Anhang

A.1 Schnellreferenz

Reservierte Wörter

Folgende Wörter sind in JavaScript als Schlüsselwörter vorgesehen – sie dürfen nicht als Variablen- oder Funktionsnamen verwendet werden:

```
abstract    debugger    final       instanceof  public      transient
boolean     default     finally     int         return      true
break       delete      float       interface   short       try
byte        do          for         long        static      typeof
case        double      function    native      super       undefined
catch       else        goto        new         switch      var
char        enum        if          null        synchronized void
class       export      implements  package     this        volatile
const       extends     import      private     throw       while
continue    false       in          protected   throws      with
```

Farbtabelle der 16 Grundfarben

Farbname	Farbe	Code
red	rot	#FF0000
yellow	gelb	#FFFF00
green	grün	#008000
lime	hellgrün	#00FF00
teal	blaugrün	#008080
olive	ocker	#808000
blue	blau	#0000FF
aqua	cyan (hellblau)	#00FFFF

Farbname	Farbe	Code
navy	dunkelblau	#000080
purple	violett	#800080
fuchsia	magenta	#FF00FF
maroon	dunkelrot	#800000
gray	dunkelgrau	#C0C0C0
silver	hellgrau	#808080
black	schwarz	#000000
white	weiß	#FFFFFF

Speziellere Farben

aliceblue
antiquewhite
aquamarine
azure
beige
bisque
blanchedalmond
blueviolet
brown
burlywood
cadetblue
chartreuse
chocolate
coral
cornflowerblue
darkslategray
darkturquoise
darkviolet
deeppink
deepskyblue
dimgray
dodgerblue
firebrick
floralwhite
forestgreen
gainsboro
ghostwhite
gold
goldenrod
greenyellow
lightpink
lightsalmon
lightseagreen
lightskyblue
lightslategray
lightsteelblue
lightyellow
limegreen
linen
magenta
mediumaquamarine
mediumblue
mediumorchid
mediumpurple
mediumseagreen
paleturquoise
palevioletred
papayawhip
peachpuff
pery
pink
plum
powderblue
rosybrown
royalblue
saddelbrown
salmon
sandybrown
seagreen
seashell

cornsilk	honeydew	mediumslateblue	sienna
crimson	hotpink	mediumspringgreen	skyblue
darkblue	indianred	mediumturquoise	slateblue
darkcyan	indigo	mediumvioletred	slategray
darkgoldenrod	ivory	midnightblue	snow
darkgray	khaki	mintcream	springgreen
darkgreen	lavender	mistyrose	steelblue
darkkhaki	lavenderblush	moccasin	tan
darkmagenta	lawngreen	navajowhite	thistle
darkolivegreen	lemonchiffon	oldlace	tomato
darkorange	lightblue	olivedrab	turquoise
darkorchid	lightcoral	orange	violet
darkred	lightcyan	orangered	wheat
darksalmon	lightgoldenrodyellow	orchid	whitesmoke
darkseagreen	lightgreen	palegoldenrod	yellowgreen
darkslateblue	lightgrey	palegreen	

Eine Übersicht der Farbnamen mit hexadezimalen und RGB-Werten finden Sie in den Übungsdateien.

A.2 Bit-Operatoren

Bit-Operatoren werden auf Zahlen und boolesche Werte angewendet, die entsprechend ihrer binären Darstellung verknüpft werden (die Bits können z. B. für bestimmte Eigenschaften stehen). Die booleschen Werte `true` und `false` entsprechen dabei den Werten 1 und 0.

In den folgenden Beispielen werden die Operatoren auf Dezimalzahlen angewendet.

Name	Operator	Syntax	Beispiel	Ergebnis
Bitweises NICHT	`~`	`~Operand`	`~12`	0000...00001100 12 1111...11110011 = **-13**
Bitweises ODER	`|`	`Operand1 | Operand2`	`12 | 6`	0000...00001100 12 0000...00000110 6 0000...00001110 = **14**
Bitweises UND	`&`	`Operand1 & Operand2`	`12 & 6`	0000...00001100 12 0000...00000110 6 0000...00000100 = **4**
Bitweises XODER (Exklusiv-ODER)	`^`	`Operand1 ^ Operand2`	`12 ^ 6`	0000...00001100 12 0000...00000110 6 0000...00001010 = **10**
Arithmetische Linksverschiebung	`<<`	`Operand << Anzahl`	`12 << 2`	0000...00001100 12 0000...00110000 = **48**
Arithmetische Rechtsverschiebung	`>>`	`Operand >> Anzahl`	`13 >> 2`	0000...00001101 13 0000...00000011 = **3**
			`-13 >> 2`	1111...11110011 -13 1111...11111100 = **-4**
Logische Rechtsverschiebung	`>>>`	`Operand >>> Anzahl`	`13 >>> 2`	0000...00001101 13 0000...00000011 = **3**
			`-13 >>> 2`	1111...11110011 -13 0011...11111100 = **1073741820**

Anmerkung zur Verschiebung

- Die arithmetische Linksverschiebung liefert den Zahlenwert von `Operand` um `Anzahl` Bit-Stellen nach links verschoben. Mathematisch gesehen handelt es sich hierbei um eine Multiplikation mit 2 ^{Anzahl}.
- Die arithmetische Rechtsverschiebung liefert den Zahlenwert von `Operand` um `Anzahl` Bit-Stellen nach rechts verschoben, wobei das Vorzeichen erhalten bleibt. Mathematisch gesehen handelt es sich hierbei um eine ganzzahlige Division mit 2 ^{Anzahl}.
- Die logische Rechtsverschiebung liefert den Zahlenwert von `Operand` um `Anzahl` Bit-Stellen nach rechts verschoben, wobei mit Nullen aufgefüllt wird. Diese Verschiebung wird auch Zero-Fill-Rechtsverschiebung genannt.

A.3 Übersichten der Objektmodelle

Die folgenden Listen geben Ihnen einen schnellen Überblick über die Eigenschaften der Objektmodelle der Browser. Dabei wurde Wert darauf gelegt, möglichst die wichtigsten und browserunabhängigen Eigenschaften darzustellen (Eigenschaften, die nur in einem Browser vorhanden sind, werden als solche gekennzeichnet). Außerdem erscheinen einige Eigenschaften, die im Buch nicht oder nur unvollständig besprochen wurden. Dazu finden Sie am Ende dieses Buches interessante Links für weiterführende Informationen im Internet.

`window`-, `location`- und `frames`-Objekte

window

- alert(message: string)
- blur()
- clearInterval(intervalID)
- clearTimeout(timeOutID)
- close()
- confirm(message: string) : boolean
- defaultStatus: string
- document: [Object]
- event: [Object] (nur IE)
- focus()
- frames: [Array]
- history: [Object]
- location: [Object]
- moveBy(x,y: integer)
- moveTo(x,y: integer)
- open(Url, windowname [,windowFeatures]: string)
- opener: [Object]
- prompt(message, defaultReply: string) : string
- scrollBy(x,y: integer)
- scrollTo(x,y: integer)
- self: [Object]
- setInterval(code: string, msek: integer) : intervalID
- setTimeOut(code: string, msek: integer) : timeOutI
- status: string

window

- location
  - hash: string
  - host: string
  - hostname: string
  - href: string
  - pathname: string
  - port: string
  - protocol: string
  - reload()
  - replace(Url: string)
  - search: string

window

- frames
  - length: integer
  - location: string
  - name: string
- frames[index]
  - parent: [Object: frame]
  - self: [Object: frame]
  - top: [Object: window]

`document`- und `images`-Objekte

- **window**
  - document
    - alinkColor: string
    - anchors: [Array]
    - applets: [Array]
    - bgColor: string
    - captureEvents()
    - clear()
    - close()
    - cookie: string
    - documentElement: [Array]
    - fgColor: string
    - forms: [Array]
    - getElementById(text: string)
    - getElementsByName(text: string)
    - getElementsByTagName(text: string)
    - images: [Array]
    - lastModified: string
    - linkColor: string
    - links: [Array]
    - location: string
    - open(mimeType: string [, "replace"])
    - plugins: [Object]
    - referrer: string
    - title: string
    - url: string
    - vlinkColor: string
    - write(text: string)
    - writeln(text: string)

- **window**
  - document
    - images[index]
      - border: integer
      - complete: string
      - height: integer
      - hspace: integer
      - lowsrc: string
      - name: string
      - src: string
      - vspace: integer
      - width: integer

`screen`- und `navigator`-Objekte

- **window**
  - screen
    - availHeight: integer
    - availWidth: integer
    - colorDepth: integer
    - height: integer
    - width: integer
    - availLeft: integer (nur Firefox)
    - availTop: integer (nur Firefox)
    - left: integer (nur Firefox)
    - pixelDepth: integer (nur Firefox)
    - top: integer (nur Firefox)

- **window**
  - navigator
    - appCodeName: string
    - appName: string
    - appVersion: string
    - javaEnabled: boolean
    - platform: string
    - userAgent: string
    - mimeTypes: [Array] (Firefox, Chrome, Safari, Opera)
    - plugins: [Array] (Firefox, Chrome, Safari, Opera)

`forms`- und `event`-Objekte

- **window**
  - document
    - forms
      - length: integer
    - forms[index]
      - action: string
      - encoding: string
      - method: string
      - name: string
      - target: string
      - length: integer
      - elements: [Array]
        - length: integer
      - elements[index]
        - checked: boolean (bei checkbox und radio)
        - defaultchecked: boolean (bei checkbox und radio)
        - defaultValue: string (bei text)
        - name: string
        - value: string
        - options: [Array] (bei select)
          - length: integer
        - options[index] (bei select)
          - index: integer
          - defaultSelected: integer
          - selectedIndex: boolean
          - selected: integer
          - text: string
        - blur()
        - click()
        - focus()
        - select()
      - reset()
      - submit()

- **window**
  - event (statisch beim IE; Übergabeobjekt bei anderen Browsern)
    - altKey: boolean
    - cancelBubble: boolean
    - clientX: integer
    - clientY: integer
    - ctrlKey: boolean
    - screenX: integer
    - screenY: integer
    - shiftKey: boolean
    - target: [Object]
    - type: string
    - data: string (Firefox, Chrome, Safari, Opera)
    - offsetX: integer (IE, Chrome, Safari, Opera)
    - offsetY: integer (IE, Chrome, Safari, Opera)
    - pageX: integer (Firefox, Chrome, Safari, Opera)
    - pageY: integer (Firefox, Chrome, Safari, Opera)
    - target: [Object] (Firefox, Chrome, Safari, Opera)
    - which: integer (Firefox, Chrome, Safari, Opera)
    - button: integer (IE, Firefox, Safari, Opera)
    - fromElement: [Object] (IE, Chrome, Safari)
    - keyCode: integer (IE, Chrome, Safari, Opera)
    - returnValue: boolean (IE, Chrome, Safari)
    - toElement: [Object] (IE, Chrome, Safari)
    - srcElement: [Object] (IE, Chrome, Safari, Opera)
    - x: integer (IE, Chrome, Safari, Opera)
    - y: integer (IE, Chrome, Safari, Opera)

`style`-Objekte

- **window**
  - document
    - documentElement: [Array]
    - getElementById(text: string)
    - getElementsByName(text: string)
    - getElementsByTagName(text: string)
      - style
        - background, backgroundAttachment, backgroundColor, backgroundImage, backgroundPosition, backgroundRepeat
        - border, borderColor, borderStyle, borderWidth
        - borderBottom, borderBottomColor, borderBottomStyle, borderBottomWidth
        - borderLeft, borderLeftColor, borderLeftStyle, borderLeftWidth
        - borderRight, borderRightColor, borderRightStyle, borderRightWidth
        - borderTop, borderTopColor, borderTopStyle, borderTopWidth
        - bottom
        - captionSide
        - clear
        - clip
        - color
        - cursor
        - direction
        - display
        - emptyCells
        - cssFloat
        - font, fontFamily, fontSize, fontStretch, fontStyle, fontVariant, fontWeight
        - height
        - left
        - letterSpacing
        - lineHeight
        - listStyle, listStyleImage, listStylePosition, listStyleType
        - margin, marginBottom, marginLeft, marginRight, marginTop
        - maxHeight, maxWidth, minHeight, minWidth
        - overflow
        - padding, paddingBottom, paddingLeft, paddingRight, paddingTop
        - pageBreakAfter, pageBreakBefore
        - position
        - right
        - tableLayout
        - textAlign
        - textDecoration
        - textIndent
        - textTransform
        - top
        - verticalAlign
        - visibility
        - width
        - wordSpacing
        - zIndex

A.4 Eventhandler (Stand HTML5)

Ereignisse des Objekts `window`

Ereignis	Wird ausgelöst ...
`onafterprint`	nachdem gedruckt wurde
`onbeforeprint`	vor dem Aufruf des Druckdialogs
`onbeforeonload`	vor dem Laden der Webseite
`onblur`	wenn das Fenster den Fokus verliert
`onfocus`	wenn das Fenster den Fokus erhält
`onhaschange`	wenn das HTML-Dokument geändert wurde
`onload`	beim Laden einer Webseite
`onmessage`	wenn eine per `postMessage` gesendete Nachricht empfangen wird
`onoffline`	wenn die Webseite offline geht
`ononline`	wenn die Webseite wieder online ist
`onpagehide`	wenn das aktive Fenster verschwindet
`onpageshow`	wenn das Fenster wieder sichtbar wird
`onpopstate`	wenn der Verlauf geändert wurde
`onredo`	wenn sich das Dokument erneut aufbaut
`onresize`	sobald die Größe des Browserfensters verändert wurde
`onstorage`	wenn Inhalte auf einem Webserver gespeichert werden
`onundo`	wenn die Änderungen in einem Dokument rückgängig gemacht werden
`onunload`	beim Verlassen einer Webseite

Tastaturereignisse

Ereignis	Wird ausgelöst ...
`onkeydown`	beim Betätigen einer Taste
`onkeypress`	beim Betätigen und beim Loslassen einer Taste
`onkeyup`	beim Loslassen einer Taste

Maus-Ereignisse

Ereignis	Wird ausgelöst ...
`onclick`	beim Klicken auf ein Element
`ondblclick`	beim doppelten Anklicken (wird kaum verwendet)
`ondrag`	wenn ein Element mit der Maus über das aktive Element gezogen wird
`ondragend`	wenn der Drag-&-Drop Vorgang beendet ist
`ondragenter`	wenn ein Element über ein Ziel bewegt wurde, auf dem das Element abgelegt werden darf
`ondragleave`	wenn das Element aus dem Ziel bewegt wird, auf dem das Element abgelegt werden darf

Ereignis	Wird ausgelöst ...
ondragover	wenn ein Element soeben über ein Ziel bewegt wird, auf dem das Element abgelegt werden darf
ondragstart	sobald ein Element mit der Maus verschoben wird
ondrop	wenn das Element über dem aktiven Element abgelegt wurde
onmousedown	beim Betätigen der Maustaste
onmousemove	beim Verschieben der Maus
onmouseout	beim Verlassen eines Elements mit der Maus
onmouseover	beim Überfahren eines Elements mit der Maus
onmouseup	nach dem Loslassen der Maustaste
onmousewheel	beim Betätigen des Scrollrads der Maus
onscroll	während gescrollt wird

Formular-Ereignisse

Ereignis	Wird ausgelöst ...
onblur	beim Verlassen eines Elements
onchange	beim Ändern von Angaben
oncontextmenu	wenn das Kontextmenü aufgerufen wird, z. B. mit der rechten Maustaste
onfocus	beim Aktivieren eines Formular-Elements
onformchange	wenn das Formular verlassen wird und der Anwender etwas eingegeben hat
onforminput	wenn ein Anwender etwas in das Formular eingibt
oninput	sobald etwas in das Eingabefeld eingegeben wird
oninvalid	wenn das Element nicht gültig ist
onreset	beim Zurücksetzen eines Formulars (nur bis HTML4)
onselect	beim Selektieren von Text in Eingabefeldern
onsubmit	beim Absenden von Formulardaten

Medien-Ereignisse

Ereignis	Wird ausgelöst ...
onabort	bei Abbruch des Skripts
oncanplay	wenn die entsprechende Datei schon so weit geladen ist, dass begonnen werden kann, sie abzuspielen
oncanplaythrough	wenn die Datei komplett ohne Wartezeit abgespielt werden kann
ondurationchange	wenn sich die Länge der geladenen Datei ändert
onemptied	wenn die Datei defekt ist oder nicht geladen werden kann
onended	wenn das Medium das Ende erreicht hat
onerror	bei einem auftretenden Fehler
onloadeddata	wenn das Medium geladen wurde
onloadedmetadata	wenn die Metadaten geladen wurden, wie z. B. Gesamtlänge
onloadstart	sobald das Medium geladen wird

Ereignis	Wird ausgelöst ...
onpause	wenn das Abspielen angehalten wurde
onplay	wenn das Medium fertig zum Abspielen ist
onplaying	während des Abspielens
onprogress	wenn der Browser die Angaben erhält, wie viel Prozent der Daten bereits geladen wurden
onreadystatechange	jedes Mal, wenn sich der Fertig-Status ändert
onseeked	nachdem das Vor- und Zurückspulen beendet wurde
onseeking	beim Vor- und Zurückspulen
onstalled	wenn der Browser keine Metadaten auslesen kann
onsuspend	zu dem Zeitpunkt, wenn das Laden der Daten unterbrochen wird
ontimeupdate	wenn die Abspielposition verändert wurde
onvolumechange	wenn die Lautstärke verändert, abgeschaltet oder eingeschaltet wurde
onwaiting	wenn der Player auf Daten wartet und pausiert

Ein Bespiel finden Sie unter *http://www.w3.org/2010/05/video/mediaevents.html.*

A.5 JavaScript-Techniken für ältere Skripte

In der modernen JavaScript-Programmierung verzichtet man auf zahlreiche Vorgehensweisen und Techniken, die man noch vor wenigen Jahren genutzt hat. Wenn Sie ältere Skripte sehen, kann es sein, dass Sie dort diese Techniken vorfinden. Der Abschnitt fasst wichtige Dinge zusammen, die dort auftreten können.

Fenster öffnen und schließen

Bei alten Webseiten wurden sehr oft neue Browserfenster geöffnet, um weitere Inhalte anzuzeigen. Das wird von modernen Browsern unterbunden, und deshalb ist diese Technik nur noch selten zu finden. Grundsätzlich geht das aber mit Methoden und Eigenschaften des `window`-Objekts.

Zum Öffnen von Browserfenstern interessante Eigenschaften des `window`-Objekts

Eigenschaft	Bedeutung
opener	Die Eigenschaft `opener` enthält die Referenz auf das Fenster, welches das aktuelle Fenster geöffnet hat.
self	Die Eigenschaft `self` enthält immer die Referenz auf das aktuelle Objekt.

Grundsätzlich geht das Öffnen mit der Methode `open()`.

```
window.open("URL", "Fenstername", [Optionen])
```

- ✔ Die angegebene URL wird in einem Fenster mit den angegebenen Fensternamen geöffnet. Geben Sie eine leere Zeichenkette an, wird eine leere Seite angezeigt.
- ✔ Damit Sie die Inhalte von geöffneten Fenstern ansprechen können, müssen Sie dem Fenster einen eindeutigen Namen geben.
- ✔ Über weitere Optionen können Sie die Anzeige des Fensters konfigurieren.
- ✔ Als Rückgabewert wird eine Referenz auf das `window`-Objekt des erstellten Fensters geliefert.

Beispiel: *Anhang/objekt_window_open.html*

Beim Laden der Webseite sollen zwei weitere Fenster per JavaScript geöffnet werden. Damit Sie die geöffneten Fenster per JavaScript steuern können, werden die Referenzen auf die `window`-Objekte der neuen Fenster in Variablen gespeichert.

```
<body>
  <h3>Das Hauptfenster</h3>
  <script type="text/javascript">
①   Fenster1 = window.open("objekt_window_status.html", "FensterEins");
②   Fenster2 = window.open("", "FensterZwei");
  </script>
</body>
```

① Innerhalb des <`body`>-Tags wird der JavaScript-Bereich ausgeführt. Mit `window.open` öffnen Sie das HTML-Dokument *objekt_window_status.html* in einem neuen Fenster. Intern erhält dieses Fenster den Namen `FensterEins`. Das `window`-Objekt wird der Variablen `Fenster1` übergeben, sodass Sie die Möglichkeit haben, die Elemente des neuen Fensters z. B. über `Fenster1.Eigenschaft` anzusprechen.

② Das zweite Fenster ist leer, erhält den Namen `FensterZwei` und kann über die Objektreferenz `Fenster2` angesprochen werden.

Beim Öffnen eines neuen Fensters können Sie weitere Optionen für die Anzeige des Fensters festlegen, z. B. ob im geöffneten Fenster die Statuszeile oder die Adresszeile angezeigt werden soll. Folgende Einstellungen können Sie für verschiedene Browser anwenden, wobei die Unterstützung Browser-abhängig ist und wie gesagt in neueren Browsern grundsätzlich verhindert wird.

Option	Erklärung
`menubar`	Anzeige der Menüleiste ① (`yes`, `no` = Standard)
`toolbar`	Anzeige der Symbolleiste ② (`yes`, `no` = Standard)
`location`	Anzeige der Adresszeile ③ (`yes`, `no` = Standard)
`status`	Anzeige der Statuszeile ④ (`yes`, `no` = Standard)
`resizable`	Fenstergröße ⑤ veränderbar (`yes` = Standard, `no`)
`scrollbars`	Bildlaufleisten ⑥ einblenden (`yes`, `no` = Standard). Firefox und Opera blenden sie erst ein, wenn der Inhalt der Webseite entsprechend lang ist.
`height`	Festlegung der Höhe des Fensters in Pixeln
`width`	Festlegung der Breite des Fensters in Pixeln
`top`	Anzeigeposition von oben
`left`	Anzeigeposition von links

Beispiel

Sie öffnen ein Fenster, in dem die Anzeige aller Fensterelemente abgeschaltet ist. Die Höhe und die Breite des Fensters sollen dabei jeweils 300 Pixel betragen.

```
<script type="text/javascript">
  Fenster = window.open("","Individuell",
            "menubar=no, location=no, resizable=no,
             status=no, height=300, width=300");
</script>
```

Beachten Sie, dass die Fensteroptionen in einer Zeichenkette angegeben werden. Die einzelnen Optionen werden jeweils durch ein Komma voneinander getrennt.

Ein Fenster können Sie über die Methode `window.close()` schließen. Beispielsweise lässt sich diese Funktion als Link im HTML-Dokument darstellen.

```
<a href="javascript:window.close();">Aktuelles Fenster schließen</a>
// oder
<a href="javascript:self.close();">Aktuelles Fenster schließen</a>
```

Nutzen des Objektfeldes `frames`

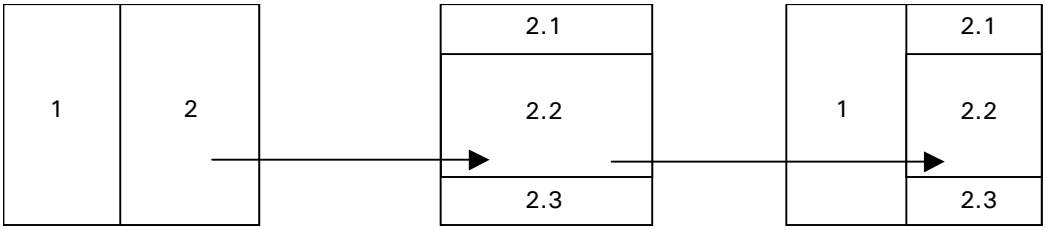
Früher hat man Webseiten oft mit Segmenten unterteilt, die Frames genannt wurden. In einem Fenster können mehrere Frames enthalten sein. In jeden Frame kann wiederum ein HTML-Dokument geladen werden, in dem erneut Frames definiert sind.

Das `frames`-Objektfeld ist eine Eigenschaft des `window`-Objekts, das selbst die gleichen Eigenschaften und Methoden wie das `window`-Objekt selbst besitzt. Auf die einzelnen Unterframes können Sie über das `frames`-Objektfeld zugreifen.

Eigenschaft	Bedeutung
`frames[]`	Hiermit erhalten Sie über ein Array Zugriff auf alle geladenen Frames. Diese können Sie jeweils durch die Angabe des Index ansprechen.
`frames[].name`	Den jeweiligen Namen des Frames, der mit `<frame name="">` definiert wurde, erhalten Sie mit der Abfrage der Eigenschaft `name`.
`length`	Die Anzahl der im Browser angezeigten Frames können Sie mit der Eigenschaft `length` auslesen.
`self, window`	Über die Eigenschaften `self` und `window` sprechen Sie den aktuellen Frame an.
`parent`	Mit `parent` können Sie auf das übergeordnete Fenster des Frames zugreifen.

Um einen Frame ansprechen zu können, geben Sie entweder den Namen des Frames aus `window.FrameName` an oder verwenden Sie einen Index. Beachten Sie, dass der Zähler bei null beginnt, d. h., das erste Frame-Fenster sprechen Sie mit `frames[0]` an, das zweite Frame-Fenster mit `frames[1]` usw. Beim Zählen gilt die Reihenfolge, in der die Frames im Frameset definiert sind.

Die Ermittlung der Eigenschaften eines Framesets soll Ihnen anhand eines Beispiels verdeutlicht werden.



Aufteilung eines Framesets in vier Frames

Beispiel: *Anhang/objekt_frames.html*

Zuerst erstellen Sie eine Webseite mit einem Frameset, das wiederum zwei Frames beinhaltet:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Objekt: frames</title>
</head>
<frameset cols="50%,*">
  <frame name="FLinks" src="flinks.html">   <!-- Frame 1 -->
  <frame name="FRechts" src="frechts.html"> <!-- Frame 2 -->
</frameset>
</html>
```

Beispiel: *Anhang/frechts.html*

Weiterhin erstellen Sie den rechten Frame (2), der als Frameset für weitere drei Frames genutzt wird.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Objekt: frames</title>
</head>
<frameset rows="1,1,1">
  <frame name="FOben" src="foben.html">       <!-- Frame 2.1 -->
  <frame name="FZentrum" src="fzentrum.html"> <!-- Frame 2.2 -->
  <frame name="FUnten" src="funten.html">     <!-- Frame 2.3 -->
</frameset>
</html>
```

Beispiel: *Anhang/fzentrum.html, Anhang/foben.html, Anhang/funten.html, Anhang/flinks.html*

Nachfolgend wird der Code der HTML-Dokumente des rechten Frames 2.2 näher betrachtet. Alle weiteren Frames *foben.html*, *funten.html* und *flinks.html* enthalten die gleichen Anweisungen.

```
    <!DOCTYPE html>
    <html>
    <head>
      <meta charset="utf-8">
      <title>Objekt: frames</title>
    <script type="text/javascript">
      function ausgabe()
      {
        document.write('Frame 2.2<br>');
①       document.write("Das übergeordnete Dokument heißt:"+parent.name + "<br>");
②       document.write("Frames im übergeordneten Dokument: " +                  ⏎
                       parent.frames.length + "<br>");
        document.write("Diese lauten: ");
③       for(var i = 0; i < parent.frames.length; i++)
④         document.write(parent.frames[i].name + " ");
        document.write("<br>");
⑤       document.write("Dieses Frame heißt: " + self.name + "<br>");
⑥       document.write("Frames im Hauptdokument: " + top.frames.length + "<br>");
      }
    </script>
    </head>
    <body onload="ausgabe();">
    </body>
    </html>
```

① Das `frames`-Objekt besitzt, wie viele andere Objekte auch, die Eigenschaft `name`. Die Objektvariable `parent` verweist dabei auf den übergeordneten Frame, der den aktuellen Frame aufgerufen hat.

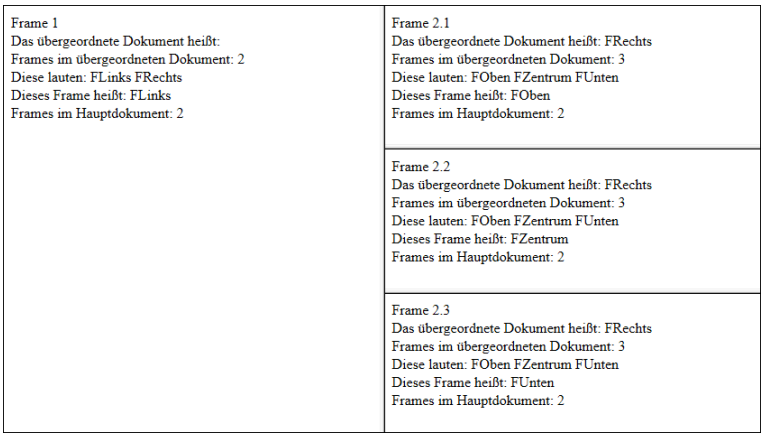
② Die Eigenschaft `parent.frames.length` gibt die Anzahl der Frames zurück, die vom übergeordneten Frame erstellt wurden.

③ Um die Namen aller im übergeordneten Frame befindlichen Frames auszulesen, müssen Sie die Anzahl der vorhandenen Frames abfragen (`parent.frames.length`).

④ Innerhalb der `for`-Schleife wird die Zählvariable `i` um den Wert 1 erhöht. Der Name jedes einzelnen Frames wird daraufhin über `parent.frames[i].name` ausgelesen.

⑤ Den Namen des aktiven Frames erhalten Sie mit der Anweisung `self.name`.

⑥ Auf das oberste Frameset greifen Sie über die Eigenschaft `top` zu. Die Angabe `top.frames.length` gibt daraufhin die Anzahl der darin befindlichen Frames zurück.



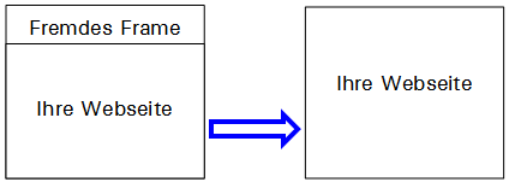
Ausgabe der einzelnen Frame-Eigenschaften

Durch das Auslesen der Frames-Eigenschaften können Sie gezielt auf die Frames zugreifen und den Inhalt verändern, ohne die Namen der einzelnen Frames zu kennen.

Zwei weitere Beispiele sollen Ihnen die Wirkungsweise des `frames`-Objekts sowie des bereits erklärten `location`-Objekts erläutern. Beispielsweise kann ein anderer Autor von seiner Webseite auf Ihre Webseite über einen Link verweisen. Verwendet die aufrufende Webseite Frames, kann es vorkommen, dass Ihre Seite innerhalb seines Framesets aufgerufen wird. Ein Besucher kann dadurch den Eindruck gewinnen, dass Ihre Webseite Bestandteil der fremden Homepage ist. Um dies zu verhindern, verwenden Sie das `location`-Objekt, um das oberste Frameset zu überladen.

Seite aus einem Frameset befreien

```
<script type="text/javascript">
  if(top.frames.length > 0)
    top.location.href = self.location;
</script>
```

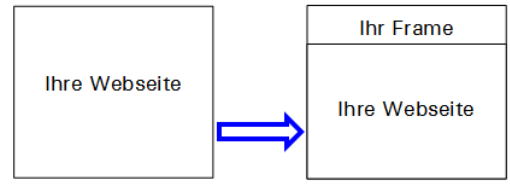


Das Skript kontrolliert, ob sich Ihre Webseite in einem Frameset befindet (`frames.length > 0`). Ist dies der Fall, wird die Seite in das oberste Frameset geladen (`top.location.href`). Die Adresse Ihrer Webseite erhalten Sie durch die Abfrage `self.location`.

Ein weiteres Beispiel ist das Laden von Webseiten in einen Frame. Öffnen Besucher über den Link einer Suchmaschine eine Ihrer Webseiten, die eigentlich Teil eines Framesets ist, können Sie das vollständige Frameset nachladen lassen.

Seite in ein Frameset laden

```
<script type="text/javascript">
  if(top.frames.length == 0)
    top.location.href = "frame.html";
</script>
```



Dieses Skript überprüft, im Gegensatz zum vorherigen Skript, ob sich die Seite nicht in einem Frame befindet (`frames.length == 0`). Trifft dies zu, wird die Webseite mit der Frame-Definition geladen, in diesem Beispiel die Webseite `frame.html`.

Benannte `break`- und `continue`-Anweisungen

Durch die Angabe einer Sprungmarke **kann in JavaScript auch aus Schleifen (auch aus verschachtelten)** herausgesprungen werden.

```
var j = 0;
label:
while(j == 0)
{
  for(j = 1; j < 10; j++)
  {
    ...
    break label;
    // oder
    // continue label;
  }
}
//hier geht's weiter
```

- ✔ Die Sprungmarke (Label) muss direkt vor einem Anweisungsblock liegen, der die entsprechende `break`- oder `continue`-Anweisung umschließt. Dabei können die Anweisungen auch tiefer verschachtelt sein.
- ✔ Der Sprung zum Label kann innerhalb einer beliebig tiefen Verschachtelungsebene des betreffenden Anweisungsblocks erfolgen.
- ✔ Es können mehrere Verweise auf ein Label existieren, aber ein Labelbezeichner darf nicht mehrfach verwendet werden.
- ✔ Der `break`- oder `continue`-Anweisung folgt dann der Name des Labels. Bei der Verwendung von `continue` wird wieder zu der beim Label stehenden Anweisung gesprungen und diese ausgeführt. Bei Schleifen bedeutet dies, dass die Schleifenbedingung überprüft wird (eine `for`-Schleife wird jedoch nicht erneut initialisiert). Bei der Verwendung von `break` wird aus der Schleife gesprungen und die erste Anweisung nach dem Anweisungsblock ausgeführt.

Im obigen Beispiel wird während des ersten Durchlaufs der `for`-Anweisung zum Label `label` gesprungen. Da der Sprung über `break` erfolgte, wird die nächste Anweisung nach dem folgenden Block, also nach der `while`-Anweisung, ausgeführt.

- ✔ Im Beispiel würde der Aufruf von `continue` mit einem Label bezüglich der `for`-Anweisung nichts bringen. Dadurch, dass sich das Label im äußeren Anweisungsblock befindet, wird die `while`-Bedingung erneut geprüft. Deren Ausdruck `j == 0` liefert `false`, da `j` jetzt den Wert 1 besitzt. Die Anweisungen werden wie bei der Verwendung von `break` nach der `while`-Anweisung fortgesetzt.
- ✔ Diese Anweisungen sollten Sie sehr sparsam einsetzen, da sie zu einer schlechteren Verständlichkeit des Quelltextes führen.

A.6 Informationen im Internet

JavaScript: Freie JavaScripte, Tutorials, Beispiele, Referenzen, Hilfe

https://developer.mozilla.org/en/JavaScript

http://de.selfhtml.org/javascript/

http://www.javascriptsource.com/

http://www.pageresource.com/jscript/

http://www.webreference.com/programming/javascript/index.html

DOM 2.0 Standard

http://www.w3.org/DOM/

DHTML Lab: Free Dynamic HTML Tutorials, DHTML Scripts, Programming, Tools, Code and Examples

http://www.webreference.com/dhtml/

jQuery, jQuery UI: Offizielle Webseite, Dokumentation und Erweiterungen

http://jquery.com/

http://docs.jquery.com/Tutorials

http://api.jquery.com/browser/

http://jqueryui.com/

http://jqueryui.com/demos/

A.7 Glossar

Anker	Bestimmte Stelle innerhalb eines HTML-Dokuments, zu der der Benutzer mit einem Verweis springen kann
ActiveX	Active eXtension ist eine von Microsoft entwickelte Technologie, um einen Datenaustausch zwischen Softwarekomponenten zu ermöglichen, so auch im Browser.
Ajax	Asynchronous JavaScript and XML; Konzept der asynchronen Datenübertragung zwischen Browser und Server, ohne die Webseite neu zu laden
Auszeichnungssprache	Sprache, die die logische Struktur von Dokumenten beschreibt. Verschiedene Aspekte innerhalb des Dokuments (z. B. Überschrift oder Schriftart) werden in dieser Sprache ausgezeichnet. HTML ist eine Auszeichnungssprache.
Browser	Programm zum Anzeigen von Inhalten des World Wide Web (WWW). Die bekanntesten Programme sind Microsoft Internet Explorer, Mozilla Firefox und Opera.
CSS	Formatierungssprache, um HTML- und XML-Dokumente entsprechend darzustellen
Client	Computer eines Nutzers, der in einem Netzwerk (Internet/Intranet) die Dienste von Servern in Anspruch nimmt
Compiler	Er übersetzt Quelltexte einer Programmiersprache in Maschinensprache. Übersetzte Programme können zu einem ausführbaren Programm gebunden und in einer Datei gespeichert werden.
Debugger	Programm, das hilft, Fehler in Programmen zu finden und zu beseitigen
DOM	Document Object Model. Es ist eine definierte Schnittstelle, um auf Inhalte von HTML- oder XML-Dokumenten zuzugreifen.
Ereignis, Event	Ist in JavaScript an ein HTML-Objekt gekoppelt; eine Benutzeraktion, wie z. B. ein Mausklick auf dieses Objekt, löst das Ereignis aus.
Eventhandler	Der Browser reagiert auf ein Ereignis durch Aufruf des zugehörigen Eventhandlers (meist eine selbst geschriebene JavaScript-Funktion).
Frame	In HTML definierter Rahmen, in dem eigenständig wie in einem Fenster ein eigenes Dokument geladen und angezeigt werden kann
Frameset	Definition einer Aufteilung des Browserfensters in verschiedenen Frames
HTML	Hypertext Markup Language; Auszeichnungssprache, die im WWW verwendet wird, um Webseiten zu erstellen
HTTP	HyperText Transfer Protocol; standardisiertes Protokoll zur Übertragung von HTML-Dateien und Multimedia-Objekten im WWW
Hyperlink	Text- oder Grafikverweis im Internet auf ein weiteres Dokument

Internet	Weltweite Verbindung von Netzwerken mithilfe des Protokolls TCP/IP; der Begriff umfasst dabei die Kategorien WWW, Mail, FTP, Gopher, Telnet usw.
Interpreter	Der Quelltext einer Skriptsprache wird Schritt für Schritt ausgeführt. Im Gegensatz zu ausführbaren Programmen (vgl. Compiler) muss der Quelltext bei jedem Aufruf des Skripts neu ausgewertet werden. Browser haben einen JavaScript Interpreter.
Java-Applet, Applet	Java-Programm, das für ein HTML-Dokument entwickelt wird
Java-Application	Eigenständiges Java-Programm
jQuery	Populäres JavaScript-Framework zur Manipulation des DOM und zur einfachen Verwendung der Ajax-Funktionalität
JSON (JavaScript Object Notation)	Ein Format zum Datenaustausch, das auf der Objektnotation von JavaScript basiert
MIME	Multipurpose Internet Mail Extension; Erweiterung des SMTP-Protokolls, das die Übertragung von binären Dateien unterstützt (Versenden von Dateien, Grafiken usw.)
Netzwerk	System von Verknüpfungen von sonst unabhängigen Computern, das den Transfer von Informationen zwischen den verknüpften Rechnern ermöglicht
Plug-ins	Software-Produkte, die sich in andere Programme integrieren lassen
Quelltext	ASCII-Text, der von einem Compiler oder einem Interpreter weiterverarbeitet wird
Server	Ein Programm, das Anfragen von einem Client beantwortet. Wird aber oft auch in der Umgangssprache mit einem Rechner gleichgesetzt , der in einem Netzwerk für mehrere Clients zentralisierte Aufgaben übernimmt, z. B. Drucker, E-Mail-Dienste, Speicherplatz
Skript	Quellcode, der in einer Skriptsprache geschrieben wurde und von einem Interpreter ausgeführt wird
Skriptsprache	Eine Programmiersprache zur Erstellung von einem Skript, das von einem Interpreter ausgewertet wird
Tag (HTML-Tag)	Element einer Auszeichnungssprache
URL	Unified Resource Locator; weltweit einmalige Internetadresse, die das Protokoll der Übertragung und den Domain-Namen enthält, z. B. *ftp://ftp.mozilla.com/* oder *http://www.w3.org/MarkUp*
VBScript	Visual Basic Script ist eine JavaScript-ähnliche Skriptsprache, die von Microsoft entwickelt wurde.
WWW	World Wide Web; der grafisch orientierte Teil des Internets. Das WWW ist aufgrund seiner multimedialen Eigenschaften der derzeit meistgenutzte Dienst im Internet; wird häufig mit dem Internet selbst verwechselt.
XML	Extended Markup Language; Auszeichnungssprache, um strukturierte Daten in Form von Text hierarchisch darzustellen; dient zum plattformunabhängigen Datenaustausch