**Javascript:**

Finding HTML Elements

document.getElementById(id) Find an element by element id

document.getElementsByTagName(name) Find elements by tag name

document.getElementsByClassName(name) Find elements by class name

Changing HTML Elements

element.innerHTML = new html content Change the inner HTML of an element

element.attribute = new value Change the attribute value of an HTML element

element.setAttribute(attribute, value) Change the attribute value of an HTML element

element.style.property = new style Change the style of an HTML element

Adding and Deleting Elements

document.createElement(element) Create an HTML element

document.removeChild(element) Remove an HTML element

document.appendChild(element) Add an HTML element

document.replaceChild(element) Replace an HTML element

document.write(text) Write into the HTML output stream

Adding Events Handlers

document.getElementById(id).onclick = function(){code} Adding event handler code to an onclick event

Finding HTML Objects

document.anchors Returns all <a> elements that have a name attribute

document.applets Returns all <applet> elements (Deprecated in HTML5)

document.baseURI Returns the absolute base URI of the document

document.body Returns the <body> element

document.cookie Returns the document's cookie

document.doctype Returns the document's doctype

document.documentElement Returns the <html> element

document.documentMode Returns the mode used by the browser

document.documentURI Returns the URI of the document

document.domain Returns the domain name of the document server

document.domConfig Obsolete. Returns the DOM configuration

document.embeds Returns all <embed> elements

document.forms Returns all <form> elements

document.head Returns the <head> element

document.images Returns all <img> elements

document.implementation Returns the DOM implementation

document.inputEncoding Returns the document's encoding (character set)

document.lastModified Returns the date and time the document was updated

document.links Returns all <area> and <a> elements that have a href attribute

document.readyState Returns the (loading) status of the document

document.referrer Returns the URI of the referrer (the linking document)

document.scripts Returns all <script> elements

document.strictErrorChecking Returns if error checking is enforced

document.title Returns the <title> element

document.URL Returns the complete URL of the document

Changing HTML Content

document.getElementById("p1").innerHTML = "New text!";

**Jquery:**

Selector Example Selects

\* $("\*") All elements

#id $("#lastname") The element with id="lastname"

.class $(".intro") All elements with class="intro"

.class,.class $(".intro,.demo") All elements with the class "intro" or "demo"

element $("p") All <p> elements

el1,el2,el3 $("h1,div,p") All <h1>, <div> and <p> elements

:first $("p:first") The first <p> element

:last $("p:last") The last <p> element

:even $("tr:even") All even <tr> elements

:odd $("tr:odd") All odd <tr> elements

:first-child $("p:first-child") All <p> elements that are the first child of their parent

:first-of-type $("p:first-of-type") All <p> elements that are the first <p> element of their parent

:last-child $("p:last-child") All <p> elements that are the last child of their parent

:last-of-type $("p:last-of-type") All <p> elements that are the last <p> element of their parent

:nth-child(n) $("p:nth-child(2)") All <p> elements that are the 2nd child of their parent

:nth-last-child(n) $("p:nth-last-child(2)") All <p> elements that are the 2nd child of their parent, counting from the last child

:nth-of-type(n) $("p:nth-of-type(2)") All <p> elements that are the 2nd <p> element of their parent

:nth-last-of-type(n) $("p:nth-last-of-type(2)") All <p> elements that are the 2nd <p> element of their parent, counting from the last child

:only-child $("p:only-child") All <p> elements that are the only child of their parent

:only-of-type $("p:only-of-type") All <p> elements that are the only child, of its type, of their parent

parent > child $("div > p") All <p> elements that are a direct child of a <div> element

parent descendant $("div p") All <p> elements that are descendants of a <div> element

element + next $("div + p") The <p> element that are next to each <div> elements

element ~ siblings $("div ~ p") All <p> elements that are siblings of a <div> element

:eq(index) $("ul li:eq(3)") The fourth element in a list (index starts at 0)

:gt(no) $("ul li:gt(3)") List elements with an index greater than 3

:lt(no) $("ul li:lt(3)") List elements with an index less than 3

:not(selector) $("input:not(:empty)") All input elements that are not empty

:header $(":header") All header elements <h1>, <h2> ...

:animated $(":animated") All animated elements

:focus $(":focus") The element that currently has focus

:contains(text) $(":contains('Hello')") All elements which contains the text "Hello"

:has(selector) $("div:has(p)") All <div> elements that have a <p> element

:empty $(":empty") All elements that are empty

:parent $(":parent") All elements that are a parent of another element

:hidden $("p:hidden") All hidden <p> elements

:visible $("table:visible") All visible tables

:root $(":root") The document's root element

:lang(language) $("p:lang(de)") All <p> elements with a lang attribute value starting with "de"

[attribute] $("[href]") All elements with a href attribute

[attribute=value] $("[href='default.htm']") All elements with a href attribute value equal to "default.htm"

[attribute!=value] $("[href!='default.htm']") All elements with a href attribute value not equal to "default.htm"

[attribute$=value] $("[href$='.jpg']") All elements with a href attribute value ending with ".jpg"

[attribute|=value] $("[title|='Tomorrow']") All elements with a title attribute value equal to 'Tomorrow', or starting with 'Tomorrow' followed by a hyphen

[attribute^=value] $("[title^='Tom']") All elements with a title attribute value starting with "Tom"

[attribute~=value] $("[title~='hello']") All elements with a title attribute value containing the specific word "hello"

[attribute\*=value] $("[title\*='hello']") All elements with a title attribute value containing the word "hello"

:input $(":input") All input elements

:text $(":text") All input elements with type="text"

:password $(":password") All input elements with type="password"

:radio $(":radio") All input elements with type="radio"

:checkbox $(":checkbox") All input elements with type="checkbox"

:submit $(":submit") All input elements with type="submit"

:reset $(":reset") All input elements with type="reset"

:button $(":button") All input elements with type="button"

:image $(":image") All input elements with type="image"

:file $(":file") All input elements with type="file"

:enabled $(":enabled") All enabled input elements

:disabled $(":disabled") All disabled input elements

:selected $(":selected") All selected input elements

:checked $(":checked") All checked input elements

**Javascript Frameworks:**

* Jquery
* Angular JS
* Aurelia
* Ember

JavaScript Bibliotheken die sorgen für Vereinfachung und Wiederverwendung.

Marktanteile:

* Jquery
* Bootstrap
* Modernizer
* MooToolsd
* ASP.NET Ajax

**Jquery:**

* jQuery 1.x wird von allen Webbrowser unterstützt
* jQuery 2.x nur für moderne Webbrowser(ab IE9.0)

Mit jQuery kann man HTML Elemente selektieren und Operationen durchführen.

jQuery ist stark dass es unter verschiedenen Browser richtig läuft.

Funktionalität ist mitteln Plugin Schnittstelle erweiterbar.

jQuery Funktionen werden wie Javascript in HTML eingebettet oder mit Link eingebunden.

jQuery Funktion ist als Javascript Code implementiert.

Bibliothek: <version<.min.js auf <http://jquery.com/>

Einbinden Lokal: <script type=“text/javascript“ src=“jquery-1.9.0.min.js“></script>

Zentrale Webserver von Content Delivery Network:

Google CDN: <script src=”http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js”>

Mircosoft CDN: <script src=”http://ajax.aspnetcdn.com/ajax/libs/jQuery/jquery-1.10.2.min.js”>

Wichtigste Konzept von jQuery ist die Verwendung von Elementselektoren.

Javascript: getElementByXX() jQuery: $(…) (eine oder mehrere Elemente können ausgewählt werden)

Beispiele:

\* $("\*") All elements

#id $("#lastname") The element with id="lastname"

.class $(".intro") All elements with class="intro"

.class,.class $(".intro,.demo") All elements with the class "intro" or "demo"

element $("p") All <p> elements

el1,el2,el3 $("h1,div,p") All <h1>, <div> and <p> elements

jQuery Code wird normalerweise in der $(document).ready Funktion gestartet.

Diese wird aufgerufen sobald das ganze HTML Dokument inklusive Javascript Dateien geladen sind.

**jQuery Traversierung:**

**Aufwärts:**

parent() Gibt den direkten Vater Elemente zurück.

parents() Gibt alle Vater Elemente(inkl. Vater-Vater usw.) zurück.

parentsUntil() Gibt alle Vater Elemente zurück zwischen den den angegenen Argument.

**Abwärts:**

children() Gibt alle direkten Kind Elemente zurück.

find() Gibt alle Kind Elemente vom angegebenen Element zurück.

**Seitwärts:**

siblings() Gibt alle benachbarten Elemente zurück.

next() Gibt den nächsten benachbarten Element zurück.

nextAll() Gibt alle benachbarten Elemente nach dem ausgewählten zurück.

nextUntil() Gibt alle benachbarten Elemente zwischen den den angegenen Argument zurück.

prev() Gibt den vorherig benachbarten Element zurück.

prevAll() Gibt alle benachbarten Elemente vor dem ausgewählten zurück.

prevUntil() Gibt alle benachbarten Elemente zwischen den den angegenen Argument zurück.

**Filter:**

first() Gibt das erste Element der selektierten Elemente.

last() Gibt das letzte Element der selektierten Elemente.

eq() Gibt das Element der selektierten Index zurück.

filter() Gibt das passende Element der selektierten Filter.

not() Gibt das nicht passende Element der selektierten Filter.

**jQuery DOM Manipulation:**

Get/Set Content

text() Textinhalte hinzufügen oder ändern

html() komplettes Element und Inhalte hinzufügen oder ändern(mit HTML Tags und Markups)

val() Wert von Formular hinzufügen oder ändern

**Set Attributen**

attr() Attributewert setzen oder ändern

**Add New HMTL Content**

append() Inhalt hinten anhängen.

prepend() Inhalt vorne anhängen.

after() Inhalte nach dem selektierten Element einfügen.

before() Inhalte vor dem selektierten Element einfügen

**Remove Elements/Content**

remove() Selektieres Element und darunterliegende Kindelemente entfernen.

empty() Nur die darunterliegende Kindelemente entfernen.

**jQuery Manipulating CSS**

addClass() Klasse dem selektieren Element zuordnen.

removeClass() Klasse dem selektieren Element wegnehmen.

toggleClass() Beim zuordnen/wegnehmen toggeln.

css() Style attribute setzen oder ändern.

**jQuery Forms:**

Validate() Selektierte Formular wird validiert.

Valid() Überprüft ob gesamtes valid.

Rules() Kann so eine neue Regel hinzufügen oder entfernen.

**Beispiel:**

|  |  |  |
| --- | --- | --- |
| Validate: | Valid: | Rules: |
| $(".selector").validate({  debug: true  }); | $( "button" ).click(function() {  alert( "Valid: " + form.valid() );  }); | $( "#myinput" ).rules( "add", {  minlength: 2  }); |

* required, remote, minlength, maxlength, rangelength, min, max,
* range, step, email, url, date, dateISO, number, digits, equalTo

**Google Maps:**

**Einbinden:**

**Mit Gratiskey:**

<script async defer

src="https://maps.googleapis.com/maps/api/js?key=YOUR\_API\_KEY&callback=initMap">

</script>

**Ohne Key:**

<script

src="https://maps.googleapis.com/maps/api/js?v=3.exp">

</script>

Div erstellen in welchem die MAP eingezeigt warden soll. <div id=“map“></div>

Map Object per Javascript erstellen.

Function initMap() {

var myLatLng = {lat: -25.363, lng: 131.044};

var map = new google.maps.Map(document.getElementById(‘map’), {

zoom: 4,

center: myLatLng

});

var marker = new google.maps.Marker({

position: myLatLng,

map: map,

title: ‘Hello World!’

});

}

**JSON:**

JSON = JavaScript Object Notation

Leichtgewichtiges Datenaustauschformat

Basiert auf der Object Literal Notation von JavaScript

Lesbar von Menschen

Wegen Einfachheit von vielen Programmiersprachen unterstützt

JSON ist eine syntax für das Speichern und Austauschen von Daten.

JSON ist ein ‘easier-to-use’ Alternative zu XML.

<http://www.JSON.org>

|  |  |
| --- | --- |
| **JSON Example** | **XML Example** |
| {"employees":[  {"firstName":"John", "lastName":"Doe"},  {"firstName":"Anna", "lastName":"Smith"},  {"firstName":"Peter", "lastName":"Jones"}  ]} | <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> |

**JQuery Ajax:**

AJAX = Asynchronous JavaScript and XML

Anwendungsbeispiele: Google Maps, Gmail, Youtube, Facebook

Konzept der asynchronen Datenübertragung zwischen Browser und Server

http Anfragen während HTML Seite angezeigt wird.

Seite kann verändert werden ohne die Seite komplett neu zu laden (partielle Updates).

jQuery bietet so eine einfache Möglichkeit um im Hintergrund Anfragen an den Server zu senden.

Verbindet die Technologien HTML, CSS und JavaScript

Webtechniken:

* HTML und XHTML
* Document Object Model zur Repräsentation der Daten oder Inhalte
* Javascript zur Manipulation des DOMs und zur dynamischen Darstellung der Inhalte.
* Javascript dient auch als Schnittstelle zwischen den Komponenten
* Das XMLHttpRequest-Objekt, Bestandteil vieler Browser, um Daten auf asynchroner Basis mit dem Webserver austauschen zu können.

Datenübertragung:

* reSt-ähnliche Verfahren
* JSON
* Diverse proprietäre XML Formate
* SOAP

**JQuery Promise:**

In jQuery werden Aufrufe die ‚quasi‘ im Hintergrund ausgeführt werden über Promises abgearbeitet.

Promise = Versprechen

Die Aufgabe welche übergeben wird, wird gelöst und im Fehlerfall eine Rückmeldung gegeben.

Promises können selber erstellt werden, aber auch innerhalb jQuery verwendet werden.

Promise hat folgende API:

then()

always()

done()

fail()

notify()

**JQuery Handlebars:**

jQuery Handlebars ist ein Framework um dynamische Webseiten zu erstellen.

Mit Handlebars können HTML Templates erstellt werden welche bei Bedarf von Javascript ein oder ausgeblendet werden.

Handlebars bietet ausserdem noch eine einfache Templating Engine um Clientseitig dynamische Content zu generieren.

Ablauf:

Handlebars Templates in entsprechenden Script Tags erstellen.

Kompilieren

Dem Template einen Kontext übergeben.

Gerenderte HTML im DOM anzeigen lassen.