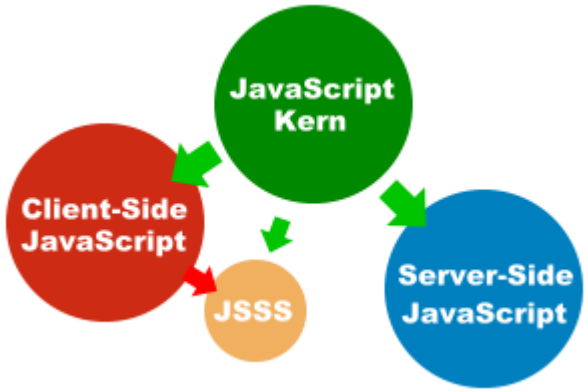
Was ist JavaScript?

JavaScript ist eine von Netscape (1995) geschaffene Plattform, welche das Erstellen von Anwendungen und Dokumenten, die über das Internet laufen/gestartet werden können ermöglicht. Zudem bietet es Möglichkeiten aktiv auf Dokumente einzugreifen und diese zu lesen, zu schreiben oder zu verändern. Letztlich bildet es zusammen mit HTML und CSS die Grundlage für dynamisches HTML (DHTML, engl. dynamic HTML).

JavaScript wird aufgeteilt in 2 Bereiche (oder besser Varianten): Client-Side JavaScript und Server-Side JavaScript. Client-Side JavaScript enthält dabei, neben dem Grundgerüst noch einen neueren kleinen Teil der sich JavaScript Style Sheets (JSSS) nennt. Die folgende Grafik soll diese Unterteilungen nochmals verdeutlichen:



Gerade serverseitiges JavaScript at sich in den vergangenen Jahren stark verbreitet. Durch serverseitiges JavaScript kann z. B. das Verhalten eines Browsers simulieren oder eben serverseitige Aufgaben einer Website erledigen.

JavaScript-Bibliotheken und –Frameworks

Der Kern von JavaScript ist eine eher schlichte Programmiersprache, die sich besonders für den Einsatz innerhalb des Browsers anbietet. Probleme bereitet vielen Programmierern jedoch die Schnittstelle zur Website: das DOM (Document Object Model). Genau hier setzen JavaScript-Frameworks und -Bibliotheken an: Sie bieten Entwicklern Arbeitserleichterungen in diesem und anderen Feldern der Programmierung. Wir werden uns hier auf das weitverbreitete jQuery-Framework konzentrieren.

JQuery

Die umfangreiche Bibliothek jQuery ist die am meisten genutzte JavaScript-Library. Dies liegt unter anderem daran, dass man mit ihr einfach browserübergreifend Code schreiben kann und für sie sehr viele Plugins existieren. Die quelloffene Bibliothek ist Bestandteil vieler Content-Management-Systeme (CMS) wie WordPress, DruPal oder Joomla. Aber auch Web-Frameworks wie **Bootstrap** greifen auf jQuery zurück. jQuery dient vor allem als komfortable **Schnittstelle zum DOM** und bietet

hierbei diverse Funktionen: Mit **CSS3-Selektoren** lassen sich Webseiten-Elemente sehr unkompliziert auswählen und manipulieren. Besonders geschätzt wird jQuery auch aufgrund der Möglichkeit, Ajax-Anfragen (HTTP-Anfragen ohne das erneute Laden der Webseite) einzubinden.

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

```
<script src="https://code.jquery.com/jquery-latest.js"></script>
```

JQuery UI

Bei jQuery UI handelt es sich um eine freie Erweiterung für jQuery. Sie dient der Gestaltung und Umsetzung einer Benutzeroberfläche (englisch: „User Interface“ oder abgekürzt „UI“) – beispielsweise von Webseiten oder Web-Apps Der Schwerpunkt liegt auf der **einfachen Gestaltung von Effekten und Interaktionen**. Zum Funktionsumfang von jQuery UI zählen die Umsetzung von Interaktionsmöglichkeiten (wie Drag-and-drop, Vergrößerung bzw. Verkleinerung von Webseitenelementen etc.), Animationen und Effekten sowie Widgets (wie Autocomplete, Slider, Datepicker etc.). Über den Grafikeditor ThemeRoller können eigene Themes erstellt, aber auch bereits vorhandene verwendet und angepasst werden – durch den modularen Aufbau werden hierbei auch nur benötigte Komponenten implementiert.

Weitere bekannte Bibliotheken und Frameworks

React, Zepto, CreateJS, AngularJS, Angular, Ember.js, Vue.js, Meteor, und weitere ….

Wie bindet man Javascript in die Webseite ein?

Ähnlich wie bei CSS in HTML, wird auch Javascript eingebunden. Javascript kann entweder im <head> oder im <body> angegeben werden. Oftmals stellt sich auch die Frage ob am Beginn oder Am Ende der HTML Seite. Wenn es sich um Funktionalitäten handelt die erst nach dem vollständigen Laden der HTML-Seite erfolgen(und das ist meist der Fall) dann sollte man Javascript am Ende einbinden. Oftmals können die Javascript-Files sehr groß sein (zB Bootstrap), was sich negativ auf die Seitenaufbauzeit auswirkt.

INLINE SCRIPT

```
<script language="JavaScript">
 <!--
  … JavascriptCode
-->
</script>
```

Das language-Argument muss nicht zwingend angegeben werden.

EXTERNES SCRIPT

Die Einbindung über externe .js-Files ist der gängigste und auch am meisten empfohlene Weg. Diverse Libaries wie jQuery, bootstrap.js, … werden auch über **ContentDeliveryNetworks** wie (cloudflare) angeboten. Die Verwendung von CDNs trägt positv zur Lastverteilung und zu den Seitenladezeiten bei, erzeugt jedoch auch eine Gewisse Abhängigkeit vom Anbieter.

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

Kombination von Javascript und jQuery

Ein gängiges Einsatzgebiet ist die Kombination von reinem JavaScript und jQuery. Je nachdem was für die jeweilige Aufgabenstellung besser geeignet ist.

ZB.: Wenn das HTML-Dokument fertig geladen ist, alle Datepicker auf der Seite initialisieren.

JQuery-Code einbinden

Haben wir nun die jquery.min.js eingebunden, können wir jQuery nutzen. Dazu wird im <head>-Bereich der gewünschte jQuery-Code eingebunden:

```
<script> /* Hier der jQuery-Code */ </script>
```

Um zu gewährleisten dass für die Verwendung von jQuery alle Elemente der Seite fertig geladen sind, wird der jQuery-Code üblicherweise in einer document-Ready-Bedingung abgehandelt. (Ausser die Funktionalität soll mitels einem Event wie onclick, onsubmit erfolgen)

```
<script>
  $(document).ready(function(){
    /* Hier der jQuery-Code */ });
</script>
```

JQuery – Selectoren

JQuery-Selector orientieren sich am CSS3-Standard und ermöglichen das Auffinden und Manipulieren von HTML-Elementen. Die Selektoren können Element-Namen, -Ids, -Klassen, -Typen, -Attribute, Werte von Attributen und vieles mehr beinhalten.

jeder Selector in jQuery beginnt mit einem $();

Element-Selector

Selektiert mittels angegebener HTML-Tags innerhalb des DOM.

$(“p”); //alle Paragraphs der Seite

Id-Selector

Greif über eine eindeutige ID auf den DOM-Baum zu.

$(“#list_results”)

Class-Selector

Sucht innerhalb des DOM nach Elementen welchen die angegebene Klasse zugeordnet ist.

$(“.line”)

Weitere Selektoren

Syntax	Description	Example
$("*")	Alle Elemente	
$(this)	das aktuelle Element	
$("p.intro")	alle <p> mit der Klasse class="intro"	
$("p:first")	Das Erste<p> element	
$("ul li:first")	Das Erste element vom Ersten	
$("ul li:first-child")	Das erste element von jedem	
$("[href]")	Alle Elemente mit einem href Element	
$("a[target='_blank']")	alle <a> Elemente wo der target-Attribut-Wert gleich "_blank" ist.	
$("a[target!='_blank']")	alle <a> Elemente wo der target-Attribut-Wert nicht gleich "_blank" ist.	
$(":button")	alle <button> Elemente und <input> Elemente mit der type="button"	

JQuery – DOM-Manipulation

Mit JQuery kann man sehr einfach auf Ereignisse reagieren und den Inhalt des DOM verändern. ZB den Inhalt eines Div-Elementes ersetzen, entfernen, aktualisieren.

3 einfach anzuwendende Funktionen sind:

- .text() - setzt oder retourniert den Text-Inhalt eines Elements
- .html() - setzt oder retourniert den Html-Inhalt eines Elementes (zB im Zusammenspiel mit ajax-Requests)
- .val() - setzt oder retourniert den Inhalt von Formular-Feldern

Beispiel: gibt den Inhalt des Formular-Feldes zurück wenn der #button geklickt wurde

```
$("#button").click(function(){
  alert("Value: " + $("#vorname").val());
});
```

JQuery Anwendungsbeispiele

- HTML-Code on the fly tauschen (Jumbotron gegen eine Bootstrap-Message)
- Options für eine Selectbox tauschen.
- CSS-Styles ändern
- Document.ready();
- Validierung von Formularen
- DOM-Manipulationen
- … u.v.m.