JavaScript – DOM und BOM

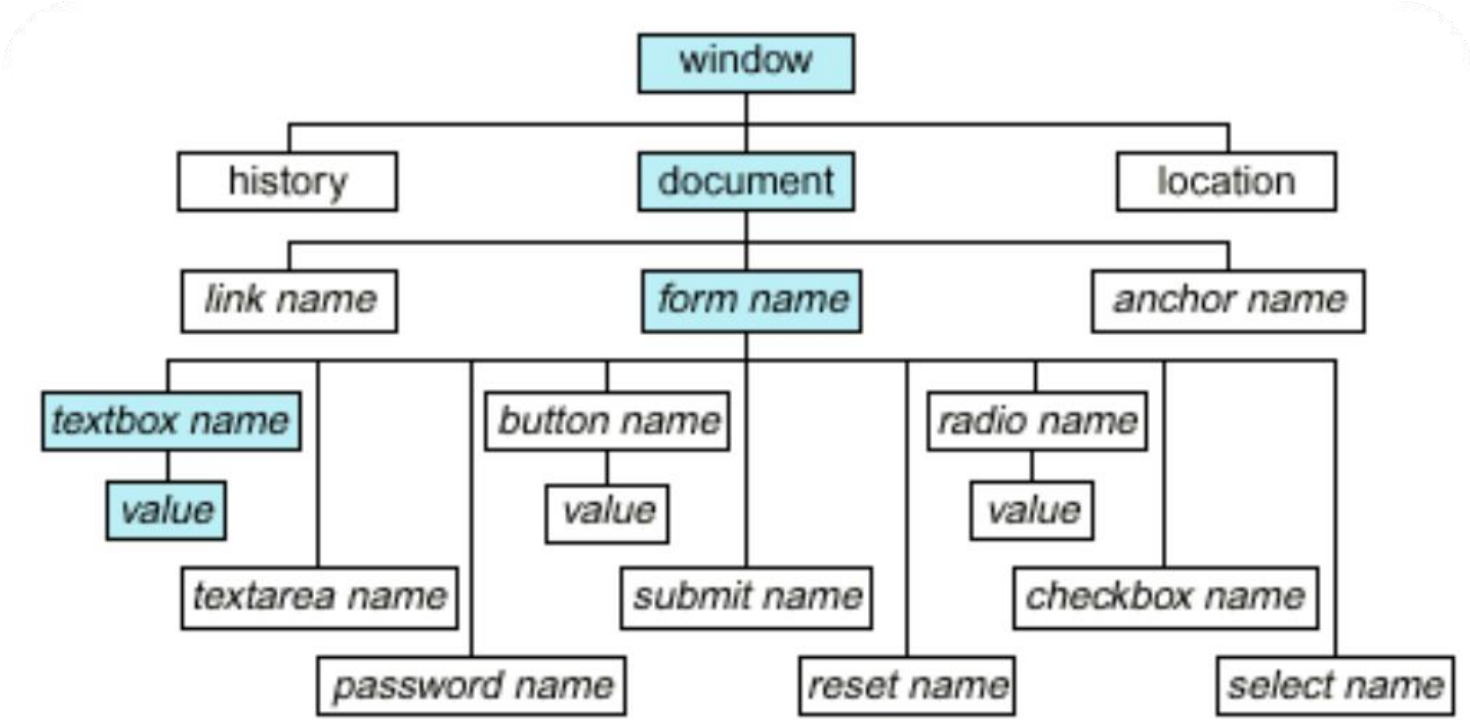
Ingo Köster

Diplom-Informatiker (FH)

DOM-API

› Jedes HTML-Element eines Dokuments ist über die DOM-API für JavaScript erreichbar

› Fast jedes Element kann verändert/angesprochen werden



DOM-Elemente selektieren

- Zugriff über die ID des Elements
  - `let elem = document.getElementById("id");`
- Zugriff über den Namen des Elements (`name="…"`)
  - `let elem = document.getElementsByName("name");`
- Zugriff über eine Objekt-Referenz eines Elements
  - ```
    let el = document.getElementById("nav");
    let imgTags = el.getElementsByTagName("img");
    ```
  - Liefert ein Array von Image-Elementen, die Kinder des Objekts **`el`** sind
    ```

DOM-Elemente selektieren

- Zugriff über die Klasse des Elements
  - `let elem = document.getElementsByClassName("navi");`

- Elemente mit mehreren Klassen
  - `document.getElementsByClassName("red navi");`

- Kombinationen
  - `document.getElementById("main").getElementsByClassName("navi");`

DOM-Elemente selektieren – II

- Zugriff über alle in CSS3 erlaubten Selektoren :
  - `let links = document.querySelectorAll("nav#top > li");`

- Liefert eine Knotenliste mit allen Elementen, auf die der Selektor passt

- Um nur das erste Element zu erhalten, das mit einem CSS-Selektor übereinstimmt wird die Funktion `querySelector("")` verwendet

DOM-Elemente selektieren – III

› Objekte aus dem DOM können ausgewählt werden, um deren Attribute zu verändern

```
function change(state)
{
  let lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".gif";
  let statusDiv =
    document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state;
}
```

…

```
<img id="lamp" src="lamp_on.gif" />
```

Zugriff auf Attribute

- Alle Universalattribute wie `id`, `name`, `style` und `class` lassen sich auslesen und ändern

- Durch das Attribut `style` hat man Zugriff auf die CSS-Eigenschaften eines DOM-Elements

```
let elem = document.getElementById("id");
elem.style.width = "100px";
elem.style.backgroundColor = "#FF0000";
```

Zugriff auf Daten (data-Attribute)

› Seit HTML5 ist das Verknüpfen von Zusatzinformationen mit HTML-Elementen besonders einfach, es gibt die `data`-Attribute:

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe</div>
```

```
let el = document.querySelector('#user');
// el.id == 'user'
// el.dataset.id === '1234567890'
// el.dataset.user === 'johndoe'
// el.dataset.dateOfBirth === ''

el.dataset.dateOfBirth = '1960-10-03';
```

Zugriff auf Daten (data-Attribute)

› Seit HTML5 ist das Verknüpfen von Zusatzinformationen mit HTML-Elementen besonders einfach, es gibt die data-Attribute:

```
<div id="user" data-id="1234567890"
     data-user="johndoe"
     data-date-of-birth>John Doe</div>
```

Syntax

```
string = element.dataset.camelCasedName;
element.dataset.camelCasedName = string;
```

Zugriff über DOM-Eigenschaften

- Jedes Objekt im DOM (Knoten) hat eine Menge von Eigenschaften und Methoden
- Erlaubt es den DOM-Baum zu durchsuchen und gezielt anzupassen

- Oft benötigte Eigenschaften sind:
  - `element.childNodes`
  - `element.parentNode`
  - `element.nextSibling`
  - `element.previousSibling`
  - `element.firstChild`
  - `element.lastChild`

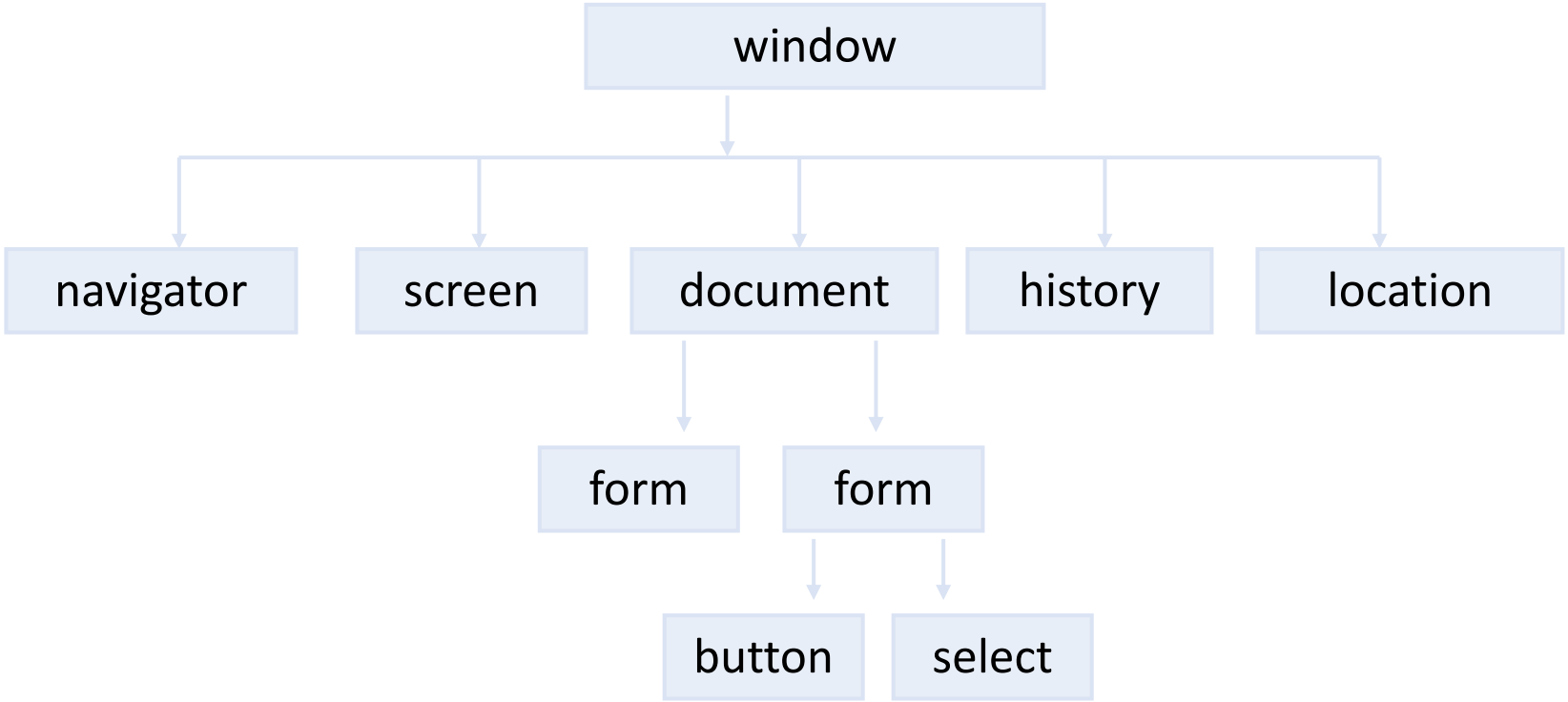
DOM-Methoden

- Jedes Objekt im DOM (Knoten) hat eine Menge von Eigenschaften und Methoden, die wichtigsten sind:
  - `appendChild()`
  - `replaceChild()`
  - `removeChild()`
  - `insertBefore()`
  - `hasChildNodes()`
  - `hasAttributes()`
- Dokumente können auch zur Laufzeit Objekte erzeugen:
  - `createElement()`
  - `createAttribute()`

BOM – Browser Object Model

- Jeder Browser stellt einige read-only Informationen über sich und die geladenen Dokumente bereit:
  - `window`
    - Oberster Knoten in der DOM-Hierarchie des aktuellen Fensters
    - Informationen über das Browser-Fenster mit Verweisen auf weitere Browser-Objekte (location, history etc.)
  - `document`
    - Referenz auf das aktuelle Dokument
  - `screen`
    - Eigenschaften des Browser im Bezug auf die Darstellung
  - `navigator`
    - Informationen über den verwendeten Browser

BOM – Struktur



Events & Event-Handler

›Handlungen des Anwenders wie
- › Klicks
- › Tastatureingaben
- › Mausbewegungen

können Ereignisse auslösen, auf welche mittels JavaScript reagiert werden kann
- › Durch Event-Handler

Events & Event-Handler – II

› Zuweisung eines Event-Handlers in JavaScript

› Sog. Unobtrusive JavaScript bzw. unaufdringliches JavaScript

```
let img = document.getElementById("myImage");

img.onclick = imageClicked;

function imageClicked()
{ … }
```

Events & Event-Handler – III

- Maus-Ereignisse:
  - `onclick, onmousedown, onmouseup`
  - `onmouseover, onmouseout, onmousemove`
- Tastatur-Ereignisse:
  - `onkeypress, onkeydown, onkeyup`
- Interface-Ereignisse:
  - `onblur, onfocus`
  - `onscroll`
- Form-Ereignisse
  - `onsubmit, onchange`

Events & Event-Handler

› Skripte erst dann starten, wenn die HTML-Seite komplett geladen wurde

› Wird JavaScript zuerst ausgeführt, können z.B. Elemente mittels `getElement-`-Funktionen nicht gefunden werden, da sie noch nicht existieren

```
window.onload = function ()
{
    // …
};
```

Event Methode mit Übergabeparameter

- Event Methoden können bei der Zuweisung keine Übergabeparameter enthalten
  - `let img = document.getElementById("myImage");`
  - `img.onclick = imageClicked;`
  - …
  - `function imageClicked() {…}`
- Lösung mittels anonymer Methode
  - `let img = document.getElementById("myImage");`
  - `img.onclick = function() { imageClicked("x.png") };`
  - …
  - `function imageClicked(file) {…}`

Events - Timer

› Eine weitere Möglichkeit Events auszulösen sind Timer

```
let timer = setTimeout('bang()', 5000);
```

5 Sekunden nach dieser Zeile wird die Funktion ausgeführt, es sei denn der Timer wird vorher beendet

```
clearTimeout(timer);
```

Diesen Timer beenden

Events - Timer

› Eine weitere Möglichkeit Events auszulösen sind Intervalle

```
let timer = setInterval('clock()', 1000);
```

Jede Sekunde ausführen, bis der Timer beendet wird

```
clearInterval(timer);
```

Diesen Timer beenden

Events - Timer

› Anstelle eines Strings nur die Funktion an `setTimeout` oder `setInterval` übergeben

› Dann jedoch ohne die runden Klammern

```
let timer = setTimeout(bang, 5000);
let timer = setInterval(clock, 1000);
…
function bang() { … }
function clock() { … }
```

Timer – Beispiel

```
<script>
  function timerFunc() {
    let now = new Date();
    let hour = now.getHours();
    let min = now.getMinutes();
    let sec = now.getSeconds();
    document.getElementById("clock").value =
      "" + hour + ":" + min + ":" + sec;
  }
  setInterval(timerFunc, 1000);
</script>
<input type="text" id="clock" />
```