

# InDesign Satzautomation

Übersicht | JavaScript | InDesign Objektmodell | InDesign Programmierung | InDesign XML

---

## Das Objektmodell von InDesign

Objektorientiertes Klassenmodell, alle Objekte sind von der Basisklasse „Application“ abgeleitet

Vereinfacht: Alle Menu-Einträge/Auswahlen haben eine Entsprechung (und noch mehr ... )

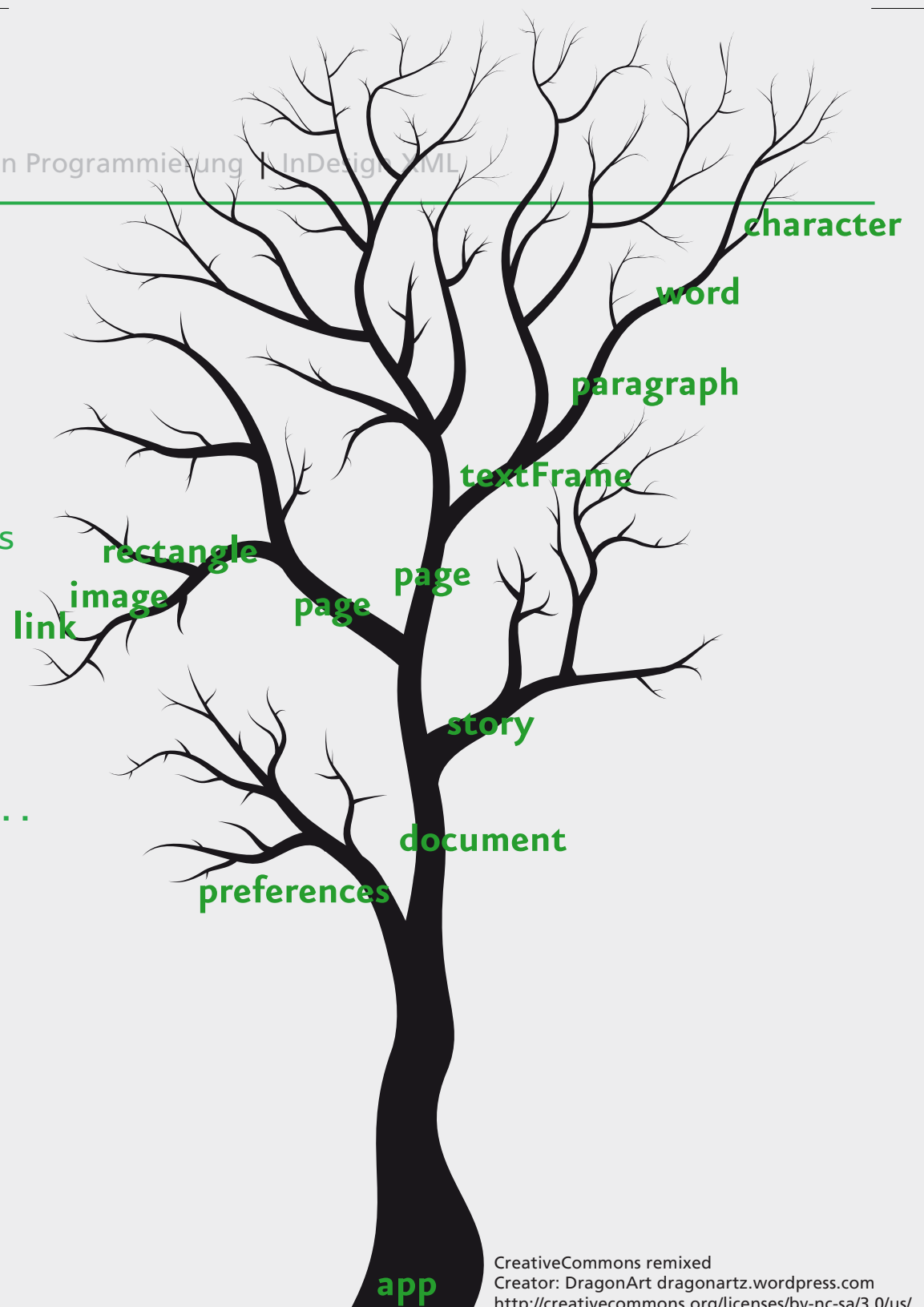
- Objektmodell als **Baum**: Ein Stamm der immer weiter bis zu den Blättern verzweigt ist. Baumstruktur wird oft verwendet: **Filesystem** (Dateien, Verzeichnisse), **XML**
- Das Objektmodell kann in der Hilfe und im **Datenbrowser** des ESTK erkundet werden.
- Alternativ gibt es ein in HTML umgesetztes Objektmodell von Theunis de Jong  
Download: <http://www.jongware.com/idjshelp.html>  
Ebenfalls Versionen des InDesign JavaScript Reference Guide in CHM (Windows Help)
- Hierarchische Struktur
  - Eltern (parents)
  - Kinder (children)
- Die Eigenschaft `parent` führt immer zum **Elternelement**
- Die Kinder sind etwas schwieriger zu finden, **Syntax mit dem Punkt** `Eltern.Kind`
- Neben „Hinauf-“ und „Heruntergehen“ in der Hierarchie kann auch kombiniert werden.  
`app.selection[0].paragraphs[0].words[1]`

# InDesign Satzautomation

Übersicht | JavaScript | InDesign Objektmodell | InDesign Programmierung | InDesign XML

## Objektmodell als Baum

Stamm	app
Dicke Äste	documents aber auch: books, documentPreferences
Dokumentäste	textFrames, rectangles, stories, links, fonts, ...
Blätter	geometricBounds, paragraphs, characters, ...



# InDesign Satzautomation

Übersicht | JavaScript | InDesign Objektmodell | InDesign Programmierung | InDesign XML

---

## Hangeln in Bäumen

Das folgende Bild zeigt verschiedene Möglichkeiten, an das erste Zeichen im ersten Textrahmen in einem neuen Dokument zu gelangen.

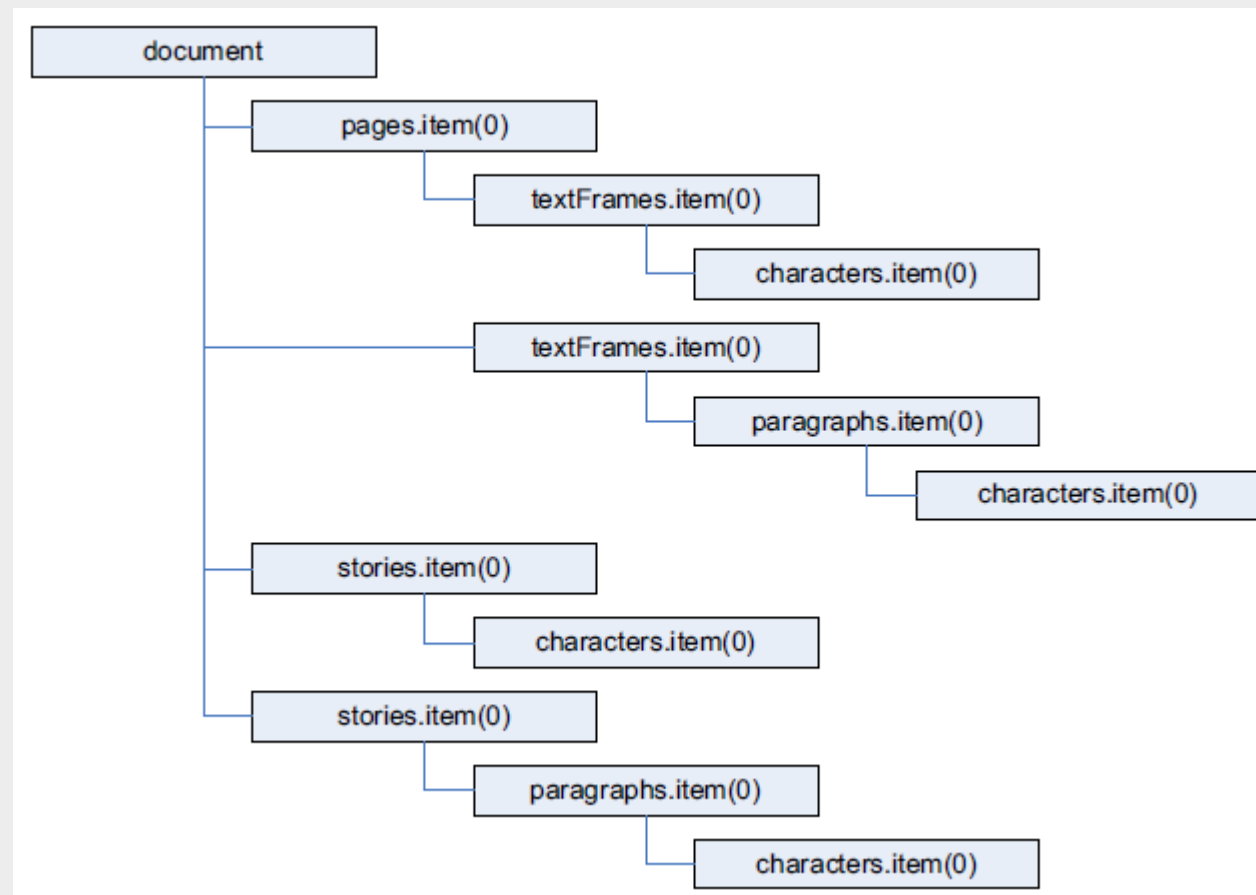


Bild entnommen aus Adobe Scripting Guide

# InDesign Satzautomation

## Übungsaufgabe

---

### Navigation im Baum

- Erstellen Sie ein neues Skript.
- ! Das Programm **InDesign** kann über das Root-Element `app` angesprochen werden.  
Was liefert die Eigenschaft `name`?  
Was liefert die Eigenschaft `activeDocument`?
- ! Dokumente haben ebenfalls die Eigenschaft `name`. Was liefert diese?  
Die Seiten eines Dokuments können über die Sammlung `pages` angesprochen werden.  
Wie kann man zur ersten Seite des Dokuments gelangen?  
Wieviele Seiten hat ein Dokument?
- ! Eine Seite kann verschiedene `pageItems` enthalten.  
Was haben all diese `pageItems` gemeinsam?
- ! Navigieren Sie zum zweiten Textrahmen auf der dritten Seite im aktuellen Dokument. Erstellen Sie dazu vorher ein entsprechendes Dokument per Hand.

# InDesign Satzautomation

## Übungsaufgabe

## Objektmodell

- ! Vergleichen Sie den Objektmodell Viewer vom ESTK und die HTML Version.
- ! Suchen Sie bestimmte Objekte, verwenden Sie Copy & Paste um Objekte oder Eigenschaften in Ihre Skripte zu übernehmen!

**ExtendScript Toolkit CS4**

Suchen | Alle

**Brower**  
Adobe InDesign CS4 (4.0) Object Model

**Klassen**

- TextFrames
- TextFramePreference
- TextFrameContents
- TextFrame
- TextExportPreference
- TextExportCharacterSet
- TextEditingPreference

**Typen**

- Instance

**Eigenschaften und Methoden**

- verticalScale: number
- visibleBounds: Measurement Unit (Number)
- words: Words
- addPath (with): PageItem
- applyObjectStyle (using, clearingOverrides, ...)
- bringForward ()
- bringToFront ()
- checkIn (): bool

**TextFrame.verticalScale**  
Data Type: [number](#)  
Adobe InDesign CS4 (4.0) Object Model  
The vertical scaling applied to the text as a percentage of its current size. (Range: 1-1000)

**TextFrame.words** (Read Only)  
Data Type: [Words](#)  
Adobe InDesign CS4 (4.0) Object Model  
A collection of words.

**TextFrame**  
Adobe InDesign CS4 (4.0) Object Model  
A text frame. Base Class: PageItem

**Adobe InDesign CS4 (6.0) Object Model JS: TextFrame - Mozilla Firefox**

Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe

file:///C:/Dokumente und Einstellungen/hp/Eigene Dateien/techref/OC

Jongware object m... ABP

**Class TextFrame**

A text frame. Base Class: [PageItem](#)

**QuickLinks** [addPath](#), [applyObjectStyle](#), [autoTag](#), [bringForward](#), [bringToFront](#), [changeGlyph](#), [changeGrep](#), [changeObject](#), [checkIn](#), [checkOut](#), [clearObjectStyleOverrides](#), [clearTransformations](#), [convertShape](#), [createOutlines](#), [detach](#), [excludeOverlapPath](#), [exportFile](#), [extractLabel](#), [findGlyph](#), [findGrep](#), [findObject](#), [findText](#), [fit](#), [fliptem](#), [getElement](#), [insertLabel](#), [intersectPath](#), [makeCompoundPath](#), [markup](#), [minusBack](#), [move](#), [override](#), [place](#), [placeXML](#), [recomp](#), [redefineScaling](#), [reframe](#), [releaseCompoundPath](#), [remove](#), [removeOverride](#), [resize](#), [resolve](#), [revert](#), [select](#), [sendToBack](#), [store](#), [subtractPath](#), [toSource](#), [toSpecifier](#), [transform](#), [transformAgain](#), [transformAgainIndividually](#), [transformSequenceAgain](#), [transformSequenceAgainIndividually](#), [transformValuesOf](#)

**Hierarchy**

[Spread](#) | [MasterSpread](#) | [PageItem](#) | [Oval](#) | [Rectangle](#) | [Polygon](#) | [GraphicLine](#) | [Group](#) | [State](#) | [Page](#) | [Ch...](#)

[PageItem](#)

**TextFrame**

[AnchoredObjectSetting](#) | [BaselineFrameGridOption](#) | [Button](#) | [Character](#) | [ContentTransparencySetting](#) | [FillTransparencySetting](#) | [Footnote](#) | [FormField](#) | [HiddenText](#) | [InsertionPoint](#) | [Line](#) | [Note](#) | [PageItem](#) | [Paragraph](#) | [Path](#) | [StrokeTransparencySetting](#) | [Table](#) | [Text](#) | [TextColumn](#) | [TextFramePreference](#) | [TextPath](#) | [TextStyle](#) | [TextVariableInstance](#) | [TextWrapPreference](#) | [TransparencySetting](#) | [Word](#)

**Properties**

## Sammlung von Objekten

Sammlungen (Datentyp Collection) sind sehr ähnlich zu **Arrays**!

- Um mehrere **Kinderelemente** abzubilden gibt es Sammlungen.
- **Sammlungen** sind eine Datenstruktur, in der alle Kinder versammelt sind.
  - z. B. `app.selection[0].paragraphs[0].words` enthält alle Worte eines Absatzes.
    - Anhand eines Indizes kann weiter eingegrenzt werden
    - `app.selection[0].paragraphs[0].words[-1]` gibt das letzte Wort zurück.
- Der **Umfang** von Sammlungen kann mit `length` ermittelt werden.
- Sammlungsobjekte haben die Funktion `add()` mit der ein neues Objekt erstellt werden kann:  
`...textFrames.add(); //liefert einen neuen TextFrame zurück`

## Eigenschaften von Objekten

- Objekte in InDesign haben eine oder mehrere **Eigenschaften**, sie beschreiben den Zustand z. B. eine zugewiesene Schriftart, Schriftschnitt oder Schriftgrad; aber auch der oben erwähnte Umfang `length` ist eine Eigenschaft.
- Einige Eigenschaften können nur ausgelesen (**read only**) werden, andere können zusätzlich auch neu beschrieben (**read/write**) werden.

## Objektmethoden

- Im Gegensatz zu Eigenschaften erledigen **Methoden** bzw. **Funktionen** eine Aufgabe und liefern im Normalfall ein Ergebnis.
- Viele Objekte haben z. B. die Funktion `add()` die ein neues Objekt erstellt.

# InDesign Satzautomation

Übersicht | JavaScript | InDesign Objektmodell | InDesign Programmierung | InDesign XML

## Rahmen

InDesign ist ein **rahmenbasiertes** DTP Programm. Sowohl Texte als auch Bilder werden in Rahmenobjekten auf Seiten platziert. Wir haben in den Übungsskripten bereits Textrahmen erstellt.

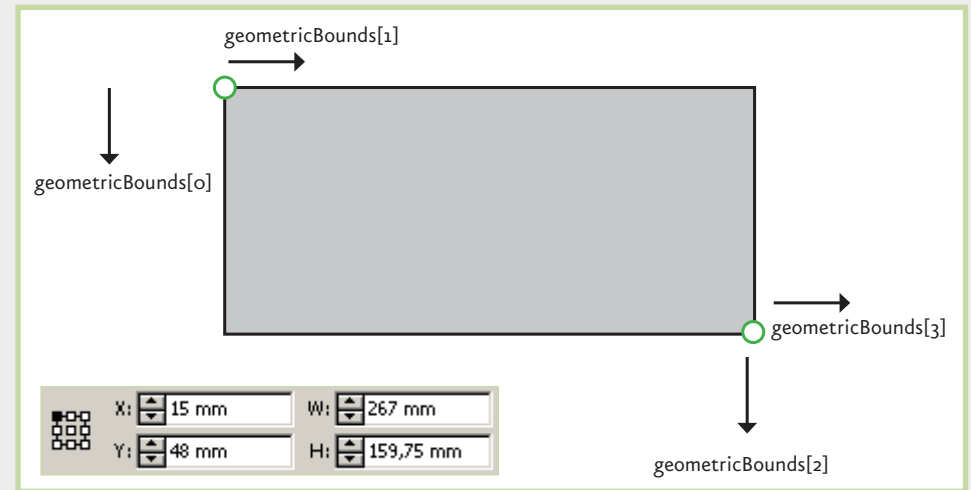
Die Dimensionen/Größen werden im Skripting nicht über Höhe und Breite angegeben sondern immer mit Hilfe von **Koordinaten**.

Die Eigenschaft `geometricBounds` ist ein Array mit den vier Koordinaten `y1,x1,y2,x2`

Daraus leiten sich auch Höhe und Breite ab:

`_höhe = geometricBounds[2] - geometricBounds[0]`

`_breite = geometricBounds[3] - geometricBounds[1]`





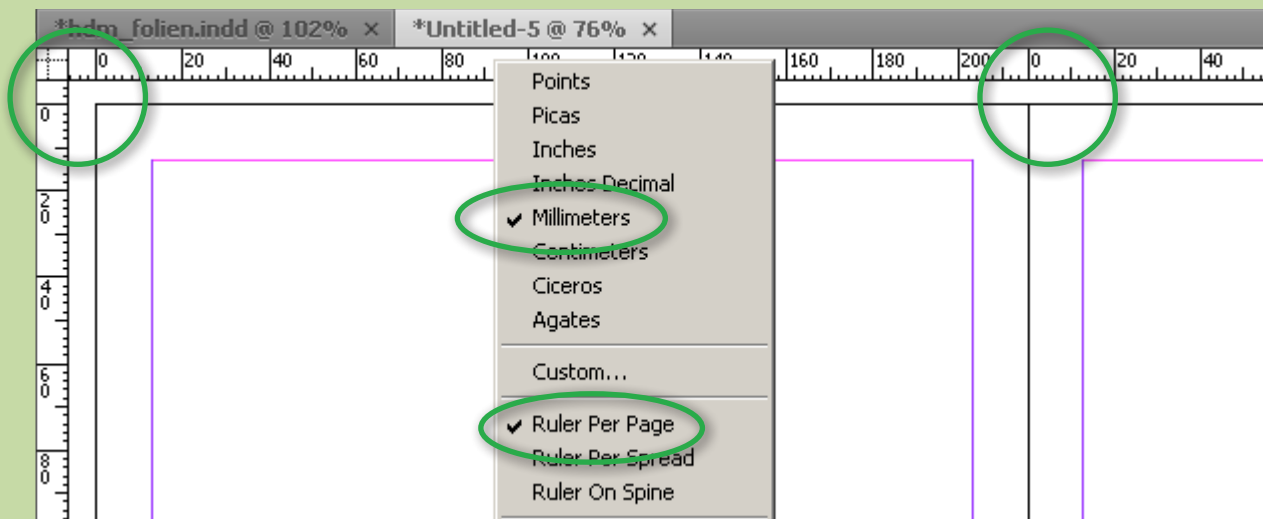
# InDesign Satzautomation

## Übungsaufgabe

### Koordination

- Die Dateien befinden sich im Ordner `02_jsx_termin_02`
- Öffnen Sie die Datei `koordinaten.jsx` im Extended Script Toolkit

! Die Koordinaten aller Rahmenobjekte orientieren sich an den **Linealen** (!) des Dokuments.



Die Einstellungen können in den `viewPreferences` des Dokuments per Skript eingestellt werden.

! Prüfen Sie die Einstellungen im Skript.

Für automatisierte Abläufe sollten diese Einstellungen immer geprüft werden.

! Erstellen Sie einen Textrahmen an der Position  $x = 15 \text{ mm}$ ;  $y = 25 \text{ mm}$  mit der Höhe  $75 \text{ mm}$  und der Breite  $60 \text{ mm}$ .

! **Bezugspunkt** für `geometricBounds`:

```
app.layoutWindows[0].transformReferencePoint = AnchorPoint.TOP_LEFT_ANCHOR
```

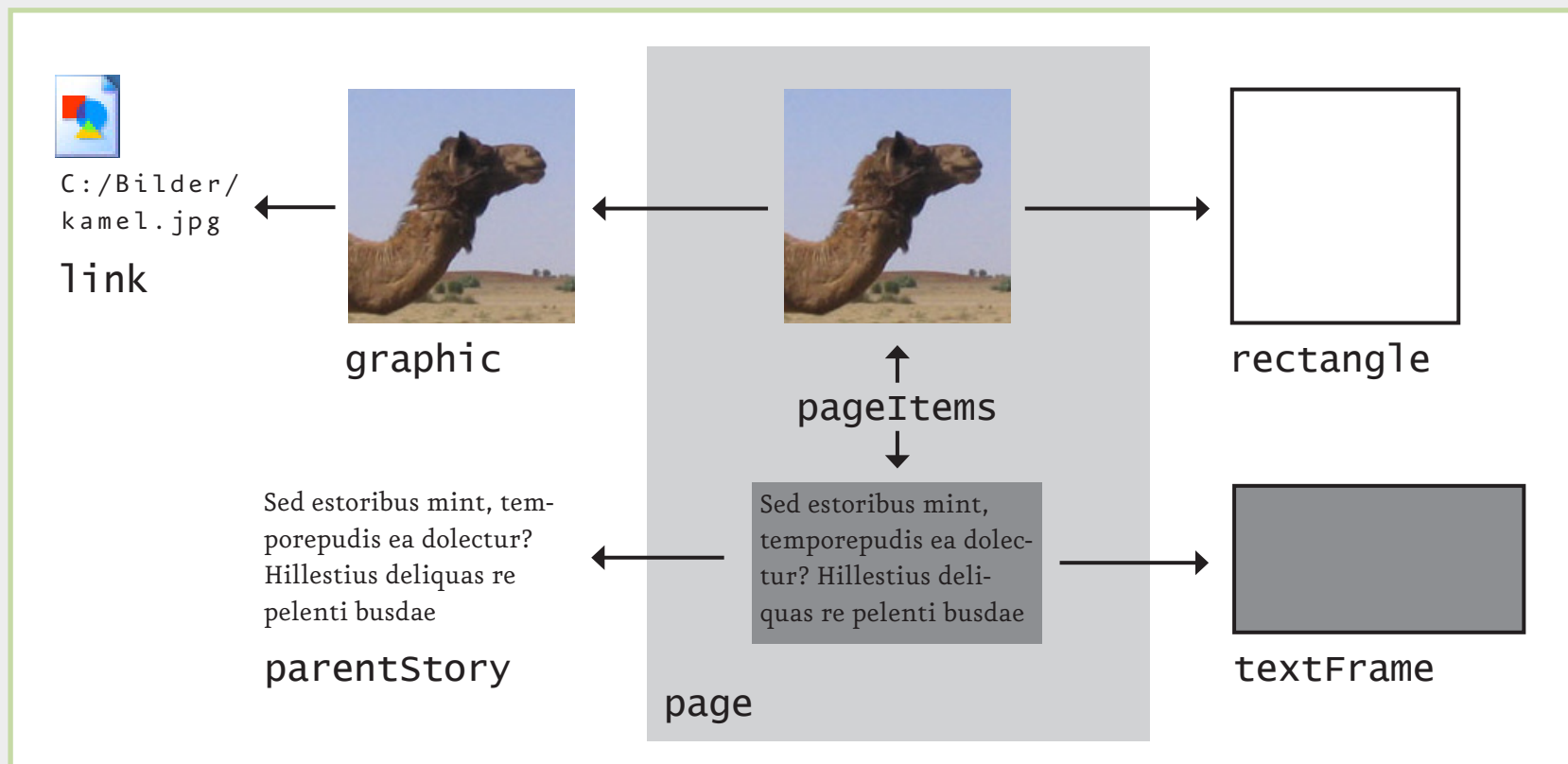
oder einfach im Template das für die Automatisierung verwendet wird umstellen.

# InDesign Satzautomation

Übersicht | JavaScript | InDesign Objektmodell | InDesign Programmierung | InDesign XML

## Wichtige Rahmen und Eigenschaften

- textFrame (Textrahmen)
  - geometricBounds
  - contents
  - parentStory
  - place()
- rectangle (Rechteck meist mit Bildrahmen)
  - graphics
  - ...



### Rahmen bauen

- Erstellen Sie ein neues Skript.
- Die verwendete Textdatei und das Bild liegen im Ordner `02_material`
- ! Erstellen Sie ein neues Dokument
- ! Erstellen sie einen **Textrahmen** (`textFrame`) und ein **Rechteck** (`rectangle`)
- ! Positionieren/verschieben Sie beide Objekte auf der Seite und weisen Sie beiden eine **Breite** 100 mm und **Höhe** 50 mm zu
- ! Laden Sie Text und Bild aus dem Ordner `01_material` in die jeweiligen Objekte. Sie benötigen dazu die Funktion `place()` des Rahmenobjekts. Der Funktion `place(fileName)` muss eine Datei übergeben werden, Erstellen Sie dazu eine Variable mit der **Dateireferenz**. Der folgende Code öffnet einen „Datei öffnen Dialog“ und gibt die Dateireferenz der ausgewählten Datei zurück:  
`File.openDialog()`
- ! Machen Sie sich mit den Eigenschaften `parentTextFrames` und `parentStory` vertraut.
- ! Warum muss `parentTextFrames` über den Index adressiert werden?
- ! Erstellen Sie einen zweiten Textrahmen und verknüpfen Sie diesen mit dem ersten Textrahmen.