eigenschaften des Instanzmanagers steuern

Dieses Beispiel zeigt, wie Sie mit den Eigenschaften des Instanzmanagers (der zum XML Form Object Model gehört) zur Laufzeit Informationen über Teilformulare abrufen.

Im folgenden Formular nutzen die vier Schaltflächen die Skripteigenschaften des Instanzmanagers und liefern Informationen über Subform1. Wenn ein Formularbenutzer beispielsweise auf die Schaltfläche "Max" klickt, wird eine Meldung eingeblendet, welche die maximal zulässige Anzahl unterstützter Subform1-Instanzen angibt.



15.13.1. Skripten zur Ausgabe des Wertes der Eigenschaft "count" im Meldungsfeld

Das folgende Skript verwendet die `messageBox` -Methode, um den Wert der `count` -Eigenschaft auszugeben:

```
xfa.host.messageBox("Die aktuelle Anzahl von Subform1-Instanzen im Formular ist:"
+ properties.Subform1.instanceManager.count, "Instanz-Manager-Eigen-
schaften",3);
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die `count` -Eigenschaft des Instanzmanagers zu referenzieren, wie hier dargestellt:

```
xfa.host.messageBox("Die aktuelle Anzahl von Subform1-Instanzen im Formular
ist:" + properties._Subform1.count, "Instanz-Manager-Eigenschaften", 3);
```

Die Unterstrich-Notation (_) ist besonders wichtig, wenn das Formular zurzeit keine Teilformularinstanzen enthält.

15.13.2. Skripten zur Ausgabe des Wertes der Eigenschaft "max" im Meldungsfeld

Das folgende Skript verwendet die `messageBox` -Methode, um den Wert der `max` -Eigenschaft auszugeben:

```
xfa.host.messageBox ("Die maximale Anzahl von Instanzen, die für Subform1
zulässig sind, lautet: "+ properties.Subform1.instanceManager.max,
"Instanz-Manager-Eigenschaften", 3);
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die `max` -Eigenschaft des Instanzmanagers zu referenzieren, wie hier dargestellt:

```
xfa.host.messageBox ("Die maximale Anzahl von Instanzen, die für Subform1
zulässig sind, lautet: " + properties._Subform1.max, "Instanz-Manager-Eigen-
schaften", 3);
```

15.13.3. Skripten zur Ausgabe des Wertes der Eigenschaft "min" im Meldungsfeld

Das folgende Skript verwendet die `messageBox` -Methode, um den Wert der `min` -Eigenschaft auszugeben:

```
xfa.host.messageBox ("Die Mindestanzahl der Instanzendie für Subform1 zulässig
sind, lautet: "+ properties.Subform1.instanceManager.min,
"Instanz-Manager-Eigenschaften", 3);
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die `min` -Eigenschaft des Instanzmanagers zu referenzieren, wie hier dargestellt:

```
xfa.host.messageBox ("Die Mindestanzahl von Instanzen, die für Subform1 zulässig
sind, lautet: " + properties._Subform1.min, "Instanz-Manager-Eigenschaften", 3);
```

15.13.4. Skripten zur Ausgabe des Namens der Teilformulareigenschaft im Meldungsfeld

Das folgende Skript verwendet die `messageBox` -Methode, um den Namen der `subform` -Eigenschaft auszugeben:

```
xfa.host.messageBox ("Der Namen des Teilformulars, der die
Instanz-Manager-Namenseigenschaft verwendet, lautet: " + proper-
ties.Subform1.instanceManager.name + ".\n\nHinweis: Dieser Wert unterscheidet
sich von dem Wert, den die Namenseigenschaft des Objekts für Subform1 zurückge-
geben hat. "Instanz-Manager-Eigenschaften", 3);
```

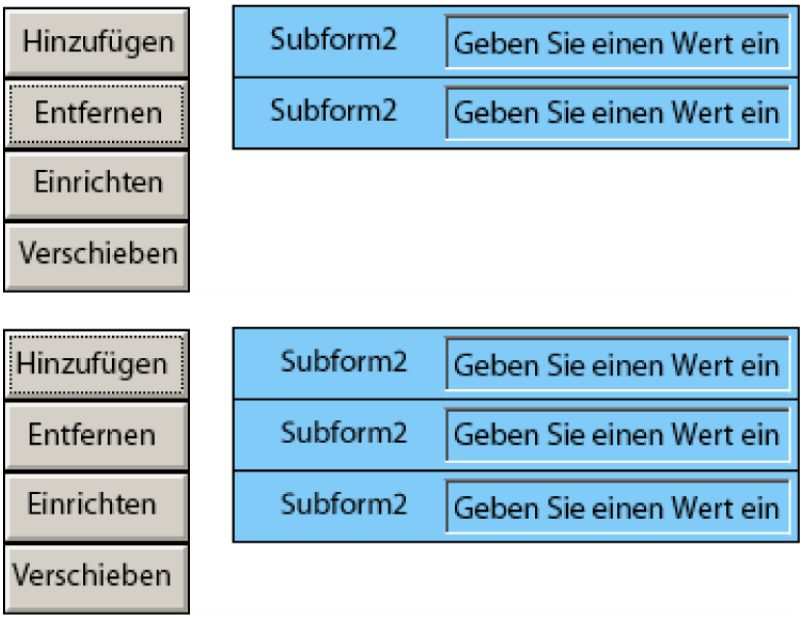
Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaft "name" des Instanzmanagers zu referenzieren, wie in diesem Beispiel:

```
xfa.host.messageBox("Der Namen des Teilformulars, der die
Instanz-Manager-Namenseigenschaft verwendet, lautet: " + properties._Sub-
form1.name + ".\n\nHinweis: Dieser Wert unterscheidet sich von dem Wert, den die
Namenseigenschaft des Objekts für Subform1 zurückgegeben hat.
"Instanz-Manager-Eigenschaften", 3);
```

15.14. Teilformulare mit Hilfe der Methoden des Instanzmanagers steuern

Dieses Beispiel demonstriert, wie Sie mit den Methoden des Instanzmanagers (der zum XML Form Object Model gehört) zur Laufzeit Vorgänge für Teilformularobjekte durchführen. Beispielsweise können Sie Instanzen eines bestimmten Teilformulars, einer Tabelle oder einer Tabellenzeile hinzufügen oder entfernen.

Im folgenden Formular verwendet der Formularbenutzer die vier Schaltflächen, um die verschiedenen Skriptmethoden des Instanzmanagers zu einzusetzen. Wenn der Formularbenutzer beispielsweise auf die Schaltfläche "Hinzufügen" klickt, wird dem Formular eine neue Subform2-Instanz hinzugefügt.



HINWEIS: *Die Schaltfläche "Verschieben" ordnet die ersten beiden Subform2-Instanzen neu an und die Schaltfläche "Einrichten" zeigt die Höchstanzahl der Subform2-Instanzen an. In beiden Fällen müssen Sie möglicherweise Teilformulare hinzufügen oder entfernen oder die Daten in den Textfeldern ändern, um die auf die Subform2-Instanzen angewendeten Änderungen sehen zu können.*

15.14.1. Skripten zum Feststellen, ob einem Formular die Höchstanzahl von Teilformularen hinzugefügt wurde

Das folgende Skript ermittelt, ob die maximal unterstützte Anzahl von Subform2-Instanzen bereits im Formular vorhanden ist. Falls ja, wird eine entsprechende Meldung eingeblendet. Falls nein, wird dem Formular eine neue Subform2-Instanz hinzugefügt.

```
if (methods.Subform2.instanceManager.count ==
methods.Subform2.instanceManager.max) {
xfa.host.messageBox("You have reached the maximum number of items allowed.",
"Instance Manager Methods", 1);
}
else {
methods.Subform2.instanceManager.addInstance(1); xfa.form.recalculate(1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (Methoden._Subform2.count == methods._Subform1.max) {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen.",
"Instanz-Managers-Methode", 1);
}
else {
methods._Subform2.addInstance(1); xfa.form.recalculate(1);
}
```

15.14.2. Skripterstellung zum Feststellen, ob aus dem Formular weitere Teilformulare entfernt werden können

Das folgende Skript ermittelt, ob das Formular Subform2-Instanzen enthält. Falls nein, wird eine entsprechende Meldung eingeblendet. Falls ja, wird die erste Instanz aus dem Formular entfernt.

```
if (methods.Subform2.instanceManager.count == 0) {
xfa.host.messageBox("Es gibt keine Teilformularinstanzen zum Entfernen.",
"Instance Manager Methods", 1);
}
else {
methods.Subform2.instanceManager.removeInstance(0); xfa.form.recalculate(1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (Methoden._Subform2.count == 0) {
xfa.host.messageBox("Es gibt keine Teilforminstanzen zum Entfernen.",
"Instanz-Manager-Methoden", 1);
}
else {
methods._Subform2.removeInstance(0);
xfa.form.recalculate(1);
}
```

15.14.3. Skripten zum Festlegen, dass im Formular vier Teilformularinstanzen erscheinen sollen

Das folgende Skript bewirkt, dass im Formular vier Subform2-Instanzen erscheinen, unabhängig davon, wie viele Instanzen zurzeit vorhanden sind:

```
methods.Subform2.instanceManager.setInstances(4);
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
Methoden._Subform2.setInstances(4);
```

15.14.4. Skripten zum Vertauschen des ersten und zweiten Teilformulars im Formular

Das folgende Skript weist die erste und zweite Subform2-Instanz an, auf dem Formular die Position zu tauschen.

```
methods.Subform2.instanceManager.moveInstance(0,1);
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanzmanagers zu referenzieren, wie in diesem Beispiel:

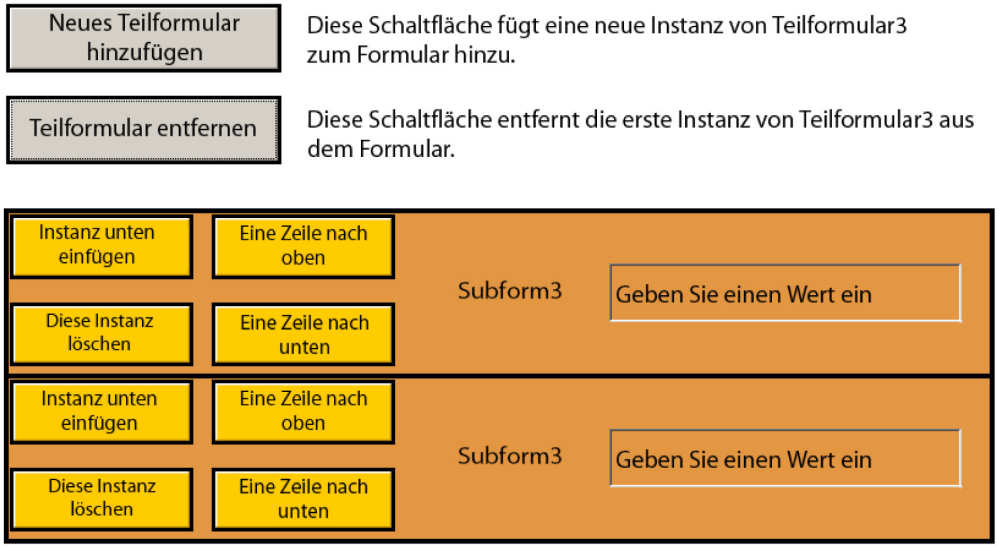
```
Methoden._Subform2.moveInstance(0,1);
```

15.15. Teilformulare mit Hilfe des Instanzmanagers zur Laufzeit steuern

Dieses Beispiel zeigt, wie Sie mit den Eigenschaften und Methoden des Instanzmanagers zur Laufzeit Informationen über Teilformulare abrufen und Vorgänge für Teilformularobjekte durchführen.

In diesem Beispiel verwendet der Formularbenutzer die Schaltflächen, um mit Hilfe von Subform3-Instanzen verschiedene Aktionen auszuführen. Wenn ein Formularbenutzer beispielsweise auf die Schaltfläche "Instanz unten einfügen" klickt, wird unterhalb der aktuellen Instanz eine neue Subform3-Instanz eingefügt.

HINWEIS: *Möglicherweise ist es erforderlich, Teilformulare hinzuzufügen oder zu entfernen oder die Daten im Textfeld ändern, um die auf die Subform3-Instanzen angewendeten Änderungen sehen zu können.*



HINWEIS: *Wenn keine Instanzen eines bestimmten Teilformulars in Ihrem Formular existieren, müssen Sie die Unterstrich-Notation (_) verwenden (siehe nachfolgende Beispiele). Weitere Informationen zur Verwendung der Unterstrich-Notation (_) finden Sie in der Designer-Hilfe.*

15.15.1. Skripten für die Schaltfläche „Neues Teilformular hinzufügen“

Das folgende Skript ermittelt, ob die maximal unterstützte Anzahl von Subform3-Instanzen bereits im Formular vorhanden ist. Falls ja, wird eine entsprechende Meldung eingeblendet. Falls nein, wird dem Formular eine neue Subform3-Instanz hinzugefügt.

```
if (advanced.Subform3.instanceManager.count ==
advanced.Subform3.instanceManager.max) {
xfa.host.messageBox("Sie haben die maximale Anzahl der zulässigen Elemente
erreicht.","Combining Instance Manager Concepts", 1);
}
else {
advanced.Subform3.instanceManager.addInstance(1); xfa.form.recalculate(1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (erweitert._Subform3.count == advanced._Subform3.max) {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen
erreicht.", "Kombination von Instanz-Manager-Konzepten", 1);
}
else {
advanced._Subform3.addInstance(1); xfa.form.recalculate(1);
}
```

15.15.2. Skripten für die Schaltfläche „Teilformular entfernen“

Das folgende Skript ermittelt, ob das Formular Subform3-Instanzen enthält. Falls nein, wird eine entsprechende Meldung eingeblendet. Falls ja, wird die erste Instanz aus dem Formular enthält.

```
if (advanced.Subform3.instanceManager.count == 0) {
xfa.host.messageBox("Es gibt keine Teilforminstanzen zum Entfernen.", "Kombina-
tion von Instanz-Manager-KOnzepten", 1);
}
else {
advanced.Subform3.instanceManager.removeInstance(0);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (erweitert._Subform3.count == 0) {
xfa.host.messageBox("Es gibt keine Teilforminstanzen zum Entfernen.", "Kombina-
tion von Instanz-Manager-Konzepten", 1);
}
else {
advanced._Subform3.removeInstance(0);
}
```

15.15.3. Skripten für die Schaltfläche „Instanz unten einfügen“

Die folgende if-else-Anweisung verhindert die Fortsetzung des Skriptes, falls das Formular zurzeit die maximal zulässige Anzahl von Subform3-Instanzen enthält:

```
if (Subform3.instanceManager.count < Subform3.instanceManager.occur.max) {
//oNewInstance speichert eine Instanz von Subform3, die von der addIns-
tance()-Methode erstellt wurde.
var oNewInstance = Subform3.instanceManager.addInstance(1);
//nIndexFrom und nIndexTo speichern die Vor- und Nach-Index-Werte, um sie mit der
moveInstance()-Methode zu verwenden.
var nIndexFrom = oNewInstance.index; var nIndexTo = Subform3.index + 1;
```

In diesem Fall, wenn das Skript den Wert `nIndexFrom`referenziert, wird die neue Instanz von Subform3 zum Formular an der Position hinzugefügt, die in der `moveInstance` -Methode angegeben ist:

```
Subform3.instanceManager.moveInstance(nIndexFrom, nIndexTo);
}
else {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen
erreicht.", "Kombination von Instanz-Manager-Konzepten", 1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (_Subform3.count < _Subform3.occur.max) {
var oNewInstance = _Subform3.addInstance(1);
var nIndexFrom = oNewInstance.index;
var nIndexTo = Subform3.index + 1;
_Subform3.moveInstance(nIndexFrom, nIndexTo);
}
else {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen
erreicht.", "Kombination von Instanz-Manager-Konzepten", 1);
}
```

15.15.4. Skripten für die Schaltfläche „Diese Instanz löschen“

Die folgende if-else-Anweisung verhindert die Fortsetzung des Skriptes, falls das Formular zurzeit die mindestens erforderliche Anzahl von Subform3-Instanzen enthält:

```
    if (Subform3.instanceManager.count > Subform3.instanceMa-
nager.occur.min) {
```

Dieses Skript verwendet eine `removeInstance` -Methode zum Entfernen einer Instanz von Subform3.

HINWEIS: *Dieses Skript verwendet den Wert `parent.parent.index` um die Subform3-Instanz zu entfernen Die `parent` -Referenz bezeichnet den Container des Objekts, das die Referenz verwendet. In diesem Fall würde die Verwendung der Referenz `parent.index` das unbenannte Teilformular bezeichnen, das die Schaltflächen „Instanz unten einfügen“, „Diese Instanz löschen“, „Eine Zeile nach oben“ und „Eine Zeile nach unten“ enthält.*

```
Subform3.instanceManager.removeInstance(parent.parent.index);
}
else {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen
erreicht.", "Kombination von Instanz-Manager-Konzepten", 1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (_Subform3.count > _Subform3.occur.min) {
Subform3.removeInstance(Subform3.index);
}
else {
xfa.host.messageBox("Sie haben die maximale Anzahl von zulässigen Elementen
erreicht.", "Kombination von Instanz-Manager-Konzepten", 1);
}
```

15.15.5. Skripten für die Schaltfläche „Eine Zeile nach oben“

Die folgende if-else-Anweisung verhindert die Fortsetzung des Skriptes, falls die Instanz von Subform3 als erste Instanz in der Liste aufgeführt wird:

```
if (Subform3.index != 0) {
//nIndexFrom and nIndexTo speichern die Vor- und Nach-Index-Werte, um sie mit der
moveInstance-Methode zu verwenden. var nIndexFrom = Subform3.index; var nIndexTo
= Subform3.index - 1; Subform3.instanceManager.moveInstance(nIndexFrom,
nIndexTo);
}
else {
xfa.host.messageBox("Das aktuelle Element kann nicht verschoben werden, weil es
die erste Instanz in der Liste ist.", "Kombination von
Instanz-Manager-Konzepten", 1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
if (Subform3.index != 0) {
var nIndexFrom = Subform3.index;
var nIndexTo = Subform3.index - 1;
Subform3.moveInstance(nIndexFrom, nIndexTo);
}
else {
xfa.host.messageBox("Das aktuelle Element kann nicht verschoben werden, weil
es die erste Instanz in der Liste ist.", "Kombination von
Instanz-Manager-Konzepten", 1);
}
```

15.15.6. Skripten für die Schaltfläche „Eine Zeile nach unten“

Diese Variable speichert den Indexwert der Subform3-Instanz:

```
var nIndex = Subform3.index;
```

Die folgende if-else-Anweisung verhindert die Fortsetzung des Skriptes, falls die Instanz von Subform3 als letzte Instanz in der Liste aufgeführt wird:

```
if ((nIndex + 1) < Subform3.instanceManager.count) {
// nIndexFrom and nIndexTo speichern die Vor- und Nach-Index-Werte, um sie mit
der moveInstance()-Methode zu verwenden. var nIndexFrom = nIndex; var nIndexTo =
nIndex + 1; Subform3.instanceManager.moveInstance(nIndexFrom, nIndexTo);
}
else {
xfa.host.messageBox("Das aktuelle Element kann nicht verschoben werden, weil es
die erste Instanz in der Liste ist.", "Kombination von
Instanz-Manager-Konzepten", 1);
}
```

Sie können dieses Skript auch mit der Unterstrich-Notation (_) schreiben, um die Eigenschaften und Methoden des Instanz-Managers zu referenzieren, wie in diesem Beispiel:

```
var nIndex = Subform3.index;
if ((nIndex + 1) < Subform3.instanceManager.count) {
var nIndexFrom = nIndex;
var nIndexTo = nIndex + 1;
_Subform3.moveInstance(nIndexFrom, nIndexTo);
}
else {
xfa.host.messageBox("Das aktuelle Element kann nicht verschoben werden, weil es
die erste Instanz in der Liste ist.", "Kombination von
Instanz-Manager-Konzepten", 1);
}
```