CSS

Styling Content

Lecture 2: Jan. 11 2016

How CSS applies to HTML

Syntax, inheritance, the cascade, and selectors

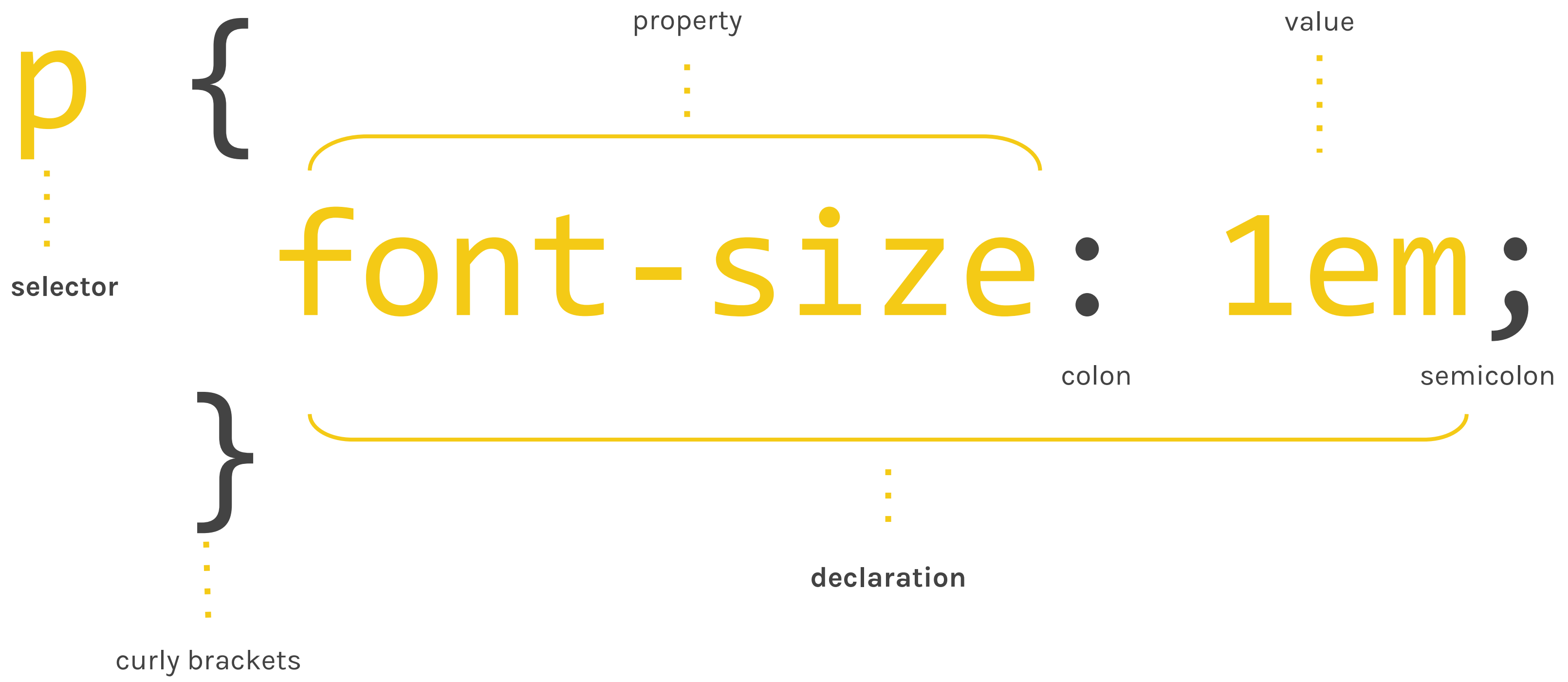
CSS

CSS is the language determining the look and feel of HTML elements.

CSS = `Cascading Stylesheets`

- CSS defines the **form** of a website's content by listing rules for how elements should appear.
- **CSS** = **selector** (targeted html element) + **declaration** (rules applied to that element)
- **Declaration** = **property** (what you're formatting, such as size and color) + **value** (what you're setting the specified property to be)

Components of CSS



Inline CSS

Inline styles are specified as an attribute inside of an individual HTML element with the syntax `style=“property: value;”` and applies only to that element. This is the least desirable way, as it can be inefficient and lead to inconsistencies.

Code

```
<p style= “color: yellow;”> First paragraph text.
</p>
<p> Second paragraph text. </p>
```

Output

First paragraph text.

Second paragraph text.

Internal CSS

Code

```
<html>
<head>
     <style type="text/css">
          p {
               color: yellow;
          }
     </style>
</head>
<body>
     <p>First paragraph text.</p>
     <p>Second paragraph text.</p>
</body>
</html>
```

Internal CSS is declared within the head of the document. Selectors apply to all instances of that page.

This keeps the HTML markup clean and uncluttered. It's much more efficient than inline styling, but not as effective as external stylesheets.

Output

First paragraph text.

Second paragraph text.

External CSS

External CSS keeps all css declarations in a separate document that gets pulled into the webpage with `<link rel="stylesheet" type="text/css" href="styles.css" >` Selectors apply to all instances of elements in all webpages that use the same stylesheet (as in link to the same css file.)

This method is the ideal method of formatting your site, since it ensures consistency across pages. It also keeps things flexible: by changing one property in the external stylesheet, all instances could be easily changed.

Inheritance

Inheritance refers to how children take on css properties of their parents if they don't have that property specified. Not all css declarations are inherited.

Stylesheet

```
div {
    color:blue;
    border: 1px solid gray;
}

span {
    color:red;
}
```

HTML

```
<div>
    <h1>Cascading
    <span>Style</span>
    Sheets</h1>
</div>
```

Output

```
Cascading Style Sheets
```

h1 inherits the color property of parent div,
but not the border property

The Cascade

Cascade refers to the way the stylesheet processes and assigns **weights** to rules that apply to the same element, ultimately determining which properties will modify a given element.

Origin and Importance: Stylesheets may come from 3 different sources, processed in the following order

1. **User agent:** The browser's default style sheet.
2. **User:** Such as the user's browser options.
3. **Author:** This is the CSS provided by the page (whether inline, embedded or external.) Inline > Internal > External

note: `!important` allows overrides

Specificity: when rules within the same stylesheet conflict, the type of **selector** determines which has more weight

Rule order: between style rules of identical weight, last one wins

Selectors

By understanding inheritance and the cascade, we can write overarching rules that apply to most elements, then override the properties on individual elements. **Selectors** help tailor our rules to define specific elements. The more specific the selector, the higher its priority.

We've already seen the **type** selector that matches element names.

By separating selectors with commas, you can apply the same rule to multiple HTML elements.

- See list of selectors

Classes and IDs

```
.highlight {
    background: yellow;
}

#demo {
    font-weight: bold;
}
```

```
<p id="demo">This is Demo text</p>
<p class="highlight"> Paragraph text 1</p>
<p> Paragraph text 2 </p>
<ul>
    <li class="highlight">List item 1 </li>
    <li>List item 2</li>
    <li>List item 3</li>
</ul>
```

By assigning a **class** to elements (with the class attribute) in your HTML, we can apply your rule to just elements that have that particular class. In your stylesheet, all class names are preceded by a period (.)

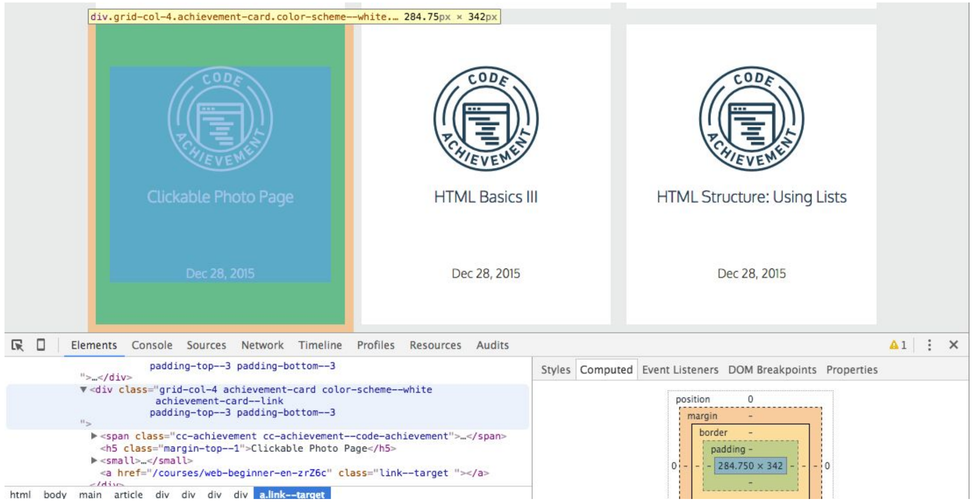
Ids work similarly to classes, but **only one** element may be given a particular **id**. You define id names by preceding them with a hash symbol (#)

Relationship selectors

Certain selectors can target elements based on how they are nested (relationships)

selector	description	sample syntax
Child (>)	matches an element that is a direct child of another	`li>a { }`
Descendant ( )	matches an element that is a descendent of another, not just a direct child	`p a { }`
Adjacent sibling selector (+)	matches only the specified element that immediately follows the former specified element	`h1+p { }`
General sibling selector (~)	matches the second element only if it is preceded by the first, and both share a common parent	`h1~p { }`

Use the Inspector



```
            padding-top--3 padding-bottom--3
  ">…</div>
▼<div class="grid-col-4 achievement-card color-scheme--white
             achievement-card--link
            padding-top--3 padding-bottom--3
  ">
   ▶<span class="cc-achievement cc-achievement--code-achievement">…</span>
    <h5 class="margin-top--1">Clickable Photo Page</h5>
   ▶<small>…</small>
    <a href="/courses/web-beginner-en-zrZ6c" class="link--target "></a>
```

The element inspector is a great way to verify the CSS properties applied to your elements.

When thinking about CSS properties, imagine that there is an invisible box around each HTML element.

Understanding CSS properties

Color, text, and the box model

Color

Format	sample syntax

RGB(A)	`color: rgb (255,0,0);`
	`color: rgba (255,0,0,0.5);`
	`/* 50% transparent */`

HSL(A)	`color: hsl(0,100%,50%);`
	`color: hsla(0,100%,50%,0.5);`
	`/* 50% transparent */`

Hex	`color: #ff0000;`
	`color: #f00;`

Name	`color: red;`

Color can be specified in various formats. You can inspect color codes by using the color picker in your dev tools (inspector.)

note: the `color` css property only specifies the foreground **text-color** of elements. Use `background-color` to specify the color of your entire element area.

Text

Some common text-styling properties:

property	description	possible values
`text-align`	alignment of text	`left, right, center, justify`
`font-family`	what typeface	*see* typefaces
`font-size`	text size	*see* units
`font-weight`	the weight of your type	`normal, bolder, 700`
`font-style`	text formatting	`normal, italic, oblique`
`line-height`	"leading" or height of box surrounding line	best declared **unit-less,** relative to font (see why)
`text-transform`	text case	`capitalize, uppercase, lowercase, none`

Typefaces

There are 3 sources of fonts from which you can choose:

System fonts: These are fonts installed on your computer. Your users must also have the font installed on their machine. There are several fonts known to be installed across Windows and Mac computers, such as Arial, Helvetica, Times New Roman, Georgia, Verdana. Choices are limited.

Externally hosted webfonts: Google Fonts (free!) & Typekit offer services with easy implementation.

Self-hosted webfonts: if you upload your own font files, you can use the @font-face rule. Font Squirrel has a web font generator that converts desktop fonts to web font formats, but you may not have the permissions. More on the @font-face use and web font formats.

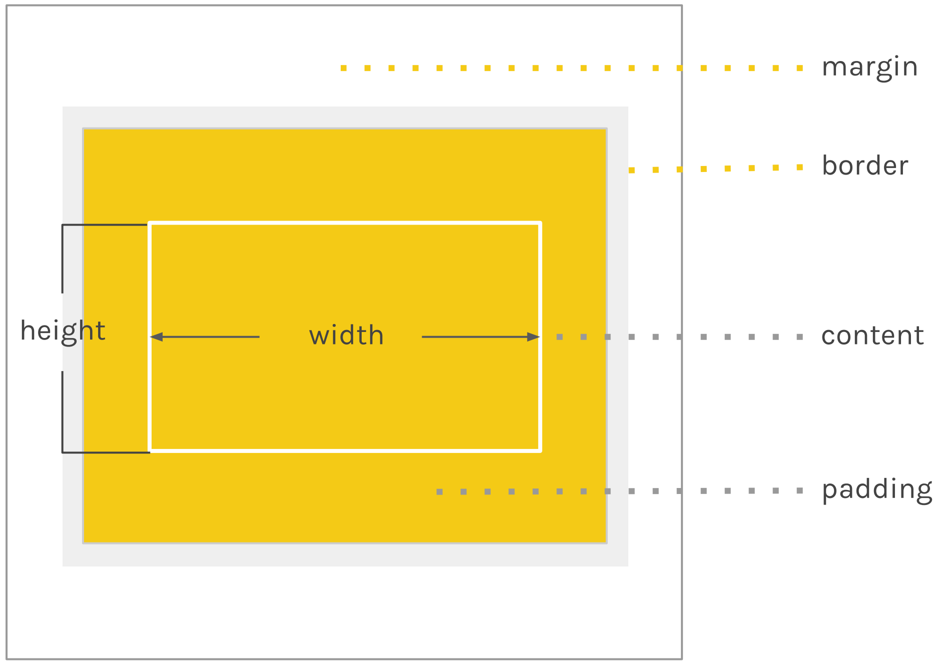
Units

	`body { font-size: 100%; }`	`body { font-size: 150%; }`
`font-size: 1em;`	your text size	your text size
`font-size: 16px;`	your text size	your text size
`font-size: 100%;`	your text size	your text size

Pixels are “absolute” measurements, relative to the resolution of the screen. The same type size will look larger when a screen has a resolution of 800x600 than it would when it is 1280x800.

Ems and **Percentages** are scalable measurements. The units refer to the current font size, relative to the inherited font size of the element. It is best to use scalable units in order for your text to be legible across various devices.

The Box Model



The box model refers to how **block-level** elements can be controlled.

```
.box {
     width: 300px;
     height: 200px;
     border: 6px solid #cccccc;
     padding: 10px;
     margin: 10px 20px 10px auto;
}
```

shorthand notation for:

```
margin-top: 10px;
margin-right: 20px;
margin-bottom: 10px;
margin-left: auto;
```

Negative Space

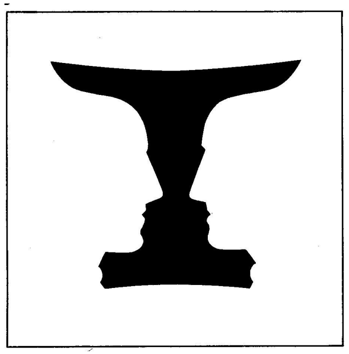
Negative space refers to the empty space in between objects.

Use of negative space is a fundamental tool in design.

It allows you to:

- Dissociate / associate elements on a page
- Give “breathing room” to increase the legibility of text
- Highlight / give priority to particular elements

Less is more!



Introduction to CSS: Review

You should now have an understanding of how to:

- ❏ Write CSS rules
- ❏ Use external CSS stylesheets
- ❏ Leverage how styles “cascade”
- ❏ Target elements with selectors
- ❏ Specify color and text properties
- ❏ Use the box model