31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Der Schmetterlingseffekt

CSS in größeren Projekten

Sebastian Reiners

open knowledge GmbH

Worum soll es gehen?

- Was haben Schmetterlinge und CSS gemeinsam?
- Was macht ein wartbares CSS aus?
- Was kann mir die Arbeit erleichtern?
- Bekomme ich das alles in meine Projektabläufe integriert?

Kleine Ursache ...

Wir haben die Webseite im Vorstand präsentiert.
Die hätten das Ganze lieber in wärmeren Farben. Mit wesentlich mehr Blautönen!

Kleine Ursache ...

Ja, aber ...

Und die Navigation soll statt oben an der linken Seite sein.

Aber in 2 Wochen ist der Livegang!

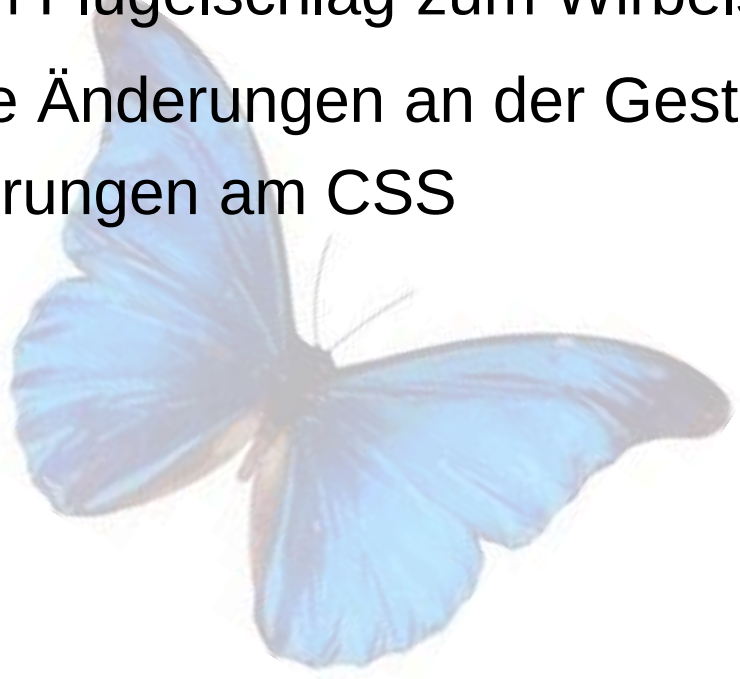
Kleine Ursache ...

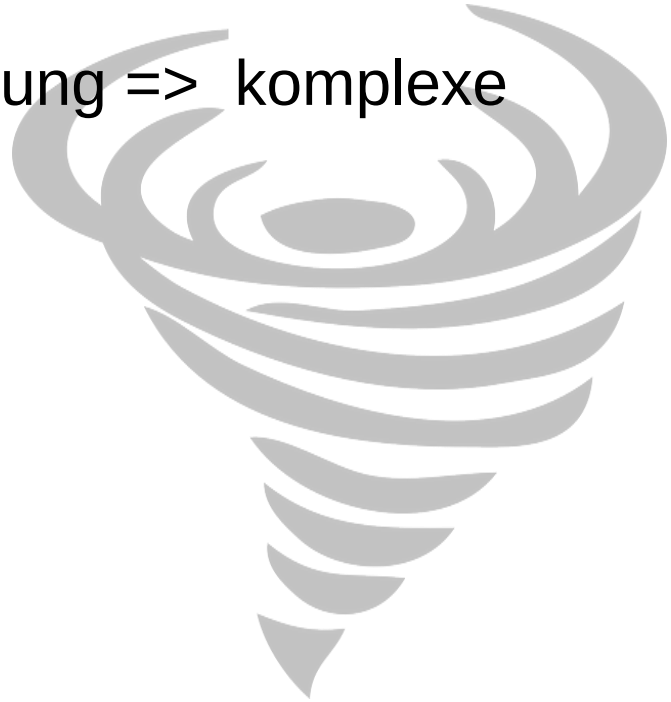
Das weiß der Vorstand. Deshalb muss das auch diese Woche fertig werden, damit sie nochmal drüberschauen können.

Außerdem ist Blau keine warme Farbe!

... und große Wirkung

- Der Schmetterlings-Effekt
  - vom Flügelschlag zum Wirbelsturm
- kleine Änderungen an der Gestaltung => komplexe Änderungen am CSS





Inhalt

1. Ausgangslage
2. Aufbau und Strukturierung
3. CSS Frameworks
4. CSS Preprocessing
5. Qualitätssicherung und Buildprozess
6. Fazit

Größere Projekte?

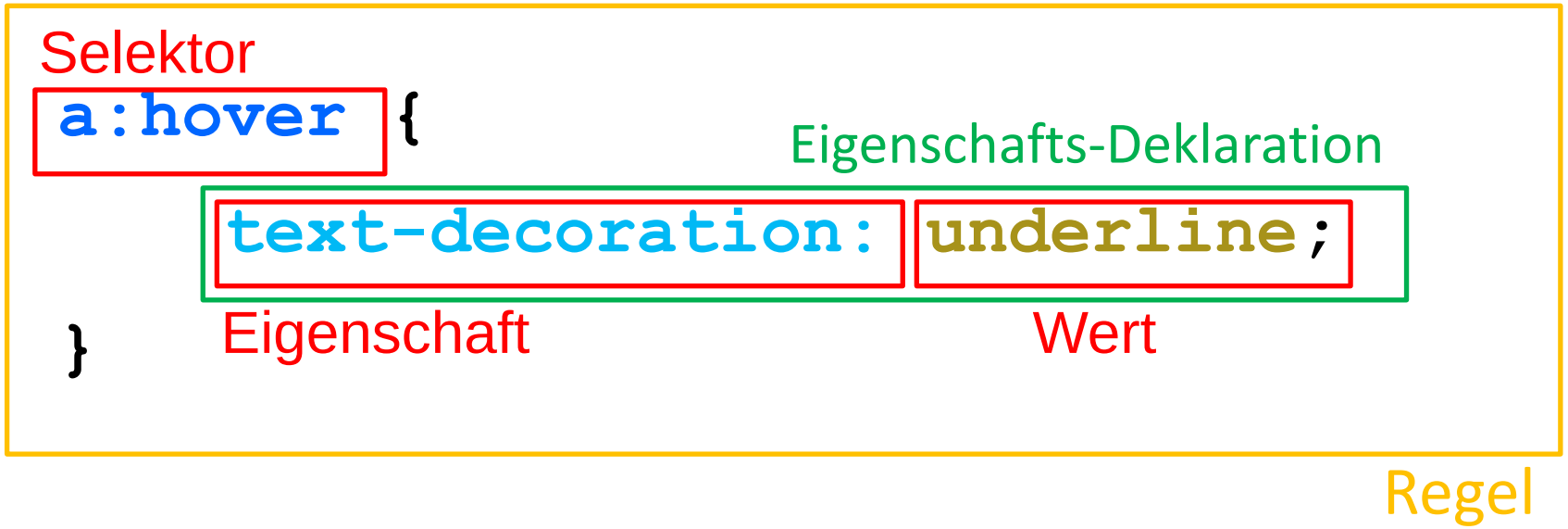
- Nicht nur **umfangreiches** CSS
- sondern auch:
  - Projekte mit mehreren Leuten
  - CSS mit häufigen Änderungen
  - CSS in mehreren Varianten

Was sind die Ursachen?

- CSS ...
  - ... wird häufig stiefmütterlich behandelt
  - ... ist eine Arbeit die "nebenher" erledigt wird
  - ... wird häufig unterschätzt
  - ... wird genau so komplex werden wie jeder andere Teil des Sourcecode

kurze Begriffsklärung

Aufbau von Regeln:



Gestaltungsfaktoren CSS

- Dateiaufteilung
- Sortierung / Aufbau der Regeln
- Aufbau der Selektoren
- Aufbau der Deklarationen

Dateistrukturen für CSS

- Basis Stylesheet (Reset, Einzelelement CSS)
- Layout (ggf. Unterscheidung Desktop/Mobile)
- Module (klar separierbare oder wiederkehrende Elemente)
- Status (Gestaltung der Module in versch. Zusammenhängen)
- Theming ("kundenspezifische" Einstellungen)

Dateistrukturen für CSS

- bestimmte Dateien nur laden, wenn sie benötigt werden
  - bei Mobile Stylesheets
  - bei sehr komplexem CSS für einzelne Module

Organisation in den Dateien

- Regeln thematisch sortieren
  - mit Kommentaren
- Andere Sortierungen unhandlich bis unsinnig.
- Ständige Frage: *Gehört die Regel noch in diese Datei?*

Aufbau der Regeln

- *Regeln sollten ...*
  - ... nicht redundant sein
  - ...
- *Sele*
  - ...

```
div#mySpecialContainer > div#mySpecialSubContainer {
        background-color: #999;
        color: #222;
        border: 1px solid #922;
}
```

  Elemente beziehen
  - ... möglichst wenig IDs enthalten
  - ... nicht überspezifisch sein

Aufbau der Regeln

- *Klassen sollten ...*
  - ... semantisch be
- *Eigenschafts-Deklarationen sollten ...*
  - ... alphabetisch s
  - ... Kurzschreibweisen benutzen
  - ... auf `!important` verzichten
  - ... nicht in style-Attribute geschrieben werden

Ein Fehler ist aufgetreten!

Ein Fehler ist aufgetreten!

Objektorientiertes CSS

- Möglichkeit der CSS Strukturierung
- überträgt OO-Prinzipien auf HTML/CSS
- wiederverwendbare Komponenten
- konzeptioniert durch Nicole Sullivan

Objektorientiertes CSS

- Was ist ein "Objekt" in OOCSS?
- Verbund aus:
  - HTML
  - CSS
  - zugehörige Elemente (bspw. Bilder)

Objektorientiertes CSS - Grundprinzipien

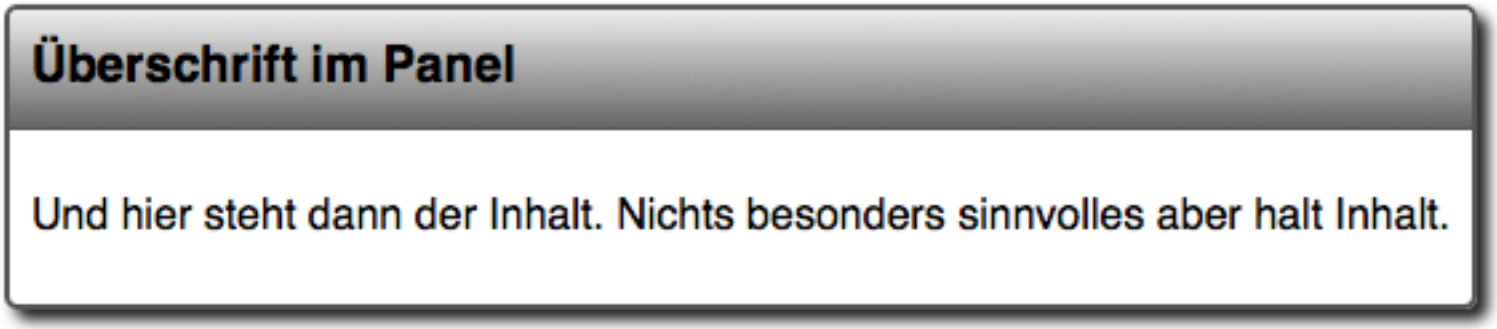
- Trennen von Struktur und Gestaltung
  - wiederholte Styles herausziehen
  - Klassen zur Benennung von Objekten und Komponenten
- Container von Inhalt trennen
  - keine "ortsabhängigen" Styles
  - stattdessen eigene Klasse

Objektorientiertes CSS

- Inhalt
  - HTML Elemente + direktes CSS
- Container
  - komplexere Strukturen, die Inhalte gruppieren
- Objekte
  - Container + Layout + Skin CSS

Beispiel-Objekt

```
<div class="panel panel-warning">
    <d
    </
</div>
```



OOCSS – Pro und Contra

Pro:

- Wiederverwendbare Strukturen
- erheblich kleineres CSS
- bessere Verteilbarkeit im Team

Contra:

- leicht größeres HTML (Classitis)
- bricht mit Prinzip der Kaskadierung

Über CSS hinaus

- Dokumentation, auch für das Stylesheet!
  - auch wenn man alleine arbeitet
  - aufwändigere Erklärungen nicht im CSS!
- Festlegung von Coding Guidelines
  - Was gehört wo hin?
  - Code Formatierung

Immer wieder neu...

- *Muss ich bei jedem Projekt immer wieder von vorne anfangen?*
  - vorgefertigte Stylesheets
  - fertiger Werkzeugkasten

CSS Frameworks

- Bieten vorgefertigte Lösungen
  - Erstellung von Grids
  - Formatierung von typischen Elementen
    - Buttons
    - Tab-Leisten
    - ...
  - automatische Anpassung für Mobile Browser

Frameworks - Vorteile

- sind sehr robuste Stylesheets
- kümmern sich idR. schon um Cross-Browser Probleme
- sehr schnell anwendbar
  - gemeinsame Grundlage für alle Beteiligten
- nehmen einem "Basisarbeit" ab

Frameworks - Nachteile

- häufig ein eigenes Look & Feel
- setzen bestimmten Stil voraus
- sind oft "mehr als man braucht"
  - da häufig Multi-Purpose
- Kollisionsgefahr mit eigenem CSS und JavaScript (sofern schon etwas besteht)
- jeder muss das Framework kennen

Frameworks - Einsatz

- Ich benutze Frameworks wenn ...
  - ich schnell ein Ergebnis brauche
  - keine besonderen Anforderungen an das Aussehen gestellt werden
  - ich Arbeit abgenommen bekommen möchte
  - ich noch am Anfang des Projektes stehe

Welche Frameworks gibt es?

Foundation
Start here, build everywhere.

Kurzes Beispiel

Anhand von Bootstrap

"Aber CSS ist doof ..."

- CSS wirkt manchmal etwas unvollständig
- hat eine nicht ganz intuitive Syntax
  - bspw. Wiederholungen in Selektoren
- bei bestimmten Änderungen sehr schlecht handhabbar

"Aber CSS ist doof ..."

- Was wenn man eine Skriptsprache hätte, die all das kann, was man möchte?
- Was wenn man über diese ein CSS generieren könnte?
- CSS Preprocessor
  - {Less}
  - Sass (Syntactically Awesome Stylesheets)
  - Myth (CSS the way it was imagined.)

{Less} – Ein Preprocessor

- in JavaScript geschrieben
  - kann als Script in HTML eingebunden werden
  - oder über Node.js ausgeführt werden
- {Less} Skripte sind eine Obermenge von CSS
  - alles aus CSS ist in {Less} gültig

Was bietet {Less}?

- Variablen
- Nesting
  - Verschachtelung von Klassen
- Vererbung und Mixins
- Funktionen und Operationen

Kurzes Beispiel

Wir machen was mit {Less}.

Preprocessing - Vorteile

- forciert DRY Pattern im CSS
- teilweise intuitiver
- leichtere Änderbarkeit
- führt zu besser organisiertem CSS
- **zumindest im Quellcode**

Preprocessing - Nachteile

- Eine weitere Schicht an Komplexität
  - fehleranfälliger
- erschwert Debugging
- generiertes CSS kann "explodieren"
- "von Hand" kann besser sein

Wann benutze ich das?

- für Theming wie geschaffen
- um umfangreiche Stylesheets sauber strukturieren zu können
- auch nachträglich einführbar
- für das Plus an Komfort

Gutes CSS sicherstellen

- Habe ich automatisch ein gutes CSS, wenn ich das alles mache?
  - Vielleicht Ja. Wahrscheinlich Nein!
- Noch wesentlich mehr Fehlerquellen, auf die man achten sollte.
  - gerade in großen Projekten

Was sind das für Fehler?

- inhaltliche Fehler
- Eine kleine Auswahl:
  - fehlende Fallback-Werte bei neueren CSS Features
  - CSS Hacks
  - Box-Sizing und daraus resultierende Probleme
  - übermäßiger Gebrauch von Web-Fonts

Kann ich das abfangen?

- Überprüfung mit Developer Tools
- Code-Reviews
- Rege
- ist konfigurierbar
  - nicht jede Regel macht in jedem Fall Sinn
- zumindest einmal vor dem Live-Gang das fertige Stylesheet überprüfen
- besser: in die Build-Umgebung einbinden

Die Buildumgebung

- Kann ich das auch automatisch machen (lassen)?
  - {Less}
  - CSS Lint
- Als Maven-Plugin verfügbar
- Sehr einfach über Grunt
  - "normale" Umgebung der Tools ist idR. Node.js / Grunt
- für alle Vorteile: Grunt aus Maven ansteuern

Fazit

- **Tagelanges Probieren erspart mir Stunden an Planung!**
- Vieles kann auch bei bestehendem CSS nachträglich eingeführt werden:
  - Modularisierung
  - Einsatz eines Preprocessors
  - Qualitätssicherung

Fazit

- CSS wird immer Änderungen unterworfen sein.
  - gerne auch zu unmöglichen Zeitpunkten
- Ich muss den Änderungen mit Struktur begegnen.
- Woanders plane ich ausgiebig. **Warum nicht auch bei CSS?**

Fragen?

Vielen Dank

Sebastian Reiners

open knowledge **GmbH**

www.openknowledge.de

facebook.com/openknowledge

twitter.com/_openknowledge