

# JavaScript

Dr. F.-K. Koschnick, Sybit GmbH

# JavaScript (JS)

- Skriptsprache ursprünglich von Netscape 1995 für dynamisches HTML in Browsern entwickelt
- Heute nicht nur client-seitig im Browser, auch auf Servern (Beispiel Node.js)
- Sprachkern von JS standardisiert: [ECMAScript® 2018](#)
- Syntax von JS ähnelt den C-Abkömmlingen (also auch Java)

# Wozu JavaScript?

- Dynamische Manipulation von Webseiten über das Document Object Model
- Plausibilitätsprüfung von Formulareingaben noch vor der Übertragung zum Server
- Anzeige von Dialogfenstern
- Senden und Empfangen von Daten, ohne dass der Browser die Seite neu laden muss (Ajax)
- Vorschlagen von Suchbegriffen während der Eingabe
- Schreib- und Lesezugriff auf Cookies und den Web Storage innerhalb des Browsers
- .....

1. **HTML** to define the content of web pages
2. **CSS** to specify the layout of web pages
3. **JavaScript** to program the behavior of web pages

# JavaScript: Einbinden

## (1. script-Tag im head)

```
<!DOCTYPE html>
<html><head>
<script>
function myFunction() {
    document.getElementById("test").innerHTML = "Der Text wurde geändert";
}
</script>
</head>
<body>
<h1>A Web Page</h1>
<p id="test">Ein Text</p>
<button type="button" onclick="myFunction()">Test</button>
</body>
</html>
```


# JavaScript: Einbinden

## (2. script-Tag im body)

```
<!DOCTYPE html>
<html>
<body>
<h1>A Web Page</h1>
<p id="test">Ein Text</p>
<button type="button" onclick="myFunction()">Test</button>

<script>
function myFunction() {
  document.getElementById("test").innerHTML = "Der Text wurde geändert";
}
</script>

</body>
</html>
```



Dann script-Tag am Ende vom body  
(Performance beim Seite-Laden)

# JavaScript: Einbinden

## (3. externe JS-Datei)

In externer Datei: myScript.js

```
function myFunction() {  
    document.getElementById("test").innerHTML = "Paragraph changed.";  
}
```

In head oder body:

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

Bemerkung: myScript.js darf kein script-Tag enthalten!

# JavaScript: Einbinden

## (3. externe JS-Datei)

### Mehrere Skripte einbinden:

```
<script src="myScript1.js"></script>  
<script src="myScript2.js"></script>
```

### Relative URL:

```
<script src="/js/myScript1.js"></script>
```

### Absolute URL:

```
<script src="https://www.xyz.de/js/myScript1.js"></script>
```

# JavaScript: Einbinden

## (3. externe JS-Datei)

### **Vorteile, wenn man externe JavaScript-File einbindet:**

- Separation von HTML und JS-Code
- Macht HTML und JavaScript leichter lesbar und damit leichter wartbar
- Browser chached JavaScript-Files, Performance beim Laden
- Man kann mehrere JavaScript-Files verwenden



# JavaScript: Variable

[https://www.w3schools.com/js/js\\_variables.asp](https://www.w3schools.com/js/js_variables.asp)

## **Variablen-Namen:**

- case sensitive
- keine Umlaute
- kein Bindestrich
- erstes Zeichen Buchstabe oder \$ oder \_
- keine JavaScript-Schlüsselwörter

## **Variablendeklaration:**

*var* meineVariable = 8;

# JavaScript: 4 primitive Datentypen

[https://www.w3schools.com/js/js\\_datatypes.asp](https://www.w3schools.com/js/js_datatypes.asp)

## Primitive Variablen-Typen:

- Numerische Variable: pi = 3.14; oder zahl = 47;
- String: meinString = "Hallo"; oder meinString = 'Hallo'
- 
- Boolean: isGreen = true; oder isGreen = false;
- Undefined: var x;  
oder var x = 7;  
x = undefined;

# JavaScript: String

[https://www.w3schools.com/js/js\\_strings.asp](https://www.w3schools.com/js/js_strings.asp)

[https://www.w3schools.com/js/js\\_string\\_methods.asp](https://www.w3schools.com/js/js_string_methods.asp)

## Sonderzeichen bei Strings

Ausdruck	Bedeutung
\r	Wagenrücklauf
\n	Neue Zeile
\t	Tabulator
\b	Backspace (Löschtaste)
\f	Seitenvorschub
\	Escape-Zeichen (z.B.: meinPfad = "C:\\user")

String-Methoden:

- meinString.length
- meinString.indexOf("substring")
- .....

# JavaScript: Objekte

[https://www.w3schools.com/js/js\\_object\\_definition.asp](https://www.w3schools.com/js/js_object_definition.asp)

```
var person = {  
  firstName: "Hans",  
  lastName: "Meier",  
  age: 50,  
  eyeColor: "blue",  
  fullName: function() {return this.firstName + " " + this.lastName;}  
};
```

**Zugriff auf Properties:** person.firstName oder person[" firstName "]

**Zugriff auf Methoden:** person.fullName()

- **Achtung:** Ohne die Klammern wird die Funktion an sich ausgegeben

person = null; // immer noch vom Typ Object

person = undefined; // jetzt ist person vom Typ undefined

# JavaScript: Array

[https://www.w3schools.com/js/js\\_arrays.asp](https://www.w3schools.com/js/js_arrays.asp)

[https://www.w3schools.com/js/js\\_array\\_methods.asp](https://www.w3schools.com/js/js_array_methods.asp)

```
var animals = ["Hund", "Katze", "Maus", "Elefant"];
```

~~var animals = new Array("Hund", "Katze", "Maus", "Elefant");~~

2 Elemente: 35 und 10

var myArray = new Array(35,10);

var myArray = new Array(35);

35 undefinierte Elemente

Zugriff über nullbasierten Index

```
var animals, text, animalsLen, i;
animals = ["Hund", "Katze", "Maus", "Elefant"];
animalsLen = animals.length;
text = "<ul>";
for (i = 0; i < animalsLen; i++) {
    text += "<li>" + animals[i] + "</li>";
}
text += "</ul>";
```

Hinzufügen - Entfernen

```
var animals = ["Hund", "Katze", "Maus", "Elefant"];
animals.push("Hase"); //fügt Element als Letztes hinzu
animals[animals.length] = "Pferd"; //fügt Element als Letztes hinzu

var lastAnimal = animals.pop(); //entfernt letztes Element

//Siehe auch .unshift (fügt zu Beginn ein) und .shift entfernt erstes Element
```

# JavaScript: Array oder Objekt?

[https://www.w3schools.com/js/js\\_arrays.asp](https://www.w3schools.com/js/js_arrays.asp)

[https://www.w3schools.com/js/js\\_array\\_methods.asp](https://www.w3schools.com/js/js_array_methods.asp)

Wann Objekte und wann Arrays nutzen:

- JavaScript unterstützt keine assoziativen Arrays, Index also rein zahlenbasiert
- Nutze Arrays, wenn die Elemente mit Zahlen adressiert werden sollen
- Nutze Objekte, wenn die Elemente mit Namen versehen sein sollen

## ACHTUNG FALLE:

```
var person = [];  
person["firstName"] = "John";  
person["lastName"] = "Doe";  
person["age"] = 46;  
var x = person.length;  
var y = person[0];
```

//Definition eines Arrays

//JavaScript wandelt automatisch den Array in ein Objekt um!!!

// person.length gibt 0 zurück

// person[0] gibt undefined zurück

# JavaScript: Syntax

[https://www.w3schools.com/js/js\\_statements.asp](https://www.w3schools.com/js/js_statements.asp)

Java und JavaScript haben eigentlich nichts miteinander zu tun.

Dennoch gibt es Ähnlichkeiten in der Syntax:

- Semikolon am Zeilenende, bzw. nach einer Anweisung ;
- Klammerung {}
- Kontrollstrukturen: for, while ,if
- Kommentare: // bzw. mit /\*...\*/
- Funktionen, Methoden (bei Objekten)

Es gibt zahlreiche vordefinierte Funktion in JavaScript, welche verschiedene Aufgaben erfüllen...

alert("Text") (PopUp-Fenster)

parseInt("7") (wandele String in Zahl)

Number(Ausdruck) (Konvertierung von Ausdruck nach Zahl)

```
z.Bsp: function add(a,b){  
var z = a + b;  
return z;  
}
```

```
if (bedingung) {  
anweisungen;  
}  
else {  
anweisungen;  
}  
while (bedingung) {  
anweisungen;  
}  
for (startausdruck; bedingung;  
iterationsausdruck) {  
anweisungen;  
}
```

# JavaScript: Funktionen

[https://www.w3schools.com/jS/js\\_function\\_definition.asp](https://www.w3schools.com/jS/js_function_definition.asp)

```
function add(a,b){  
    var z = a + b;  
    return z;  
}
```

-----

## Funktion in Variable gespeichert

```
var x = function(a, b) {return a * b};  
var z = x(8,2);
```

-----

## Sich selbst aufrufende Funktion

...

```
<p id="demo"></p>
```

...

```
<script>
```

```
(function() {
```

```
    document.getElementById("demo").innerHTML = "Hallo, ich habe mich selbst aufgerufen!";
```

```
})();
```

```
</script>
```

...

Anonyme Funktion



# JavaScript: Funktionen

[https://www.w3schools.com/jS/js\\_function\\_definition.asp](https://www.w3schools.com/jS/js_function_definition.asp)

## Parameter:

```
function add(a,b){  
    var z = a + b;  
    return z;  
}
```

## Aufruf mit weniger Parameter:

x = add(5) **//-> b: undefined**

```
function add(a, b) {  
    if (b === undefined) {  
        b = 0;  
    }  
    return a + b  
}
```

## Aufruf mit mehr Parametern als definiert oder mit beliebigen Parametern:

x = findMax(1, 123, 500, 115, 44, 88);

```
function findMax() {  
    var i;  
    var max = -Infinity;  
    for (i = 0; i < arguments.length; i++) {  
        if (arguments[i] > max) {  
            max = arguments[i];  
        }  
    }  
    return max;  
}
```

# JavaScript: Closures

## Counter-Problem

```
// Initiate counter
var counter = 0;

// Function to increment counter
function add() {
  counter += 1;
}

// Call add() 3 times
add();
add();
add();

// The counter should now be 3
```

### Was ist schlecht an diesem Beispiel?



Die globale Variable counter kann von allen JS-Funktionen überschrieben werden. Die globale Variable counter ist nicht gekapselt (versteckt).

# JavaScript: Closures

[https://www.w3schools.com/js/js\\_function\\_closures.asp](https://www.w3schools.com/js/js_function_closures.asp)

Anonyme, sich selbst aufrufende Funktion

## Counter-Problem

Closure

```
var add = (function () {  
    var counter = 0;  
    return function () {counter += 1; return counter}  
})();  
  
add();  
add();  
add();  
  
// the counter is now 3
```

### Wie funktioniert das?

1. Die anonyme, sich selbst aufrufende Funktion läuft einmal automatisch durch.
2. Dadurch wird zum einen die Variable counter auf 0 gesetzt und zum anderen wird die innere Funktion (Closure) als Return-Wert an die Variable add zurückgegeben.
3. Bei jedem Aufruf von add wird nur die innere Funktion aufgerufen, diese hat aber immer noch Zugriff auf die gekapselte Variable counter.

Ein **Closure** ist eine Funktion, die Zugriff auf die Variablen des Elternelements (Elternfunktion) hat, wobei die Elternfunktion beendet (abgelaufen) ist.

# JavaScript: Operatoren

[https://www.w3schools.com/js/js\\_operators.asp](https://www.w3schools.com/js/js_operators.asp)

Operator	Beschreibung	Beispiel	Ergebnis (Wert von a)
+	Addition	a = 8 + 5	13
-	Subtraktion	a = 7 - 5	2
*	Multiplikation	a = 9 * 4	36
/	Division	a = 9 / 4	2.25
%	Modulo (Restrechnung)	a = 8 % 5	3
-	Negation	b = 5 a = -b	-5
Operator	Bedeutung	Langform	Kurzform
+=	Addition	a = a + b	a += b
-=	Subtraktion	a = a - b	a -= b
*=	Multiplikation	a = a * b	a *= b
/=	Division	a = a / b	a /= b
%=	Modulo	a = a % b	a %= b
Operator	Beschreibung	Beispiel	Ergebnis (Wert für a)
==	Gleich	a = (3 == 7) a = ("Java" == "JavaScript")	false
!=	Ungleich	a = (8 != 4) a = ("Java" != "JavaScript")	true
>	Größer als	a = (3 > 8)	false
<	Kleiner als	a = (3 < 7)	true
>=	Größer oder gleich	a = (3 >= 8)	false
<=	Kleiner oder gleich	a = (3 <= 5)	true

# JavaScript: Best Practice

[https://www.w3schools.com/js/js\\_best\\_practices.asp](https://www.w3schools.com/js/js_best_practices.asp)

// Declare and initiate at the beginning

```
var firstName = "",
    lastName = "",
    price = 0,
    discount = 0,
    fullPrice = 0,
    myArray = [],
    myObject = {};
```

// Never declare primitive datatypes as objects

```
var x = "John";
var y = new String("John");
(x === y) // is false because x is a string and y is an object.
```

// Use Parameter Defaults

```
function myFunction(x, y) {
  if (y === undefined) {
    y = 0;
  }
}
```

// Beware of Automatic Type Conversions

```
var x = "Hello"; // typeof x is a string
x = 5;           // changes typeof x to a number
var x = "6" - 8; // is -2, typeof x is a number
var x = 5 - "x"; // x.valueOf() is NaN, typeof x is a number
```

// Use === Comparison (type and value)

```
0 == "";    // true
1 == "1";   // true
1 == true;  // true

0 === "";   // false
1 === "1";  // false
1 === true; // false
```

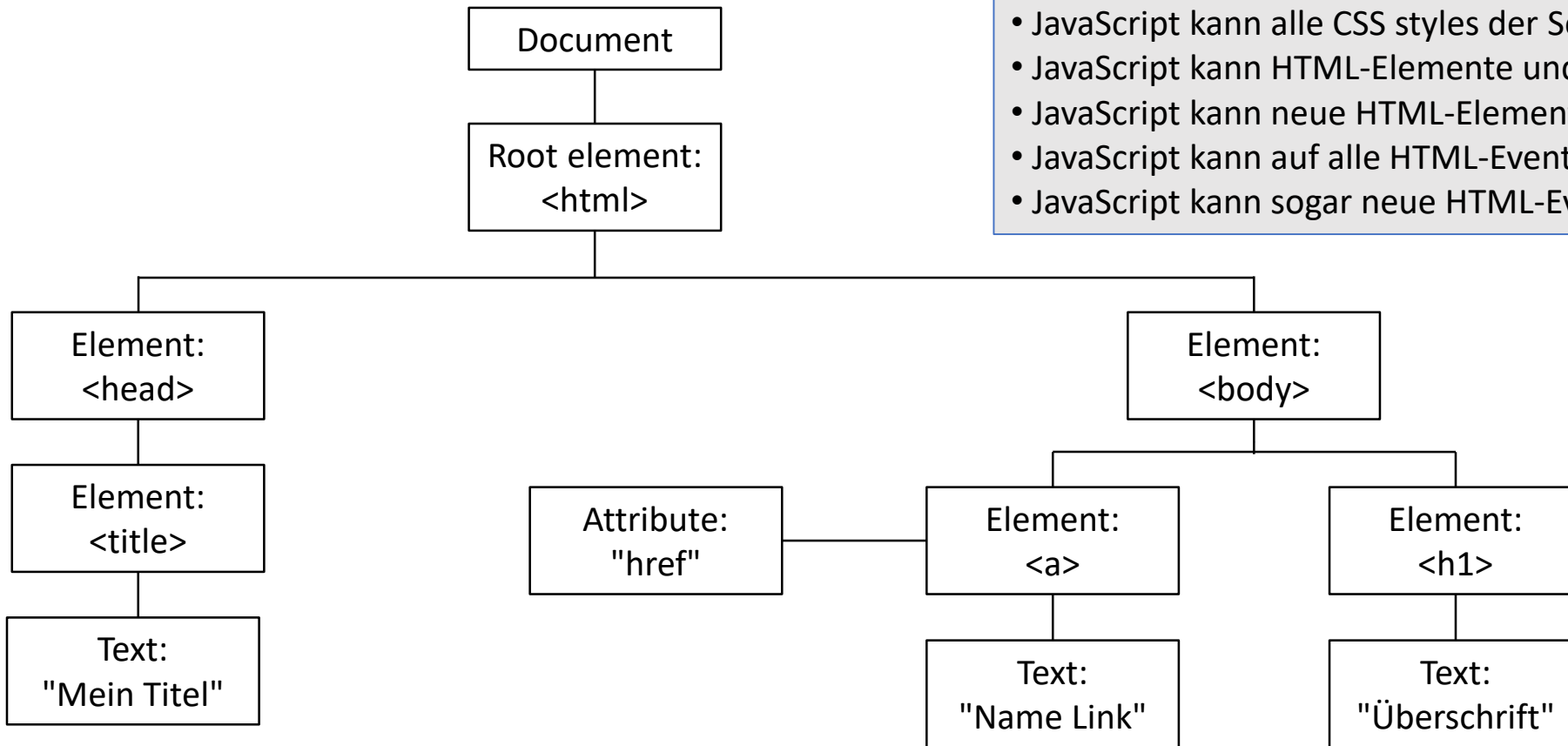
// End Your Switches with Defaults

```
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
    break;
  default:
    day = "Unknown";
}
```

# Document Object Model (DOM)

[https://www.w3schools.com/js/js\\_htmlDOM.asp](https://www.w3schools.com/js/js_htmlDOM.asp)

## DOM-Baum



- JavaScript kann HTML-Elemente und deren Attribute auf der Seite ändern
- JavaScript kann alle CSS styles der Seite ändern
- JavaScript kann HTML-Elemente und deren Attribute löschen
- JavaScript kann neue HTML-Elemente und Attribute hinzufügen
- JavaScript kann auf alle HTML-Events der Seite reagieren
- JavaScript kann sogar neue HTML-Events auf einer Seite erzeugen

HTML-DOM ist das Standard-Programming-Interface. Es definiert Elemente als Objekte mit Properties, Zugriffs-Methoden auf die Elemente und die Events für alle HTML-Elemente.

# JavaScript: Methoden (DOM)

[https://www.w3schools.com/js/js\\_htmlDOM\\_document.asp](https://www.w3schools.com/js/js_htmlDOM_document.asp)

## Elemente finden:

```
var myElement = document.getElementById("myId");
var x = document.getElementsByTagName("p");

var x = document.getElementById("main");
var y = x.getElementsByTagName("p");

var x = document.getElementsByClassName("intro");

var x = document.querySelectorAll("p.intro");
```

## Elemente manipulieren:

```
// Überschreibt Text des Elements
var element = document.getElementById("idX1");
element.innerHTML = "Neuer Text";

// Überschreibt das Attribut
document.getElementById("myImage").src = "image.jpg";
```

document.createElement(*element*)

document.removeChild(*element*)

document.appendChild(*element*)

document.replaceChild(*element*)

document.write(*text*)

```
var para = document.createElement("p");
var node = document.createTextNode("This is a new paragraph.");
var element = document.getElementById("div1");
para.appendChild(node);
element.appendChild(para);
```

# JavaScript: HTML-Events

[https://www.w3schools.com/js/js\\_htmldom\\_events.asp](https://www.w3schools.com/js/js_htmldom_events.asp)

## Kopplung von JavaScript-Code an HTML-Elemente über Events

### Beispiele für HTML-Events:

- Mouse-Click
- Seite oder Bild geladen
- Mouse-Over
- Mouse-Move
- HTML-Form submitted
- Eine Taste gedrückt
- ...

[https://www.w3schools.com/jsref/dom\\_obj\\_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)

### Weiterführend - DOM Event Listener:

[https://www.w3schools.com/js/js\\_htmldom\\_eventlistener.asp](https://www.w3schools.com/js/js_htmldom_eventlistener.asp)

```
<body onload="JavaScriptFunction()">

<input type="text" onchange="...">

<form ... onsubmit="return checkInputs()">
```

checkInputs() liefert boolean zurück. Bei false, wird Form nicht abgeschickt.

\\ onclick-Event direkt am HTML-Element

```
<button onclick="myFunction()">Try it</button>
```

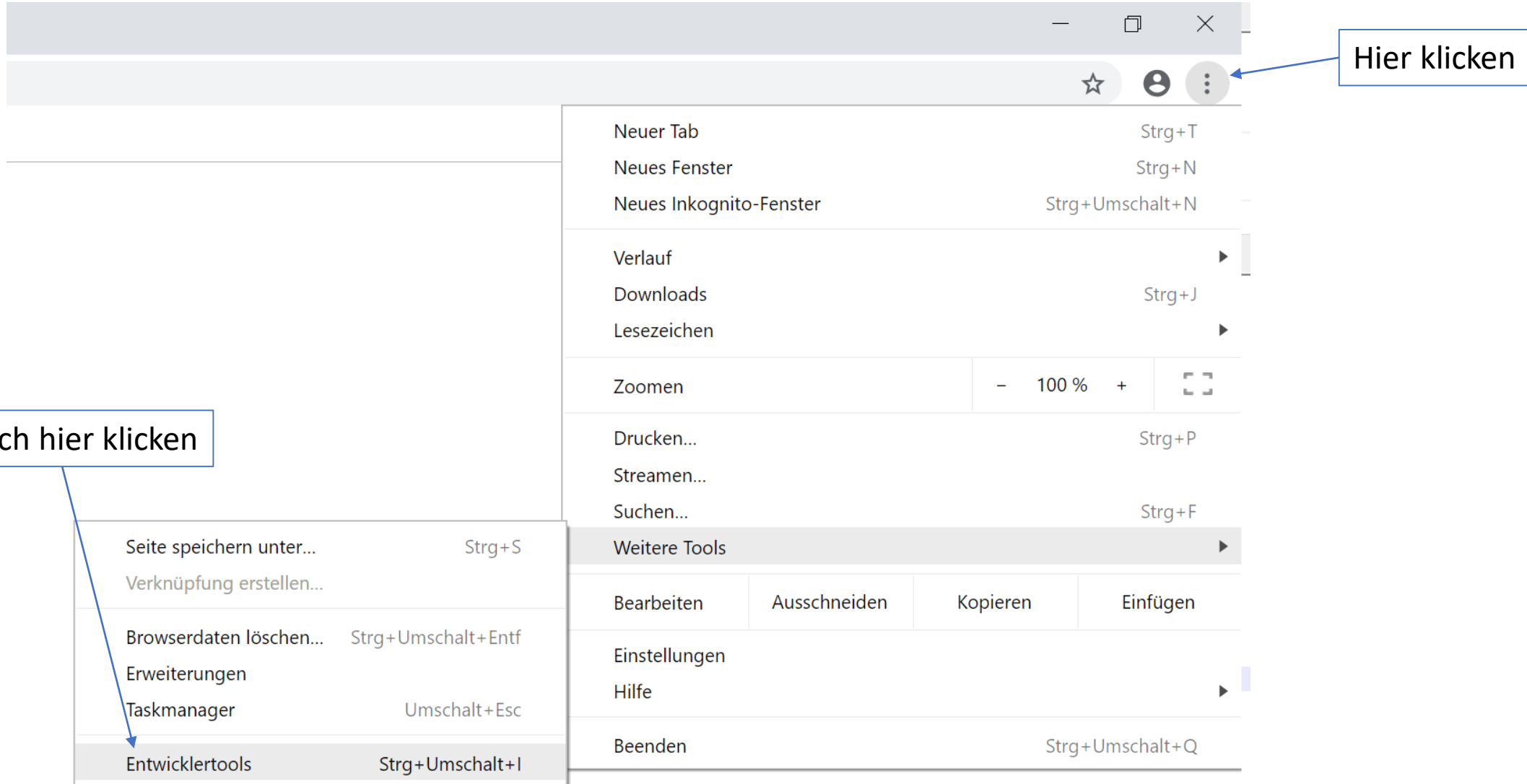
\\ Zuweisen des onclick-Events über JavaScript

```
<script>
document.getElementById("myBtn").onclick = myFunction;
....
function myFunction() {
    ....
}
</script>
```



# JavaScript Debuggen

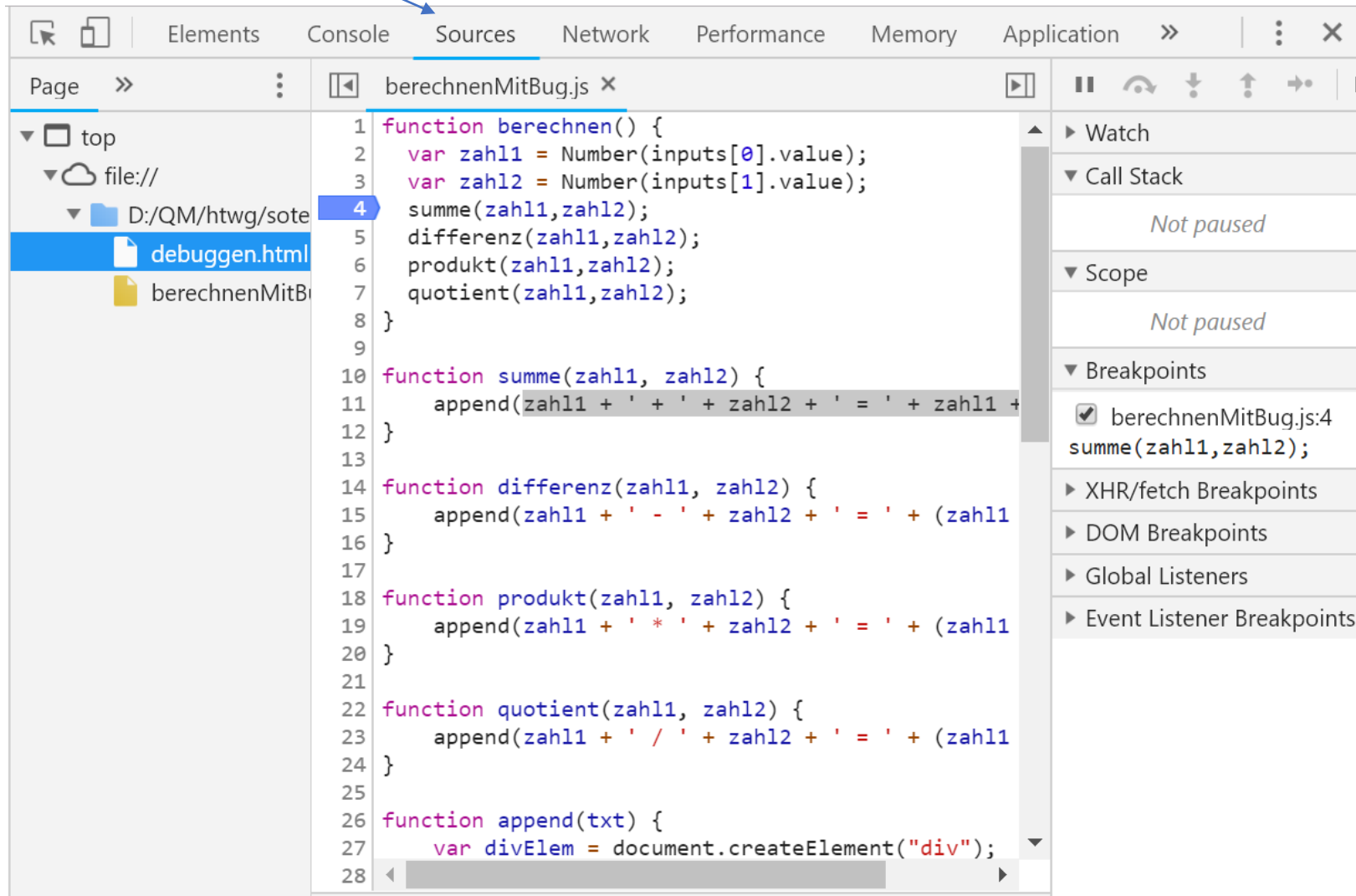
Beispiel Chrome-Browser: <https://developers.google.com/web/tools/chrome-devtools/javascript/>



Hier klicken

# JavaScript Debuggen

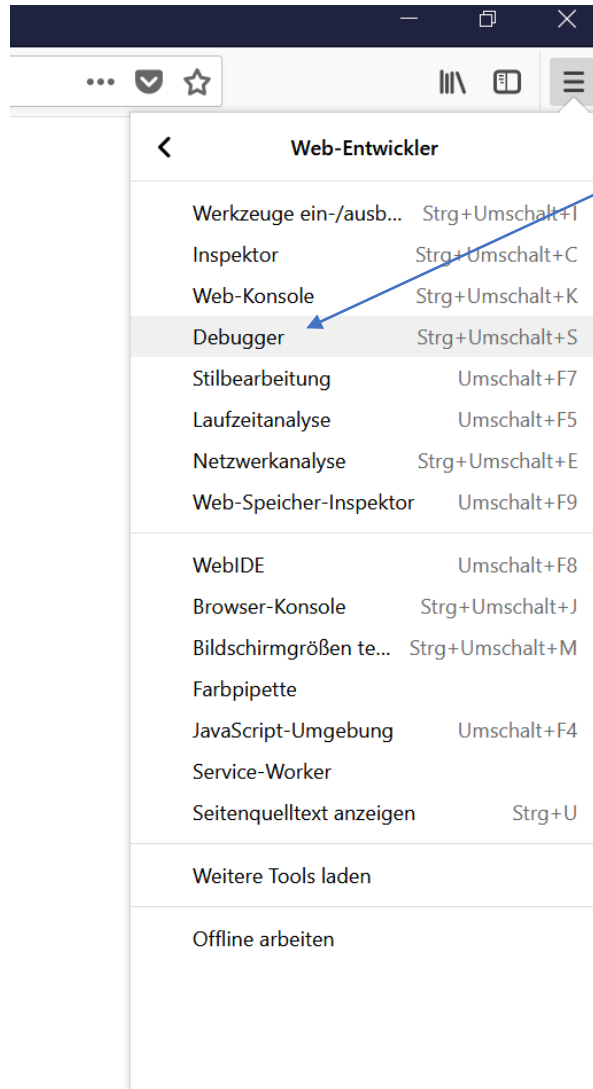
Beispiel Chrome-Browser: <https://developers.google.com/web/tools/chrome-devtools/javascript/>



# JavaScript Debuggen

Beispiel Firefox

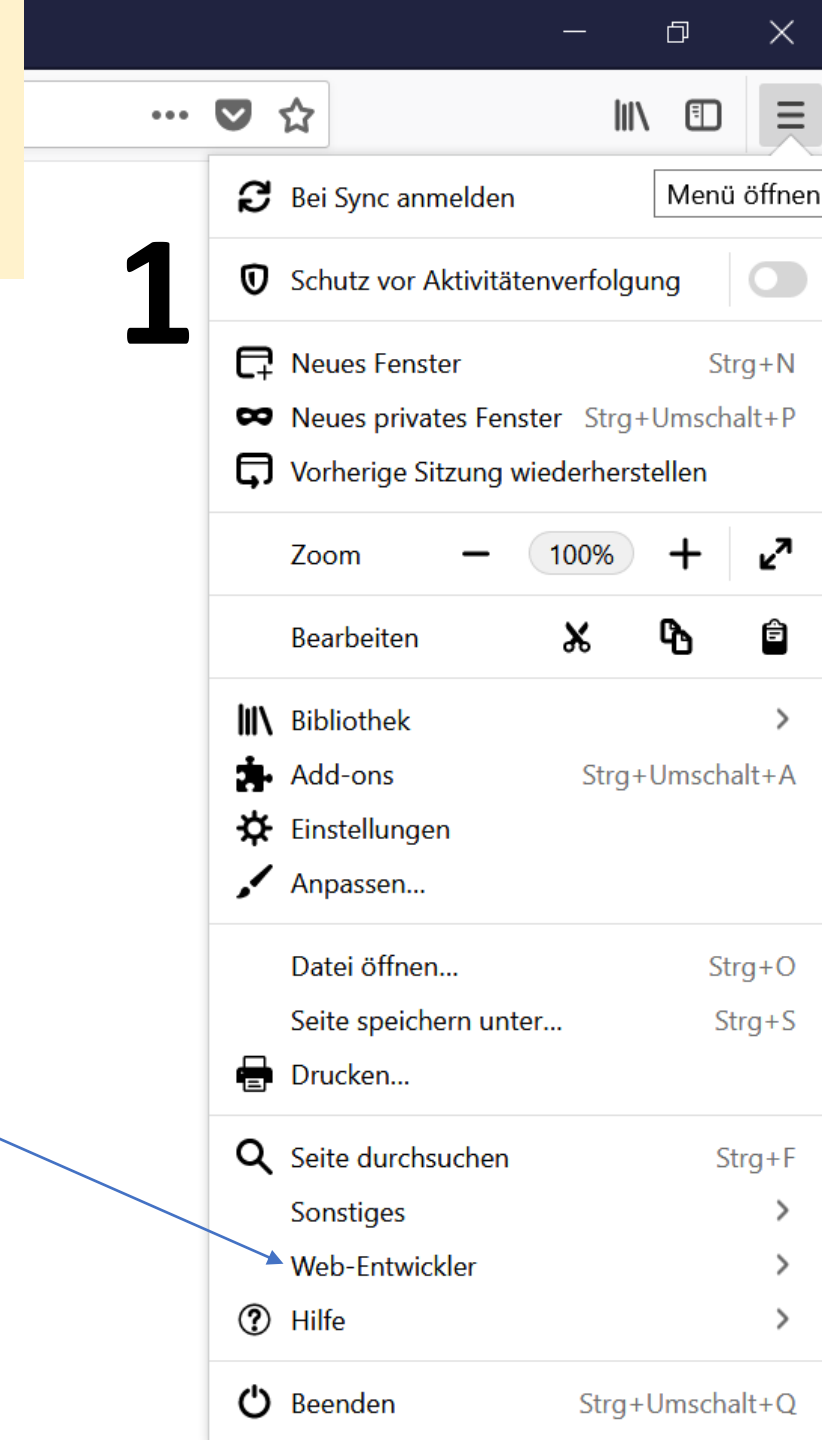
2



Hier klicken

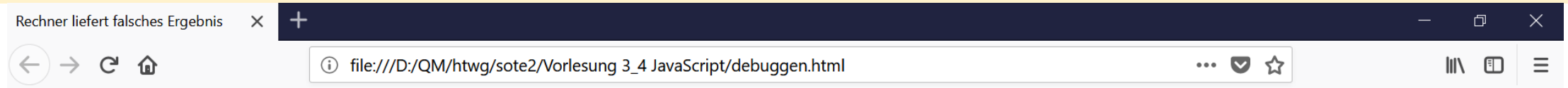
Hier klicken

1



# JavaScript Debuggen

Beispiel Firefox



Zahl 1  
1. Faktor

Zahl 2  
2. Faktor

berechne

Brakepoint



# JavaScript Object Notation (JSON)

[https://www.w3schools.com/js/js\\_json\\_intro.asp](https://www.w3schools.com/js/js_json_intro.asp)

- JSON ist ein leichtgewichtiges, textbasiertes Datenaustauschformat
- Es gibt JavaScript-Funktionen, um JSON zu parsen
- Es gibt JavaScript-Funktionen, um Objekte in JSON umzuwandeln

// Wandelt einen JSON-String in ein Objekt um

```
var obj = JSON.parse('{ "name":"John", "age":30, "city":"New York"}');
```

// Wandelt ein Objekt in einen JSON-String um:

```
var myJSON = JSON.stringify(obj);
```

- File-Type (Extension): ".json"
- MIME-Type ist "application/json"

# JSON: Syntax

## Syntax:

- Daten werden in Name-Wert-Paare gespeichert
- Die Namen werden immer in Anführungszeichen gesetzt
- Die Daten werden durch Komma separiert
- Objekte werden in geschweifte Klammern gesetzt
- Arrays werden in eckige Klammern gesetzt
- Mit JSON ist eine beliebig tiefe Verschachtelung möglich

## Datentypen:

- String
- Number
- object (JSON object)
- Array
- Boolean
- null

## JSON (Object)

```
var person = { "name":"John", "age":30 }
```

## JavaScript (Object)

```
var person = { name:"John", age:30 }
```

Zugriff wie bei JavaScript-Objekten:

```
person.name; oder person["name"];
```

# JSON vs. XML

## JSON:

**Array** 

```
{ "employees": [  
  { "firstName": "John", "lastName": "Doe" },  
  { "firstName": "Anna", "lastName": "Smith" },  
  { "firstName": "Peter", "lastName": "Jones" }  
]
```

## XML:

```
<employees>  
  <employee>  
    <firstName>John</firstName>  
    <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName>  
    <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName>  
    <lastName>Jones</lastName>  
  </employee>  
</employees>
```

# Asynchronous JavaScript And XML (AJAX)

[https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)

AJAX ist keine Programmiersprache.

AJAX ist eine Kombination von:

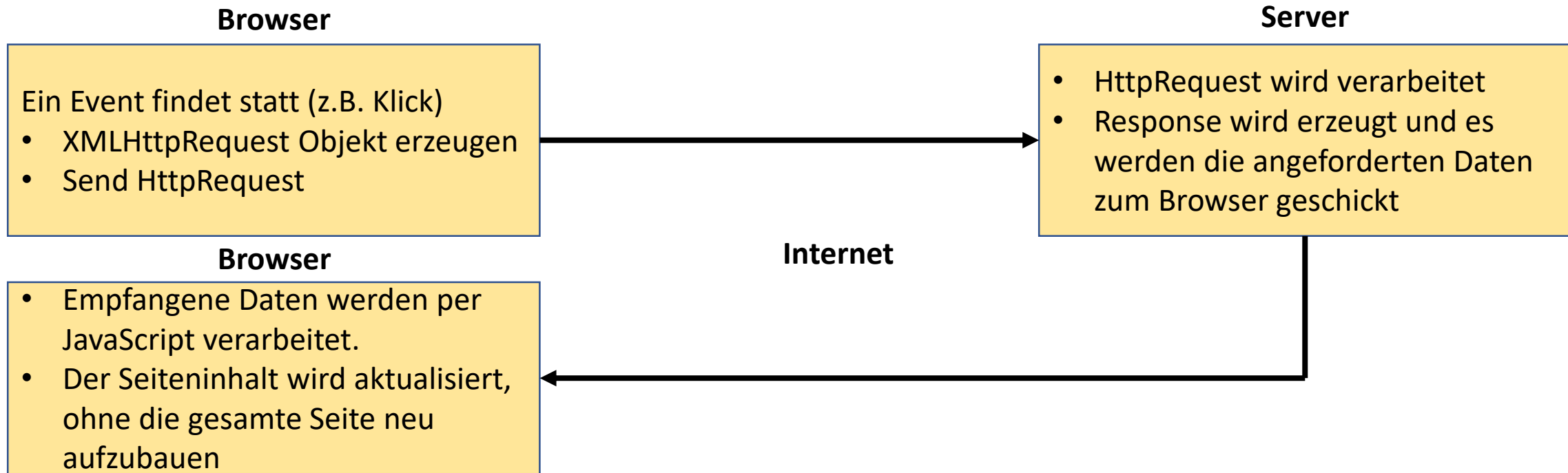
- Ein XMLHttpRequest-Objekt, das im Browser eingebaut ist (um einen Webserver anzufragen)
- JavaScript und HTML DOM (um Daten anzuzeigen, ohne die Seite neu zu laden)

Der Name AJAX ist irreführend. AJAX kann XML verwenden, um Daten zu senden, aber es ist genauso üblich JSON dazu zu verwenden.



# AJAX

[https://www.w3schools.com/xml/ajax\\_intro.asp](https://www.w3schools.com/xml/ajax_intro.asp)



1. Ein Event findet in der Webseite statt (z.B.: Ein Button wird angeklickt)
2. Ein XMLHttpRequest-Objekt wird mit JavaScript erzeugt
3. Das XMLHttpRequest-Objekt sendet einen Request zum Webserver
4. Der Server bearbeitet den Request
5. Der Server sendet einen Response mit Daten zurück zum Browser (gleiche Webseite)
6. Der Response wird mit JavaScript verarbeitet
7. Eine entsprechende Aktion (z.B. Update von Daten) wird mit JavaScript auf der Seite ausgeführt, ohne diese neu zu laden.

# Beispiel: AJAX

XMLHttpRequest-Objekt: [https://www.w3schools.com/xml/ajax\\_xmlhttprequest\\_create.asp](https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp)

```
<!DOCTYPE html>
<html>
<body>

<h2>Make a table based on the value of a drop down menu.</h2>

<select id="myselect" onchange="change_myselect(this.value)">
<option value="">Choose an option:</option>
<option value="customers">Customers</option>
<option value="products">Products</option>
<option value="suppliers">Suppliers</option>
</select>
```

Event, der AJAX hier auslöst

```
<p id="demo"></p>
```

```
<script>
function change_myselect(sel) {
  var obj, dbParam, xmlhttp, myObj, x, txt = "";
  obj = { "table":sel, "limit":20 };
  dbParam = JSON.stringify(obj);
  xmlhttp = new XMLHttpRequest();
  xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      myObj = JSON.parse(this.responseText);
      txt += "<table border='1'>"
      for (x in myObj) {
        txt += "<tr><td>" + myObj[x].name + "</td></tr>";
      }
      txt += "</table>"
      document.getElementById("demo").innerHTML = txt;
    }
  };
  xmlhttp.open("POST", "json_demo_db_post.php", true);
  xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
  xmlhttp.send("x=" + dbParam);
}
</script>
```

Das hier läuft zuerst ab!

Anonyme Funktion, die der Property „onreadystatechange“ (Event) zugewiesen wird.

Check, dass Response vollständig empfangen wurde. „this“ bezieht sich auf das xmlhttp-Objekt.

Das hier läuft zuerst ab!

readyState	Status der XMLHttpRequests:
0:	request not initialized
1:	server connection established
2:	request received
3:	processing request
4:	request finished, and response is ready

Funktion, die mittels des Events aufgerufen wird und als Parameter den ausgewählten Wert aus der Select-Box bekommt.

JSON-String, der als Requestparameter mit POST weggeschickt wird.

Verarbeitung des Response.

Update der Seite

Abschicken des Requests

# JSON - AJAX: Beispiel

## dynamisches Laden einer HTML-Tabelle

[https://www.w3schools.com/js/tryit.asp?filename=tryjson\\_html\\_table\\_dynamic](https://www.w3schools.com/js/tryit.asp?filename=tryjson_html_table_dynamic)

# AJAX: Mehrere Calls auf einer Seite

```
loadDoc("url-1", myFunction1);
```

```
loadDoc("url-2", myFunction2);
```

```
function loadDoc(url, cFunction) {  
    var xhttp;  
    xhttp=new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            cFunction(this);  
        }  
    };  
    xhttp.open("GET", url, true);  
    xhttp.send();  
}
```

```
function myFunction1(xhttp) {  
    // action goes here  
}
```

```
function myFunction2(xhttp) {  
    // action goes here  
}
```

Call Back Functions:

Funktionen werden als Funktionsparameter übergeben:

myFunction1

myFunction2

Bei mehreren AJAX-Aufrufen auf einer Seite:

- Eine Funktion, die das XMLHttpRequest-Objekt erzeugt und aufruft und der man eine URL und eine Funktion übergeben kann.
- Zu jedem AJAX-Aufruf eine Call-Back-Funktion

# jQuery

<https://www.w3schools.com/jquery/default.asp>

jQuery ist eine JavaScript-Bibliothek  
jQuery vereinfacht JavaScript-Programmierung

## Einbinden ins HTML (3 Möglichkeiten):

1. Bibliothek herunterladen:

```
<head>  
  <script src="jquery-3.3.1.min.js"></script>  
</head>
```

2. oder aus CDN Google:

```
<head>  
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>  
</head>
```

3. bzw. CDN Microsoft:

```
<head>  
  <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.3.1.min.js"></script>  
</head>
```

- HTML/DOM Manipulation
- CSS Manipulation
- HTML Methoden für Events
- Effekte und Animationen
- AJAX
- Utilities

**Bemerkung:** Die komprimierte JS-Datei nutzen (jquery-3.3.1.min.js) schneller im Download als unkomprimierte Datei (jquery-3.3.1.js)!

# jQuery: Syntax

[https://www.w3schools.com/jquery/jquery\\_ref\\_overview.asp](https://www.w3schools.com/jquery/jquery_ref_overview.asp)

Basis-Syntax:

`$(selector).action()`

Selektor-Element: `$()`

<code>\$(document)</code>	Das gesamte DOM
<code>\$("p:first")</code>	Das erste p-Element
<code>\$(":button")</code>	Alle Button-Elemente
<code>\$("tr:even")</code>	Alle geraden Zeilen einer Tabelle
<code>\$("tr:odd")</code>	Alle ungeraden Zeilen einer Tabelle
<code>\$("#myId")</code>	Element mit Attribut id="myId"
...	

Aktion `.action()`

`$("p:first").hide()` //Effekt  
`$(":button").fadeIn()` //Effekt  
`.text()` //Wert auslesen  
`.html()` //Wert auslesen  
`.val()` //Wert auslesen  
`.append()` //DOM-Manipulation  
`.remove()` //DOM-Manipulation  
`.next()` //Navigation durch DOM  
`.load(URL,data,callback)` //AJAX  
`$.get(URL,callback)` //AJAX  
`$.ajax({...})` //AJAX

...

**Beispiel:**

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").hide();
    });
});
```

# jQuery: Events

[https://www.w3schools.com/jquery/jquery\\_ref\\_events.asp](https://www.w3schools.com/jquery/jquery_ref_events.asp)

## Einige Events:

hover()  
keypress()  
keyup()  
mousedown()  
mouseup()  
ready()  
resize()  
scroll()  
submit()  
...

Event

```
$("#p").click(function(){  
    // action goes here!!  
});
```

## Achtung:

Kein Leerzeichen  
dazwischen!!!

## ACHTUNG: DOM muss geladen sein!!!

=> jQuery immer einbetten in:  
\$(document).ready(function(){  
 ...  
})

oder kürzer:

```
$(function(){  
    ...  
});
```

## Mehrere Eventhandler für ein Element einrichten:

```
$("#p").on({  
    mouseenter:function(){  
        $(this).css("background-color", "lightgray");  
    },  
    mouseleave:function(){  
        $(this).css("background-color", "lightblue");  
    },  
    click:function(){  
        $(this).css("background-color", "yellow");  
    }  
});
```

# jQuery: AJAX, get(...)

Event **ready** vom *DOM*

Event **click** von  
Element *button*

Anonyme Funktion  
für das **click** mit  
AJAX-Call **get**

Anonyme Funktion  
für das **get** mit der  
Response-Liste  
*notes*

key ist laufende  
Nummer, note ist  
das Objekt

jQuery-Schleife  
mit anonymer  
Funktion

Funktion zur dynamischen Erzeugung  
der Tabelle anhand der Notes-Liste

Properties des  
Objekts entspricht  
Tabellenspalte

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.get("http://localhost:8080/easynotes/api/notes", function(notes){
      createTable(notes);
    });
  });
});

function createTable(notes) {
  var txt="<table border='1'><tr><th>Bemerkung</th><th>Inhalt</th><th>created</th></tr>"
  $.each( notes, function( key, note ) {
    txt+="<tr><td>"+note.title+"</td><td>"+note.content+"</td><td>"+note.created+"</td></tr>";
  });
  txt+="</table>"
  $("#paragraph").html(txt);
}
</script>
<title>AJAX-Example</title>
</head>
<body>

<button>Send an HTTP GET request to a page and get the result back</button>
<p></p>

<p id="paragraph"></p>

</body>
</html>
```

id	title	content	created	updated
1	Bemerkung1	Inhalt1	2018-07-23 17:49:52	2018-07-24 13:01:09
2	Bemerkung2	Inhalt2	2018-07-24 13:00:52	2018-07-24 13:01:03
3	Bemerkung3	Inhalt3	2018-07-24 13:02:01	2018-07-24 13:02:01
4	Bemerkung4	Inhalt4	2018-07-24 13:02:29	2018-07-24 13:02:29



# jQuery: AJAX, \$.ajax({...})

Event **ready**  
vom *DOM*

Event **click** von  
Element *button*

`$.ajax({name:value,name:value,...})`

Anonyme Funktion  
für den Erfolgsfall  
mit der Response-  
Liste *notes*

```

<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $.ajax({url: "http://delia.sybit.de:8080/easynotes/api/notes",
      error: function(e){
        alert("Ein Fehler ist aufgetreten: " + e.status + " " + e.statusText);
      },
      success: function(notes){
        createTable(notes);
      }
    });
  });
});

```

Anonyme Funktion  
für das **click** mit  
AJAX-Call

Anonyme Funktion  
für den Fehlerfall

Funktion zur dynamischen Erzeugung  
der Tabelle anhand der Notes-Liste,  
wie in der Folie zuvor

# jQuery: AJAX, \$.ajax({...}), Syntax

## *\$.ajax({name:value, name:value, ... })*

Name	Value/Description
async	A Boolean value indicating whether the request should be handled asynchronous or not. Default is true
beforeSend( <i>xhr</i> )	A function to run before the request is sent
cache	A Boolean value indicating whether the browser should cache the requested pages. Default is true
complete( <i>xhr,status</i> )	A function to run when the request is finished (after success and error functions)
contentType	The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded"
context	Specifies the "this" value for all AJAX related callback functions
data	Specifies data to be sent to the server
dataFilter( <i>data,type</i> )	A function used to handle the raw response data of the XMLHttpRequest
dataType	The data type expected of the server response.
error( <i>xhr,status,error</i> )	A function to run if the request fails.
global	A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true
ifModified	A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false.
jsonp	A string overriding the callback function in a jsonp request
jsonpCallback	Specifies a name for the callback function in a jsonp request
password	Specifies a password to be used in an HTTP access authentication request.
processData	A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true
scriptCharset	Specifies the charset for the request
success( <i>result,status,xhr</i> )	A function to be run when the request succeeds
timeout	The local timeout (in milliseconds) for the request
traditional	A Boolean value specifying whether or not to use the traditional style of param serialization
type	Specifies the type of request. (GET or POST)
url	Specifies the URL to send the request to. Default is the current page
username	Specifies a username to be used in an HTTP access authentication request
xhr	A function used for creating the XMLHttpRequest object

# JS Tutorial

(zum Selbststudium)

In dieser Vorlesung konnte JS nur angerissen werden. Das unten erwähnte Tutorial kann als Nachschlagewerk dienen. Prima sind die Beispiel mit „Try it yourself“.

<https://www.w3schools.com/jS/default.asp>