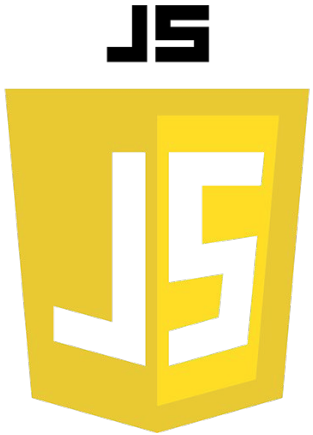
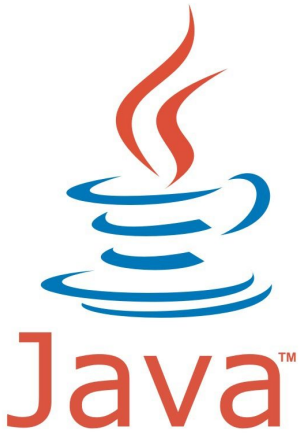
JavaScript

Eine kurze Einführung


JS
JS

≠


Java™

Geschichte

Der Vater von JS

Brendan Eich (1961)

1996 entwickelt Mocha
für Netscape 2.0 Beta
in 10 Tagen

Name
Mocha → LiveScript → JavaScript

Was ist JavaScript?

Script Language

JavaScript Engine im Browser

Java like Syntax

Die Idee hinter JavaScript

Wir wollen statische Webseiten mit dynamischem Inhalt zu schmücken

Ursprung & Entwicklung

1995	von Netscape (später Mozilla) eingeführt und lizenziert unter ECMA Standard Microsoft bringt JScript als Erweiterung
1998	W3C veröffentlicht ersten ECMAScript Standard
1998 - 2004	Ausbreitung schleppend und chaotisch

AJAX Revolution

Asynchronous **J**avaScript **A**nd **X**ML

1998	Konzept für XMLHttpRequest
1999	MS Internet Explorer 5.0 enthält Vorgänger
2000	MS Outlook Web App
2004	Aufstieg von JavaScript u.a. dank Google (AJAX für Suche)
2006	W3C veröffentlicht ersten offiziellen Standard
2009	ECMAScript 5 Node.js als Web-Server entwickelt

JQuery - Der (Gross)Vater der Libraries

26.August 2006 wird
Version 1.0 veröffentlicht

```
jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.
```

Eine der häufigst genutzten Libraries im Web (immer noch)

Aktuelle Version 3.3 (released am 19. Januar 2018)

https://jquery.com/
https://www.w3schools.com/jquery/

Heute

Anwendungsgebiete

Eine der **wichtigsten Scriptsprachen / Webtechnologien**

Wird von **allen Browsern** unterstützt (ECMAScript 5)

Entwickelt sich **rasant schnell** und hat eine **riesen Community** weltweit

Bewegt sich hin zur **Full-Stack Webtechnologie** (Node.js)

Vielzahl an Packages, Libraries & Frameworks

- *Microservices, Package Management*
- *React.js, Vue.js etc. für GUI Komponenten*
- *GraphQL für REST-APIs etc.*

Konkurrenziert **native Programmiersprachen** für Mobile Devices

Frameworks, Frameworks, Frameworks

Bootstrap.js

Express.js

Vanilla.js

Angular2

React & Vue

Meteor

...

Packages, Packages, Packages

“Don’t implement what has already been implemented”

JS Packages (z.B. JQuery) einbinden und Funktionen nutzen

Package Manager helfen beim Installieren, der Versionierung und dem Deployment (Bundle)

Herausforderungen

- Standards (Browser Kompatibilität)
- Stabilität (Package Kompatibilität)
- Performance (Client-Side Single-Threaded Prozess)
- Ladezeiten (Webseiten enthalten zu viel Code und brauchen lange zum laden)
- Mobile Device Unterstützung

Der Code

Hello World

```
<!DOCTYPE HTML>
<html>
  <head>
     <title>Hello World</title>
  </head>

  <body>
    <script type="text/javascript">
      // Define a function
      var helloWorld = function() {
        alert( 'Hello, world!\nI am Javascript' );
      }
      // and now let's call the function
      helloWorld();
    </script>

    <p>I am HTML</p>
  </body>
</html>
```

JS in HTML einbinden

Im Body
Als Code

```
<!DOCTYPE HTML>
<html>
  <body>
    <script type="text/javascript">
      // JavaScript Code
    </script>

    ...
  </body>
</html>
```

Im Head
Als externes Script

```
<!DOCTYPE HTML>
<html>
  <head>
    ...
    <script src="script.js" type="text/javascript"></script>
  </head>

  <body>
    ...
  </body>
</html>
```

Everything is an Object

Keine Klassen, nur Objekte

- Funktionen sind Objekte
- prototype als “Ersatz” ~~für Klassen~~
- Duck-Typing
  (dynamische Type Definitionen zur Laufzeit)

If it walks like a duck and it quacks like a duck, then it must be a duck.

JavaScript Basics

```
// String
let text = 'My Text'
let textConcat = `In this line ${text} is included`

// Number
let number = 1234
let square = number * number

// Constant
const PI = 3.141


// Array
const list = ['abc', 'cde', 'fgh']
list.concat(['ijk', 'lmn'])
list.push('opq')
list.map(block => block.toUpperCase())
list.slice(1,3)
list.sort(...)
list.filter(...)
list.reduce(...)
// ...
```

```
// Object
const person = {
  // Value property
  name: 'Peter Krause',
  age: 42,

  // Function property
  eat: function (food) {
    console.log(`Peter eats ${food}`)
  },
}


// Access properties and functions from object
console.log(`Name: ${person.name} (${person.age})`)
person.eat('Steaks')


// Copy Object
let personCopy = Object.assign({}, person)
personCopy = { ...person }
```

Function Definition

```
// Function Definition
const myFunction = function(text) {
  return text.toUpperCase()
}


// Arrow Functions
const myFunction = text => {
  return text.toUpperCase()
}

const myFunction = text => text.toUpperCase()


// Anonymous arrow Functions
(text => text.toUpperCase())('text to transform')
```

Global Context

```
// Window & Document
window.scrollTo(0, 1000)

document.getElementById('todo-list')

// Global Function (Object)
function myGlobalFunction(text) {
  return text.toUpperCase()
}
window.myGlobalFunction('text in kleinbuchstaben')

// Global Object
function Person(name, age) {
  this.name = name
  this.age = age
  this.eats = function(food) {
    console.log(`${this.name} eats ${food}`)
  }
}

let personObject = new Person('Max Muster', 33)
personObject.eats('fish & chips')
```

Prototype

No classes just objects

Prevent inheritance issues

Objects have a prototype chain (pass functions from one prototype to another)

```
// Person has a name and can eat
function Person(name) {
  this.name = name
}
Person.prototype.eats = function(food) {
  console.log(`${this.name} eats ${food}`)
}


// A woman has a specific gender and a function buy shoes
function Woman(name) {
  this.name = name
  this.gender = 'Female'
}
Woman.prototype.buysShoes = function(swissFrancs) {
  console.log(`${this.name} buys shoes for ${swissFrancs}.- Fr`)
}


// User prototype chaining to make Woman a Person too
Woman.prototype.__proto__ = Person.prototype


const heidi = new Woman('Heidi')
heidi.eats('Rüeblichueche') // Now, woman can eat
heidi.buysShoes(180) // And buy shoes
```

Use with care!

Only override your OWN objects

Do **NOT** override predefined Prototypes (e.g. Event)

DOM API

Document **O**bject **M**odel

- Provides a Application Program Interface (DOM API)
- JavaScripts manipulates the DOM
- HTML, CSS, SVG, XML
- Add or remove tags
- Add, remove or update attributes
- Handle events (click, scroll, input changes, etc.)
- ...

jQuery

DOM Interaction Simplifier

JQuery = $

jQuery is a JavaScript Library.

jQuery greatly simplifies JavaScript programming.

jQuery is easy to learn.

- Aktuelle stabile Version 3.3
- Einbindung über ***<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>***

```
<html>
  <head>
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  </head>

  <body>
    <script>
      console.log('jQuery Object', jQuery)
      console.log('$ Object', $)
    </script>
  </body>
</html>
```

jQuery Beispiel

```
<script type="text/javascript">
  var count = 0

  // Wait for DOM to be loaded
  $(document).ready(function(){

      // Set initial text
      $("#text").text('The button has not been clicked.');

      // On click, raise counter by 1, set new textes and change the text color
      $("#countButton").click(function(){
          count += 1;
          $("#text").text('The button has been clicked').addClass('blue');
          $("#boldText").html(`<b>${count} times</b>`).addClass('red');
      });
  });
</script>
```

jQuery Functions

```
// Select DOM element
const h1Element = $('h1')

// Get the first element
const firstElement = $('p').first()

// Set text of element
firstElement.text('Changed text for element')

  // Create an element
const textElement = $('<p>')

// Set attribute for element
textElement.attr("data-hook", "newText");

// Set text to an element
textElement.text('This is a new <p> element')

// Append element to DOM
$('body').append(textElement)
```

```
// Check if the element exists
if ($('p#text').length > 0) {
  console.log('<p> Element exits')
}

// Remove llst <p> element from DOM
$('p#text').remove()

// Append items with inner HTML content to a list
const list = $('#list')
const item = $('<li>')
const itemContent = $('<h3>')
itemContent.text('I am an Item Text')
item.html(itemContent)
list.append(item)

// Access CSS Object Model and change styles
$('h1').css('color', 'blue')
$('h1').css('font-size', '72px')

// Add a class to an object
$('p[data-hook=newText]').last().addClass('mark-text');
```

Event Handling

```
// Attach an event listener
$('#animateCircle').on('click', () => {
  // do something
})

// Attach a click listener
$('#myButton').click((event) => {
  // do something with event
  // e.g. console.log(event.target.clientX)
})

// Remove/Deactivate an event listener
$('#myButton1').off('click')

// Attach an event listener over document
$(document).on('click', '#myButton', () => {
    // ... do something
})
```

```
<body>
  <ul id="todo-list" class="list">
    <!-- Todos come here -->
  </ul>
  <form id="new-todo-form">
    <input type="text" id="new-todo-text" />
    <input type="submit" value="Add todo" />
  </form>

  <script>
     $('#new-todo-form').on('submit', (event) => {
       event.preventDefault();

       // Get input fields value
       const newTodoInput = $('#new-todo-text')
       const newTodoValue = newTodoInput.val()

       // Add new todo to list
       if (newTodoValue.trim() > '') {
         const todoList = $('#todo-list')
         const todoItem = $('<li>')
         todoItem.text(newTodoValue)
         todoList.append(todoItem)
       }

       // Set input value to empty string
       newTodoInput.val('');
     });
   </script>
 </body>
```

Und jetzt wie weiter?

Beispiele

Einfache JavaScript Beispiele
https://github.com/joelmeiller/fhnw-ws3-javascript-introduction

Todo-App (aus letztem Workshop)
https://github.com/bierik/jquery-todo

Tipps & Tricks

Learning by doing

Dev-Tools helfen (Google Chrome Browser bietet sehr gute Unterstützung)

CSS vor JS (z.B. Animations)

Libraries & Frameworks verwenden (wie z.B. jQuery oder Bootstrap.js)

Ignoriert Data-Processing für Prototyping (Mocking, Statische Daten)

Web-Dokumentation zu verstehen braucht Übung

Fun Fun Function

Youtube Channel
https://www.youtube.com/channel/UCO1cgjhGzsSYb1rsB4bFe4Q

Mein Liebling → Composition over Inheritance
https://www.youtube.com/watch?v=wfMtDGfHWpA

Libraries & Links

jQuery API Documentation
https://api.jquery.com/

AOS Library (Scrolleffekte)
https://michalsnik.github.io/aos/

D3.js Charts & Graphs
https://d3js.org/

Velocity (Advanved Animations)
http://velocityjs.org/

Sanfter Seitenwechsel (Advanced)
https://css-tricks.com/add-page-transitions-css-smoothstate-js/

Referenzen

Wikipedia
https://en.wikipedia.org/wiki/JavaScript

Auth0 Blog about history of JavaScript
https://auth0.com/blog/a-brief-history-of-javascript/

W3Schools
https://www.w3schools.com/jsref/

Mozilla Developper Network (MDN)
https://developer.mozilla.org/en-US/docs/Web/JavaScript

jQuery Homepage
https://jquery.com/