JÖNKÖPING UNIVERSITY

School of Engineering

CSS & HTML AND CSS FRAMEWORKS

Web Development with JavaScript and DOM

TWJK14 Spring 2017

Peter Larsson-Green

JÖNKÖPING UNIVERSITY
School of Engineering

CSS

HTML: Mark what type of data text represents.

- Browsers render the page.

CSS: Tell the browser how to render the data.

CSS LEVELS

1996: CSS 1

1998: CSS 2

2011: CSS 2.1

- Candidate recommendation 2004.

CSS 3 is made up of modules.

- Some have finished specifications.
- Some have almost finished specifications.
- Some are still early drafts.

CSS 4 continues with modules.

VERSIONS AND BROWSERS

- No modern browser supports all new CSS features.
- Old browsers do not support any new CSS feature.

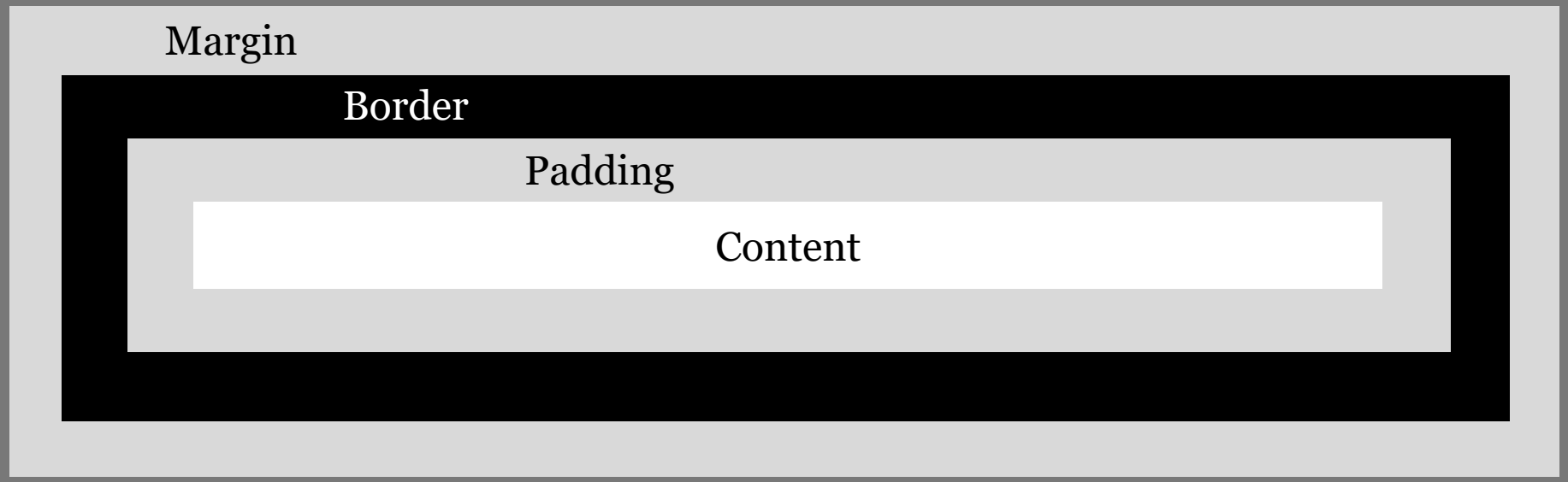
Which CSS features can we use?

- http://caniuse.com

THE BOX MODEL

Explains how the browsers render elements.

- All elements are rendered as boxes:



WHERE TO WRITE CSS CODE

1. In the global `style` attribute:

```
<p style="CSS-CODE">Some text</p>
```

   - Can't re-use our CSS code on other elements ☹

2. In the `<style>` element:

```
<style type="text/css">CSS-CODE</style>
```

   - Need to specify which elements that should be affected (selectors).
   - Can't re-use our CSS code in other files ☹

WHERE TO WRITE CSS CODE

3. In a separate `.css` file:

```
<link rel="stylesheet" href="the-css-file.css">
```

```
CSS-CODE
```

- Need to specify which elements that should be affected (selectors).
- Can use the same CSS code in multiple files ☺
- CSS files can be cached ☺

CSS SYNTAX

Declaration:

```
property-name: value;
```

```
<p>Some text.</p>
<p style="color: red">Some text.</p>
```

Some text.

Some text.

CSS SYNTAX

Declaration:

```
property-name: value;
```

Rule:

```
selector{
  declarations
}
```

```
<style>
p{
  color: red
}
</style>
<p>Some text.</p>
<p>Some text.</p>
```

Some text.

Some text.

CSS SELECTORS

`tagname`	The elements with the tag `tagname`.
`#the-id`	The element with the attribute: `id="the-id"`
`.a-class-name`	The elements with the attribute: `class="a-class-name"`
`*`	All elements.

EXAMPLE

```
<style>
  p{ color: red }
  #cool{ font-weight: bold }
  .happy{ background-color: lime }
</style>

<p>Some text.</p>
<p class="happy">Some text.</p>
<p id="cool">Some text.</p>
<div class="happy">Some text.</div>
```

RELATIONAL SELECTORS

```
selectorA selectorB
```

The elements mathed by `selectorB` that are inside an element matched by `selectorA`.

```
selectorA > selectorB
```

The elements matched by `selectorB` that are direct children to an element matched by `selectorA`.

```
selectorA + selectorB
```

The elements matched by `selectorB` that comes directly after an element matched by `selectorA`.

And more!

EXAMPLE

```
<style>
  p span{ color: red }
</style>

<p>Some <span>text</span>.</p>
<span>Some text.</span>
<p>
  Some <strong><span>text</span></strong>.
</p>
```

Some text.

Some text.

Some **text**.

EXAMPLE

```
<style>
  p > span{ color: red }
</style>

<p>Some <span>text</span>.</p>
<span>Some text.</span>
<p>
  Some <strong><span>text</span></strong>.
</p>
```

Some text.

Some text.

Some **text**.

MULTIPLE SELECTORS

```
selectorA, selectorB
```

The elements mathed by `selectorA` or `selectorB`.

```
selectorAselectorB
```

The elements matched by `selectorA` and `selectorB`.

EXAMPLE

```
<style>
  p.happy{
    color: red;
  }
</style>

<p>Some text.</p>
<p class="happy">Some text.</p>
<span class="happy">Some text</span>
```

Some text.

Some text.

Some text.

SELECTORS WITH PSEUDO-CLASSES

`theSelector:first-child`	The elements mathed by `theSelector` when they are the first child in its parent.
`theSelector:focus`	The elements mathed by `theSelector` when they has focus.
`theSelector:hover`	The elements mathed by `theSelector` when the mouse hovers over them.
`theSelector:visited`	The links mathed by `theSelector` when they have been visited.

And more!

SELECTORS WITH ATTRIBUTES

```
theSelector[attr]
```

The elements mathed by `theSelector` and have the attribute `attr`.

```
theSelector[attr=value]
```

The elements mathed by `theSelector` and have `attr="value"`.

And more!

CONFLICTING RULES

```
<style>
  p{ color: red }
  #cool{ color: blue }
  .happy{ color: yellow }
</style>

<p id="cool" class="happy">Some text.</p>
```

Some text.

SELECTOR SPECIFICITY

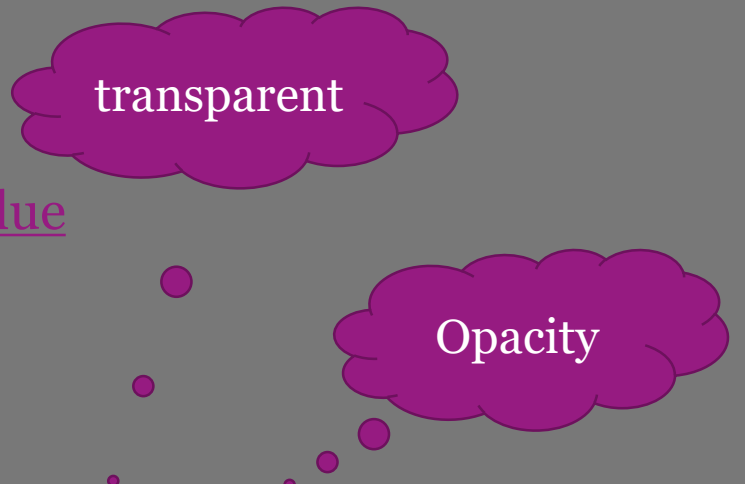
https://www.w3.org/TR/css3-selectors/#specificity

Generalized rules (not always valid!):

1. Style attribute.
2. Id selector.
3. Class selector.
4. Tag name selector.
5. Universal selector (*).

CSS COLORS

- The name of the color.
  https://developer.mozilla.org/en-US/docs/Web/CSS/color_value
- `transparent`
- `rgb(R, G, B)` 0 <= R, G, B <= 255
- `rgba(R, G, B, a)` 0 <= R, G, B <= 255, 0 <= a <= 1
- `#RRGGBB` 00 <= RR, GG, BB <= FF



CSS UNITS

https://www.w3.org/Style/Examples/007/units.en.html

- Absolute:
  - cm, mm, in, px, pt…
  - Usually don't work well on different screen sizes ☹
- Relative:
  - % - percentage of parent
  - em
  - vw - % of view width
  - vh - % of view height
  - vmin - % of the smallest of the view width and view height
  - vmax - % of the biggest of the view width and view height

MEDIA QUERIES

Use different CSS depending on capabilities.

```
<link rel="stylesheet" href="file.css" media="MEDIA-QUERY">
```

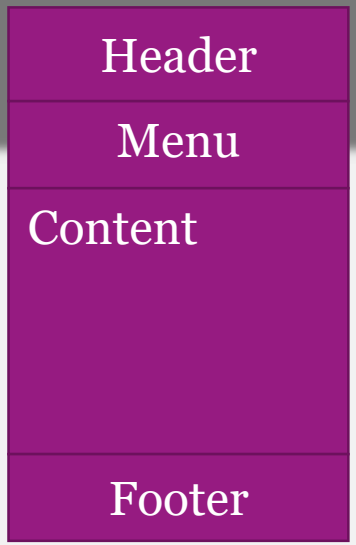
```
<style>
  @media MEDIA-QUERY {
    /* Ordinary CSS code. */
  }
</style>
```

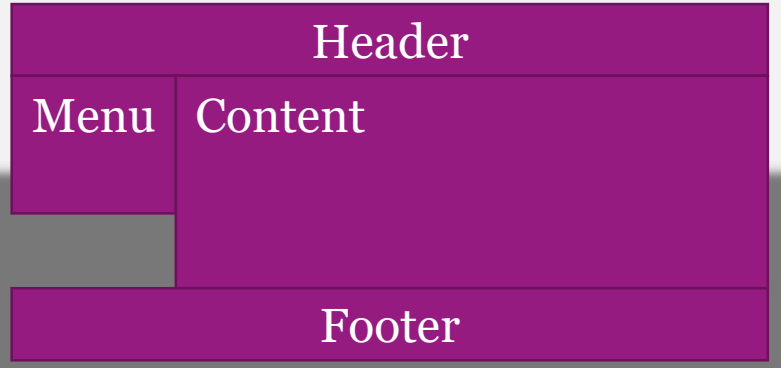
MEDIA QUERY EXAMPLES

```
<style>
  @media screen{
    /* CSS code for screens. */
  }
  @media print{
    /* CSS code for printers. */
  }
</style>
```

MEDIA QUERY EXAMPLES

```
<style>
  @media screen and (max-width: 300px){
    /* CSS code for small screens. */
  }
  @media screen and (min-width: 301px){
    /* CSS code for big screens. */
  }
</style>
```





CREATING LAYOUTS

Has always been hard using HTML & CSS!

- Want to support old browsers → we can't use new features.
- Even if a feature is implemented, it might work differently in different browsers (bugs).

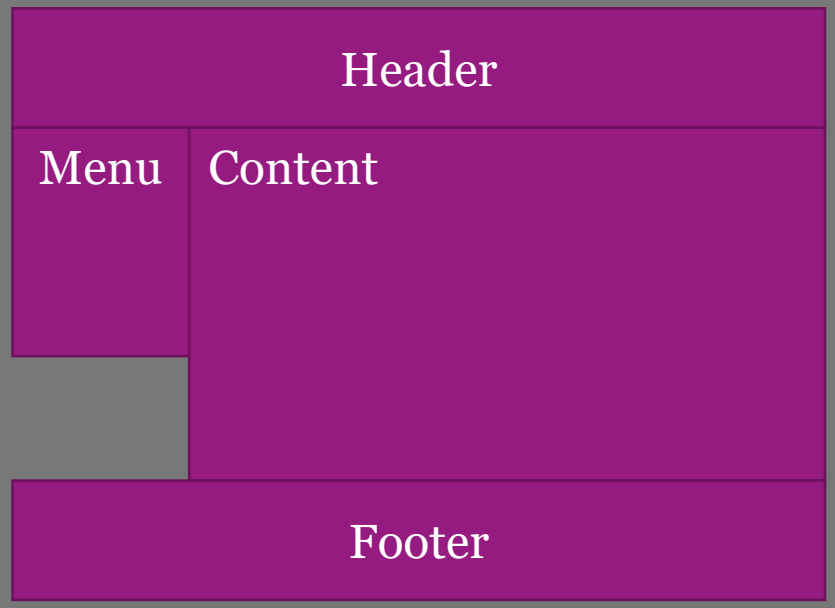
http://learnlayout.com

LAYOUT EXAMPLE

Easy using `display: inline-block`.

- Works in ~99.9% of the browsers in use today.

  http://caniuse.com/#feat=inline-block



See code files!

HTML & CSS FRAMEWORKS

Creating layouts has always been hard.

- Different browsers supports different features.
- Should work on different screen sizes.
- Should be controllable through different devices:
  - Mouse.
  - Keyboard.
  - Touchscreen.
  - ...
- Should work for users with disabilities.

Solution: Use a CSS & HTML framework.

HTML & CSS FRAMEWORKS

How to use them (in general):

- Link to CSS files in your HTML files.
- Use the HTML components the framework provides you with.
  - Usually HTML elements with classes.

NORMALIZE.CSS

Makes browsers render your HTML code more consistent.

- Consists of a single CSS file:
  https://necolas.github.io/normalize.css/latest/normalize.css
- Is hosted on GitHub:
  https://github.com/necolas/normalize.css

BOOTSTRAP

An HTML, CSS & JavaScript framework.

- Started out as a project by Twitter.
- Released as open source 2011.
- Second most starred project on GitHub.
  - https://github.com/search?q=stars:%3E1&s=stars&type=Repositories
- Webpage: http://getbootstrap.com

RECOMMENDED READING

W3Sschools

- CSS tutorial
  - https://www.w3schools.com/css/default.asp

W3's CSS specifications:

- https://www.w3.org/Style/CSS/specs

Learn layouts

- http://learnlayout.com