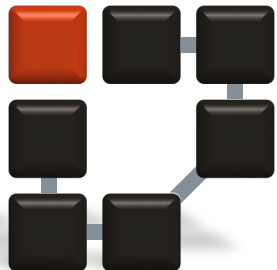


# Informatik 1 für Nebenfachstudierende Grundmodul

## HTML – Scripting

Kai-Steffen Hielscher  
Folienversion: 03. Dezember 2019



**Informatik 7**  
Rechnernetze und  
Kommunikationssysteme



**FRIEDRICH-ALEXANDER  
UNIVERSITÄT  
ERLANGEN-NÜRNBERG**  
TECHNISCHE FAKULTÄT

# Inhaltsübersicht

- Kapitel 2 - HTML
  - Einführung
  - Übersicht
  - HTML - Grundbegriffe
  - HTML - Texte und Verweise
  - HTML - Fortgeschrittene Techniken
  - Cascading Style Sheets CSS
  - **Scripting**

# Scripting

- Dynamisches HTML
- Client Side Scripting (DOM)
- Server Side Scripting (PHP, Perl, Python, . . .)

# Dynamisches HTML

- Dynamisches HTML (engl. "Dynamic HTML" oder abgekürzt "DHTML") ist keine klassische HTML-Erweiterung in Gestalt neuer HTML-Elemente, keine neue Sprache
- Dynamisches HTML ist vielmehr der Sammelbegriff für verschiedene Lösungen, um dem Autor einer Web-Seite zu ermöglichen, Elemente der Webseite erst während der Anzeige dynamisch zu erzeugen, sei es automatisch oder durch Einwirken des Anwenders
- Im Unterschied zum gewöhnlichen (d.h. statischen HTML) wird hierbei dem Anwender die Möglichkeit gegeben, durch Interaktion das Geschehen am Bildschirm selbst mit zu bestimmen

# Dynamisches HTML

- Beispiel: Das Ausfüllen und Abschicken eines Formulars und anschließender Reaktion, einer erst jetzt erstellten endgültigen Web-Seite, angezeigt vom Browser des Anwenders
- Mit der Dokument-Beschreibungssprache HTML alleine lassen sich keine dynamisch generierten Web-Inhalte erzeugen
- dazu benötigt man eine Skriptsprache
- Solche Skriptsprachen verarbeiten vom Anwender eingegebene Daten entweder client-seitig (z.B. JavaScript) oder server-seitig (z.B. PHP, Perl, Python)
- Ergebnis ist immer eine vollständig generierte Web-Seite, die vom Browser des Anwenders dann angezeigt wird

# Client Side Scripting: Document Object Model

## ■ Document Object Model (DOM)

- plattform- und sprachunabhängige Schnittstelle zum Zugriff auf
  - Inhalt
  - Struktur
  - Stil von HTML-Dokumenten
- HTML-Dokumente hierarchisch strukturiert: Baum-Darstellung

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
<html>   
  ▼ <head>  
    <title>Zweites CSS-Beispiel: Zentrale Formatierung</title>  
    ▶ <style type="text/css"></style>  
  </head>  
  ▼ <body>  
    <h1>Zentrale Formatierung</h1>  
    ▶ <p></p>  
    ▶ <p></p>  
  </body>  
</html>
```

# Client Side Scripting: Document Object Model

## Zentrale Formatierung

Die Elemente, die in diesem HTML-Dokument verwendet werden, wurden durch CSS zentral formatiert.

Dies kann besonders viel Arbeit sparen, wenn man dabei geschickt vorgeht.

h1 1434.67 x 37.4

The screenshot shows a web browser's developer tools interface. The top part of the browser window displays the page content, which consists of two paragraphs of text. The first paragraph reads: "Die Elemente, die in diesem HTML-Dokument verwendet werden, wurden durch CSS zentral formatiert." The second paragraph reads: "Dies kann besonders viel Arbeit sparen, wenn man dabei geschickt vorgeht." A tooltip is visible over the first paragraph, showing the bounding box for the `h1` element: "h1 1434.67 x 37.4".

The developer tools interface is open at the bottom. The left pane shows the DOM tree with the following structure:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <title>Zweites CSS-Beispiel: Zentrale Formatierung</title>
    <style type="text/css">
      h1 { color:#336699; font-family:sans-serif; } p { font-size:14px; font-family:sans-serif; }
    </style>
  </head>
  <body>
    <h1>Zentrale Formatierung</h1>
    <p>
      Die Elemente, die in diesem HTML-Dokument verwendet werden, wurden durch CSS zentral formatiert.
    </p>
    <p>
      Dies kann besonders viel Arbeit sparen, wenn man dabei geschickt vorgeht.
    </p>
  </body>
</html>
```

The right pane shows the CSS rules for the selected `h1` element:

```
Regeln
Berechnet
Animationen
Schriftarten
Stile filtern
Element {
}
h1 {
  color: #336699;
  font-family: sans-serif;
}
Inline:2
```

The breadcrumb at the bottom of the developer tools shows the path: `html > body > h1`.

# Client Side Scripting: Document Object Model

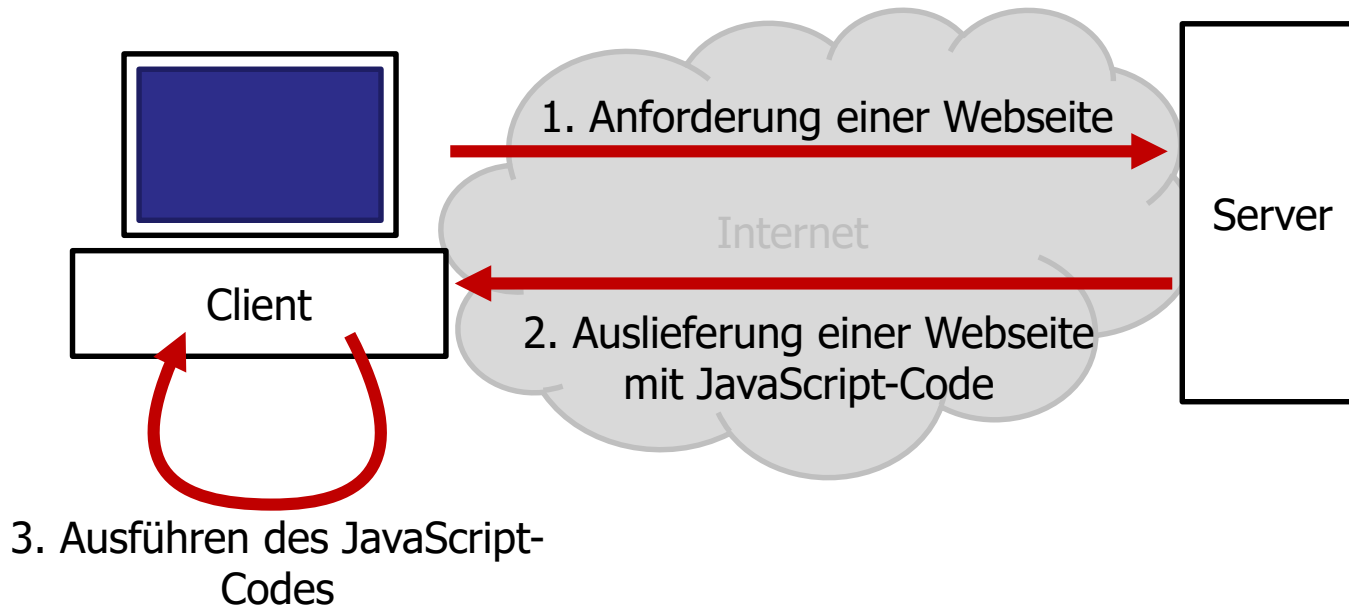
- Unterschiedliche Level (Versionen)
  - Level 0 (keine formale Spezifikation)
  - Level 1 (1998)
  - ...
  - Level 3 (2004)
- Unterschiedliche Module
  - DOM Core
    - Bewegung im Baum, Manipulation von Knoten
  - DOM HTML
    - Erweiterungen zum Zugriff auf Elemente in HTML-Dokumenten
  - in aktuelleren Leveln noch weitere Module, z.B. Style, CSS, ...



# Client Side Scripting: Document Object Model

- Zugriff auf Blätter im Baum über Wurzel und Zweige
- beliebige Bearbeitung über verschiedene Programmiersprachen, z.B. Perl, JavaScript, Java, C, C++, PHP, Python, ...
- Das DOM legt fest, wie und auf welche Elemente eines Dokuments eine Skriptsprache zugreift

# Client Side Scripting



- Webdesigner schreibt JavaScript-Programmcode direkt in HTML-Dokument, Dokument liegt auf Server
- Client fordert Seite an (1), erhält HTML-Dokument mit eingebettetem JavaScript (2)
- Browser des Clients führt den eingebetteten JavaScript-Code auf dem Client-Rechner aus (3)

# Client Side Scripting: Beispiel

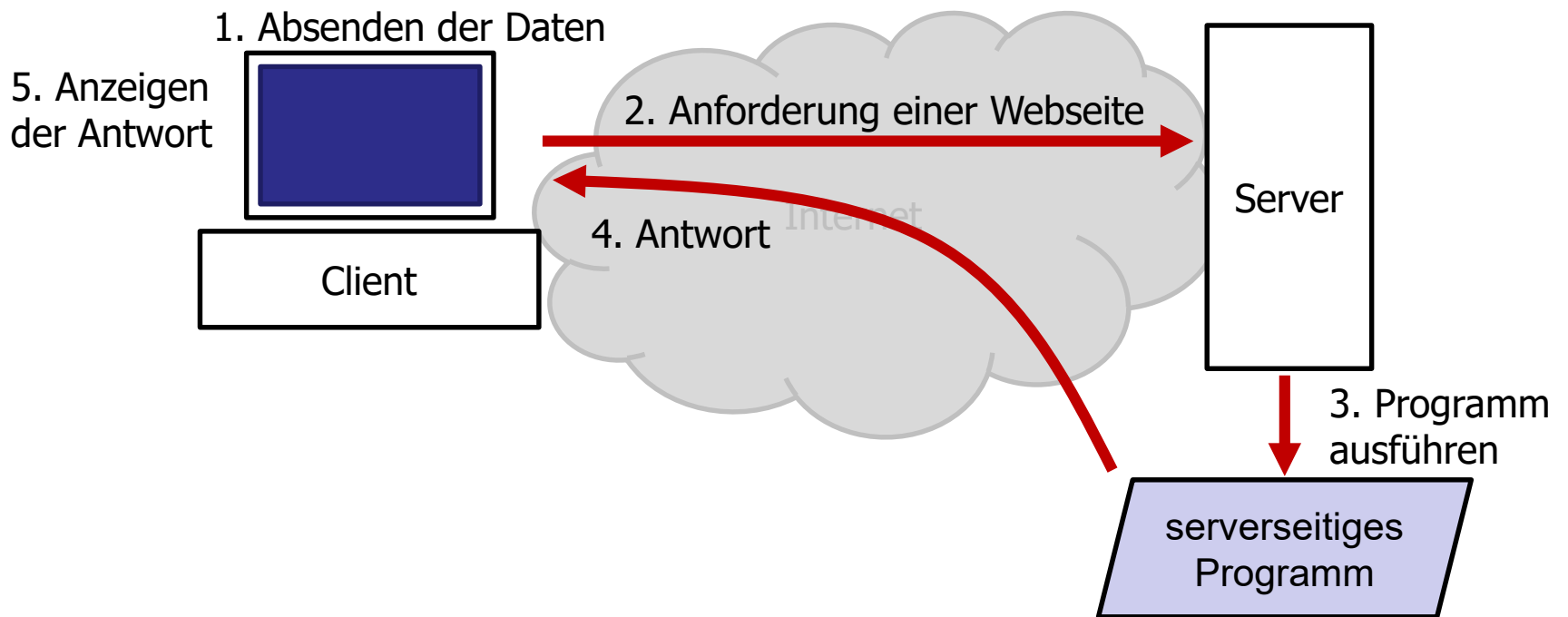
```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
<!--
function Quadrat() {
var Ergebnis = document.Formular.Eingabe.value *
document.Formular.Eingabe.value;
alert("Das Quadrat von " + document.Formular.Eingabe.value + " = " +
Ergebnis);
}
//-->
</script>
</head>
<body>
<form name="Formular" action="">
<input type="text" name="Eingabe" size="3">
<input type="button" value="Quadrat errechnen" onClick="Quadrat()">
</form>
</body>
</html>
```

Notation zur Einbettung  
des Skripts

Programmtext in HTML-Datei

*Event Handler*

# Server Side Scripting



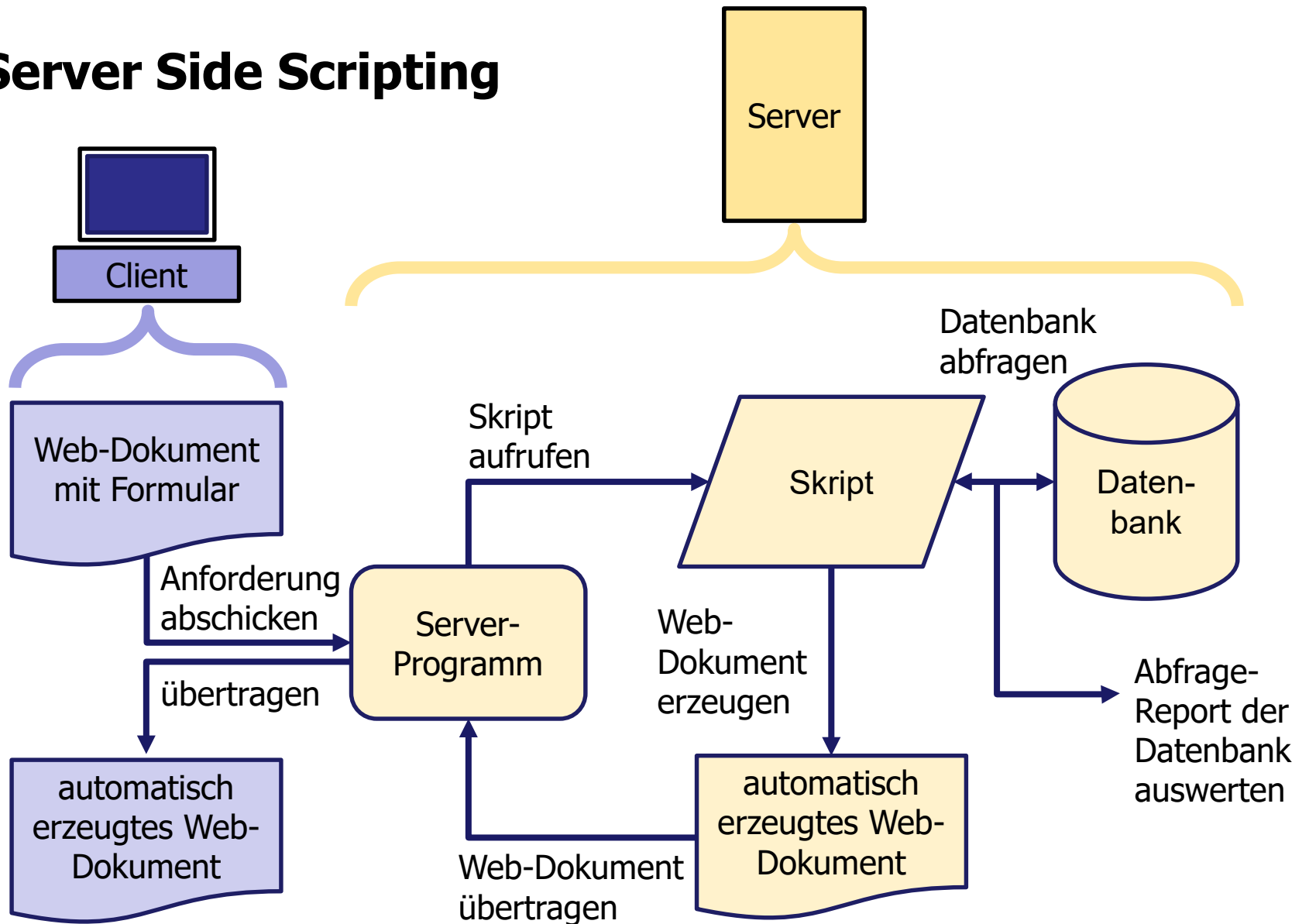
# Server Side Scripting

- Beispielsweise füllt der Webbesucher ein Formular aus und klickt auf den Abschicken-Button des Formulars (1), der Abschicken-Button ist mit dem URL (Adresse) eines serverseitigen Programms über das `action`-Attribut verbunden (s. Vorlesungsabschnitt über Formulare).
- Der Webbrowser schickt dem Webserver, der in der URL angegeben ist, eine Aufforderung, dieses Programm aufzurufen, um diesem die Formulardaten zu übergeben (2).
- Der Webserver ruft das Programm auf und dann wird es auf dem Server-Rechner ausgeführt und verarbeitet die Formulardaten (3).
- Das Ergebnis ist z.B. der HTML-Code einer Antwortseite
- Der Webserver schickt dann diese vom Programm generierte Webseite an den Browser zurück (4)
- Der Webbesucher empfängt diese Seite und sieht somit in seinem Browser die Bestätigung, dass seine Formulardaten angekommen sind und korrekt verarbeitet wurden (5)

# Server Side Scripting

- Server Side Scripting ist sehr mächtige Form der Webtechnologie
  - Geeignet für anspruchsvolle Aufgaben oder das Zurückliefern von Informationen an den Webserver (z.B. Bestellungen und Aufträge bei allen Internet-Versandsystemen wie amazon.de usw.)
  - Client Side Scripting geeignet für dynamische Effekte und schnelle Interaktion mit dem Webbesucher für anspruchslosere Problemstellungen (z.B. Formulardaten vor Absenden validieren)

# Server Side Scripting



# Server Side Scripting

## ■ Beispiel: PHP (serverseitig)

```
<html>
<head>
<title>Beispiel</title>
</head>
<body>
<?php
$datum = date("Y-m-d");
echo "Hallo, heute ist $datum!";
?>
</body>
</html>
```



# Server Side Scripting

- Beispiel: PHP (zurückgeliefertes HTML auf Client-Seite)

```
<html>  
<head>  
<title>Beispiel</title>  
</head>  
<body>  
Hallo, heute ist 2016-12-21!  
</body>  
</html>
```

# Server Side Scripting

- Erzeugung und Rückgabe von aktueller Information
  - Umgebungsvariablen
  - Datenbankabfragen
  - Datenbankeinträge
  - Berechnungen
- Server arbeitet Skript ab und liefert Resultat
  - Kein (auszuführender) Programmtext mehr im HTML-Dokument für den Browser
  - Benutzer am Browser kann Programm nicht sehen

# Scripting

- Scriptsprachen werden interpretiert und nicht kompiliert
- Kompilation versus Interpretation
  - Der erste Schritt beim Programmieren besteht immer darin, den Quelltext des Programms aufzusetzen
  - dieser ist einfacher, unformatierter ASCII-Text, in dem man unter Beachtung der Syntax der verwendeten Programmiersprache die Befehle niederschreibt, die der Rechner bei Ausführung des Programms abarbeiten soll
  - der Rechner kann allerdings den Quelltext nicht direkt abarbeiten; dazu ist ein zweiter Schritt erforderlich, nämlich die Umwandlung des Quelltextes in Maschinencode (Binärcode als Folge von `0` und `1` ; stellen einen kleinen Satz elementarer Befehle dar, die der Prozessor des Rechners `verstehet`)
  - Zur Umwandlung eines Quelltextes in Maschinencode gibt es zwei unterschiedliche Konzepte: **Kompilation** und **Interpretation**

# Scripting

## ■ Kompilation

- Hierzu wird ein spezielles Programm (der Compiler) aufgerufen, das den Quelltext des Programms in Maschinencode übersetzt
- das Ergebnis ist eine von der CPU ausführbare Datei, die als eigenständiges Programm direkt aufgerufen und ausgeführt werden kann
- Kompilierte Programme sind meist schneller und effizienter als vergleichbare interpretierte Programme
- Kompilierte Programme sind nicht portabel, da der vom Compiler erzeugte Maschinencode immer auf einen bestimmten Prozessortyp abgestimmt ist und von anderen Prozessoren nicht verarbeitet werden kann
- Beispiele für kompilierte Programmiersprachen sind Pascal, C/C++ und Java

# Scripting

## ■ Interpretation

- Hier fallen Übersetzung und Ausführung des Programms zusammen, das macht der Interpreter
- Dieser liest das Programm Zeile für Zeile ein, erzeugt entsprechenden Maschinencode und lässt diesen direkt ausführen
- Interpretierte Programme sind in der Ausführung langsamer als vergleichbare kompilierte Programme, da im Hintergrund immer der Interpreter läuft und die Zeit für die Umsetzung in Maschinencode hinzugerechnet werden muss
- Interpretierte Programme können auf andere Rechner portiert werden, da sie nur als Quelltext vorliegen (vorausgesetzt ein passender Interpreter ist auf diesen installiert)
- Beispiele für interpretierte Programmiersprachen sind Basic, Perl, JavaScript und PHP

# Scripting

## ■ AJAX

- Asynchronous JavaScript and XML
- Kombination von clientseitigem und serverseitigem Scripting
- Kommunikation mit dem Server, ohne eine neue HTML-Seite anzufordern
- aktuell geladene Webseite wird aufgrund von Informationen des Servers verändert
  - Eingaben auf der Client-Seite werden an Server übermittelt (Client Side Scripting)
  - Server verarbeitet die Eingaben, generiert Antworten (Server Side Scripting)
  - Antworten werden an den Client übermittelt
  - Client ändert Elemente der Webseite (Client Side Scripting, DOM)
- Beispiele
  - Google Docs, ...