Webentwicklung

Frontend: JavaScript & jQuery

Inhalt dieser Einheit

1. Einführung
2. Syntax
3. Laufzeitumgebungen & DOM-API
4. Events
5. Objekte und “Klassen”
6. Weitere APIs
7. Debugging

JavaScript-Einführung

JavaScript

- eine der weltweit populärsten Programmiersprachen.
  - Tiobe Index: Platz 6 (Oktober 2017)
- mind. eine JavaScript-Umgebung auf jedem PC
- *die* Skriptsprache des Webs
  - aber auch für andere Zwecke

Geschichte

- Von Brendan Eich für Netscape entwickelt
- Aus Marketinggründen “Java” im Name
  - nur syntaktische Ähnlichkeiten
- interpretiert, single-threaded, schwach typisiert, dynamisch, funktional und objektorientiert
  - Vererbung über Prototypen (statt Klassen)
- Literale für Arrays und Objekte
- Läuft im Browser in Sandbox

Namen und Versionen

- **Mocha** und **LiveScript**, Netscape (vor “JavaScript”)
- **JScript**, Microsoft
- **ECMAScript**, European Computer Manufacturers Association, Standards
  - 2009, ECMAScript 5: “strict mode”
  - 2015, ECMAScript 2015 (“ES6”): Module, Arrow-Functions, Promises, …
  - 2017: ECMAScript 2017: Nebenläufigkeit, …

Welche Version?

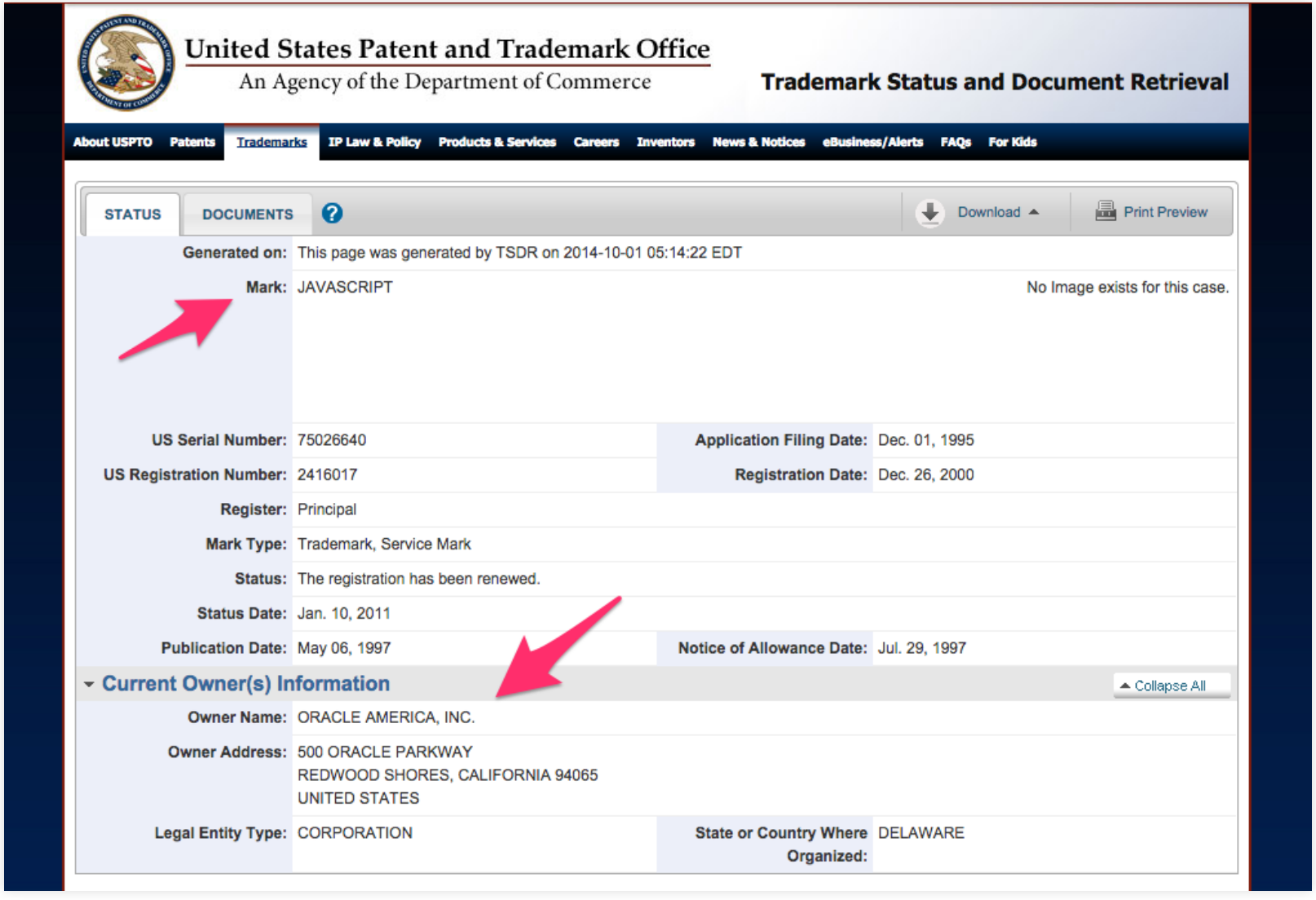
- ECMAScript-Unterstützung von Browsern
  - Daten von http://kangax.github.io/compat-table

Version	IE 11	Edge 16	FF 56	CH 62
5	99%	100%	100%	98%
6	11%	96%	97%	97%
2016+	3%	91%	100%	100%

- Marktanteil IE (Desktop): global 7-15%
  - Also: Kein Verlassen auf ES6-Features
    - (Transpiler wie Babel können helfen)
  - Diese Einheit: *kein* ES6

Quelle: https://en.wikipedia.org/wiki/Usage_share_of_web_browsers#Summary_tables

Geschichte



Quelle: http://tarr.uspto.gov/servlet/tarr?regser=serial&entry=75026640

Geschichte

> *The by-design purpose of JavaScript, was to make the monkey dance when you moused over it. Scripts were often a single line. We considered ten line scripts to be pretty normal, hundred line scripts to be huge, and thousand line scripts were unheard of. The language was absolutely not designed for programming in the large, and our implementation decisions, performance targets, and so on, were based on that assumption.*

Quelle: Eric Lippert

jQuery

- freie JavaScript-Bibliothek (Werkzeugsammlung)
  - gleicht außerdem Browser-Unterschiede und -Fehler aus
- Meistverwendete Bibliothek
  - auf über 95% der Websites mit JavaScript
- Umfang:
  - DOM-Navigation, -Manipulierung
  - Event-Behandlung
  - Ajax
  - Helfer (z.B. `each`)
  - Animationen
- Reichhaltiges Plugin-Ökosystem

Quelle: https://w3techs.com/technologies/overview/javascript_library/all

Syntax

Kontrollstrukturen

```
// Kommentar
/* Kommentar */
```

```
if      (1 === '1') { alert('Nope'); }
else if (1 == '1')  { alert('Yep');  }
else                { alert('Nope'); }
```

```
while (bedingung) { anweisungen; }

do { anweisungen; } while (bedingung);

for (var i = 0; i < 100; i++) { anweisungen; }
```

Variablen

```
var eins = 1; /* global */
zwei = 2;     /* global */

function x() {
  var drei = 3; /* lokal in Funktion x */
  vier = 4;     /* global */
  fuenf = 5;    /* lokal, wg. Hoisting */
  var fuenf;
}
x();
```

- Variablen haben Gültigkeitsbereich
  - `var` deklariert Variable in aktueller Funktion, bzw. global
  - undeklarierte Variablen sind immer global

Datentypen

- Eingebaute Typen
  - `Number` (immer Fließkomma, keine Ganzahlen)
  - `String`
  - `Boolean`
  - `Function`
  - `Object`
- Automatische Umwandlung
  - `"1" + 2 + 3; // "123"`
  - `1 + 2 + "3"; // "33"`
- Prüfen auf Typgleichheit
  - `1 == "1"; // true`
  - `1 === "1"; // false`

Funktionen

```
function hallo (name) {
   alert('Hallo' + name + '!');
}

var plus = function (a, b) { return a + b; }
plus(1, 2); // 3

'abcdefg'.charAt(2).toUpperCase(); // 'C'
'abcdefg'['charAt'](2)['toUpperCase'](); // 'C'
```

Objekte / Arrays

```
var brot = { // Object
   name: "Matze", // Property
   zutaten: [ // Array
      { "name": "Mehl", "menge": 2 },
      { name: "Wasser", menge: 1 }
   ],
   getRezept: function () { // Method
      return this.zutaten[0].name +
         " + " + this["zutaten"][1]["name"] +
         " = " + this.name;
   }
};

// member operator
brot.name === brot["name"];
```

Ausführung von JavaScript

Im Browser und anderswo

Ausführung/Einbindung von JS

1. Im Webbrowser
    1. Als eingebundene Ressource
    2. Inline in `script`-Elementen
    3. Als Link-Ziel
        - Bookmarklets
    4. Über Nutzerskripte
    5. Console der Developer Tools
2. In separater Laufzeitumgebung
    - Node.js

1. Webbrowser: Ressource & inline

```
<!-- Ja -->
<script src="script.js"></script>

<!-- Nein -->
<script type="text/javascript" language="JavaScript" src="script.js">

<!-- HTML5 -->
<script src="script.js" async defer></script>
<!-- defer (HTML 4): erst wenn DOM fertig -->
<!-- async (HTML 5): parallel neben DOM-Erstellung -->

<!-- Inline -->
<script>
   confirm('Ok?');
</script>

<noscript> ....Normaler HTML-Code... </noscript>
```

1. Webbrowser: Als Linkziel

```HTML
<a
  href="javascript:var b = 2 * 4; alert('Hallo Welt ' + b)"
>Klick</a>
<a
  href="javascript:function a(b){return b*2;}; alert(a(5))"
>mich</a>
```

Klick mich

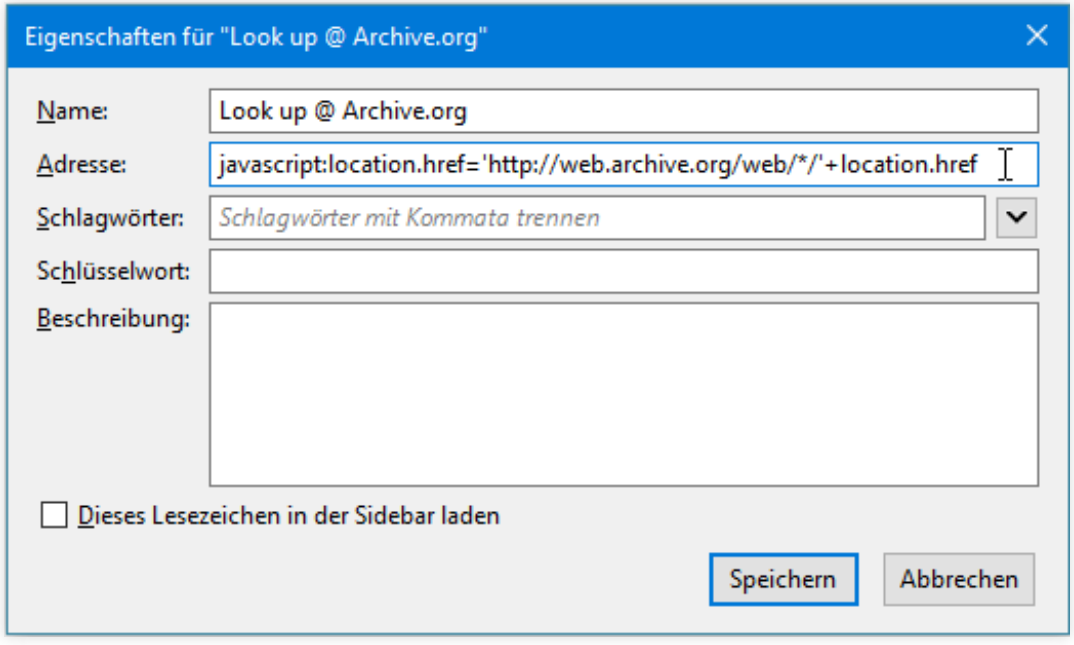
- `javascript:` ist ein Pseudoprotokoll
  - Verhalten wie normale Links
  - Code wird bei Klick ausgeführt

1. Webbrowser: Bookmarklet

- Links lassen sich als Lesezeichen speichern
  - Beispiel: Die aktuelle Adresse bei Archive.org nachschlagen

```
location.href='http://web.archive.org/web/*/'+location.href
```

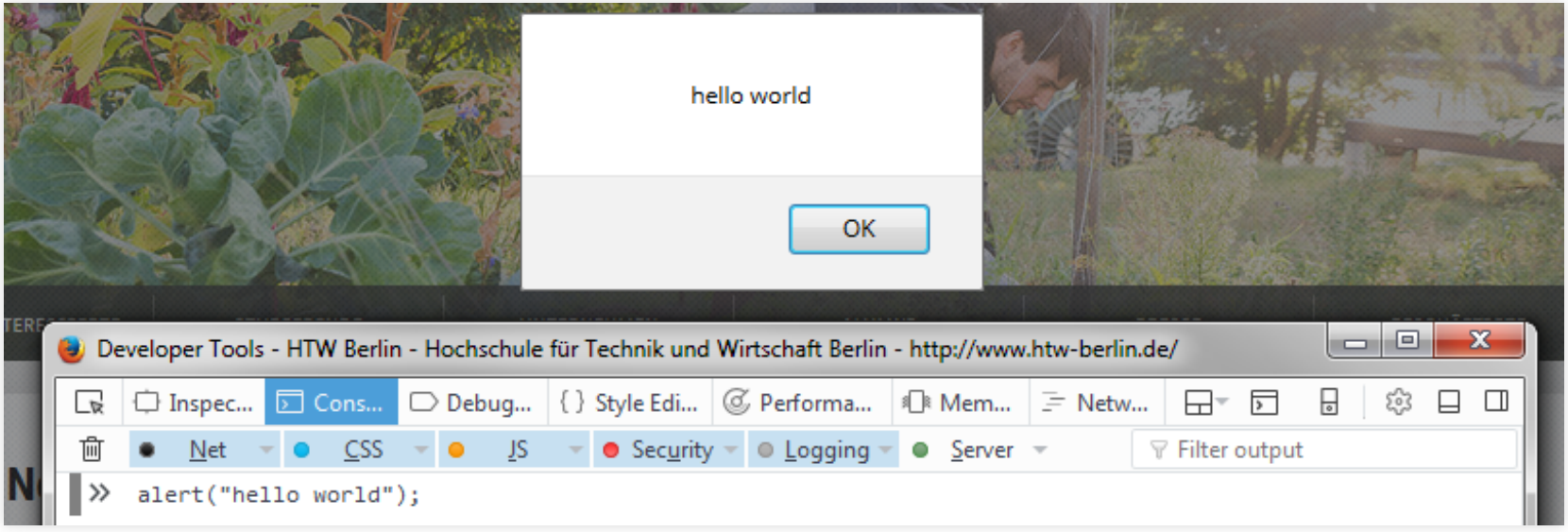
- Bookmarklet: Als Makro in die Lesezeichenleiste



1. Webbrowser: Nutzerskripte

- AddOns wie **Greasemonkey**
  - führt lokal hinterlegte JavaScript-Skripte bei bestimmten Adressen aus
- Beispiele:
  - Aussehen von Webseiten verändern (Teile ausblenden, vergrößern, verschieben)
  - Standardauswahl bei häufig benutzten Formularen ändern
- Weitere AddOns und Skripte:
  - https://greasyfork.org/de
  - https://openuserjs.org/

1. Webbrowser: Console



2. Laufzeitumgebung

- Node.js: https://nodejs.org/de/
  - Interaktiv

```
$> node
> function add(a,b) { return a+b; }
undefined
> add(6,7);
13
```

  - Skripte ausführen

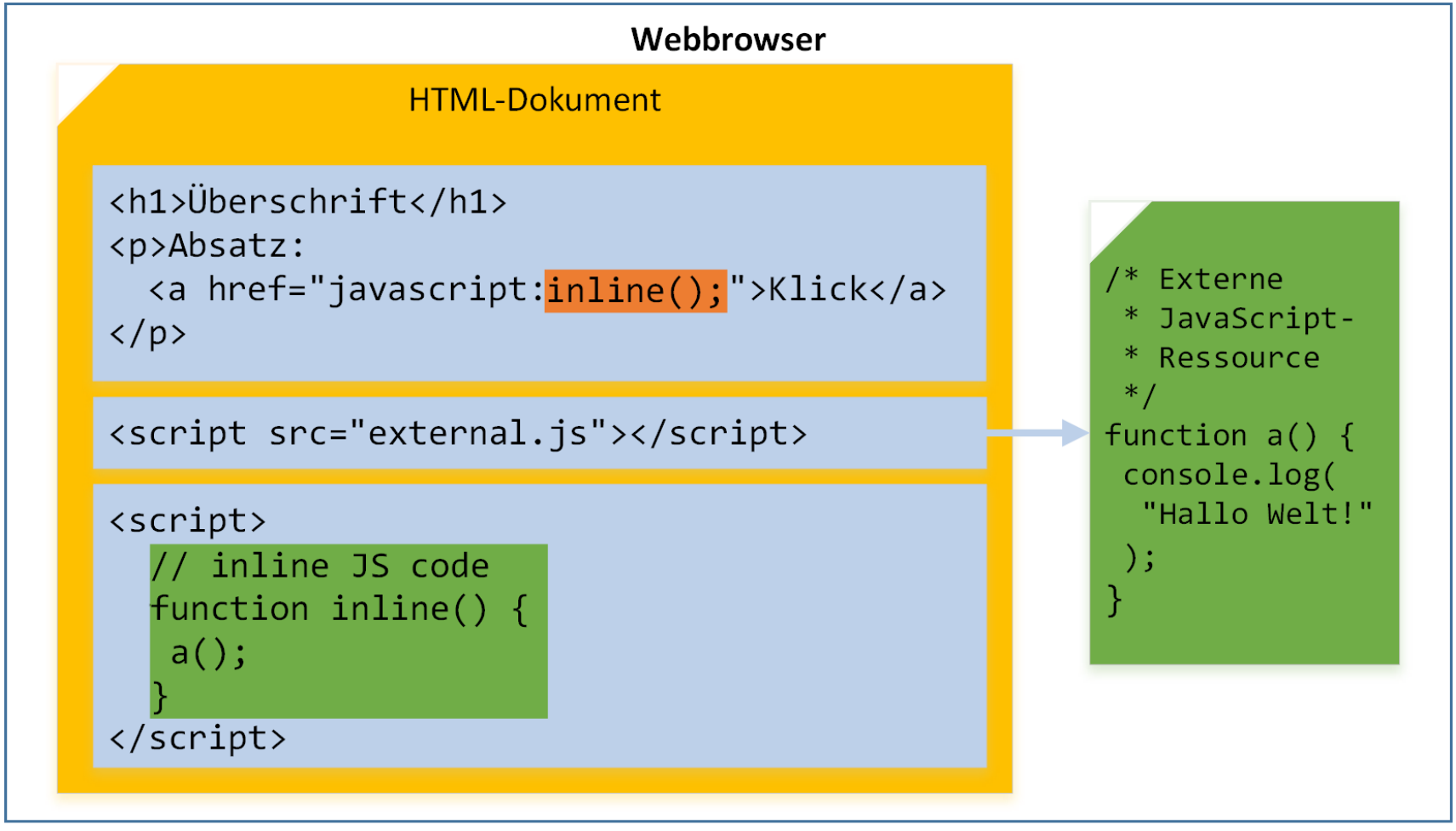
```
function add(a,b) { return a+b;}
console.log(add(56, 15));
```

```
$> node calc.js
71
```

- hier: erstmal nur Webbrowser
  - kommt wieder bei Einheit Optimierung

Browser als JavaScript-Engine

- grün: Ausführen beim Laden des Dokuments
- orange: Ausführung bei Klick



DOM: Überblick

- DOM: API, über die JavaScript mit Website interagiert
  - (Erinnerung: HTML-Dokument darstellbar als Baum)
  - DOM erlaubt Zugriff, Traversierung und Manipulation
- Globales Objekt: `window`
  - darin das Dokument: `window.document`

DOM: `window`

- `window` ist das aktuelle Fenster, das ein Dokument beeinhaltet
  - https://developer.mozilla.org/en-US/docs/Web/API/Window

```
window
    .location
    .history.forward() / .back()
    .alert("Et boum!")
    .confirm("Ok?")
    .prompt("Ja?", "Ja!")
    .open("http://google.de");
```

DOM: `document`

- Aktuelles Dokument über `window.document`
  - auch direkt über `document`
  - https://developer.mozilla.org/en-US/docs/Web/API/document

```
document
   .head / .body
   .contentEditable = true
   .links[0].href
   .images
   .cookie // delete durch expires in Vergangenheit
   .createElement()
   .get*
   .querySelector*
   .[Formularname][Inputname].value
```

DOM: `Node`

- `document`, `document.head`, `document.body`, und alle HTML-Elemente implementieren `Node`-Interface
  - https://developer.mozilla.org/en-US/docs/Web/API/Node

```
aNode.childNodes // direkte Kinder
aNode.lastChild
aNode.firstChild === anyNode.childNodes[0]
aNode.parentNode
aNode.nextSibling
aNode.previousSibling
aNode.nodeValue
aNode.getAttribute(someString)
```

Dokument und DOM

Webbrowser

HTML-Dokument

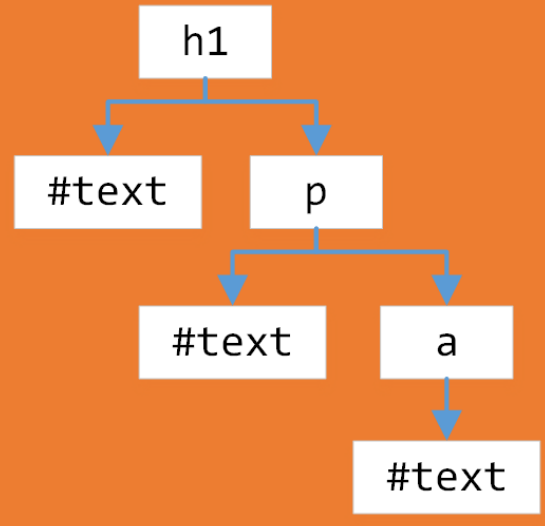
```
<h1>Überschrift</h1>
<p>Absatz:
  <a href="javascript:inline();">Klick</a>
</p>
```

```
<script src="external.js"></script>
```

```
<script>
  // inline JS code
  function inline() {
    a();
  }
</script>
```

DOM

```
window
document
...
```



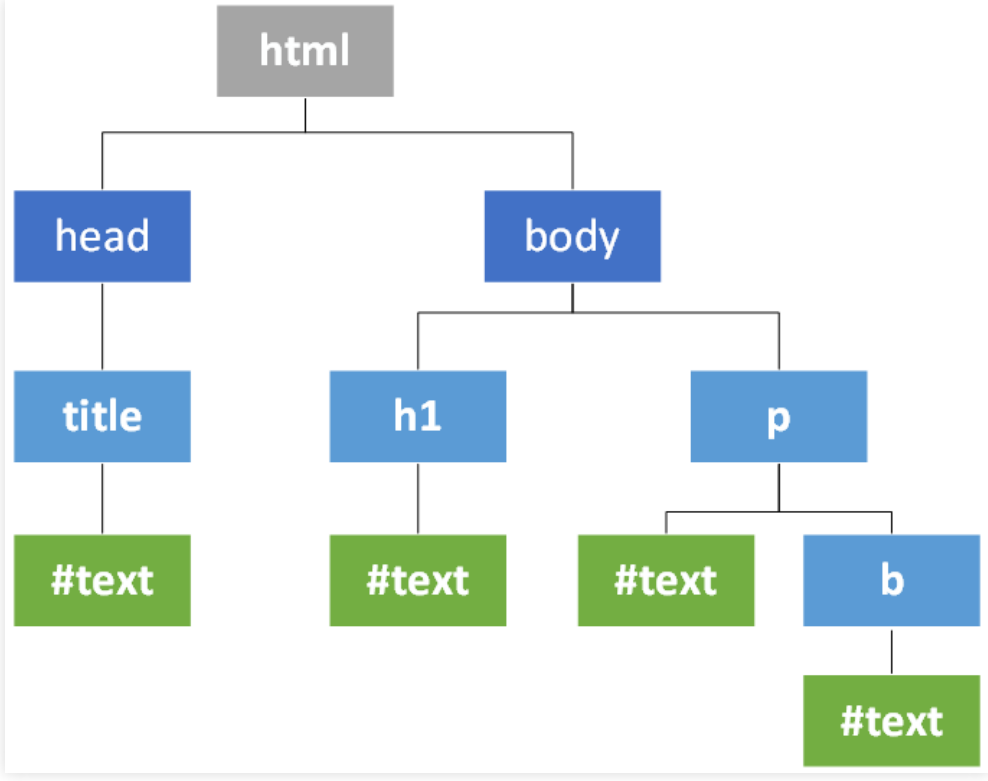
DOM: Traversieren & Manipulieren

- Inhalt des Knoten unten rechts auf “Hi!” setzen, z.B.

```
document
.getElementsByTagName('p')[0]
.childNodes[1]
.textContent="Hi!";
```

- Eleganter: Auswahl mit CSS-Selektor

```
p > b
```

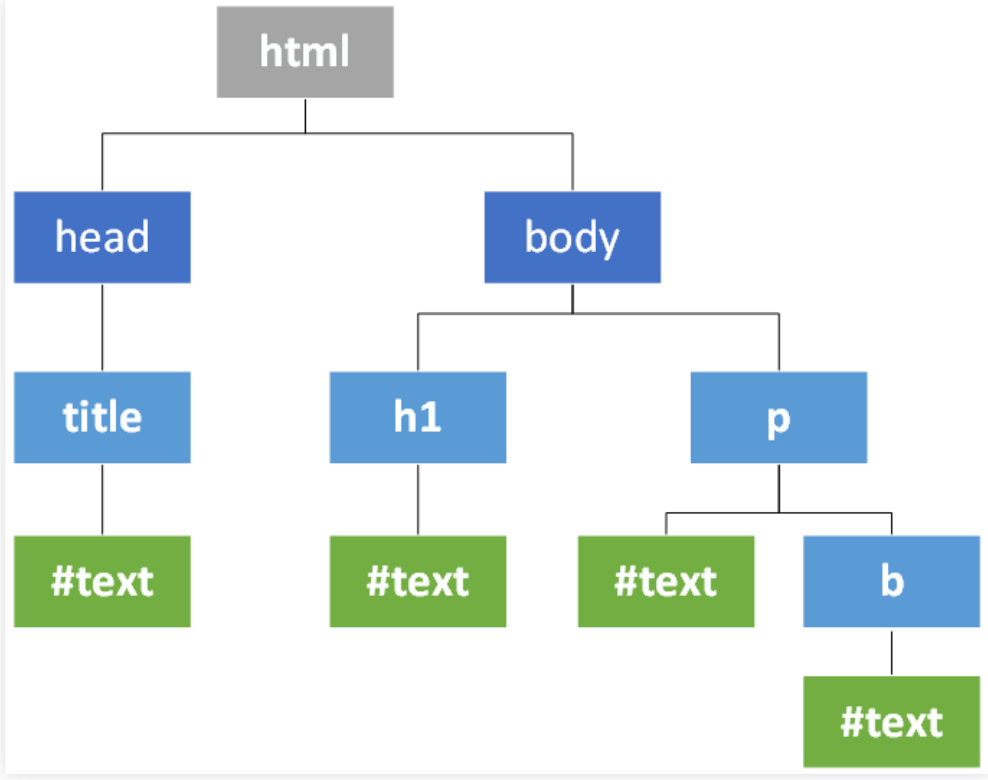


jQuery: Traversieren

- Mit jQuery:

```
$('p > b').text('Hi!');
```

  - (Achtung: Wie CSS selektiert das alle `b`-Kinder von *allen* `p`-Elementen)



jQuery: Traversieren

```
/* Selektieren */
$('#id .klasse element[attr]');
$('#container .panel h1:first');

/* Mit Selektionen arbeiten */
$('.panel').find('h1').parent().eq(2).next().each(function(index) {
   console.log( index + ': ' + $(this).text() );
});
```

- “Fluent Interfaces” / “Chaining”
- Weitere Beispiele:
  - https://learn.jquery.com/using-jquery-core/selecting-elements/
  - https://learn.jquery.com/using-jquery-core/working-with-selections/

jQuery: Element-Manipulation

```
/* CSS-Eigenschaften setzen */
$('#container .panel h1:first').css('color', 'red');

$('.panel').toggleClass('active').empty().html('Stille');
```

- Weitere Beispiele:
  - https://learn.jquery.com/using-jquery-core/manipulating-elements/

Zusammenfassung bisher

- JavaScript im HTML-Dokument eingebunden
  - wird in Lesereihenfolge* ausgewertet
- Aus JavaScript heraus: DOM ansprechen
  - direkt über DOM-API
  - bequemer über jQuery und CSS-Selektoren
- Möglichkeiten
  - Traversierung (Bewegen im Baum, Auswahl von Elementen)
  - Manipulation (Ändern von Inhalten und Attributen)

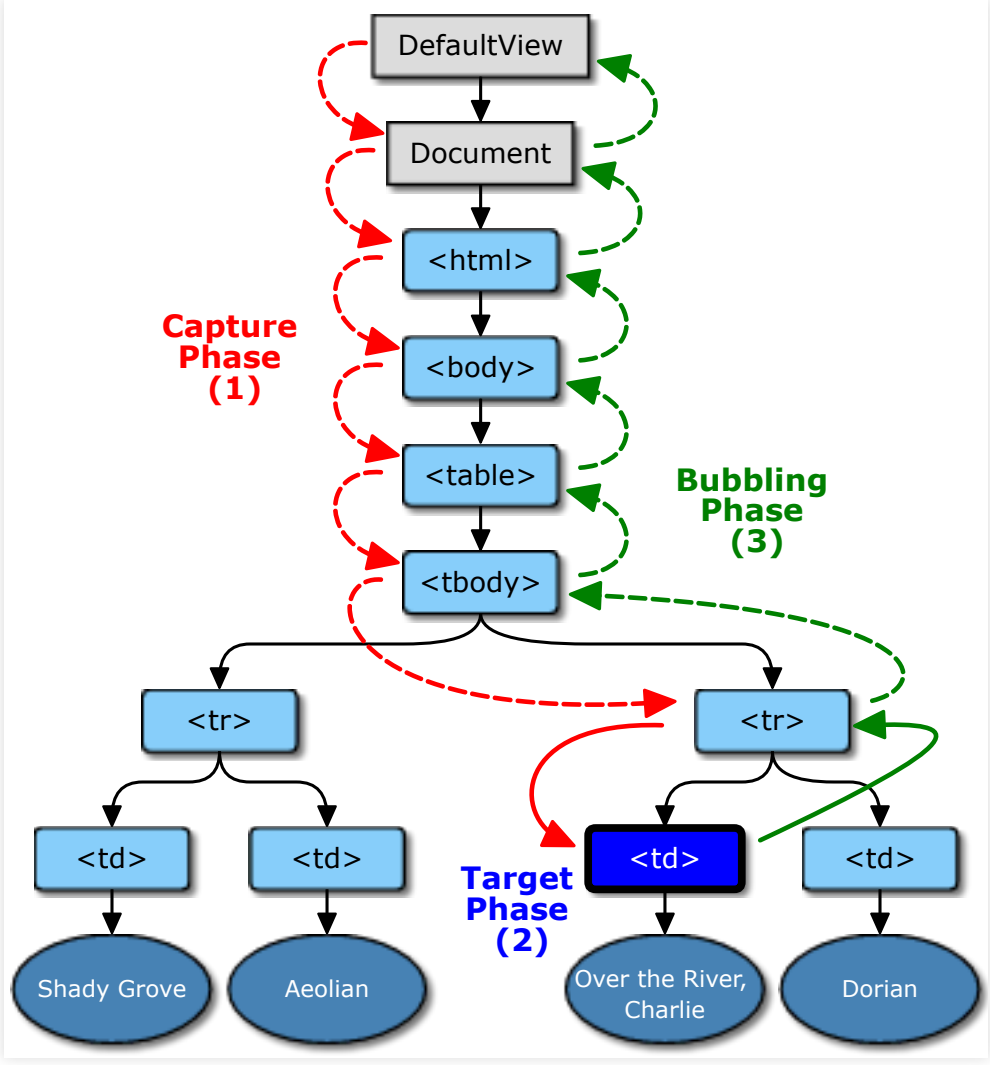
Ereignisse

Reagieren auf Ereignisse

- Nutzer-Interaktion: DOM kennt dutzende Events, z.B.

```
onabort     // bei Abbruch
onblur      // beim Verlassen
onchange    // bei erfolgter Änderung
onclick     // beim Anklicken
ondblclick  // bei doppeltem Anklicken
onerror     // im Fehlerfall
onfocus     // beim Aktivieren
onkeydown   // bei gedrückter Taste
onkeypress  // bei gedrückt gehaltener Taste
onkeyup     // bei losgelassener Taste
onload      // beim Laden einer Datei
onmousedown // bei gedrückter Maustaste
onmousemove // bei weiterbewegter Maus
onmouseout  // beim Verlassen des Elements mit der Maus
onmouseover // beim Überfahren des Elements mit der Maus
onmouseup   // bei losgelassener Maustaste
onreset     // beim Zurücksetzen des Formulars
onresize    // bei Größenänderung des Fensters
onselect    // beim Selektieren von Text
onsubmit    // beim Absenden des Formulars
onunload    // beim Verlassen der Datei
```

Capturing & Bubbling



Quelle: w3.org/TR/DOM-Level-3-Events

Bubbling: Beispiel

```HTML
<div onClick="this.style.backgroundColor='darkblue'">
  <button onClick="this.innerText='Nacht'">Tag</button>
</div>
```

- In Bubbling-Phase:
  1. `button.onClick`
  2. `div.onClick`
- Achtung: Nicht alle Events bubblen (z.B. `focus`/`blur`)

Bubbling beeinflussen

```
// ein Listener
element.onclick = function () { this.innerText = 'Nacht'; }

// mehrere Listener
element.addEventListener('click', function (event) {
   this.innerText = 'Nacht';
   event.stopPropagation(); // stop bubbling up
   event.preventDefault(); // bubble, but prevent default action
});
```

- Beispiele für Default-Actions
  - Klick auf Checkbox: Ankreuzen
  - Tippen in Input-Feld: Buchstaben eintragen
- Mehr:
  - https://developer.mozilla.org/en-US/docs/Web/API/Event/preventDefault

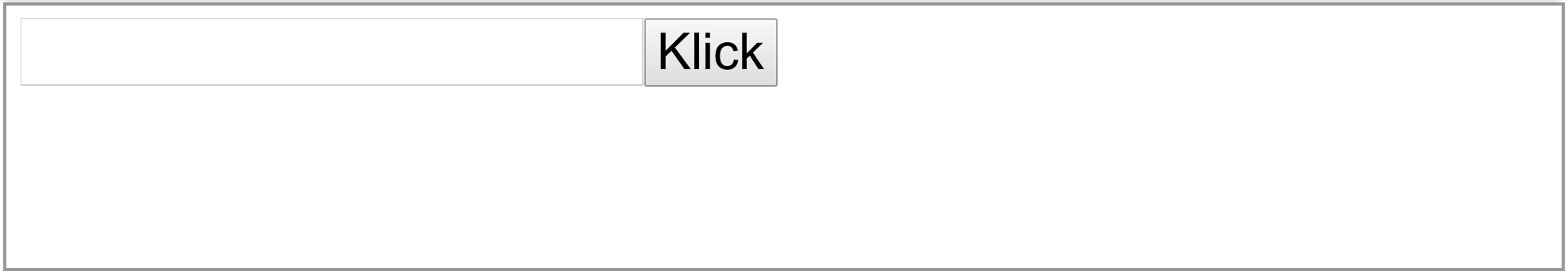
jQuery: Events

```js
$('.btn').on('click', function(e) {
   if (Math.random() > 0.5)
   { $('input').val(''); alert('Hupsi ...'); }
});

$('.message').on('keyup', function(e) {
   $('.counter').html(this.value.length + ' Zeichen');
});
```

JS

```html
<input class="message"></input><button class="btn">Klick</button>
<span class="counter"></span>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></scri
```

HTML



Capturing-Event-Listener?

- jQuery: nur Bubbling (buttom-up, Element -> Eltern)
- DOM-API kennt beides:

```
aNode.addEventListener(event, myListener); // bubbling
aNode.addEventListener(event, myListener, false); // bubbling
aNode.addEventListener(event, myListener, true); // capturing
```

- Capturing: top-down, von Wurzel zu Element

Objekte und "Klassen"

Objekte – Literal

```
var person = {
   name: 'Alice',
   alter: 42
}
```

Objekte – Konstruktor-Funktion

```
function Person(name, alter) { // Konstruktur-Funktion
   this.name = name;
   this.alter = alter;
}
function introduce(person) {
   return "Hallo, mein Name ist " + person.name +
          " und ich bin " + person.alter + " Jahre alt.";
}

var alice = new Person('alice', 42);
// new: neues, leeres Objekt (`this`)
// Person(...): Aufruf der Funktion auf diesem Objekt
alice.alter == 42; // true
introduce(alice);
```

Objekte – Konstruktor-Funktion

```
function Person(name, alter) {
   this.name = name;
   this.alter = alter;

   this.introduce = function() {
      return "Hallo, mein Name ist " + this.name +
             " und ich bin " + this.alter + " Jahre alt.";
   }
}

var alice = new Person('Alice', 42);
alice.introduce();
```

Objekte – Konstruktor-Funktion

```
function Person(name, alter) {
    this.name = name;
    this.alter = alter;
}

// Veränderung des Prototypen
Person.prototype.introduce = function() {
    return "Hallo, mein Name ist " + this.name +
           " und ich bin " + this.alter + " Jahre alt.";
}

var alice = new Person('Alice', 42);
alice.introduce();
```

Revealing Module Pattern

```
function Person(name, alter) {
   // Gib ein Objekt mit diesen Eigenschaften zurück
   return {
      name: name,
      alter: alter
   };
}

// hier kein `new`
var alice = Person('alice', 42);
alice.alter == 42; // true
```

Privatsphäre von Objekten


```
function Person(name) {
    this.name = name;
    this._secret = 'sonnenschein';
    this.check = function(pwd) {
        return pwd === this._secret;
    };
}

var alice = new Person('alice');
alice.check('sommerregen');
  // false
alice.check(alice._secret);
  // true 😳 🙈
```

Revealing Interface

```
function Person(name, alter) {
   var secret = 'sonnenschein';
   function check (password) {
      return password === secret;
   }
   return { name, alter, check }; // ES6-Syntax
}

var alice = Person('alice', 42);
alice.check('sommerregen'); // false
alice.secret // undefined 😊 👏
// `secret` ist eine lokale Variable der Person-Funktion
```

Immediately Invoked Functions

```
var msg = "Ich bin global verfügbar";

// Definieren einer Variablen vom Typ function
(function() {
   var msg = "Ich nicht.";
   console.log(msg);
})(); // <-- sofortiger Aufruf

console.log(msg);
```

JavaScript APIs

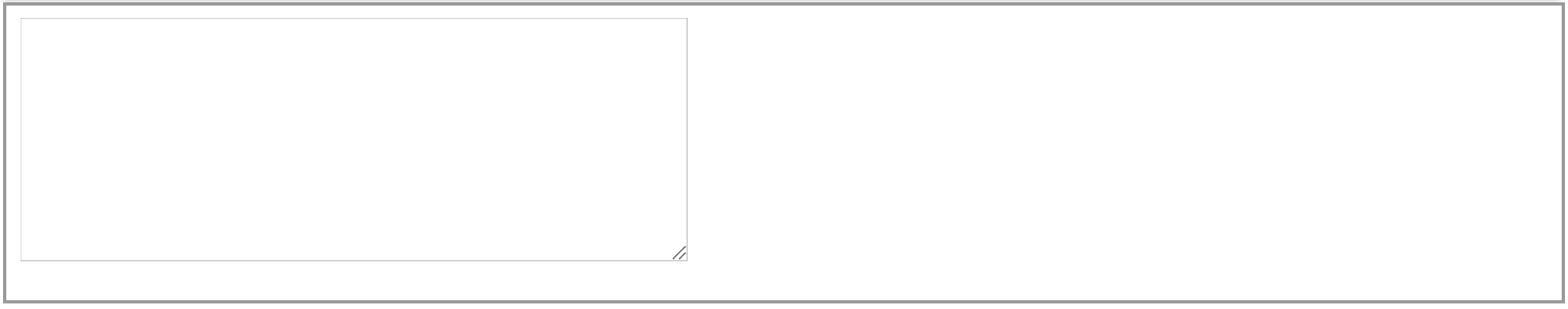
DOM Storage

- `sessionStorage` und `localStorage`
- clientseitiger Key-Value-Speicher (String -> String)
  - meist 5MB
- gut als Cache (z.B. Mails)
- Inter-Tab-Kommunikation (bei gleicher Domain)

```
var initProducts = [
  {id: 1, name: 'Kuchen'}, {id: 2, name: 'Auto'}
];
// save objects as JSON (a String)
localStorage.setItem('products', JSON.stringify(initProducts));

var products = localStorage.getItem('products');
var loadedProducts = JSON.parse(products);
```

LocalStorage API

```JS
var ta = document.querySelector('textarea');

ta.value = localStorage.getItem('backup') || '';

ta.onkeyup = function() {
  localStorage.setItem('backup', this.value);
}
```



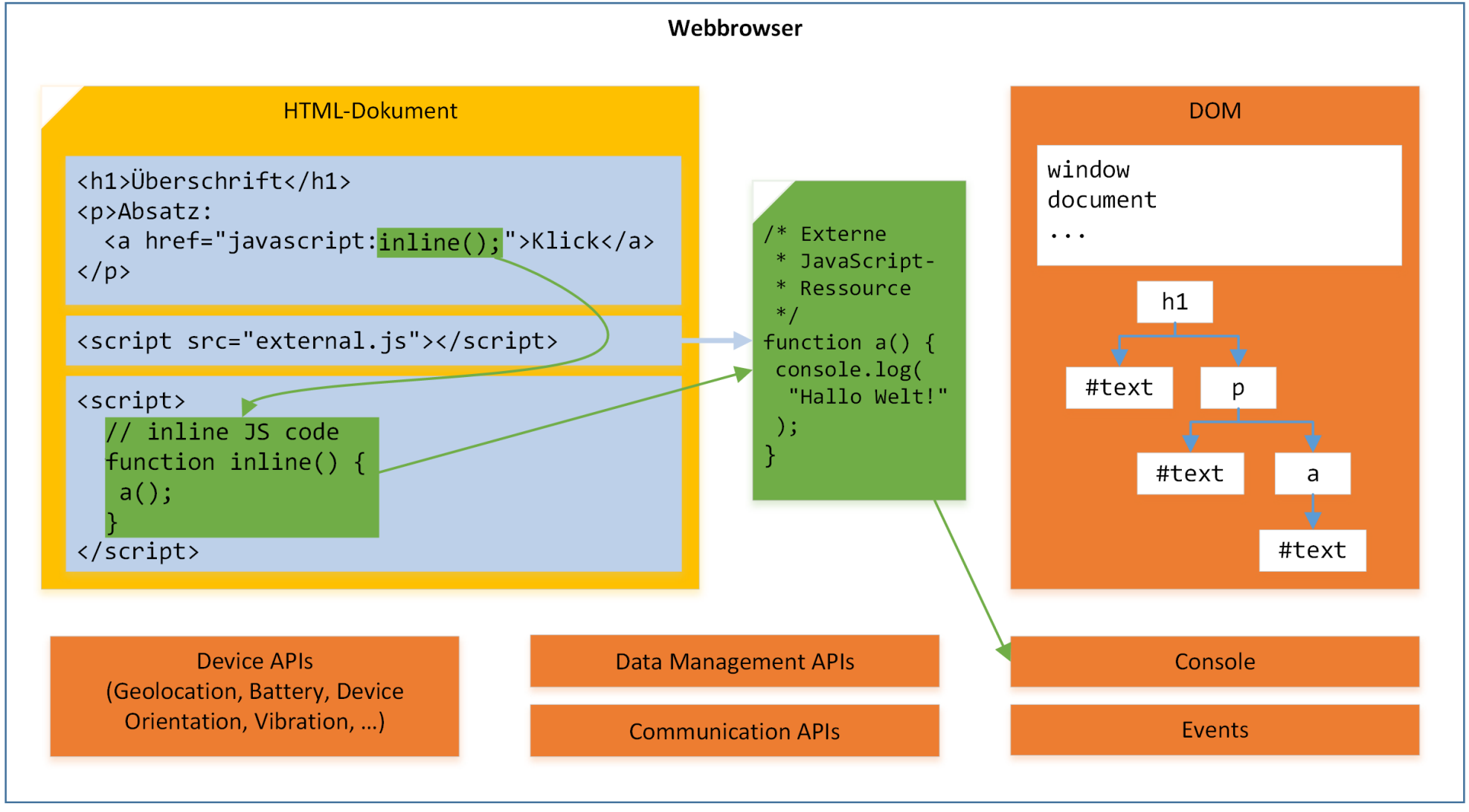
Geolocation API

```JS
var out = document.querySelector('pre');
var success = function (position) {
  var msg = 'latitude: ' + position.coords.latitude + ', longitude: '
  out.innerHTML += msg + '<br />';
};

// navigator.geolocation.watchPosition(success);
```

Notification API

```JS
var button = document.querySelector('button');
var input = document.querySelector('input');

var notify = function() {
  var notification = new Notification('Hallo', { body: input.value })
  notification.onclick = function() { alert('Hihi'); }
};

button.onclick = function(){
  if (Notification.permission === 'granted') notify();
  else if (Notification.permission !== 'denied') {
    Notification.requestPermission(function (permission) {
      if (permission === 'granted') notify();
    });
  }
};
```

Überblick


Webbrowser
HTML-Dokument
<h1>Überschrift</h1>
Absatz:
Klick

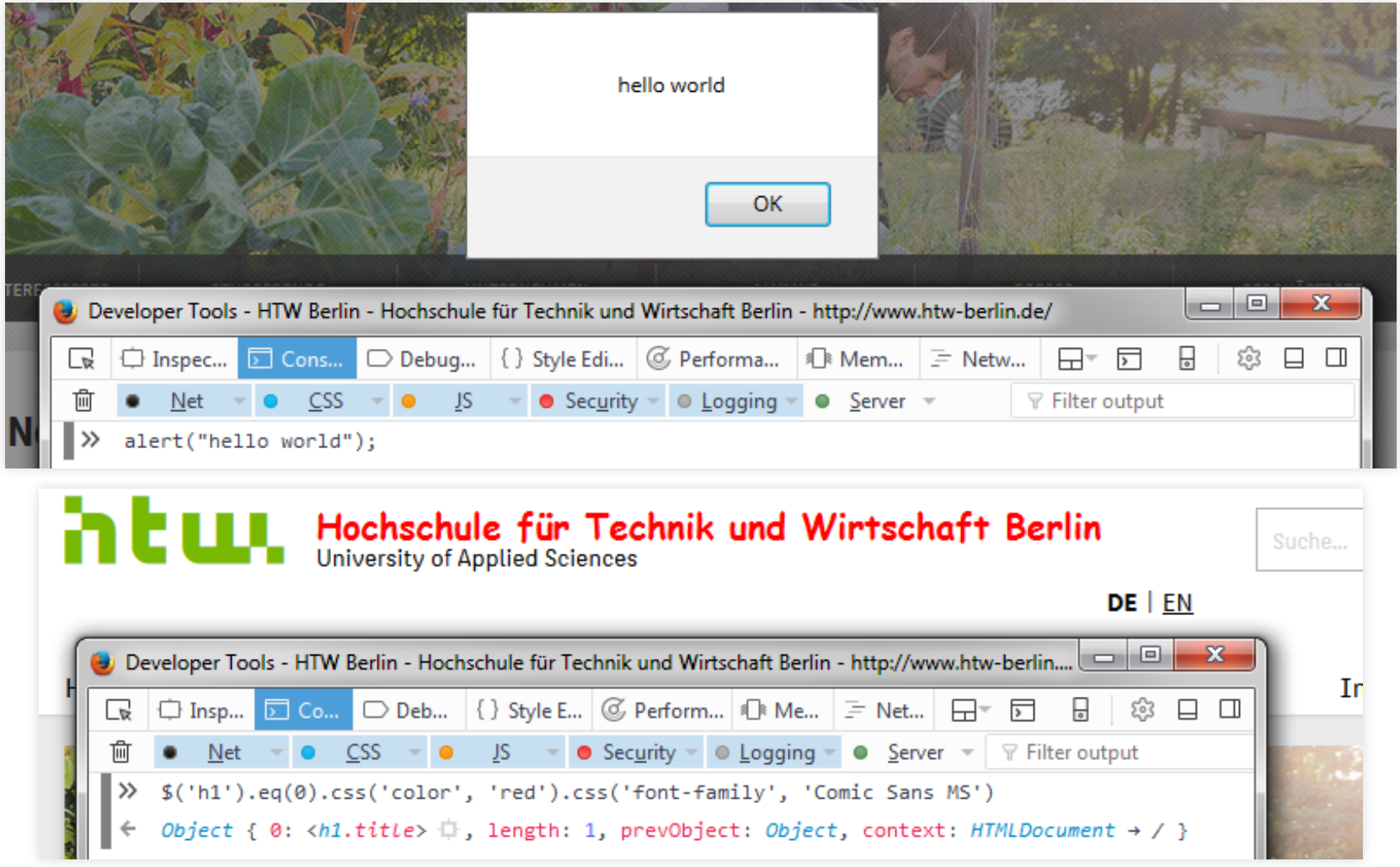
<script src="external.js"></script>
<script>
// inline JS code
function inline() {
a();
}
</script>
/* Externe
* JavaScript-
* Ressource
*/
function a() {
console.log(
"Hallo Welt!"
);
}
DOM
window
document
...
h1
#text
p
#text
a
#text
Device APIs
(Geolocation, Battery, Device
Orientation, Vibration, ...)
Data Management APIs
Communication APIs
Console
Events

Links

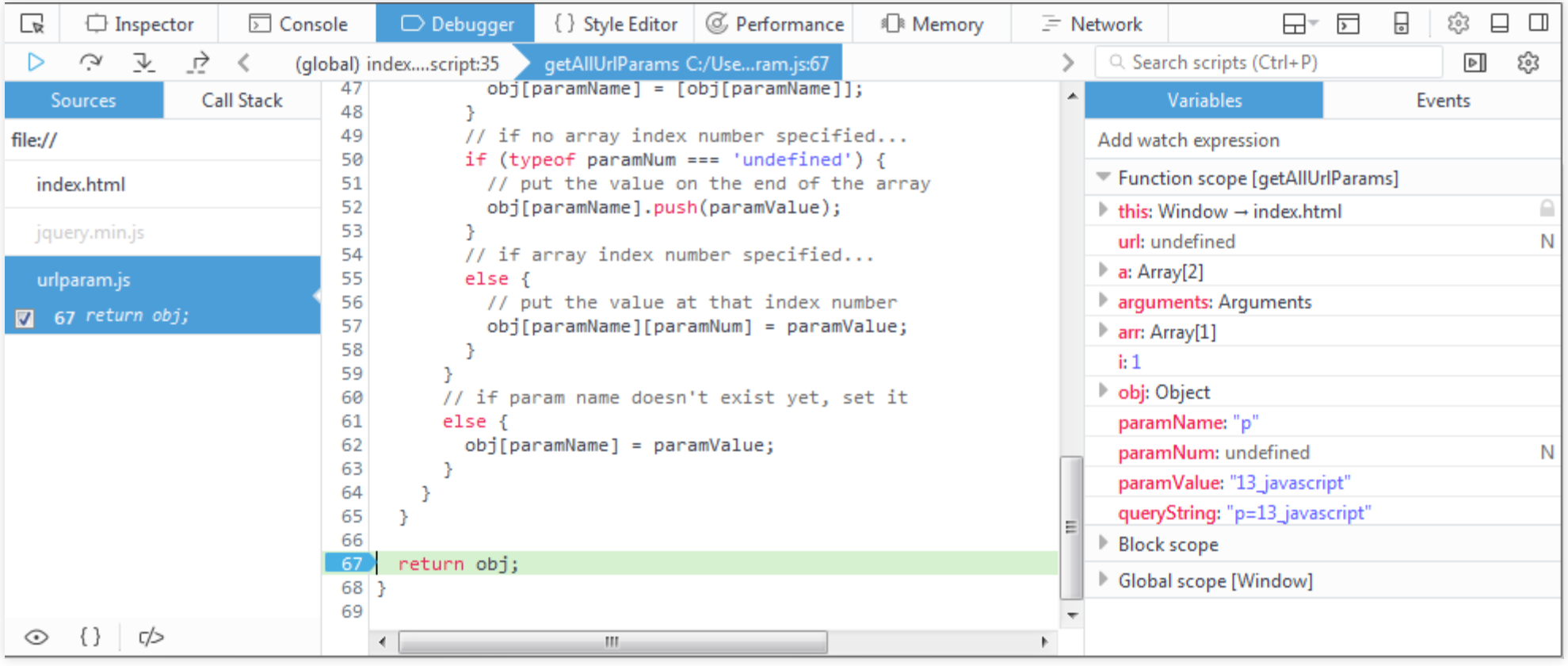
- JavaScript:
  - Mozilla Developer Network - DOM
  - Mozilla Developer Network - Web APIs
  - SelfHTML - JavaScript / DOM
- jQuery
  - try.jquery.com
  - learn.jquery.com
  - api.jquery.com
  - devdocs.io/jquery

Debugging

Debugging-Hinweise: F12



Debugger mit Breakpoints



Zusammenfassung: Heutige Inhalte

- JavaScript-Syntax
- DOM-Manipulation
- Objekt-Orientierung in JavaScript
- Event-System
- Browser-APIs

Danke!