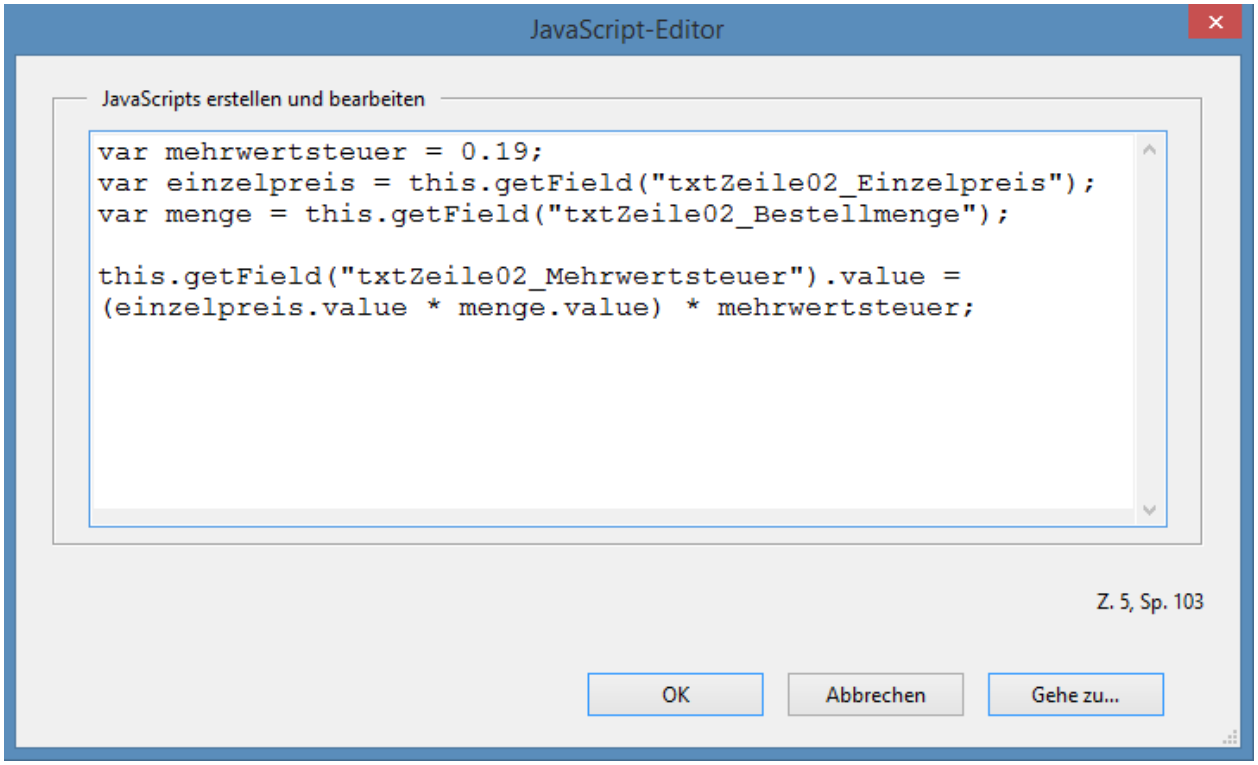
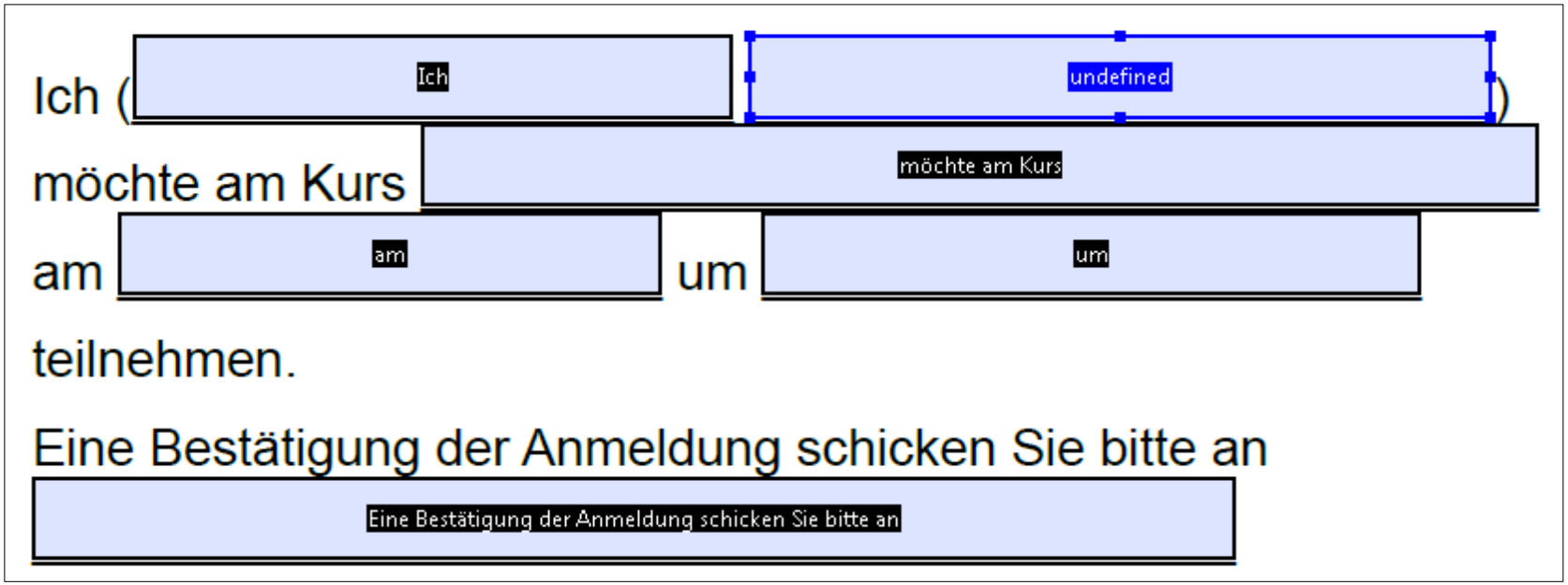
JavaScript in PDF-Formularen



PDF-Formulare

- Elektronisches Abbild eines Papierformulars.
- Ausfüllen eines Formulars am Bildschirm.
- Validierung und Berechnung von Daten in Abhängigkeit der eingegebenen Informationen.
- Entwurf in einem Layout- oder Textverarbeitungsprogramm.

… in der Bearbeitungsansicht



… öffnen

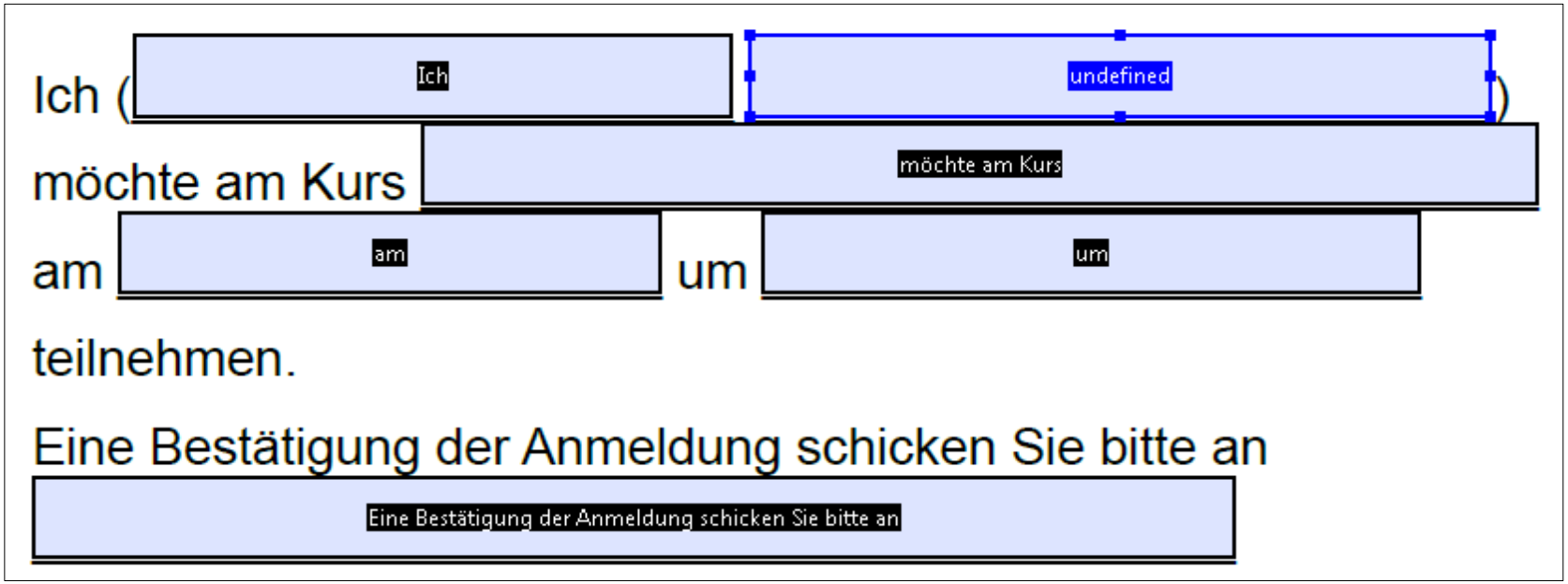
- Voraussetzung: Das Dokument ist als Formular im PDF gespeichert.
- *Werkzeuge – Formular vorbereiten*.
- In der Registerkarte [Dokument...] werden als Symbolleiste alle Feldarten angezeigt.
- Am rechten Rand wird das entsprechende Werkzeugfenster angezeigt.

Bearbeitungsansicht schließen

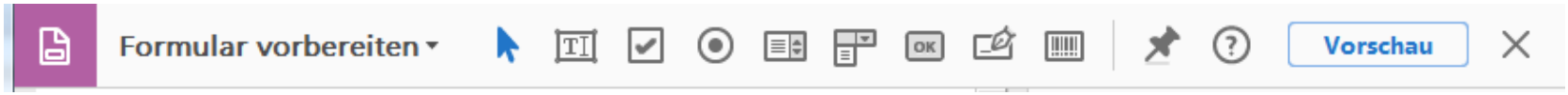
- Mit Hilfe des Kreuzes zu dem Werkzeug [Formular vorbereiten] wird das Werkzeug geschlossen
- Das Formular wird aus der Sicht des Nutzers angezeigt.

Formularfelder in der Bearbeitungsansicht

- Die Formularfelder sind farbig gekennzeichnet.
- Um jedes Formularfeld ist ein schwarzer Rahmen gezogen.
- In der Mitte des Formularfeldes wird der Name des Feldes auf schwarzen Hintergrund angezeigt.



Formularfeld-Typen



- Textfelder zur freien Eingabe oder Berechnung von Daten.
- Kontrollkästchen zur Auswahl von mehreren Elementen aus einer Gruppe.
- Optionsfelder, Listenfelder und DropDown-Listen zur Auswahl eines Elements aus einer Gruppe von Möglichkeiten.
- Schaltflächen, um automatisiert Vorgänge abzuspielen.
- Felder zum digitalen Unterschreiben.
- Barcode-Felder.

Auswahl eines Feldes im Formular

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über ein Formularfeld.
- Mausklick direkt in das Formularfeld.
- Das Formularfeld wird als Ankerfeld gekennzeichnet.

Schaltflächen (Button)

- Anklickbare Schaltfläche.
- Durch einen Mausklick wird eine Aktion gestartet.
- Durch den Mausklick werden automatisiert Anweisungen ausgeführt.

… in Standardgröße dem Formular hinzufügen

- *Werkzeuge – Formular vorbereiten*.
- Klick auf das Icon [Schaltfläche] in der Symbolleiste [Formular vorbereiten].
- Mausklick an der Einfügeposition der Schaltfläche.
- Im Dialog wird der Name der Schaltfläche eingegeben. Durch einen Klick auf den Link *Alle Eigenschaften* kann die Darstellung der Schaltfläche angepasst werden.

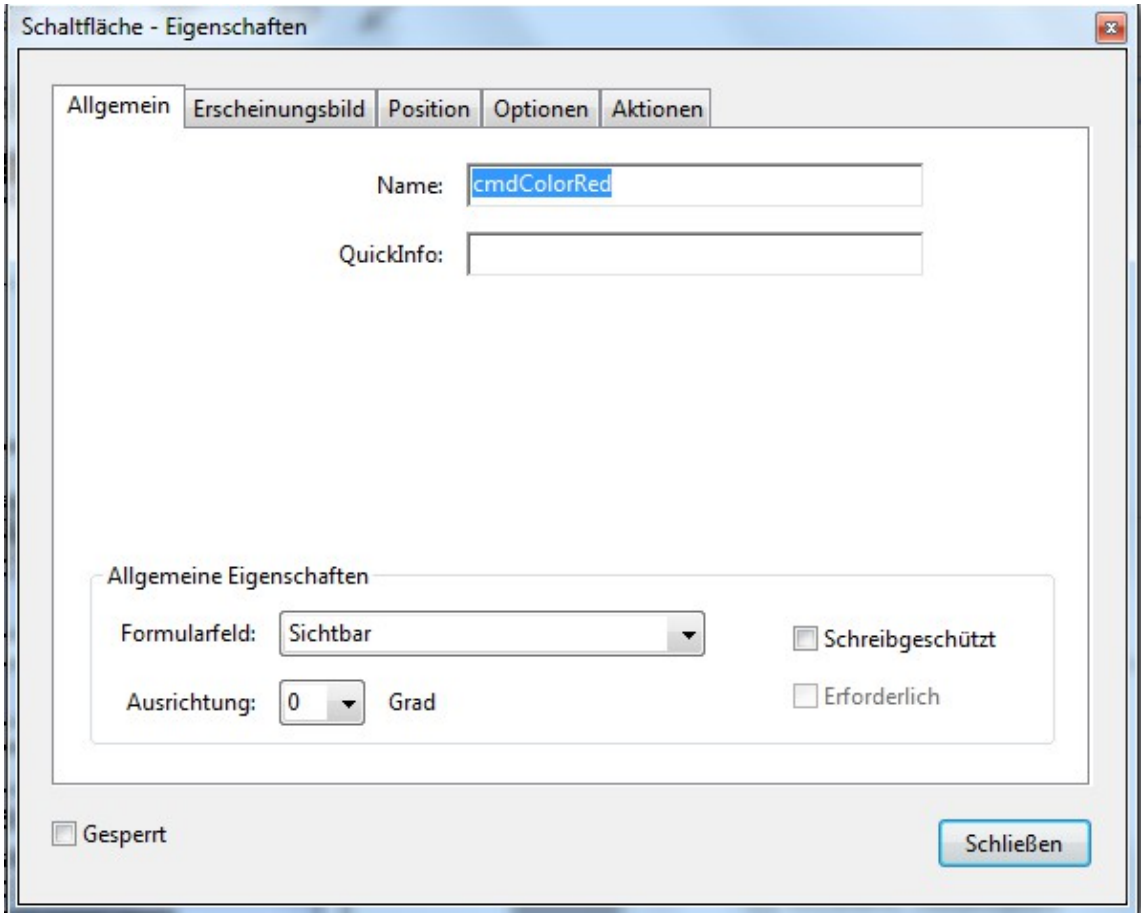
… in benutzerdefinierter Größe hinzufügen

- *Werkzeuge – Formular vorbereiten*.
- Klick auf das Icon [Schaltfläche] in der Symbolleiste [Formular vorbereiten].
- Mit Hilfe der Maus wird ein Auswahlrahmen an der gewünschten Position aufgezogen. Die Schaltfläche wird in Abhängigkeit der Größe des Rahmens eingefügt.
- Im Dialog wird der Name der Schaltfläche eingegeben. Durch einen Klick auf den Link *Alle Eigenschaften* kann die Darstellung der Schaltfläche angepasst werden.

Eigenschaften eines Formularfeldes

- Der Mauszeiger schwebt über einem Formularfeld.
- Durch einen Klick mit der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet.
- Im Kontextmenü wird Eintrag *Eigenschaften* ausgewählt.

Dialog „Eigenschaften“



Änderungen der Eigenschaften speichern

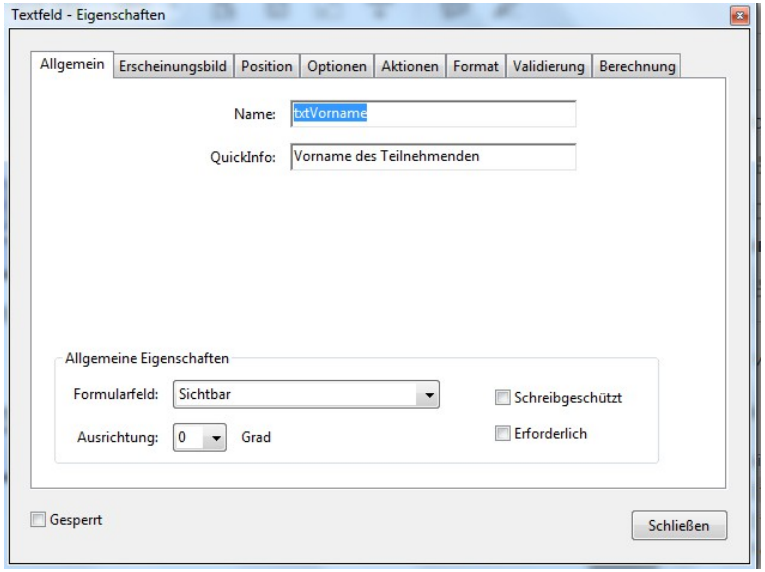
- Automatische Speicherung bei Auswahl einer neuen Eigenschaft.
- Automatische Speicherung bei Auswahl eines neuen Formularfeldes.
- Die Änderung wird mit Hilfe der <Eingabe>-Taste geschlossen.

Dialog schließen

- Klick auf die Schaltfläche *Schließen* am unteren Rand.
- Klick auf das Icon in der Titelleiste am rechten Rand.

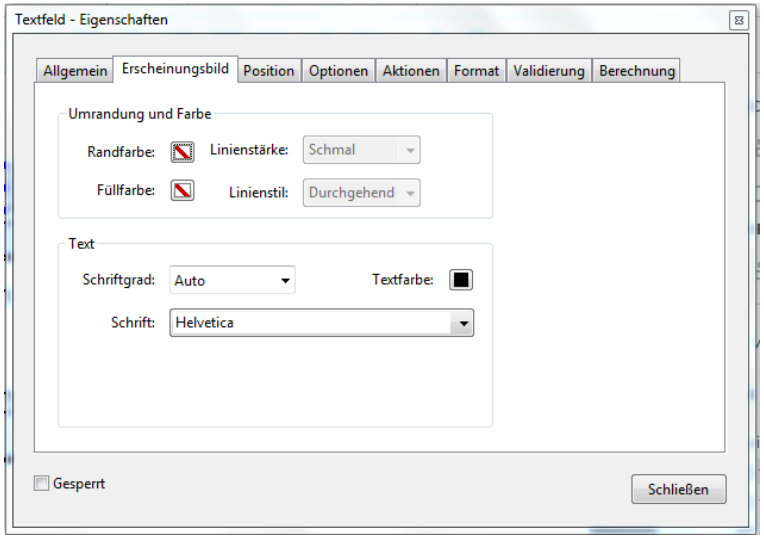
Registerkarte „Allgemein“

- In das Textfeld [Name] wird eine Bezeichnung für das Formularfeld eingegeben.
- In das Textfeld [Quickinfo] wird eine kurze Beschreibung der Aufgabe der Schaltfläche eingegeben.



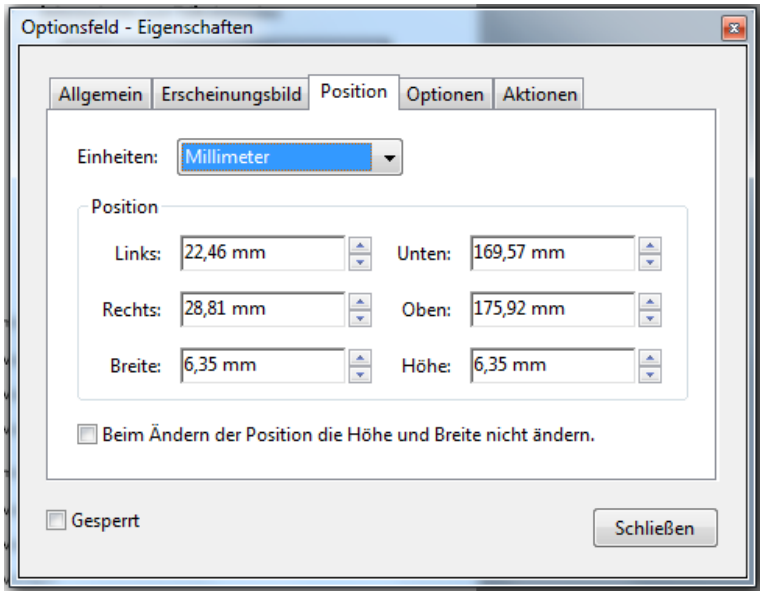
Registerkarte „Erscheinungsbild“

- Die [Randfarbe] legt die Farbe des Rahmens um die Schaltfläche fest.
- Die [Füllfarbe] färbt die Innenfläche der Schaltfläche ein. Die Füllfarbe deckt alle Elemente hinter der Schaltfläche ab.
- Mit Hilfe von [Text] kann die Schrift auf der Schaltfläche festgelegt werden.

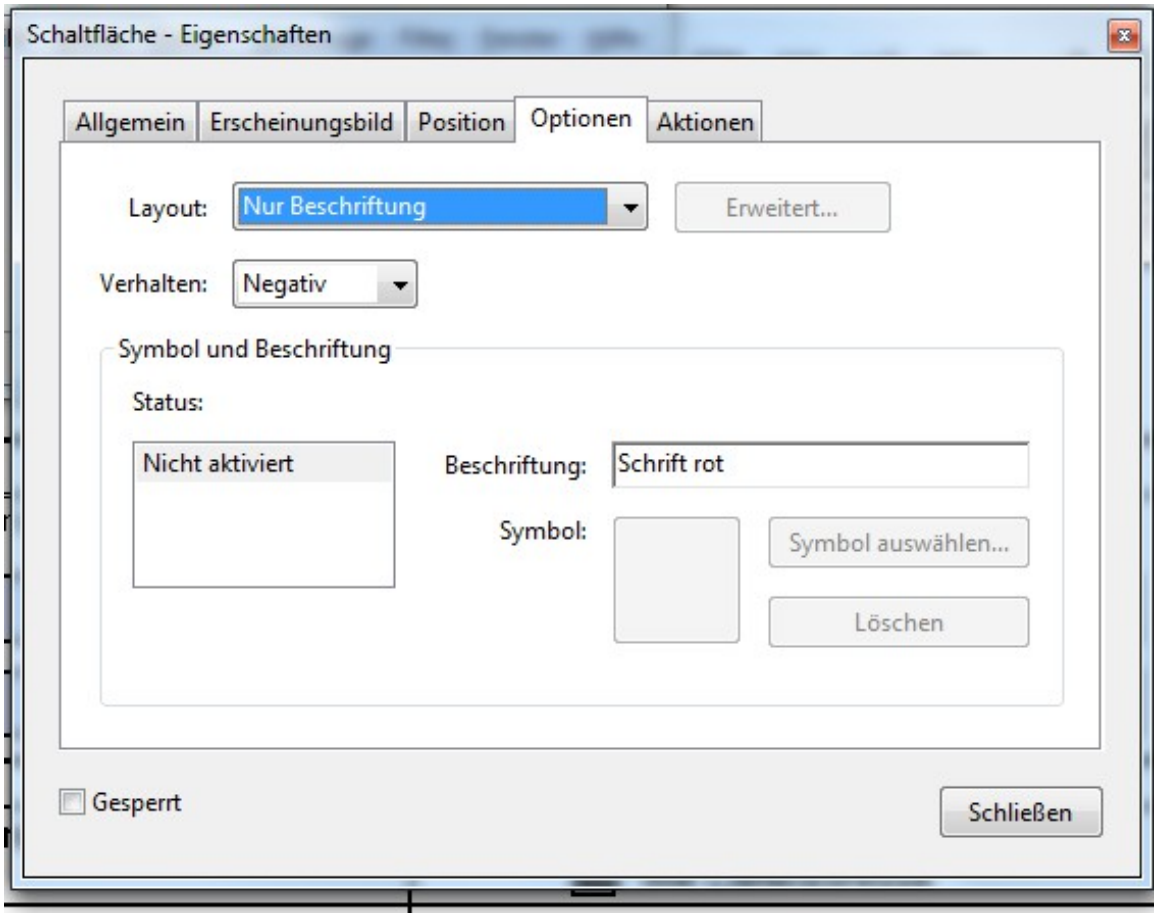


Registerkarte „Position“

- Alle Schaltflächen in einer Zeile haben die gleiche Höhe.
- Schaltflächen in einer Zeile sind oben an der gleichen horizontalen Linie ausgerichtet.
- Die Breite wird in Abhängigkeit der Beschriftung / Symbol gewählt.



Registerkarte Optionen



Layout einer Schaltfläche

- Mit Hilfe des DropDown-Liste [Layout] wird ausgewählt, ob eine Beschriftung und / oder ein Symbol auf der Schaltfläche angezeigt wird.
- Standardmäßig wird nur eine Beschriftung, zentriert, auf der Schaltfläche angezeigt.

Beschriftung einer Schaltfläche

- Die Beschriftung wird in das Textfeld [Beschriftung] eingetragen.
- Der eingegebene Text wird einzeilig, zentriert auf der Schaltfläche angezeigt.

Hinweise

- Die Text beschreibt exakt die Aktion, die die Schaltfläche ausführt.
- Die Beschriftung beschreibt kurz und knapp, welcher Vorgang ausgelöst wird,
- In Abhängigkeit der Kultur und der Sprache des Nutzerkreises wird eine Beschriftung gewählt.
- Häufig werden Verben für die Beschriftung genutzt.
- Jedes Wort in der Beschriftung sollte mit einem Großbuchstaben beginnen.

Symbol für eine Schaltfläche

- Mit Hilfe der Schaltfläche *Symbol auswählen* wird ein Icon ausgewählt.
- Mit Hilfe der Schaltfläche *Erweitert* kann das Symbol entsprechend der Größe der Schaltfläche skaliert werden.

Hinweise

- Ein Symbol muss sich von selbst erklären.
- Das Symbol sollte möglichst einfach gestaltet werden.
- Das Symbol sollte auf der Schaltfläche klar erkennbar sein.
- Das Icon sollte unabhängig von der Sprache und Kultur des Nutzerkreises gewählt werden.
- Alle Bildformate, die Adobe Acrobat anzeigen kann, können genutzt werden.

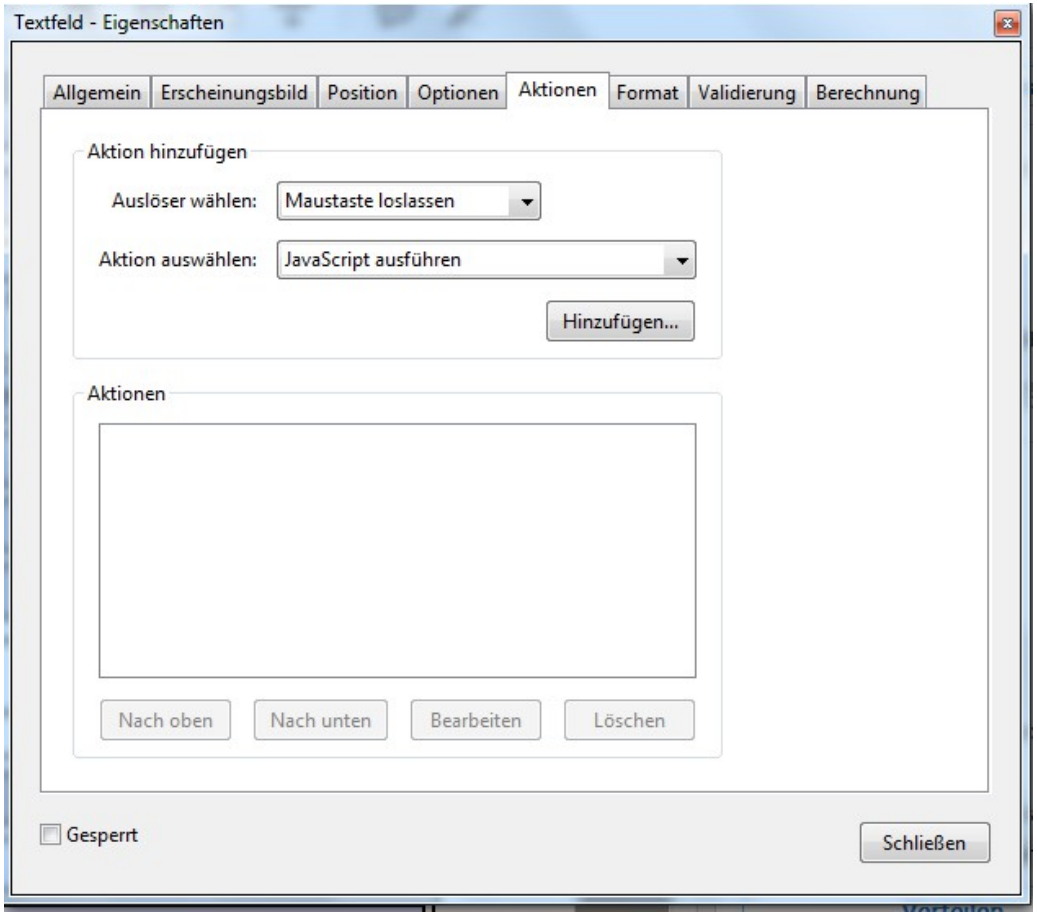
Verhalten der Schaltfläche

- Sobald die Schaltfläche gedrückt wurde, verändert sich das Layout.
- Kennzeichnung einer gedrückten Schaltfläche.

Möglichkeiten

- [Ohne]. Das Erscheinungsbild der Schaltfläche verändert sich nicht.
- [Umrandung]. Der Rahmen der Schaltfläche wird hervorgehoben.
- [Negativ]. Es werden die dunklen und hellen Farbtöne der Schaltfläche getauscht.
- [Schaltfläche]. Für die Status [Aktiviert], [Nicht aktiviert] und [Cursor darüber] kann eine Beschriftung und ein Symbol festgelegt werden.

Registerkarte „Aktionen“



- Wodurch wird die Aktion ausgelöst?
- Welche Aktion wird ausgelöst?

Auslöser

- Interaktion des Nutzers mit dem Formular.
- Reaktion auf Aktionen mit der Maus oder Tastatur.
- Einem Formularfeld können beliebig viele Auslöser zugeordnet werden.
- Ein Auslöser startet mindestens eine Aktion.

Möglichkeiten

- [Feld aktiviert]. Mausklick in ein Formularfeld. Mit Hilfe der <TAB>-Taste wird ein Feld aktiviert. In einem Textfeld ist die Einfügemarke sichtbar.
- [Feld deaktiviert]. Mausklick außerhalb des momentan aktiven Formularfelds. Das Feld wird mit der <TAB>-Taste verlassen.
- [Maus in Feld]. Der Mauszeiger ist über dem Formularfeld positioniert.
- [Maus aus Feld]. Der Mauszeiger wird vom Feld weggezogen.
- [Maustaste drücken]. Mausklick in das Formularfeld.
- [Maustaste loslassen]. Die Maustaste wird nach dem Klick in das Feld losgelassen.

Aktionen

- Zusammenfassung von Anweisungen, um ein bestimmtes Ziel zu erreichen.
- Einige Aktionen sind von Adobe Acrobat vordefiniert.
- Ausführung von JavaScript-Code.
- Einem Auslöser können beliebig viele Aktionen zugeordnet werden.

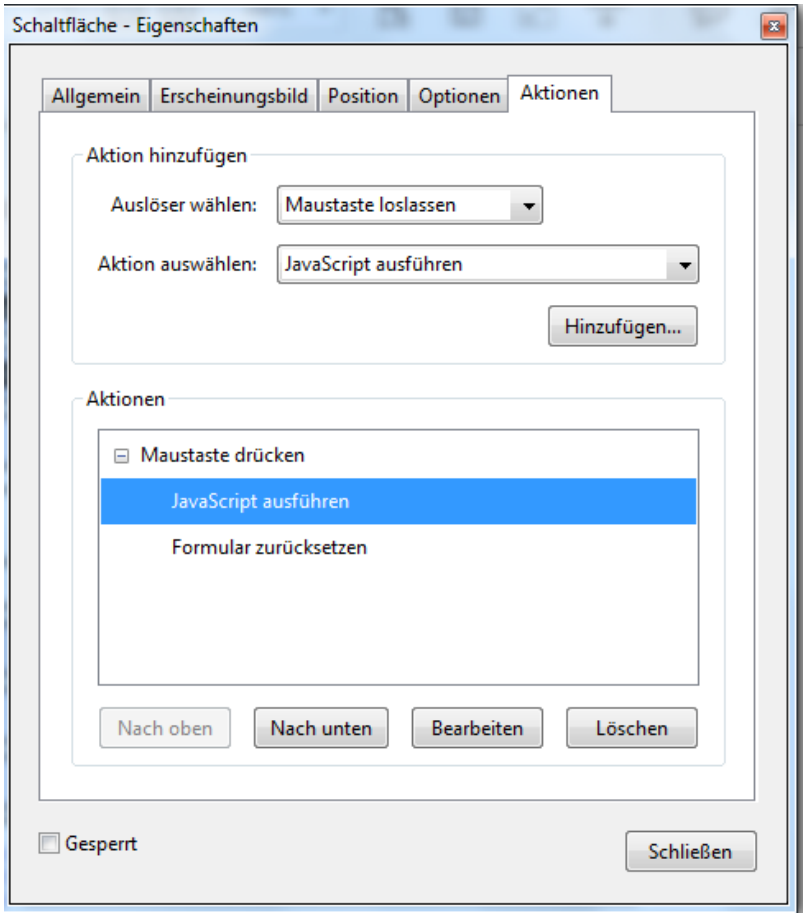
… einem Auslöser zuordnen

- Das Dialog [Eigenschaften] ist geöffnet.
- Die Registerkarte [Aktionen] ist aktiv.
- Mit einem Mausklick wird aus der Liste [Auslöser] das gewünschte Element gewählt. Zum Beispiel bei einer Schaltfläche wird der Auslöser [Maustaste drücken] ausgewählt.
- In der Liste [Aktion auswählen] wird die auszuführende Aktion ausgewählt.
- Durch einen Klick auf die Schaltfläche *Hinzufügen* wird ein Dialog in Abhängigkeit der Aktion für weitere Einstellungen geöffnet.
- Nach dem Schließen des Dialogs wird der Auslöser und die dazugehörigen Aktionen im Bereich [Aktionen] angezeigt.

Beispiel: Zurücksetzen des Formulars

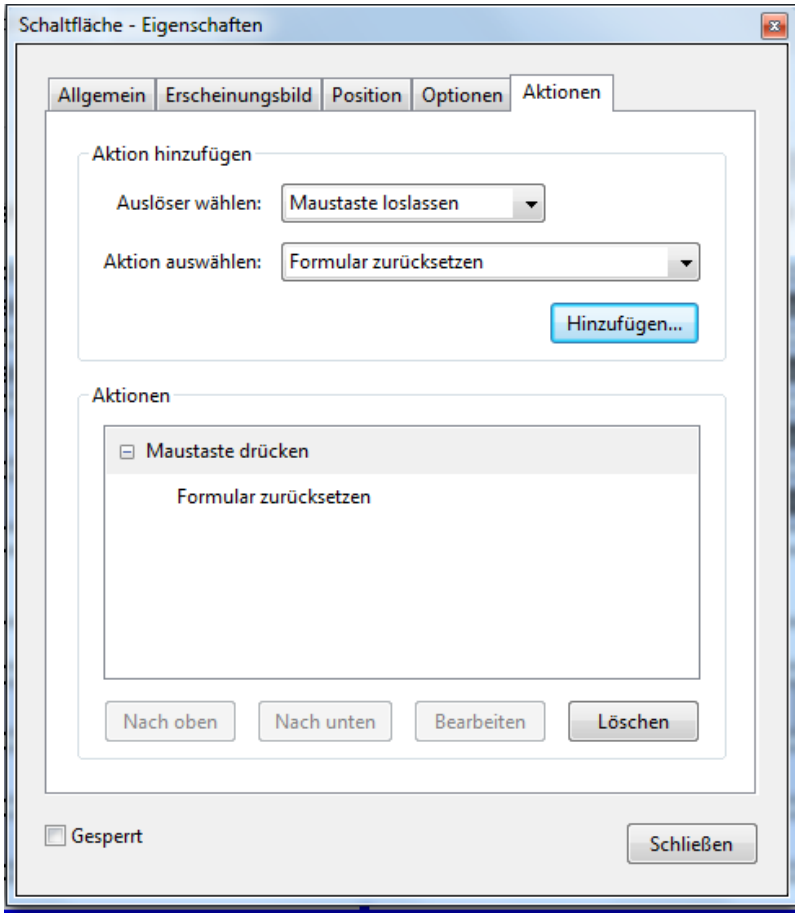
- Der Auslöser [Maustaste drücken] wird gewählt.
- In der Liste [Aktion auswählen] wird das Element [Formular zurücksetzen] gewählt.
- Mit Hilfe der Schaltfläche *Hinzufügen* wird der Dialog [Formular zurücksetzen] geöffnet.
- In dem Dialog werden mit Hilfe von Kontrollkästchen die Felder ausgewählt, die auf die Standardwerte zurückgesetzt werden sollen. Die Felder werden wie beim ersten Öffnen des Formulars angezeigt.
- Mit einem Klick auf die Schaltfläche *OK* wird der Dialog geschlossen.

Aktion bearbeiten



- Mit Hilfe der Maus wird eine Aktion ausgewählt. Die gewählte Aktion wird farbig hinterlegt.
- Mit Hilfe der Schaltfläche *Bearbeiten* wird der passende Dialog geöffnet.
- Im Dialog können die Einstellungen zu der Aktion verändert werden.

Aktion löschen



- Mit Hilfe der Maus wird eine Aktion ausgewählt. Die gewählte Aktion wird farbig hinterlegt.
- Mit Hilfe der Schaltfläche *Löschen* wird die gewählte Aktion entfernt.
- Hinweis: Auslöser benötigen mindestens eine Aktion. Andernfalls werden die Auslöser entfernt.

Mehr als eine Aktion ...

- Das Dialog [Eigenschaften] ist geöffnet.
- Die Registerkarte [Aktionen] ist aktiv.
- Mit einem Mausklick wird aus der Liste [Auslöser] ein Auslöser ausgewählt, der in dem Bereich [Aktionen] vorhanden ist.
- In der Liste [Aktion auswählen] wird die auszuführende Aktion ausgewählt. Die Aktion ist nicht in dem Bereich [Aktionen] vorhanden. Andernfalls wird die gleiche Aktion zwei Mal ausgeführt.
- Durch einen Klick auf die Schaltfläche *Hinzufügen* wird ein Dialog in Abhängigkeit der Aktion für weitere Einstellungen geöffnet.
- Nach dem Schließen des Dialogs wird der Auslöser der vorhandenen Aktionen im Bereich [Aktionen] zugeordnet.

Beispiel: Information an den Nutzer

- Der Auslöser [Maustaste drücken] wird gewählt.
- In der Liste [Aktion auswählen] wird das Element [JavaScript ausführen] gewählt.
- In dem JavaScript-Editor wird der Code eingegeben.
- Mit einem Klick auf die Schaltfläche *OK* wird der Editor geschlossen.
- In dem Bereich [Aktionen] wird dem vorhandenen Auslöser [Maustaste drücken] die gewählte Aktion zugeordnet.
- Mit Hilfe der Schaltflächen *Nach oben* oder *Nach unten* wird die Reihenfolge der Ausführung angepasst.

JavaScript-Code

```
app.alert("Alle Daten in den Formularfelder werden entfernt.", 1, 0);
```

- Der Nutzer wird über die Löschung der vorhandenen Daten informiert.

JavaScript

- Von Netscape entwickelte Programmiersprache.
- Beschreibung von Aktionen mit Hilfe von Wörtern, die der Computer versteht.
- Zusammensetzung von Wörter mit Hilfe von Regeln (einer Syntax).
- Unterstützung der Standard-Objekte und spezifischen Erweiterungen für Adobe Acrobat.
- Adobe JavaScript basiert auf der Version 1.5 des ISO-16262-Standard.

Informationen zu JavaScript

- http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/js_developer_guide.pdf
- http://www.adobe.com/content/dam/Adobe/en/devnet/acrobat/pdfs/js_api_reference.pdf
- https://helpx.adobe.com/de/acrobat/using/applying-actions-scripts-pdfs.html
- https://acrobatusers.com/tutorials/javascript_console
- http://www.adobe.com/devnet/acrobat/javascript.html

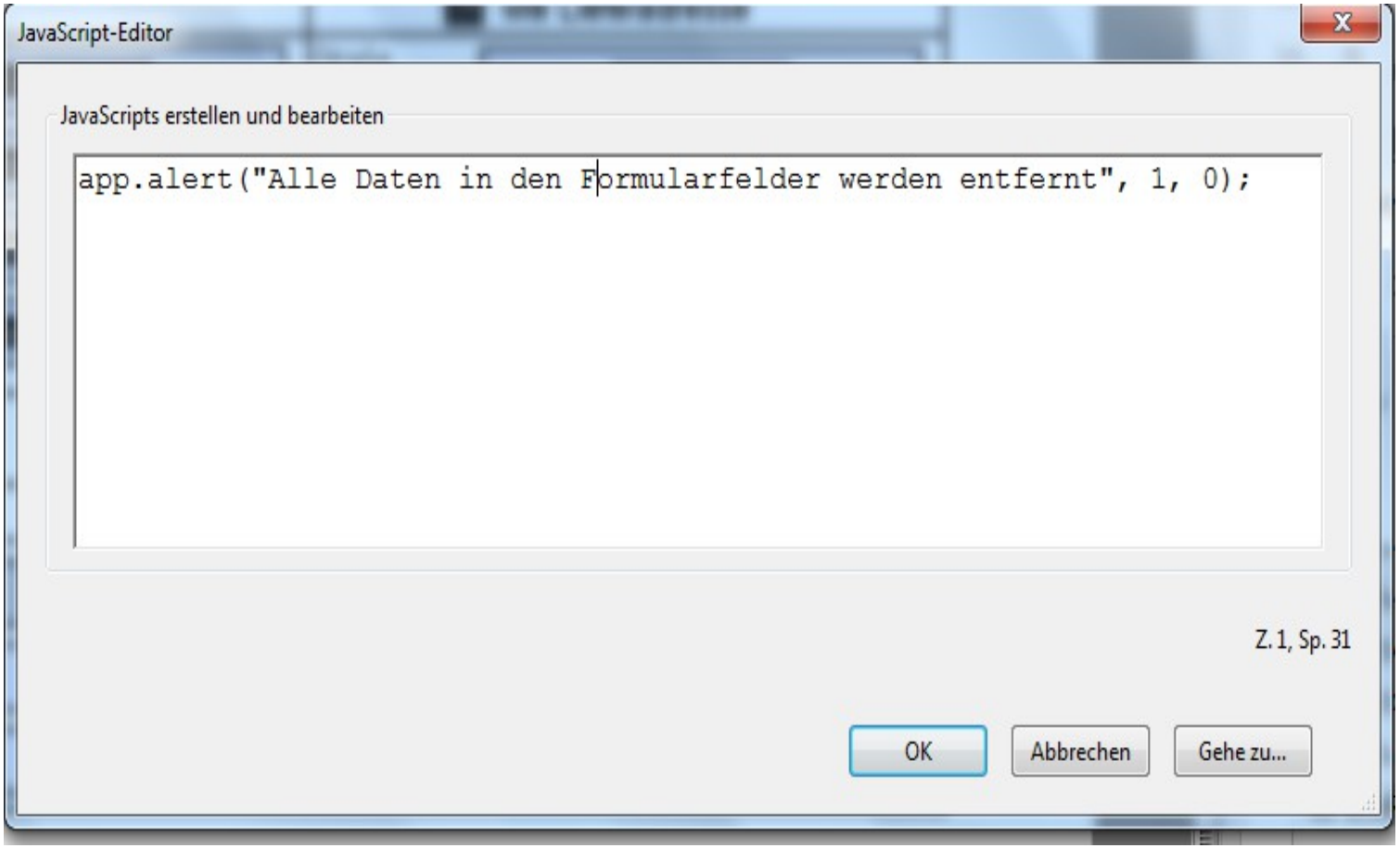
… in Adobe Acrobat

- Nutzung bei Formularfeldern. Reaktion auf Auslöser wie „Maustaste drücken“. Validierung und Berechnung von Daten.
- Aktionen beim Öffnen oder Schließen einer Seite.
- Dokument-Aktionen. Der Nutzer speichert, druckt oder schließt das Dokument.
- Globale Funktionen für das Dokument.

Eingabe von JavaScript-Code bei Formularfeldern

- Das Dialog [Eigenschaften] einer Schaltfläche ist geöffnet.
- Die Registerkarte [Aktionen] ist aktiv.
- Der Auslöser [Maustaste drücken] wird gewählt.
- In der Liste [Aktion auswählen] wird das Element [JavaScript ausführen] gewählt.
- In dem JavaScript-Editor wird der Code app.alert("Alle Daten in den Formularfelder werden entfernt.", 1, 0); eingegeben.
- Mit einem Klick auf die Schaltfläche *OK* wird der Editor geschlossen.

Eingebetteter JavaScript-Editor



Anweisungen

- Beschreibung einer Aktion in einer, für den Computer, verständlichen Sprache.
- Zusammensetzung von Schlüsselwörtern der Programmiersprache JavaScript, Operatoren und Operanden in Abhängigkeit einer vorgegebenen Syntax.

... in JavaScript

```
app.alert("Alle Daten in den Formularfelder werden entfernt.", 1, 0);
```

- Pro Zeile wird eine Anweisung geschrieben.
- Die Anweisung endet mit einem Semikolon.
- Sequenzielle Verarbeitung. Der Code wird von oben nach unten abgearbeitet.

Objekte

- Beschreibung von Dingen aus der realen Welt in einer Skriptsprache.
- Ein Objekt basiert auf einen Bauplan.
- Der Bauplan beschreibt mit Hilfe von Attributen das Objekt allgemeingültig.
- In dem Bauplan sind Methoden definiert, die das konkrete Objekt anpassen können.

Beispiele

- app. Die Applikation Adobe Acrobat.
- Doc. Ein PDF-Dokument.
- event. JavaScript-Ereignisse wie zum Beispiel „Maus gedrückt“.
- util. Formatierung von Zahlen und Zeitwerten.

Objekt „Application“

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
```

- Das Objekt app beschreibt die Applikation.
- Das Objekt basiert auf einen Bauplan „Adobe Acrobat“.

Objekt „Aktuelles Dokument“

```
this.getField("txt_Gesamtpreis");
```

- Das Objekt this ist ein Platzhalter für den Bauplan „Aktives Dokument“.
- Referenz auf das aktuell, in Adobe Reader oder Adobe Acrobat angezeigte Dokument.
- Verweis auf das aktive Dokument.

Methoden eines Objekts

- Vordefinierter Code passend zu einem Objekt.
- Methoden lesen oder verändern die Eigenschaften eines Objektes.
- Methoden beschreiben das dynamische Verhalten eines Objekts.
- Objekte und Methoden werden mit Hilfe des Punktoperators verbunden.

Beispiel

this	.	getField	(	"txt_Gesamtpreis"	)	;
Objekt	.	Methode				;

- Der Platzhalter this ist vom Typ Doc. Das Objekt repräsentiert das aktuell angezeigte PDF-Dokument.
- Die, in dem Objekt definierte Methode .getField() liefert ein Verweis auf ein Formularfeld in dem aktiven Dokument zurück.

Name einer Methode

this	.	getField	(	"txt_Gesamtpreis"	)	;
Objekt	.	Methode				;
Objekt	.	Name	(	Parameter	)	;

- Mit Hilfe des Namens wird eine Methode aufgerufen.
- Der Name muss exakt so wie im Bauplan beschrieben, eingegeben werden.
- Die Groß- und Kleinschreibung wird beachtet. Die Methode .GetField() ruft eine andere Methode als .getField() auf.

Parameter einer Methode

this	.	getField	(	"txt_Gesamtpreis"	)	;
Objekt	.	Methode				;
Objekt	.	Name	(	Parameter	)	;

- Dem Namen folgen direkt runde Klammern.
- Falls die runden Klammern leer sind, werden der Methode keine Startwerte übergeben.
- Wie viele Parameter der Methode übergeben werden, ist in der Deklaration der Methode definiert.
- In diesem Beispiel wird der Methode .getField() ein Formularfeld-Name übergeben.

Feldnamen als Parameter

```
this.getField("txtZeile02_Einzelpreis");
```

- Der Parameter, der der Methode übergeben wird, entspricht dem Textfeld [Name] in der Registerkarte [Allgemein] im Dialog [Eigenschaften].
- Der Name wird als String (Zeichenkette) übergeben. Strings beginnen und enden mit dem Anführungszeichen.

Rückgabewert einer Methode

var	feld	=	this	.	getField	(	"txt_Gesamtpreis"	)	;
Rückgabewert		=	Objekt	.	Methode				;

- Eine Methode kann einen Wert an den Aufrufer zurückgeben.
- Die Methode .getField() gibt ein Verweis auf ein Formularfeld zurück. Das Formularfeld wird mit Hilfe des übergebenen Parameters festgelegt.
- Falls das Formularfeld in dem aktiven Dokument nicht vorhanden ist, wird null zurück gegeben. Aktionen auf dem Formularfeld können nicht ausgeführt werden. Das Skript bricht ab.

Objekt und Methode verbinden

this	.	getField	(	"txt_Gesamtpreis"	)	;
Objekt	.	Methode				;

- Die Methode .getField() wird mit dem Objekt this durch den Punkt verbunden.
- In dem Objekt (links vom Punkt) ist die Methode (rechts vom Punkt) deklariert.

Methode „Formularfeld zurücksetzen“

```
this.resetForm("optGrpAnrede");
this.resetForm(["txtIBAN", "txtBIC", "txtBankname"]);
this.resetForm();
```

- Die Methode .resetForm() wird mit dem Objekt this durch den Punkt verbunden.
- Der Platzhalter this ist vom Typ Doc. Das Objekt repräsentiert das aktuell angezeigte PDF-Dokument. In dem Objekttyp ist eine Methode mit dem Namen .resetForm() definiert.
- In den runden Klammern werden die Felder angegeben, die zurück gesetzt werden.

Parameter der Methode

```
this.resetForm();
this.resetForm("optGrpAnrede");
this.resetForm(["txtIBAN", "txtBIC", "txtBankname"]);
```

- Falls die runden Klammern leer sind, werden alle Felder des dazugehörigen Formulars zurückgesetzt.
- Der zurück zu setzende Feldname wird als String angegeben. Strings beginnen und enden mit dem Anführungszeichen.
- Mehrere Feldnamen werden als Array übergeben. In der Liste werden die Feldnamen durch Kommata getrennt. Jedes Element in der Liste wird als String definiert. Die Liste beginnt und endet mit den eckigen Klammern.

Array

```
this.resetForm( ["txtIBAN", "txtBIC", "txtBankname"] );
```

- Sammlung von Werten zu einem Thema.
- Mit Hilfe der eckigen Klammern wird ein Array erzeugt.
- Ein Array kann aus beliebig vielen Elementen bestehen. Jedes Element wird durch Kommata getrennt.
- In diesem Beispiel wird ein Array von Feldnamen übergeben. Jedes Element hat einen Wert vom Typ „String“. Die Feldnamen werden durch die Anführungszeichen begrenzt.

Methode alert() der Applikation

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
```

- Das Objekt app besitzt eine Methode .alert().
- Objekt und Methode werden durch den Punktoperator verbunden.
- Die Methode .alert() öffnet ein Dialog.
- Der Dialog kann den Nutzer des Formular informieren oder warnen.

Parameterliste der Methode

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
app.alert({cMsg:"Meldung", nIcon:3, nType:1, cTitle:"Titel"});
```

- Eine Liste von Startwerten wird der Methode übergeben.
- Die Parameterliste folgt direkt im Anschluss an den Namen der Methode
- Die Liste wird durch die runden Klammern begrenzt.
- Die Liste kann kein, ein oder beliebig viele Parameter enthalten.
- Die Parameter in der Liste werden durch ein Kommata getrennt.

Benannte Parameter

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
app.alert({cMsg:"Meldung", nIcon:3, nType:1, cTitle:"Titel"});
```

- In der Definition der Methode wird für jeden Parameter ein Platzhalter angegeben.
- Der Platzhalter besteht aus einem Schlüssel-Wert-Paar. Schlüssel und Wert werden durch einen Doppelpunkt verbunden.
- Der Schlüssel (der Name) ist eindeutig.
- Eine Liste von benannten Parametern wird durch die geschweiften Klammern begrenzt. Die einzelnen Parameter in der Liste werden durch Kommata getrennt.

Meldung und Titel

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
app.alert({cMsg:"Meldung", nIcon:3, nType:1, cTitle:"Titel"});
```

- Der Titel wird in der Titelleiste des Dialogs angezeigt. Der Parameter ist optional.
- Die Meldung wird im Dialog angezeigt. Die Meldung kann den Nutzer informieren oder bei einem Fehler warnen. Die Meldung muss angegeben werden.
- Meldung und Titel werden als Strings übergeben. Beide Variablen beginnen und enden mit den Anführungszeichen.

Schaltflächen im Dialog

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
app.alert({cMsg:"Meldung", nIcon:3, nType:1, cTitle:"Titel"});
```

- Der Parameter nType (dritte Variable) kann ein Wert von 0 bis 3 haben.
- Der Parameter ist optional. Standardmäßig wird die Schaltfläche OK angezeigt.
- Der Wert 0 zeigt nur die Schaltfläche OK im Dialog an. Der Wert 1 symbolisiert die Anzeige der Schaltfläche OK und Abbrechen. Der Wert 2 zeigt die Schaltflächen Ja und Nein an. Der Wert 3 blendet die Schaltflächen Ja, Nein und Abbrechen ein.

Symbol in dem Dialog

```
app.alert("Alle Daten in den Formularfelder werden entfernt", 1, 0);
app.alert({cMsg:"Meldung", nIcon:3, nType:1, cTitle:"Titel"});
```

- Der Parameter nIcon (zweite Variable) kann ein Wert von 0 bis 3 haben.
- Der Parameter ist optional und zeigt ein Icon zum besseren Verständnis der Meldung an.
- Der Wert 0 ist der Standardwert und zeigt das Icon Halt / Stopp an. Der Wert 1 symbolisiert eine Warnmeldung. Der Wert 2 wird für eine Frage, die der Nutzer mit Ja / Nein beantworten kann, genutzt. Der Wert 3 kennzeichnet eine Informationsmeldung.

Attribute (Member)

- Allgemein gültige Beschreibung eines Objekts.
- Variablen eines Objektes.
- Eigenschaften eines Objekts wie zum Beispiel die Schriftfarbe etc.
- Attribute und Objekte werden mit dem Punkt verbunden.
- Einige Eigenschaften können nur in Adobe Acrobat gesetzt werden.

Daten in einem Formularfeld

```
this.getField("txtRechnungStrasse").value =
                    this.getField("txtLieferungStrasse").value;

this.getField("txtRechnungPostleitzahl").value = "30159"
```

- Das Attribut .value gibt den aktuell angezeigten Wert in einem Formularfeld zurück.

Zuweisung

```
this.getField("txtRechnungStrasse").value =
                    this.getField("txtLieferungStrasse").value;

this.getField("txtRechnungPostleitzahl").value = "30159"
```

- Mit Hilfe des Gleichheitszeichen wird einem Platzhalter ein Wert zugewiesen.
- Rechts vom Gleichheitszeichen wird der Wert berechnet oder direkt angegeben.
- Der Wert wird in dem Platzhalter, links vom Gleichheitszeichen, gespeichert.
- Als Platzhalter kann der Name eines Attributs oder einer Variablen genutzt werden.

„Kopie“ von variablen Werten

```
this.getField("txtRechnungStrasse").value =
                    this.getField("txtLieferungStrasse").value;
```

- Das Attribut .getField("txtLieferungStrasse").value ist ein Platzhalter für einen beliebigen Wert in dem Feld „txtLieferungStrasse“.
- Der Wert in dem Attribut .getField("txtLieferungStrasse").value wird in dem Attribut .getField("txtRechnungStrasse").value gespeichert.
- Das Formularfeld „txtLieferungStrasse“ und „txtRechnungStrasse“ im aktuellen Dokument zeigen den gleichen Wert an.

Zuweisung eines konstanten Wertes

```
this.getField("txtRechnungPostleitzahl").value = "30159";
```

- Rechts vom Gleichheitszeichen wird direkt ein Wert vom Programmierer eingegeben.
- Der konstante Wert (Literal) wird entsprechend des Datentyps angegeben. In diesem Beispiel wird ein String genutzt.
- Werte, die direkt im Code angegeben werden, stehen immer rechts vom Gleichheitszeichen.
- Der konstante Wert wird in dem Feld „txtRechnungPostleitzahl“ angezeigt.

Möglichkeiten

3		Ganzzahl
0.45	.45	Dezimalzahl
"Text"		String (Zeichenkette)
true	false	Boolean (Wahr / Falsch)

Schriftfarbe eines Formularfeldes

```
this.getField("txt_Gesamtpreis").textColor = ["RGB", 1, 0, 0];
```

- Die Schriftfarbe (.textColor) in einem Formularfeld wird angepasst.
- Dem Attribut wird eine bestimmte Farbe zugewiesen. In diesem Beispiel wird als Schriftfarbe „Rot“ genutzt.

Standardfarben in JavaScript

Farbe	Farbarray	Color	Farbe
	["G", 0] ["CMYK", 0, 0, 0, 1]	color.black	Schwarz
	["G", 1]	color.white	Weiß
	["RGB", 1, 0, 0]	color.red	Rot
	["RGB", 0, 1, 0]	color.green	Grün
	["RGB", 0, 0, 1]	color.blue	Blau
	["CMYK", 0, 0, 1, 0]	color.yellow	Gelb
	["CMYK", 0, 1, 0, 0]	color.magenta	Magenta
	["CMYK", 1, 0, 0, 0]	color.cyan	Cyan

Nutzung von Standardfarben

```
this.getField("txt_Gesamtpreis").textColor = color.black;
```

- Das Objekt color bietet einige vordefinierte Farben als Attribute an.

Nutzung von Farbpaletten

```
this.getField("txt_Gesamtpreis").textColor = ["T"];
this.getField("txt_Gesamtpreis").textColor = ["G", 0];
this.getField("txt_Gesamtpreis").textColor = ["RGB", 1, 0, 0];
this.getField("txt_Gesamtpreis").textColor = ["CMYK", 0, 0.9, 1, 0];
```

- Farben können mit Hilfe eines Arrays angegeben. Das Array wird durch die eckigen Klammern begrenzt. Die Elemente in dem Array werden durch Kommata getrennt.
- Das erste Elemente in einem Array gibt die Farbpalette an. Die Farbpalette wird als String angegeben.
- Alle anderen Elemente des Arrays definieren die Farbe in Abhängigkeit der gewählten Farbpalette.

Farbe „transparent“

```
this.getField("txt_Gesamtpreis").textColor = ["T"];
```

- Das Array „Transparent“ besteht nur aus einem Element.
- Die Farbpalette wird durch den Buchstaben T gekennzeichnet.

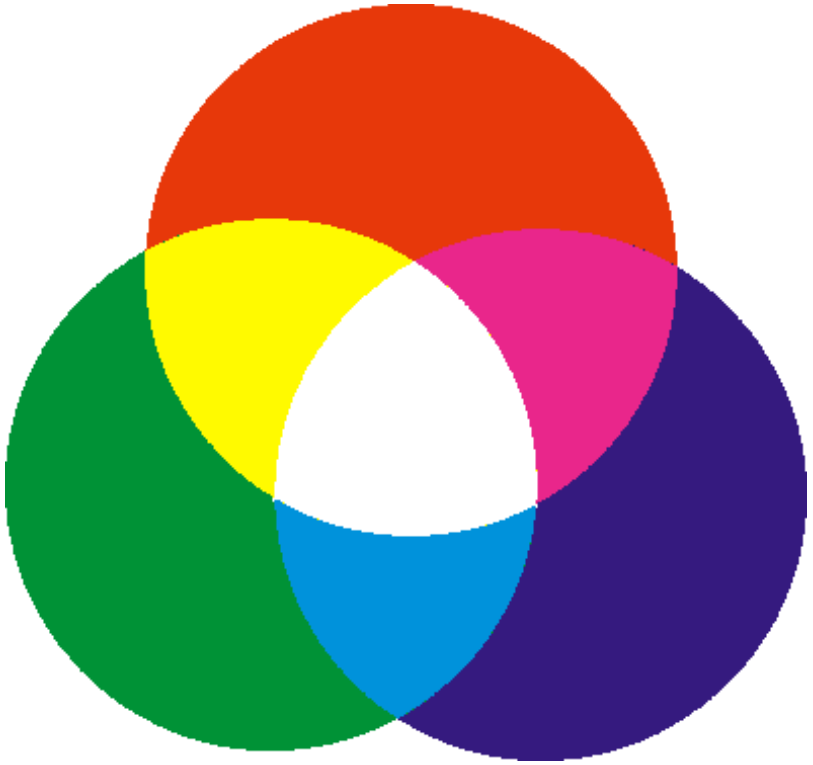
Grautöne

```
this.getField("txt_Gesamtpreis").textColor = ["G", 0];
```

- Das Array „Grautöne“ besteht aus zwei Elementen.
- Die Farbpalette wird durch den Buchstaben G gekennzeichnet.
- Das zweite Elemente gibt den Grauton mit Hilfe eines Wertes zwischen 0 (schwarz) und 1 (weiß) an.

RGB-Farben

- Darstellung von Farben am Bildschirm, Smartphone, etc.
- Das RGB-Farbsystem addiert (mischt) Licht in verschiedenen Farben.
- Jede Farbe (Rot, Grün, Blau) wird in 256 Helligkeitsstufen unterteilt.
- Um so weniger Farbe verwendet wird, um so heller wird diese.



Anteile einer RGB-Farben

- Jeder Farbton wird aus den Farben Rot, Grün und Blau gemischt.
- Jeder Farbanteil kann einen Wert zwischen 0 und 1 annehmen.
- Der Wert 0 entspricht „ausgeschaltet“. Wenn alle Farben nicht genutzt werden, wird der Farbton „Schwarz“ angezeigt.
- Der Wert 1 entspricht „ganz hell“. Wenn alle Farben genutzt werden, wird der Farbton „Weiß“ dargestellt.

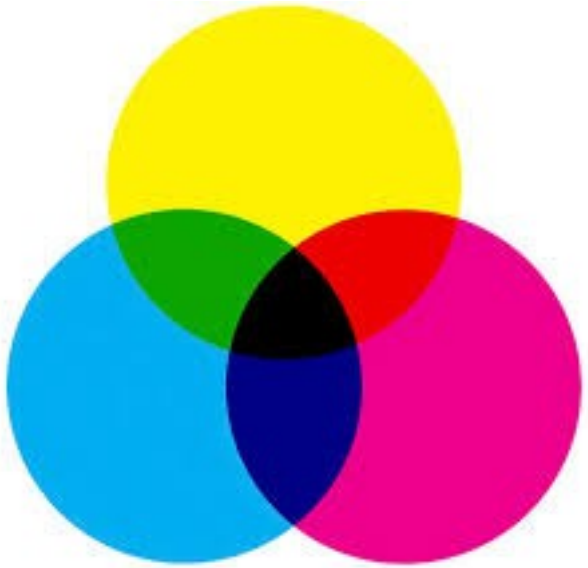
… in JavaScript

```
this.getField("txt_Gesamtpreis").textColor = ["RGB", 1, 0, 0];
```

- Das Array „RGB“ besteht aus vier Elementen.
- Das erste Element in dem Array legt das Farbmodell mit Hilfe der Buchstabenfolge RGB fest.
- Die folgenden drei Elemente geben den Rot-, den Grün- und den Blauanteil mit Hilfe von Werten zwischen 0 und 1 an.

CMYK-Farben

- Darstellung von Farben im Druck.
- Tuschkasten-Prinzip.
- Das CMYK-Farbsystem subtrahiert die Komponenten Cyan, Magenta, Yellow und den Schwarzanteil (Key).
- Je mehr Farbe zusammengemischt werden, um so dunkler werden diese.
- Schwarz wird zum Drucken von Graustufen und einem tiefen Schwarz benötigt.



Anteile einer CMYK-Farbe

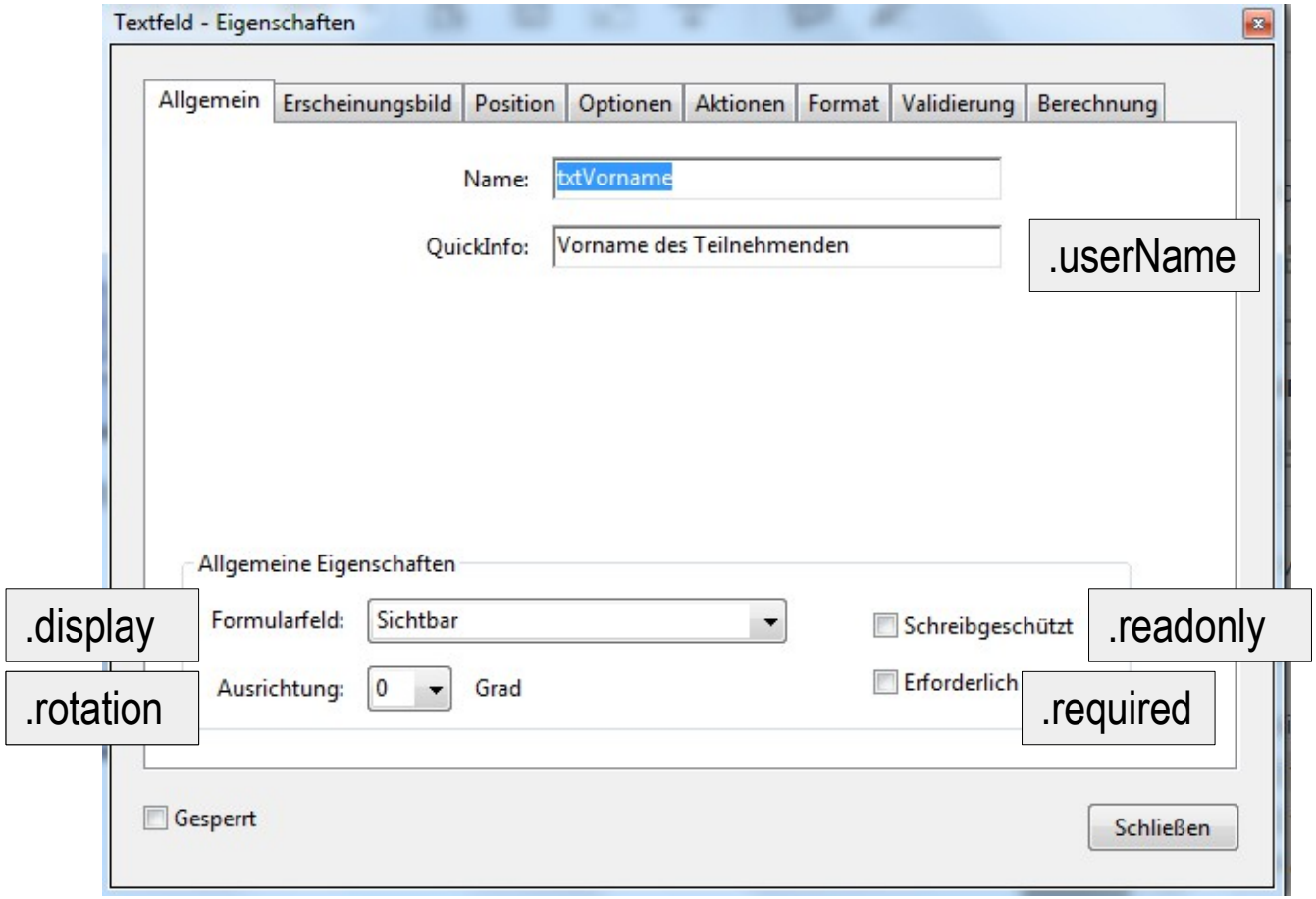
- Jeder Farbanteil (Cyan, Magenta, Yellow und Schwarzanteil) kann einen Wert von 0 % bis 100 % haben.
- Wenn alle vier Komponenten einen Anteil von 0 % haben, wird die Farbe „Weiß“ gedruckt.
- Wenn der Schwarzanteil einen Anteil von 100 % hat und alle anderen Komponenten einen Anteil von 0 %, wird die Farbe „Schwarz“ gedruckt.

… in JavaScript

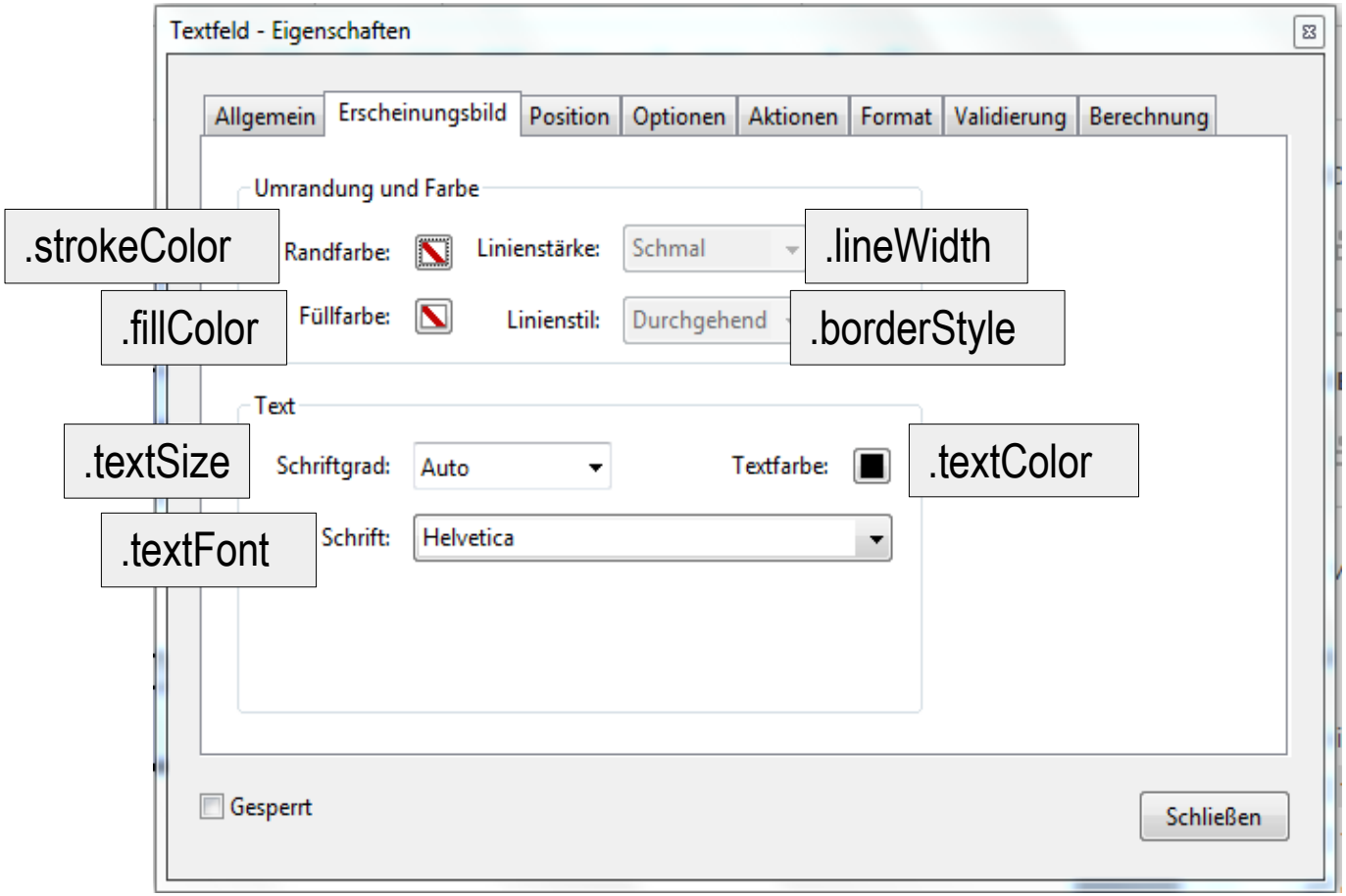
```
this.getField("txt_Gesamtpreis").textColor = ["CMYK", 0, 0.9, 1, 0];
```

- Das Array „CMYK“ besteht aus vier Elementen.
- Der erste Element in dem Array legt das Farbmodell mit Hilfe der Buchstabenfolge CMYK fest.
- Die folgenden drei Elemente geben den Cyan-, den Magenta- und den Gelb-Farbanteil an.
- Das letzte Element stellt den Schwarzanteil an der Farbe dar.
- Der Wert 1 für die verschiedenen Komponenten entspricht 100 %.

Weitere Eigenschaften

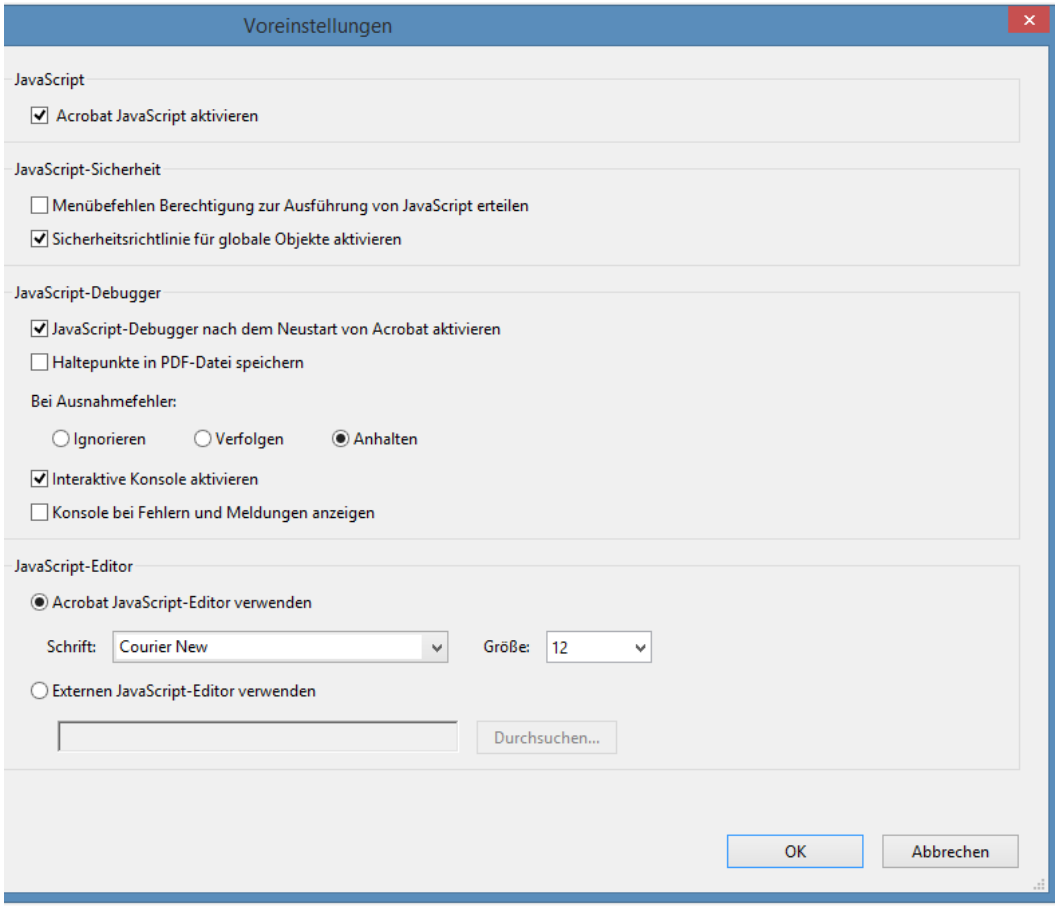


Weitere Eigenschaften



Einstellungen für JavaScript

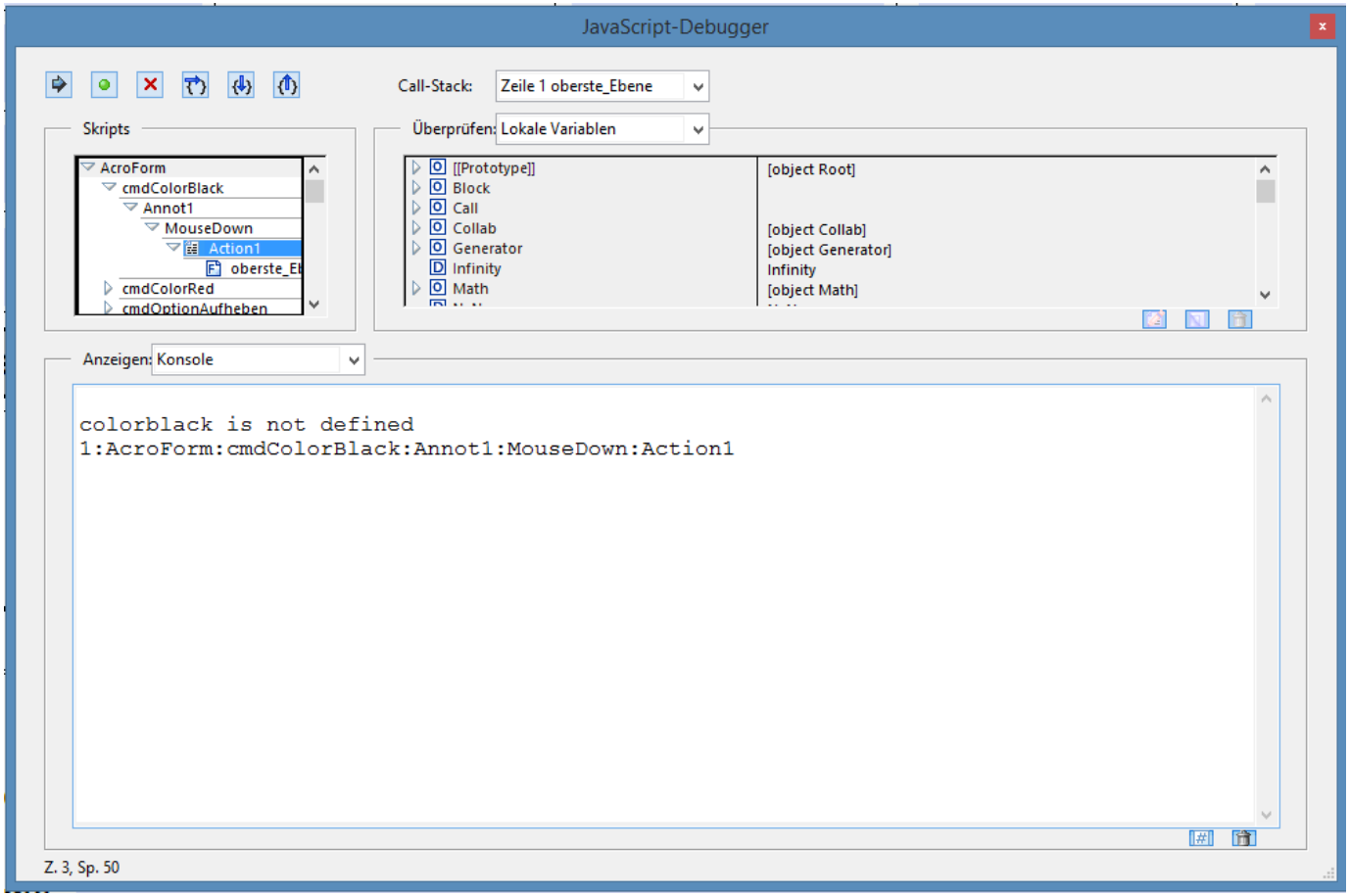
- *Bearbeiten – Voreinstellungen*.
- Kategorie *JavaScript*.



Erläuterung

- Das Kontrollkästchen [Acrobat JavaScript aktivieren] muss ein Häkchen enthalten. Andernfalls wird kein JavaScript ausgeführt.
- Mit Hilfe des JavaScript-Debuggers können Fehler im Script analysiert werden. Während der Erstellung sollte das Kontrollkästchen [JavaScript-Debugger nach dem Neustart von Acrobat aktivieren] aktiv sein. Um die Konsole im Debugger zu nutzen, muss das Kontrollkästchen [Interaktive Konsole aktivieren] ein Häkchen enthalten.
- Es kann ein externer JavaScript-Editor genutzt werden.

JavaScript-Debugger



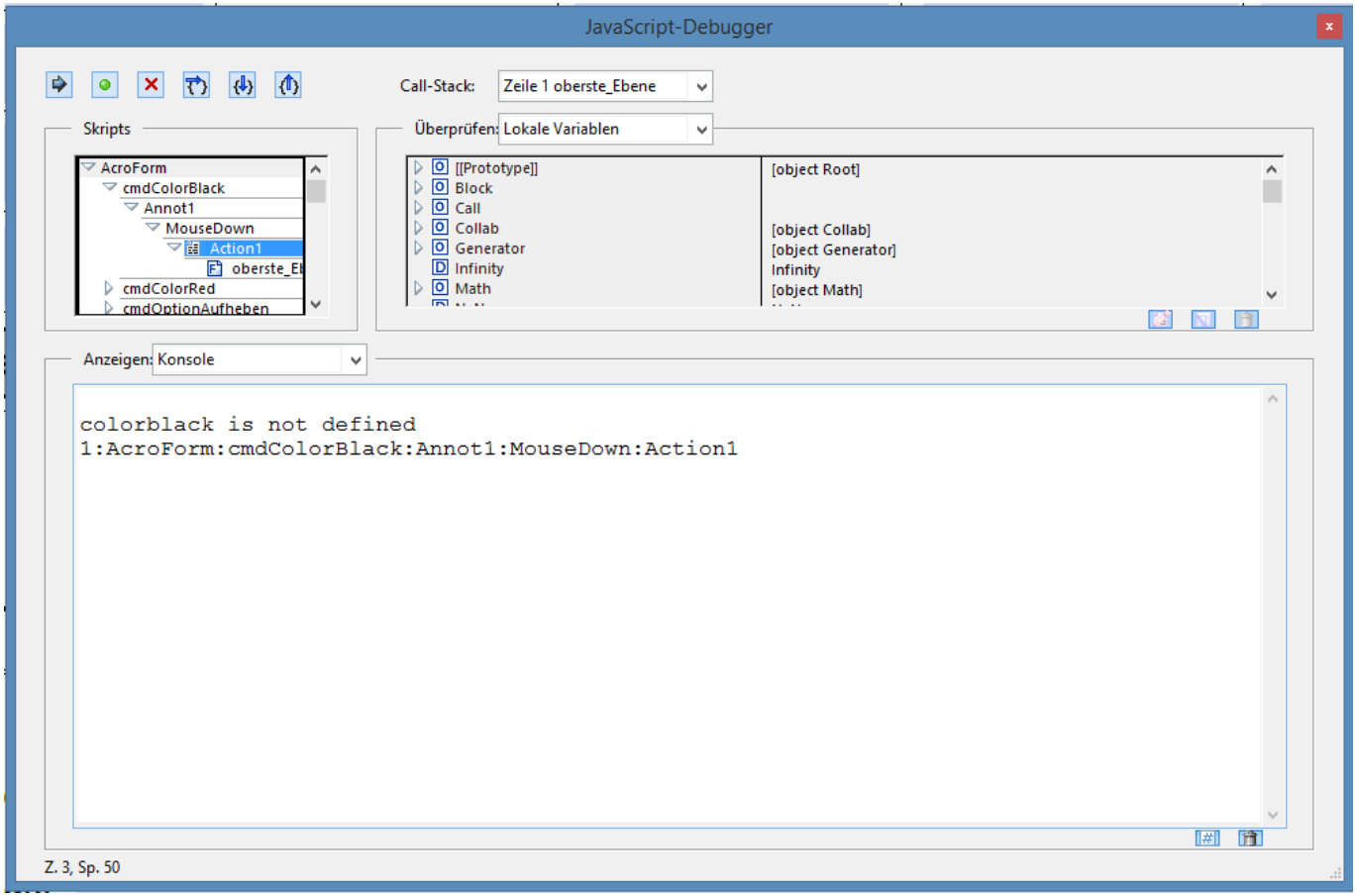
Einstellungen für den Debugger

Konsole

… starten

- *Werkzeuge – JavaScript*.
- Klick auf den Eintrag *Debugger* in der Werkzeugleiste.

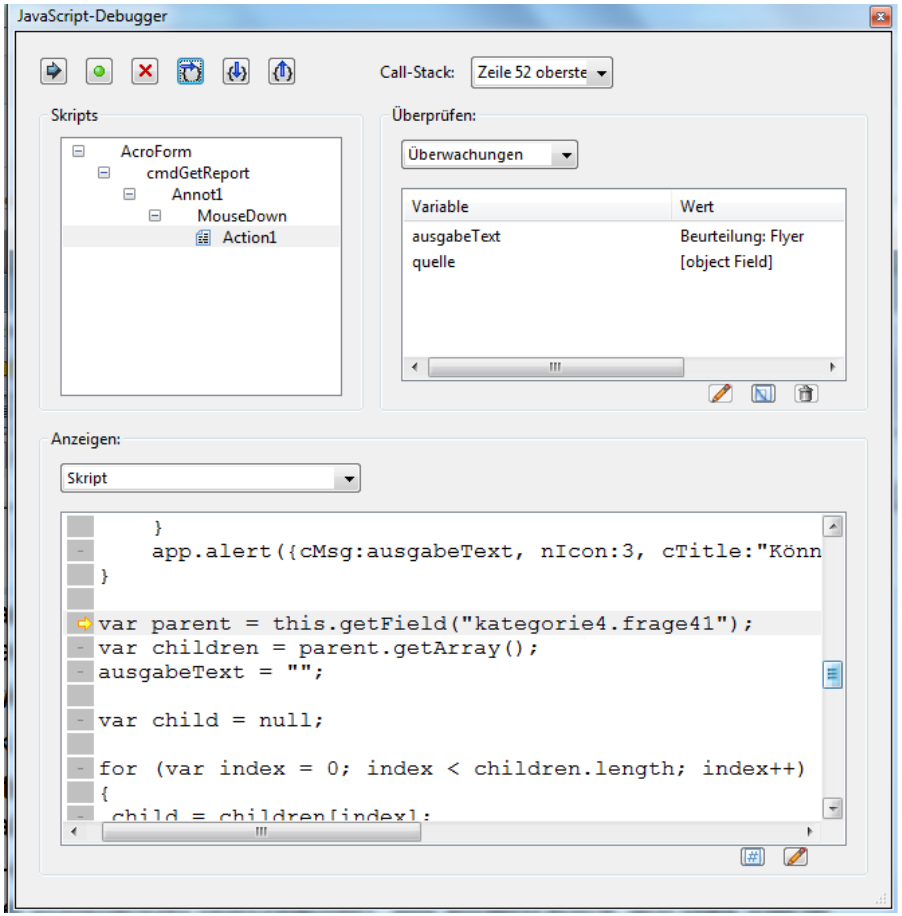
Anzeige von Fehlern



Code debuggen

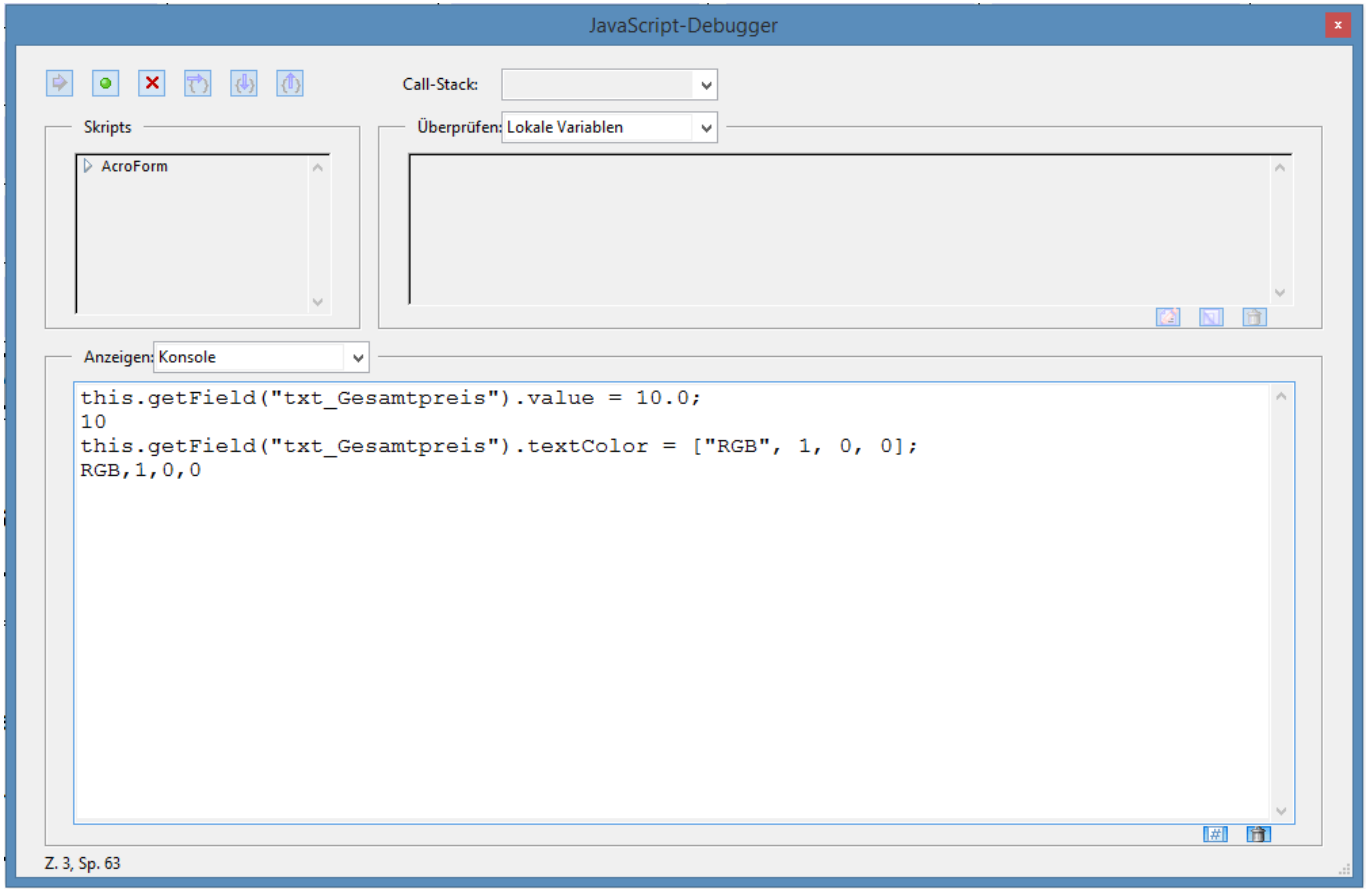
- Voraussetzung: Der Debugger ist aktiviert (*Bearbeiten* – *Voreinstellungen*. Kategorie [JavaScript]).
- *Werkzeuge* – *JavaScript*.
- Klick auf das Werkzeug *Debuggen*.
- Starten der zu überprüfenden Aktion. Die Aktion wird im JavaScript-Editor geöffnet.
- Mit Hilfe des ersten Icons am oberen rechten Rand wird der Code vollständig durchlaufen. Mit einem Klick auf das vierte Icon wird der Code zeilenweise durchlaufen.

Debugger



- Welches Skript wird durchlaufen?
- Welche Variablen werden überwacht?
- Anzeige des Skripts. Der gelbe Pfeil kennzeichnet die aktuell auszuführende Zeile.

Eingaben in die Konsole



Erläuterung

- Mit Hilfe der Tastatur können Anweisungen in die Konsole eingegeben werden.
- Der Code wird durch einen Klick auf das erste Icon am oberen linken Rand des Debuggers ausgeführt. Andere Möglichkeit: <Strg>+<Eingabe>.

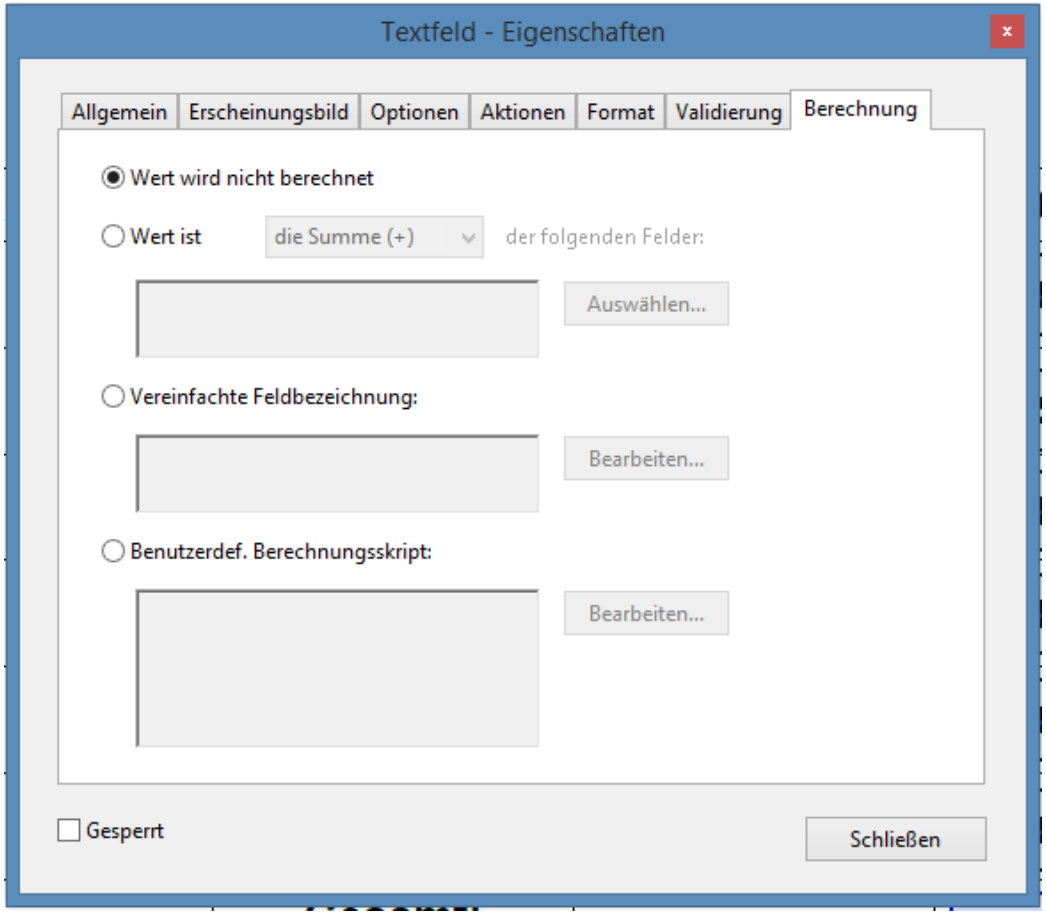
Berechnungen in einem PDF-Formular

- Automatisierte Berechnung von Daten aus vorhandenen Informationen und / oder festen Werten.
- Der eingebettete Code in der Registerkarte [Berechnung] wird automatisch beim Anzeigen des Formularfeldes gestartet.
- Felder, die in einer mathematischen Berechnung genutzt werden, müssen als Zahlen mit Hilfe der Registerkarte [Format] formatiert werden.
- Automatisiert berechnete Werte sollten schreibgeschützt werden.

… eingeben

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über dem Textfeld, deren Daten berechnet werden sollen.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet. Der Eintrag *Eigenschaften* wird gewählt.
- Die Registerkarte [Berechnung] wird geöffnet.

Registerkarte „Berechnung“



Option „Wert ist“

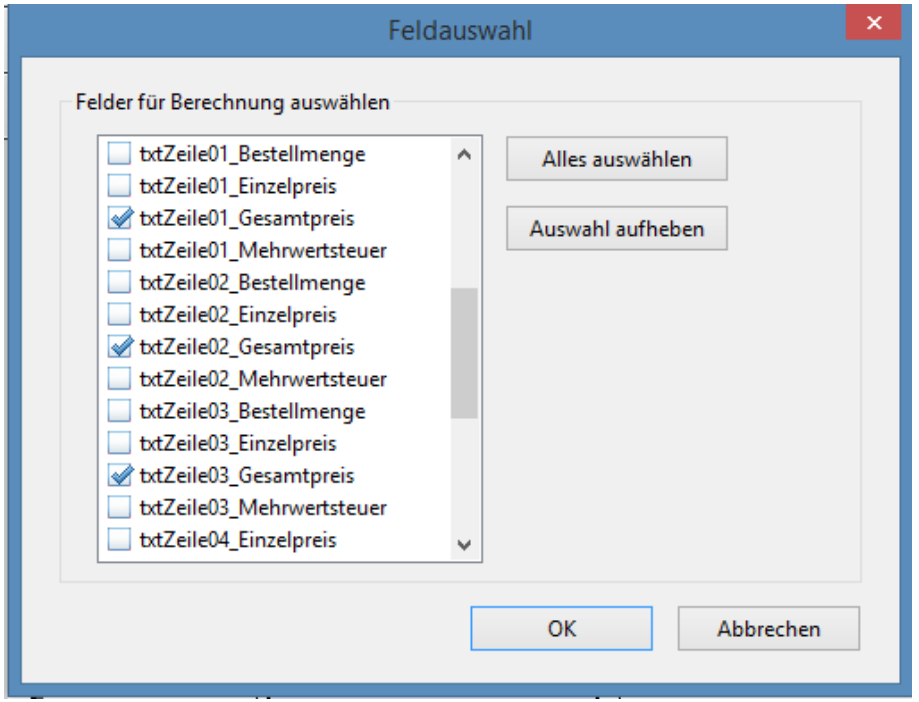
- Mit Hilfe der DropDown-Liste wird die Aggregatfunktion ausgewählt. Aggregatfunktionen fassen Werte mit Hilfe einer Berechnung zusammen.
- Im nächsten Schritt wird mit Hilfe der Schaltfläche *Auswählen* der Dialog [Feldauswahl] geöffnet.

Auswahl der Aggregatfunktion

Bezeichnung	Erläuterung
die Summe (+)	Summe aller Werte.
das Produkt (x)	Multiplikation der Werte.
der Durchschnitt	Durchschnitt aller Werte.
das Minimum	Kleinster Wert.
das Maximum	Größter Wert.

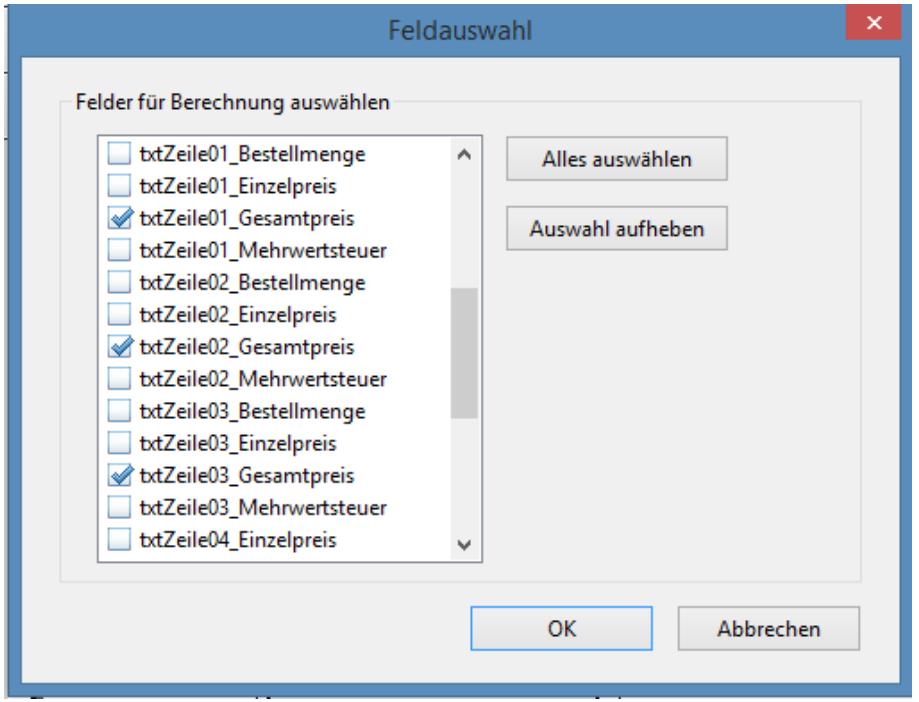
Dialog „Feldauswahl“: Alle Felder nutzen

- Mausklick auf die Schaltfläche *Alles auswählen.* Alle Felder werden mit einem Häkchen versehen. Alle Felder sind ausgewählt
- Klick auf die Schaltfläche *OK.* Die ausgewählten Felder werden mit Hilfe der Funktion zusammengefasst Der Dialog wird geschlossen.



Dialog „Feldauswahl“: Felder auswählen

- Mausklick in eines der Kontrollkästchen rechts vom Feldnamen.
- Andere Möglichkeit: Mausklick auf den Feldnamen. Drücken der <Leertaste>.
- Wenn ein Häkchen im Kästchen vorhanden ist, wird das Feld deaktiviert.
- Wenn das Kästchen leer ist, wird das Feld aktiviert. Ein Häkchen wird angezeigt.



Option „Vereinfachte Feldbezeichnung“

- Klick auf die Schaltfläche *Bearbeiten*.
- In das Textfeld des JavaScript-Editors wird der Ausdruck zur Berechnung eingegeben.
- Die Schaltfläche *OK* schließt den Editor und übernimmt die Eingaben in das entsprechende Textfeld.

Ausdrücke

```
(txtZeile01_Bestellmenge * txtZeile01_Einzelpreis) * 0.19
```

- Nach bestimmten Regeln werden Operanden und Operatoren zusammengesetzt.
- Formulierung eines Rechenschrittes.
- Der Ausdruck gibt einen Wert zurück, der im Textfeld angezeigt wird.
- Ausdrücke können geklammert werden. Die runden Klammern erhöhen die Lesbarkeit bei komplexen Ausdrücken und verändern die Rangfolge der Operatoren.

Operanden

- Feldnamen (Textfeld [Name]; Registerkarte [Allgemein]) als Platzhalter für einen Wert.
- Konstante Werte wie zum Beispiel die Ganzzahl 2, die Dezimalzahl 0.19, etc. Die Werte werden direkt in dem Code angegeben.

Arithmetische Operatoren

*	Multiplikation
/	Division
+	Addition
-	Subtraktion

Option „Benutzerdef. Berechnungsskript“

- Klick auf die Schaltfläche *Bearbeiten*.
- In das Textfeld des JavaScript-Editors wird der Ausdruck zur Berechnung eingegeben.
- Die Schaltfläche *OK* schließt den Editor und übernimmt die Eingaben in das entsprechende Textfeld.

Beispiel

```
var mehrwertsteuer = 0.19;

var einzelpreis = this.getField("txtZeile02_Einzelpreis");
var menge = this.getField("txtZeile02_Bestellmenge");

var bestellpreis = (einzelpreis.value * menge.value);

this.getField("txtZeile02_Mehrwertsteuer").value = bestellpreis * mehrwertsteuer;
```

Operanden in Anweisungen

- Variablen, die in dem JavaScript aus vorhandenen Werten berechnet werden.
- Feldnamen als Platzhalter für einen Wert, den der Nutzer des Formulars eingegeben hat.
- Werte direkt im Code wie zum Beispiel die Ganzzahl 2, die Dezimalzahl 0.19, etc.

Variablen

```
var mehrwertsteuer = 0.19;

var einzelpreis = this.getField("txtZeile02_Einzelpreis");
```

- Speicherung von variablen Werten.
- Platzhalter für einen bestimmten Wert.
- In Abhängigkeit des gespeicherten Wertes haben Variablen einen bestimmten Datentyp.

… deklarieren

var	mehrwertsteuer	;
var	Name	;

- Jede Deklaration beginnt mit dem Schlüsselwort var.
- Dem Schlüsselwort folgt der Name.
- Jede Variable sollte vor der Nutzung deklariert werden.

Hinweise zu Variablennamen

- Der Name muss mit einem Buchstaben, Zahl oder Unterstrich beginnen.
- Variablennamen enthalten nur Buchstaben aus dem englischsprachigen Alphabet, Zahlen und den Unterstrich.
- Die Bezeichnung darf keine Leerzeichen enthalten. Jedes Wort in einer Bezeichnung beginnt mit einem Großbuchstaben.
- Beachtung der Groß- und Kleinschreibung.
- Schlüsselwörter der Sprache JavaScript dürfen nicht als Namen genutzt werden.

Hinweise zur Wahl des Namens

- Die Bezeichnung spiegelt die Nutzung der Variablen wieder.
- Mit Hilfe der Präfixe str (Zeichenkette), num (Zahlen), bool (Ja/Nein) wird der Datentyp des zu speichernden Wertes gekennzeichnet.
- Variablen, die als Zähler genutzt werden, bestehen häufig nur aus einem Buchstaben.

… initialisieren

```
var mehrwertsteuer;
mehrwertsteuer = 0.19;
```

- Eine deklarierte Variable hat einen nicht definierten Wert.
- Vor der Nutzung in Berechnungen wird der Variablen ein definierter Wert zugewiesen. Die Variable wird initialisiert.

… deklarieren und initialisieren

var	mehrwertsteuer	=	0.19	;
var	name	=	wert	;

- Die Anweisung var name deklariert die Variable.
- Die Anweisung name = wert weist der Variablen einen Anfangswert zu.
- Die Anweisungen können zusammengefasst werden. Die Variable wird bekannt gemacht und gleichzeitig initialisiert.
- Der Anfangswert kann jederzeit im Code verändert werden.

Datentyp „Zahlen“

3		Ganzzahl
0.45	.45	Dezimalzahl

- Dezimalzahlen nutzen in JavaScript die amerikanische Notation.
- Als Dezimaltrennzeichen wird der Punkt genutzt.

Positive und negative Zahlen

	Positive Zahl	var zahl = 5;
+	Positive Zahl	var zahl = +5;
-	Negative Zahl	var zahl = -5;

Arithmetische Operatoren

*	Multiplikation	30 = 6 * 5
/	Division	1.2 = 6 / 5
%	Modulo (Rest der Division)	1 = 6 / 5
+	Addition	11 = 6 + 5
-	Subtraktion	1 = 6 - 5

Hinweise zur Division

undefined	=	5	/	0
2.5	=	5	/	2
2.5	=	5.0	/	2

Allgemeine Hinweise

- Felder, die als Zahl formatiert sind, können in Berechnungen genutzt werden.
- Es gilt die Punkt- vor Strichrechnung.
- Komplexe Ausrücke sollten zum besseren Verständnis geklammert werden.

String (Zeichenkette)

```
var besteller = "";
besteller = "Herr ";
```

- Strings bestehen aus Zahlen, Buchstaben, Satzzeichen etc.
- Verkettung von Unicode-Zeichen.
- Strings beginnen und enden mit den Anführungszeichen.

Länge eines Strings

```
var anzahlZeichen = besteller.length;
```

- Jeder String hat das Attribut length.
- Das Attribut gibt die Anzahl der Zeichen in einem String zurück.
- Bei einer Länge von 0 ist der String leer.

… verknüpfen

```
var besteller = this.getField("txtNachname").value;

besteller = this.getField("txtVorname").value + " " +  besteller;
```

- Strings können mit Hilfe des Pluszeichens verknüpft werden.

Zeilenumbruch in Strings

```
var besteller = this.getField("txtNachname").value;

besteller = this.getField("txtVorname").value + " " +  besteller;
besteller = "Frau / Herr\n" + besteller;
```

- Die Escape-Sequenz \n erzeugt einen Zeilenumbruch in einem String.
- Escape-Sequenzen beginnen immer mit dem Backslash.

Escape-Sequenzen

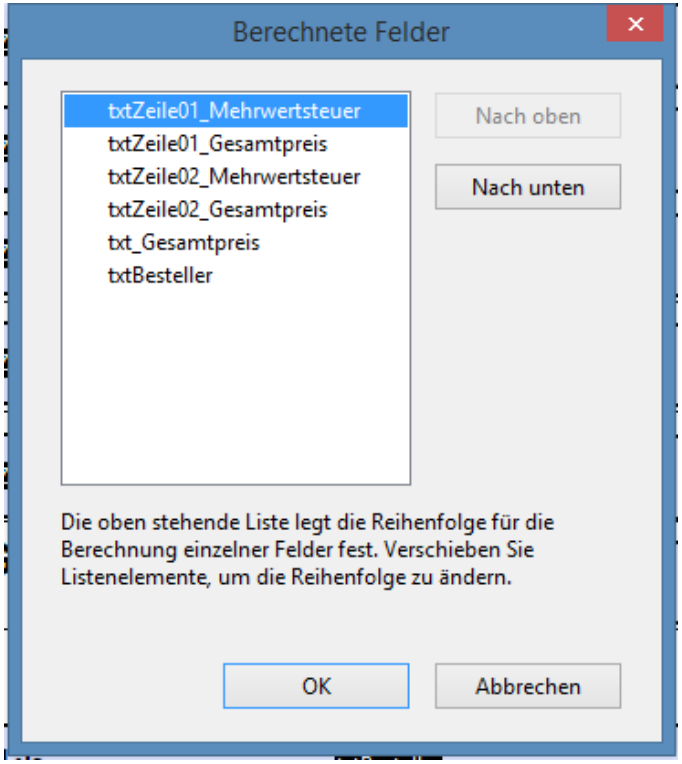
\b	Back space. Gehe einen Schritt zurück.
\f	Form feed. Seitenumbruch.
\n	New Line. Gehe zum Anfang einer neuen Zeile.
\r	Carriage return. Gehe zum Anfang der aktuellen Zeile.
\t	Horizontal Tabulator.

Escape-Sequenzen

\"	Anführungszeichen.
\'	Apostroph.
\\	Umgekehrter Schrägstrich. Backslash.

Feldberechnungsreihenfolge

- Aktivierung eines berechneten Formularfeldes.
- Klick auf den Eintrag *Mehr* im Werkzeugfenster rechts.
- Auswahl des Menüs *Feldberechnungsreihenfolge festlegen*.
- Mit Hilfe der Schaltflächen *Nach oben* und *Nach unten* wird die Reihenfolge der Berechnung festgelegt.
- Standardmäßig werden Berechnungen in der Reihenfolge der Erstellung ausgeführt.



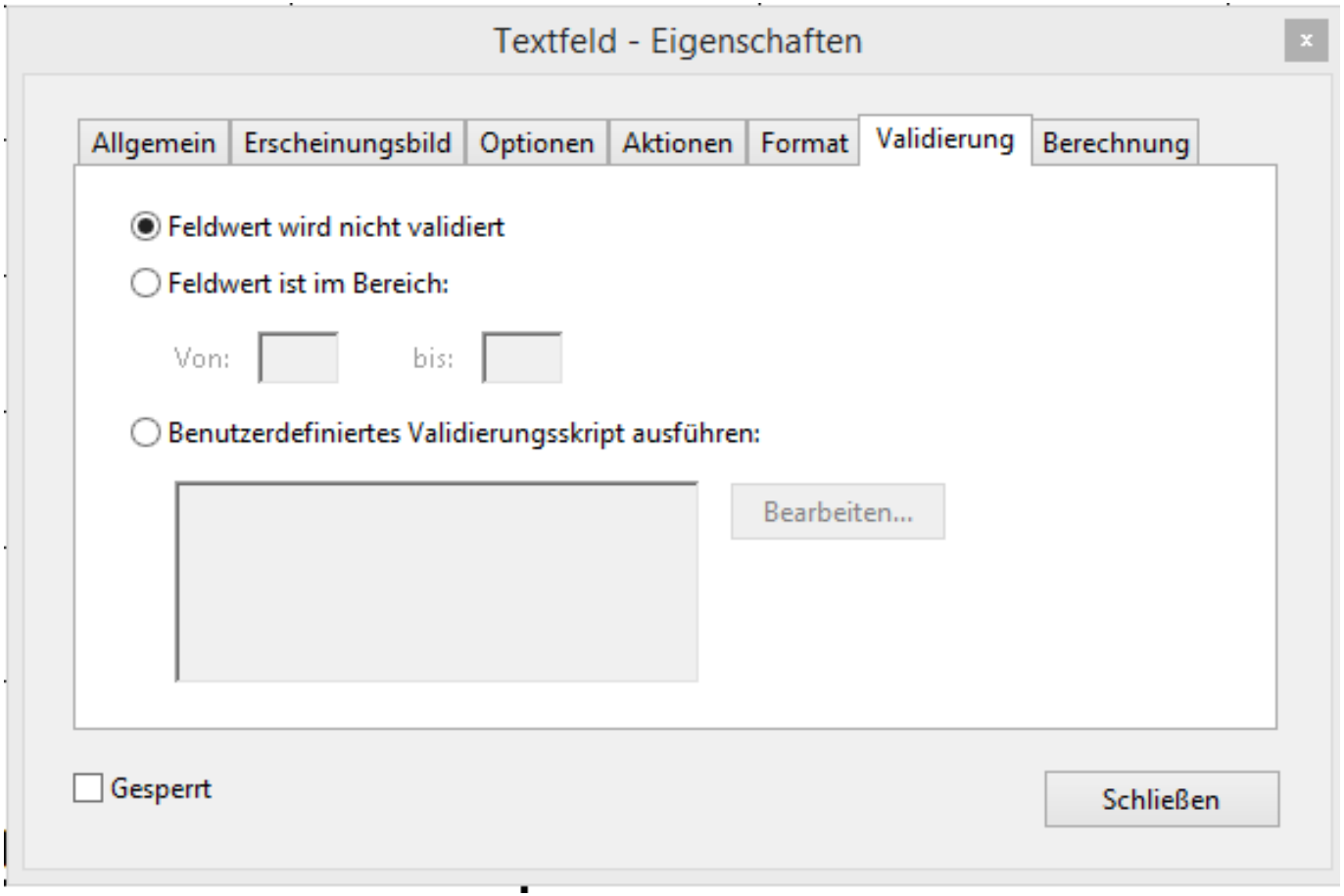
Validierung von Daten

- Gültigkeit der Daten.
- Überprüfung der Daten auf Korrektheit.
- Der Nutzer darf nur bestimmte Werte, Zeichen oder Wertebereiche eingeben.

… eingeben

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über dem Textfeld, deren Daten validiert werden sollen.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet. Der Eintrag *Eigenschaften* wird gewählt.
- Die Registerkarte [Validierung] wird geöffnet.
- Durch einen Klick auf die Schaltfläche *Schließen* wird der Dialog [Textfeld – Eigenschaften] geschlossen.

Registerkarte „Validierung“



Option „Feldwert ist im Bereich“

- Der eingegebene Wert liegt in dem Bereich von … bis ...
- Falls der Wert nicht in dem angegebenen Bereich liegt, wird eine Standard-Warnmeldung „Ungültiger Wert“ ausgegeben.

Option „Benutzerdef. Validierungsskript ausführen“

- Klick auf die Schaltfläche *Bearbeiten*.
- In das Textfeld des JavaScript-Editors wird der Ausdruck zur Berechnung eingegeben.
- Die Schaltfläche *OK* schließt den Editor und übernimmt die Eingaben in das entsprechende Textfeld.

Beispiel-Code

```
event.rc  = true;
var menge = parseInt(event.value);

if(menge <= 0)
{
  app.alert("Es muss mindestens ein Artikel bestellt werden.");
  event.target.textColor = color.black;
  event.rc = false;
}
else if(menge > 10)
{
  var message = "Von einem Artikel können maximal 10 bestellt werden.";
  message = message + " Bitte ändern Sie die Bestellmenge dementsprechend."
  app.alert(message);
  event.target.textColor = color.red;
}
else
{
  event.target.textColor = color.black;
}
```

Wenn … dann - Anweisung

```
if(menge <= 0)
{
  app.alert("Es muss mindestens ein Artikel bestellt werden.");
  event.target.textColor = color.black;
  event.rc = false;
}
```

- Anweisungen werden in Abhängigkeit einer Bedingung ausgeführt.
- Wenn die Menge kleiner als 0 ist, führe den dazugehörigen Anweisungsblock aus.
- Wenn (if) die Bedingung zutrifft, führe die Anweisungen aus ...

Beispiele

- Wenn der String eine Länge größer als 0 hat, dann ...
- Wenn der Wert im Formularfeld kleiner gleich 0 ist, dann ...
- Wenn die Temperatur die Höchstgrenze erreicht hat, dann ...
- Wenn die Warenmenge eine Mindestmenge unterschreitet, dann ...

Anweisungsblock

```
{
  app.alert("Es muss mindestens ein Artikel bestellt werden.");
  event.target.textColor = color.black;
  event.rc = false;
}
```

- Mit Hilfe der geschweiften Klammern werden Anweisungen zusammengefasst.

Bedingungen

```
if(menge <= 0)
```

- Überprüfung von Werten.
- Ausdrücke, die einen boolschen Wert zurückgeben.
- Fragen, die mit ja (true) oder nein (false) beantwortet werden können.
- Bedingungen müssen in if-Anweisungen geklammert werden. Der Anfang und das Ende der Bedingung wird durch runde Klammern gekennzeichnet.

Vergleich von Werten

```
menge <= 0
```

- Mit Hilfe von Vergleichsoperatoren werden zwei Werte miteinander verglichen.
- Der Ausdruck gibt einen boolschen Wert zurück. Der Vergleich ist wahr (true) oder falsch (false).
- Die zu vergleichenden Werte sollten vom gleichen Datentyp sein.

Vergleichsoperatoren

==	Ist gleich	false = (6 == 5)
!=	Ist nicht gleich	true = (6 != 5)
<	Kleiner als	false = (6 < 5)
>	Größer als	true = (6 > 5)
<=	Kleiner gleich als	false = (6 <= 5)
>=	Größer gleich als	true = (6 >= 5)

Verknüpfung von Bedingungen

```
if((menge <= 0) || (menge > 10))
{
  app.alert("Falsche Mengenangabe.");
  event.target.textColor = color.red;
}
```

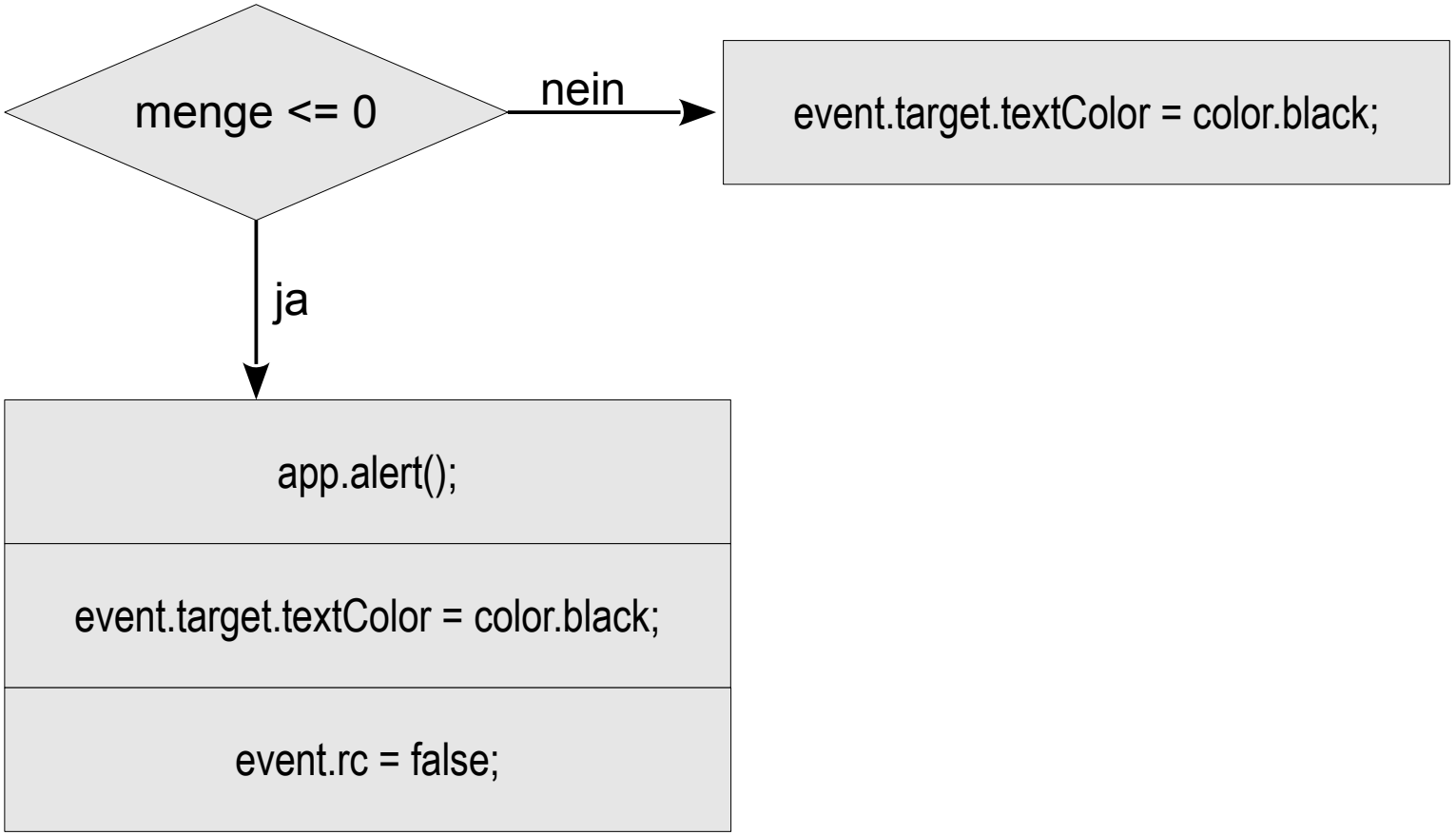
- Oder-Verknüpfung (||). Eine der Bedingungen muss zutreffen.
- Und-Verknüpfung (&&). Alle Bedingungen müssen wahr sein.

Wenn … dann … andernfalls ...

```
if(menge <= 0)
{
  app.alert("Es muss mindestens ein Artikel bestellt werden.");
  event.target.textColor = color.black;
  event.rc = false;
}
else
{
  event.target.textColor = color.black;
}
```

- Wenn (if) die Bedingung zutrifft, führe … aus.
- Andernfalls (else) führe … aus.

Grafische Darstellung



… andernfalls ...

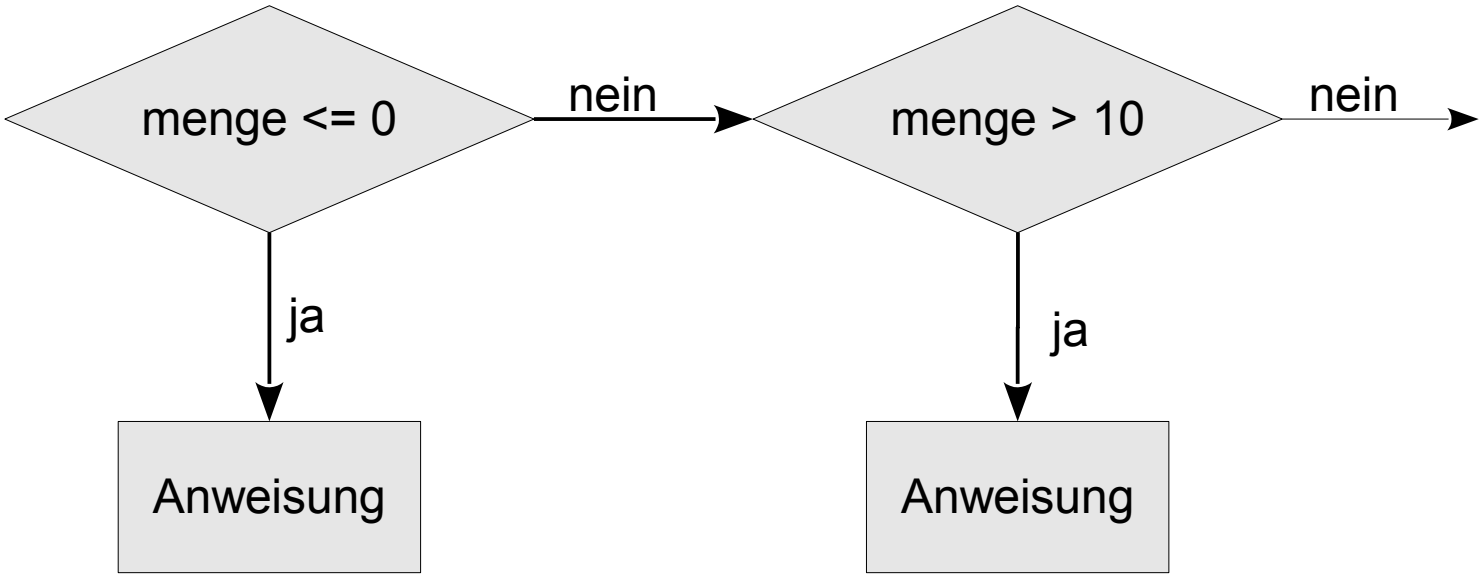
```
if(menge <= 0)
{
}
else
{
  event.target.textColor = color.black;
}
```

- Andernfalls (else) führe … aus.
- Beschreibung des Standardfalls.
- Else-Anweisungen beziehen sich immer auf eine if-Anweisung.
- Die else-Anweisung ist optional.

Fallunterscheidung

```
if(menge <= 0)
{

}
else if(menge > 10)
{
  var message = "Von einem Artikel können maximal 10 bestellt werden.";
  message = message + " Bitte ändern Sie die Bestellmenge dementsprechend."
  app.alert(message);
  event.target.textColor = color.red;
}
else
{

}
```

Grafische Darstellung



Erläuterung

- Im ersten Schritt wird die Bedingung der if-Anweisung überprüft.
- Falls diese nicht zutrifft, wird erste else if-Anweisung überprüft. Wenn die Bedingung zutrifft, wird der dazugehörige Anweisungsblock ausgeführt. Alle nachfolgenden Bedingungen sowie die else-Anweisung werden ignoriert.
- Falls diese nicht zutrifft, wird die nächste else if-Anweisung überprüft.
- Falls keine der Bedingungen zutrifft, wird, falls vorhanden, die else-Anweisung ausgeführt.

if-Anweisungen verschachteln

```
if (nachname.length >  0)
{

   besteller = nachname;
   var vorname = this.getField("txtVorname").value;

   if (vorname.length >  0)
   {
      besteller = vorname + " " +  besteller;
   }

}
```

Einrückungen der Anweisungen

- Um die Lesbarkeit zu erhöhen, sollten die if-Anweisung entsprechend der Verschachtelungsebene eingerückt werden.
- Entsprechend der Einrückungen werden die else if- und else-Anweisungen den if-Anweisungen zugeordnet.
- Mehr als fünf verschachtelte if-Anweisungen führen zu einer Unübersichtlichkeit des Codes.

Ereignis (Event)

- Auslöser auf der Registerkarte [Aktionen] in dem Dialog [Eigenschaften].
- Handlung des Nutzers.
- Mausklick oder Mausbewegung. Aktivierung oder Deaktivierung von zum Beispiel Formularfeldern.
- Mit Hilfe von JavaScript-Code wird auf ein Ereignis reagiert.

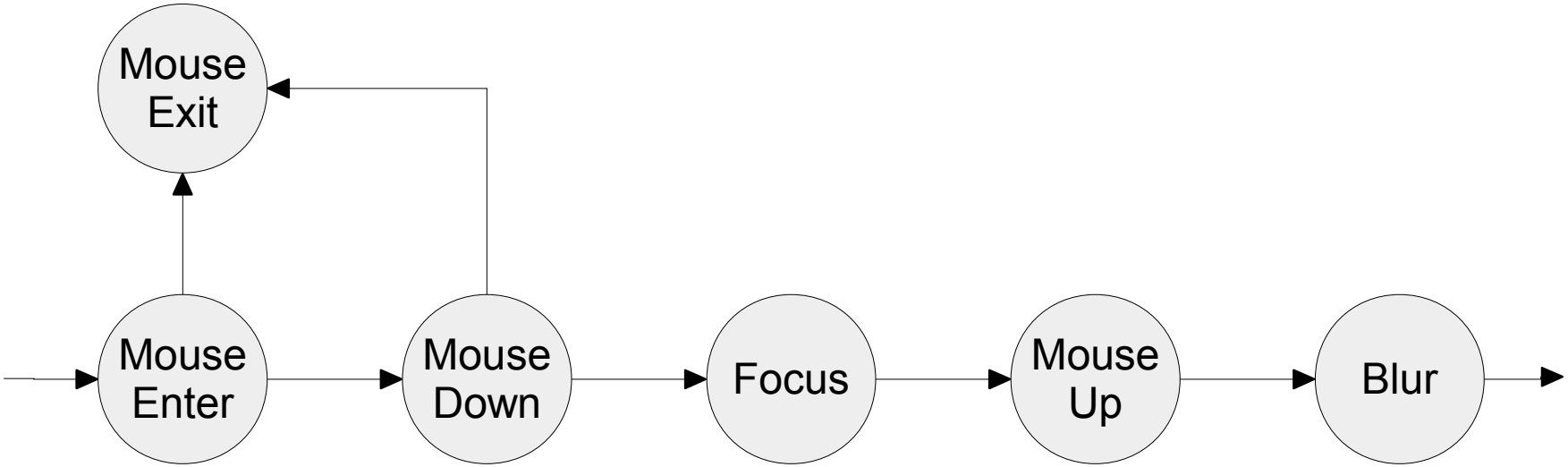
… für Felder

Focus	Feld aktivieren
Blur	Feld deaktivieren
Calculate	Registerkarte [Berechnung]
Format	Registerkarte [Format]
Validate	Registerkarte [Validierung]
Keystroke	Tastendruck Registerkarte [Format]

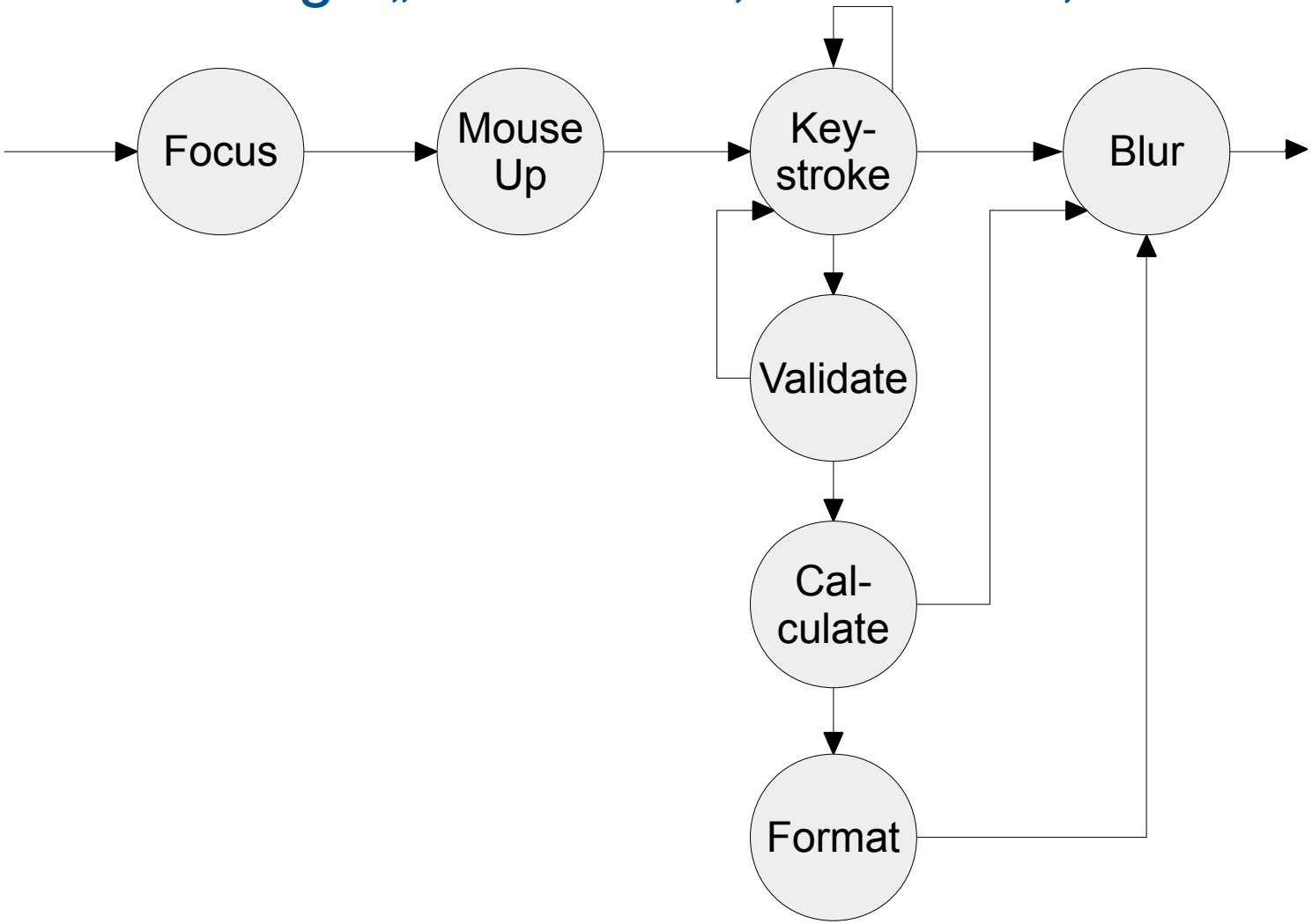
… für Felder

Mouse Down	Maustaste gedrückt
Mouse Up	Maustaste loslassen
Mouse Enter	Maus in Feld
Mouse Exit	Maus aus Feld

Reihenfolge „Maus-Ereignisse“



Reihenfolge „Berechnen, Validieren, Formatieren“



… in JavaScript

```
var message = "Ziel des Ereignisses: " + event.target;
message = message + "\nName des Feldes: " + event.target.name;
message = message + "\nName des Ereignisses: " + event.name;
```

- Das Objekt event wird automatisiert von der Acrobat JavaScript Engine beim Auftreten eines Ereignisses erzeugt.
- Mit Hilfe der Attribute und Methoden wird das ausgelöste Ereignis beschrieben.

Welches Ereignis wurde ausgelöst?

```
message = message + "\nName des Ereignisses: " + event.name;
```

- Objekt und Attribut werden durch den Punktoperator verbunden.
- Attribut .name. Welche Tätigkeit hat der Nutzer durchgeführt? Bezeichnung des Auslösers als String.

Wer ist von dem Ereignis betroffen?

```
var message = "Ziel des Ereignisses: " + event.target;
message = message + "\nName des Feldes: " + event.target.name;
```

- Das Objekt event.target gibt einen Verweis auf das betroffene Formularfeld zurück.
- Das Attribut event.target.name gibt den Namen des Feldes zurück. Der Nutzer interagiert mit diesen Feld.

Nutzereingabe im Feld

```
var menge = parseInt(event.value);

event.value = 1;
```

- Das Attribut event.value gibt den eingegebenen Wert zurück.
- Der Attribut wird in Abhängigkeit des ausgelösten Ereignisses gesetzt.
- Das Attribut wird nur von den Ereignissen validate, calculate, keystroke und blur gesetzt. Bei allen anderen Ereignissen wird ein leerer String zurückgegeben.

Wert des Formularfeldes

```
var menge = parseInt(event.target.value);

event.target.value = 1;
```

- Das Objekt event.target gibt einen Verweis auf den Auslöser des Ereignisses zurück.
- In Abhängigkeit des Verweises gibt das Attribut event.target.value den aktuellen Feld zurück.

… in eine Ganzzahl konvertieren

```
var menge = parseInt(event.value);
```

- Die vordefinierte Funktion parseInt() konvertiert einen String in eine Ganzzahl (Integer).
- Die Funktion wird mit den Namen aufgerufen. In JavaScript wird die Groß- und Kleinschreibung beachtet.
- Der zu konvertierende Wert wird als Parameter in den runden Klammern der Funktion übergeben.
- Falls eine Konvertierung nicht möglich ist, wird NaN (Not a number) zurückgegeben.

Name einer Funktion

var	menge	=	parseInt	(	event.value	)	;
			name				

- Mit Hilfe des Namens wird eine definierte Funktion aufgerufen.
- Die Groß- und Kleinschreibung wird beachtet. Die Methode .ParseInt() ruft eine andere Methode als .parseInt() auf.

Parameter einer Funktion

var	menge	=	parseInt	(	event.value	)	;
			name	(	parameter	)	

- Dem Namen folgen direkt runde Klammern.
- Falls die runden Klammern leer sind, werden der Methode keine Startwerte übergeben.
- Wie viele Parameter der Methode übergeben werden, ist in der Deklaration der Funktion definiert.
- In diesem Beispiel wird der Methode .parseInt() ein Wert in Abhängigkeit der Handlung des Nutzers übergeben.

Rückgabewert einer Methode

var	menge	=	parseInt	(	event.value	)	;
rückgabewert		=	name	(	parameter	)	

- Eine Funktion kann einen Wert an den Aufrufer zurückgeben.
- Der Rückgabewert wird mit Hilfe des Gleichheitszeichens einer Variablen zugewiesen.
- In diesem Beispiel wird versucht den Parameter in eine Ganzzahl zu konvertiert. Das Ergebnis der Konvertierung wird in einer Variablen gespeichert.

Beispiel

12	=	parseInt("12")
12	=	parseInt("12.54")
12	=	parseInt("12 13")
12	=	parseInt(" 12")
12	=	parseInt("12 Stück")
NaN	=	parseInt("Menge 12")
undefined	=	parseInt("A")

Wurde der String konvertiert?

```
var menge = parseInt(event.value);

if (isNaN(menge) == true)
{
   var strPreis = this.getField("txtZeile02_Einzelpreis").value;
   var curPreis = parseFloat(strPreis).toFixed(2);

}
```

- Die vordefinierte Funktion isNaN() fragt ab, ob der String umgewandelt wurde oder nicht.
- Funktionen, die mit „is“ beginnen, liefern in den meisten Fällen einen boolschen Wert zurück. Falls der Wert false zurückgeliefert wird, enthält der String eine Ganzzahl. Andernfalls ist die Konvertierung fehlgeschlagen.

… in eine Dezimalzahl konvertieren

```
var curPreis = parseFloat(strPreis).toFixed(2);
```

- Die vordefinierte Funktion parseFloat() konvertiert einen String in eine Dezimalzahl (Float).
- Der zu konvertierende Wert wird als Parameter in den runden Klammern der Funktion übergeben.

Anzahl der Dezimalstellen

```
var curPreis = parseFloat(strPreis).toFixed(2);
```

- Für Dezimalzahlen kann die Anzahl der Nachkommastellen mit Hilfe der Funktion toFixed() festgelegt werden.
- Der Parameter der Funktion legt die Anzahl der Nachkommastellen eines Wertes fest. In diesem Beispiel werden zwei Nachkommastellen angezeigt.
- Die Zahl wird entsprechend des Parameters gerundet.

Eingabe akzeptieren

```
event.rc = true;
event.rc = false;
```

- Standardmäßig hat das Attribut rc den Wert true. Der, durch das Ereignis geänderte Wert wird angezeigt.
- Bei einem Wert von false wird das Ereignis abgebrochen. Der, durch das Ereignis geänderte Wert, wird widerrufen. Der Wert, vor Auslösung des Ereignisses wird angezeigt
- Das Attribut wird von den Ereignissen validate, calculate und keystroke beachtet.

Ist das Textfeld leer?

```
var ausgabeText = "";
var quelle = null;

quelle = this.getField("kategorie4.frage42");

if((quelle.value != " ") && (quelle.value.length > 0))
{
   ausgabeText = quelle.value;
   app.alert({cMsg:ausgabeText, nIcon:3, cTitle:"Weitere Themen"});
}
```

Erläuterung der Bedingung

```
if((quelle.value != " ") && (quelle.value.length > 0))
```

- Wenn das Textfeld nicht nur ein Leerzeichen enthält (quelle.value != " ") ...
- Wenn das Textfeld mindestens ein Zeichen enthält (quelle.value.length > 0) ...
- Die zwei Bedingungen werden mit Hilfe des Und-Operators (&&) verknüpft. Beide Bedingungen müssen wahr sein. Beide Bedingungen muss der Inhalt des Formularfeldes erfüllen.

Auswertung von Optionsfeldern u. Kontrollkästchen

```
var ausgabeText = "";
var quelle = null;

quelle = this.getField("kategorie1.frage11");

if(parseInt(quelle.value) > 0)
{
   ausgabeText = "Beurteilung: " + quelle.value;
   app.alert({cMsg:ausgabeText, nIcon:3, cTitle:"Frage 1"});
}
```

Rückgabewerte der aktiven Felder

- Das aktive Kontrollkästchen gibt den Exportwert (Registerkarte [Optionen]) zurück.
- Eine aktives Optionsfeld in einer Gruppe gibt die Optionsfeldauswahl (Registerkarte [Optionen]) zurück.
- Nachteil: Wenn die Angaben in den entsprechenden Registerkarten geändert werden, muss auch das JavaScript angepasst werden.

Exportwert eines Kontrollkästchens

```
var ausgabeText = "";
var quelle =  this.getField("kategorie4.frage41.checkbox#1");
var aktivStatus = quelle.exportValues;

if(quelle.value == aktivStatus)
{
    ausgabeText = "Beurteilung: " + quelle.value;

    app.alert({cMsg:ausgabeText, nIcon:3, cTitle:"Frage 4 "});
}
```

- Das Attribut .exportValues gibt den Exportwert eines Kontrollkästchens zurück.

Ist das Kontrollkästchen / Optionsfeld aktiv?

```
var ausgabeText = "";
var quelle = null;

quelle = this.getField("kategorie3.frage34#1");

if( quelle.isBoxChecked(0) == true)
{
    ausgabeText = "Beurteilung: " + quelle.value;

    app.alert({cMsg:ausgabeText, nIcon:3, cTitle:"Frage 4"});
}
```

Erläuterung der Bedingung

```
if( quelle.isBoxChecked(0) == true)
```

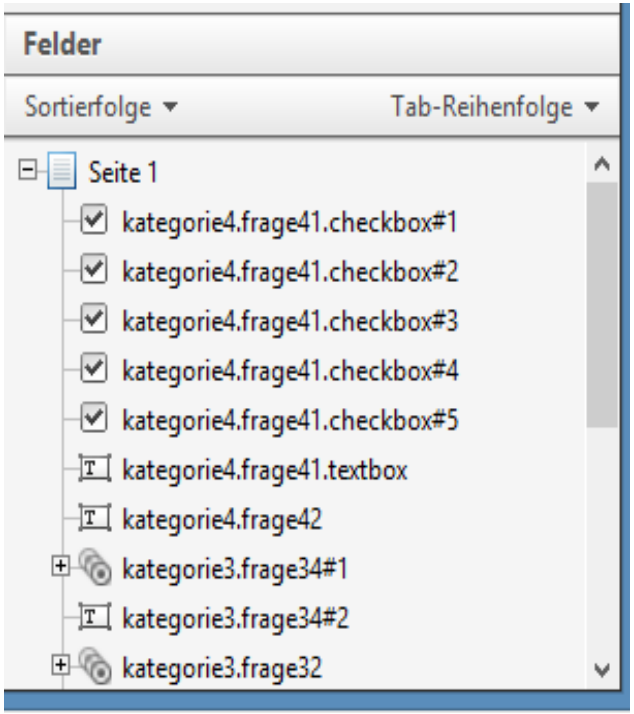
- Die Methode .isBoxChecked() eines Kontrollkästchens oder Optionsfeldes prüft, ob das Feld aktiv ist oder nicht.
- Der Methode wird in den runden Klammern ein Index in Abhängigkeit der Erstellung übergeben. Das erste Element, welches erstellt wurde, hat den Index 0. Hinweis: Der benötigte Index kann nicht mit Hilfe einer Methode ermittelt oder verändert werden.
- Wenn die Methode true zurück gibt, ist das Feld aktiv.

Weitere Möglichkeit

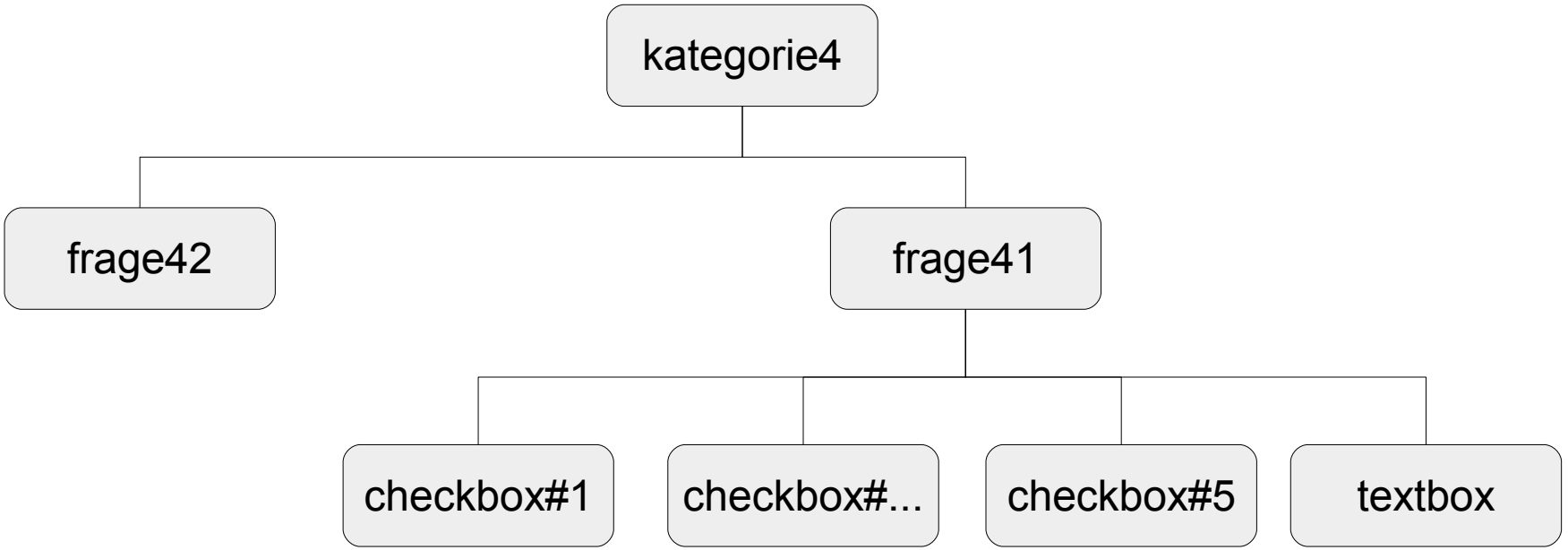
```
var parent = this.getField("kategorie4.frage41");
var children = parent.getArray();
var ausgabeText = "";
var child = null;

for (var index = 0; index < children.length; index++)
{
   child = children[index];

   if (child.type == "checkbox")
   {
      .....
   }
   else if (child.type == "text")
   {
     ausgabeText = ausgabeText + child.value;
   }

}
```

Liste „Felder“ in dem Werkzeugfenster

- Die Felder werden seitenweise angezeigt.
- Felder zu einem Thema oder Frage haben die gleiche Wurzel.
- Die verschiedenen Hierarchieebenen werden durch den Punkt dargestellt.
- Jeder Name bezeichnet eine Ebene innerhalb der Hierarchie.
- Das Hashzeichen plus einer laufenden Nummer kennzeichnet Felder in der selben Ebene.

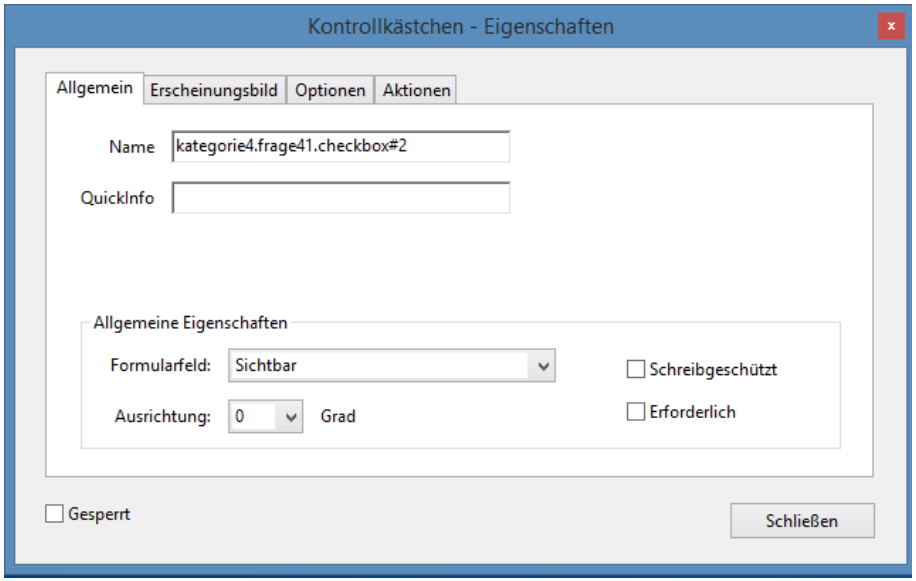


Beispiel


kategorie4
frage42
frage41
checkbox#1
checkbox#...
checkbox#5
textbox

Eingabe des Feldnamens

- Der Mauszeiger schwebt über einem Formularfeld.
- Klick mit der rechten Maustaste. Das dazugehörige Kontextmenü geöffnet.
- Auswahl des Eintrages *Eigenschaften*.
- Die Registerkarte [Allgemein] ist aktiv.
- Die Hierarchieebenen und der Feldname werden vollständig eingegeben.



Verweis auf die Wurzel (Eltern)

```
var parent = this.getField("kategorie4.frage41");
```

- Mit Hilfe der Methode getField() wird ein Verweis auf die Wurzel in der Variablen parent gespeichert.

Eine Ebene unterhalb der Wurzel (Kinder)

```
var children = parent.getArray();
```

- Die Methode getArray() eines Objektes vom Typ „Formularfeld“ gibt Verweise auf Felder eine Hierarchieebene tiefer zurück.
- Die Methode getArray() gibt alle Kinder der Eltern kategorie4.frage41 zurück.

Array von ...


[0]
checkbox#1
[1]
checkbox#2
checkbox#3
checkbox#4
checkbox#5
[5]
textbox
var children = "kategorie4.frage41".getArray()

Array

- Liste von Werten, die unter einem Namen abgespeichert werden.
- Sammlung von Werten zu einem Thema. Zum Beispiel alle möglichen Antworten zu einer Frage.
- Zusammenfassung von Variablen. In diesem Beispiel werden Verweise auf Objekte vom Typ „Formularfeld“ zusammengefasst.
- Gruppierung von Elementen von einer bestimmten Kategorie.

Elemente eines Arrays

- Jedes Element in einem Array hat einen bestimmten Wert.
- Die Elemente in einem Array müssen nicht den gleichen Datentyp haben.
- Die Elemente in einem Array werden von 0 bis n durchnummeriert. Das erste Element hat den Index 0. Das letzte Element hat den Index (Länge des Arrays) – 1.

Index

```
var child = null;
child = children[index];
child = children[0];
```

- Der Index des Elements wird direkt im Anschluss an den Namen des Arrays in eckigen Klammern angegeben.
- Der Index kann als Ganzzahl oder mit Hilfe einer Variablen angegeben werden.

Wert eines Elements

```
var child = null;
child = children[index];
```

- In diesem Beispiel wird der Wert eines Array-Elements einer Variablen zugewiesen.
- Der Wert eines Elementes kann genauso wie bei Variablen oder Attributen durch einen Ausdruck verändert oder einer anderen Variablen zugewiesen werden.
- Array-Elemente werden wie variable Werte genutzt.

Jedes Element in einem Array ...

```
for (var index = 0; index < children.length; index++)
{
   child = children[index];

}
```

- Mit Hilfe einer for-Schleife wird ein Array vollständig durchlaufen.
- In Abhängigkeit der Bedingung in dem Schleifenkopf wird der dazugehörige Anweisungsblock ausgeführt.

Zählschleife

for (var index = 0; index < children.length; index++)	Schleifenkopf

{
child = children[index];
}
	Schleifenrumpf (Anweisungsblock)

- Der Schleifenkopf beginnt mit dem Schlüsselwort for. Dem Schlüsselwort folgen in runden Klammern drei Anweisungen.
- Der Schleifenrumpf fasst die zu durchlaufenden Anweisungen zusammen. In Abhängigkeit eines Zählers wird der Schleifenrumpf durchlaufen.

Anweisungen im Schleifenkopf

for	(	int index = 0;	index < children.length;	index++	)
		Initialisierung;	Bedingungen;	Reinitialisierung	

- Mit Hilfe der runden Klammern werden drei Anweisungen zusammengefasst.
- Die drei Anweisungen beeinflussen das Verhalten der Zählschleife.
- Die ersten zwei Anweisungen müssen mit einem Semikolon abgeschlossen werden.

1. Anweisung: Initialisierung der Zählvariablen

for	(	var index = 0;	)
for	(	index = 0;	)

- In der ersten Anweisung wird die Zählvariable initialisiert. In diesem Beispiel bekommt die Variable index den Startwert 0 zugewiesen.
- In der ersten Zeile wird die Zählvariable gleichzeitig deklariert und initialisiert. Die Variable ist nur in der Schleife bekannt.
- In der zweiten Zeile wird der Zählvariablen ein Anfangswert zugewiesen. Vor der Nutzung wurde die Variable deklariert.

2. Anweisung: Bedingung

for	(	int index = 0;	index < children.length;	)

- Die zweite Anweisung legt die Anzahl der Durchläufe mit Hilfe einer Bedingung fest.
- Solange wie die Bedingung erfüllt ist, wird der Schleifenrumpf ausgeführt.
- In diesem Beispiel wird mit Hilfe des Attributs .length die Anzahl der Elemente in dem Array ermittelt. Solange der Index kleiner als die Anzahl der Elemente ist, wird der Schleifenrumpf ausgeführt.

1. Anweisung: Reinitialisierung der Zählvariablen

for	(	int index = 0;	index < children.length;	index++	)

- Die dritte Anweisung berechnet den Wert für den nächsten Durchlauf.
- Der neu berechnete Wert und der Überprüfungswert sollten vom gleichen Typ sein.
- In diesem Beispiel wird mit Hilfe des Inkrement-Operators ++ die Variable um eins erhöht. Das nächste Element im Array wird bearbeitet.

Wenn das Feld vom Typ … ist, dann ...

```
if (child.type == "checkbox")
{

}
```

- Das Attribut .type gibt den Feldtyp als String zurück.
- Beispiel: Wenn das Feld vom Typ „Kontrollkästchen“ ist, dann...

Möglichkeiten

Kontrollkästchen	"checkbox"
Optionsfeld	"radiobutton"
Textfeld	"text"
Listenfeld	"listbox"
DropDown-Listen	"combobox"
Schaltfläche	"button"
Unterschriften-Feld	"signature"

Reguläre Ausdrücke

```
var musterPlz = /30159/;
var reZahlen = /^\d$/;
var reAll = /^.$/;
```

- String von Zeichen, die ein Muster repräsentieren.
- Beschreibung einer Zeichenkette mit Hilfe von Platzhaltern, Buchstaben, Zahlen, Satzzeichen etc.

… nutzen

- Durchsuchen von String nach einem bestimmten Muster.
- Validierung eines Textes mit Hilfe eines Musters.
- Validierung direkt bei Eingabe des Zeichens.

… in JavaScript

```
var musterPlz = /30159/;
var reZahlen = /^\d$/;
var reAll = /^.$/;
```

- Beginn und Ende mit Hilfe der Schrägstriche.
- Zwischen den Schrägstrichen können Zeichen der Tastatur, Platzhalter für eine bestimmte Gruppe von Zeichen etc. in einer bestimmten Reihenfolge stehen.
- Bei der Anwendung wird die Groß- und Kleinschreibung der Zeichen beachtet.
- Objekt RegExp in JavaScript. Siehe https://wiki.selfhtml.org/wiki/JavaScript/Objekte/RegExp.

Beispiele

```
var musterPlz = /30159/;
var reZahlen = /^\d$/;
var reAll = /^.$/;
```

- In dem ersten Beispiel wird das Muster „30159“ gesucht.
- In dem zweiten Beispiel wird ein Zeichen gesucht. Das Zeichen muss eine Zahl von 0 bis 9 sein.
- In dem dritten Beispiel wird ein beliebiges Zeichen gesucht. Der zu testende String darf nur eine Länge von ein Zeichen haben.

Zeichenklassen

\d	Zahlen von 0 bis 9.
\D	Alle anderen Zeichen bis auf 0 bis 9.
\w	Buchstaben A-Z und a-z. Zahlen 0-9. Unterstrich.
\W	Alle anderen Zeichen bis auf den Unterstrich, die Zahlen 0-9, die Buchstaben A-z und a-z.
\s	White spaces (Leerzeichen, Tabulator, Zeilenumbruch, Zeilenvorschub).
\S	Alle Zeichen bis auf White spaces.

Zeichenklassen

.	Ein beliebiges Zeichen außer der Zeilenumbruch.
^	Der Beginn des Strings.
$	Das Ende des Strings.
[xyz]	Eines der Zeichen in den eckigen Klammern.
[^xyz]	Alle Zeichen bis auf die Zeichen in den eckigen Klammern.
[0-9] [a-d]	Zeichen in dem Bereich von … bis ...

Quantifizierer

?	Das, vor dem Fragezeichen stehende Elemente kommt nicht oder maximal einmal vor.
+	Das, vor dem Pluszeichen stehende Elemente kommt mindestens einmal vor.
*	Das, vor dem Sternchen stehende Elemente kommt nicht oder beliebig oft vor.
{n}	Die, vor den geschweiften Klammern stehende Elemente kommen genau n Mal vor.

Objekt „RegExp“

```
var strPlz = event.value;

var muster = /[0-9]{5}/;
var ergebnis = muster.test(strPlz);
```

- Siehe https://wiki.selfhtml.org/wiki/JavaScript/Objekte/RegExp.
- Das Objekt stellt Methoden und Attribute zum Arbeiten bereit.

Ist das Suchmuster vorhanden?

```
var strPlz = event.value;

var muster = /[0-9]{5}/;
var ergebnis = muster.test(strPlz);
```

- Die Methode wird mit einem regulären Ausdruck durch einen Punkt verbunden. Links vom Punkt steht ein Platzhalter für ein Objekt vom Typ „RegExp“. Rechts vom Punkt steht die anzuwendende Methode.
- Die Methode .test() überprüft, ob ein Muster in einem String vorkommt. Der Methode wird der, zu überprüfende String strPlz in runden Klammern übergeben. Falls das Muster in dem String vorhanden ist, wird true zurückgeliefert.

Nutzung von String-Methoden

```
var strPlz = event.value;

var ergebnis = strPlz.search(/[0-9]{5}/);
var ergebnis = strPlz.match(/[0-9]{5}/);
```

- Die Methoden und der String werden mit dem Punktoperator verbunden.
- Links vom Punkt steht ein Platzhalter für eine Zeichenkette. Die Methode wird auf dieses Objekt angewendet.
- Rechts vom Punkt steht die anzuwendende Methode.

Position eines Suchmusters in einem String

```
var strPlz = event.value;

var ergebnis = strPlz.search(/[0-9]{5}/);
```

- Der Methode .search() wird ein regulärer Ausdruck in den runden Klammern übergeben.
- Die Methode liefert die Position des ersten Auftretens des Musters in dem, zu durchsuchenden String zurück.
- Falls das Suchmuster nicht vorhanden ist, wird -1 zurückgegeben.

Übereinstimmende Ausdrücke

```
var strPlz = event.value;

var ergebnis = strPlz.match(/[0-9]{5}/);
```

- Der Methode .match() wird ein regulärer Ausdruck in den runden Klammern übergeben.
- Die Methode gibt alle Zeichenfolgen, auf die das Suchmuster passt, zurück.

Validierung bei Abschluss der Eingabe

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über dem Textfeld, deren Tastatureingaben überprüft werden sollen.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet. Der Eintrag *Eigenschaften* wird gewählt.
- Die Registerkarte [Validierung] wird geöffnet.
- Die Formatkategorie [Benutzerdefiniert] wird ausgewählt.
- Durch ein Klick auf die Schaltfläche *Bearbeiten* wird ein Tasteneingabeskript in den JavaScript-Editor eingegeben.

Beispiel-Code

```
var strPlz = event.value;

if(strPlz.search(/[0-9]{5}/) == 0)
{
    event.rc = true;
}
else
{
    event.rc = false;
    app.alert("Postleitzahlen bestehen aus 5 Zahlen");
}
```

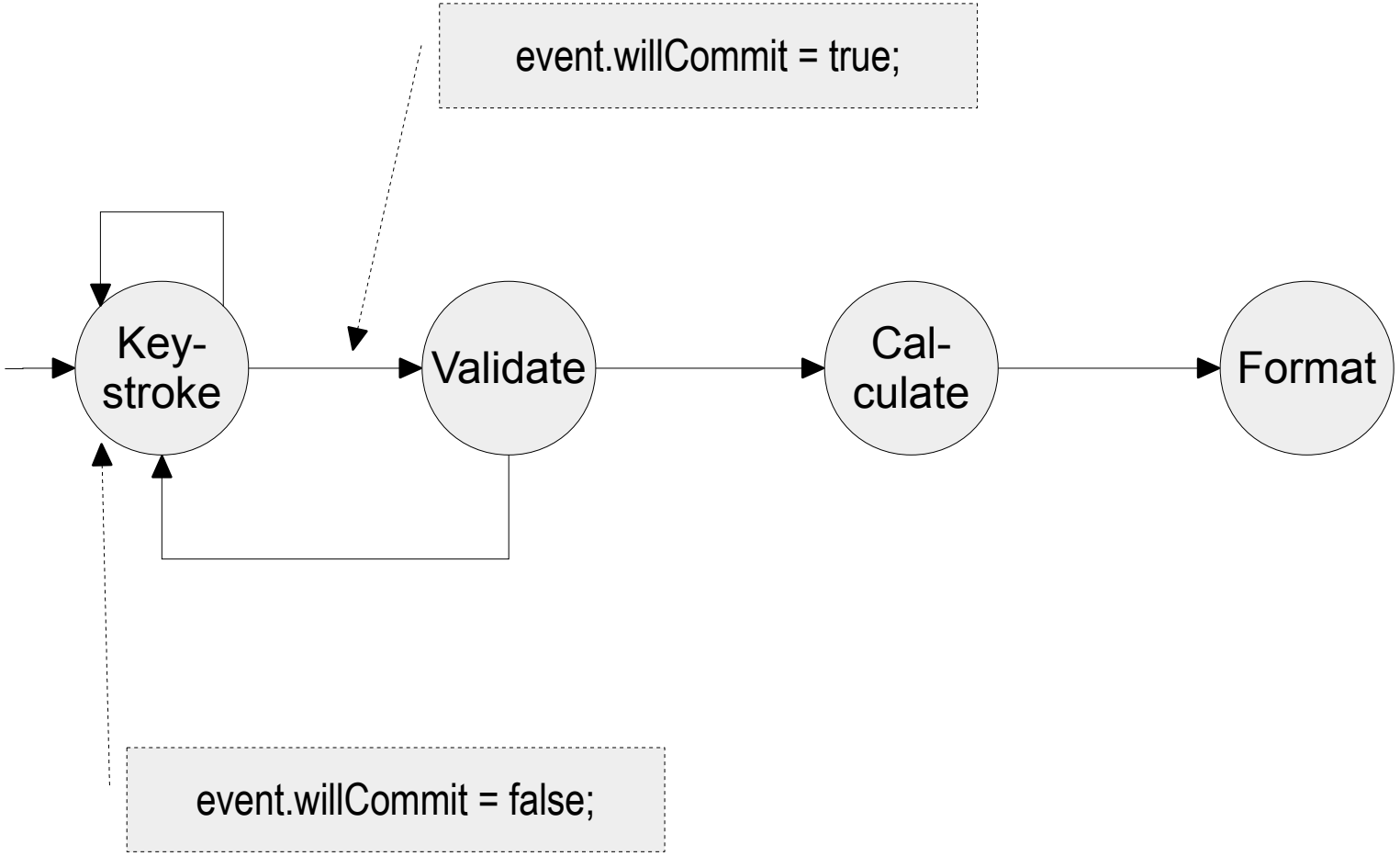
Validierung bei der Eingabe eines Zeichens

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über dem Textfeld, deren Tastatureingaben überprüft werden sollen.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet. Der Eintrag *Eigenschaften* wird gewählt.
- Die Registerkarte [Format] wird geöffnet.
- Die Formatkategorie [Benutzerdefiniert] wird ausgewählt.
- Durch ein Klick auf die Schaltfläche *Bearbeiten* wird ein Tasteneingabeskript in den JavaScript-Editor eingegeben.

Beispiel-Code

```
var reZahlen = /^\d$/;
var reAll = /^.$/;

if(event.willCommit == false)
{
    if (reAll.test(event.change) == true)
    {
        if(reZahlen.test(event.change) == false)
        {
            app.alert("Es sind nur Zahlen zwischen 0 und 9 erlaubt");
            event.rc = false;
        }

    }
}
```

Wird das Zeichen am Bildschirm angezeigt?



Erläuterung

```
event.willCommit = false;
```

- Der Nutzer hat eine Taste gedrückt. Aber die Eingabe wurde noch nicht an das Formularfeld übergeben.
- Das Attribut .willCommit wird von den Ereignissen format und keystroke gesetzt.

Anzuzeigende Daten

```
var blnProof = reAll.test(event.change);
```

- Das Attribut change liest die anzuzeigenden Daten ein.
- Bei einem Tastendruck enthält das Attribut ein Zeichen.
- Falls die Daten aus der Zwischenablage in das Feld kopiert werden sollen, enthält das Attribut eine Zeichenkette.

Formatierung von Eingaben

- Das Formular ist im Bearbeitungsmodus geöffnet.
- Der Mauszeiger liegt über dem Textfeld, deren Tastatureingaben überprüft werden sollen.
- Mit Hilfe der rechten Maustaste wird das dazugehörige Kontextmenü geöffnet. Der Eintrag *Eigenschaften* wird gewählt.
- Die Registerkarte [Format] wird geöffnet.
- Die Formatkategorie [Benutzerdefiniert] wird ausgewählt.
- Durch ein Klick auf die Schaltfläche *Bearbeiten* wird ein benutzerdefiniertes Formatierungsskript in den JavaScript-Editor eingegeben.

Beispiel-Code

```
var strIBAN = event.value;
var reIBAN = /^(DE\d{2})(\d{4})(\d{4})(\d{4})(\d{4})(\d{2})$/;
var reIBANFormat = /^(DE\d{2}) (\d{4}) (\d{4}) (\d{4}) (\d{4}) (\d{2})$/;

if(reIBANFormat.test(strIBAN) == false)
{
  if(reIBAN.test(strIBAN) == true)
  {
    var formatIBAN = RegExp.$1;
    formatIBAN = formatIBAN + " " + RegExp.$2  + " " + RegExp.$3;
    formatIBAN = formatIBAN + " " + RegExp.$4 + " " + RegExp.$5 + " " + RegExp.$6;

    event.value = formatIBAN;
  }
  else
  {
    var message = "Die IBAN beginnt mit den zwei Buchstaben DE";
    app.alert(message);
  }
}
```

Teilung des regulären Ausdrucks

```
var reIBAN = /^(DE\d{2})(\d{4})(\d{4})(\d{4})(\d{4})(\d{2})$/;

if(reIBAN.test(strIBAN) == true)
  {
    var formatIBAN = RegExp.$1;
    formatIBAN = formatIBAN + " " + RegExp.$2  + " " + RegExp.$3;
    formatIBAN = formatIBAN + " " + RegExp.$4 + " "
    formatIBAN = formatIBAN + RegExp.$5 + " " + RegExp.$6;

    event.value = formatIBAN;
  }
```

„Wörter“ in einem Muster

^	(DE\d{2})	(\d{4})	(\d{4})	(\d{4})	(\d{4})	(\d{2})	$
	$1	$2	$3	$4	$5	$6	
	RegEx						

- In den runden Klammern werden „Wörter“ eines Suchmusters definiert.
- Ein regulärer Ausdruck kann sich aus bis zu neun „Wörter“ zusammensetzen.
- In diesem Beispiel besteht das erste und letzte Wort aus zwei Zahlen. Alle anderen Wörter dazwischen bestehen aus 4 Zahlen.

Platzhalter für die „Wörter“

^	(DE\d{2})	(\d{4})	(\d{4})	(\d{4})	(\d{4})	(\d{2})	$
	$1	$2	$3	$4	$5	$6	
	RegEx						

- Jedes Wort kann mit Hilfe eines Index angesprochen werden.
- Die Wörter werden von links nach rechts nummeriert.
- Der Platzhalter für die Wörter setzt sich aus dem Dollarzeichen und der fortlaufenden Nummer zusammen.

Seitenaktionen

- Jede Seite im Dokument die Ereignisse Seite öffnen und Seite schließen.
- Das Dokument selber hat kein Ereignis „Öffnen“. Der ersten Seite muss dementsprechend ein JavaScript zugeordnet werden. Das Dokument muss mit dieser Seite immer geöffnet werden (*Datei* – *Eigenschaften*. Registerkarte [Ansicht beim Öffnen]).

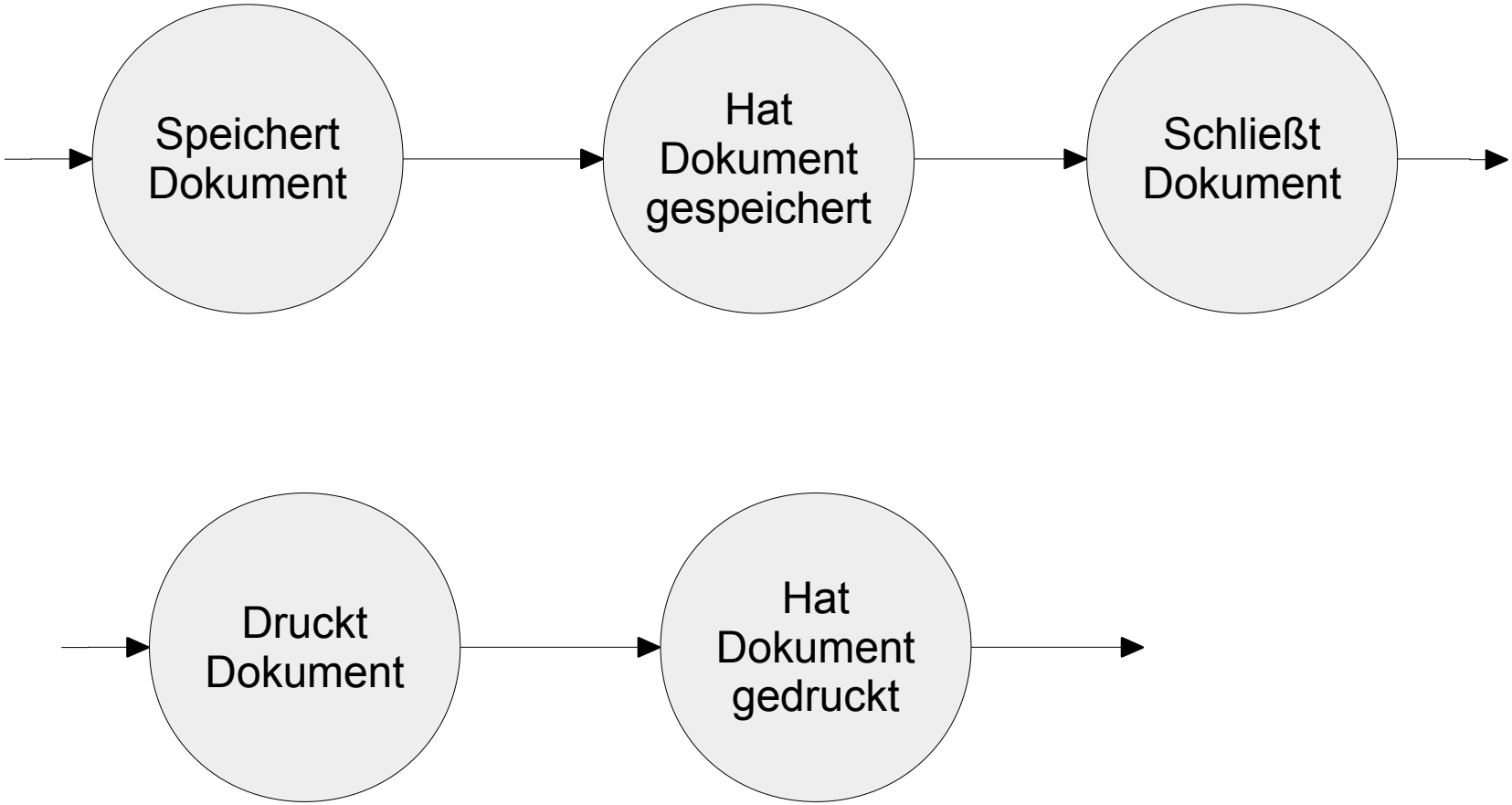
Code einhängen

- Das Dokument ist geöffnet.
- Die [Seitenminiaturen] werden im Navigationsfenster links angezeigt.
- Rechter Mausklick auf die erste Seite.
- Im Kontextmenü wird der Eintrag *Seiteneigenschaften* ausgewählt.
- Im Dialog [Seiteneigenschaften] ist die Registerkarte [Aktionen] aktiv.
- Der Auslöser [Seite öffnen] oder [Seite schließen] wird ausgewählt.
- Als Aktion wird [JavaScript ausführen] gewählt.
- Klick auf die Schaltfläche *Hinzufügen*.
- In dem JavaScript-Editior wird der benötigte Code eingegeben.

Beispiel-Code

```
var gesamtpreis = parseFloat(this.getField("txt_Gesamtpreis").value);

if(isNaN(gesamtpreis) == true)
{
  this.getField("txtBestelltAm").value = "Bestelldatum";
  this.getField("txtLieferungAm").value = "Lieferdatum";
  this.getField("txtDruckdatum").value = "Gedruckt am";

}
```

Dokumentaktionen

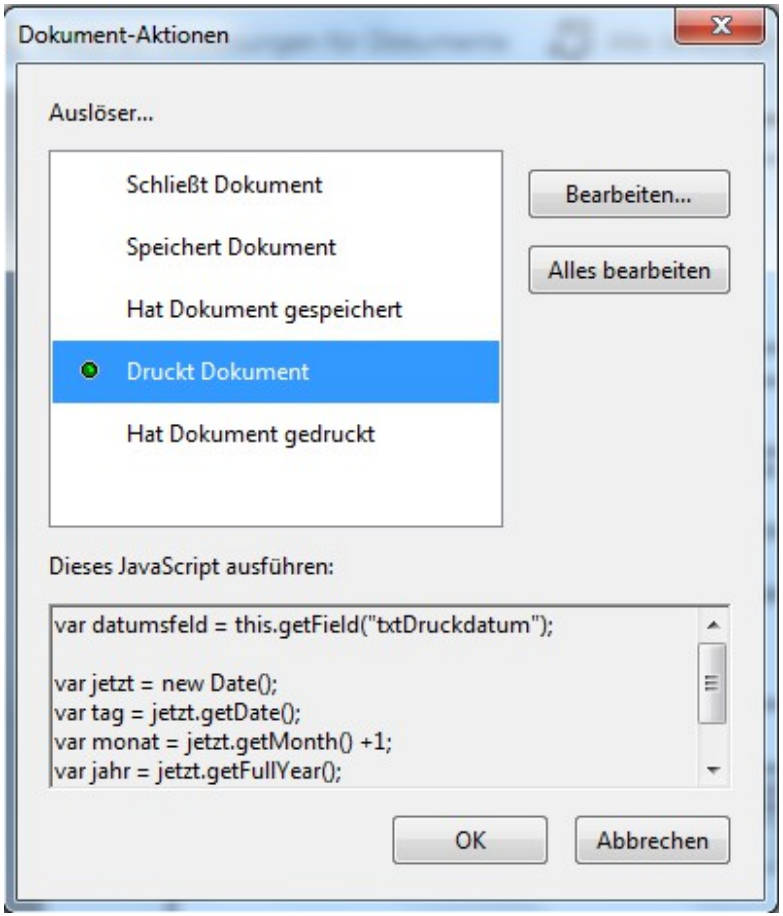


Aktionen durch

- *Werkzeuge – JavaScript*.
- Klick auf den Eintrag *Dokumentaktionen* im Werkzeugleiste.
- Im Listenfeld [Auslöser] wird das Ereignis, auf das reagiert werden soll, ausgewählt.
- Durch einen Klick auf die Schaltfläche *Bearbeiten* wird der JavaScript-Editor geöffnet.
- *OK* schließt den Dialog [Dokument-Aktionen].

Dialog „Dokument-Aktionen“

- Auslöser, an denen ein Script angehängt ist, werden mit einem grünen Kreis gekennzeichnet.
- Im unteren Textfeld wird der Code zu dem gewählten Auslöser angezeigt. Der Code kann nicht direkt im Textfeld bearbeitet werden.



Beispiel-Code

```
var datumsfeld = this.getField("txtDruckdatum");

var jetzt = new Date();
var tag = jetzt.getDate();
var monat = jetzt.getMonth() +1;
var jahr = jetzt.getFullYear();

datumsfeld.value = tag + "." + monat + "." + jahr;
```

Datum und Uhrzeit in einem Formular

- Anzeige in einem Formularfeld beim Öffnen oder Drucken eines Formulars.
- Validierung von Daten.
- Formatierung mit Hilfe der Registerkarte [Format] in dem Dialog [Eigenschaften].

… in JavaSript

```
var jetzt = new Date();
```

- Das Objekt Date speichert Datums- und Zeitangaben in Millisekunden.
- Als Basiswert für die Berechnung eines Datums in Millisekunden wird der 1. Januar 1970, 0:00 Uhr UTC genutzt.
- Siehe https://wiki.selfhtml.org/wiki/JavaScript/Objekte/Date.

Objekt „Date“ konstruieren

var	jetzt	=	new	Date	(	)	;

- Das Schlüsselwort new erzeugt ein Objekt vom Typ … mit Hilfe eines vorgefertigten Bauplans.
- In diesem Beispiel wird der Bauplan Date genutzt, um ein Objekt zu konstruieren.
- Die runden Klammern sind leer. Das Objekt nutzt das aktuelle Systemdatum und die aktuelle Systemzeit.
- Der Verweis auf das neu erstellte Objekt wird in der Variablen jetzt gespeichert.

… und formatieren

```
this.getField("txtBestelltAm").value =
                    util.printd("dd.mm.yyyy HH:MM", new Date());
```

- Das Objekt util bietet verschiedene Methoden, um Strings und Zeitwerte zu formatieren.
- Die Methode .printd() liefert einen formatieren Datums- und Zeitwert zurück.

Parameter der Methode

```
this.getField("txtBestelltAm").value =
                    util.printd("dd.mm.yyyy HH:MM", new Date());
```

- Als erster Parameter wird der Methode ein Formatierungsstring übergeben. Der String beginnt und endet mit den Anführungszeichen.
- Der zweite Parameter enthält die zu formatierende Zeitangabe. In diesem Beispiel das aktuelle Systemdatum und die aktuelle Systemzeit.

Formatierungsstring

dd	Zweistelliger Monatstag
mm	Zweistellige Monatsangabe
yyyy	Vierstellige Jahresangabe
HH	24-stündige Zeitangabe, zweistellig
MM	Minutenangabe, zweistellig

Datumsangaben

```
var tag = jetzt.getDate();
var monat = jetzt.getMonth() +1;
var jahr = jetzt.getFullYear();
```

- Ein Objekt Date wird mit den, darin definierten Methoden durch den Punktoperator verbunden.
- Die Methode getDate() gibt den Monatstag zurück.
- Die Methode getMonth() gibt den Monat als Ganzzahl zurück. Der Monat Januar hat den Wert 0.
- Die Methode getFullYear() gibt das Jahr als vierstellige Zahl zurück.

Beispiel

19	=	getDate("08 19, 2016")
7	=	getMonth("08 19, 2016")
2016	=	getFullYear("08 19, 2016")

Berechnung eines Tages in Millisekunden

```
var einTag = 24 * 60 * 60 * 1000;
```

- Ein Tag hat 24 Stunden.
- Eine Stunde hat 60 Minuten.
- 1 Minute hat 60 Sekunden.
- 1 Sekunde hat 1000 Millisekunden.

Addition von x Tagen

```
var morgen =   heute.getTime() + (24 * 60 * 60 * 1000);
```

- Die Datumsangabe wird mit Hilfe der Methode getTime() in eine Zeit von x Millisekunden umgerechnet.
- Der Zeitangabe wird eine Anzahl von x Tagen addiert.

… umwandeln in eine Datumsangabe

```
var morgen =   new Date(heute.getTime() + (24 * 60 * 60 * 1000));
```

- Dem Objekt Date wird in den runden Klammern eine Tagesangabe in Millisekunden übergeben.
- Mit Hilfe dieses Startwertes wird ein neues Objekt mit Hilfe des Bauplans „Zeitangabe“ erstellt.