DHBW Ravensburg

WEBPROGRAMMIERUNG

WWI218

2. Client/Server-Prinzip

AGENDA

1. Grundlagen technische Kommunikation über Netzwerke
2. **Client/Server-Prinzip – Webbrowser**
3. Grundlagen Webentwicklung
4. Serverumgebungen für Webseiten
5. Entwicklungsumgebungen für Webprogrammierung
6. Formale Darstellungssprachen zur Entwicklung von Webseiten/Anwendungen
7. Programmiersprachen für die Webentwicklung
8. Datenhaltung und Datenbanken
9. Debugging und Testing
10. Responsive Webseiten und mobile Anwendungen
11. Hybride Entwicklung: Webapps
12. Open Source Internet-Anwendungen und Anwendungsgebiete

WIEDERHOLUNG

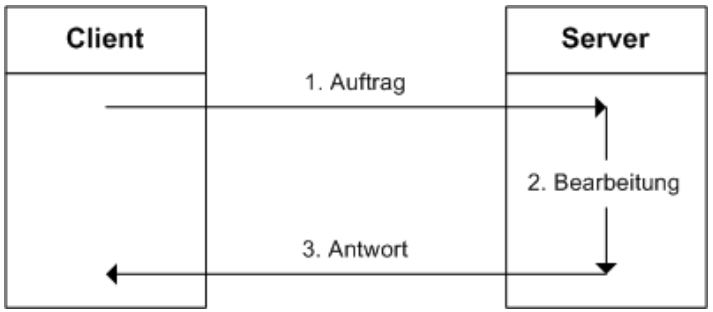
- OSI-Modell als Grundlage der Internet-Kommunikation
- Webprogrammierung als Tool, um die Schichten zusammenzufügen
- Verbindungen zwischen Client und Server über Netzwerke
- Sicherstellen einer „gemeinsamen Sprache“ bzw. Übersetzung
- Anlegen eines Webspace zur späteren Entwicklung einer Webseite

CLIENT/SERVER-PRINZIP
Übersicht

- Server
  - Anbieter, Bereitsteller, Dienstleister
  - Wichtig: Server ist eine Funktion/Rolle, ein physikalischer Computer kann auch Client und Server sein
- Client
  - Kunde, Nutzer des Diensts
  - Client fordert Dienst beim Server an (Request) und erhält Ergebnis (Response)
- Service
  - Inanspruchnahme einer definierten Aufgabe des Client beim Server
- Request
  - Anforderung eines Clients an den Server, welcher Service benötigt wird
- Response
  - Antwort des Servers auf die Anforderung des Clients

CLIENT/SERVER-PRINZIP
Übersicht

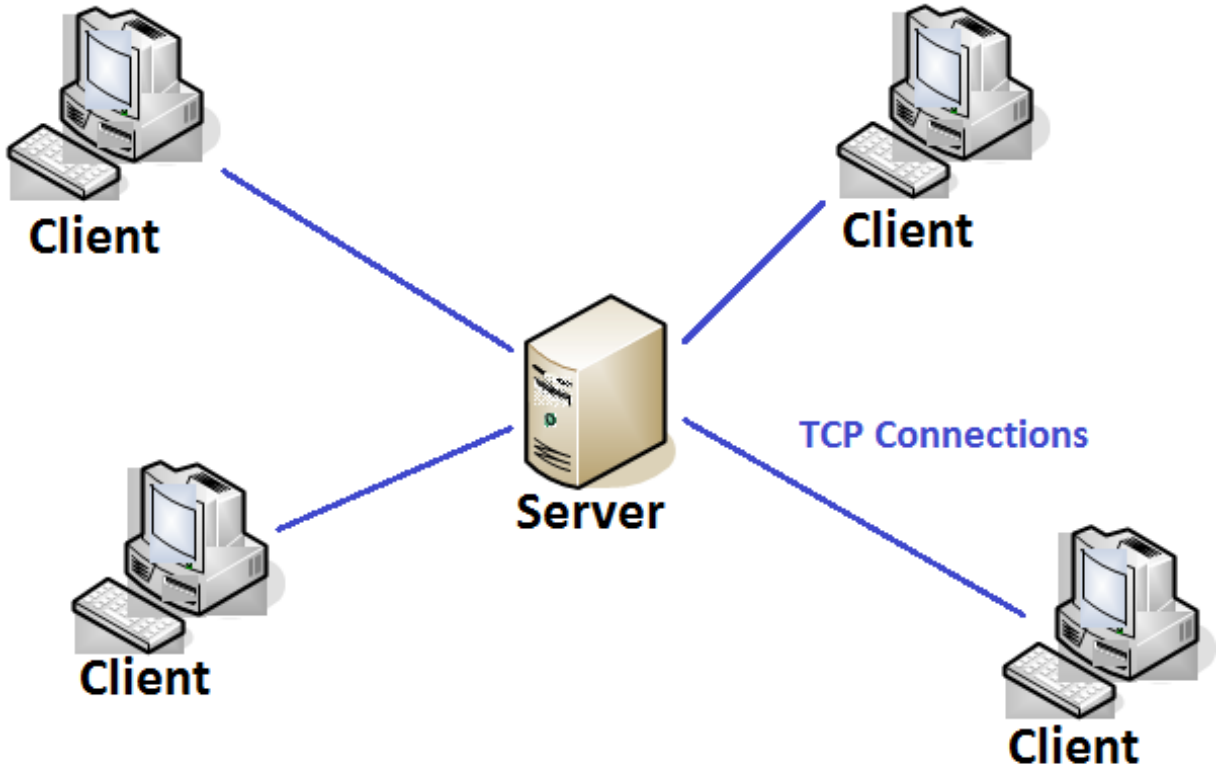
- Architekturprinzip zur Verteilung von Aufgaben und Dienstleistungen innerhalb eines Netzwerks
- Die Unterteilung erfolgt auf Basis von Programmen/Software, die auf Clients und Server verteilt werden
- Bei Bedarf fordert der Client einen Dienst an bzw. nimmt diese in Anspruch - der Server bearbeitet die Anfrage, erfüllt die Anforderungen des Clients und liefert das Ergebnis zurück



http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen/Client-Server-Architektur

CLIENT/SERVER-PRINZIP

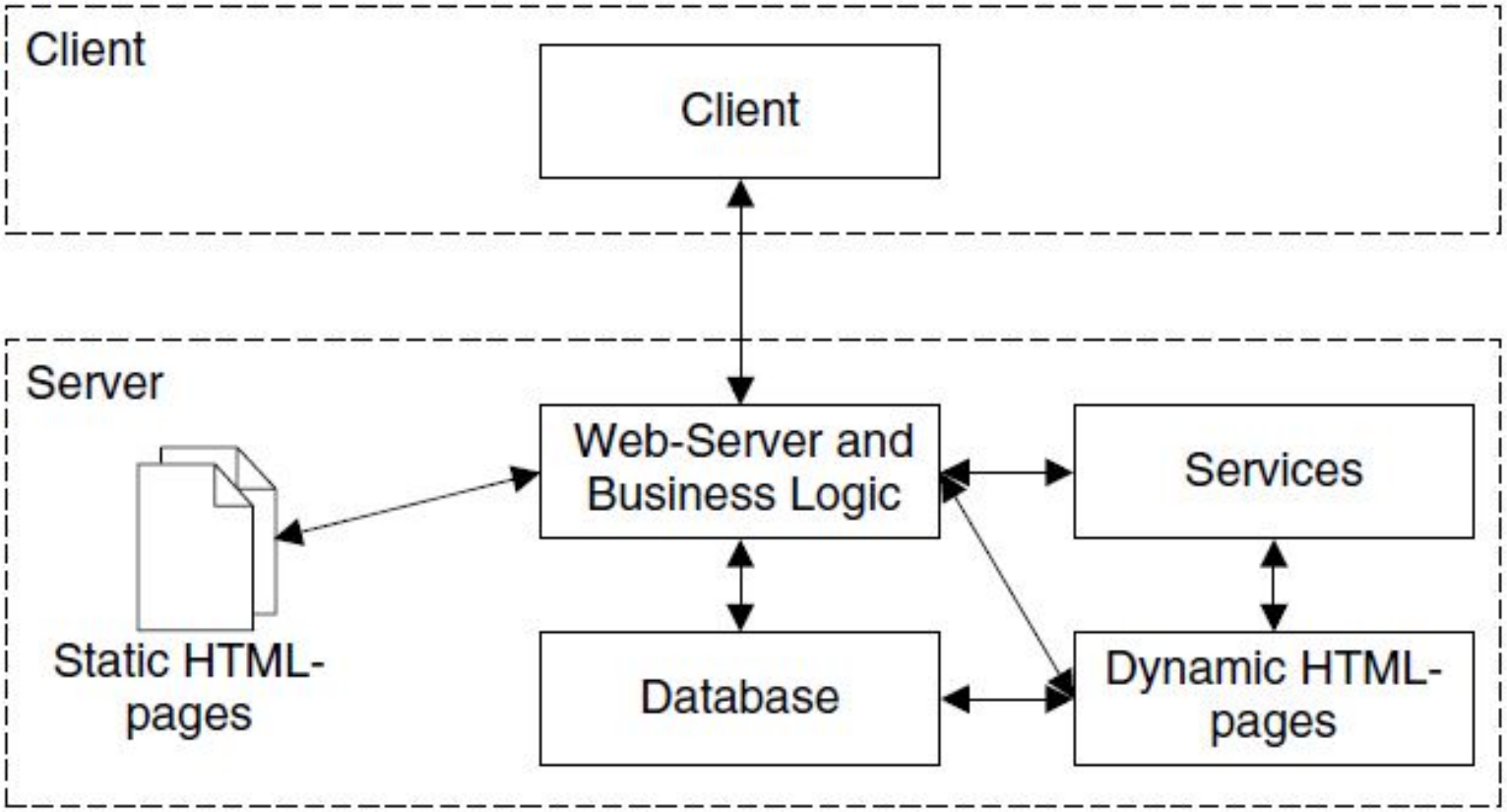
Einfacher Aufbau



http://practice.geeksforgeeks.org/ckeditor/images/uploads/1491250148_client_server.png

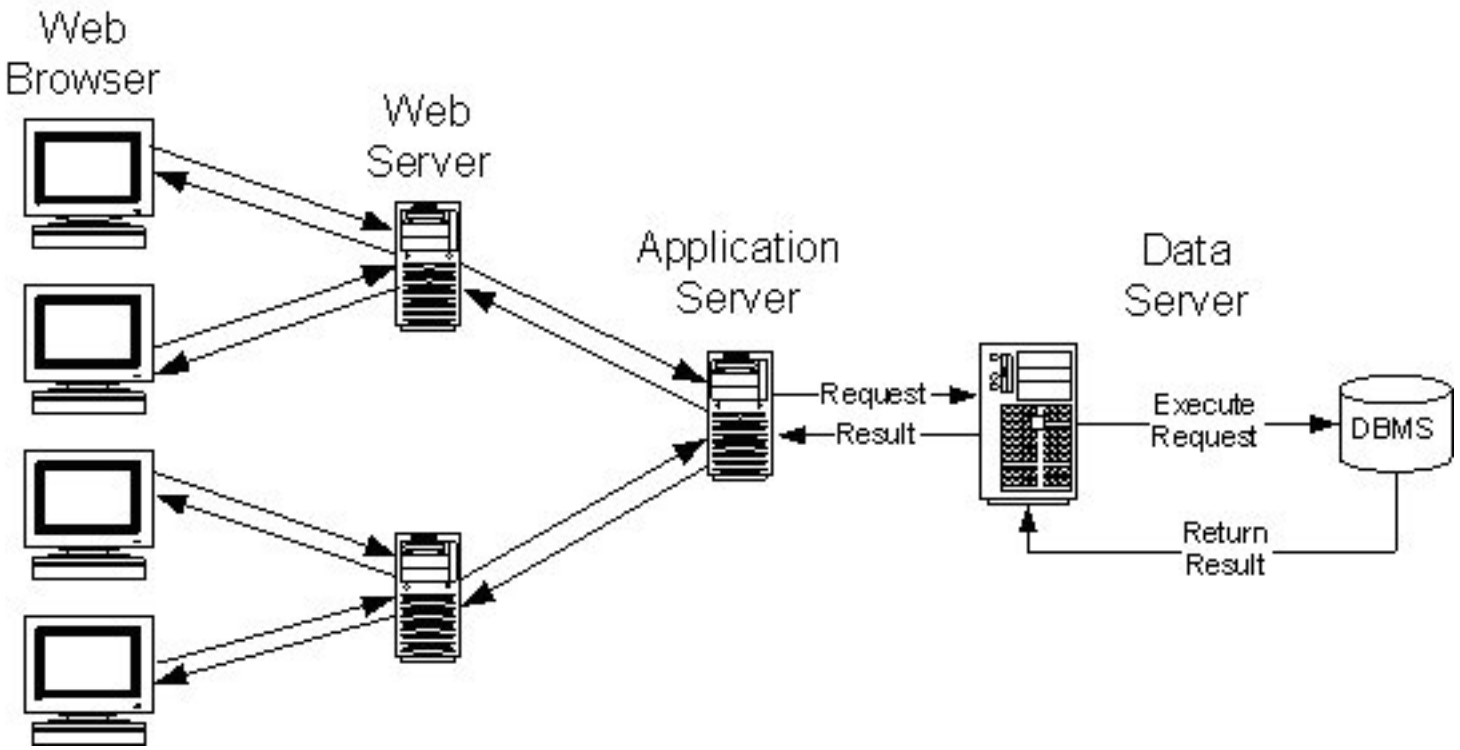
CLIENT/SERVER-PRINZIP

Funktionsweise



https://github.com/parrt/cs601/blob/master/lectures/client-server.md

CLIENT/SERVER-PRINZIP
Funktionsweise



https://github.com/parrt/cs601/blob/master/lectures/client-server.md

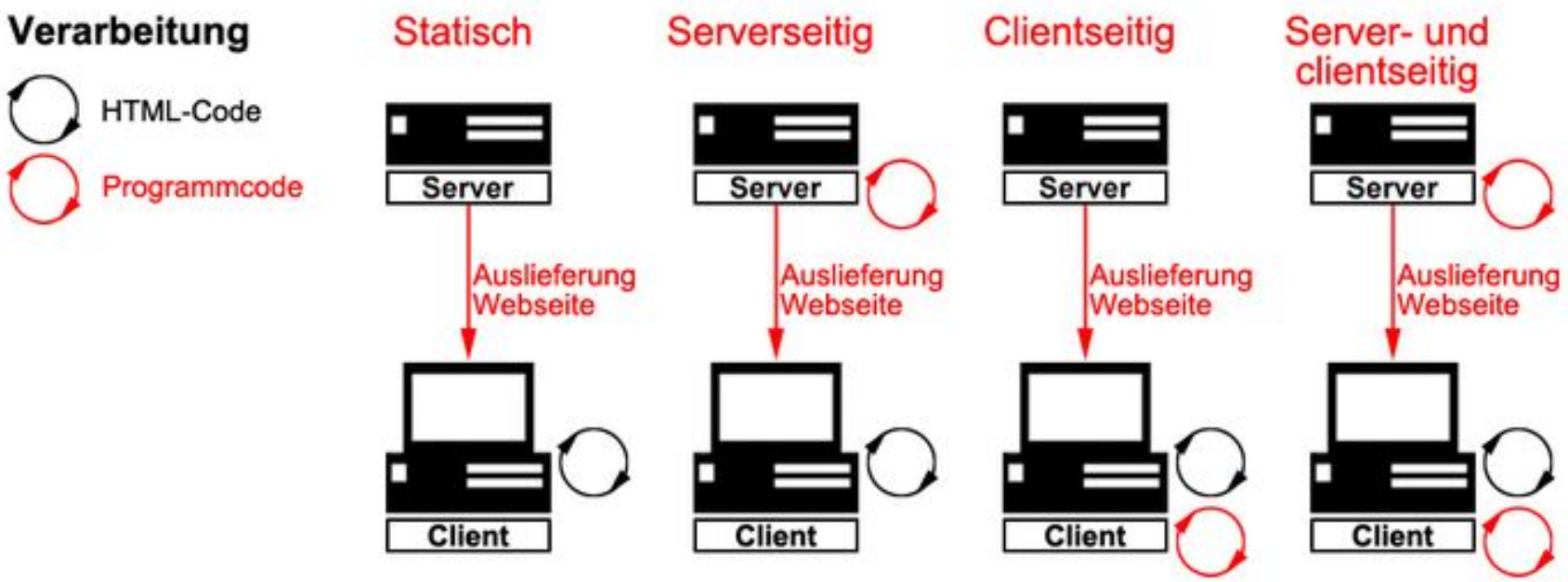
CLIENT/SERVER-PRINZIP
Anwendungsfelder Beispiele

- **Applikations-Server/Terminal-Server:** SAP, Windows Server…
- **Datenbank-Server/Data Warehouse:** SQL Server, Oracle…
- **File-Server:** NAS, Cloud…
- **Mail-Server:** Microsoft Exchange, Linux Postfix…
- **Print-Server**
- **Proxy-Server**
- **Präsentations-Server**
- **Web-Server**

- **Webbrowser**
- **Webapp**

CLIENT/SERVER-PRINZIP

Web-Programmierung Verarbeitung Code



Quelle: https://www.edv-lehrgang.de/serverseitige-clientseitige-programmierung/

CLIENT/SERVER-PRINZIP

Vor-/Nachteile clientseitige Programmierung

PRO	CON
Gewährleistet Berücksichtigung der technischen Umgebung User	User kann Verarbeitung des Programmcodes unterbinden
Aktualisierung ohne Neuladen der Seite möglich	Code kann Schadsoftware transportieren
Schnellere Bearbeitung möglich	Kein voller Zugriff auf Hardware des Users
Weniger Serverauslastung	Der ganze Quellcode ist frei ersichtlich
	Aufwändiger Testprozess

CLIENT/SERVER-PRINZIP

Vor-/Nachteile serverseitige Programmierung

PRO	CON
Rechenaufwand auf Serverseite	Ausführung des Programmcodes erfolgt nur bei Seitenaufruf
Rechenleistung leichter skalierbar	Zur Aktualisierung einer Seite muss diese neu geladen werden
Code muss nur auf einem System getestet werden	Sicherheit des Webservers muss sichergestellt werden
Code ist geschützt vor Zugriffen von außen	
Speichern und Abrufen von Daten möglich	

CLIENT/SERVER-PRINZIP

Auswahl client-/serverseitige Programmierung

Wann entwickle ich client-, wann serverseitig?

- **Ressourcenbedarf:** Wie rechenintensiv ist die auszuführende Aufgabe?
- **Zeit:** Wie schnell erwarte ich ein Ergebnis?
- **Performance:** Wie entscheidend ist die Ausführung der Aufgabe?
- **Ziel:** Was möchte ich erreichen?
- **Aufgabenumfang:** Wie aufwändig ist die Umsetzung meiner Aufgabe?
- **Speicherbedarf:** Wieviel Platz benötige ich physikalisch?
- **Berechtigungen:** Welche Rechte hat der User, der die Aufgabe nutzt?
- **Datenmenge:** Welche Datenmenge muss übertragen werden?

CLIENT/SERVER-PRINZIP

Beispiele client-/serverseitige Programmierung

- clientseitig
  - Validierung Formularfelder
  - Dropdown-Menüs
  - Slider
  - Bildergalerien
- serverseitig
  - Formularverarbeitung
  - Ermittlung der Uhrzeit
  - Laden von Text in eine HTML-Seite

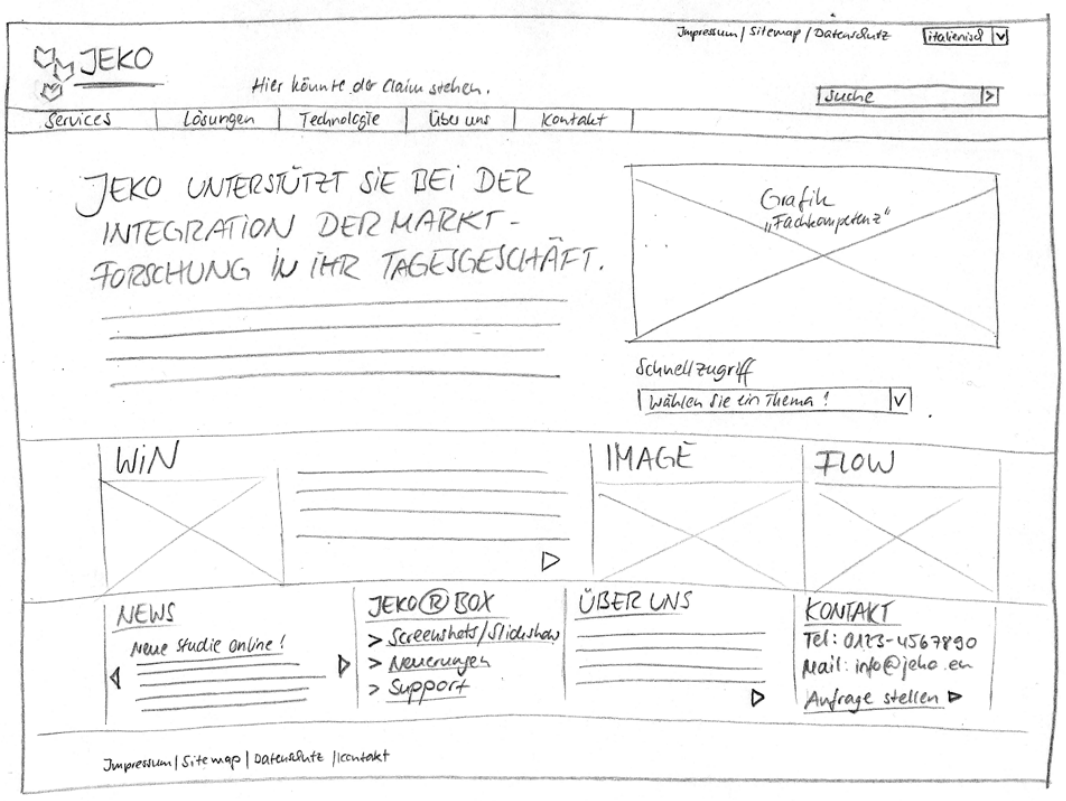
CLIENT/SERVER-PRINZIP
Skriptsprachen

Client	Server
JavaScript	PHP
Java-Applets	ASP
Flash	Perl
ActiveX	Python
	C/C++

CLIENT/SERVER-PRINZIP
Konzeption

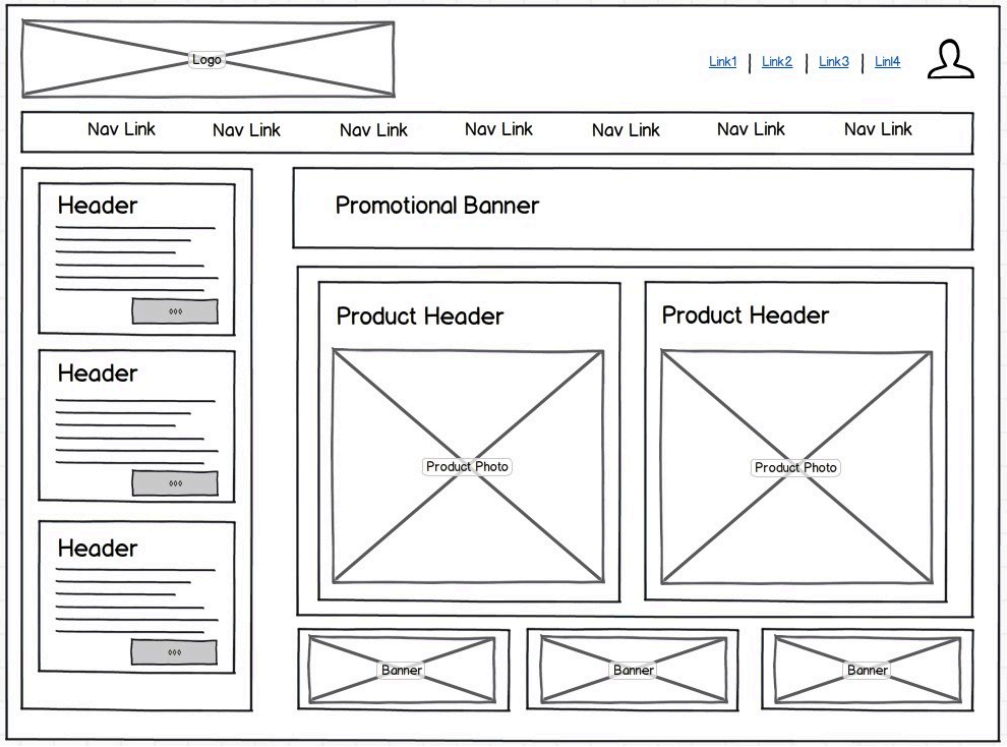
- Welche Aufgaben soll meine Webanwendung erfüllen?
- Welche Ressourcen habe ich zur Verfügung?
- Was ist meine Zielgruppe?
- Welche Browser bzw. Endgeräte sollte meine Anwendung unterstützen?
- In Hinblick auf die zu erfüllenden Aufgaben: welche Aufgabe entwickle ich server-, welche clientseitig?
- Welche Skriptsprache nehme ich für welche Aufgabe/Funktion?
- Wie bette ich die Funktionen am besten in meine Anwendung ein?
- Wie sieht meine Anwendung aus?

CLIENT/SERVER-PRINZIP
Konzeption Beispiel



https://blog.seibert-media.net/wp-content/uploads/2011/05/wireframe_beispiel_website-startseite.png

CLIENT/SERVER-PRINZIP
Konzeption Beispiel



https://blog.myhermes.de/wp-content/uploads/2013/03/Beispiel-Balsamiq-Wireframe.jpg

CLIENT/SERVER-PRINZIP
Aufgaben

- Legen Sie sich einen Account auf Github an: https://github.com
- Schicken Sie mir anschliessend Ihre Mail-Adresse, sodass ich Sie für das gemeinsame Repository berechtigen kann

- Für die Vorlesung benutzen wir das weit verbreitete Code-Versionierungssystem Git. Unser gemeinsames Repository finden Sie hier: https://github.com/p-feucht/WWI218.git

- Fügen Sie sich in Ihrem Workspace einen Ordner für alle Inhalte hinzu und klonen Sie den aktuellen Stand des Repos:
  -> Rechtsklick auf Ordner -> Open in Terminal
  -> Im Terminal eingeben: git clone https://github.com/p-feucht/WWI218.git

CLIENT/SERVER-PRINZIP
Aufgaben

- Fügen Sie sich nun einen eigenen Ordner mit Ihrem Nachnamen hinzu (im Ordner Studis)
- Legen Sie darin eine neue Datei an: index.html
- Committen Sie Ihre neuen Änderungen mit folgenden Befehlen:
  git add ./ (fügt allen neuen Dateien/Ordnern einen Stempel hinzu)
  git commit -m “was wurde verändert“ (stellt ein neues Paket zusammen)
  git push (schiebt das Paket auf den Server)

CLIENT/SERVER-PRINZIP
Linktipps

- http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Softwarearchitektur/Architekturparadigmen/Client-Server-Architektur
- http://www.informatik.uni-ulm.de/dbis/papers/vdb-buch/vdb99_12.pdf
- https://www.edv-lehrgang.de/serverseitige-clientseitige-programmierung/
- http://openbook.rheinwerk-verlag.de/c_von_a_bis_z