JavaScript
erste Schritte

Was ist JavaScript?

- JavaScript (JS) ist eine zumeist client-seitig eingesetzte Skriptsprache, die ursprünglich für dynamisches HTML in Internet-Browsern entwickelt wurde, um:
  - Benutzerinteraktionen auszuwerten,
  - Inhalte zu verändern, ohne die Webseite nachzuladen,
  - eine Webseite per Programmcode nachzuladen via Asynchronous JavaScript and XML (AJAX) und um
  - einen Seiteninhalt dynamisch zu generieren.
- JavaScript wird von allen gängigen Internet-Browsern interpretiert.

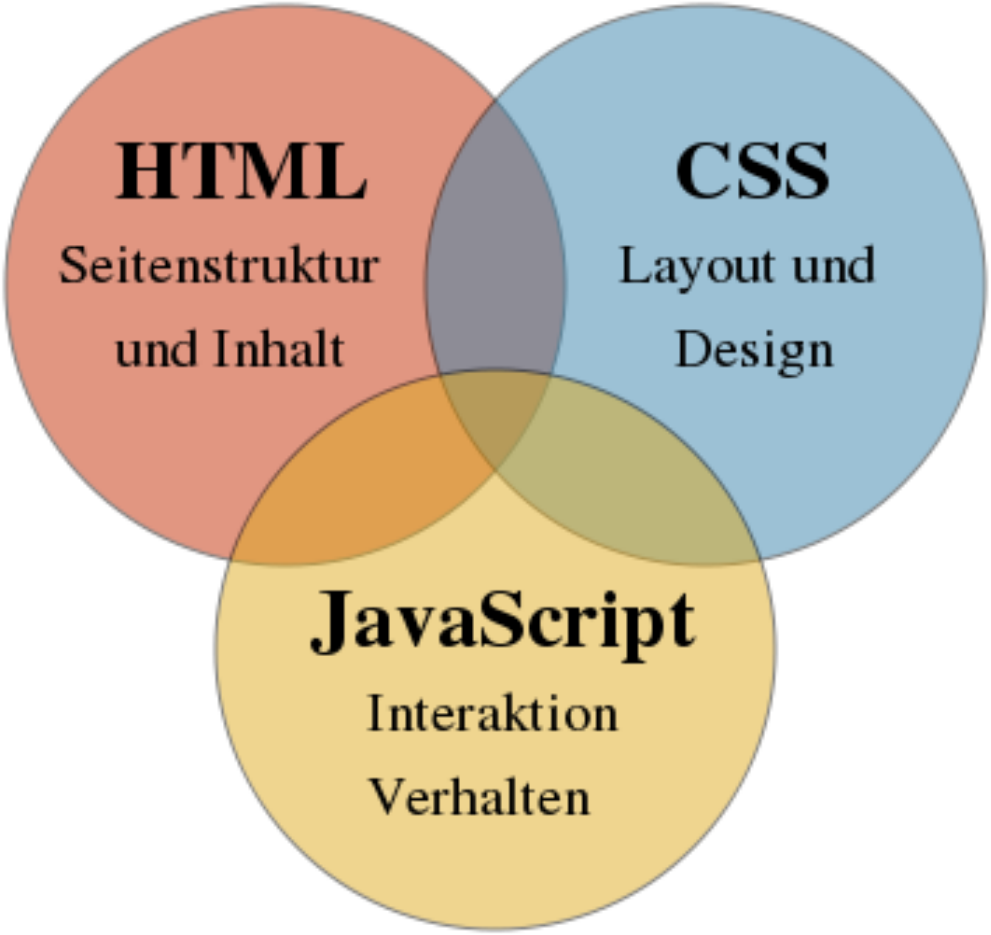
Was ist JavaScript?

- Typische Anwendungsgebiete von JavaScript in einem Internet-Browser sind...
  - die dynamische Manipulation von Webseiten über das Document Object Model (DOM).
  - die Plausibilitätsprüfung und Datenvalidierung von Formulareingaben noch vor der Übertragung zum Server.
  - die Anzeige von Dialogfenstern.
  - das Senden und Empfangen von Daten, ohne dass der Browser die Seite neu laden muss.
  - das Vorschlagen von Suchbegriffen während der Eingabe.
  - die Anzeige von Werbebannern oder Laufschriften.
  - die Verschleierung von E-Mail-Adressen zur Bekämpfung von Spam.

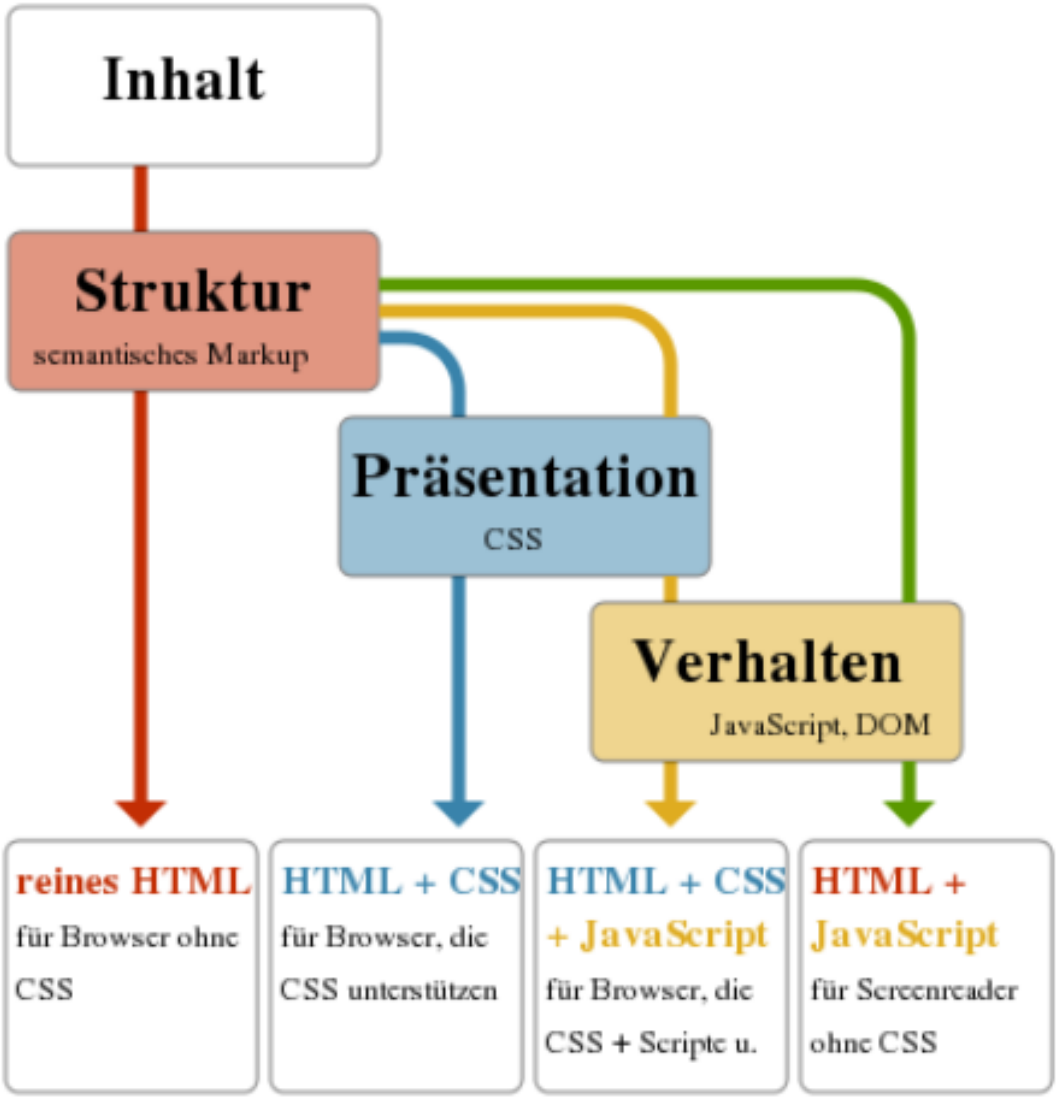
Was ist JavaScript?

- JavaScript ist Turing-vollständig und erfüllt damit die Kriterien einer vollwertigen Programmiersprache.
- JavaScript ist vollständig dynamisch typisiert;
  die Zuweisung von Werten an Variablen unterliegt keinen typbasierten Einschränkungen.
  - Aufgrund dessen ist der Datentyp keine Eigenschaft einer Variablen, sondern Laufzeit-bezogen die Eigenschaft ihres aktuellen Wertes.
  - Der Datentyp eines Wertes lässt sich zur Laufzeit mit dem unären Operator **`typeof`** ermitteln.

HTML, JavaScript und CSS: Die Basistechnologien des Clients



HTML, JavaScript und CSS: Die Basistechnologien des Clients



Verschiedene Arten der Einbettung

- Es gibt verschiedene Wege JavaScript in HTML einzubetten:
  - Intern
  - Extern
  - Über die DOM-API
- Die Reihenfolge der Einbettung ist wichtig, da der Code auf alle vor ihm eingebunden externen oder internen Dateien zugreifen kann.
- Die Scripte werden geladen und ausgeführt, sobald der Parser auf ein Script-Tag stößt.
- JavaScript-Code kann auf allen vor ihm eingebundenen externen oder internen Code zugreifen.
- Die dritte Variante hat nur wenige Anwendungsfälle wie z.B. das Einbinden von Google Maps Code.
- Es sollte generell die externe Variante bevorzugt werden.

Interne Einbettung durch script-Tag

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8" />
    <script>alert("Hallo Welt!");</script>
    <noscript>Sie haben JavaScript deaktiviert.</noscript>
    <title>Titel der Seite</title>
</head>

<body> <!-- Sichtbarer Dokumentinhalt -->
    <h1>Hallo Leute</h1>
    <p>Sehen Sie sich den Quellcode dieser Seite an.</p>
</body>

</html>
```

Einbettung einer externen Datei

```
<!DOCTYPE html>
<html lang="de">
<head>
    <meta charset="utf-8" />
    <title>Titel der Seite</title>
</head>

<body> <!-- Sichtbarer Dokumentinhalt -->
    <h1>Hallo Leute</h1>
    <p>Sehen Sie sich den Quellcode dieser Seite an.</p>
</body>

<script src="extern.js"></script>

</html>
```

Einbettung mittels der DOM-API zur Laufzeit

- Während der Laufzeit kann mittels der DOM-API ein Script-Element erzeugt und anschließend dem DOM hinzugefügt werden.

```
var googleMapsApiKey = '...';
var script = document.createElement('script');
script.id = 'google-maps-script';
script.src = 'https://maps.googleapis.com/maps/api/js?key='
                + googleMapsApiKey
                + '&libraries=places&callback=initMap';
document.body.appendChild(script);
```

JavaScript in der Sandbox

- JavaScript wird im Browser in einer sogenannten Sandbox ausgeführt.
- Dies bewirkt, dass man in JavaScript im Allgemeinen nur Zugriff auf die Objekte des Browsers hat und somit nicht auf das Dateisystem zugreifen und dadurch keine Dateien lesen oder schreiben kann.
- Um Sicherheitsprobleme, wie das sogenannte Cross-Site-Scripting zu verhindern, wird jede Website oder Webanwendung innerhalb des Browsers isoliert ausgeführt und ein Datenaustausch unterbunden.
  - Ohne diesen Schutz wäre es möglich, über eine Seite Schadcode auszuführen, der beispielsweise Bank- oder Logindaten in anderen parallel geöffneten Browserfenstern ausliest oder manipuliert.