

# **BROWSERAPIS 1**

**MMWP2024 - LV05**

## INHALTSVERZEICHNIS

- Organisation
- Javascript und DOM
- Ereignismanagement
- Vorstellung einzelner APIs

## INHALTSSCHWERPUNKTE

- Einführung in Javascript und TypeScript
- Schnittstelle von Webseite und Browser/Computer
- Erklärung von Browser APIs

## VORAUSSETZUNG

Der Ausgangspunkt dieser Vorlesungsreihe ist das Wissen über Javascript und wie dieses vom Browser ausgeführt wird

- Javascript Interpreter
- Einsatzzweck von Javascript
- Grundlegendes Verständnis von Javascript

## ZIELE

- Kurze Erklärung von Javascript und Typescript
- Definition des DOM-Modells und Verwendungszwecke
- Erklärung von Eventmanagement in Javascript
- Vorstellung einiger Browser-APIs mit Javascript

## KURZE ANMERKUNG ZUM EINSATZ VON JAVASCRIPT IN HTML5

- JavaScript ist eine Skript-Programmiersprache und die dritte bedeutende Technologie neben HTML5 und CSS3, die weit von Browsern unterstützt und angewendet wird
- Für die Kommunikation zwischen HTML, JavaScript und CSS stehen APIs zur Verfügung (z.B. für die Übergabe von Instanzen und ihren Werten)
- [Liste aller aktuellen BrowserAPI](#)

## WEITERE VERWENDUNG VON JAVASCRIPT AUSSERHALB VON DES WEBKONTEXTES

- Node.js: serverseitige Plattform zum Betrieb von Netzwerkanwendungen (z.B. Webserver)
- JSON (JavaScript Object Notation): Datenformat in Textform zum Datenaustausch zwischen Anwendungen
- [Espruino](#): Mikrocontroller mit JavaScript

# JAVASCRIPT VARIABLEN

- Explizite Deklaration von Variablen vor ihrem Gebrauch durch strict mode (“use strict“;) erzwingen, um Fehler leichter zu finden
- Lokal definierte Variablen mit gleichem Namen wie globale Variablen haben im lokalen Kontext höhere Priorität (statische/lexikalische Bindung)

	Globale Variablen	Lokale Variablen
Definition	Außerhalb von Funktionsblöcken	Innerhalb eines Funktionsblocks
Geltungsbereich, Sichtbarkeit	Im gesamten Programm	Innerhalb desselben Funktionsblocks
Lebensdauer	Gesamtlaufzeit des Programms	Laufzeit des Funktionsblockaufrufs

Quelle

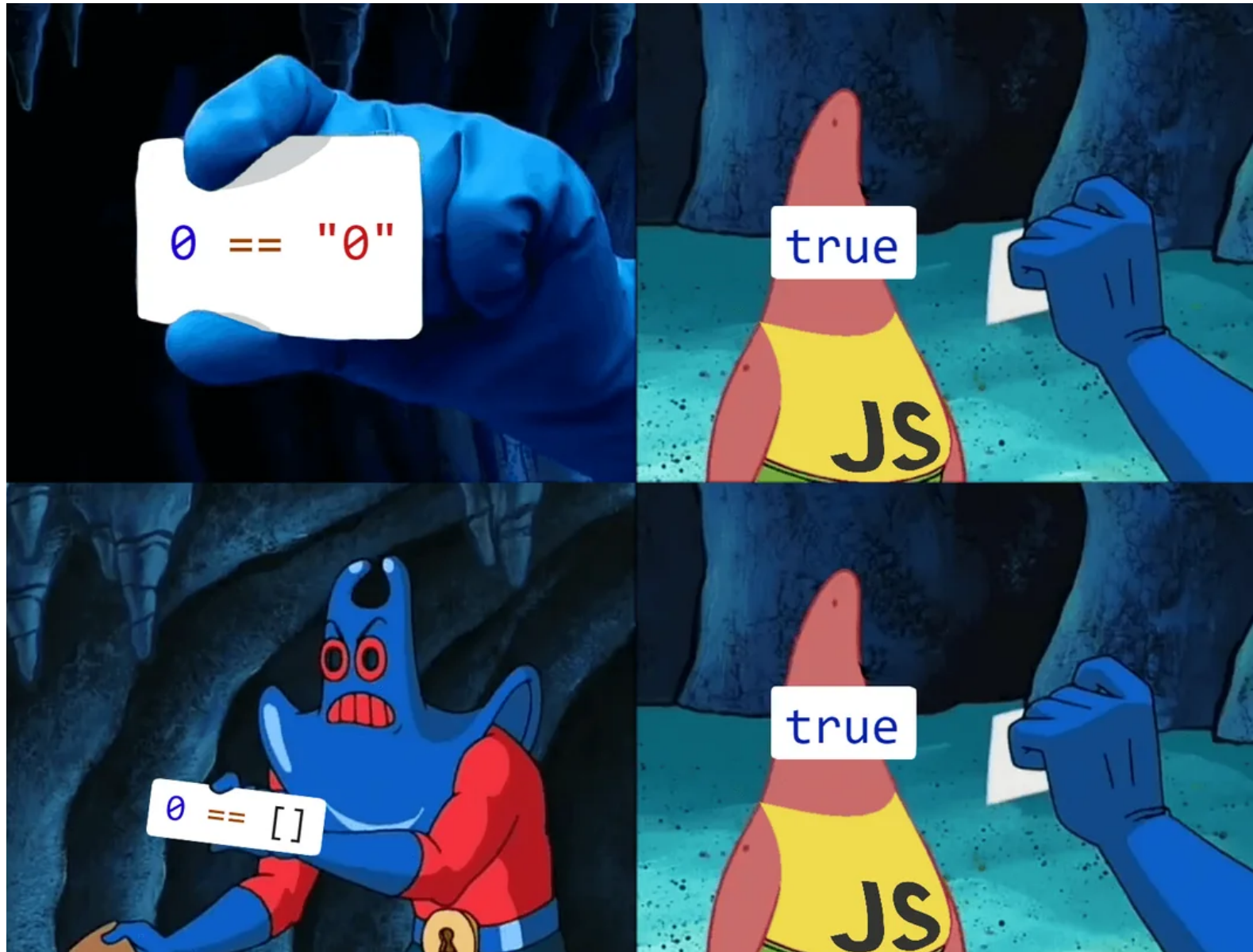


## JAVASCRIPT VARIABLEN - PROBLEME - 1

- JavaScript hat schwach getypte Variable, d.h. der Variablentyp kann sich zur Laufzeit verändern und ist ungenau bei Vergleich von Werten
- Die Typprüfung wird in JavaScript nicht schon bei der Kompilierung durchgeführt, sondern erst zur Laufzeit des Programms (dynamische Typisierung)

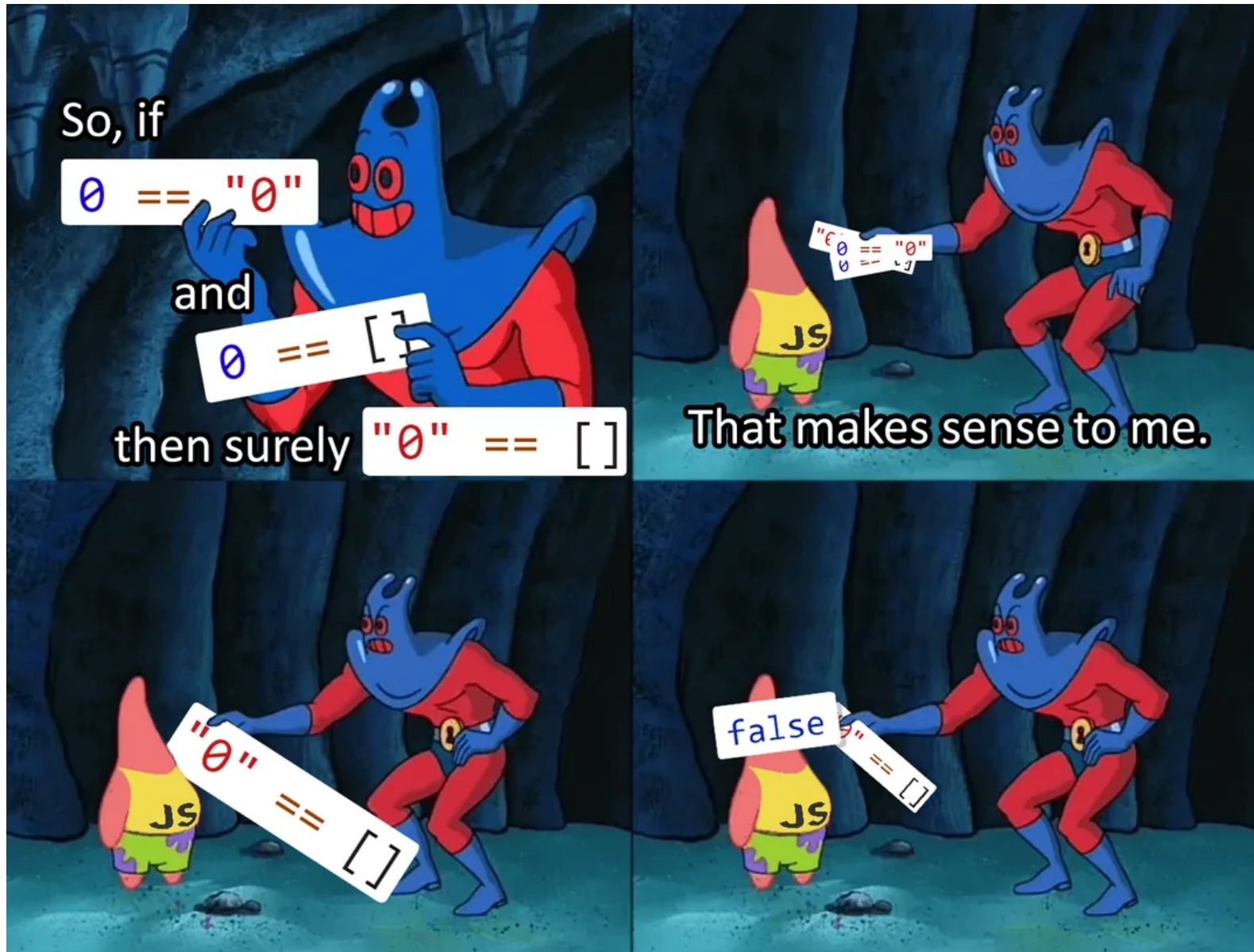
```
let x;  
x = 1;  
x = "1";  
x = false;
```

# JAVASCRIPT VARIABLEN - PROBLEME - 2



Quelle

# JAVASCRIPT VARIABLEN - PROBLEME - 3



Quelle

## JAVASCRIPT VARIABLEN

Um Vergleichprobleme zu vermeiden, sollte immer `===` benutzt werden, bei größer, kleiner gibt es weiterhin problematische Vergleiche:

```
0 > null -> false
0 >= null -> true
0 < null -> false
0 <= null -> true
```

## TYPESCRIPT ZU JAVASCRIPT

In Typescript, einer JavaScript dominierenden Skriptsprache mit strenger Typisierung, wird die Zuordnung anders gemacht:

```
let myVar: string;

myVar = 'Hello'; // Ok
myVar = 'World!'; // Also Ok
myVar = 42; // Not Ok: Type 'number' is not assignable to type 'string'
myVar = false; // Not Ok: Type 'boolean' is not assignable to type 'string'
myVar = null; // Ok
```

# FUNKTIONEN IN JAVASCRIPT

```
function find(n) //Funktionskopf mit Schlüsselwort 'function', Funkti
{ //Funktionskörper (Closure) in geschweiften Klammern
  let entry = "";
  return entry; //Wertrückgabe über return
}
```

- Aufruf mit geringerer Argumentenanzahl möglich
- Wertübergabe bei einfachen Datentypen (Variablen, Strings) durch call by value
- Bei Objekten (Funktionen, Felder) erfolgt Wertübergabe mittels call by reference

## **WEITERE VERWENDUNG VON JAVASCRIPT AUSSERHALB VON DES WEBCONTEXTES**

- Jscript.NET: leicht modifizierte Skriptsprache für Windows von Microsoft
- ActionScript: erweiterte Skriptsprache in Flash
- Standardprogrammiersprache von GNOME Shell
- NativeScript: Framework zur Entwicklung von iOS und Android Apps

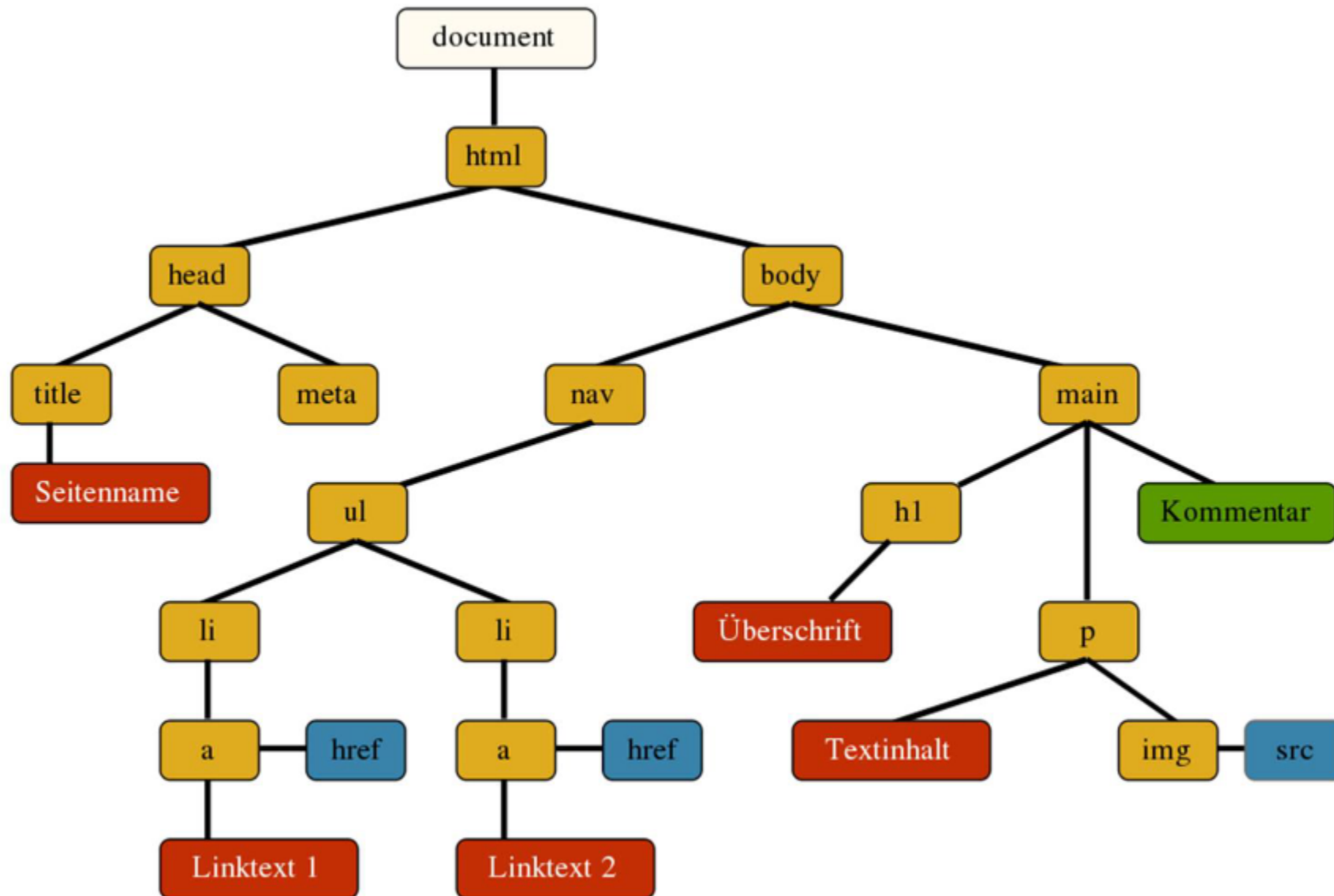


## DOM-MODELL - 1

- Das Document Object Modell einer HTML5-Datei ist ein Baumgraph-Modell der HTML-Datei im aktuellen Zustand zur Laufzeit
- Das DOM ist eine Darstellung des HTML-Dokuments als strukturierte Gruppe von Elementen und Objekten mit Eigenschaften und Methoden
- HTML-Elementen sind Events (Ereignisse) zugeordnet. Deren Event-Handler wird durch Nutzerinteraktion (z.B. Mausklick) oder automatisch (z.B. laden von Elementen) ausgelöst
- **Element Interfaces**

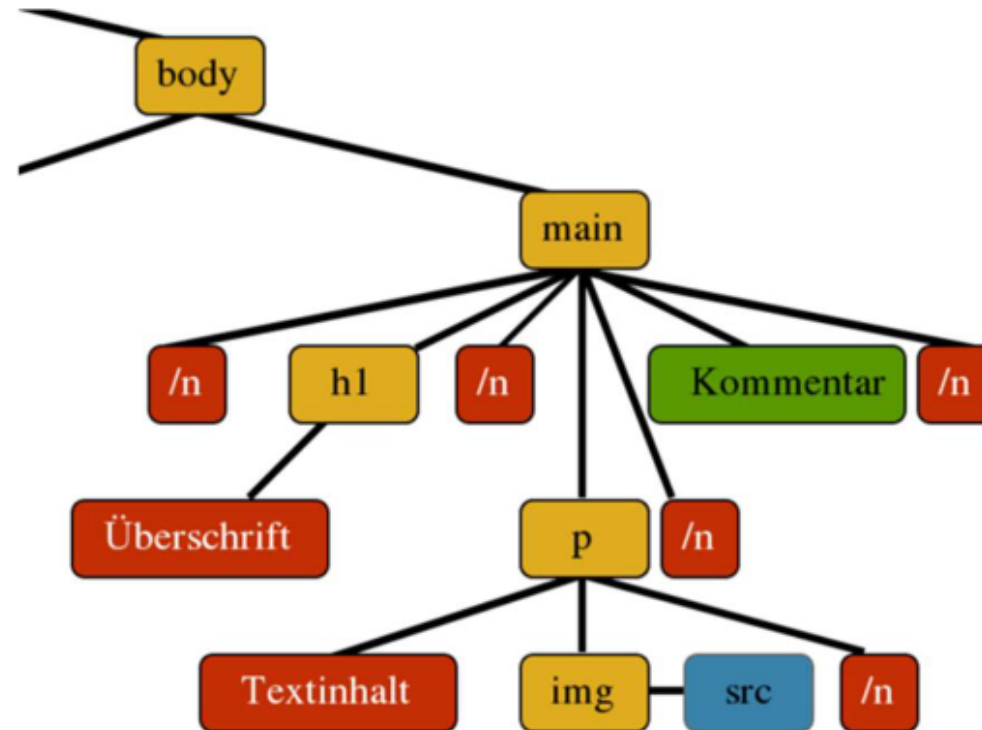


# DOM-MODELL - 2



Quelle

## DOM-MODELL - 3



Mit Textknoten an jeder möglichen Stelle (Mixed Content).

Quelle

## EINFLUSS VON JAVASCRIPT AUF DEN DOM

- Vom DOM der aktuellen Seite wird eine entsprechend geänderte Instanz erstellt
- Es wird versucht, das HTML-Dokument (im Sinne von Elementen, Attributen, Event-Handlern, Texten etc., auch CSS) zu ändern
- Im Erfolgsfall wird das DOM durch die vorab geänderte Instanz ersetzt
- Im Misserfolgsfall wird die geänderte Instanz verworfen

# JAVASCRIPT EINBINDEN

- Guter Stil zur Einbindung von JavaScript in Webseiten ist unter HTML5 das Einbinden mit Hilfe von Verweisen auf einzubindende, externe JavaScript- Bibliotheken
- Man beginnt im Element `<head>` die Liste mit `<script>`-Elementen mit den Standardbibliotheken und schreitet dann zu den Submodulen und zu den eigenen Spezialdateien (für Webseiten kritische Bibliotheken)
- Einbindung von weniger wichtigen Javascript Bibliotheken am Ende des HTML5 Documents (weniger Render-Blockierung beim ersten Laden der Seite)

## JAVASCRIPT ABARBEITEN - 1

- Es sei angemerkt, dass eine zu Beginn geladene JavaScript-Funktion oft nicht sofort nach ihrem Laden ausgeführt werden kann
- Wenn HTML-Elemente weiter unten im Quelltext angesprochen werden, müssen diese erst gerendert worden sein, bevor sie angesprochen werden können – und sei es, um ihnen das Ereignis onclick oder ähnliche formal zuordnen zu können

## JAVASCRIPT ABARBEITEN - 2

Eine gute Idee ist, den ersten Aufruf der Funktion über Ereignisse zu regeln:

- Idee 1: `window.onload = caller()`
- Das onload-Ereignis tritt erst ein, wenn das gesamte Dokument mitsamt aller externen Ressourcen vom Webserver heruntergeladen wurde (Das kann sehr lange dauern)
- Idee 2: Ereignis `DOMContentLoaded` nutzen
- Ereignis wird ausgelöst, wenn das HTML-Dokument vollständig geparkt wurde

# EREIGNISMANAGEMENT

Möglichkeiten der Zuordnung von Event-Handlern  
durch Zuweisung von Attributwerten:

```
1 <element id='IDname' onereignistyp='meineFunktion()' ... />  
2  
3 getElementByID(#IDname).onereignistyp= meineFunktion;
```

# EREIGNISMANAGEMENT IN HTML5 - 1

- HTML5 hat einen Paradigmenwechsel in der Ereignis-zu-Element-Zuordnung vollzogen
- Wurden bisher Paare (Ereignistyp, Handler) beim Element in Form von Ereignisattributen mit Handlerfunktionen registriert, so werden jetzt Paare (Element, Handler) beim Ereignistyp in einem neuen Eventmanager registriert
- Mittlerweile auf allen Browsern Unterstützt
- JavaScript funktioniert ereignisgesteuert. Die in HTML5 verfügbaren Ereignistypen registrieren HTML-Elemente unter Zuweisung von Handlern



## EREIGNISMANAGEMENT IN HTML5 - 2

- Dabei kann man ein Paar (Ereignis, Handler) mehreren Elementen zugleich zuordnen, und ein Element kann mehrfach bei Ereignissen gleichen Typs mit verschiedenen Handlern registriert werden
- Zur Wahl der Elemente gibt es neue Selektoren in JavaScript, die teils Arrays zurückgeben

## EREIGNISMANAGEMENT IN HTML5 - 3

```
1 //Angenommen <body> enthält mehrere <p> - Elemente:
2 function showmessage() {
3     alert('Element angeklickt');
4 }
5 function clickelement() {
6     document.getElementsByTagName('p')[0].onclick=showmessage;
7 }
8 window.onload=clickelement;
9 //Alternative einfache Selektoren:
10 getElementById(#IDname);
11 getElementsByClassName(.classname);
12 getElementsByName('name');
```

## EREIGNISMANAGEMENT IN HTML5 - 4

```
1 //Es gibt neue Selektoren in JavaScript:
2 function clickelement2() {
3     document.querySelector('#main p').onclick= showmessage;
4 }
5 //Dieser Selektor gibt immer nur das erste gefundene Element zu
6 //nächste gibt ein Array an:
7 function clickelement3() {
8     var list=document.querySelectorAll('#main p');
9     list[0].onclick=showmessage;
10    for(var f=0; f<list.length; f++) {
11        list[f].onclick=showmessage;
12    }
13 }
```

## EREIGNISMANAGEMENT IN HTML5 - 5

```
1 //In HTML5 wird nun eine neue Funktion eingeführt:
2 function clickelement4() {
3     var p1=document.getElementsByTagName('p')[0];
4     p1.addEventListener('click', showmessage, false);
5 }
6 window.addEventListener('load', clickelement4, false);
7 //Die Anzahl der Event Listener pro Element kann größer als Eins sein
8 //Events als Auslöser.
```

## EREIGNISMANAGEMENT IN HTML5 - 6

- `addEventListener('load', clickelement4, false)`  
true hat extra Zweck - [Event bubbleing](#)
- 3. Parameter betrifft 'useCapture' (Event Capturing und Event-Bubbleling)
- [Beispiel](#)

# EVENT BUBBLING

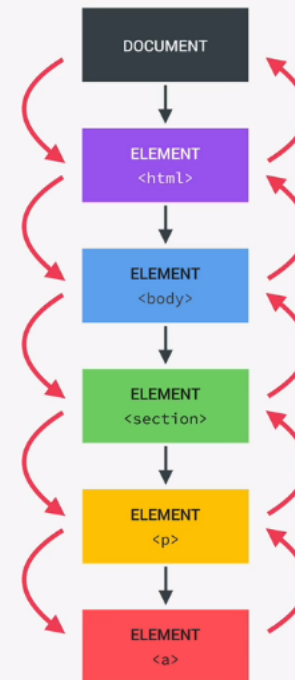
## BUBBLING AND CAPTURING

```
<html>
<head>
  <title>A Simple Page</title>
</head>
<body>
  <section>
    <p>A paragraph with a <a href="#">link</a> </p>
    <p>A second paragraph</p>
  </section>
  <section>
    
  </section>
</body>
</html>
```

(THIS DOES NOT HAPPEN  
ON ALL EVENTS)

1  
CAPTURING  
PHASE

2  
TARGET PHASE



Click event

3  
BUBBLING  
PHASE

```
document
.querySelector('section')
.addEventListener('click', () => {
  alert('You clicked me 🍕');
});
```

127.0.0.1:8080 says  
You clicked me 🍕

```
document
.querySelector('a')
.addEventListener('click', () => {
  alert('You clicked me 🍕');
});
```

127.0.0.1:8080 says  
You clicked me 🍕

Quelle

## EVENTFUNKTIONEN ENTFERNEN - 1

Ereignisse können ähnlich auch wieder deregistriert werden, wobei aber Registrierung und Deregistrierung sensibel voneinander abhängen ([siehe hier](#))

```
1 document.getElementsByTagName('p')[0].addEventListener('mousemove',  
2 document.getElementsByTagName('p')[0].removeEventListener('mousemove',  
3 //removes the listener that exactly matches the function that was a
```

## EVENTFUNKTIONEN ENTFERNEN - 2

Ereignisse können ähnlich auch wieder deregistriert werden, wobei aber Registrierung und Deregistrierung sensibel voneinander abhängen ([siehe hier](#))

```
1 var mevent = function() {alert("NO!")};  
2 document.getElementsByTagName('p')[0].addEventListener('mousemove', mevent);  
3 document.getElementsByTagName('p')[0].removeEventListener('mousemove', mevent);  
4 //removes the listener that exactly matches the function that was added
```



## EREIGNISMANAGER - REIHENFOLGEN

In welcher Reihenfolge wird dies abgearbeitet?

```
1 function start () {  
2   var pElement = document.getElementById("interaktiv");  
3   pElement.addEventListener("click", meldung1, false);  
4   pElement.addEventListener("click", meldung2, false);  
5 }  
6 function meldung1 () {  
7   window.alert("Erste Handler-Funktion ausgeführt!");  
8 }  
9 function meldung2 () {  
10  window.alert("Zweite Handler-Funktion ausgeführt!");  
11 }
```

## JAVASCRIPT EINBINDEN

```
1 function dokumentGeladen (e) {  
2   alert("Das Ereignis " + e.type + " ist passiert.")  
3 }  
4 document.addEventListener("load", dokumentGeladen, false);  
5 document.addEventListener("DOMContentLoaded",dokumentGeladen, false);
```

## WEITERE EVENTS IN HTML5 UND NÜTZLICHE LINKS

- In HTML5 sind im Vergleich zu HTML 4 viele Events hinzugekommen, nicht nur für die Maus, auch für Fenster, Formulare, Tastatur und Medien
- Neuste Events: [Quelle](#)
- EventHandlering: [Quelle](#)
- Aktueller Stand: [Quelle](#)

# GRUND DER ENTWICKLUNG VON BROWSERAPIS

- Browser-APIs sind in Webbrowsern integriert und können Daten aus dem Browser und der umgebenden Computerumgebung offenlegen
- Im Hintergrund verwendet der Browser einen komplexen Code auf niedrigerer Ebene (z. B. C++ oder Rust), welche vom Javascript angesprochen werden (aber unterschiedliches Verhalten und Geschwindigkeit)
- Entwickelt um Webseiten einfache interaktive Funktionalitäten realisieren zu können
- Für die Kommunikation zwischen HTML, JavaScript und CSS stehen APIs zur Verfügung (z.B. für die Übergabe von Instanzen und ihren Werten)
- [Liste aller aktuellen BrowserAPI](#)
- [Mehr Informationen zu APIs](#)

## SUPPORT VON BROWSERAPIS

- Auf praktisch allen Webbrowsern verfügbar
- Kann jedoch deaktiviert werden (generell oder dynamisch, z.B. durch **Noscript** oder Browsereinstellungen)
- **Die Hauptfunktion einer Webseite sollte daher nicht von JavaScript abhängen**

## APIS FÜR JAVASCRIPT IN HTML5

- In HTML5 gibt zahlreiche APIs mit klarer Zweckbestimmung, um mit einfacher JavaScript-Syntax komplexe Vorgänge steuern zu können
- Im Folgenden werden einige APIs umrissen und ihre derzeitige Unterstützung durch Webbrowser aufgezeigt
- Jede API braucht ein tieferes Verständnis für dessen Funktionsweise, um genutzt werden zu können (Nutzen Sie hierfür die Selbststudienzeit)

## CANVAS-API - 1

- Die Canvas-API liefert Zeichenwerkzeuge in einer mächtigen Zeichenoberfläche `<canvas>`
- Fähigkeiten sind unter anderem die dynamische Erstellung von Grafiken, die Animation von Texten und Bildern, die Verarbeitung von Texten und Bildern. Videos werden frameweise als Bilder behandelt
- Das Bild auf der Canvas ist eine Bitmap, die durch API-eigene Funktionen erstellt und laufend verändert werden kann

## CANVAS-API - 2

- Ein gravierender Nachteil der Canvas-API ist, dass es genau eine Zeichenfläche ohne Layer gibt. Sich partiell verdeckende Bildelemente zu verändern ist schwierig
- Änderung von Bildelementen geht nur mit einem dem Bildelement entsprechenden Objekt in der JavaScript-Schicht der Canvas
- Es muss neu gezeichnet werden, wobei mitunter vorher erst sein sichtbarer Anteil im Bild gelöscht werden mussn



## CANVAS-API - 3

- Viele Aktionen auf der Canvas sind den Konzepten der Scalable Vector Graphics (SVG) nachempfunden, aber in JavaScript implementiert
- Es besteht die Möglichkeit, Screenshots von Teilbereichen abzuspeichern, solange keine fremden Bilder oder Videos in die Canvas geladen worden sind. (Bild- /Videoquelle = gleiche Domäne)
- Das folgende Beispiel zeigt, wie ein eigenes Bild oder Video live in Farbe und zugleich in Schwarz-Weiß gezeigt werden kann
- [Video filter](#)

## CANVAS-API - BEISPIELE

```
1 <canvas id="myCanvas" width="200" height="100" style="border:1px solid black;">
2 Your browser does not support the HTML canvas tag.</canvas>
3
4 <script>
5     var c = document.getElementById("myCanvas");
6     var ctx = c.getContext("2d");
7     ctx.beginPath();
8     ctx.arc(95,50,40,0,2*Math.PI);
9     ctx.stroke();
10 </script>
```

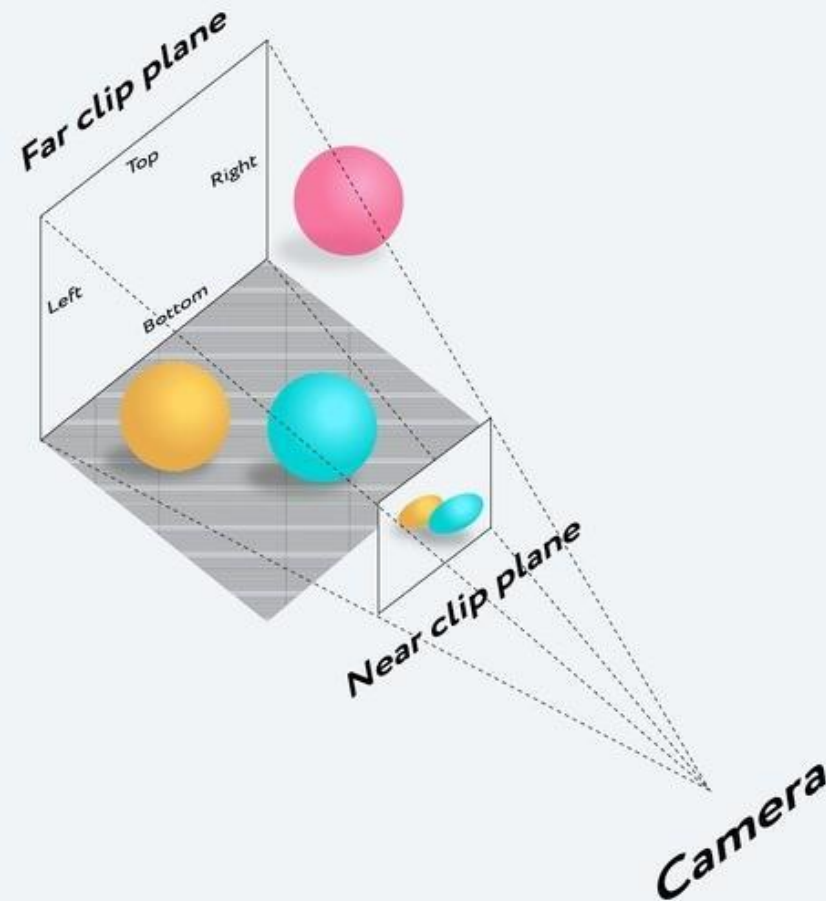
## ANWENDUNGEN BASIEREND AUF CANVAS

- [Openstreetmap](#)
- [Excalidraw](#)
- [Canvas-Effects](#)
- [Canvas-Guide](#)

## WEBXR,VR,3D-TECHNOLOGIE (AUF BASIS VON CANVAS UND WEBGPU)

- <https://threejs.org/>
- <https://aframe.io/>
- <https://needle.tools/>

# THREEJS - FUNKTIONSWEISE




Quelle

## CANVAS - WEITERES


- Spezifikation des W3C, [Quelle](#)
- [Canvas tutorial](#)
- 48 Potential Flash-Killing Demos, [Quelle](#)
- Bibliothek Explorercanvas (excanvas.js), [Quelle](#)
- 3D-Experimente mit Shadern: [Quelle](#)
- Blockzerlegung von HTML5 Videos: [Quelle](#)

# CANVAS SUPPORT

Canvas (basic support)  - LS

Usage % of all users Global 97.01% + 0.05% = 97.06%

Method of generating fast, dynamic graphics using JavaScript.

Current aligned Usage relative Date relative Filtered All 

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
		3.1-3.2	2-3.5		6-8							2.1-2.3				
4-128	12-128	4-17.6	3.6-130	10-113	9-10		3.2-17.6	4-24		12-12.1		3-4.4.4				2.5
129	129	18.0	131	114	11	129	18.0	25	all	80	15.5	129	130	14.9	13.52	3.1
130-132		18.1-TP	132-134				18.1									


Quelle



## DRAG UND DROP-API


- Das Ziel dieser API ist es, durchaus komplexe Elemente von Webseiten nach dem Konzept von Drag&Drop mit der Maus in andere Elemente der Webseite zu ziehen
- Das DOM und die HTML-Datei ändern sich dadurch
- Die Elemente können Text, Grafiken, Links und anderes enthalten. Aber auch Dateien können vom Desktop in die Webseite im Browser gezogen werden, allerdings nicht als Ikone

# DRAG UND DROP - SUPPORT

Drag and Drop  - LS

Usage % of all users Global 93.46% + 0.6% = 94.06%

Method of easily dragging and dropping elements on a page, requiring minimal JavaScript.

Current aligned Usage relative Date relative Filtered All 

Chrome	Edge *	Safari	Firefox	Opera	IE	Chrome for Android	Safari on iOS *	Samsung Internet	Opera Mini *	Opera Mobile *	UC Browser for Android	Android Browser *	Firefox for Android	QQ Browser	Baidu Browser	KaiOS Browser
	2 12-17		2-3	10-11.5	1 3 6-9		3.2-14.8			12						
4-128	18-128	3.1-17.6	3.5-130	12.1-113	2 3 10		15-17.6	4-24		12.1		2.1-4.4.4				2.5
129	129	18.0	131	114	2 3 11	4 129	18.0	25	all	4 80	4 15.5	4 129	130	14.9	13.52	3.1
130-132		18.1-TP	132-134				18.1									

Quelle

## DRAG UND DROP - BEISPIEL 1

```
1 <script>
2 function allowDrop(ev) {
3     ev.preventDefault();
4 }
5
6 function drag(ev) {
7     ev.dataTransfer.setData("text", ev.target.id);
8 }
9
10 function drop(ev) {
11     ev.preventDefault();
12     var data = ev.dataTransfer.getData("text");
13     ev.target.appendChild(document.getElementById(data));
14 }
15 </script>
```

Beispiel

## DRAG UND DROP-API - WEITERES

- Spezifikation des W3C, [Quelle](#)
- [W3C Drag und Drop](#)
- [API](#)
- [API Operationen](#)

## GEOLOCATION-API

- Mit der Geolocation-API kann der Standort des Clients festgestellt werden, der die Webseite lädt
- Genutzt werden Netzwerktriangulation (z.B. IP-Adressen) oder GPS-Signale. Die Rückgabe besteht aus geografischer Länge und Breite
- Die Genauigkeit ist erstaunlich hoch, in Leipzig ohne GPS bis auf unter 50 Meter genau (mit GPS auf 3-5m genau)

## GEOLOCATION-API - VORGEHENSWEISE

- Mit aufsteigender Genauigkeit werden je nach Anfrageclient IP-Adresse, Wlan-Netzwerk oder Funksignale (Mobilfunk) genutzt
- GPS-Sender nur bei ‚enableHighAccuracy‘ (z.B. Einstellbar bei Android)
- Die Genauigkeit ist erstaunlich hoch, in Leipzig ohne GPS bis auf unter 50 Meter genau (mit GPS auf 3-5m genau)
- Beispielweise: [Quelle](#)
- [Beispiele](#)

# GEOLOCATION-API - SUPPORT



Quelle

## GEOLOCATION-API - WEITERES

- Spezifikation des W3C, <http://www.w3.org/TR/geolocation-API/>
- Demo im Web, [Quelle](#)
- Tutorial, [Quelle](#)



## QUELLEN

- JavaScript-Tutorial online, [Quelle](#)
- Web application APIs, [Quelle](#)
- GeoLocation Testing for Smartphones: [Quelle](#)
- Eine eCommerce-Anwendung: [Quelle](#)
- Reverse-Geocoding: [Quelle](#)
- [OpenStratmap](#)

## **ABSPANN**

Fünftes Level geschafft weitere Folgen!

Fragen und Feedback?