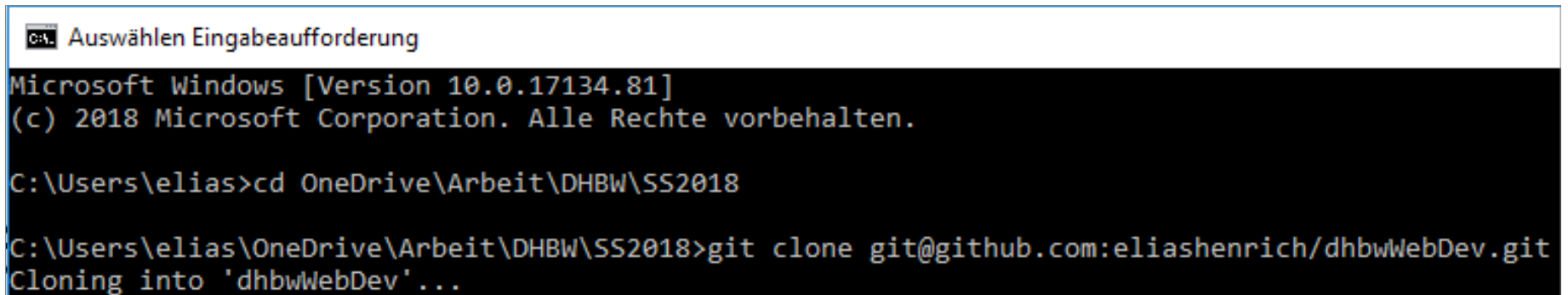

GitHub-Repository

Die Unterlagen werden nach jeder Vorlesung im folgenden Repository veröffentlicht:

<https://github.com/eliashenrich/dhbwWebDev>

GitHub installiert? Klonen Sie das Repo:

`git clone https://github.com/eliash...`



```
Microsoft Windows [Version 10.0.17134.81]
(c) 2018 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\elias>cd OneDrive\Arbeit\DHBW\SS2018

C:\Users\elias\OneDrive\Arbeit\DHBW\SS2018>git clone git@github.com:eliashenrich/dhbwWebDev.git
Cloning into 'dhbwWebDev'...
```

Überblick Heute

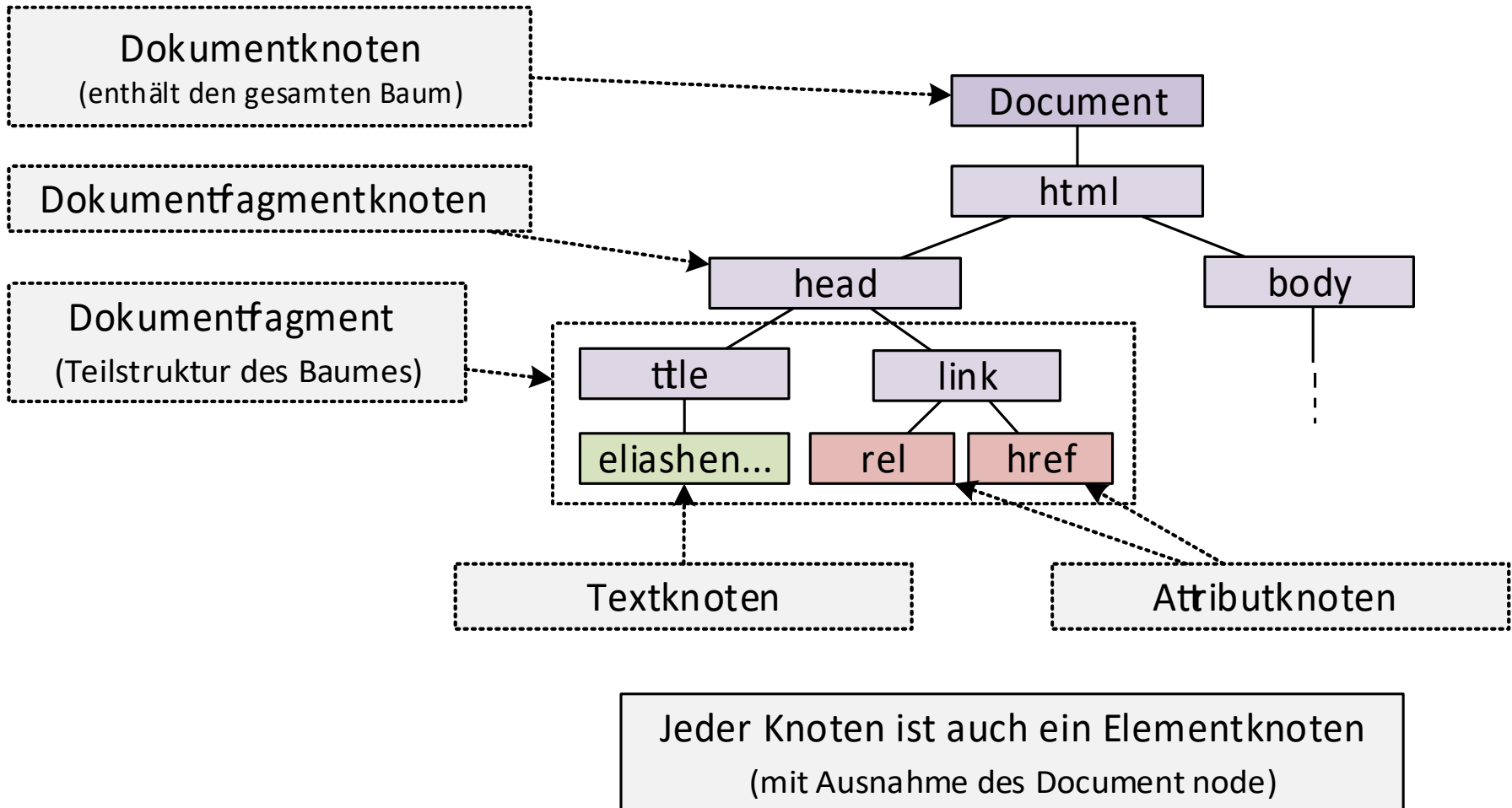
- Das DOM
- Wiederholung HTML-Grundlagen
- Userdaten erfassen mittels Formulare
- Vorstellung Semesterprojekt
- Einführung in CSS
- Inline-Styling
- Kleine Farblehre
- Selektoren, Kombinatoren
- Interne CSS-Styles

Document Object Model

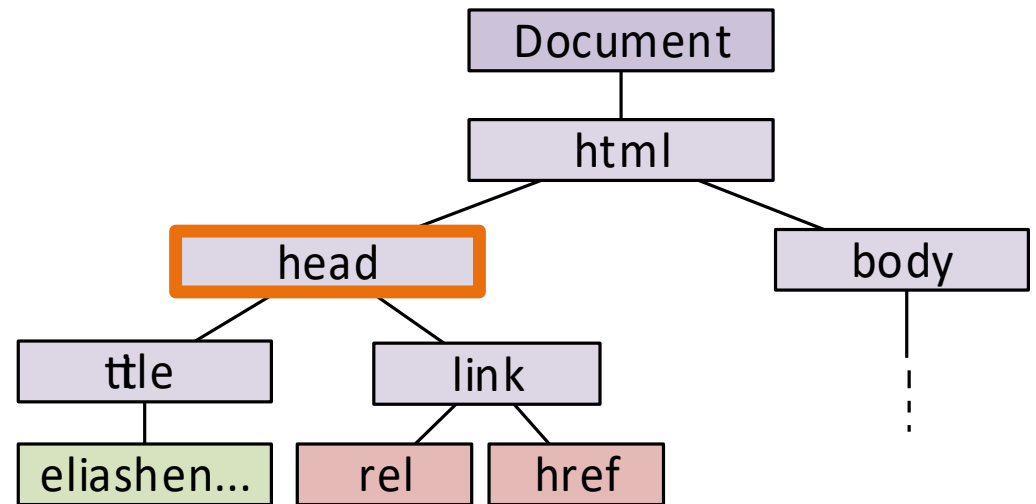
„...is the object presentation of the HTML document and the interface of HTML elements to the outside world like JavaScript.“

- Interne Darstellung eines HTML-Dokuments im Browser (aber auch aller anderen XML-Daten)
- Baumstruktur bestehend aus Knoten (*Node*)
- unterschiedliche Knotenarten

Knotenarten



DOM-Beziehungen



head ist...

- ... child von *html*
- genauer: ...*firstChild* von *html*
- ... parent von *title* und *link*
- ... sibling von *body*

→ mit diesen Beziehungen kann jeder Knoten von jedem Knoten erreicht werden.

DOM: Übung

Nutzen Sie die Developer Tools, um folgende Aktionen auf **<http://eliashenrich.de>** durchzuführen:

- Identifizieren für die Grafik von *Chief O'Brien*:
 - Parent, child und sibling node(s)
 - Pfad vom Bild bis zum Hyperlink
- Ändern Sie das *src*-Attribut von *img* zu einer Grafik von lorempixel.com
- Fügen Sie der Grafik einen Hyperlink mit Ziel lorempixel.com hinzu

HTML-Wiederholung

„HTML [ist] die Sprache, die die Struktur und Semantik der Inhalte eines Web-Dokuments beschreibt.“

- HTML-Dokumente bestehen aus sog. *Tags*:
 - *Starttag* mit *Attributliste*
 - Inhalt
 - *Endtag*
 - Ausnahme: *Empty Tags* haben keine Kindelemente

Das *Form*-Tag

Formulare erlauben Benutzern Daten einzugeben und an den Server zu übertragen.


```
<form action="http://..." method="POST">  
  <input type="submit" value="Absenden">  
</form>
```

Reminder:
<input> gehört zu
den Empty Tags!

<form> - Attribute

```
<form Attributsliste>
```

```
</form>
```

Attribut	Bedeutung
action	URL, an die Formular-Daten übertragen werden
method	HTTP-Methode, um Daten zu übertragen <ul style="list-style-type: none">• GET (Standard)• POST
name	Name des Formulars
autocomplete 	Browser vervollständigt Daten automatisch

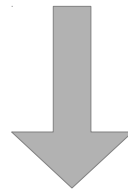


`<input type="text">`

= einzeliges Textfeld

wird als Fallback bei Inputs mit unbekanntem Type angezeigt.

Textfeld: `<input type="text" name="txtFeld" value="Vorbelegter Wert">`



Textfeld:



`<input type="checkbox">`

= Mehrfachauswahl

gleicher Name verknüpft die Felder

Wichtige Elemente auswählen: `
`

`<input type="checkbox" name="chkBox">` Element A `
`

`<input type="checkbox" name="chkBox">` Element B `
`

`<input type="checkbox" name="chkBox">` Element C `
`



Wählen Sie benötigte Elemente aus:

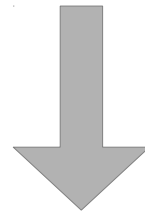
- ☒ Element A
- ☐ Element B
- ☒ Element C



`<input type="submit">`

= Aufruf und Übergabe der Daten an die
in *action* angegebene URL

```
<input type="submit" value="Absenden">
```



Absenden



Weitere Input-Types

`type="radio"`

☐ Antwort 1

☐ Antwort 2

`type="password"`

.....

`type="file"`

Datei auswählen test.html

`type="reset"`

Zurücksetzen

`type="hidden"`



Neue Input-Types

Achtung: Darstellung ist vom Browser abhängig!

type="email"

Mai 2017 ▾

Mo	Di	Mi	Do	Fr	Sa	So
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

type="date"

Farbe

Grundfarben:

Benutzerdefinierte Farben:

Farben definieren >>

OK Abbrechen

FarbeBasis

Farben hinzufügen

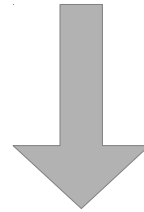
type="color"

type="number"

Mehr: https://www.w3schools.com/html/html5_new_elements.asp

Vollständiges Formular

```
<form name="formRegister" action="register.php"
      method="POST" autocomplete="FALSE"
>
  Name:    <input type="text" name="name" placeholder="Vor-/Nachname"><br>
  E-Mail:  <input type="email" name="email"><br>
  Passwort: <input type="password" name="password"><br>
  <input type="submit" value="Registrieren">
</form>
```



Name:	<input type="text" value="Vor- und Nachname"/>
E-Mail:	<input type="email"/>
Passwort:	<input type="password"/>
<input type="submit" value="Registrieren"/>	


Form-Elemente

input mit verschiedenen *types*

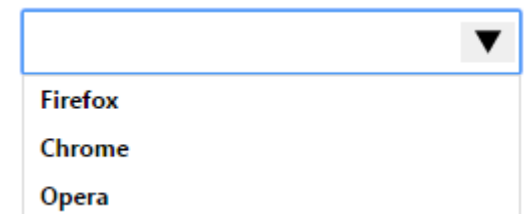
```
<select name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```



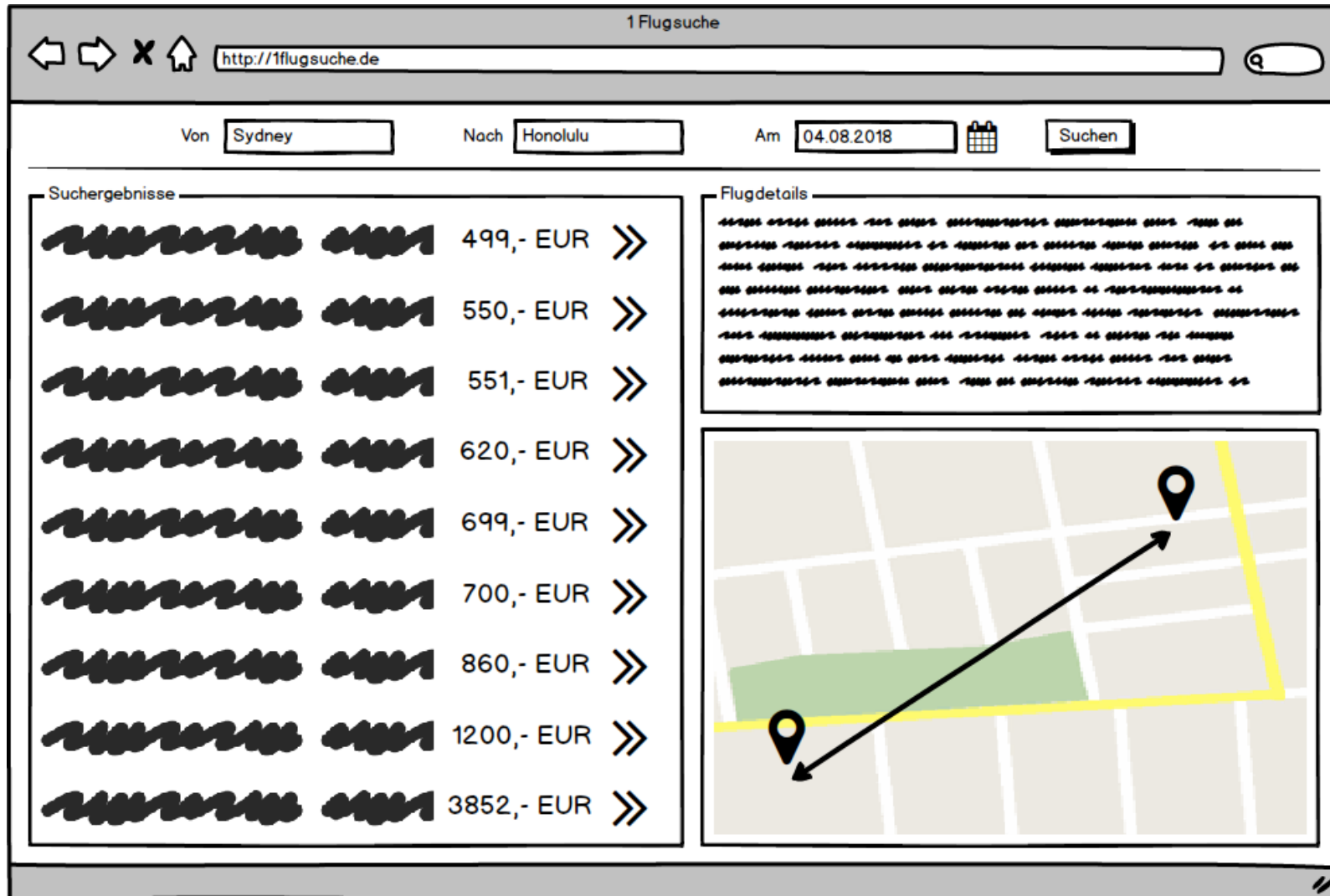
```
<textarea name="message" rows="3" cols="30">Inhalte...</textarea>
```



```
<input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
  </datalist>
```



Semesterprojekt



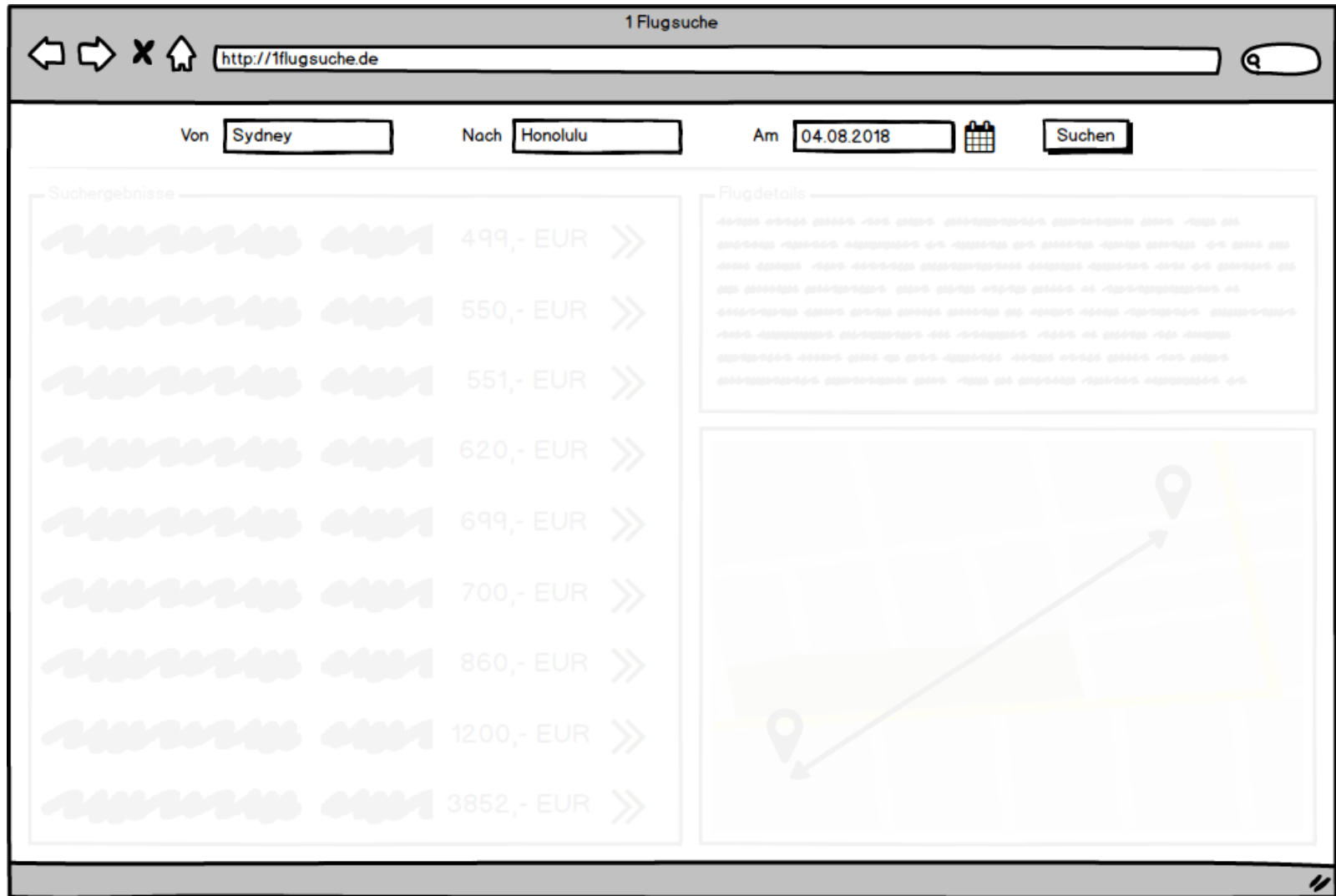
(Wireframe erstellt mit Balsamiq)

Semesterprojekt

Verfügbare Daten:

- Flughäfen im Pazifikraum
 - Stadt, Name, Land, Koordinaten
- An- und abgehende Flugverbindungen
 - Airline, Flugzeugtyp
- Wochentage und Uhrzeiten
- Routing von A nach B (Dijkstra-Algorithmus)
- geschätzte Flugkosten (nur One-Way)

Semesterprojekt



Übung: Formulare

<input type="text" value="SYD"/>	<input type="text" value="LAX"/>	<input type="button" value="Suchen"/>
----------------------------------	----------------------------------	---------------------------------------

- Erstellen Sie ein Formular mit zwei Textfeldern „Von“ und „Nach“ sowie einem Submit-Button
- Ziel des Formulars:

<http://flights.eliaschenrich.de/form.php>

- Verwenden Sie die GET-Methode zur Übertragung
- Beachten Sie die Rückgabewerte!
- Ziel: Die Liste aller Wege vom Flughafen **SYD** nach **LAX** im JSON-Format

Cascading Style Sheets



„Gestufte
Druckformatvorlagen“

= Beschreibung der
Darstellung von HTML-
bzw. allgemein XML-Daten.

HTML: Inhalt

CSS: Optik

→ **mit DOM die Grundlagen aller Web-Apps**



CSS: Historie

1991

HTML: Tabellen zur Platzierung von Objekten

Mitte '90er

HTML 3.2: Style-Attribut zur
einfachen Text- und Objektformatierung
→ Optik wird wichtiger!

Aber: Browser-Hersteller führen proprietäre Tags ein

Ende 1996


W3C veröffentlicht CSS1-Standard
Text- und Objektformat., float (wie Text + Grafiken verbinden)

Ende '90er

CSS2 führt Positionierung ein

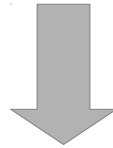
Wieso gestuft/cascading?

Priorisierung der Darstellungsvorgaben:

- Standarddarstellung des Browsers
 - Externe CSS-Dateien
 - Interne CSS-Anweisungen
 - Inline Styles
- 

Inline Styles

```
<p style="background-color: yellow;text-align:center;width: 150px;">  
    Box ist 150 Pixel breit und bricht ggf. Zeilen autom. um!  
</p>
```



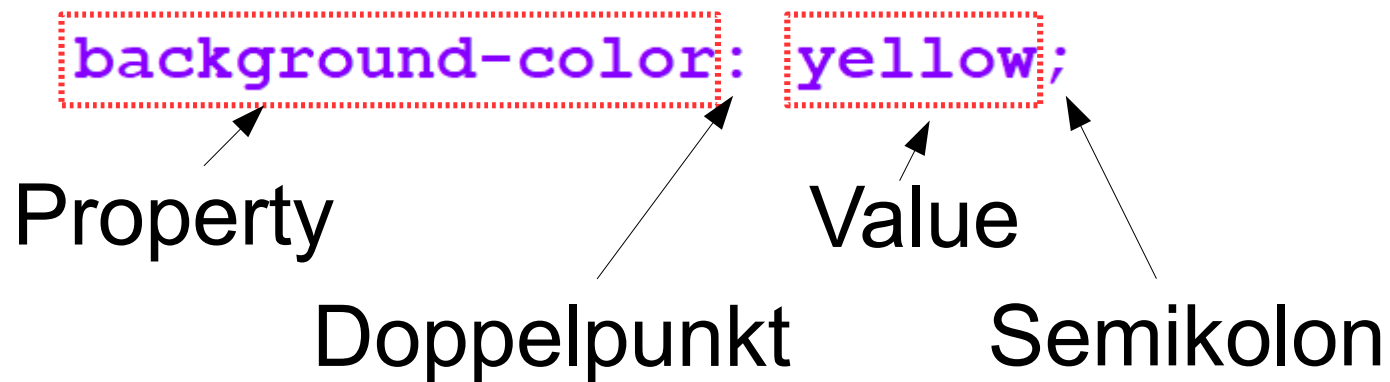
Box ist 150 Pixel breit
und bricht ggf. Zeilen
autom. um!

CSS-Anweisung(en) als Attribut
des HTML-Tags

→ überschreibt alle anderen Anweisungen

Aufbau CSS-Anweisung

```
<p style="background-color: yellow;text-align:center;width: 150px;">  
  Box ist 150 Pixel breit und bricht ggf. Zeilen autom. um!  
</p>
```



White-Spaces werden ignoriert,
aber: Code-Formatierung wichtig!

Inline Styles

```
<p style="background-color: yellow;text-align:center;width: 150px;">  
    Box ist 150 Pixel breit und bricht ggf. Zeilen autom. um!  
</p>
```

```
<p style="background-color: yellow;text-align:center;width: 100px;">  
    Diese Box hat fast die gleichen Property-Values  
</p>
```

Der Wartungsaufwand steigt während sich die Flexibilität verringert. Inline-Styles sind an ein Dokument gebunden und können nicht an zentraler Stelle bearbeitet werden.

CSS: Größeneinheiten

Dimension	Bedeutung	Bezug
px	Pixel	Absolutwert
mm	Millimeter	Absolutwert
pt	Punkt	Absolutwert, vgl. Schriftsatzgröße
%	Prozent	Relativ zur Parent-Eigenschaft
em	Element	Relativ zur Schriftgröße des Elements (2em = 2 fache Höhe der Schrift)
rem	Root Element	Relativ zur Schriftgröße des body-Tags
vw / vh	Viewport Width / Height	Relativ zu 1% der Breite / Höhe des Viewports (Browser-Fenster)

Übung: Inline Styles

VON:	SYD	NACH:	LAX	Suchen
-------------	-----	--------------	-----	---------------

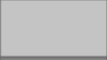
Erweitern Sie Ihr Flug-Formular um folgendes Styling:

- Inhaltsbereich hat Schriftart „Arial Black“ mit Schriftgröße 12px
- Textfelder haben Breite = 50 Pixel und zentrierten Inhalt
- Formular hat eine Hintergrundfarbe (*lightgray*)
- 2px breite Linie unterhalb des Formulars (*border-bottom*) mit Farbe „DarkCyan“
- Formular erstreckt sich über die gesamte Bildschirmbreite
- Der Submit-Button ist **fett** geschrieben (*font-weight*)

Theorie: Farben

HTML 4-Standard (1999)


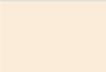



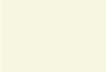
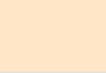


- 16 Farben, die von jedem Browser unterstützt werden sollen
- wurden von jedem VGA-Monitor dargestellt

Color Name	Hexadecimal Value	Sample
Black	#000000	
Silver	#C0C0C0	
Gray	#808080	
White	#FFFFFF	
Maroon	#800000	
Red	#FF0000	
Purple	#800080	
Fuchsia	#FF00FF	
Green	#008000	
Lime	#00FF00	
Olive	#808000	
Yellow	#FFFF00	
Navy	#000080	
Blue	#0000FF	
Teal	#008080	
Aqua	#00FFFF	

Theorie: Farben

CSS-Farbnamen

- 140 Farben
- mit (englischer) Bezeichnung
- case-insensitive

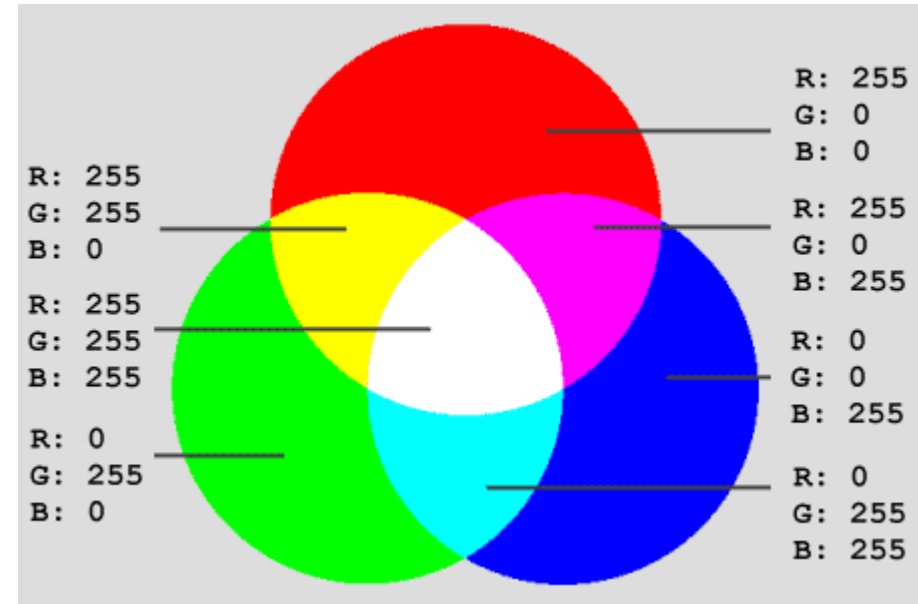
Color Name	HEX	Color
<u>AliceBlue</u>	<u>#F0F8FF</u>	
<u>AntiqueWhite</u>	<u>#FAEBD7</u>	
<u>Aqua</u>	<u>#00FFFF</u>	
<u>Aquamarine</u>	<u>#7FFFD4</u>	
<u>Azure</u>	<u>#F0FFFF</u>	
<u>Beige</u>	<u>#F5F5DC</u>	
<u>Bisque</u>	<u>#FFE4C4</u>	
<u>Black</u>	<u>#000000</u>	
<u>BlanchedAlmond</u>	<u>#FFEBCD</u>	

`background-color: AliceBlue;`

Theorie: Farben

RGB- / RGBa-Value

- Byte-Triplet, bestehend aus
 - 1.B: Rot-Anteil (0-255)
 - 2.B: Grün-Anteil (0-255)
 - 3.B: Blau-Anteil (0-255)
- Ggf. Alpha-Wert (0.00 – 1.00)
0.00: Vollständig transparent



`background-color: rgb(240,248,255);`

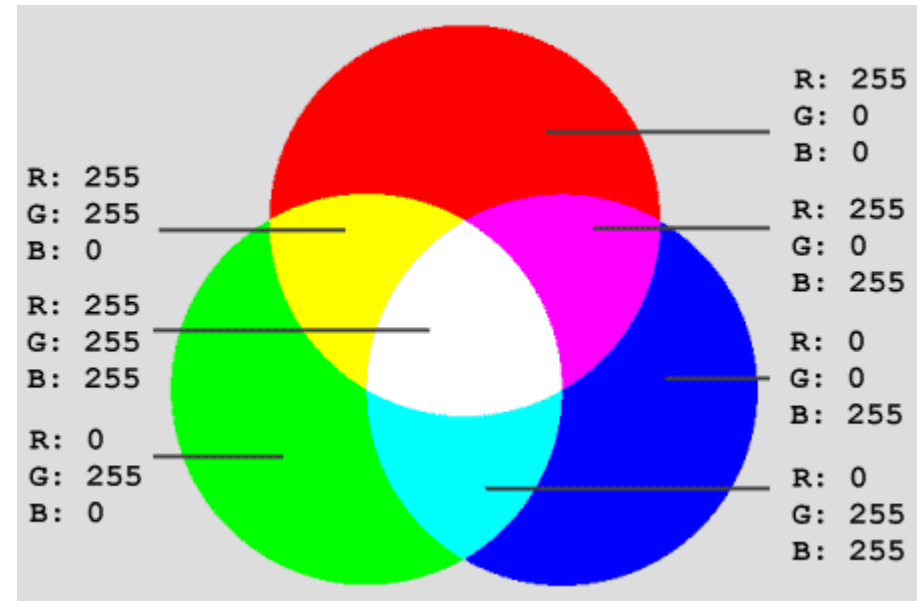
`background-color: rgba(240,248,255,0.5);`

Quelle: <http://www.informatikzentrale.de/rgb-farbmodell.html>

Theorie: Farben

Hexadezimal-Wert

- Beginnt mit #-Symbol
- Byte-Triplet:
 - 1. B: Rot-Anteil (00-FF)
 - 2. B: Grün-Anteil (00-FF)
 - 3. B: Blau-Anteil (00-FF)
- case-insensitive
- populärste Form (?)



`background-color: #F0F8FF;`

Theorie: Farben

- $\text{FFFFFF}_{16} = 16^6 = \text{ca. } 16,7 \text{ Mio Kombinationen}$
 - Farbdarstellung immer vom Display abhängig
 - Menschliches Auge hat nur begrenzte Farbwahrnehmung (insb. Graustufen)
 - Für andere Medien wenig geeignet, insb. beim Druck (vgl. CMYK-Farbraum)
- Unterschiedliche Farbwiedergabe i.d.R. unkritisch, je nach Anwendungsfall aber zu berücksichtigen.

CSS: einfache Selektoren

Selektor	Beispiel	entspricht
Universalselektor	*	alle Elemente
Typselektor	p / h1 / strong	HTML-Tagname
Klassenselektor	.klassenName	Alle Elemente mit Klassen <i>klassenName</i>
ID-Selektor	#elementId	Element mit ID <i>elementId</i>
Attributsselektor	[href]	Präsenz des Attributs <i>href</i>
	[type="text"]	Attribut <i>type</i> mit Wert <i>text</i>
	[href^="http://"]	Wert d. Attributs <i>href</i> beginnt mit <i>http://</i>
	[href\$=".pdf"]	... endet mit <i>.pdf</i>
	[href*=".de"]	... beinhaltet <i>.de</i>

Selektoren verschachteln

```
<p class="gelbeBox">  
  <input type="reset" value="Reset">  
  <input type="submit" value="Absenden">  
</p>
```

```
p {  
  background-color: #ffff00;  
  text-align: right;  
}
```

→ alle HTML-Tags mit Name *p*

Selektoren verschachteln

```
<p class="gelbeBox">  
  <input type="reset" value="Reset">  
  <input type="submit" value="Absenden">  
</p>
```

. ist CSS-Klassenselektor

```
.gelbeBox {  
  background-color: #ffff00;  
  text-align: right;  
}
```

→ alle HTML-Elemente, deren Attribut *class*
u.a. die Klasse *gelbeBox* enthält

Selektoren verschachteln

```
<p class="gelbeBox">  
  <input type="reset" value="Reset">  
  <input type="submit" value="Absenden">  
</p>
```

```
p.gelbeBox {  
  background-color: #ffff00;  
  text-align: right;  
}
```

→ alle HTML-Elemente mit Tagname *p*
und Klasse *gelbeBox*

Selektoren verschachteln

```
<p class="gelbeBox">  
  <input type="reset" value="Reset">  
  <input type="submit" value="Absenden">  
</p>
```

```
p.gelbeBox {  
  background-color: #ffff00;  
  text-align: right;  
}
```

→ alle HTML-Elemente mit Tagname *p*
und Klasse *gelbeBox*

Kombinatoren

Beispiel	Bezeichnung	Angesprochene Elemente
$p > \text{input}$	Child selector	Alle <i>input</i> -Elemente, die direkt zu p gehören
$p \text{ input}$ (Leerzeichen)	Descendant selector	Alle <i>input</i> -Elemente, deren Vorfahre p ist
$p + h1$	Neighbor selector	Alle $h1$, die direkter Nachbar von p sind
$p \sim h1$	Sibling selector	Alle $h1$, die einen gemeinsamen Parent p haben

Selektoren kombinieren

```
<p class="gelbeBox">  
  <input type="reset" value="Reset">  
  <input type="submit" value="Absenden">  
</p>
```

```
p.gelbeBox > input[type="submit"] {  
  color:#009900;  
}
```

→ alle HTML-Elemente mit Tagname *input*, deren Attribut *type* den Wert *submit* hat und, die Child eines Elements mit Tagname *p* und Klasse *gelbeBox* sind

Interne CSS-Styles

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titelseite</title>
    <style>
      body {
        font-family: "Arial Black", Gadget, sans-serif;
        font-size: 12px;
        background-color: lightBlue;
      }
      h1 {
        font-size: 2rem;
      }
    </style>
  </head>
```

Zentrale Style-Definition im Dokumentenkopf – dadurch:

- Wiederverwendbarkeit
- schnellere Fehlerkorrektur
- keine Wiederverwendung bei mehreren HTML-Dateien

Übung: Interne Styledefinition

Überarbeiten Sie das per inline-Style formatierte Formular, damit das gesamte Styling an zentraler Stelle geschieht.

Achten Sie auf eine sinnvolle Wiederverwendung gleicher Eigenschaften.

Verwendung mehrerer Klassen:

```
<form class="textCentered" [...]>  
  <input class="fixedWidth textCentered" [...]>  
  <input class="fixedWidth textCentered" [...]>  
</form>
```