Javascript
DOM & Events

WEB PROFESSIONALS

Fragen zu Übungen?

Was ist ein Array und was ein Objekt

```
var simpsons = [ // Array von Objekten
  {name: "Bart", age: 10, sex: 'm'},
  {name: "Homer", age: 36, sex: 'm'},
  {name: "Hugo", age: 10, sex: 'm'},
];

for (var i=0; i<simpsons.length; i++) {
  console.log(simpsons[i].age);
}
```

Ausgabe

```
10
36
10
```

```
var simpsons = [ // Array von Arrays
  ["Bart", 10, 'm'],
  ["Homer", 36, 'm'],
  ["Hugo", 10, 'm'],
];

for (var i=0; i<simpsons.length; i++) {
    console.log(simpsons[i][1]);
}
```

Was steckt hinter dem Parameter?

```
var simpsons = [
  {name: "Bart", age: 10, sex: 'm'},
  {name: "Homer", age: 36, sex: 'm'},
  {name: "Hugo", age: 10, sex: 'm'}
];

function meinSortierer(a, b) {
  console.log(a);
  console.log(b);
}
simpsons.sort(meinSortierer);
```

```
var simpsons = [
  {name: "Bart", age: 10, sex: 'm'},
  {name: "Homer", age: 36, sex: 'm'},
  {name: "Hugo", age: 10, sex: 'm'}
];

function meinFilter(a) {
  console.log(a);
}
simpsons.filter(meinFilter);
```

Wann wird eine Funktion aufgerufen?

```
function meineFunktion() {
  return 5;
}
console.log(meineFunktion);
```

```
f meineFunktion() {
  return 5;
}
```

```
console.log(meineFunktion());
```

```
5
```

Timeout und Interval

```
var intervalID = window.setInterval(myCallback, 2000);
function myCallback() {
    console.log('blub');
   // Dieser Code wird alle 2 sekunden ausgeführt
}
```

blub (nach 2 sekunden)

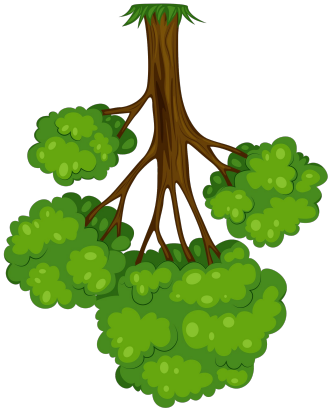
blub (nach 4 sekunden)

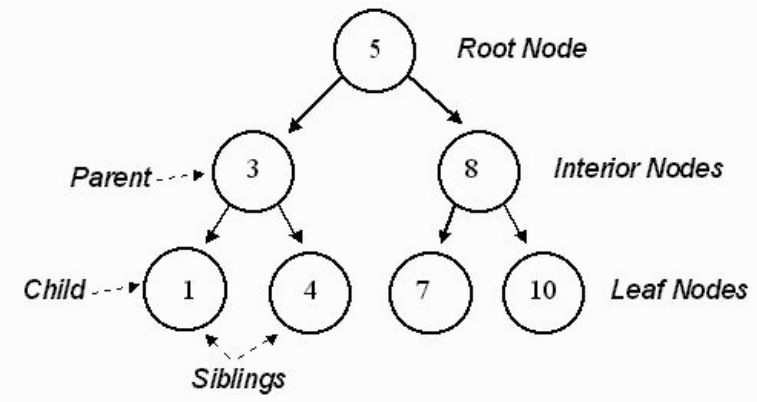
blub (nach 6 sekunden)

DOM

DOM

- DOM = Document object model
- das DOM repräsentiert HTML strukturiert als Baum welches von Javascript verwendet werden kann



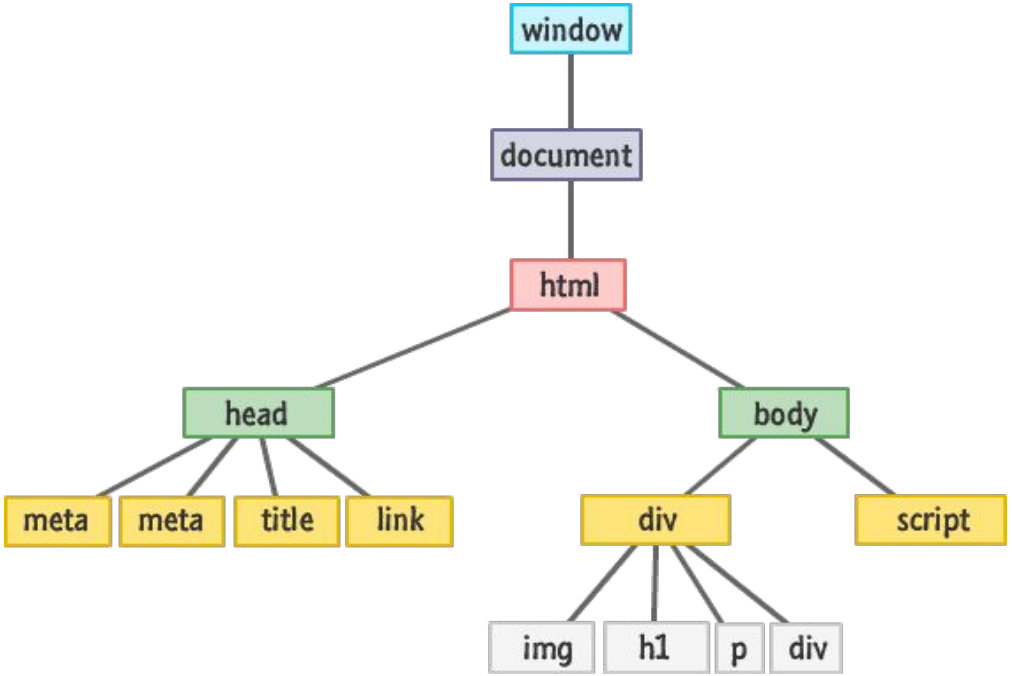


DOM

window

document

```
<html>
<head>
  <meta />
  <meta />
  <title>HTML => DOM</title>
  <link rel="stylesheet" href="...">
</head>
<body>
  <div>
    <img src="...">
    <h1>Titel</h1>
    <p>Text</p>
    <div></div>
  </div>
  <script></script>
</body>
</html>
```



2 verschiedene Welten

```html
<html>
  <body>
    <h1>Hallo</h1>
  </body>
</html>
```

HTML

```javascript
(function () {
  var init = function () {
    console.log('Hallo');
  };
  init();
})()
```

Javascript

Zugriff von Javascript auf DOM

- DOM Funktionen
- Neue Elemente werden erstellt und müssen an einer Stelle eingefügt werden
- Bestehende Elemente müssen "gesucht" werden
- Suche erfolgt mit CSS-Selektoren

DOM Zugriff by TagName (alter Ansatz)

```
<button>A Button</button>


<script>
    var buttons = document.getElementsByTagName('button');
    buttons[0].style.background = "red";
</script>
```

DOM Zugriff by ClassName (alter Ansatz)

```
<button class="my-button">A Button</button>


<script>
    var buttons = document.getElementsByClassName('my-button');
    buttons[0].style.background = "red";
</script>
```

DOM Zugriff by ID (alter Ansatz)

```
<button id="myButton">A Button</button>


<script>
    var myButton = document.getElementById('myButton');
    myButton.style.background = "red";
</script>
```

DOM Zugriff mit Selektoren (neuer Ansatz)

```
<h1 id="myID" class="myClass">Titel 1</h1>
<h1 class="myClass myOtherClass">Titel 2</h1>
<h1 class="myClass">Titel 3</h1>
```

```
var mitId = document.querySelector('#myID'); // Titel 1
var mitKlasse = document.querySelector('.myClass'); // Titel 1
var mitKlassen = document.querySelector('.myClass.myOtherClass'); // Titel 2
```

console.log(mitKlassen) => <h1 class="myClass myOtherClass">Titel 2</h1>

Mehrfach Zugriff mit Selektoren (neuer Ansa.)

```
<h1 id="myID" class="myClass">Titel 1</h1>
<h1 class="myClass myOtherClass">Titel 2</h1>
<h1 class="myClass">Titel 3</h1>
```

```
var mitId = document.querySelectorAll('#myID'); // Titel 1
var mitKlasse = document.querySelectorAll('.myClass'); // Titel 1, Titel 2, Titel 3
var mitKlassen = document.querySelectorAll('.myClass.myOtherClass'); // Titel 2
```

console.log(**mitKlasse**) => NodeList [h1, h1, h1]

console.log(**mitKlassen**) => NodeList [h1]

Attribute eines DOM Elements

```
<h1 id="test-id" class="a b">
    <span>ICON</span> Titeltext
</h1>
```

`ELEMENT.innerHTML`	"ICON Titeltext"	**(Für HTML)**
`ELEMENT.innerText`	"ICON Titeltext"	**(Für normalen Text)**
`ELEMENT.attributes`	{0: id, 1: class, id: id, class: class, length: 2}	
`ELEMENT.classList`	["a", "b", value: "a b"]	
`ELEMENT.className`	"a b"	
`ELEMENT.id`	"test-id"	
`ELEMENT.tagName`	"H1"	
`ELEMENT.style`	"width: 500px; height: 200px; color: blue;"	

Elemente manipulieren

Hast du ein bereits im DOM erstelltes Element per `querySelector` gefunden oder eines mit createElement erstellt, so kannst du dessen Attribute beliebig verändern.

Möchtest du beispielsweise einem Element mittels DOM-Manipulation Text oder HTML eintragen, kannst du das folgendermassen machen:

```
var titel = document.querySelector('.titel');
var div = document.querySelector('div');
titel.innerText = 'Das ist der Titel per Javascript';
div.innerHTML = '<strong>HTML Werte per Javascript</strong>';
=> <h1 class="titel">Das ist der Titel per Javascript</h1>
=> <div><strong>HTML Werte per Javascript</strong></div>
```

Javascript Node Element
zu manipulierendes Attribut
Zu setzende Wert

Style von Elementen manipulieren

Wichtig: CSS-Eigenschaften welche einen Bindestrich beinhalten müssen beim setzen in den sogenannten camelCase umgeschrieben werden.

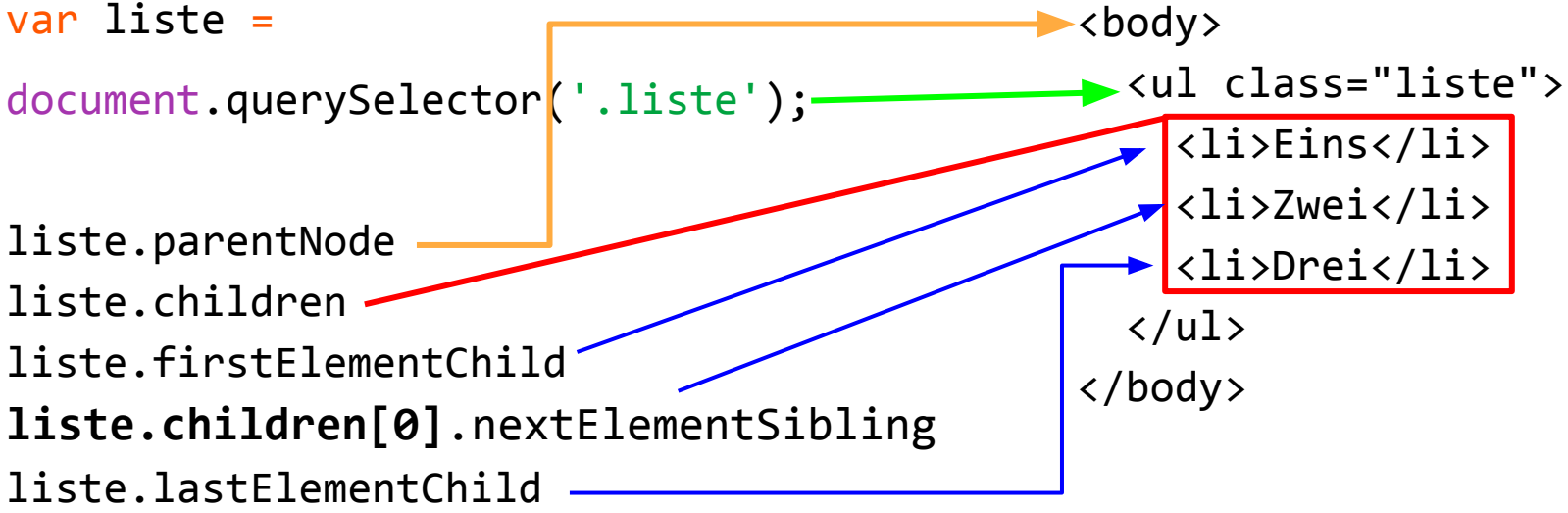
Aus **background-color** (CSS) wird also **backgroundColor** (DOM-Manipulation).

```
var element = document.querySelector('div');
element.style.width = '100px';
element.style.height = '100px';
element.style.backgroundColor = 'red';
```

Javascript Node Element
zu manipulierendes Attribut
zu manipulierende CSS Eigenschaft
Zu setzende Wert

```
=> <div style="width: 100px; height: 100px; background-color:red"></div>
```

Weitere Attribute: Die DOM-Familie



neue DOM Elemente erstellen

```
// Element wird erzeugt
var newElement = document.createElement('div');

// Style Properties des Elements anpassen
newElement.innerText = "Das ist der Text im Div";

// Element in body anhängen (nicht vergessen!!)
document.body.appendChild(newElement);
```

neue DOM Elemente erstellen 2

```
// Element wird erzeugt
var newElement = document.createElement('div');

// Style Properties des Elements anpassen
newElement.innerText = "Das ist der Text im Div";

// Element an bestimmtes element hängen
document.querySelector('#ziel').appendChild(newElement);
```

DOM Elemente entfernen

```
// Ein bestimmtes Element entfernen, ohne den Elternknoten zu kennen
var node = document.querySelector("li");
if (node.parentNode) {
  node.parentNode.removeChild(node);
}
```

Klasse hinzufügen (classList)

```
// Element wird erzeugt
var newElement = document.createElement('div');
newElement.classList.add('klassen-name');
document.body.appendChild(newElement);

// Mit bestehendem Element
var title = document.querySelector('h1');
title.classList.add('klassen-name');
```

Klasse entfernen (classList)

```
// Element wird erzeugt
var newElement = document.createElement('div');
newElement.classList.remove('klassen-name');
document.body.appendChild(newElement);

// Mit bestehendem Element
var title = document.querySelector('h1');
title.classList.remove('klassen-name');
```

Klasse "togglen" (classList)

```
// Element wird erzeugt
var newElement = document.createElement('div');
newElement.classList.toggle('klassen-name');
document.body.appendChild(newElement);

// Mit bestehendem Element
var title = document.querySelector('h1');
title.classList.toggle('klassen-name');
```

Attribute von Elementen setzen und abrufen

```
var link = document.createElement('a');
link.innerText = 'Google';
link.setAttribute('href', 'http://www.google.ch');
document.querySelector('body').appendChild(link);

console.log(link.getAttribute('href')); // http://www....
```

Werte von Eingabefeldern lesen & setzen

```
var inputElem = document.querySelector('input');

console.log(inputElem.value); // "5000"

inputElem.value = 1337;
```

Einbettung (inline)

In <head> (schlecht)

```
<html>
  <head>
    <title>JS Repetition</title>
    <script>
       // Javascript kommt hier
    </script>
  </head>
  <body>
    <h1 id="titel">Titel</h1>
  </body>
</html>
```

Ende <body> (gut)

```
<html>
  <head>
    <title>JS Repetition</title>
  </head>
  <body>
    <h1 id="titel">Titel</h1>
    <script>
       // Javascript kommt hier
    </script>
  </body>
</html>
```

Übungen zum DOM

Kopiert aus dem Google Drive Datei aus:
Module/Javascript/lektion3.zip in euer **htdocs** Verzeichnis in einen eigenen Ordner und macht:

- uebung_1_0.html
- uebung_1_1.html
- uebung_1_2.html
- uebung_1_3.html
- uebung_1_4.html

Javascript

Events

WEB PROFESSIONALS

Events

- Events sind "Dinge" die plötzlich geschehen
  - z.B. Benutzer klickt auf einen Button
- JavaScript kann darauf reagieren
- Machen Javascript interaktiv
- Können vom Benutzer aber auch von aussen ausgelöst werden


BALL

Beispiele Events

change	Wert auf ein Eingabefeld wurde geändert
click	Ein Element wurde angeklickt
mouseover	Der Mauszeiger wurde über ein Element geführt
mouseout	Der Mauszeiger hat das Element wieder verlassen
keydown	Eine Taste auf dem Eingabefeld wurde gedrückt
keyup	Eine Taste auf dem Eingabefeld wurde losgelassen

Event Demo

Inline Events (Sehr Schlecht)

```
<button id="myButton" onclick="doSomething()"></button>


<script>
function doSomething() {
    console.log('button clicked');
}
</script>
```

Event Handlers (Suboptimal)

```
<button id="myButton"></button>


<script>
function doSomething() {
    console.log('button clicked');
}
var myButton = document.querySelector('#myButton');
myButton.onclick = doSomething;
</script>
```

Event Listeners (Ideal)

```
<button id="myButton" onclick="doSomething()"></button>
<script>
function doSomething() {
    console.log('button clicked');
}
var myButton = document.querySelector('#myButton');
myButton.addEventListener('click', doSomething);
</script>
```

Übungen zum Events

Kopiert aus dem Google Drive Datei aus:
Module/Javascript/lektion3.zip in euer **htdocs** Verzeichnis in einen eigenen Ordner und macht:

- uebung_2_0.html
- uebung_2_1.html
- uebung_2_2.html
- uebung_2_3.html
- uebung_2_4.html