HTML

```
<!doctype html>
<html>
 <head>
  <meta charset=‘‘utf-8">
  <link rel="stylesheet" href=‘‘style.css’’ type=‘‘text/css’’>
  <script src="myscript.js"></script>
  <title>A first HTML-Example</title>
 </head>
 <body onload="someFunctionFromMyScriptJS()">
   <article>
    <p>Hello <span id="type">class</span>!</p>
    <a href="http://example.com">A link!</a>
    <form action="tables.html" method="get">
      Fach:<br>
      <input type="text" name="fach" value="Web-Prog">
      <br><br>
      <input type="submit" value="Submit">
      </form>
   </article>
 </body>
</html>
```

- ul: unordered list (mit .)
- ol: ordered list (nummeriert)
- main, article, aside, header, footer,section
- <!– –> Kommentar
- div, span
- img, audio, video
- form: (action, method, target, autocomplete, novalidate) input: text, password, submit, radio, checkbox, button, color
- <script src=URL/> laden von Javascript-Files, interpretation beim Einlesen

CSS

```
Selector [, Selector2, ..., SelectorN]
{
Property1: Value1; /* Comment */
...
PropertyN: ValueN;
}
```

Selectors

- *: alle elemente
- p: alle <p> elemente
- h1, p: alle <h1> und alle <p> elemente
- div p: alle <p> elemente in elementen
- .example: alle elemente mit class=ëxample"
- #id: alle elemente mit id=ïd"
- div > p: alle <p> elemente mit als parent
- div + p: <p> elemente, die direkt auf elemente folgen
- h1 p: alle <p> elemente die auf <h1> elemente folgen

href : elemente mit href-Attribut

href=/url.tst : elemente mit href="/url.tst"

pseudo-tags

- :active geklickt
- :hover mausschwebend
- :visited besuchter link

browser-präfixe: -ms-, -moz-, -webkit-

Media-Query

```
@media not|only mediatype and (expressions) {
CSS-Code;
}
- Example
@media screen and (min-width: 480px) {
body {
background-color: lightgreen;
}
```

```
}
```

Box-Model

Margin, Border, Padding, Content(height,width)

JavaScript

[1,2,3] // Array initializer
{x:1, y:2} // Object initializer
undefined, null, infinity,NaN sind besondere Zustände
vergleiche mit === oder !==
hoisting: variablendeklarationen werden an Funktionsanfang gezogen
üse strict"; // Aktiviert Striken Modus, z.B. für Vergleiche ohne Konvertierung
es gibt nur referenz-Typen
Objektvergleiche vergleichen die Referenz, Primitive Typen(z.B. Strings) werden Wert-verglichen
Objekte sind sammlungen von Properties (name / value pairs) [x:1, y:2]
Foreach schleife: for (var p in o) //sth.
global scope and function scope, kein Block-scope
Zugriff auf aufrufkontext: this
innere Funktionen können auf umgebende Properties Zugreifen (außer this);
closure: funktionen merken sich umgebungsvariablen in einer scope chain z.B.

```
function outer () {
   var x = 2;
   return function (){ return 2*x;};
}
outer()() // gibt 4 zurück;
```

indirekte funktionsaufrufe: apply(context, [args]) bzw. call(context, arg1, arg2, ...)
Objektorientierung: durch Prototypen
MyVec.prototype.addTo = function (v) { //sth; }
Konstruktor: var x = new MyClass();

```
Vererben:
  function Kind(x) {
  Eltern.apply(this, x);
 }

Kapselung:
function klasse(x) { // Constructor
   var _x = x;
   this.getX = function() { return _x; };
}
```

APIs

APIs: Javascript core(Object,Array...), Browser Object Model(location,XMLHttpRequest,...), DOM(document,...)

BOM

Interaktion mit dem Browser,kein standard aber in allen Browsern ähnlich
window ist der Tab, bei browsern = globales Objekt, jeder Tab eigener Interpreter
Objekte

- document Represents the HTML document as DOM object
- screen Contains information about the users screen (width, height, ...)
- location Contains the current pages address (URL) and can be used to redirect the browser to a new page
- history Very restricted access to the browser history
- navigator Contains information about the browser
- innerWidth, innerHeight Width / height of the content area
- outerWidth, outerHeight Width / height including scrollbars, etc.

Methoden

- alert(msg): Displays alert-box with a message and an OK button
- atob()/btoa():Base-64 encoding / decoding of strings
- open(url) Opens URL in a new window (Popup!)
- close() Closes the current window
- setInterval(function,milliseconds,parameters) / clearInterval() Calls a function (or evaluates an expression) at specific time intervals
- setTimeout() / clearTimeout() Calls a function (or evaluates an expression)after a specified number of milliseconds

Navigator: nicht standardisiert, weit unterstützt (appName, geolocation ...);
location: Bearbeiten der URL; Properties:

- hash: teil nach #
- hostname
- href: ganze URL
- pathname:
- port

DOM

Erlaubt Manipulation des Dokumentes zur laufzeit;
Darstellung der Elemente durch Nodes (Inhalt in TextNode)
Nodes sind in Baumstruktur gegliedert, incl. Geschwisterliste
zugriff: getElementById (): liefert HTML-Element
getElementsByClassName(), getElementsByName(): liefern liste von HTML-Elementen
Attribute auf HTML-Elementen:
innerHTML, setAttribute(name, value), style.property;
nodeName /*Tag*/, nodeValue /*Text in TextNodes*/
Events:
Im HTML: <h1 onclick="this.innerHTML='Foo!'»Click me!</h1>
Im JS: element.addEventListener(event, function); (event ist ohne ön")
Event-weitergabe (Defaultverhalten) stoppen: event.preventDefault(); event.stopPropagation();
Baum-navigation: parentNode, childNodes[idx], firstChild,lastChild, nextSibling, previousSibling
Elemente Einfügen

```
var para = document.createElement("p");
var node = document.createTextNode("This is new.");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para); // analog dazu insertBefore(), replaceChild(), removeChild()
```

URL codieren:
encodeURIComponent("http://w3schools.com/my test.asp?name=ställe& car=saab");

Examples

Drag& Drop

```
<script>
function allowDrop(ev) {
ev.preventDefault();
}
function drag(ev) {
ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
ev.preventDefault();
var data = ev.dataTransfer.getData("text");
ev.target.appendChild(document.getElementById(data));
}
</script>
...
<div id="div1" ondrop="drop(event)" ondragover="allowDrop(event)"></div>
<p id="drag1" draggable="true" ondragstart="drag(event)"/>
```

Local Storage

```
if (typeof(Storage) !== "undefined") {
// Store
localStorage.setItem("lastname", "Smith");
// Retrieve
document.getElementById("result").innerHTML = localStorage.getItem("lastna
} else {
document.getElementById("result").innerHTML = "Sorry, your browser does nc
}
```

0.1 http

generisch, plain text, zustandslos

```
message = request-line | status-line
*(message-header CRLF)
CRLF
[message-body]
```

request-line : METHOD URL HTTP/1.1 response line : HTTP/1.1 STATUS-CODE STATUS-MESSAGE

Status-code: 1XX - Informational (request received, continues processing) 2XX - Success 3XX - Redirection (further action required in order to fulfill the request) 4XX - Client error (request contains bad syntax or cannot be fulfilled) 5XX - Server error (server failed to fulfill an otherwise valid request)

Sessions

Http ist zustandslos, Sitzungen durch austausch von Session-Identifiern bei Login. SessionID muss eindeutig sein. Client schickt SessionID bei jeder Anfrage, z.B. per Cookie, URL
Cookie: key-value-paare als string, JS-zugriff möglich

https

TLS = Transport Layer Security zwischen TCP und http => schutz vor Man-in-the-middle, authentifikation der Website; Verschlüsselung der Verbindung
Attacken: wenn URI erraten: Known-plaintext, Traffic-analysen

AJAX

Laden von Inhalten ohne neuladen der Seite

```
var req = new XMLHttpRequest();
var response = null;
req.open("GET", URL, true);
req.onreadystatechange = function () {
if (req.readyState == 4) {
response = req.responseText; // responseXML when transferring XML
// do something with the response
}
};
req.send(null)
```

CGI

Aufruf von Kommandozeilenbefehlen durch Web-Server: HTTP server setzt umgebungsvariablen (z.B. QUERY_ STRING, REQUEST_ URI, REQUEST_ METHOD, HTTP_ ACCEPT, ...) und Ruft Skript auf(mit PUT/POST -> stdin), stdausgabe wird versendet
Nachteile: Nach Angriff beliebiger Code direkt auf Betriebssystem einschleusbar, Erzeugt Prozess für jeden Aufruf

Verbesserungen

Compilierte Anwendungen statt Skripte
FastCGI: Länger Laufender Interpreter führt mehrere Anfragen nacheinenader aus => weniger Prozesse
HTTP-Server modules: Interpreter als Modul des Webservers

PHP

Wird auf Server ausgeführt, ausgabe wird versendet
Einbetten in HTML: <?php ... ?>
Variablen:
$ myInt = 10;
isset($ myInt); //True wenn definiert
unset($ myInt); //Löschen