

# **Napredni razvoj programske podpore za web**

**- predavanja -  
2021./2022.**

---

**CSS  
(nastavak na W1)**

# Creative Commons



- slobodno smijete:
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **prerađivati** djelo
- pod sljedećim uvjetima:
  - **imenovanje:** morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno:** ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima:** ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, prerađu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

*U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.*

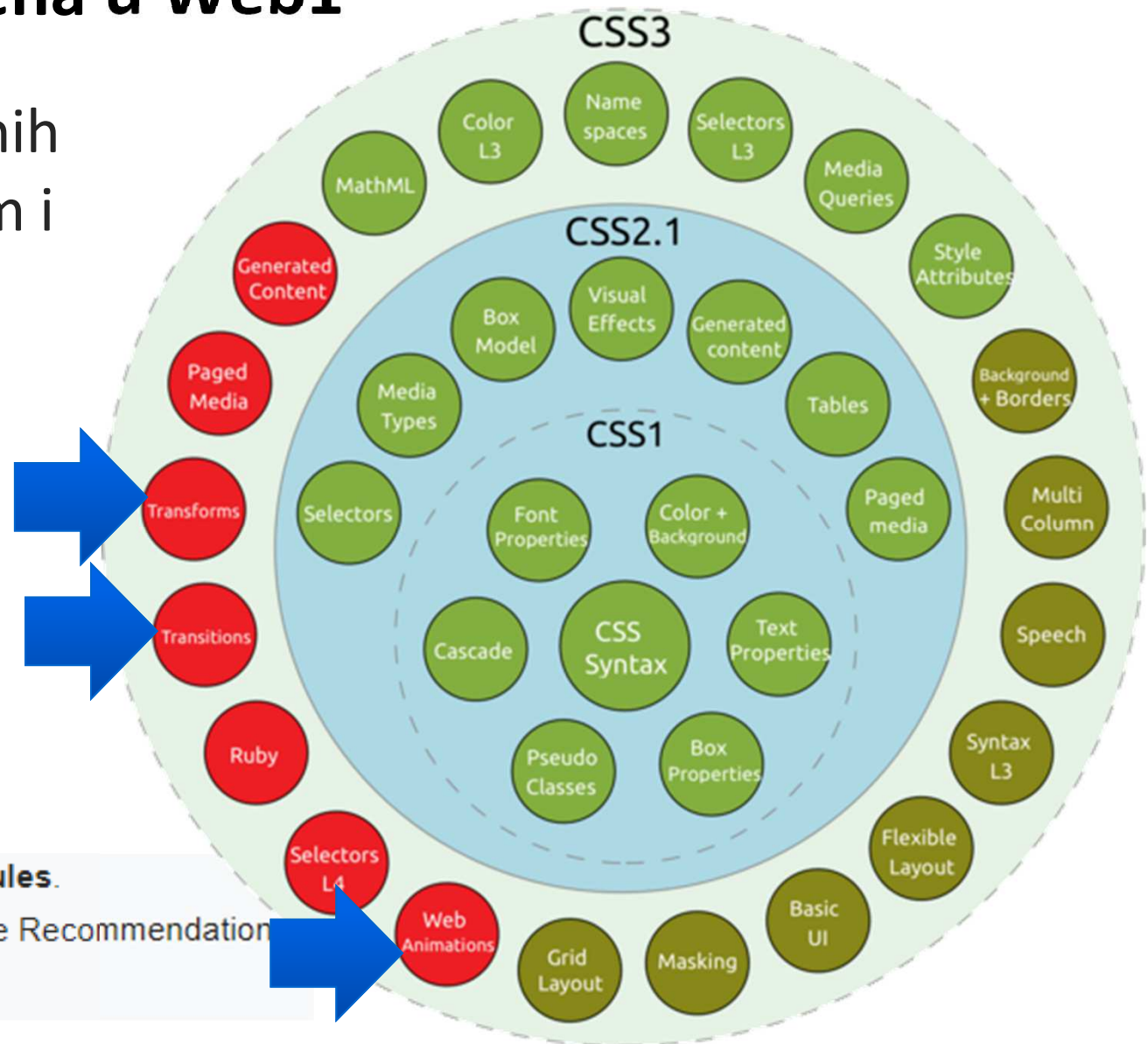
*Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.*

*Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.*

*Tekst licence preuzet je s <http://creativecommons.org/>*

# Većina obrađena u Web1

- Još par odabranih tema: transform i animacije



Taxonomy and status of CSS3 **modules**.

- Recommendation
- Candidate Recommendation
- Last Call
- Working Draft.

[#Cascading Style Sheets](#)

# Sadržaj

- CSS (nastavak):
  - Transform
  - Animations
- CSS jezici
  - SASS/SCSS
- *CSS scope*
- *CSS frameworks*

# ■ transform

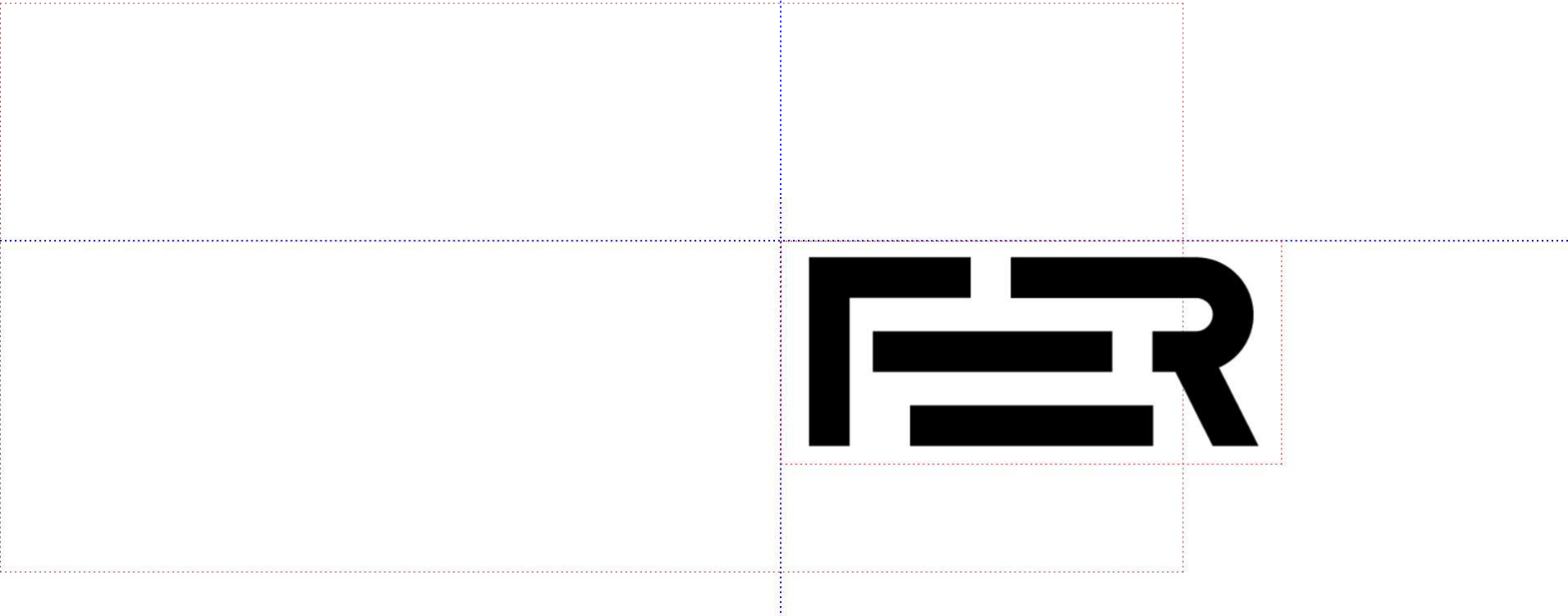
- Hardversko ubrzanje
- Elementi ostaju u toku (*flow*)
- Glavne opcije:
  - **Pomak** (možemo i s top, left, margin ali ovo je bolje)
    - transform: **translate**(120px, 50%);
  - **Skaliranje**
    - transform: **scale**(2, 0.5);
  - **Rotacija**, dodatna opcija: transform-origin (npr. center, top left, itd.)
    - transform: **rotate**(0.5turn);
  - **Iskrivljavanje**
    - transform: **skew**(30deg, 20deg);
- Poredak je bitan (**zdesna na lijevo**)
  - <https://stackoverflow.com/questions/27635272/css3-transform-order-matters-rightmost-operation-first>

# Transform demo

Transform demo

127.0.0.1:5500/index.html

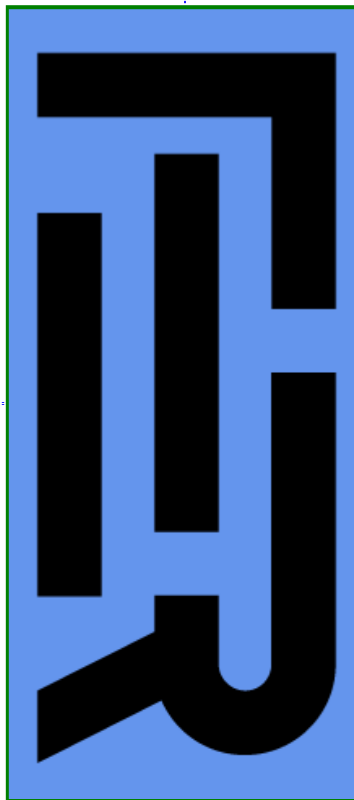
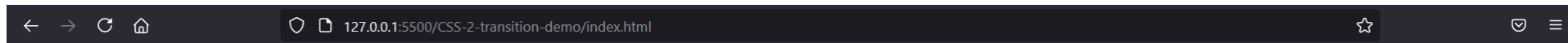
translate(120px, 50%):  scale(2, 0.5):  rotate(0.5turn):  skew(30deg, 20deg):



## ■ Transition (and animation)

- Kada mijenjamo neko CSS svojstvo to se događa momentalno (npr. boja pozadine iz bijele u plavu)
- `transition: <property> <duration> <timing-function> <delay>;`
- Transition nam omogućuje da kontroliramo, tj. animiramo promjenu svojstva, možemo kontrolirati:
  1. Koje svojstvo se mijenja, te za svako od svojstava definirati:
    - Nisu sva svojstva upravljiva: [https://developer.mozilla.org/en-US/docs/Web/CSS/CSS\\_animated\\_properties](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties)
  2. Koliko vremena traje promjena
  3. Acceleration/easing function - funkciju promjene svojstava u vremenu (default - linearno)
    - Easing function cheat sheet: <https://easings.net/>
  4. Eventualni poček

# Transition demo





# Animations

- Ako trebamo još veću kontrolu nad promjenama možemo koristiti animacije
- Prvo, definirajmo keyframes (a zatim ih referenciramo), npr.:

```
@keyframes logo-path {  
  from {  
    background-color: red;  
  }  
  to {  
    transform: translate(-50%, -50%) scale(1.3, 1.3) rotate(90deg);  
    border: 3px solid green;  
    background-color: cornflowerblue;  
  }  
}
```

Proizvoljno ime

Istog trena će prvo  
promijeniti boju  
pozadine u crvenu

Moguće je koristiti i postotke.  
from je isto kao 0%  
to 100%

# Animations

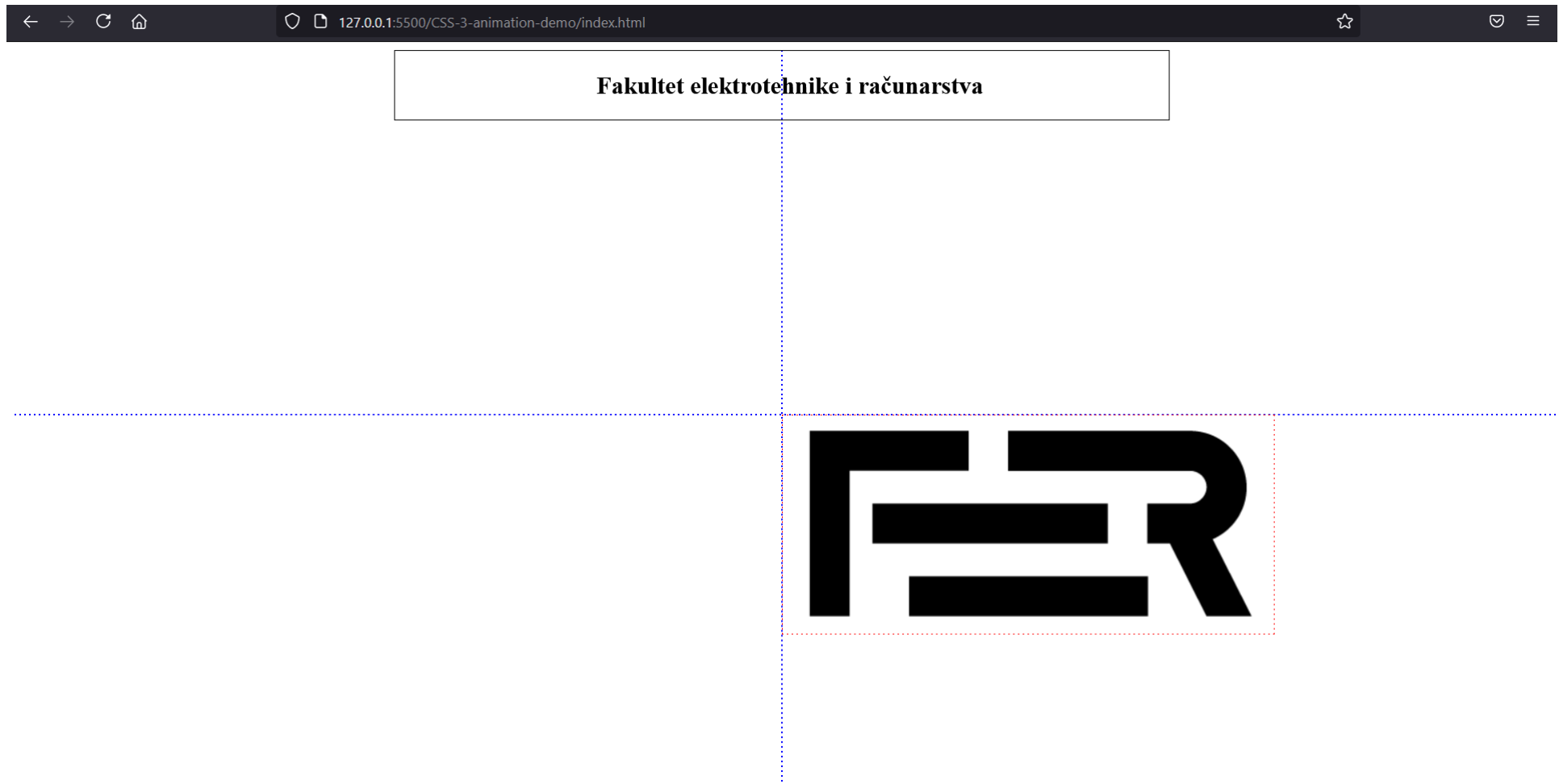
- Zatim referenciramo keyframes i podesimo parametre:
  - animation-name (ime keyframes)
  - animation-duration (trajanje)
  - animation-timing-function (acceleration/easing curve, isto)
  - animation-delay (početak)
  - animation-iteration-count (broj ponavljanja, moguće je infinite)
  - animation-direction (definira ponašanje nakon ciklusa)
  - animation-fill-mode (koje vrijednosti se primjenjuju prije i na kraju animacije - npr. želimo li da ostane u konačnom stanju?)
- Npr. animirani tekst iz sljedećeg primjer (animations demo)

```
#banner h2 {  
  white-space: nowrap;  
  animation-duration: 6s;  
  animation-name: slidetext;  
  animation-iteration-count: infinite;  
  animation-timing-function: linear;  
  animation-direction: alternate;  
}
```

Mogao je i  
*shorthand*

```
@keyframes slidetext {  
  from {  
    margin-left: 100%;  
  }  
  to {  
    margin-left: 0%;  
  }  
}
```

# Animations demo



## **Vendor prefixes**

- <https://www.w3.org/Style/CSS/current-work>
  - *Working draft, candidate recommendation (draft), candidate recommendation*
- Proizvođači preglednika žele omogućiti neka svojstva i prije nego što su u CR fazi
  - Definirati pomoću **vendor prefixa**
  - Čak i ako dođe do breaking changes neće doći do promjene u radi early-adopting web stranica
- Primjer na sljedećoj stranici

## Vendor prefixes - autoprefixer

- Napravimo c/p style sekcije iz npr. transition demo
  - <https://autoprefixer.github.io/>
  - [https://developer.mozilla.org/en-US/docs/Glossary/Vendor\\_Prefix](https://developer.mozilla.org/en-US/docs/Glossary/Vendor_Prefix)

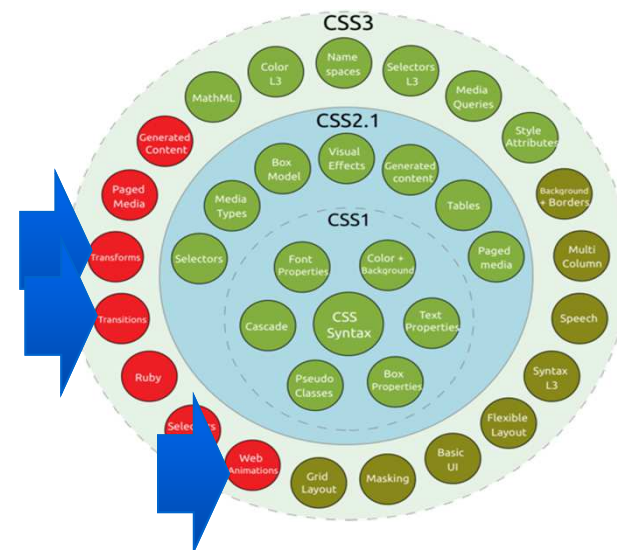
```
#ferlogo:hover {  
  -webkit-transform: translate(-50%,  
                                -50%) scale(1.3, 1.3) rotate(90deg);  
  -ms-transform: translate(-50%,  
                           -50%) scale(1.3, 1.3) rotate(90deg);  
  transform: translate(-50%,  
                       -50%) scale(1.3, 1.3) rotate(90deg);  
  
  border: 3px solid green;  
  background-color: cornflowerblue;  
}
```

# ZOKŽZV

- pogledati:
  - @supports
    - <https://developer.mozilla.org/en-US/docs/Web/CSS/@supports>
  - polyfills

## Kako dalje?

- Ovime smo prošli "sva" značajnija CSS svojstva i module



- Kako dalje?
  - **Kako pisati / ustrojiti CSS u većim projektima?**
  - Kako pisati nazive klasa?
  - Koristiti ili ne CSS radne okvire?
- **Problem/svojstvo: CSS je globalan!**

[Install](#)[Learn Sass](#)[Blog](#)[Documentation](#)[Get Involved](#)

# CSS with superpowers

Sass is the most mature, stable, and powerful professional grade CSS extension language in the world.



Current Releases: [Dart Sass 1.35.1](#) [LibSass 3.6.5](#) [Ruby Sass](#)  [Implementation Guide](#)



# ■ SASS - Syntactically Awesome Style Sheets

- *CSS preprocessor* -> prevodi se u CSS
  - Preglednici ga ne poznaju, prevođenje u razvojnoj okolini
- Dvije sintakse (~ ekstenzije):
  - .sass ~ yml, python, koriste se novi red i uvlačenje (indent)
  - .scss - novija, "ista" kao CSS, blokovi s vitičastim zagradama
- Većina primjera u nastavku slijedi <https://sass-lang.com/guide> s malim izmjenama

# Variable

- (u "međuvremenu" su podržane i u CSS-u)

```
λ sass -w .:.  
Compiled site.scss to site.css.  
Sass is watching for changes. Press Ctrl-C to stop.
```

site.scss

```
$brand-color: blue;  
$font-heading: Arial, sans-serif;  
$font-body: Helvetica, sans-serif;  
$primary-color: #333;
```

```
body {  
  font: 100% $font-body;  
  color: $primary-color;  
}
```

site.css

```
body {  
  font: 100% Helvetica, sans-serif;  
  color: #333;  
}
```

Namjerno dodan  
višak

# Gniježđenje

- HTML omogućuje gniježđenje (hijerarhijsku strukturu) - CSS ne!

- SASS omogućuje gniježđenje
  - Selektora
  - Svojstava

nesting.scss

```
nav {  
  ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
  }  
}  
body {  
  font: {  
    family: Iosevka, sans-serif;  
    size: 16px;  
    weight: bold;  
  }  
}
```

nesting.css

```
nav ul {  
  margin: 0;  
  padding: 0;  
  list-style: none;  
}  
  
body {  
  font-family: Iosevka, sans-serif;  
  font-size: 16px;  
  font-weight: bold;  
}
```

Svojstvo font-

## Partials and modules

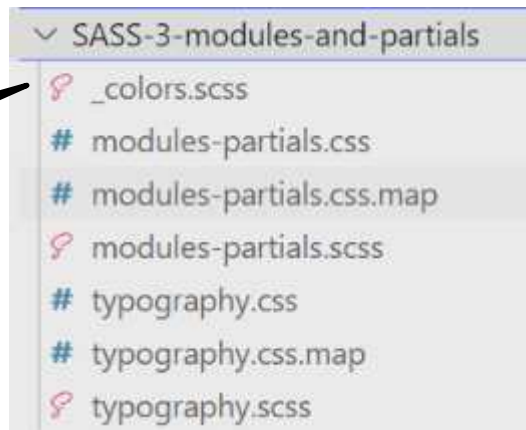
Moguće i u [CSS-u](#), ali onda imamo više datoteka

- DRY principle
- Moguće je kod izdvojiti u posebne datoteke koje zatim možemo uključiti u drugim datotekama i tako ostvariti modularni ustroj SASS projekta
  - *partials*, počinju s `_`, ne prevode se zasebno
  - *modules*, prevode se zasebno
- Nalik include u C-u, nije problem ako se više puta uključi (kao i u C-u)
- Npr, mogli bi imati:
  - `_colors.scss` i `_typography.scss`
- ...i onda ih uključivati gdje nam trebaju s `@use` (prije `@import`):
  - `@use "colors"; @use "typography"`

Ne treba navesti `_` ni ekstenziju

# SASS-3-modules-and-partials

Primijetiti da nema colors.css jer je partial.



\_colors.scss

```
$zelena: green;  
$crvena: red;  
$plava: blue;
```

typography.scss

```
body {  
  font: {  
    family: Iosevka;  
    size: 16px;  
    weight: bold;  
  }  
}
```

modules-partials.scss

```
@use "colors";  
@use "typography";  
  
body {  
  color: colors.$plava;  
}
```

modules-partials.css

```
body {  
  font-family: Iosevka;  
  font-size: 16px;  
  font-weight: bold;  
}  
  
body {  
  color: blue;  
}
```

# Naslijeđivanje

- Pomoću `@extend` možemo naslijediti (preuzeti) skup CSS svojstava od nekog selektora
  - Dodatno u primjeru koristimo "placeholder class" koja se materijalizira samo ako je naslijeđena - počinje s %

inheritance.scss

```
%button-default {  
  font-size: 16px;  
  cursor: pointer;  
}  
%ridiculous-button { font-size: 66px; }  
  
.button-success {  
  @extend %button-default;  
  background-color: green;  
}  
.button-danger {  
  @extend %button-default;  
  background-color: red;  
}
```

Nije  
naslijeđen

inheritance.css

```
.button-danger, .button-success {  
  font-size: 16px;  
  cursor: pointer;  
}  
  
.button-success {  
  background-color: green;  
}  
  
.button-danger {  
  background-color: red;  
}
```

ZOKŽZV: <https://csswizardry.com/2014/11/when-to-use-extend-when-to-use-a-mixin/>

# SASS - itd.

- Svojstva:
  - Variables
  - Nested selectors
  - Nested properties
  - **Partials and modules, ~~@import~~/@use**
  - media queries, nested
  - Inheritance **@extend**
  - Built-in modules (korisne funkcije: <https://sass-lang.com/documentation/modules> )
    - Lists, maps, npr. map of colors, math, ...
  - Simple arithmetics <https://sass-lang.com/documentation/operators>
  - Mixins <https://sass-lang.com/documentation/at-rules/mixin> <https://csswizardry.com/2014/11/when-to-use-extend-when-to-use-a-mixin/>
  - Itd.
- Vrlo slično: LESS
  - <https://lesscss.org/>
  - <https://css-tricks.com/sass-vs-less/>

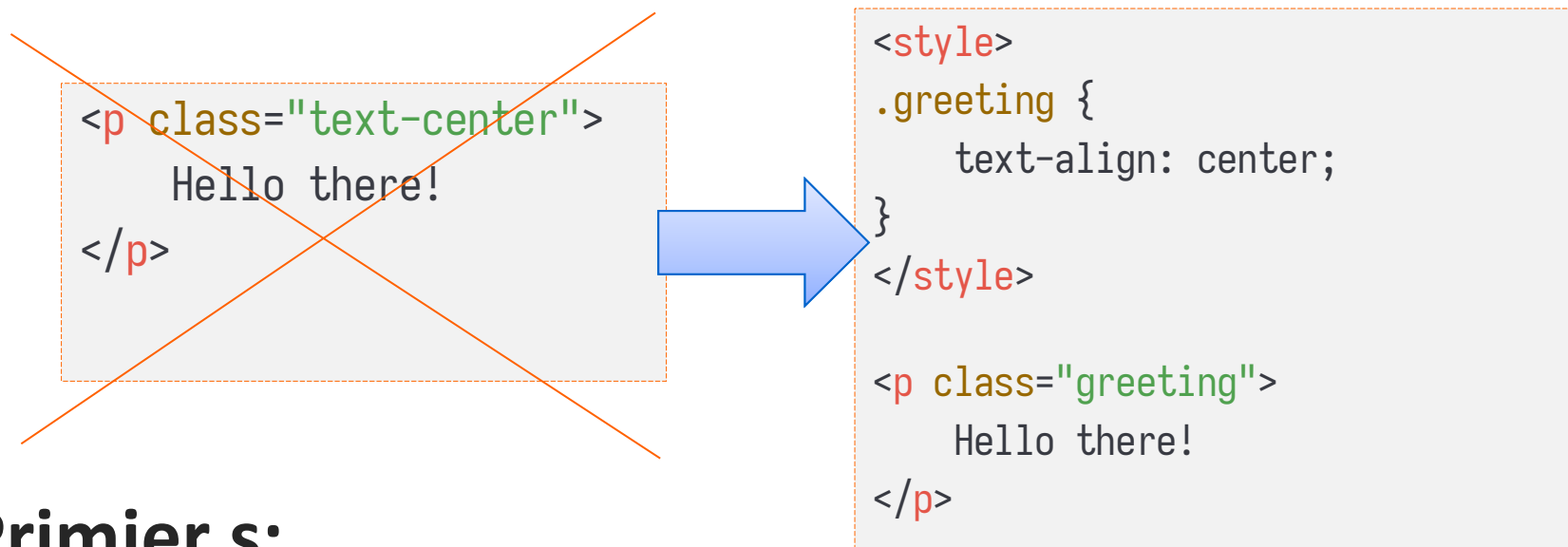
## Kako pisati CSS

- Sljedećih nekoliko slajdova preuzeto s:  
<https://adamwathan.me/css-utility-classes-and-separation-of-concerns/>



# 1. "Semantic" CSS

- "Separation of concerns"
  - HTML definira **sadržaj**
  - CSS definira **izgled**
- Primjer: <http://www.csszengarden.com/>



**Primjer s:**

<https://adamwathan.me/css-utility-classes-and-separation-of-concerns/>

**ALI: u složenijim slučajevima CSS nerijetko zrcali strukturu HTML-a**

## 2. Razdvajanje stilova od strukture

- Koristiti neku metodologiju imenovanja, npr. SMACSS ili BEM, vidjeti i npr. <https://css-tricks.com/css-style-guides/>

- BEM:

**block\_element--modifier**

može i --

- Primjer na sljedećoj stranici ->
- Niska specifičnost (*low specificity*), izravnavanje specifičnosti selektora (*specificity flatness*)
- *"BEM gives everyone on a project a declarative syntax that they can share so that they're on the same page."*
- <https://css-tricks.com/bem-101/>
- Problem održavanja koda - strah od izmjene

# <http://getbem.com/introduction/>

## Block

Standalone entity that is meaningful on its own.

### Examples

`header` , `container` , `menu` , `checkbox` , `input`

## Element

A part of a block that has no standalone meaning and is semantically tied to its block.

### Examples

`Normal button` `Success button` `Danger button`

## Modifier

A flag on a block or element. Use them to change appearance or behavior.

### Examples

`disabled` , `highlighted` , `checked` , `fixed` , `size`  
`big` , `color yellow`

We can have a normal button for usual cases, and two more states for different ones. Because we style blocks by class selectors with BEM, we can implement them using any tags we want ( `button` , `a` or even `div` ). The naming rules tell us to use `block--modifier-value` syntax.

## HTML

```
<button class="button">
  Normal button
</button>
<button class="button button--state-success">
  Success button
</button>
<button class="button button--state-danger">
  Danger button
</button>
```

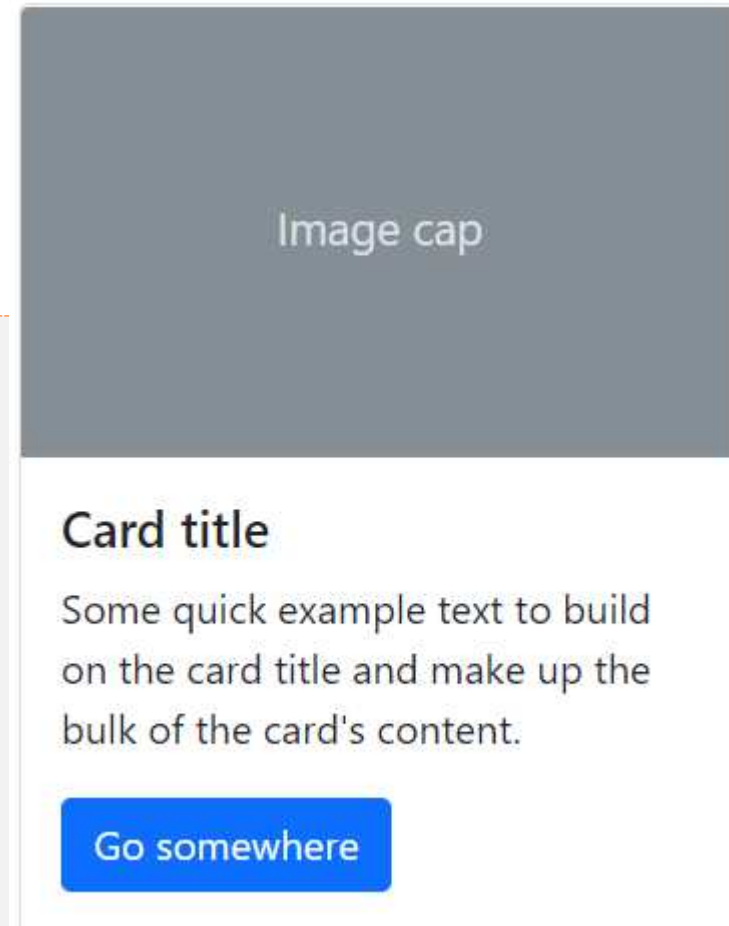
## CSS

```
.button {
  display: inline-block;
  border-radius: 3px;
  padding: 7px 12px;
  border: 1px solid #D5D5D5;
  background-image: linear-gradient(#EEE, #DDD);
  font: 700 13px/18px Helvetica, arial;
}
.button--state-success {
  color: #FFF;
  background: #569E3D linear-gradient(#79D858, #569E3D) repeat-x;
  border-color: #4A993E;
}
.button--state-danger {
  color: #900;
}
```

### 3. Nezavisni CSS s gotovim komponentama

- CSS definira stil standardnih komponenti
  - **.button**
  - **.card**
  - itd.
- **Primjer - Bootstrap**

```
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Card title</h5>
    <p class="card-text">Some quick example text
to build on the card title and make up the bulk
of the card's content.</p>
    <a href="#" class="btn btn-primary">Go
somewhere</a>
  </div>
</div>
```



## 4. Nezavisni CSS s pomočnim klasama

- CSS definira klase za
  - Text sizes, colors, and weights
  - Border colors, widths, and positions
  - Background colors
  - Flexbox utilities
  - Padding and margin helpers, ...
- Primjer – TailwindCSS, Tachyons
- ***Enforced consistency***
  - Sužen skup možnosti
  - Npr. GitLab: 402 text colors, 239 background colors, 59 font sizes
- Treba li svejedno definirati komponente?

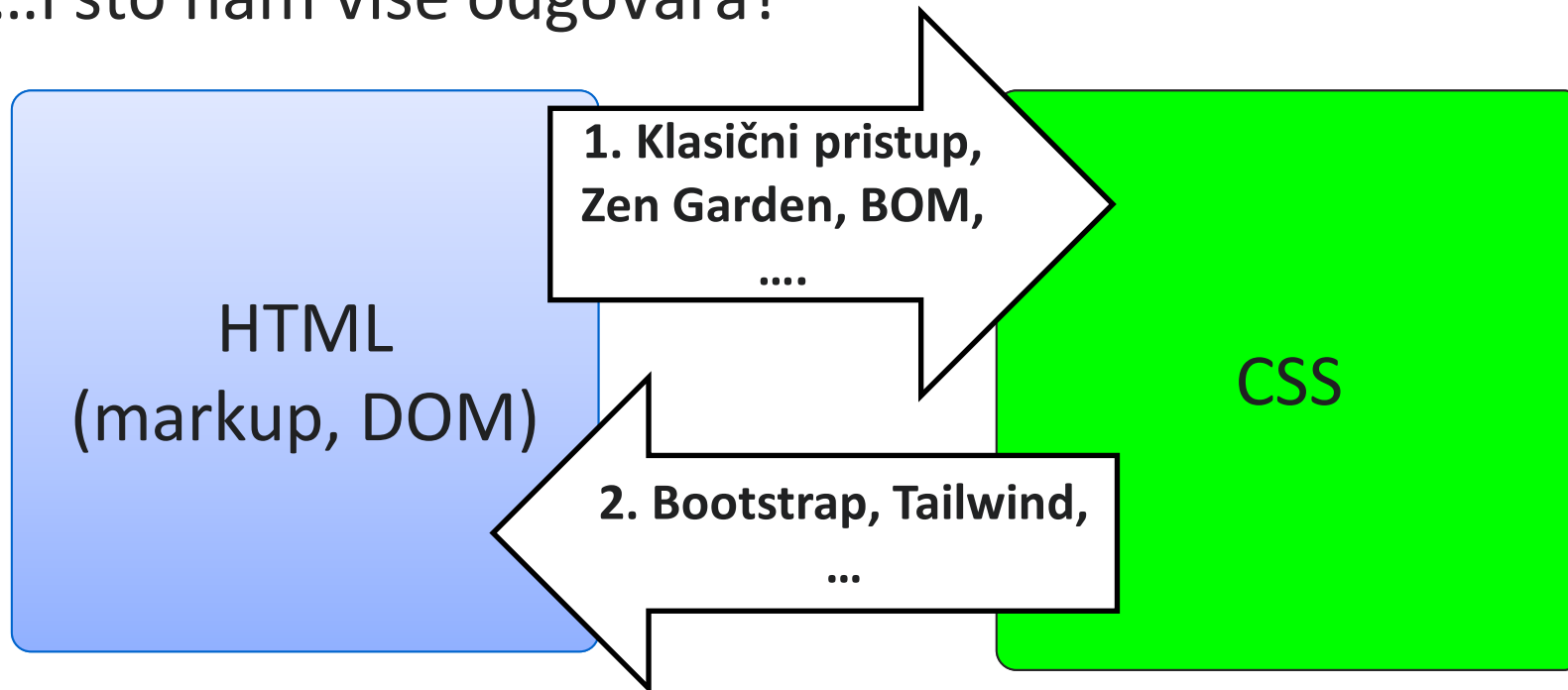
```
<button class="bg-blue-500  
  hover:bg-blue-700  
  text-white font-bold  
  py-2 px-4 rounded">  
  Button  
</button>
```



<https://v1.tailwindcss.com/components/buttons>

## Tko o kome ovisi?

- ...i što nam više odgovara?



1. HTML-u se može "izvana" promijeniti stil, ali CSS nije ponovo upotrebljiv
2. CSS ponovo upotrebljiv, HTML-u se ne može izvana promijeniti stil

# ■ Radni okviri

- Dvije kategorije:
  1. **Sveobuhvatni** radni okviri, component frameworks koji sadrže i komponente i pomoćne klase, layout sustav, itd.
    - Npr. Bootstrap, Foundation, ...
  2. Samo **pomoćne klase** (*utility classes*)
    - Npr. TailwindCSS, Tachyons, ...

# Vanilla CSS ili radni okviri?

- <https://academind.com/tutorials/vanilla-css-vs-frameworks/>

Vanilla CSS	Utility Frameworks	Component Frameworks
Full Control	Faster Development	Rapid Development
No unnecessary Code	Follow Best Practices	Follow Best Practices
Name Classes as you like	No Expert Knowledge Needed	No Need to be an Expert
Build everything from Scratch	Little Control	No or Little Control
Danger of "bad code"	Unnecessary Overhead Code	Unnecessary Overhead Code
		"All Websites Look the Same"



## ■ Nove „komplikacije” odnosno mogućnosti

- **SPA** radni okviri koji imaju **lokalni CSS scope**, na razini komponenti
  - U sljedećim predavanjima
- *Cascade layers*
  - Proposal
  - <https://css-tricks.com/cascade-layers/>