

# IN4MATX 133: User Interface Software

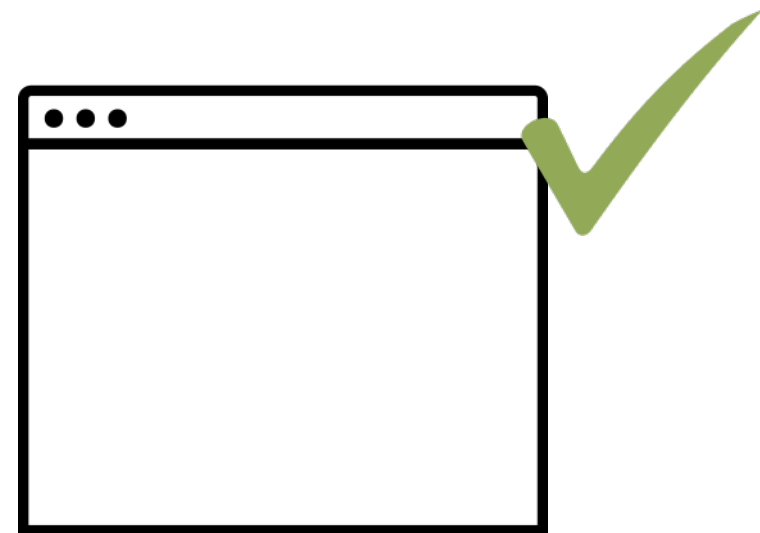
HTML & CSS

# Today's goals

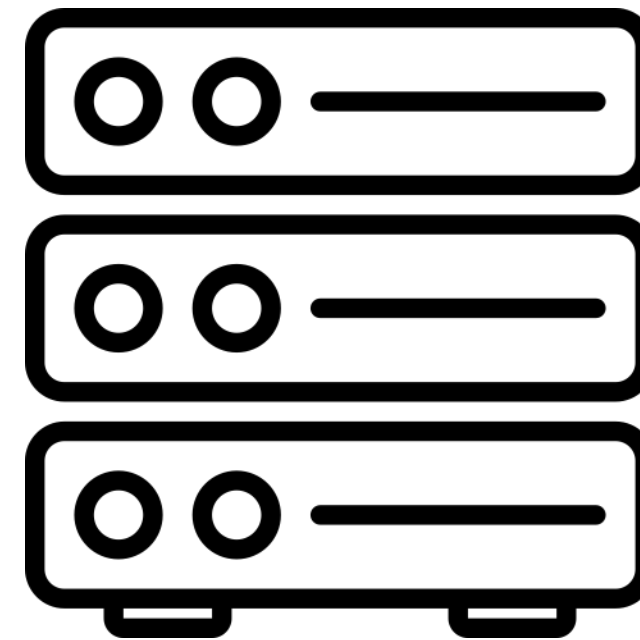
**By the end of today, you should be able to...**

- Describe the fundamentals of web communication
- Identify the syntax of HTML tags and attributes and describe their roles
- Create a HTML template which follows W3C specifications
- Explain the goals of CSS and why it exists as separate from HTML
- Describe the CSS hierarchy and fallback structure

# Client-side web development

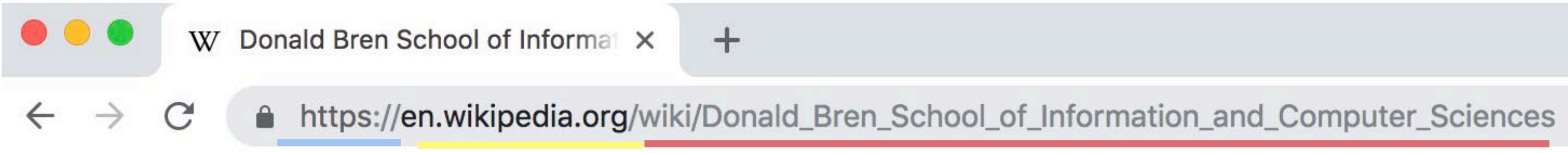


Your browser



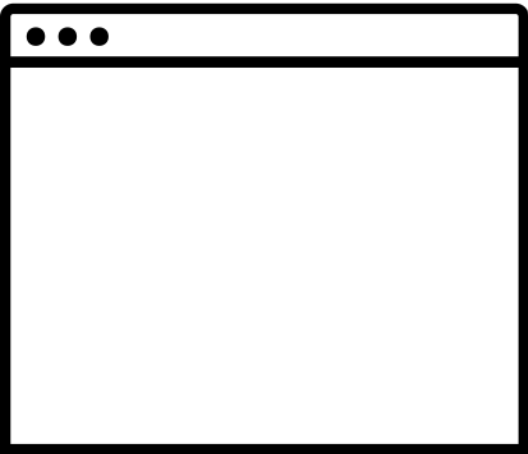
Web server

# Using the internet

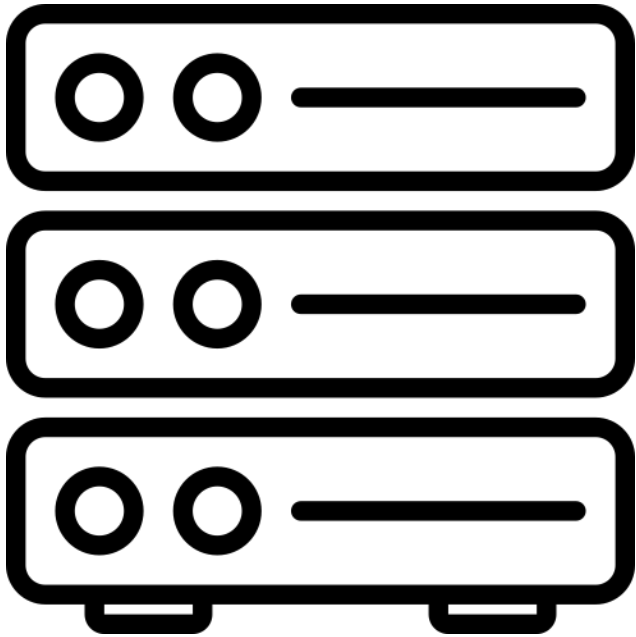
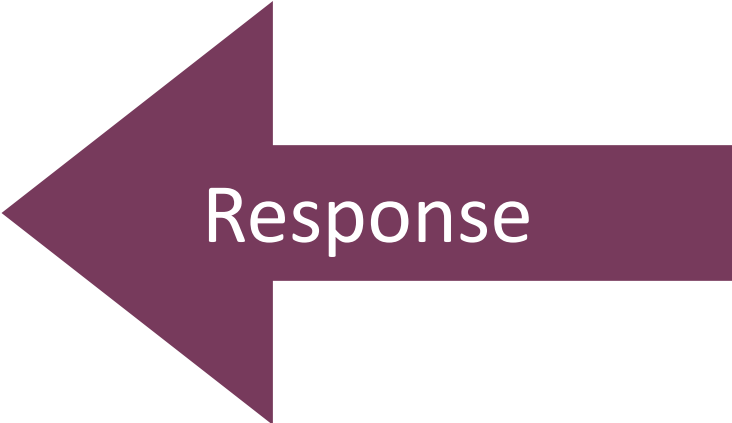
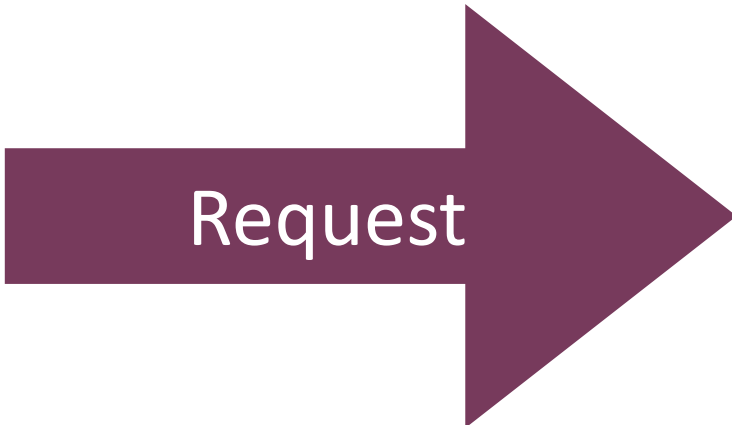


Protocol    Host    Resource  
(how to handle info)    (who has info)    (what info you want)

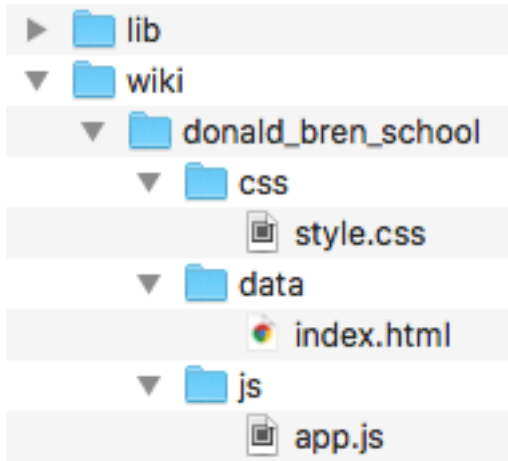
“Hey Wikipedia, I’d like to see the page for the school of ICS!”

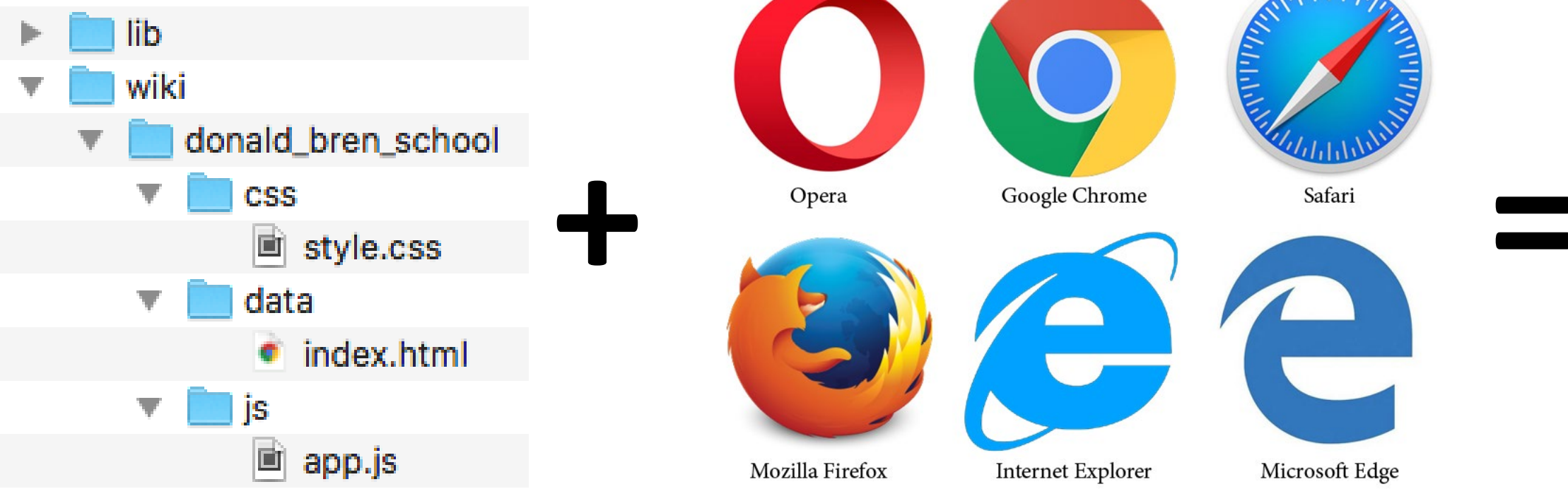


Your browser



Web server





Wikipedia article page for "Donald Bren School of Information and Computer Sciences".

URL: [https://en.wikipedia.org/wiki/Donald\\_Bren\\_School\\_of\\_Information\\_and\\_Computer\\_Sciences](https://en.wikipedia.org/wiki/Donald_Bren_School_of_Information_and_Computer_Sciences)

Not logged in | Talk | Contributions | Create account | Log in

Article | **Talk** | Read | Edit | View history | Search Wikipedia

**Wiki Loves Monuments:** The world's largest photography competition is now open! Photograph a historic site, learn more about our history, and win prizes.

## Donald Bren School of Information and Computer Sciences

From Wikipedia, the free encyclopedia


Coordinates: 33°43′N 117°42′W﻿ / ﻿33.6432°N 117.842°W﻿ / 33.6432; -117.842

**This article has multiple issues.** Please help [improve it](#) or discuss these issues on the [talk page](#). *(Learn how and when to remove these template messages)*

- This article **contains content that is written like an advertisement**. *(April 2016)*
- This article **may rely excessively on sources too closely associated with the subject**, potentially preventing the article from being [verifiable](#) and [neutral](#). *(January 2015)*

The **Donald Bren School of Information and Computer Sciences**, also known colloquially as UCI's **School of ICS** or simply the **Bren School**, is an academic unit of [University of California, Irvine](#) (UCI), and the only dedicated school of [computer science](#) in the [University of California](#) system. Consisting of nearly three thousand students, faculty, and staff,<sup>[2]</sup> the school maintains three buildings in the South-East artery of UCI's undergraduate campus, and maintains student body and research affiliations throughout UCI.<sup>[3][4]</sup>

The school of ICS consists of three departments: Computer Science, [Informatics](#), and [Statistics](#). The combined groupings focus the school around the fields of [computing](#) and processing of information. The departments confer eight undergraduate, eleven masters, and seven doctoral degrees in total, with some degree programs cooperating with affiliated schools.<sup>[5]</sup>

 Donald Bren Hall, one of the buildings on the campus of the Bren School<sup>[1]</sup>

Fundamentally, the web is  
designed to send files around

So what does a file on the web look like?

What if we wanted to specify  
**how** the content is rendered?



# HTML (HyperText Markup Language)

- Adds meaning to text
- Links documents to one another
  - Vanneaver Bush, hypertext vision



# Tags

`<div>` ← Open/start tag

Content goes here. ← Content

`</div>` ← Close/end tag

Whitespace and tag case are ignored

# Some common tags

`<h1>`Heading level 1`</h1>`

`<h2>`Heading level 2`</h2>`

...

`<p>`A paragraph`</p>`

`<!--A comment-->`

`<img>` An image

`<ul>` An unordered list (bullets)

`<li>` A list item

`<table>` A data table

`<strong>` Important content (**bolded**)

`<em>` Emphasized content (*italicized*)

`<div>` A division (section) of content

# Tags

- There are hundreds of tags!
- You may not use them all, but it’s good to explore them
- Search on Google or W3C to understand each tag’s purpose
- <https://www.w3schools.com/tags/>

HTML 5 NEW TAG			
TAG NOT SUPPORTED IN HTML 5			
<!--...-->	Define a comment		
<!DOCTYPE>	Defines the document type		
<a>	Defines a hyperlink	href, hreflang, media, ping , rel, target, type	
<abbr>	Defines an abbreviation		
<acronym>	Used to define an embedded acronyms		
<address>	Defines an address element		
<applet>	Used to define an embedded applet		
<area>	Defines an area inside an image map	alt, coords, href, hreflang, media, ping, rel, shape, target, type	
<article>	Defines an article	cite, pubdate	
<aside>	Defines content aside from the page content		
<audio>	Defines sound content	autobuffer, autoplay, controls, src	
<b>	Defines bold text		
<base>	Defines a base URL for all the links in a page	href, target	
<basefont>	Used to define a default font-color, font-size, or font-family for all the document		
<bdo>	Defines the direction of text display	dir	
<big>	Used to make text bigger		
<blockquote>	Defines a long quotation	cite	
<body>	Defines the body element		
 	Inserts a single line break		
<button>	Defines a push button	autofocus, disabled, form, formation, formenctype, formmethod, formnovalidate, formtarget, name, type, value	
<canvas>	Defines graphics	height, width	
<caption>	Defines a table caption		
<center>	Used to center align text and content		
<cite>	Defines a citation		
<code>	Defines computer code text	autobuffer, autoplay, controls, src	
<col>	Defines attributes for table columns		
<colgroup>	Defines groups of table columns	span	
<command>	Defines a command button	checked, disabled, icon, label, radiogroup, type	
<datalist>	Defines a dropdown list		
<dd>	Defines a definition description		
<del>	Defines deleted text	cite, datetime	
<details>	Defines details of an element	open	
<dialog>	Defines a dialog (conversation)		
<dfn>	Defines a definition term		
<dir>	Used to define a directory list		
<div>	Defines a section in a document		
<dl>	Defines a definition list		
<dt>	Defines a definition term		
<em>	Defines emphasized text		
<embed>	Defines external interactive content or plugin	height, src, type, width	
<fieldset>	Defines a fieldset	disabled, form, name	
<figure>	Defines a group of media content, and their caption		
<font>	Used to define font face, font size, and font color of text		
<footer>	Defines a footer for a section or page		
<form>	Defines a form	accept-charset, action, autocomplete, enctype, method, name, novalidate, target	
<frame>	Used to define one particular window (frame) within a frameset		
<frameset>	Used to define a frameset, which organized multiple windows (frames)		
<h1> to <h6>	Defines header 1 to header 6		
<head>	Defines information about the document		
<header>	Defines a header for a section or page		
<hgroup>	Defines information about a section in a document		
<hr>	Defines a horizontal rule		
<html>	Defines an html document	manifest, xmlns	
<i>	Defines italic text		
<iframe>	Defines an inline sub window	height, name, sandbox, seamless, src, width	
<img>	Defines an image	alt, src, height, ismap, usemap, width	
<input>	Defines an input field	accept, alt, autocomplete, autofocus, checked, disabled, form, formation, formenctype, formmethod, formnovalidate, formtarget, height, list, max, maxlength, min, multiple, name, pattern, placeholder, readonly, required, size, src, step, type, value, width	
<ins>	Defines inserted text	cite, datetime	
<keygen>	Defines a generated key in a form	autofocus, challenge, disabled, form, keytype, name	
<kbd>	Defines keyboard text		
<label>	Defines an inline sub window	for, form	
<legend>	Defines a title in a fieldset		
<li>	Defines a list item	value	
<link>	Defines a resource reference	href, hreflang, media, rel, sizes, type	
<map>	Defines an image map	name	
<mark>	Defines marked text		
<menu>	Defines a menu list	label, type	
<meta>	Defines meta information	charset, content, http-equiv, name	
<meter>	Defines measurement within a predefined range	high, low, max, min, optimum, value	
<nav>	Defines navigation links		
<noframes>	Used to display text for browsers that do not handle frames		
<noscript>	Defines a noscript section		
<object>	Defines an embedded object	data, form, height, name, type, usemap, width	
<ol>	Defines an ordered list	reversed, start	
<optgroup>	Defines an option group	label, disabled	
<option>	Defines an option in a drop-down list	disabled, label, selected, value	
<output>	Defines some types of output	for, form, name	
<p>	Defines a paragraph		
<param>	Defines a parameter for an object	name, value	
<pre>	Defines preformatted text		
<progress>	Defines progress of a task of any kind	max, value	
<q>	Defines a short quotation	cite	
<rp>	Used in ruby annotations to define what to show browsers that to not support the ruby element		
<rt>	Defines explanation to ruby annotations		
<ruby>	Defines ruby annotations		
<s>, <strike>	Used to define strikethrough text.		
<samp>	Defines sample computer code		
<script>	Defines a definition list	async, type charset defer, src	
<section>	Defines a section	cite	
<select>	Defines a selectable list	autofocus, disabled, form, multiple, name, size	
<small>	Defines small text		
<source>	Defines media resources	media, src, type	
<span>	Defines a section in a document		
<strong>	Defines strong text		
<style>	Defines a style definition	type, media, scoped	
<sub>, <sup>	Defines sub/super-scripted text		
<table>	Defines a table	summary	
<tbody>	Defines a table body	summary	
<td>	Defines a table cell	colspan, headers, rowspan	
<textarea>	Defines a text area	autofocus, cols, disabled, form, maxlength, name, placeholder, readonly, required, rows, wrap	
<tfoot>, <thead>	Defines a table footer / head		
<th>	Defines a table header	colspan, headers, rowspan, scope	
<time>	Defines a date/tim	datetime	
<title>	Defines the document title		
<tr>	Defines a table row	datetime	
<tt>	Used to define teletype text		
<u>	Used to define underlined text		
<ul>	Defines an unordered list		
<var>	Defines a variable		
<video>	Defines a video	autobuffer, autoplay, controls, height, loop, src, width	

HTML5 TAG CHEAT SHEET  
Created by WebsiteSetup.org

How would you specify a `<div>`  
with the `<p>` (paragraph) I **love** HTML! ?

`<div><p>I <strong>love HTML!`

`<div><p>I <strong>love</strong> HTML!</p>`

`<div><p>I <strong>love<strong> HTML!<p><div>`

`<div><p>I <strong>love</strong> HTML!</p></div>`

`<div><p>I </p><strong>love</strong><p> HTML!</p></div>`

How would you specify a `<div>`  
with the `<p>` (paragraph) I **love** HTML! ?

`<div><p>I <strong>love HTML!`

`<div><p>I <strong>love</strong> HTML!</p>`

`<div><p>I <strong>love<strong> HTML!<p><div>`

➡ `<div><p>I <strong>love</strong> HTML!</p></div>`

`<div><p>I </p><strong>love</strong><p> HTML!</p></div>`

# Nesting

- The **Content** of a tag can contain other HTML tags

```
<div><p>I <strong>love</strong> HTML!</p></div>
```

# Let's make a shopping list

## Mark's shopping list

- Milk
- Eggs
- Sandwich ingredients:
  - Bread
  - Tomato
  - Lettuce



# Nesting: HTML

- By convention, HTML is specified via the **Content** of an `<html>` element.

```
<html>      ← Start of HTML document
  <body>     ← Start of body (visible) content
    <h1>Hello, IN4MATX 133!</h1>
    <p>HTML is <em>great</em>!</p>
  </body>    ← End of body content
</html>     ← End of HTML document
```

# Attributes

- Attributes specify options and add meaning
- Attributes are space-separated lists of names and values.
  - Kind of like variables
  - Almost always Strings

```
<div attributeA="valueA" attributeB="valueB">  
  Content goes here  
</div>
```

# Attributes

```
<a href="http://inf133-fa20.baldwin.in/">IN4MATX 133</a>
```



anchor hypertext  
(hyperlink) reference

```

```



source



alternative text for  
screen readers



img tags have no (text) content,  
so no closing tag

```
<html lang="en">
```

```
...  
</html>
```



Language of document is  
English

# HTML structure

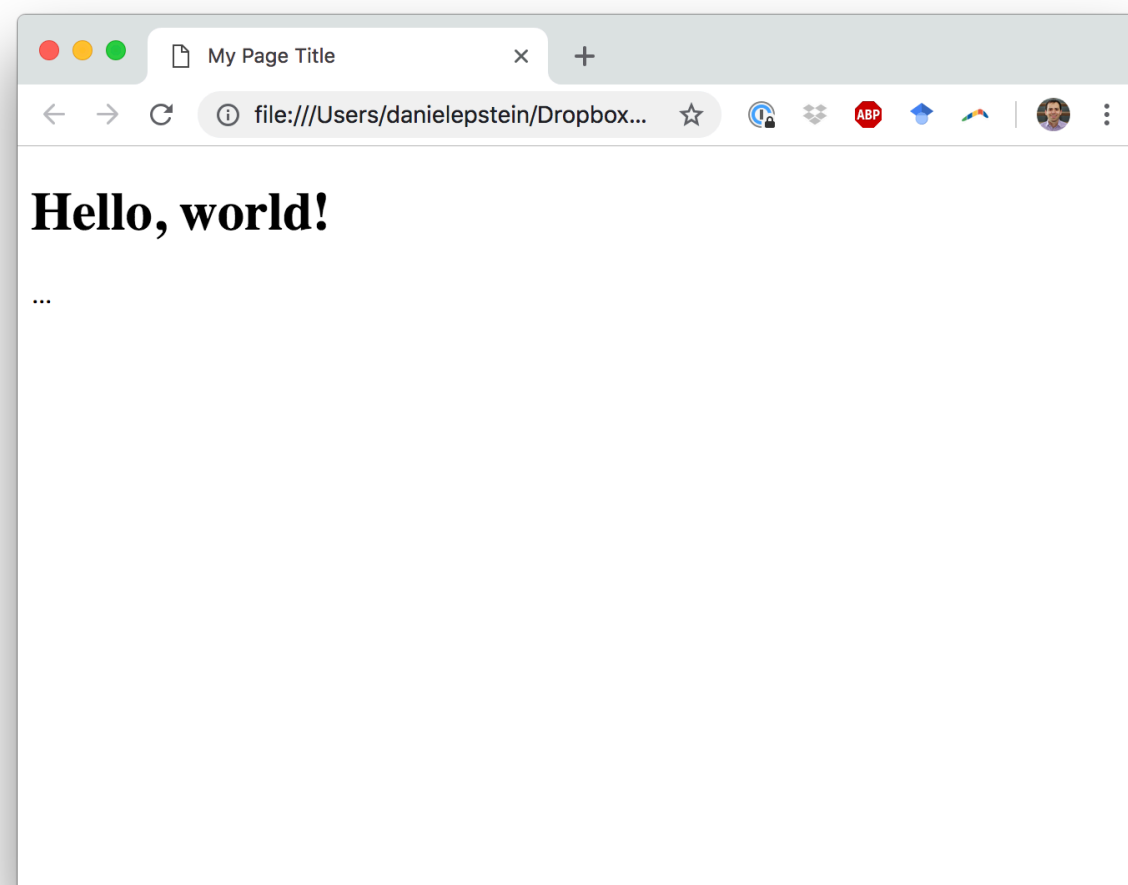
```
<!DOCTYPE html> ← Document format
<html lang="en"> ← Specify language
<head> ← Document header (content that's not shown)
  <meta charset="UTF-8"> ← Character set (for non-latin characters)
  <meta name="author" content="your name"> ← For search engines
  <title>My Webpage</title> ← Webpage title in tab
</head>
<body> ← Document body (content that's shown)
  <h1>Hello, world!</h1>
  ...
</body>
</html>
```

# HTML structure

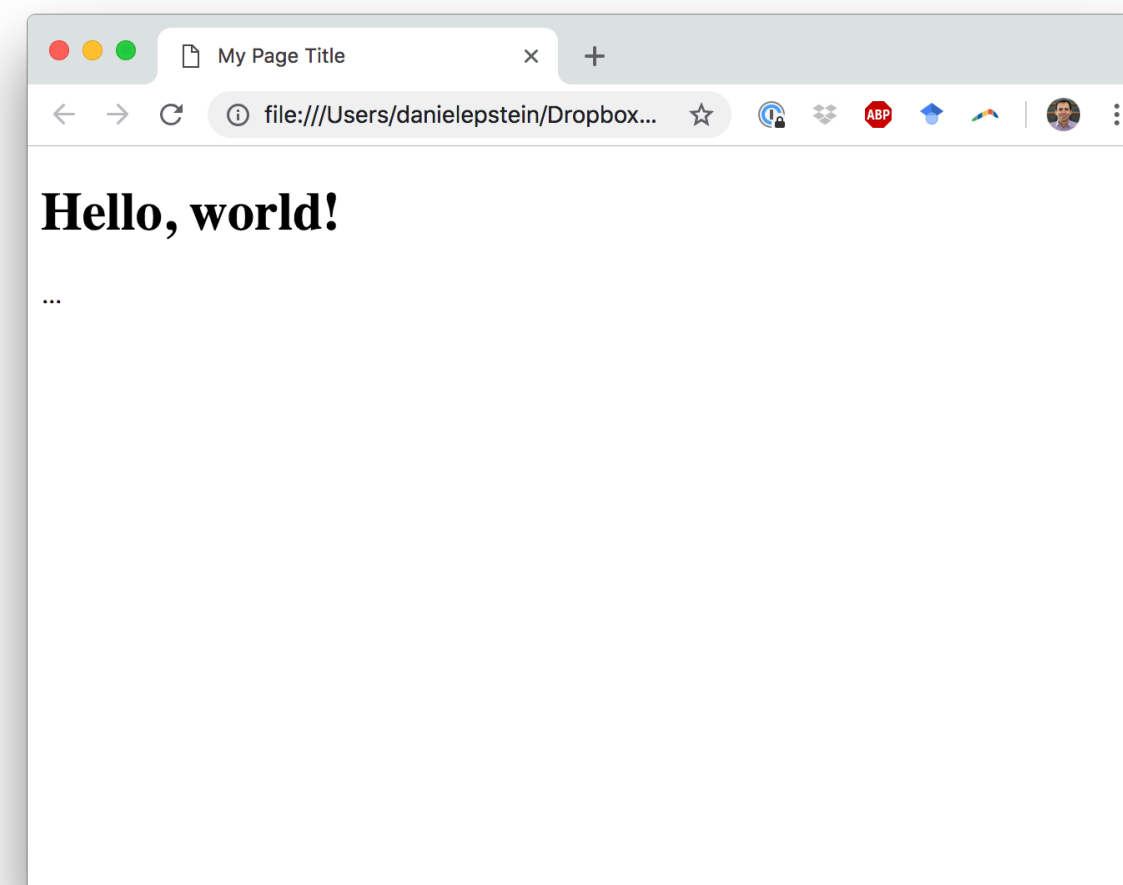
- Surprisingly, browsers are accommodating about HTML structure
- No “compiler errors”
- However, validation can help ensure browser compatibility and site usability

# HTML structure

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="author" content="your name">
  <title>My Page Title</title>
</head>
<body>
  <h1>Hello, world!</h1>
  ...
</body>
</html>
```



```
<html>
<head>
  <title>My Page Title</title>
</head>
<body>
  <h1>Hello, world!</h1>
  <p>...</p>
```



# W3C validator

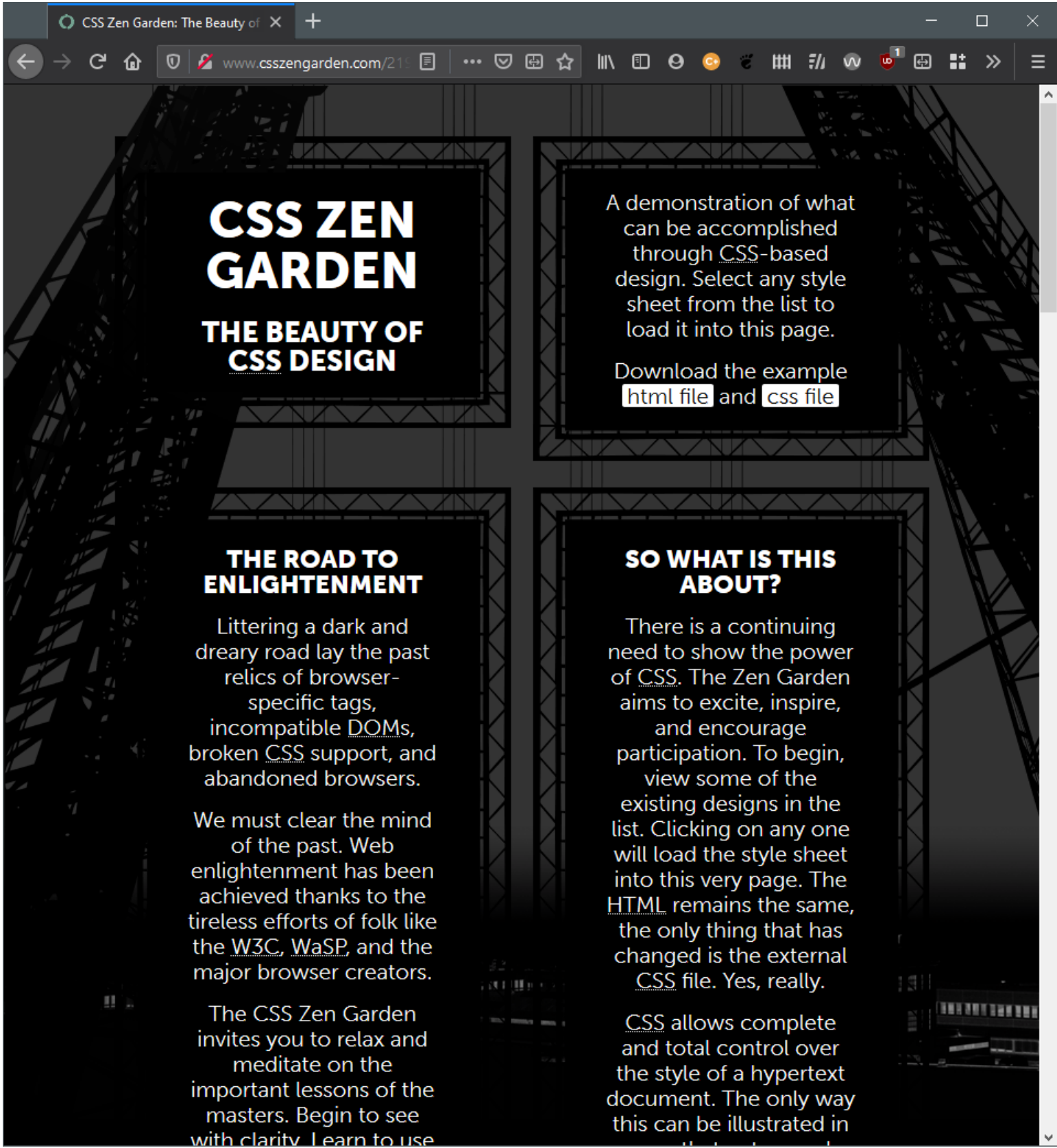
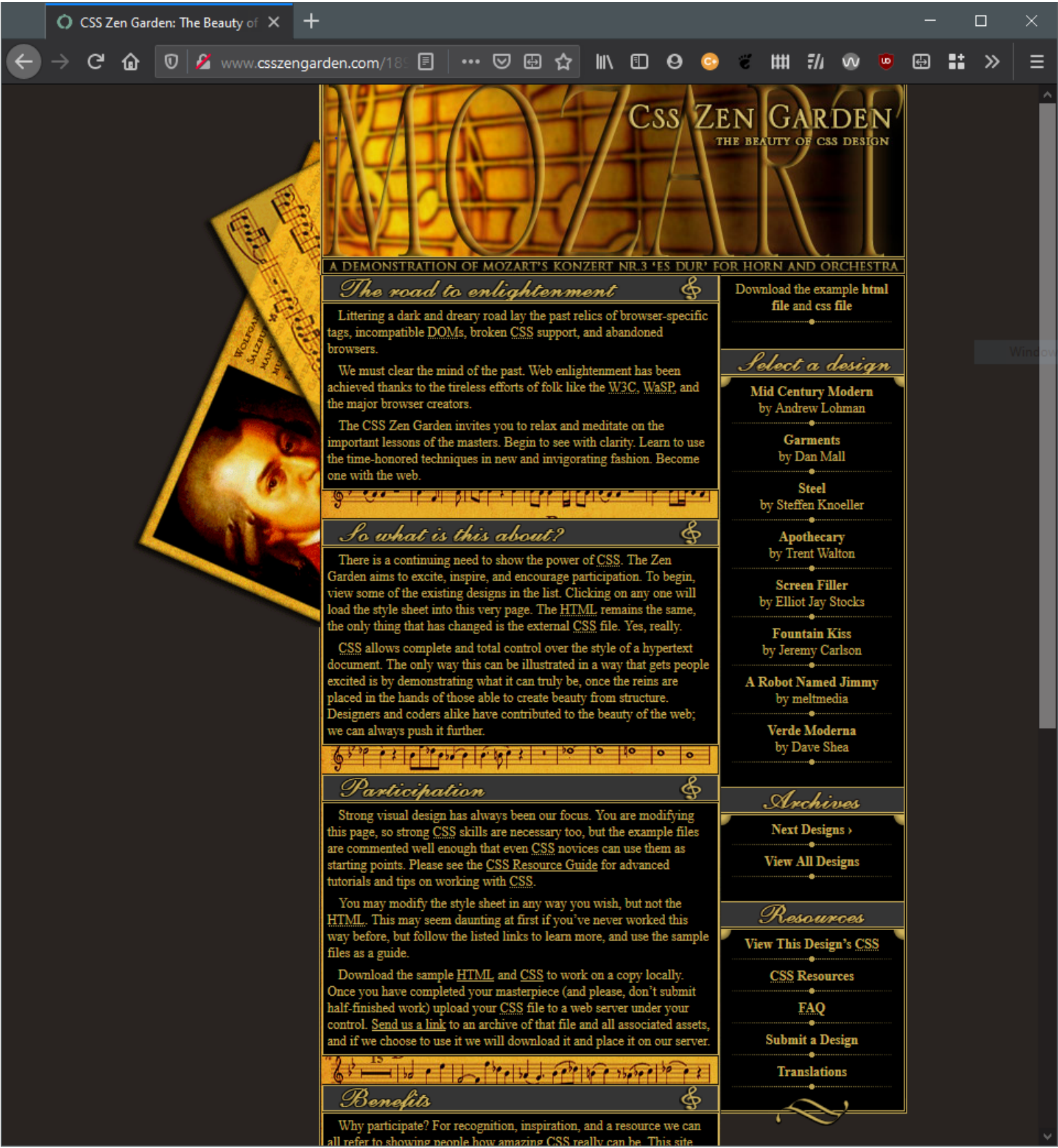
<https://validator.w3.org/>



<https://repl.it/@m5b/inf133-html-demo>



# Same page, different stylesheets



<http://www.csszengarden.com/>

# CSS

## Cascading Style Sheets

- Defines rules for styling
- Differs from HTML, which provides structure for the document

# CSS: but why?

- Reusability
  - Apply the same style to multiple web pages
- Modularity
  - Include multiple stylesheets that apply to a single page
- Sane management
  - Files can be version controlled, separate from HTML structural content
- Maintainability
  - Styles can be contained in a single type of location (style sheets)

Ok, so how do I write CSS?

# CSS syntax

- **Selectors** specify which elements a **rule** applies to
- **Rules** specify what *values* to assign to different formatting **properties**

```
/* CSS Pseudocode */
```

```
selector {  
  property: value;  
  property: value;  
  ...  
}
```

← One rule, many properties



# CSS syntax

h1 { ← Apply to all h1 tags  
font-family: 'Arial'; ← "font"  
color: blue;  
background-color: #ff0000; /\*red\*/  
}

- Link to stylesheets in HTML's <head>

<head>

<link rel="stylesheet" href="my-style.css">

</head>



relation between  
this page and reference



no content,  
so no closing tag

# Element, ID, and Class selectors

- element: what tag is being styled

```
p {  
    font-family: 'Arial';  
    color: red;  
}
```

- class: a type of element

```
.emphasize {  
    font-family: 'Arial';  
    color: red;  
}
```

- id: one specific element

```
#redtext {  
    font-family: 'Arial';  
    color: red;  
}
```

# HTML Class and ID attributes

```
<div class="widget foo" id="baz"></div>
```

- Variable-value just like any other attribute (`href`, `src`)
- An element can have many classes, only one ID
- Each page can have only one element with a given ID
  - Required to pass validation
- Can use the same class on multiple elements
  - And should; it's useful to apply the same style to many elements



# HTML Class and ID attributes

```
<div class="widget foo" id="baz"></div>
```

```
div.widget.foo#baz {  
  /*can chain selectors together!*/  
}
```

# CSS properties

- `font-family`: the “font” (fallback alternatives separated by commas)
- `font-size`: the size of the text
- `font-weight`: boldness
- `color`: text color
- `background-color` (element’s background)
- `opacity` (transparency)
- And much, much more!

# HTML vs. CSS

- HTML specifies the *semantics*
- CSS specifies the *appearance*

# HTML vs. CSS

```
<!--HTML-->  
<p> This text is <em>emphasized!</em></p>
```

```
<!--HTML-->  
<p> This text is also  
<i>emphasized!</i></p>
```



Conflates appearance and semantics



Says nothing about appearance

This text is *emphasized*

This text is also *emphasized*

# Cascading Style Sheets

- Multiple rules can apply to the same element (in a “cascade”)

```
<!--HTML-->
```

```
<p class="big blue">
```

This tag has two classes: "big" and "blue"  
(classes are separated by spaces)

```
</p>
```

```
/* CSS */
```

p { font-family: 'Verdana'; }	←	Apply to all <p> tags
.big { font-size: larger; }	←	Apply to all with class="big"
.blue { color: blue; }	←	Apply to all with class="blue"

# Cascading Style Sheets

- CSS rules are also inherited from parent tags

```
<div class="content"> <!-- has own styling -->
  <div class="sub-div"> <!-- has own styling + .content styling -->
    <ol class="my-list"> <!-- own styling + .sub-div + .content -->
      <!-- own style is ol AND .my-list rules-->
        <!-- li styling + .my-list + .sub-div + .content -->
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ol>
  </div>
</div>
```

# Cascading Style Sheets

- Rules are applied **in order** (last rule always wins among peer selectors)

```
<!--HTML-->
```

```
<p class="red green">
```

```
  <em class="blue">Text is blue!</em>
```

```
</p>
```

```
/* CSS */
```

```
p { font-family: 'Verdana'; }
```

```
.red { color: red; }
```

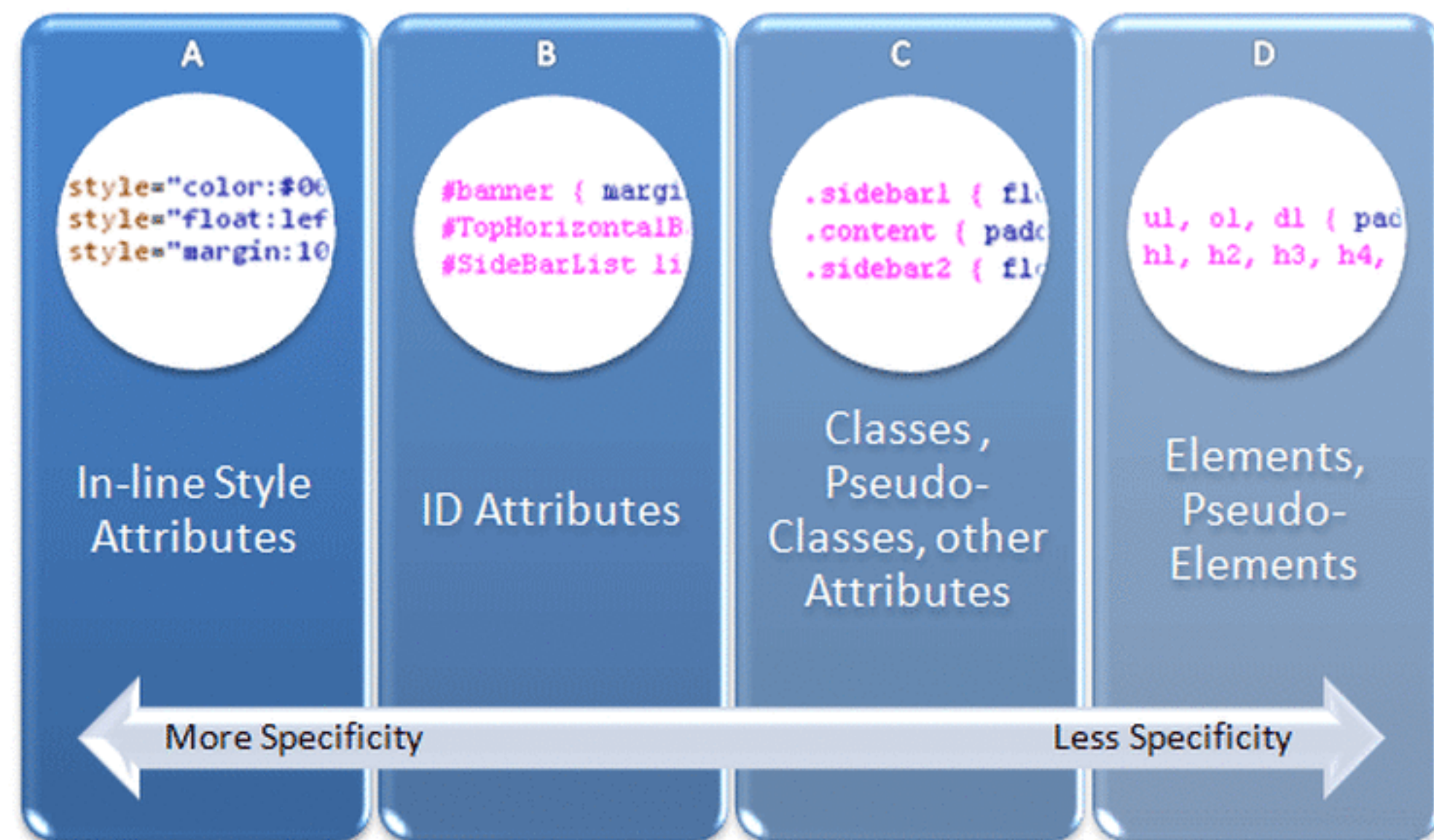
```
.green { color: green; }
```

```
.blue { color: blue; }
```



# Specifying styles

- CSS specificity is *calculated* based on which selector designates it
- General rule: rule that's “closer to the HTML element” applies
- This is difficult stuff, usually trial-and-error resolves most things





# Positioning

- HTML tags are either\*:
  - **Block** elements (line break after them)
  - **Inline** elements (no line break)

`<p>` ← Block

This is on a line.

`<em>`This is on the same line.`</em>` ← Inline

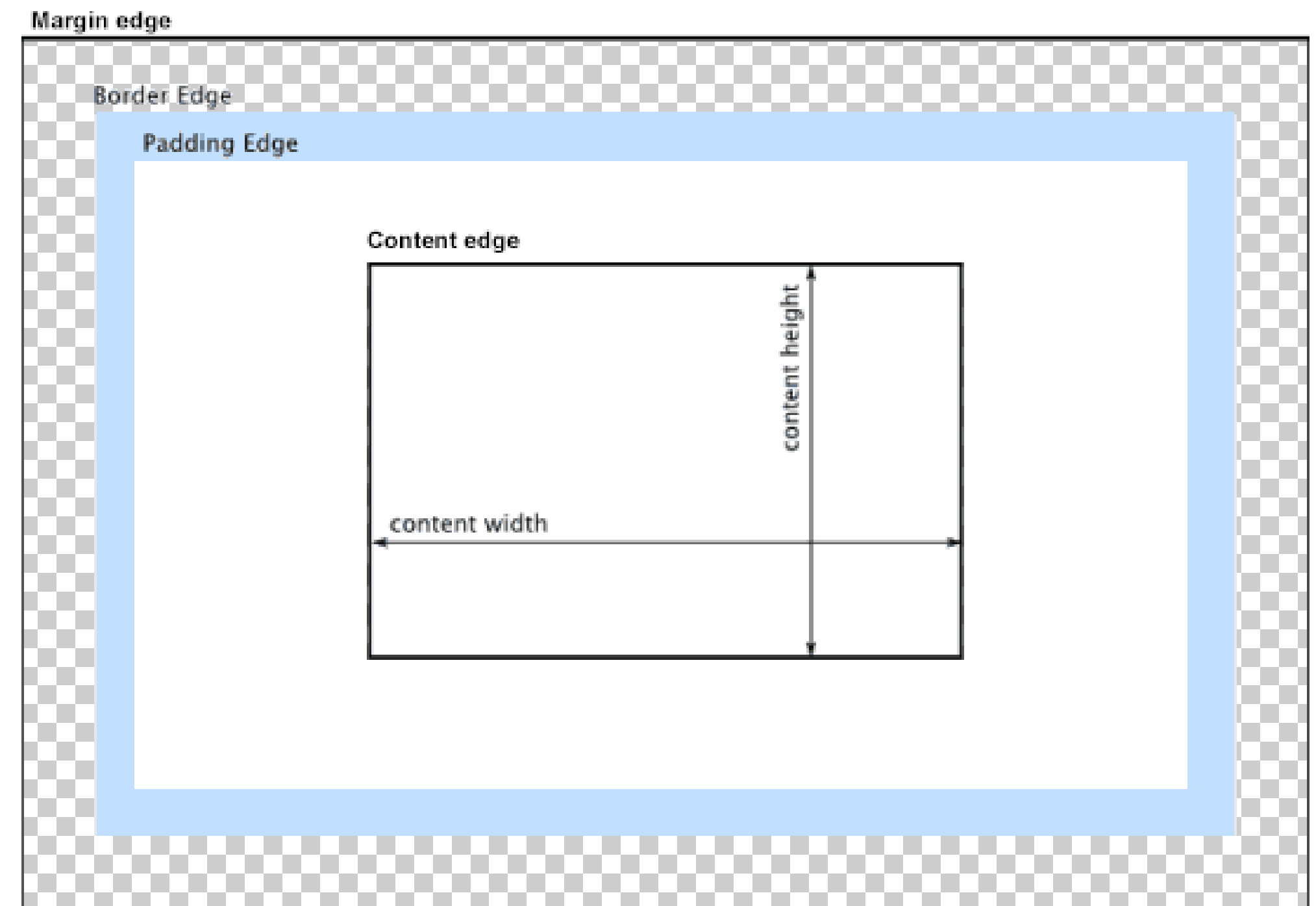
`</p>`

`<p>`This will be on a new line.`</p>`

- Don't put block elements inside inline elements!

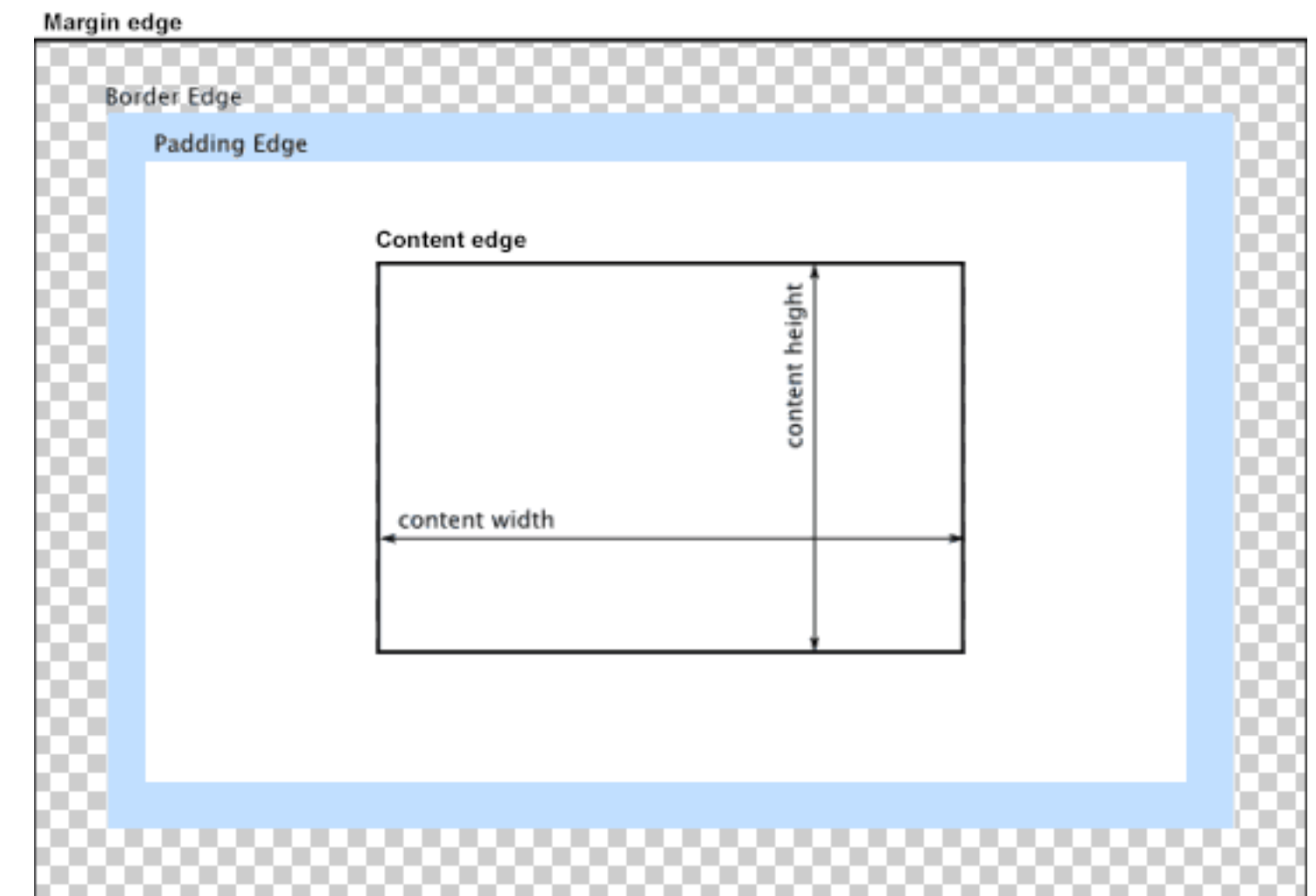
# Positioning: box model

- **Content** contains “real” content
- **Padding** extends content area
- **Border** is similar
- **Margin** is intended to separate elements from neighbors



# Positioning: box model

- Content dimensions are specified **with** `width` and `height`
- padding, border, **and** margin **have** **direction** properties (e.g., `padding-top`, `margin-right`, `border-left`)
- border **can** have `border-color`, `border-width`, **and** `border-style`
- Content color (e.g., `background-color`) extends into padding

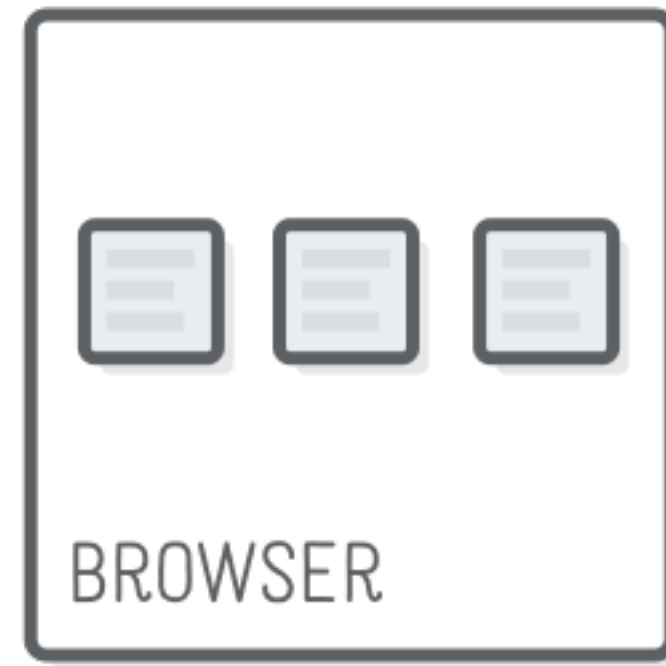


# Positioning

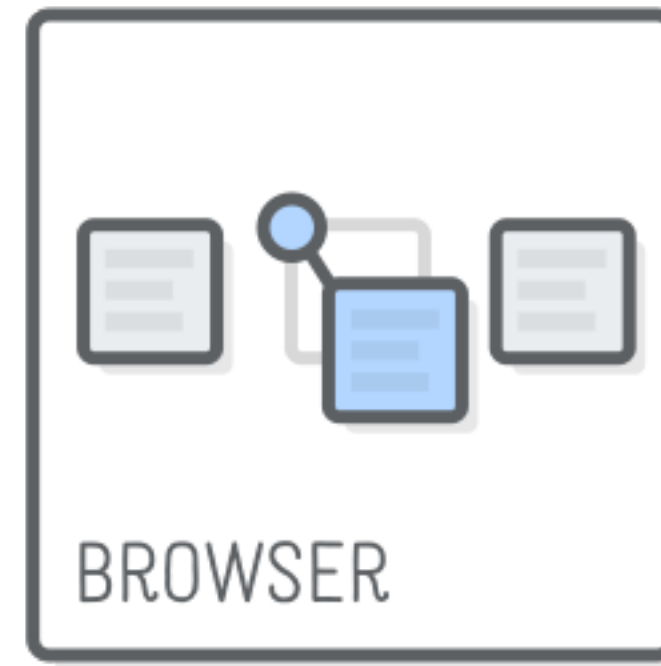
- All positioning is relative to the parent
  - If you nest tags, the child's margins, etc. are all dependent on parent's

# Positioning: types

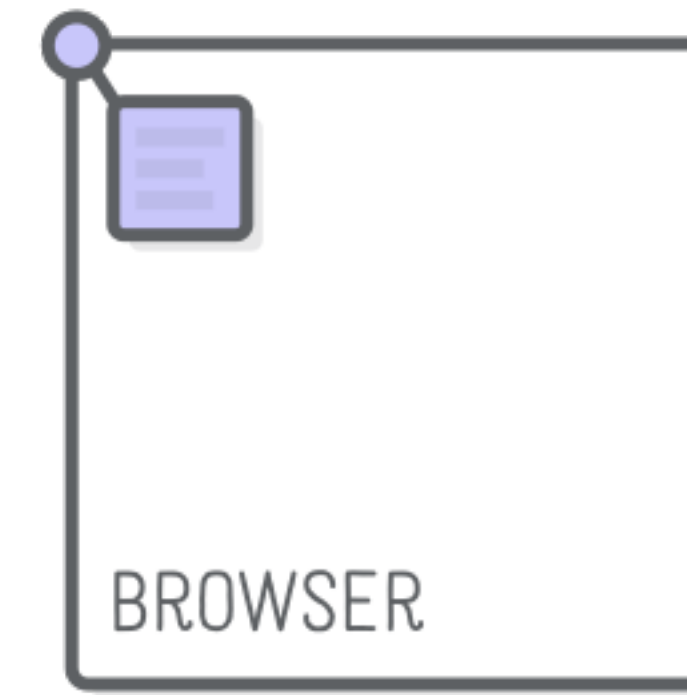
STATIC



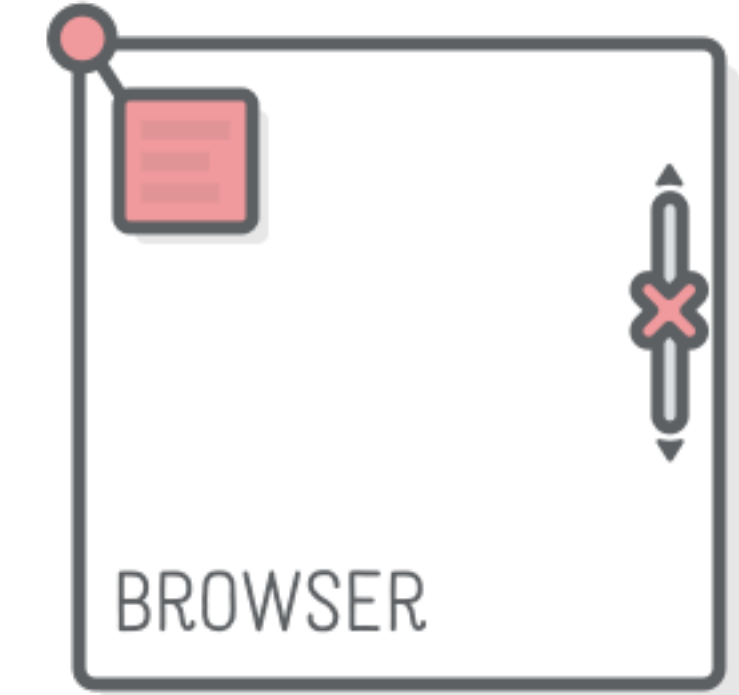
RELATIVE



ABSOLUTE



FIXED



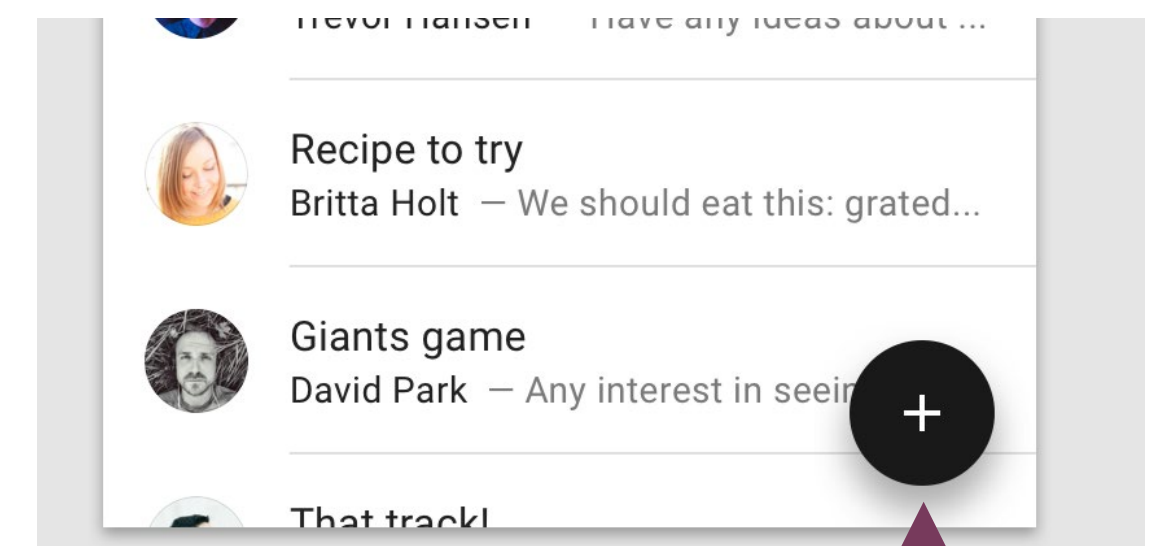
- `static` (default)
- `relative` (offset from default)
- `absolute` (from top-left)
- `fixed` (absolute + floating, fixed to the viewport)

# Positioning: types

- `static` and `relative` follow the overall flow of a page
  - `relative` helps make adjustments to the flow
- `absolute` and `fixed` ignore it entirely
  - But they're helpful in some cases, like floating action buttons (FABs)



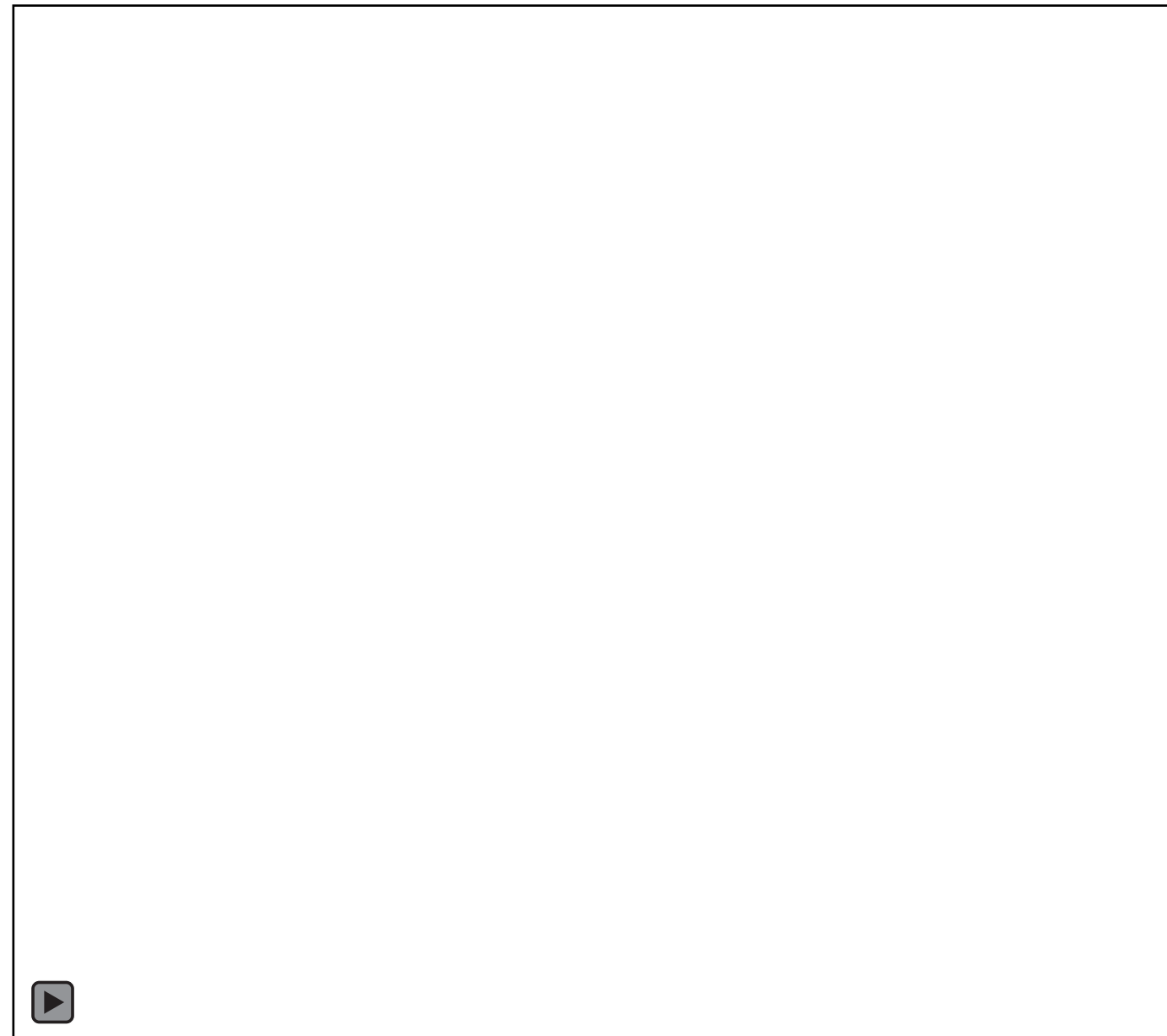
Relative position



Absolute position

# Positioning: types

- `sticky` will stop when a user scrolls past it
  - Useful for menus
  - Not all browsers support it, but getting there



# Units

- Pixels (px), element units (em), percentages (%), real-world units (in, cm)
- Use relative units (em, %) whenever possible
- Helps accessibility, people with low vision change default size (usually 16px)
  - Em fonts scale from the default, a 30px heading stays 30px
- Also useful to vary based on screen size
  - More on how to do that next lecture

	Recommended	Occasional use	Not recommended
Screen	em, px, %	ex	pt, cm, mm, in, pc
Print	em, cm, mm, in, pt, pc, %	px, ex	



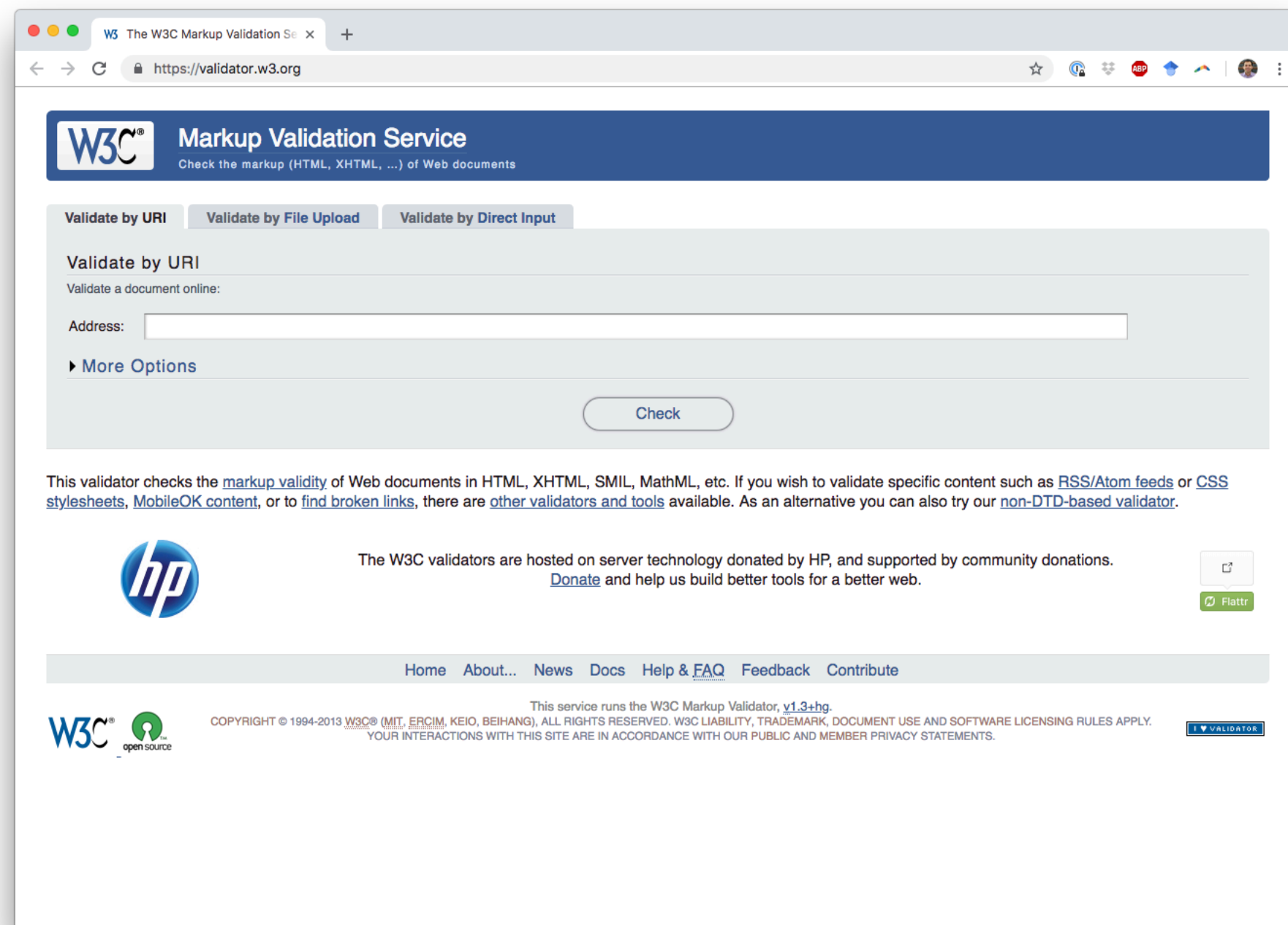


<https://repl.it/@m5b/inf133-css-demo>

# References

- <https://www.w3schools.com/cssref/>
- <https://cssreference.io/>
- <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
- <https://www.codecademy.com/learn/learn-css>

# Validation



<https://validator.w3.org>

# Today's goals

**By the end of today, you should be able to...**

- Describe the fundamentals of web communication
- Identify the syntax of HTML tags and attributes and describe their roles
- Create a HTML template which follows W3C specifications
- Explain the goals of CSS and why it exists as separate from HTML
- Describe the CSS hierarchy and fallback structure