



FH Salzburg

# **VO Web-Technologien**

**Einheit 8, Oliver Jung**

Technik  
Gesundheit  
Medien

# Best Practices der Front-End Entwicklung



## ZIELE

Technisches Optimum anstreben

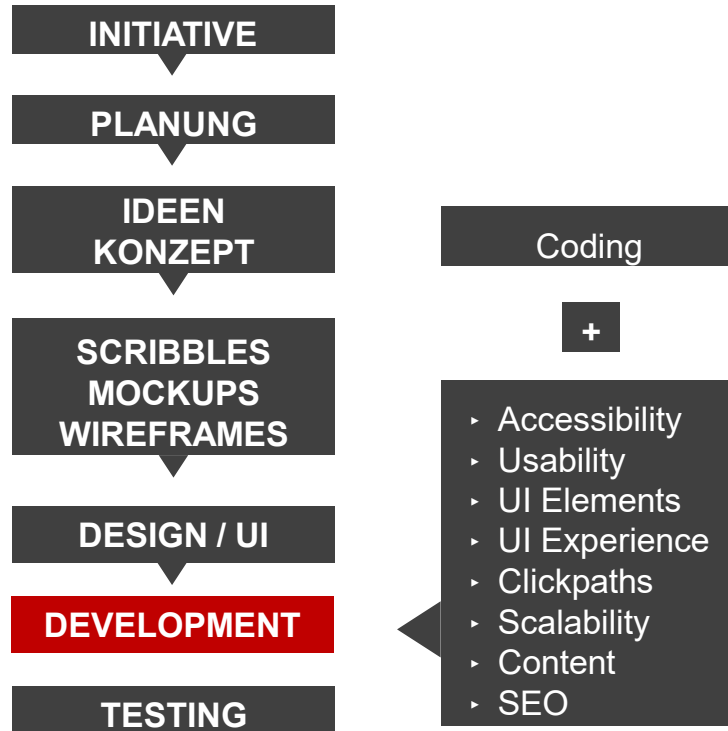
Größtmögliche Benutzerzahl erreichen

Wettbewerbsvorteil schaffen

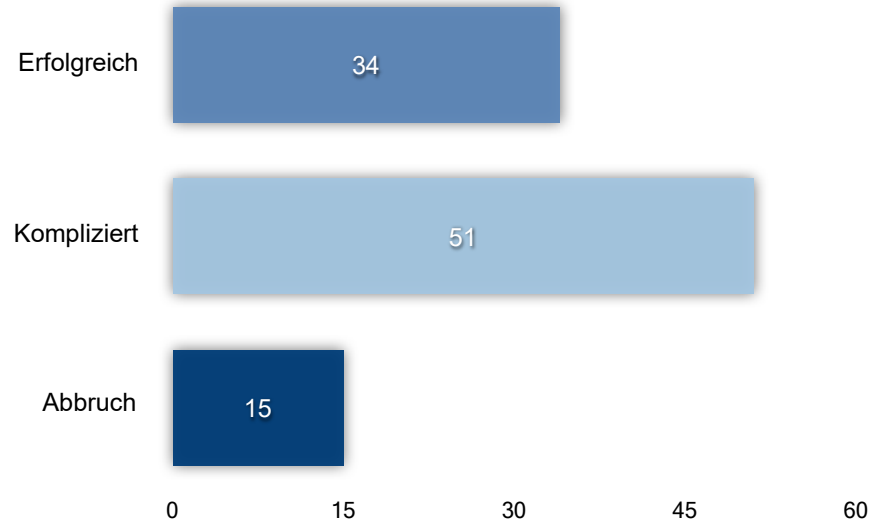
Zukunftsorientiert entwickeln

*»Continuous improvement is better than delayed perfection«*

# Bestandteile

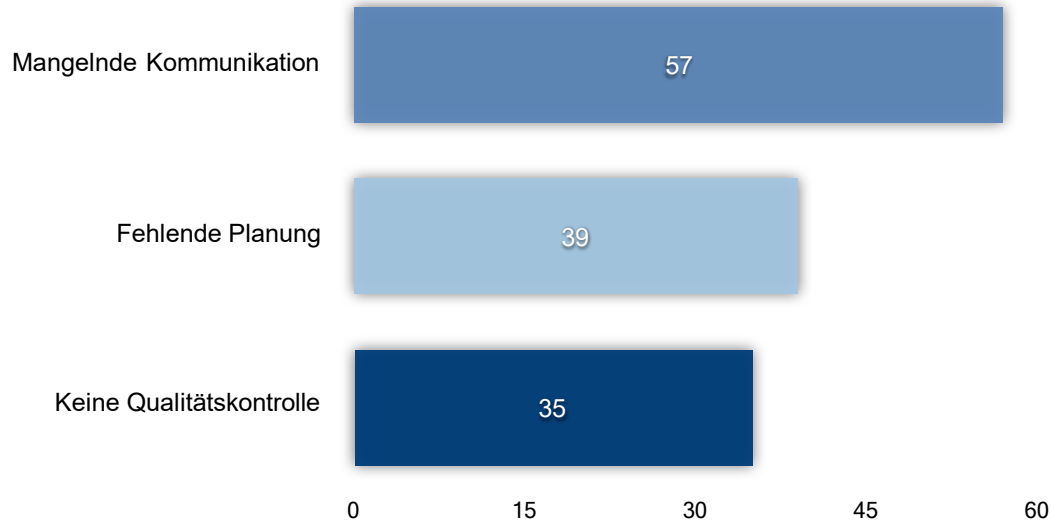


# Erfolgsraten von IT-Projekten



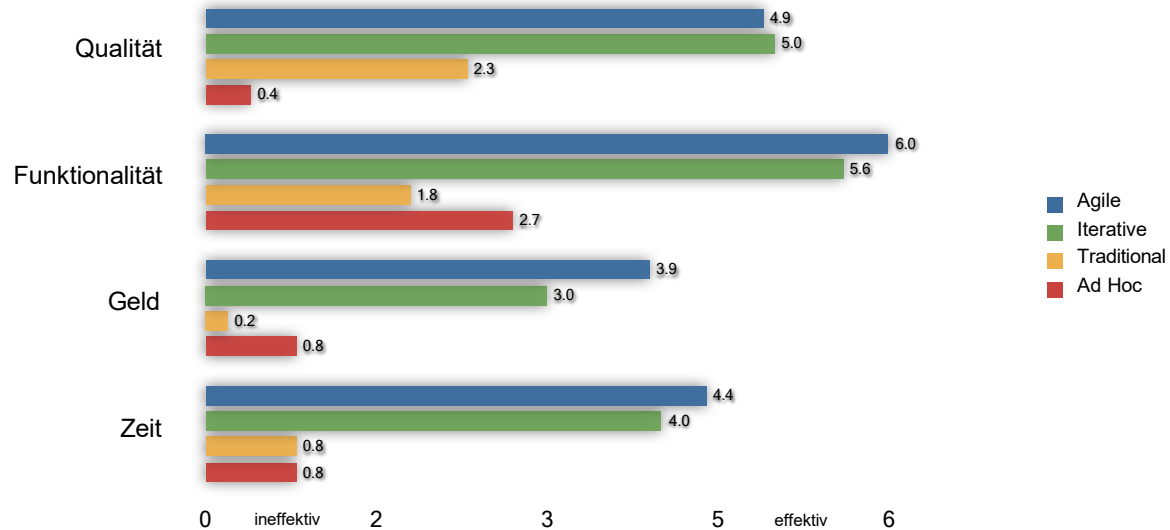
Quelle: Dr. Dobbs »The World Of Software Development« - Software Development Success Rates (24th April 2009)

# Warum IT-Projekte fehlschlagen



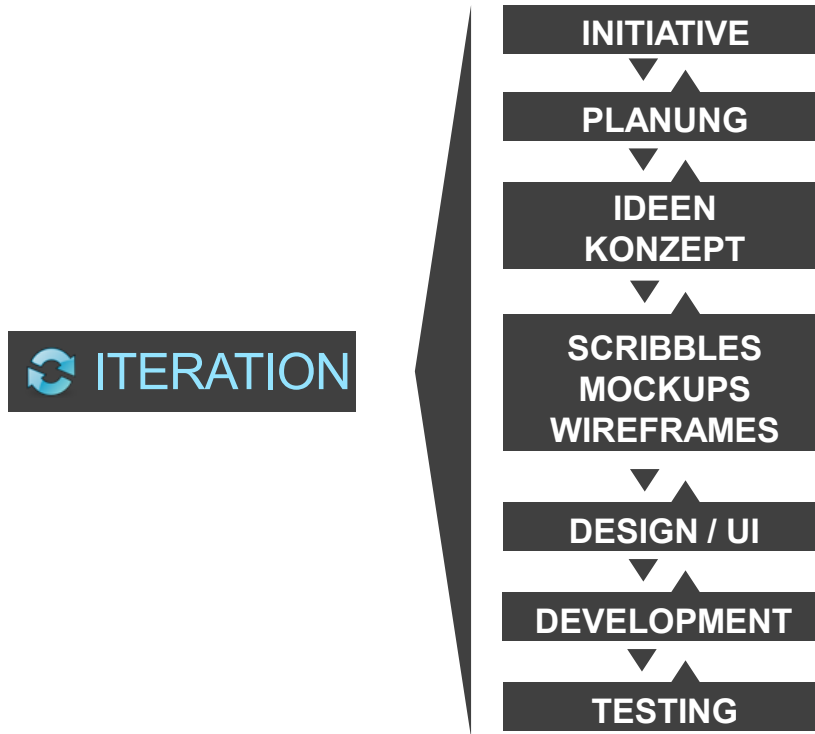
Quelle: The Bull Survey - Failure Causes Statistics auf [www.it-cortex.com](http://www.it-cortex.com) (1998)

# Effektivität von Entwicklungsparadigmen



Quelle: Dr. Dobbs »The World Of Software Development« - Software Development Success Rates (24th April 2009)

# Iteration



# Was ist Iteration?



- Feedback-Schleifen in allen Phasen
- Phasen nicht exakt getrennt, sondern ineinander übergehend
- Enge Zusammenarbeit, regelmäßige Kommunikation

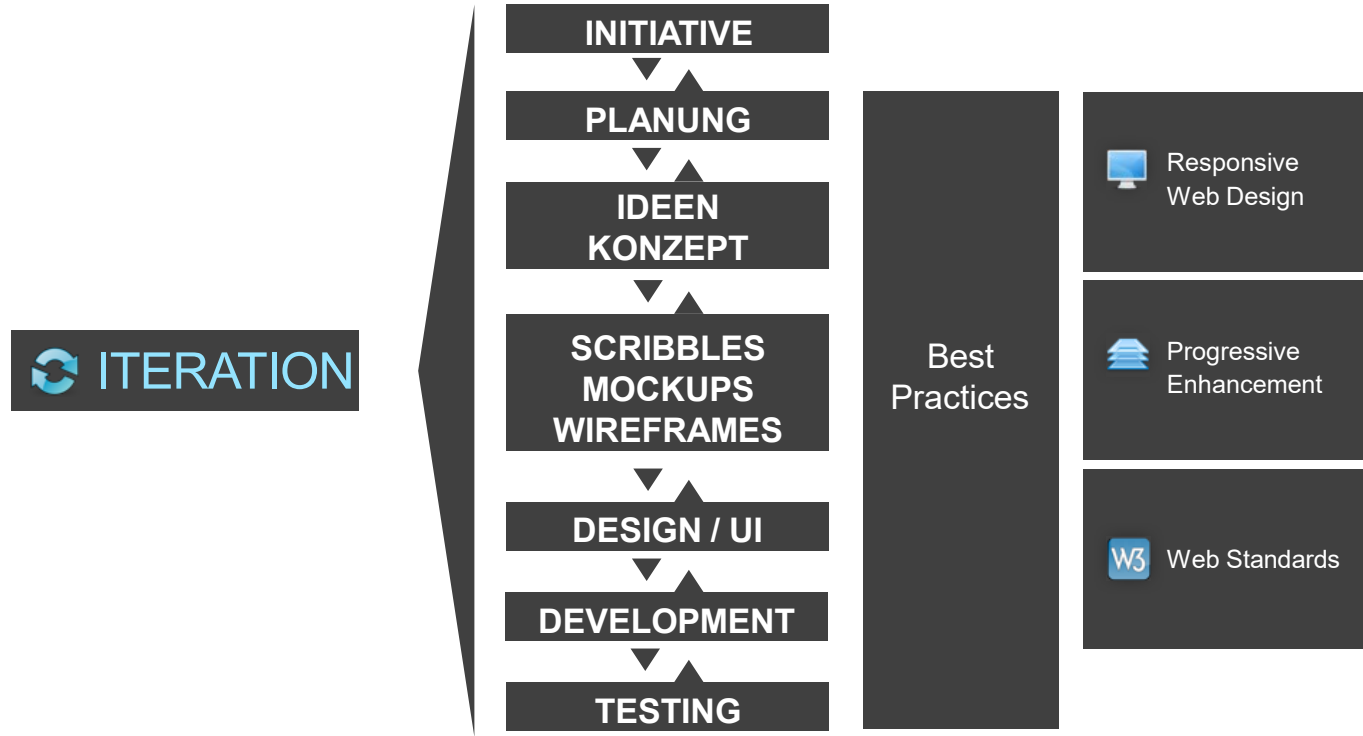


- Anwendung wird in mehrere Entwicklungsphasen zerteilt
- An deren Enden erfolgt jeweils eine Zwischenabgabe
- Team hat so die Möglichkeit, frühzeitig zu korrigieren



- Risikominimierung
- Erhöhte Qualität
- Besseres Zeitmanagement

# Best Practices



# Best Practices - Hintergrund



- Es ist einfach, diese User zu ignorieren, in dem Glauben, sie seien generell nicht technikaffin und daher nicht zur Zielgruppe gehörend.
- Unzureichender Browsersupport beschränkt sich nicht auf »Outdated Devices«. (Beispiel: Amazon Kindle mit monochrohem Display und textbasiertem Browser)
- Einige User schalten bestimmte Features ab
  - Sicherheitsgründe
  - Privatsphäre
  - schnelleres Laden
  - geringere Kosten
  - Corporate Environment

**Entwickler gehen oft davon aus, dass alle Features »enabled« sind, aber das ist eine grundlegend falsche Annahme.**

**Testing erfolgt daher auch oft in sehr sicheren und üppig ausgestatteten Umgebungen.**

- typische Bildschirmauflösung
- Default Font-Size

**Die echte Internetwelt ist viel unvorhersehbarer und unterschiedlicher.**

**Reine »Annahmen« erzeugen ein falsches Bild der Sicherheit.**


# Best Practices – Web Standards



Web-Angebote,  
die von allen Nutzern unabhängig von  
körperlichen oder technischen  
Möglichkeiten uneingeschränkt genutzt  
werden können.

**Barrierearm**

**Zugänglich**

 Web Standards

- In der EU 38 Mio. Menschen mit körperlichen Einschränkungen
- 20% der Bevölkerung > 60 Jahre
- 4/5 Behinderten nutzen das Web
- In den USA gelten 39 Mio. Menschen als behindert (15%)

- Auch **Suchmaschinen** Robots sind blinde User

## **Sinnvolles & valides HTML**

- Einhalten technischer Standards (W3C Validierung)
- Trennung von Inhalt und Layout (HTML / CSS)
- zum Inhalt passende Tags verwenden (h1-h6, p...)
- alt-Texte, link-Titel, role-Attribute

## **Navigierbarkeit**

- Seite über Tastatur navigierbar machen (tabindex)
- Navi ohne Abhängigkeit von Flash oder Javascript
- Navigation als (verschachtelte) Liste (ul > li) definieren
- Seiteninterne Sprungmarken definieren

## **Lesbarkeit & Verständlichkeit**

- Skalierbarkeit der Schriften
- blinkende und animierte Texte vermeiden
- Starke Kontraste, klare Schriften
- Text in Bildern vermeiden
- Verständliche Sprache verwenden

# Best Practices – Progressive Enhancement



Lösung für 3 konkurrierende Ziele

- Einsatz aller neuen, modernen Technologien
- universelle Zugänglichkeit
- sauberer, überschaubarer, wartbarer und modularer Code

Reines HTML, das überall funktioniert, darauf setzen CSS & JS auf.



Progressive Enhancement

*»Only after the HTML markup is as clear and descriptive as possible, we develop carefully separated CSS and JS, both written to external files, to enhance the experience«*

Javascript

Erweiterung um Dynamische UI

**Enhanced Experience**

CSS

Experience wird um Design erweitert

HTML

Funktioniert auf ALLEN Devices

**Basic Experience**

Argumente

- Anzahl an browsenden Endgeräten nimmt zu
- bestmögliche Experience für alle Benutzer
- Fokus auf Inhalt und Funktionalität
- mehr Besucher, mehr Umsatz
- positive Beispiele: Google, Facebook, Digg, Amazon

# Best Practices – Responsive Web Design



**»Mobiles Browsen wird Desktop-basierten Zugriff innerhalb der nächsten 3-5 Jahre ablösen«**



Responsive  
Web Design

## Herausforderungen

- geringe Bildschirmgröße
- mangelnder Platz für ausreichend Content
- Wechselnde Lichtverhältnisse
- Akustische Beeinträchtigungen
- Hektisches Umfeld
- Suboptimale Netzabdeckung
- Verbindungskosten und -zeit
- verschiedene gestenbasierte Interaktionsnormen (z.B. Multi-Touch: Double-Tap, Pinching)

## Statistiken

- Weltweit gibt es 5.3 Mrd. Mobilfunkverträge (77% der Weltbevölkerung)
- 25% der US-Bürger sind »mobile-only« Web-User
- Jeder 4. Deutsche hat ein Mobiltelefon (20 Mio.)
- Smartphone-Branche wiegt 400 Mrd. EUR (mehr als Automobil-Industrie)
- 2011 sind über 85% aller neuen Mobilfunkgeräte internetfähig

## Lösungen

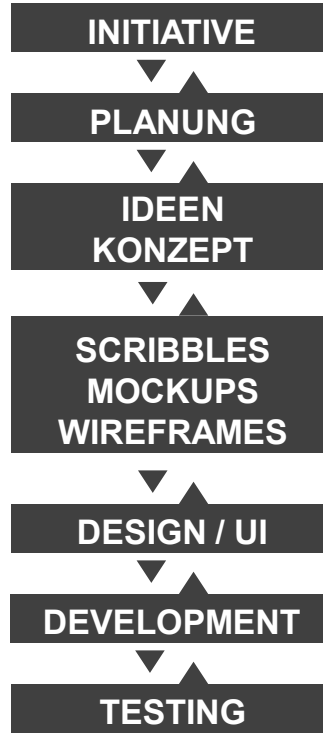
- mit relativen Größen arbeiten (keine fixen Dimensionen verwenden wie 960px)
- Container, Zwischenräume, Textgrößen und Zeilenabstände in relativen Größen definieren (bei CSS sind das em und %)
- Konzentration auf essentielle Funktionen u. Inhalte
- Angepasste Experience für die gängigsten Bildschirmgrößen bereitstellen (iPhone, iPad, iPod Touch Portrait und Landscape Modus, Android, Blackberry etc.)

# Ablauf



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



- Feedback-Schleifen
- transparente Planung
- klare Pakete / Meilensteine

# Ablauf - Initiative



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



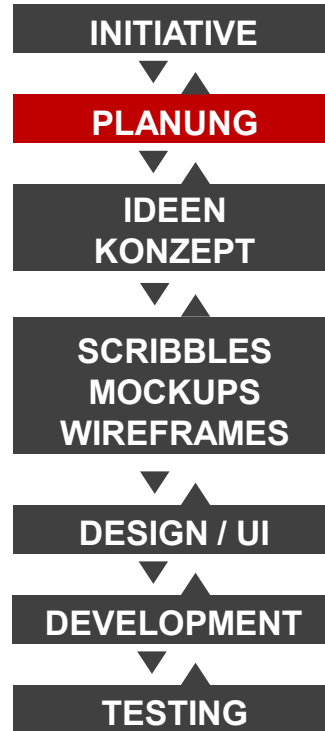
- Leistungen kommunizieren & anbieten
- Auf Vorteile hinweisen
- Wettbewerbsvorteil verdeutlichen

# Ablauf - Planung



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



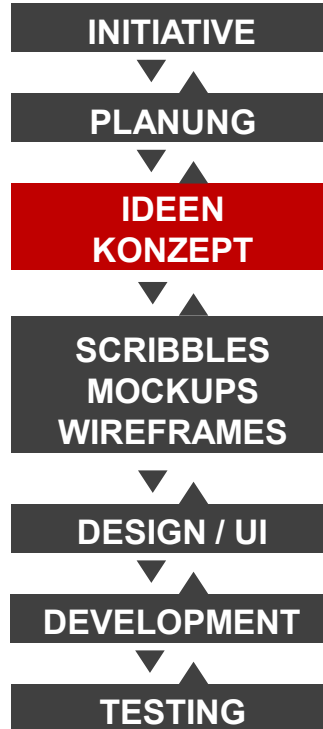
- Meilensteine / Pakete für:
  - Scribbles, Mockups, Wireframes
  - Klickbare Prototypen
  - Basic Experience
  - Enhanced Experience
  - Mobile Experience
- Nach jedem Paket Feedback & Korrektur
- Kontrolle  
(z.B. durch Kanban, Scrum)

# Ablauf - Ideen



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



- Zielgruppe
- Rollenmodelle
- Nutzungsszenarien
- Personas
- Clickpaths
- Legen einen konkreten Rahmen für Zielgruppe und Anf. an die App fest

# Ablauf - Scribbles



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



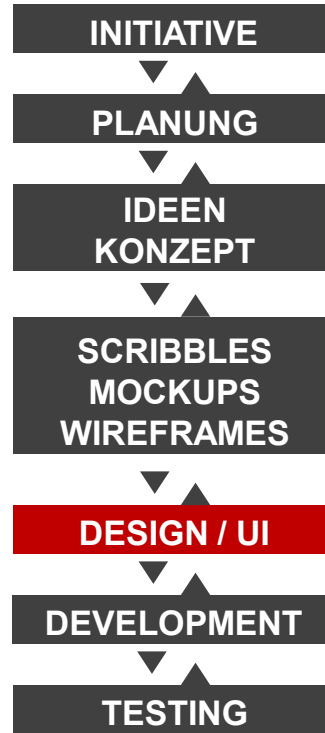
- Scribbles frei von subjektiven Einflüssen (lediglich Flächen und Texturen)
- frühe Prototypen
- verschiedene Screengrößen skizzieren
- Mapping der UI auf HTML-Elemente
- Kunde nimmt am Prozess teil
- Machbarkeit evaluieren mit Developer

# Ablauf - Design



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



- Entw. der Basic & Enhanced Experience (Erst HTML-basiert, dann "sexy")
- Prozess nah an realer Experience (Browserfonts, Container-mäßiges Design)
- Verwendung der »realen« Komponenten (aus UI-Libraries wie jQuery UI, Wijmo)
- Machbarkeit evaluieren mit Developer

# Ablauf - Development



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



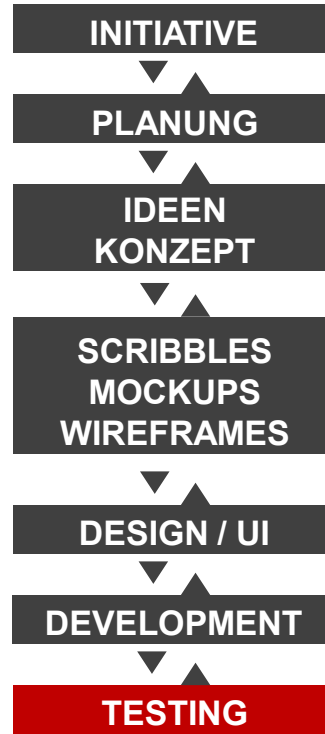
- Basic Experience
- Enhanced Experience: CSS + JS
- frühe Prototypen im Browser
- regelmäßiges Feedback mit Kunden

# Ablauf - Testing



Was können wir konkret tun?

- Iteration
- Webstandards
- Progressive Enhancement
- Responsive Web Design



- Alle Experiences ausgiebig testen (Möglichst viele Devices/Browser/OS)
- Qualitätskontrolle

# Iteration



Frühe und regelmäßige **Iteration** vereinfacht  
die nahtlose Implementierung der **Best Practices**.

Dadurch erreicht der **Kunde** mehr User,  
und jeder **User** bekommt die bestmögliche Experience.

Daher sollten technisches Optimum und  
zukunftscompatible Webseiten **unser Anspruch** sein.

# Optimierung



Die **Optimierung** von JavaScript-Quellcode ist **nicht euer Problem!** Die meiste Arbeit nimmt euch hier die **Engine** ab.

Die Optimierung des Quellcodes bringt relativ wenig und geht meistens **auf Kosten der Lesbarkeit.**

# Was zählt?



Was zählt?

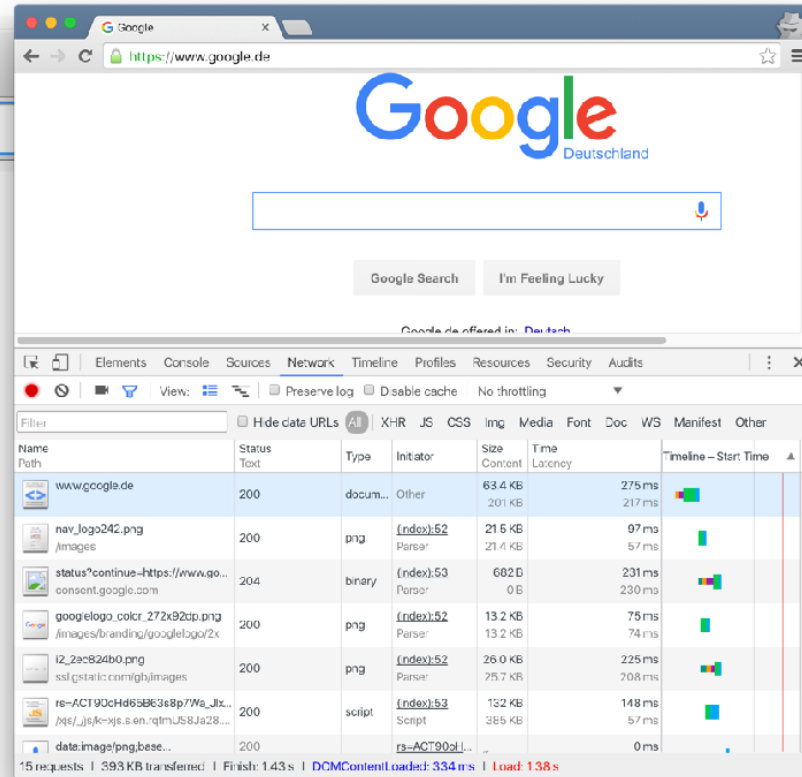
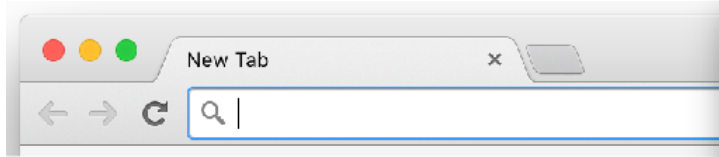
Die **Zeit** bis der Nutzer die ersten **Informationen sieht** und mit der Seite **interagieren** kann.

# Critical Rendering Path



Der **Prozess** zwischen dem **Empfang** der HTML-, CSS- und JavaScript-**Dateien** und der **Darstellung** der Information im Browser.

# Critical Rendering Path

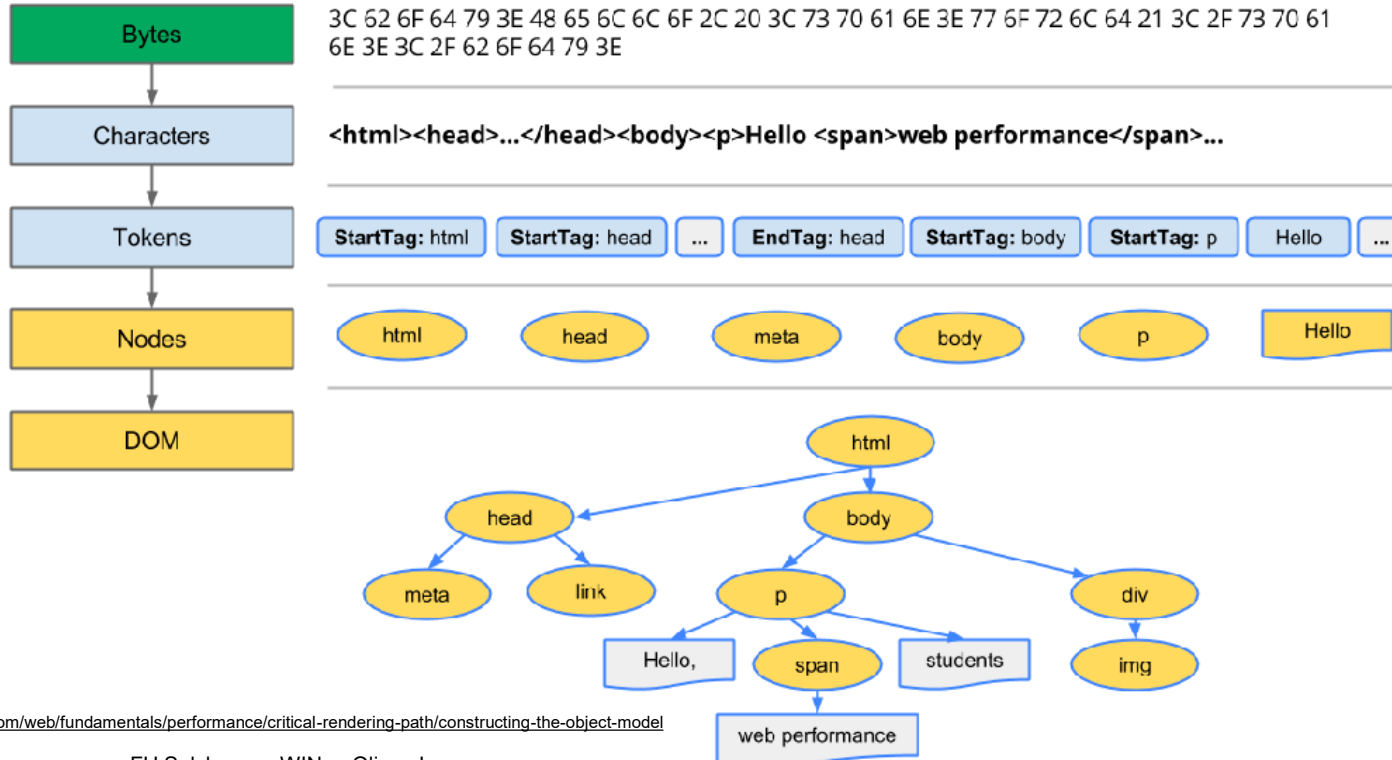


# Critical Rendering Path



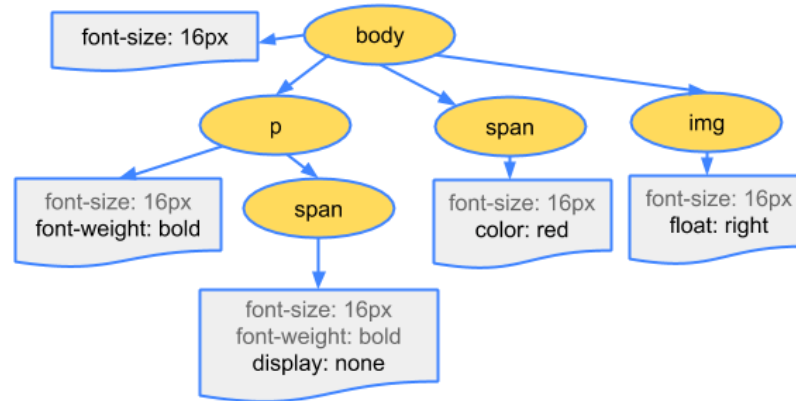
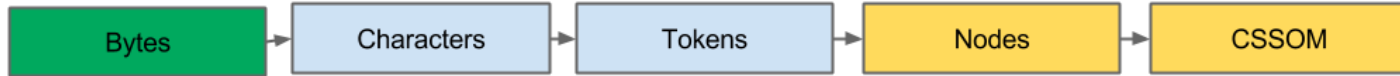
1. HTML Verarbeitung -> DOM
2. CSS Verarbeitung -> CSSOM
3. Render Tree Erzeugung
4. Darstellung

# HTML Verarbeitung

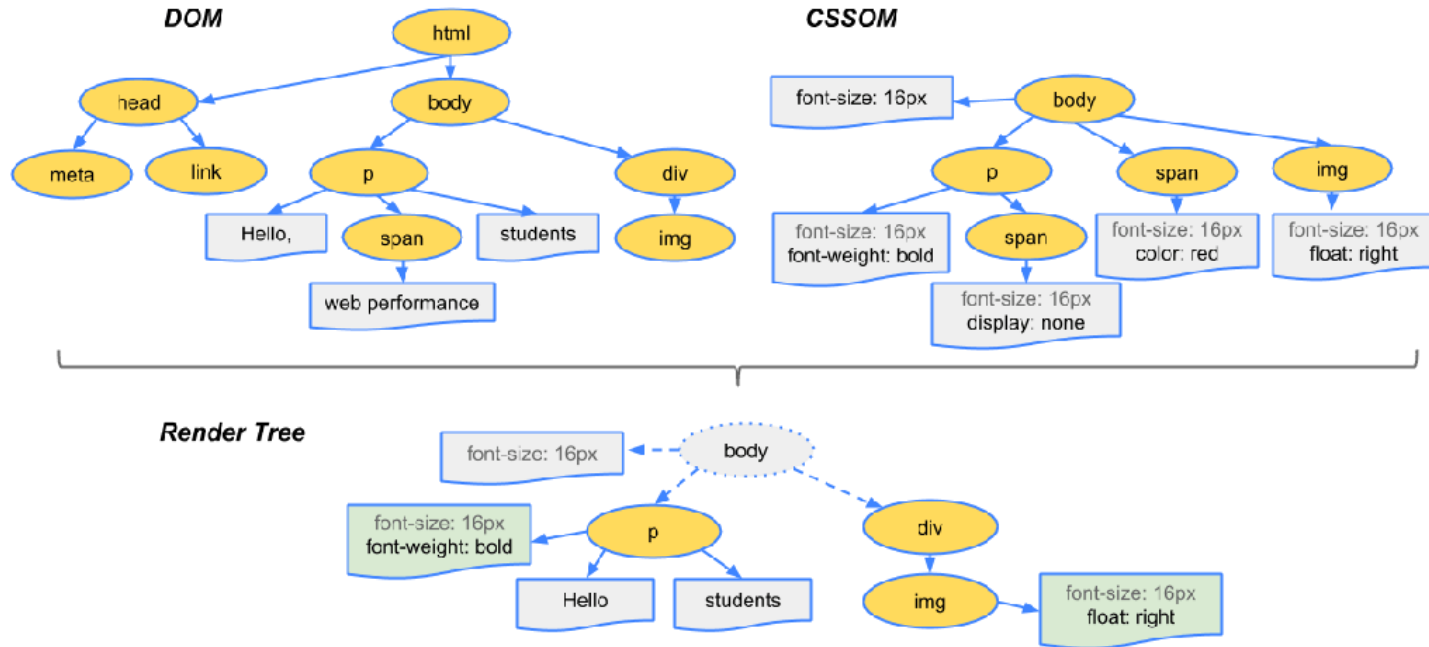


<https://developers.google.com/web/fundamentals/performance/critical-rendering-path/constructing-the-object-model>

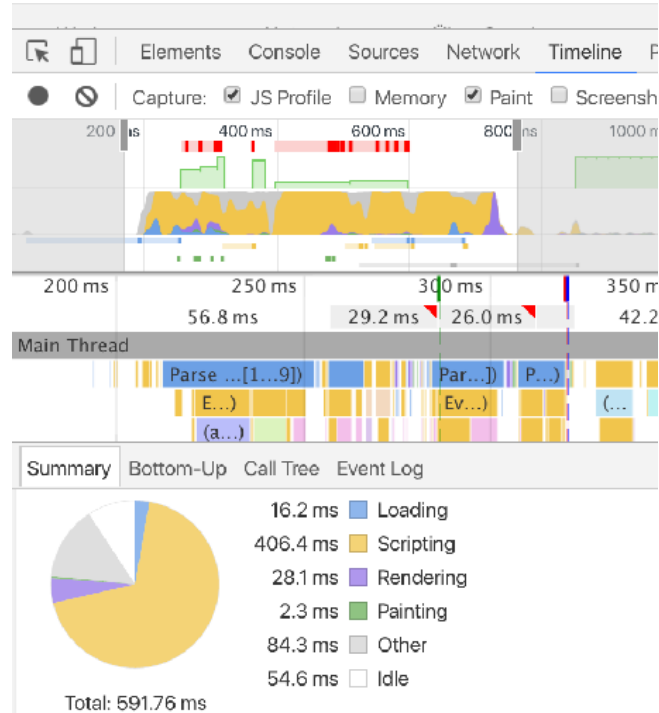
# CSS Verarbeitung



# Render Tree



# Messung



# Download von Dateien



Ein Browser hat nur eine **beschränkte Anzahl** von **parallelen Verbindungen** zu einem Server.

**Firefox: 6**

**Chrome: 6**

**Internet Explorer: 13**

# Download - Lösung



Möglichst **wenige**, möglichst **kleine** Dateien ausliefern.

**CSS**: Präprozessoren + Minifier

**JS**: Modulsystem + Minifier

**Images**: Sprites

<https://minify.js.org/>

# Service Worker



Service Worker sind separate Prozesse im Browser, die als Proxy zwischen Client und Server arbeiten. Sie können Anfragen abfangen und aus ihrem Cache direkt beantworten. Web-Applikationen werden damit wesentlich schneller und können auch offline genutzt werden.

# Render Blocking CSS



Jede CSS-Datei **blockiert** die **Darstellung** der Seite für kurze Zeit.

Je **mehr CSS** verwendet wird, desto **länger** dauert die **Blockade**.

# Render Blocking CSS



Kein **@import** verwenden - lieber alles in eine Datei.

**Weniger CSS** im Critical Rendering Path verwenden, z.B.  
**media**-Attribut im style-Tag.

Wenn sonst nichts hilft: Inline CSS.

# Render Blocking JavaScript



Alles JavaScript, das **nicht** direkt zum **initialen Pageload** benötigt wird, kann **nachgeladen** werden.

Z.B. über ein **generiertes Script-Tag** beim load-Event.

oder **lazy loading** Funktionalität des Modulloaders (z.B. webpack)

# App Shell Architecture



**Minimales** HTML, CSS und JavaScript  
als Grundlage für das **User Interface**.

- **lädt** schnell
- kann **gecacht** werden
- zeigt **dynamischen** Inhalt an

# Single Page-Applikationen



Der Benutzer bewegt sich in einer **Applikation**. Die Applikation bleibt über **längere Zeit** im Browser geöffnet.

Daten werden bei Bedarf **nachgeladen**.

Es erfolgen **keine Pageloads**.

# Single Page-Applikationen



Der **State** der Applikation wird **im Speicher** gehalten.

Die **Repräsentationen** vieler Objekte liegen **im Speicher**.

Der **Speicherverbrauch steigt** über die Laufzeit an.

# JIT (Just-in-Time) Compiler



V8 (Chrome JS Engine) kompiliert **JavaScript** bei der ersten Ausführung in **Maschinencode**.

Die passenden **Hidden Classes** werden bestimmt und der Code entsprechend **gepatcht**. Auch alle zukünftigen Verwendungen des Objekts werden mit der Hidden Class versehen und bei Bedarf von der Engine **korrigiert**.

# Garbage Collection



Der Garbage Collector **prüft** regelmäßig den **belegten Speicher** und **löscht** nicht mehr verwendete Informationen.

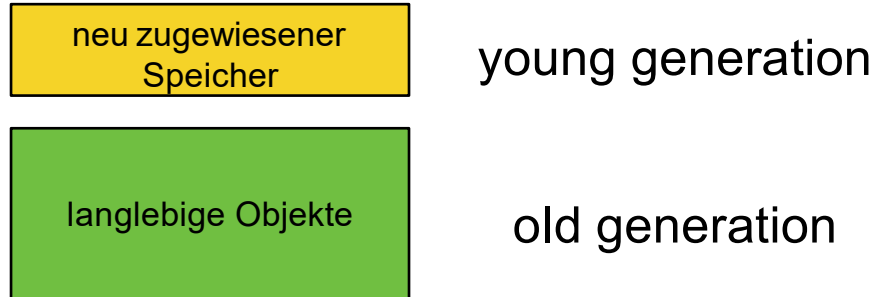
Nicht mehr verwendet heißt: **keine Referenz** mehr auf ein Objekt.

V8 verschiebt das **Memory Management** möglichst in ungenutzte **Idle Time** des Prozessors, um den Impact auf die Darstellung zu minimieren.

# Garbage Collection

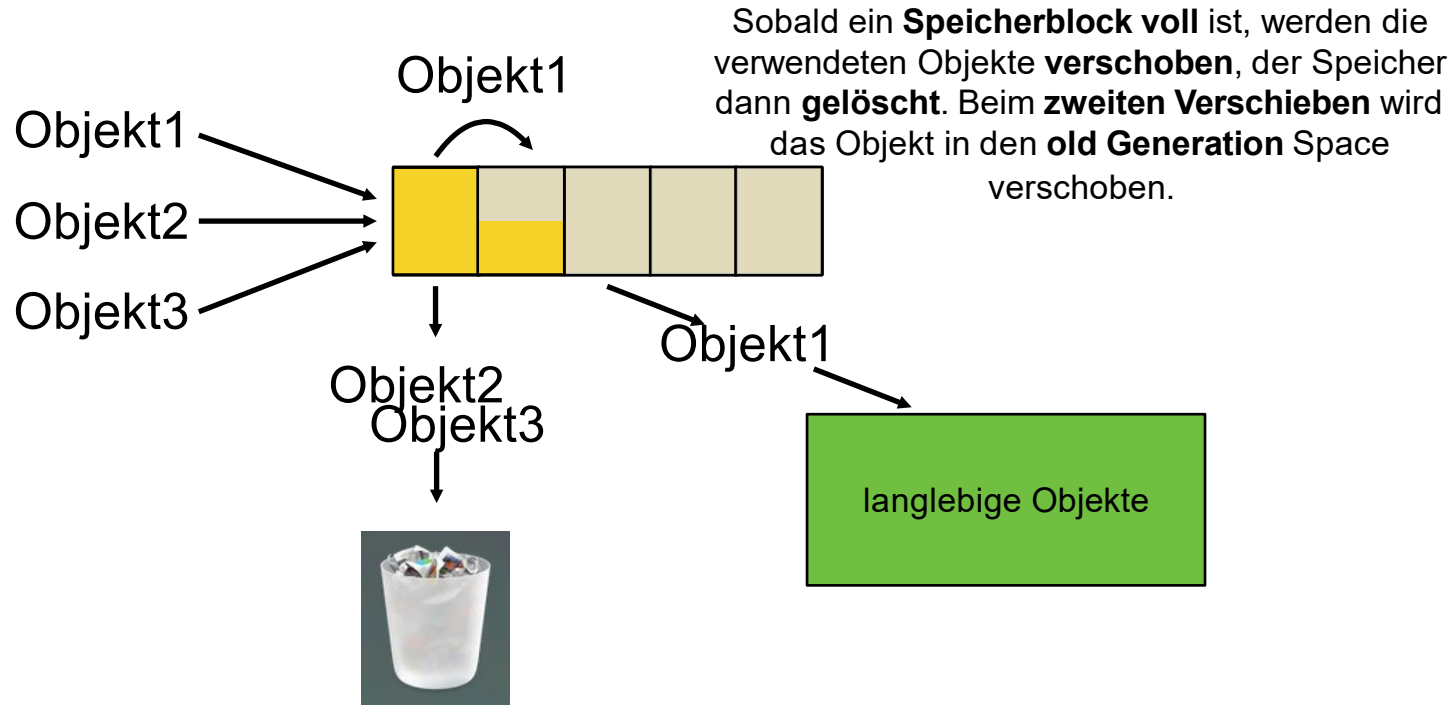


Speicheraufteilung:



Die meisten Objekte haben nur eine sehr kurze Lebensspanne.

# Garbage Collection



# Garbage Collection



Je weniger Objekte verschoben werden müssen, desto schneller ist die young generation Garbage Collection.

# Garbage Collection



Überschreitet der **old generation** space ein bestimmtes Limit, wird der Bereich mit einem **mark-and-sweep Collector** aufgeräumt.

Aktive Objekte werden markiert und anschließend alle nicht markierten Objekte gelöscht.

Ein **kompletter Durchlauf** kann **100 ms** dauern. Die Applikation wird in dieser Zeit angehalten.  
V8 kann diesen Prozess auch **inkrementell** in **5 ms**-Schritten durchführen.

# Repaints & Reflows



**Repaint:** Der Browser überprüft alle Elemente auf ihre **Sichtbarkeit, Farbe, Abmessungen** und andere **visuelle Eigenschaften** und aktualisiert die relevanten Teile des Bildschirms.

**Reflow:** Der Browser **berechnet** das **Layout** der Seite. Reflows für weitere Elemente können ausgelöst werden (Kinder, benachbarte Elemente, im DOM folgende Elemente). **Danach** wird ein **Repaint** ausgelöst.

# Auslöser für Reflows



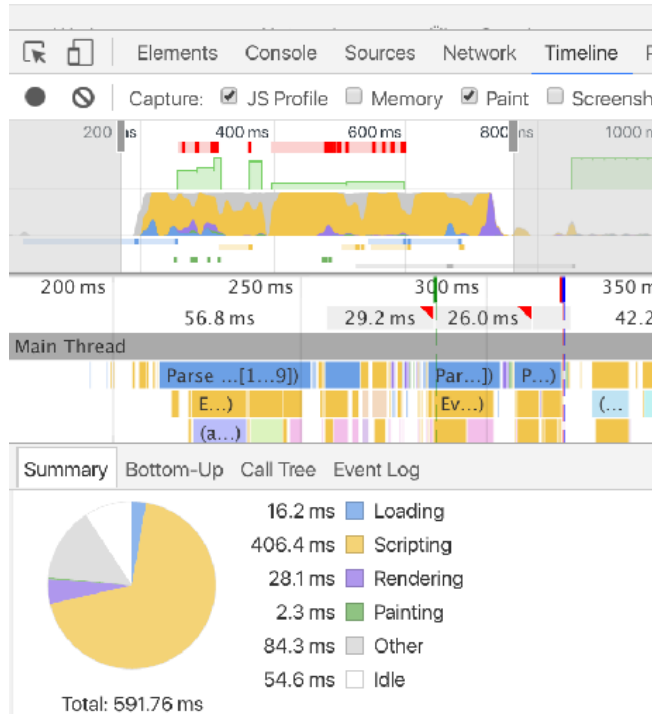
- Einfügen, Entfernen oder Aktualisieren eines DOM Elements
- Verändern des Inhalts einer Seite
- Verschieben eines Elements
- Animationen
- Abmessungen auslesen
- CSS-Eigenschaften ändern
- Klassennamen eines Elements ändern
- Stylesheet hinzufügen oder entfernen
- Fenstergröße ändern
- Scrollen

# Vermeiden von Reflows



- Folgen von **einzelnen Styleänderungen vermeiden**
- **Operationen** über Klassennamen **zusammenfassen**
- **Operationen außerhalb** des **DOMs** durchführen und danach einhängen
- **Styles** in Variablen **cachen**
- Für **Animationen** besser **fixe Positionierung** wählen

# Profiling



**Rendering:** Layoutberechnungen

**Painting:** Darstellung der Seite

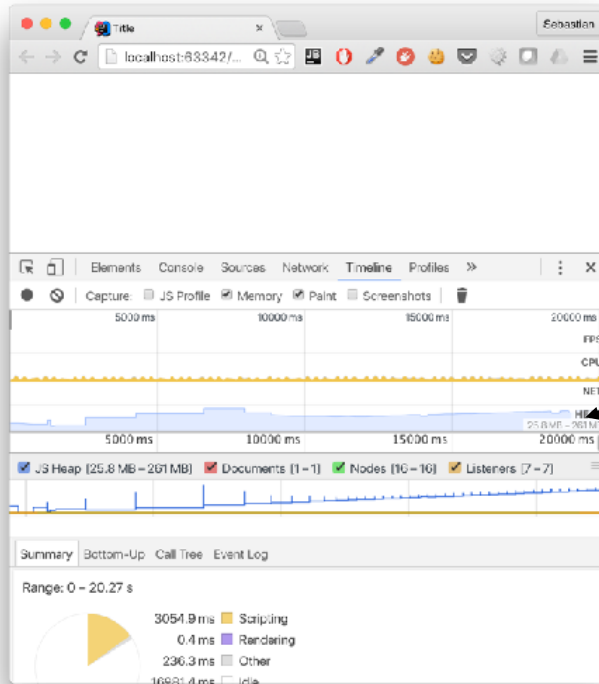
# Memory Leaks



Machen unsere **Applikationen langsam**, führen zu **Abstürzen** und **hohen Latenzen**.

Ein Memory Leak tritt auf, wenn **Speicher** nicht mehr gebraucht wird, aber **nicht** zur Garbage Collection **freigegeben** wird.

# Memory Leaks



Memory läuft langsam voll

# JavaScript Animationen



Bei einer Animation wird nach einer bestimmten Zeit eine **CSS-Eigenschaft** eines Objekts verändert. Ist die gewählte **Zeitspanne** gering genug, entsteht eine mehr oder weniger **flüssige Animation**.

# JavaScript Animationen - Nachteile



**JavaScript** wird über die **CPU** ausgeführt. Muss sich also die **Ressourcen** mit vielen anderen Programmen **teilen**.

**GC-Cycles** können zu **unschönen Effekten** führen, da die Ausführung angehalten wird.

Bei hoher **CPU-Last** sind die Animationen **nicht** mehr **flüssig**.

# CSS-Animationen



**CSS-Animationen** werden durch die **GPU** berechnet und belasten die CPU nicht.

CSS-Animationen erscheinen **flüssiger** als JavaScript-Animationen.

Der **Browser** kann die Animationen **optimieren** (z.B. bei nicht sichtbaren Tabs).

# CSS-Animationen



Wenn **Transitionen** nicht reichen, kann man über Animationen mit **@keyframes** noch wesentlich mehr herausholen.

Es gibt auch **Generatoren** wie z.B. <https://animista.net/>

Hilfestellung: [https://www.w3schools.com/cssref/css3\\_pr\\_animation-keyframes.php](https://www.w3schools.com/cssref/css3_pr_animation-keyframes.php)

# Prefetching

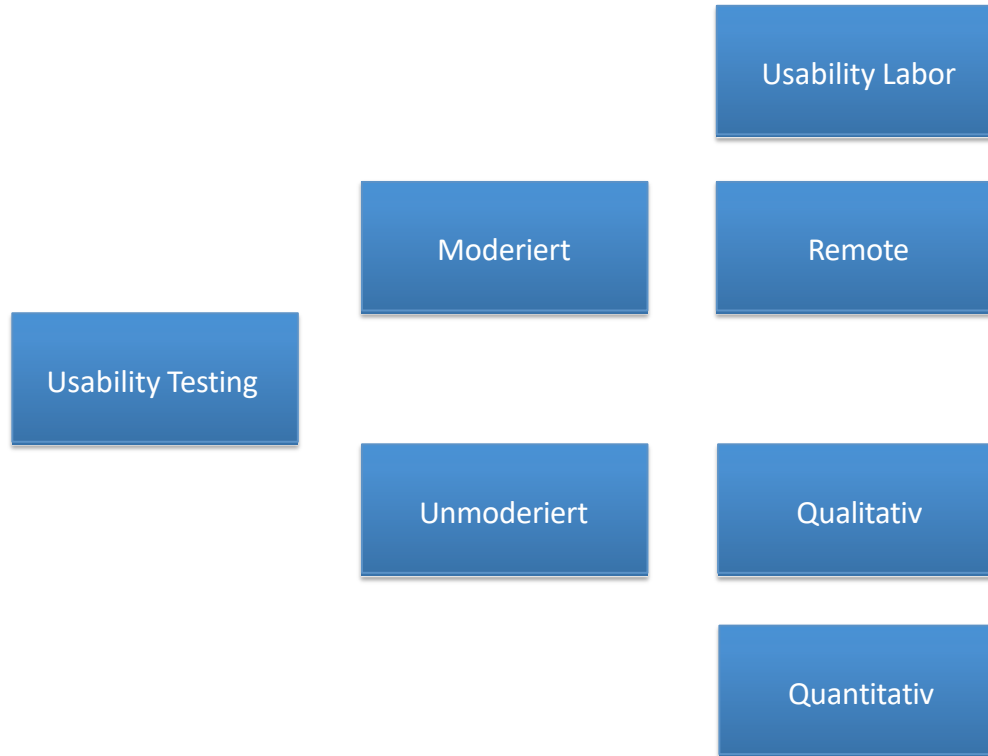


```
<link rel="prefetch" href="users.html" />
```

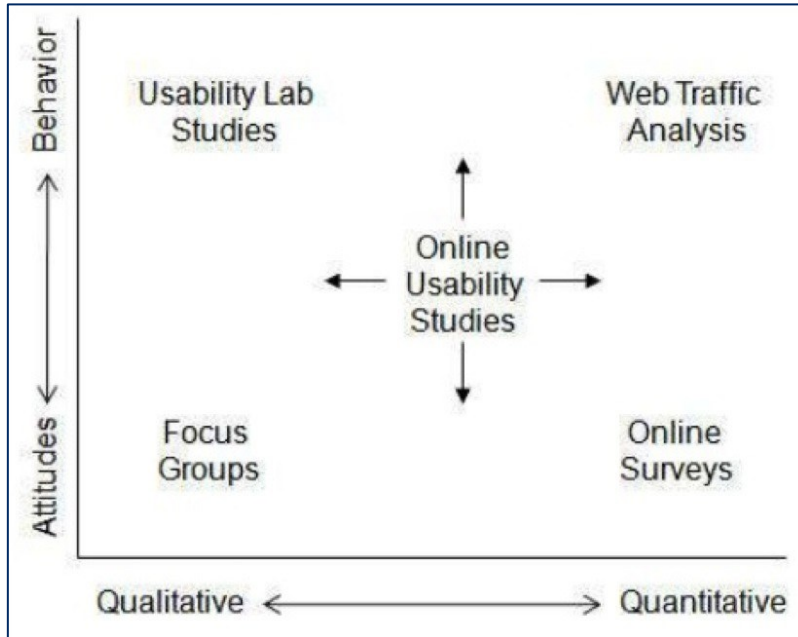
Sorgt dafür, dass der Browser gewisse **Seiten** bereits **vorlädt**, um sie schneller laden zu können.

Chrome und IE unterstützen auch **Prerendering**, bei dem die Seite bereits vorgerendert wird.

# UX Research – Überblick & Kontext



# Unmoderated Remote Usability Testing



# Ablauf



1. Studie konzipieren und implementieren, testen

2. Teilnehmer einladen

3. Hunderte Nutzer nehmen vom eigenen Endgerät aus teil

4. Daten werden im Tool aggregiert und zur Auswertung vorbereitet

The screenshot displays the UserZoom web interface. On the left, a sidebar menu includes 'Project Editor', 'Project Setup', 'Project Design', and 'Task Editor'. The main area shows 'TASK 1: NAVIGATION - TASK SETTINGS & VALIDATION' with tabs for 'Settings', 'Validation', and 'Video'. Below this, there's a 'Task Description & Starting URL' section. The 'Task Settings & Validation' section shows a list of tasks, including 'Task 1: Navigation - W', 'Task 2: Card Sorting', and 'Task 3: Card Sorting'. The 'Task 1: Navigation - W' task is selected, showing its details and a list of participants. The 'Task 2: Card Sorting' task is also visible, showing its details and a list of participants. The 'Task 3: Card Sorting' task is also visible, showing its details and a list of participants. The 'Task 1: Navigation - W' task is selected, showing its details and a list of participants. The 'Task 2: Card Sorting' task is also visible, showing its details and a list of participants. The 'Task 3: Card Sorting' task is also visible, showing its details and a list of participants.

# Stärken & Schwächen

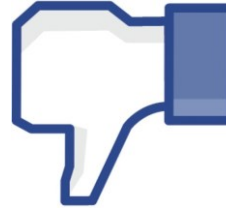


- ✓ Kosten, Ressourcen, Skalierbarkeit, Agilität
  - ✓ „Wir haben keine Zeit / kein Budget für Usability Testing“
  - ✓ „Dafür lohnt es sich nicht, eine Usability Testing Studie durchzuführen“
- ✓ Quantitativ & qualitativ
  - ✓ „Wir müssen Designentscheidungen absichern“
  - ✓ „Uns interessieren auch Meinungen, subjektive und emotionale UX Faktoren“
  - ✓ „Wir wollen vergleichen – sind wir besser geworden? Besser als Wettbewerber?“
  - ✓ „Wir brauchen überzeugende Daten für die interne Kommunikation“
- ✓ Globale Reichweite, realer Kontext
  - ✓ „Unsere Nutzer sind weltweit verteilt / haben keine Zeit in ein Usability Labor zu kommen“
  - ✓ „Im Usability Labor fehlt den Nutzern doch der natürliche Kontext“

# Stärken & Schwächen



- Spart Zeit
- Spart Geld
- Skalierbar
- Flexibel & agil
- Statistisch nutzbare Messwerte
- Vergleich (Zeit, Länder, Wettbewerb...)
- Verfügbarkeit von Teilnehmern
- Natürlicher Kontext
- Kommunikation



- Keine Nachfragen möglich
- Weniger geeignet für rein explorative Beobachtung oder extrem lange Tasks
- Nur Web- und mobile Uis
- Keine Vertraulichkeit

# Planung & Vorbereitung



- Studienziele, z.B.
  - Usability Probleme identifizieren
  - User Experience messen
  - Vergleich von Design-Varianten (A/B)
  - Benchmarking mit dem Wettbewerb
  - Navigationsstruktur optimieren ("Findability")
  - Emotionale Ansprache / Markenwahrnehmung der Nutzer messen
  - Iterativ und agil den Designprozess begleiten

# Planung & Vorbereitung

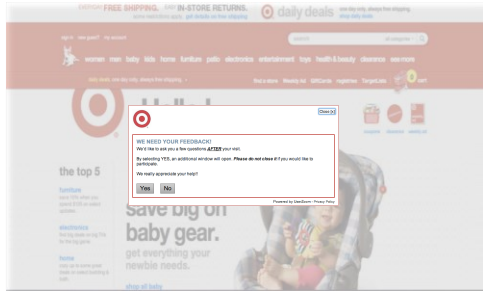



- Studienziele, z.B.
  - Usability Probleme identifizieren
  - User Experience messen
  - Vergleich von Design-Varianten (A/B)
  - Benchmarking mit dem Wettbewerb
  - Navigationsstruktur optimieren ("Findability")
  - Emotionale Ansprache / Markenwahrnehmung der Nutzer messen
  - Iterativ und agil den Designprozess begleiten
- Methode/Tool auswählen
  - Remote aufgabenbasiert (+ Lab)
  - Card Sorting / Tree Testing
  - Voice of Customer (VoC)
  - VoC + Remote aufgabenbasiert

# Rekrutierung von TeilnehmerInnen



- Internationale Online Panel Anbieter
- On-Site Rekrutierung
- E-Mail
- Social Networks
- Datenschutz beachten





Close [x]

**WE NEED YOUR FEEDBACK!**

We'd like to ask you a few questions **AFTER** your visit.

By selecting YES, an additional window will open. **Please do not close it** if you would like to participate.

We really appreciate your help!!

Powered by UserZoom - Privacy Policy

# Rekrutierung von TeilnehmerInnen

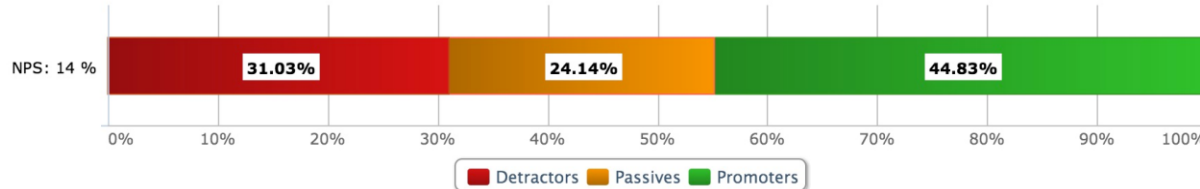


- Teilnehmerprofile hängen von der Fragestellung/Zielsetzung der Studie ab.
  - Soziodemographische Daten
  - Zielgruppe
  - Relevantes Fachwissen
  - Verhaltensdaten
- Sind Profildaten relevant für Auswertung/Design?
- Gewünschtes Konfidenzintervall definiert Anzahl der Testpersonen
- Kosten: 6-10 € / Person bei Panelanbietern, nur “Completes”

# Durchführung



- Fragen und Fragebögen
  - Vorerfahrung mit Website (→ Logik zur Verzweigung einsetzen)
  - Filter-Fragen (z.B. Demographie, Nutzungskontext)
  - Hypothetische Fragen vermeiden (“Würden Sie, ..., wenn...?”)
  - So konkret wie möglich (“täglich” statt “häufig”)
  - Doppelfragen vermeiden (“Wie einfach und schnell war das...?”)
- Standard-Fragebögen:
  - SUS, SUMI, ISONORM...
  - NPS
  - Eigene Standard-Fragebögen



# Durchführung



- Ergänzende Online Methoden und Fragearten
  - Screenshot Click Testing
  - Multimedia-Fragen
  - Tree Testing
  - Skalen, Semantische Differentiale, NPS
  - Sortieraufgaben

Wonach wählen Sie die Website aus, auf der Sie Kleidung kaufen? Bitte sortieren Sie die folgenden Elemente danach, wie wichtig sie für Ihre Entscheidung sind.

Ziehen Sie die Elemente von links nach rechts, um sie zu sortieren.

Kenne die Website schon	Am wichtigsten	1
Einfache Bedienbarkeit		2
Günstige Preise		3
Angebotsvielfalt		4
	Am wenigsten wichtig	

# Durchführung



Los geht's mit der ersten Aufgabe. Bitte klicken Sie auf dem Screenshot der bild.de Homepage **einmal** dorthin, wo Sie Informationen dazu erwarten würden, wie viel die digitale Ausgabe der BamS (Bild am Sonntag) kostet.



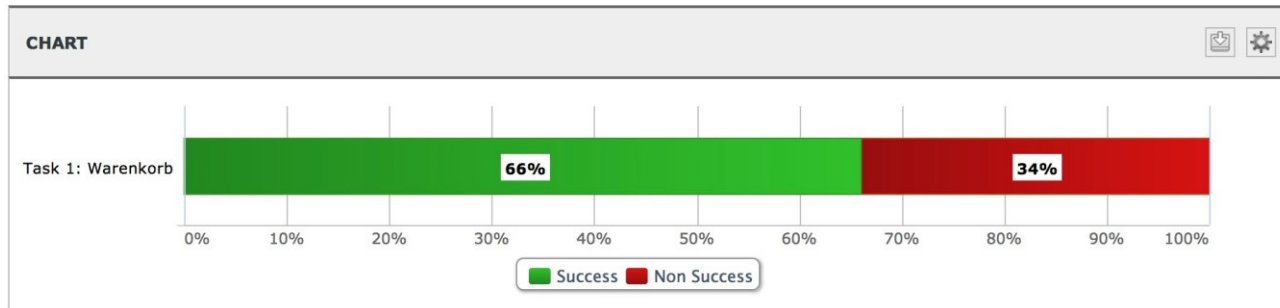
Graph  Scope  Users: 5 | Clicks: 5 ☐ Scroll lines



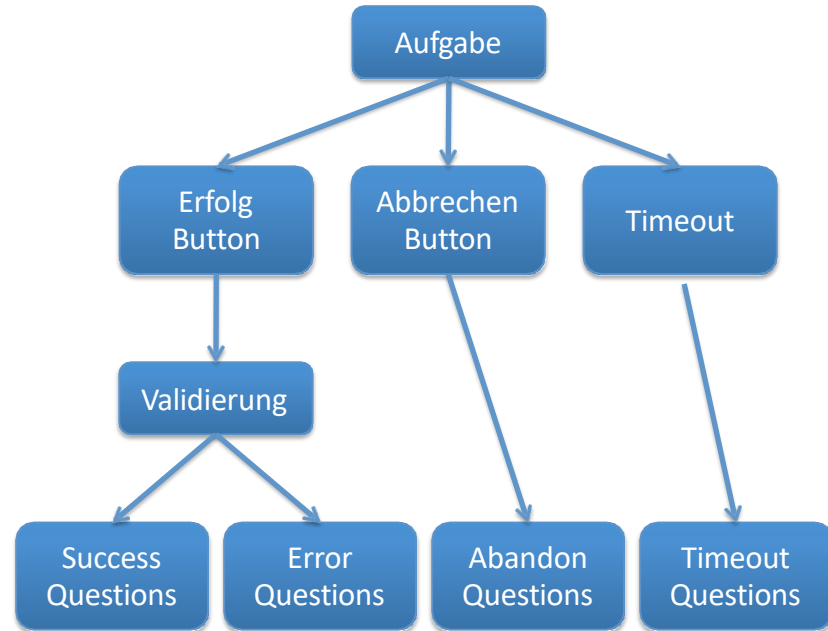
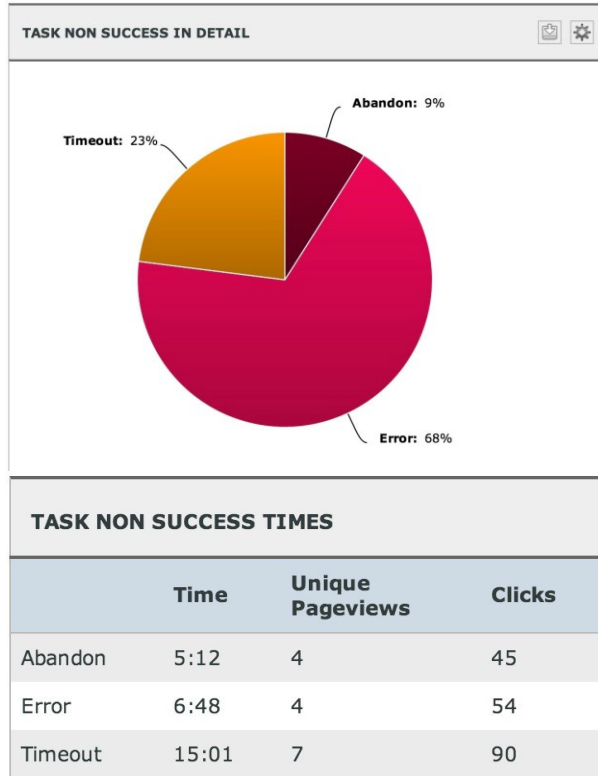
# Ergebnisse & Auswertung



- Ergebnisse & Auswertung – typische Metriken
  - Completion Rate (Success Rate)
    - Error Rate
    - Abandon Rate
    - Timeout Rate
  - Time on Task (auch subjektiv)
  - Bewertungen (Usability, Content, Zufriedenheit – pro Aufgabe und insgesamt)
  - Usability Probleme (Anzahl, Schwere)



# Ergebnisse & Auswertung



# Ergebnisse & Auswertung



- Quantitative UX Messdaten
- Quantitative “Self-reported” Daten
- Qualitatives Nutzerfeedback
- Verhaltensdaten (Clickstreams, Heatmaps)
- Qualitative, quantifizierbare Video Screen Recordings

# Fazit



URUT löst drei Herausforderungen für UX Researcher heute:

- Quantitative Messdaten komplettieren rein qualitative Daten:  
Risikomanagement, Vergleich, Kommunikation
- URUT ist vergleichsweise kostengünstig, agil und skalierbar und ermöglicht eine höhere Testfrequenz
- URUT erreicht Nutzer in ihrem natürlichen Kontext



**FH Salzburg**

# **VO Web-Technologien**

**Einheit 8, Oliver Jung**