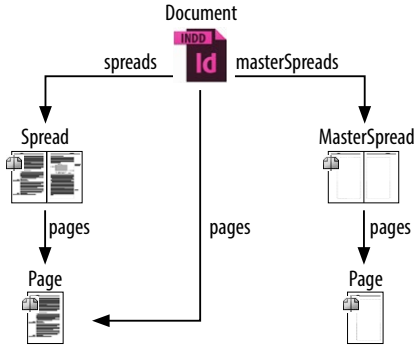
7.7 Seiten und Mustervorlagen

Wenn man mit InDesign arbeitet, ist die Unterscheidung zwischen Druckbogen, Mustervorlagen und Seiten klar, beim Skripting sollte man die Hierarchie kennen.

Abb. 62
Dokument, Seiten und Musterseiten



Die Abbildung zeigt, dass es eigentlich die klare Hierarchie Dokument, Druckbogen, Seite gibt. Das Objektmodell erlaubt jedoch den direkten Zugriff auf die Seiten über das Objekt `Document`. Diese Abkürzung wird oft verwendet, weil die Adressierung über das Objekt `Spread` unübersichtlicher ist und zu aufwändigerem Code führt. Außerdem beinhaltet das Objekt `Spread` auch die Montagefläche und Objekte, die sich darauf befinden.

Für Musterseiten gibt es kein Objekt `MasterPage`, die Druckbögen der Mustervorlagen `MasterSpread` enthalten normale Seiten. Musterseiten werden durch ihre Zugehörigkeit zu einem Musterdruckbogen gekennzeichnet. Deswegen muss man Musterseiten immer über die Eigenschaft `masterSpreads` adressieren. Dies führt leider manchmal zu Verwirrung.

Umgang mit Seiten

Im Folgenden werden die wichtigsten Eigenschaften der Klasse `Page` vorgestellt. Mit der Eigenschaft `name` kann die Seitenzahl der Seite ermittelt werden. Die Eigenschaft enthält einen String, der für die Adressierung in der Sammlung `Pages` mit der Methode `itemByName()` verwendet werden kann. Achten Sie darauf, dass es sich hier bei einer manuellen Seitennummerierung nicht unbedingt um die Position der Seite im Dokument handelt.

Position der Seite mit Dokument

Die Position im Dokument – und damit auch den Index für die Adressierung mit den eckigen Klammern – ermittelt man mit der Eigenschaft `documentOffset`.

Mit `side` kann man herausfinden, ob es sich um die rechte (`PageSideOptions.RIGHT_HAND`) oder linke Seite (`PageSideOptions.LEFT_HAND`) einer Doppelseite oder um eine Einzelseite (`PageSideOptions.SINGLE_SIDED`) handelt.

```
1  var _page = app.activeDocument.pages[0];
2  var _seite;
3  if (_page.side == PageSideOptions.LEFT_HAND) _seite = "linke";
4  else if (_page.side == PageSideOptions.RIGHT_HAND) _seite =
   "rechte";
5  else _seite = "einzelne";
6  alert("Position:" + _page.documentOffset + " , Name (Seitenzahl): "
   + _page.name + ",Typ: " + _seite + " Seite");
```

Listing 65
Eigenschaften der Seite
7-7_Seiten.jsx

Beachten Sie, dass ich in den Zeilen 3–5 die geschweiften Klammern der if- und else-Blöcke weggelassen habe. Die geschweiften Klammern würden hier den Code eher unübersichtlich machen. Wenn Sie wollen, können Sie sie natürlich hinzufügen.

Umgang mit Mustervorlagen

Auf Mustervorlagen sollten die feststehenden und/oder wiederkehrenden Objekte des Layouts platziert werden. Ähnlich wie Formate erstelle ich die benötigten Mustervorlagen nicht per Skript, sondern baue diese in InDesign. Das Dokument mit meinen Mustervorlagen verwende ich dann im Skript als Vorlage. Wenn man in den MUSTERVORLAGENOPTIONEN eindeutige Namen vergeben hat, können diese im Skript mit der Methode `itemByName()` adressiert werden.

Beachten Sie, dass sich der Name aus dem Namen und dem Präfix, so wie er auch im Bedienfeld SEITEN erscheint, zusammensetzt.

Um eine Mustervorlage auf eine Seite anzuwenden, muss die Mustervorlage der Eigenschaft `appliedMaster` zugewiesen werden.

Das folgende Beispiel setzt eine Mustervorlage mit der Präfix-Namen-Kombination `A-Mustervorlage` voraus. Das Beispieldokument *7-7_Mustervorlagen.idml* ist für die Verwendung mit den restlichen Skripten des Unterkapitels vorbereitet.

```
1  var _dok = app.activeDocument;
2  var _musterVorlage = _dok.masterSpreads.
   itemByName("A-Mustervorlage");
3  _dok.pages[0].appliedMaster = _musterVorlage;
```

Listing 66
Adressierung von Mustervorlagen
7-7_Mustervorlagen-1.jsx

Mustervorlagenobjekte mit `override()` lösen

Mustervorlagenobjekte sind auf der Seite im Layout normalerweise gesperrt, dies erkennt man an den gepunkteten Rahmenkanten. Die Idee ist, dass das Objekt nicht zur eigentlichen Seite, sondern zur Musterseite gehört. Wenn Sie z. B. eine Grafik auf der Musterseite platzieren, wird diese bei 50 Seiten nicht 50-mal platziert, sondern nur einmalig auf der Musterseite. Trotzdem kommt es vor, dass Objekte auf der Musterseite gelöst werden müssen. In InDesign klickt man dazu mit gedrückter SHIFT + BEFEHLSTASTE auf das entsprechende Objekt. Im Skripting kann dazu die Methode `override()` verwendet werden. Die Methode erwartet als Parameter die Seite, auf der das Objekt gelöst werden soll.

Im folgenden Beispiel wird das zuletzt erstellte Mustervorlagenobjekt der auf der ersten Seite des Dokuments angewendeten Mustervorlage gelöst.

Listing 67
Lösen eines Mustervorlagenobjekts
7-7_Mustervorlagen-2.jsx

```
1  var _dok = app.activeDocument;
2  var _page = app.activeDocument.pages[0];
3  var _musterVorlage = _dok.masterSpreads.
   itemByName("A-Mustervorlage");
4  _page.appliedMaster = _musterVorlage;
5  _page.masterPageItems[0].override(_page);
```

Nachdem in der Variablen `_page` die erste Seite des Dokuments gespeichert wurde, wird ihr in Zeile 3 die Mustervorlage `A-Mustervorlage` zugewiesen.

5 Vom Objekt `Page` kann man mit der Eigenschaft `masterPageItems` auf die Mustervorlagenobjekte zugreifen, die auf der Seite enthalten sind. Als Rückgabewert bekommt man einen Array; um das Objekt zu lösen, muss man den Index des Objekts kennen. Mit der Methode `override()` wird das Objekt auf der Seite, die als Parameter übergeben wird, gelöst – in diesem Fall also auf der ersten Seite, die über `_page` referenziert wurde.

Da in der Eigenschaft `masterPageItems` ein Array gespeichert ist, kann man nicht mit den Methoden der Sammlungen wie `itemByName()` auf die Mustervorlagenobjekte zugreifen. Da sich der Index innerhalb des Arrays `masterPageItems` bei der Erstellung von neuen Objekten ändert, ist der Zugriff über die Eigenschaft `masterPageItems` nur in Ausnahmefällen empfehlenswert. Besser ist es, die Objekte auf der Musterseite mit Namen zu versehen und diese dann zu lösen.

Listing 68
Lösen von Mustervorlagenobjekten
7-7_Mustervorlagen-3.jsx

```
5  var _vorlagenObjekt = _musterVorlage.pageItems.itemByName("tf");
6  _vorlagenObjekt.override(_page);
```

5 Anstatt über die Seite wird das Objekt auf der eigentlichen Musterseite adressiert. Dort muss natürlich ein Objekt mit dem Namen `tf` vorhanden sein. Wie man Objekte mit Namen versieht, wurde auf Seite 86 besprochen.

6 Das in `_vorlagenObjekt` gespeicherte Musterseitenobjekt kann jetzt mit der Methode `override()` gelöst werden. Hier wird auch deutlicher, dass als Parameter die Seite, auf der das Objekt gelöst werden soll, übergeben werden muss.

++ Musterseitenobjekte sperren

Rahmen auf Musterseiten können gesperrt werden, so dass sie nicht mehr auf der Seite, auf der die Musterseite angewendet wurde, gelöst werden können. Dazu muss man per Skript die Eigenschaft `allowOverrides` auf den Wert `false` setzen.

Primäre Textrahmen und automatischer Textumfluss

Wenn ein Textrahmen immer gelöst werden soll, beispielsweise weil ein Text platziert und automatisch umbrochen werden soll, muss auf

der Musterseite ein primärer Textrahmen eingerichtet werden. Dieser wird automatisch gelöst, wenn die Musterseite verwendet wird.

Einrichtung eines primären Textrahmens

Einen primären Textrahmen in Satzspiegelgröße können Sie bei der Erstellung des Dokuments von InDesign hinzufügen lassen. Wählen Sie dazu die Option PRIMÄRER TEXTRAHMEN im Dialog NEUES DOKUMENT an. Alternativ können Sie nachträglich einen Textrahmen umwandeln, indem Sie mit der Maus auf das Symbol in der linken oberen Ecke klicken. Das Symbol wechselt dann zu , so dass der primäre Textrahmen leicht erkennbar ist. Beachten Sie, dass der Textrahmen leer sein muss. Per Skript könnten Sie in der Eigenschaft `primaryTextFrame` der Musterseite einen Textrahmen speichern, in der Praxis ist es aber leichter, dies in der Arbeitsvorbereitung manuell zu erledigen!

Automatischer Umbruch

Für einen automatischen Umbruch muss die Voreinstellung INTELLIGENTER TEXTUMFLUSS aktiviert werden. Diese Einstellung finden Sie unter BEARBEITEN → VOREINSTELLUNGEN→ EINGABE.... Wenn der primäre Textrahmen bei der Bearbeitung einen Textüberlauf bekommt, werden automatisch Seiten hinzugefügt. Zusammen mit der Funktion `place()` aus dem vorherigen Unterkapitel können Sie einen automatischen Seitenumbruch skripten.

Listing 69
Intelligenten Textumfluss per Skript aktivieren
7-7_Textumfluss.jsx

```
1  var _dok = app.activeDocument;
2  with (_dok.textPreferences){
3    smartTextReflow = true;
4    limitToMasterTextFrames = true;
5    addPages = AddPageOptions.END_OF_STORY;
6    preserveFacingPageSpreads = true;
7    deleteEmptyPages = false;
8  }
```

2–8 Mit Hilfe eines with-Statements werden die Einstellungen für den intelligenten Textumfluss aktiviert. Das Ergebnis entspricht den Einstellungen in der folgenden Abbildung:

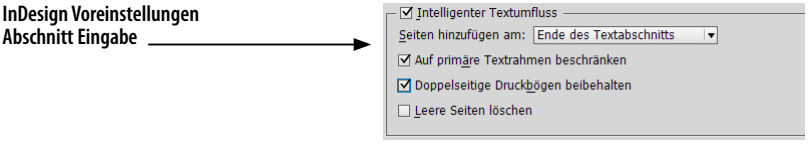


Abb. 63
Intelligenter Textumfluss

7.8 Rahmen und Seitenobjekte

Wenn man die Seiten gemeistert hat, trifft man auf die Objekte, die auf ihnen platziert sind. Dazu zählen Rechtecke, Textrahmen, Ellipsen, Linien sowie Gruppen und einige mehr – letztlich alle Objekte, die auf einer Seite platziert sein können. Alle Seitenobjekte sind von der Basisklasse `PageItem` abgeleitet.

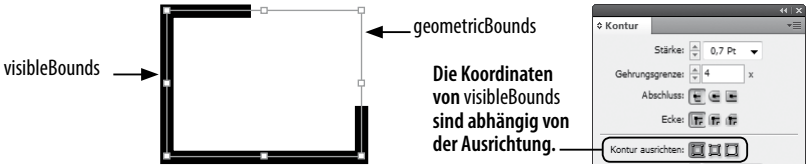
Tab. 17
Wichtige, von `PageItem` abgeleitete Objekte

InDesign-Objekt	Skripting-Objekt
Textrahmen	`TextFrame`
Rechteckrahmen	`Rectangle`
Ellipsenrahmen	`Oval`
Grafik	`Graphic`
Linie	`GraphicLine`
Gruppe	`Group`
Schaltfläche	`Button`

Der Umgang mit all diesen Objekten ist recht ähnlich; die wichtigsten Eigenschaften werden gleich vorgestellt.

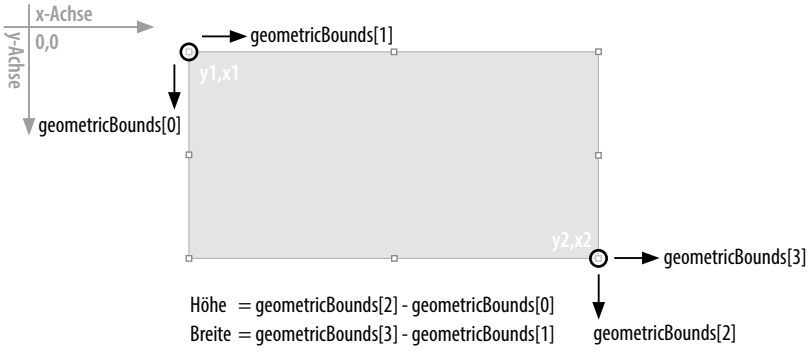
Mit der Eigenschaft `geometricBounds` bzw. `visibleBounds` wurden bereits Größen und Positionen von Seitenobjekten ausgewertet. Während Erstere die Rahmenkanten repräsentieren, enthalten Letztere die Größe des Rahmens inklusive der Rahmenkontur. Die folgende Abbildung zeigt den Zusammenhang zwischen der Konturposition und den Werten der beiden Eigenschaften.

Abb. 64
Unterschied zwischen geometricBounds und visibleBounds



Beide Eigenschaften enthalten einen Array mit vier Koordinaten. Das Problem beim Skripting ist meist, dass man die Ansicht und die Werte aus dem Bedienfeld Steuerung gewohnt ist. Hier werden die x- und y-Position des Rahmens sowie Breite und Höhe angezeigt. In der Eigenschaft `geometricBounds` bzw. `visibleBounds` finden sich nur vier Koordinaten, die außerdem noch vertauscht sind. Die ersten beiden Werte definieren die linke obere Ecke des Rahmens, die letzten beiden Werte die rechte untere Ecke des Rahmens. Es wird immer zuerst die vertikale Koordinate (y-Wert) angegeben, danach folgt die horizontale Koordinate (x-Wert).

Abb. 65
Koordinaten von Seitenobjekten



Höhe und Breite kann man sich leicht errechnen:

```
1  var _ausw = app.selection[0];
2  var _hoehe = _ausw.geometricBounds[2] - _ausw.geometricBounds[0];
3  var _breite = _ausw.geometricBounds[3] - _ausw.geometricBounds[1];
4  alert("Das Objekt hat die Höhe " + _hoehe + " und Breite " + _breite);
```

Listing 70
Höhe und Breite eines Seitenobjekts berechnen
7-8_HoeheBreite.jsx

Ein Objektformat verwenden

Rahmen und Seitenobjekte haben viele Attribute bezüglich des Aussehens, die Details finden Sie im Objektmodell. Oft verwendet werden `strokeWeight` für die Konturenstärke, `strokeColor` für die Farbe der Kontur, `fillColor` für die Hintergrundfarbe. Alle diese Eigenschaften können auch in Objektformaten hinterlegt werden.

Bei der Einrichtung von Objektformaten gibt es die Möglichkeit, Bereiche mit bestimmten Attributen wie Fläche, Kontur oder Transparenz zu deaktivieren. Bei der Anwendung des Formats auf ein Seitenobjekt werden die deaktivierten Attribute unverändert beibehalten, entsprechend wird auch keine Abweichung zum Format im Bedienfeld angezeigt. Die nicht vom Format definierten Attribute können mit der Schaltfläche am unteren Rand des Bedienfelds Objektformate gelöscht werden. Die Attribute werden dann auf die Werte des Objektformats [Ohne] gesetzt.

```
var _of = app.activeDocument.objectStyles.itemByName("NameOF");
app.selection[0].applyObjectStyle(_of, false, false);
```

Objektformat zuweisen

Ein Objektformat weisen Sie am besten mit der Methode `applyObjectStyle()` zu. Ihr wird als erster Parameter das Objektformat übergeben. Mit dem zweiten Parameter kann gesteuert werden, ob Formatabweichungen des Objekts gelöscht werden. Mit dem Wert `true` werden alle abweichenden Formatattribute entfernt, bei `false` bleiben diese erhalten. Mit dem dritten Parameter können Sie die nicht vom Format definierten Attribute löschen bzw. erhalten. Mit `true` werden sie gelöscht, bei `false` nicht. Der zweite und dritte Parameter sind optional, der Standardwert für den zweiten ist `true`, für den dritten `false`.

```
app.selection[0].appliedObjectStyle = app.activeDocument.objectStyles.
   itemByName("NameOF");
```

Alternativ könne Sie auch mit der Eigenschaft `appliedObjectStyle` das Objektformat festlegen, dann werden die Abweichungen zum Format gelöscht, aber die nicht vom Format definierten Attribute beibehalten.

Weitere wichtige Eigenschaften

Skalierung und Drehung

Die Drehung und Skalierung von Seitenobjekten sollen oft unabhängig von einem Objektformat eingerichtet werden. Für die Skalierung kommen die Eigenschaften `absoluteHorizontalScale` und `absoluteVerticalScale` zum Einsatz. Beiden wird der gewünschte Pro-

zentwert als Zahl übergeben. Bei der Skalierung stehen zwei Eigenschaften zur Verfügung, denen beliebige Winkel von –360° bis 360° als Zahl übergeben werden können. Mit `absoluteRotationAngle` kann der Drehwinkel relativ zum Elternobjekt angegeben werden. Mit der Eigenschaft `rotationAngle` wird der eigentliche Drehwinkel des Objekts angegeben – der Wert, den Sie auch in der Benutzeroberfläche sehen. Zur Verdeutlichung kann man sich eine in einem Rechteckrahmen platzierte Grafik vorstellen: Wenn diese um 10° gedreht wird, enthält die Eigenschaft `absoluteRotationAngle` des Rechtecks die Zahl 10, die der Grafik 0. Der Wert der Eigenschaft `rotationAngle` ist in beiden Fällen 10.

Zugriff auf alle Seitenobjekte

Alle Seitenobjekte eines Dokuments, einer Seite oder einer Ebene erhalten Sie über die Eigenschaft `allPageItems`, die einen Array mit den Seitenobjekten enthält. Da in Seitenobjekten weitere Seitenobjekte enthalten sein können, z. B. in einer Gruppe, hat auch die Klasse `PageItem` die Eigenschaft `allPageItems`. Zusätzlich haben diese Objekte auch die Eigenschaft `pageItems`, in der sich die Sammlung der Seitenobjekte befindet. Die Sammlung unterscheidet sich vom Array darin, dass in der Sammlung die Hierarchieebenen von gruppierten, verankerten oder ineinanderkopierten Seitenobjekten abgebildet sind. Der Inhalt der Sammlung entspricht einer Hierarchieebene im Bedienfeld Ebenen. Der Vorteil der Sammlung ist, dass mit der Methode `itemByName()` ein benanntes Objekt adressiert werden kann.

Eltern von Seitenobjekten

Wenn Sie den Rahmen nicht durch die klassische Hierarchie über die Seite adressiert haben, kommen Sie mit der Eigenschaft `parentPage` auf die Seite, auf der sich der Rahmen befindet. Wenn sich der Rahmen über zwei Seiten erstreckt, wird die Seite mit der größeren Fläche zurückgeliefert. Wenn sich der Rahmen auf der Montagefläche befindet, liefert die Eigenschaft `null`. Die Eigenschaft `parent` führt vom Rahmen zum Druckbogen. Ein klassischer Anwendungsfall ist die automatische Platzierung von Bildern, deren Dateinamen im Manuskript enthalten sind. Hier sucht man die entsprechenden Textstellen und ermittelt dann über `parentPage` die Seite, auf der das Bild platziert werden soll (→ Unterkapitel 12.4). Für Preflight- oder Analyse-Skripte ist es ebenfalls hilfreich, die Seite, auf der ein Fehler enthalten ist, zu kennen.

Konturenführung

Wenn man die Konturenführung nicht über ein Objektformat festlegen will, kann man dies auch direkt per Skript erledigen. Alle Seitenobjekte haben die Eigenschaft `textWrapPreferences`, in der die Einstellungen für die Konturenführung hinterlegt sind.

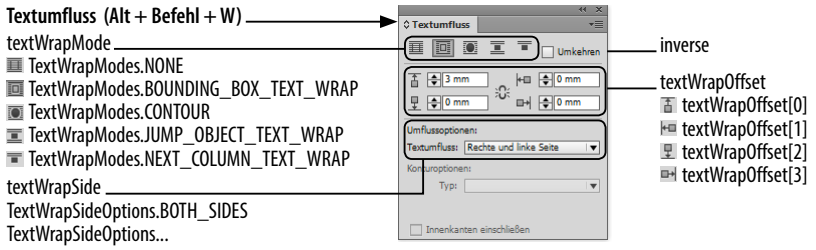


Abb. 66
Zusammenhang Konturenführung-Bedienfeld und Klasse textWrapPreferences

Das nächste Skript zeigt, wie man diese Einstellungen per Skript setzen kann. Dazu können Sie das Dokument *7-8_Konturenfuehrung.idml* verwenden. Vor der Ausführung muss der Bildrahmen ausgewählt werden.

Listing 71
7-8_Konturenfuehrung.jsx

```
var _auswahl = app.selection[0];
with (_auswahl.textWrapPreferences) {
  textWrapMode = TextWrapModes.BOUNDING_BOX_TEXT_WRAP;
  textWrapOffset = [3,3,0,3];
  textWrapSide = TextWrapSideOptions.BOTH_SIDES;
}
```

4 Die Angabe der Abstände mit `textWrapOffset` muss in Abhängigkeit des `textWrapMode` unterschiedlich vorgenommen werden. Im Fall `TextWrapModes.BOUNDING_BOX_TEXT_WRAP` wird ein Array mit vier Werten übergeben `[oben, links, unten, rechts]`. Für den Wert `TextWrapModes.JUMP_OBJECT_TEXT_WRAP` müssen zwei Werte im Format `[oben, unten]` angegeben werden. Im Fall `TextWrapModes.NEXT_COLUMN_TEXT_WRAP` und `TextWrapModes.CONTOUR` wird ein einzelner Wert ohne Array übergeben.

QR-Codes

Seit InDesign CC können in Seitenobjekten QR-Codes eingesetzt werden, üblicherweise werden sie in einem Rahmen platziert. Das folgende Skript erstellt einen QR-Code, der zur Webseite des Buches führt. Vor der Ausführung muss ein Rahmen ausgewählt werden.

Listing 72
7-8_QRCode.jsx

```
var _auswahl = app.selection[0];
var _url = "http://www.indesignjs.de";
var _swatch = app.activeDocument.swatches[3];
_auswahl.createHyperlinkQRCode( _url, _swatch);
```

4 Die Methode `createHyperlinkQRCode()` übernimmt zwei Parameter. Der erste enthält die URL zur Webseite. Mit dem zweiten, optionalen Parameter kann die Farbe des QR-Codes gesteuert werden. Beide Parameter wurden der Übersichtlichkeit halber zuvor in Variablen gespeichert, könnten aber auch direkt übergeben werden. Im Unterkapitel 14.3 finden Sie ein weiteres Beispiel im Praxiseinsatz.

Abb. 67
Der generierte QR-Code

Die folgenden QR-Code-Typen können erstellt werden. Die Parameter mit den Werten des QR-Codes werden als String übergeben. Die Angabe eines Farbfelds im letzten Parameter ist optional; wenn Sie keine Farbe übergeben, wird Schwarz verwendet.

Tab. 18 QR-Codes ab InDesign CC

Typ	Funktion und Parameter
E-Mail	`createEmailQRCode (emailAddress, subject, body, [qrCodeSwatch])`
Webadresse	`createHyperlinkQRCode (urlLink, [qrCodeSwatch])`
Text	`createPlainTextQRCode (plainText, [qrCodeSwatch])`
SMS	`createTextMsgQRCode(cellNumber, textMessage, [qrCodeSwatch])`
VCard	`createVCardQRCode (firstName, lastName, jobTitle, cell Phone, phone, email, organisation, streetAddress, city, adrState, country, postalCode, website, [qrCodeSwatch])`

Die Methoden `create...QRCode()` generieren jeweils ein Objekt vom Typ `EPS` in einem Rahmen. Die Eigenschaften des QR-Codes können nicht nachträglich angepasst werden. Wenn das nötig ist, kann man einfach einen neuen QR-Code erstellen.

7.9 Textrahmen

Textrahmen sind von der Klasse `PageItem` abgeleitet, weisen aber ein paar Besonderheiten auf. Aus InDesign kennt man die Möglichkeiten, Textrahmen zu verketten. Dazu besitzt die Klasse `TextFrame` die Eigenschaften `nextTextFrame` und `previousTextFrame`. Beiden kann man als Wert den Textrahmen, der verkettet werden soll, zuweisen. Für die folgenden Skripte kann das Dokument *7-9_Textrahmen.idml* verwendet werden.

Listing 73 Verkettung von zwei Textrahmen *7-9_Verketten.jsx*

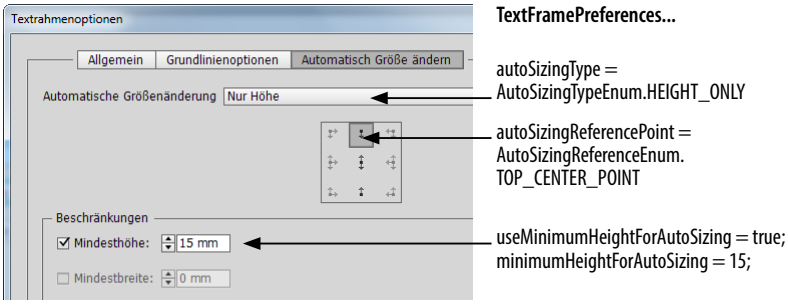
```
1  var _seite1 = app.activeDocument.pages[0];
2  var _seite2 = app.activeDocument.pages[1];
3  _seite1.textFrames[0].nextTextFrame = _seite2.textFrames[0];
```

3 Der Eigenschaft `nextTextFrame` des Textrahmens auf der ersten Seite wird der Textrahmen der zweiten Seite zugewiesen. Die beiden Rahmen sind nun verkettet.

Textrahmengröße automatisch festlegen

Ein interessante Möglichkeit von Textrahmen ist die Funktion Automatisch Grösse ändern, mit der die automatische Größenänderung gesteuert werden kann. Die Einstellungen können in den Textrahmenoptionen vorgenommen werden.

Abb. 68 Textrahmenoptionen für Automatisch Größe ändern



```
1  var textFrame = app.selection[0];
2  with (textFrame.textFramePreferences) {
3    autoSizingType = AutoSizingTypeEnum.HEIGHT_ONLY;
4    autoSizingReferencePoint =
     AutoSizingReferenceEnum.TOP_CENTER_POINT;
5    useMinimumHeightForAutoSizing = true;
6    minimumHeightForAutoSizing = 15;
7  }
```

Listing 74
Automatisch Größe ändern per Skript
7-9_Auto_Textrahmen.jsx

2–7 Wie üblich ist die grafische Benutzeroberfläche in einer Preferences-Klasse abgebildet, die einzelnen Eigenschaften befinden sich hier in `textFramePreferences`. Mit der Eigenschaft `autoSizingType` wird festgelegt, wie der Textrahmen angepasst werden soll. Entsprechend sind dann auch nicht immer alle Eigenschaften möglich. Im Falle der automatischen Höhenanpassung `AutoSizingTypeEnum.HEIGHT_ONLY` kann bspw. der `autoSizingReferencePoint` nur auf `AutoSizingReferenceEnum.TOP_CENTER_POINT`, `CENTER_POINT` bzw. `BOTTOM_CENTER_POINT` gesetzt werden.

Die Einstellungen können auch in einem Objektformat hinterlegt werden, was das oben gezeigte Skript überflüssig macht!

Komplexe Rahmen funktionieren nicht.

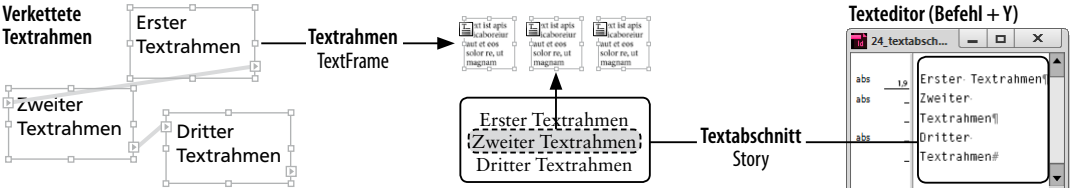
Wenn Sie die automatische Texteinpassung verwenden, sollten Sie auf die Funktion Spalten ausgleichen in Kombination mit der Absatzformatoption Spaltenspanne verzichten. Die Bearbeitung kann unangenehm langsam werden und InDesign CS6 sogar abstürzen.

Textrahmen vs. Text

Text und Textabschnitt

Der Textinhalt von einem bzw. der Textfluss von mehreren verketteten Textrahmen wird als *Textabschnitt* bezeichnet. Die zugehörige Klasse im Objektmodell heißt `Story`. Die Sammlung `Stories`, in der sich alle Textabschnitte des Dokuments befinden, gehört zum Dokument. Ein Textabschnitt enthält wie alle Textobjekte nicht nur den Inhaltstext als String, sondern alle Formatinformationen.

Abb. 69
Textabschnitte vs. Textrahmen



Ein Textabschnitt kann sich über mehrere Textrahmen (und Textpfade) erstrecken. Diese sind in einem Array gesammelt, den man über die Eigenschaft `textContainers` ansprechen kann. Verwechseln Sie `textContainers` nicht mit der ebenfalls vorhandenen Eigenschaft `textFrames`, hier sind die verankerten Textrahmen des Textabschnitts gespeichert.

Vom Textrahmen zum Textabschnitt

Wie aber kommt man nun vom Textrahmen zum Textabschnitt – die Eigenschaft `parent` führt ja zum Druckbogen? Für die Abbildung dieser Beziehung haben Textrahmen die Eigenschaft `parentStory`.

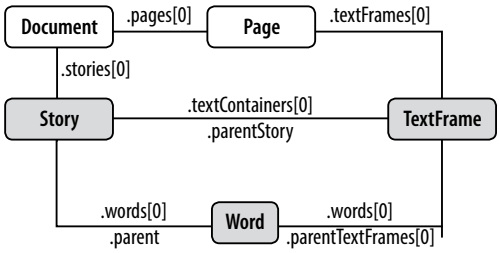
Vom Textobjekt zum Textrahmen

Die Textabschnitte untergliedern sich weiter in Absätze, Wörter und Zeichen. Diese Objekte haben als Elternobjekt den Textabschnitt, sind aber natürlich auch in Textrahmen platziert. Von Textobjekten, wie z. B. `paragraph`, kann man mit der Eigenschaft `parentTextFrames` zu dem oder den Textrahmen, die das Objekt enthalten, navigieren. Achten Sie darauf, dass hier immer ein Array enthalten ist, weil z. B. ein Absatz auf mehrere Textrahmen verteilt sein kann. Im Normalfall adressiert man den Textrahmen deswegen mit:

```
app.activeDocument.stories[0].words[2].parentTextFrames[0];
```

Da alle Textobjekte von der gleichen Klasse abgeleitet sind, gilt dies sogar für Zeichen und Einfügemarken, obwohl diese nur in einem Textrahmen stehen können.

Abb. 70 Klassenhierarchie von Textrahmen, -abschnitten und -objekten



Der folgende Code zeigt die Navigation zum Textrahmen, in dem das letzte Wort des Textabschnitts steht. Das Beispiel ist nicht praxisrelevant, zeigt aber die Zusammenhänge innerhalb der Hierarchie auf.

Listing 75 Navigation zum letzten Textrahmen der Story *7-9_NavigationText.jsx*

```
1  var _textRahmenSeite1 = app.activeDocument.textFrames[0];
2  var _textAbschnitt = _textRahmenSeite1.parentStory;
3  var _letztesWort = _textAbschnitt.words[-1];
4  var _letzterTextRahmen = _letztesWort.parentTextFrames[0];
```

Den letzten Textrahmen eines Textabschnitts adressieren

In der Praxis sollte man den letzten von mehreren verketteten Textrahmen über die Eigenschaft `endTextFrame` adressieren:

```
var _letzterTextRahmen = app.activeDocument.textFrames[0].endTextFrame;
```

Auf die Textobjekte Absatz, Wort, Zeichen kann man außerdem auch direkt vom Textrahmen zugreifen, was das Programmieren zwar einfacher, aber auch etwas unübersichtlicher macht.

Wenn man den Inhalt eines Textrahmens, der mit anderen Rahmen verknüpft ist, ermitteln will, kann die Eigenschaft `texts` verwendet werden. Die Klasse `Text` ist gleichzeitig die Basisklasse für alle Textobjekte.

Übersatz

Um wiederum herauszufinden, ob sich ein Text, der z. B. über die Suche lokalisiert wurde, im Übersatz befindet, prüft man die Eigenschaft `parentTextFrames`. Wenn der enthaltene Array keine Elemente hat, ist der Text nicht im Layout sichtbar.

```
var _letztesZeichen = app.activeDocument.stories[0].characters[-1];
if (_letztesZeichen.parentTextFrames.length == 0) { // Übersatz
```