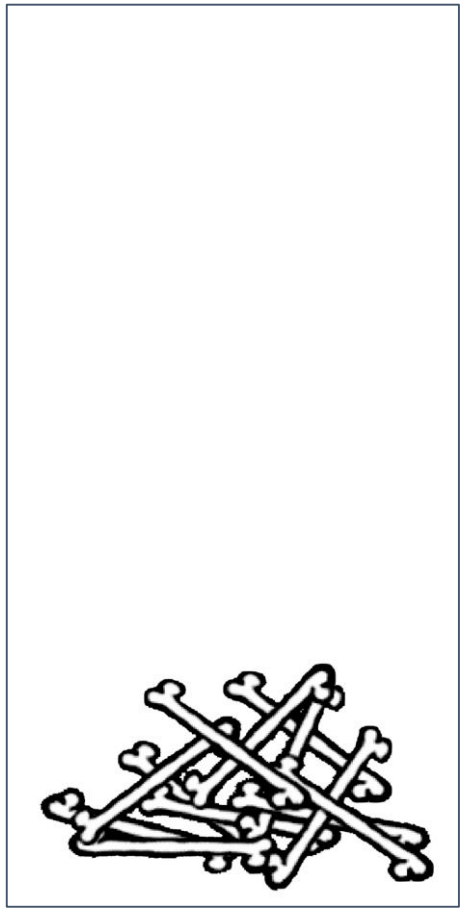
Lecture 1b

Internet, WWW, HTML & CSS basics

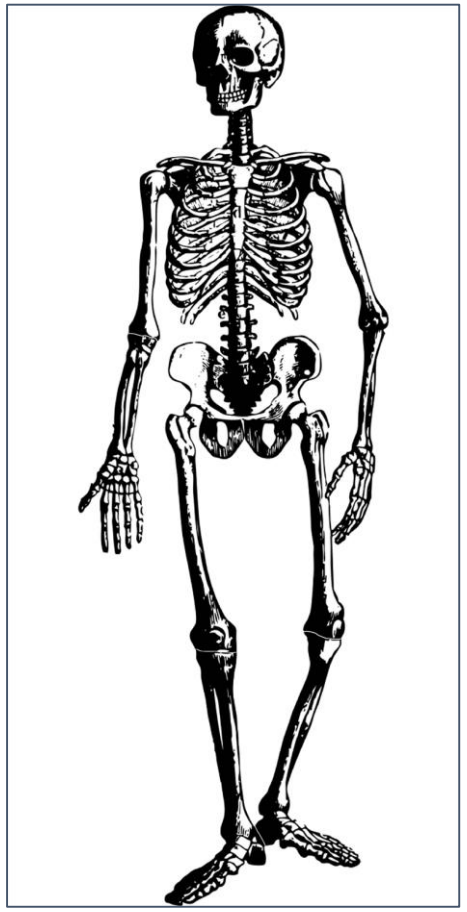
List of topics

- Browser, Internet, WWW, The Web, HTTP
- Static website & Dynamic website
- HTML
- CSS basics
- Layout with Box Model

So What is a Web Page?



CONTENT



STRUCTURE

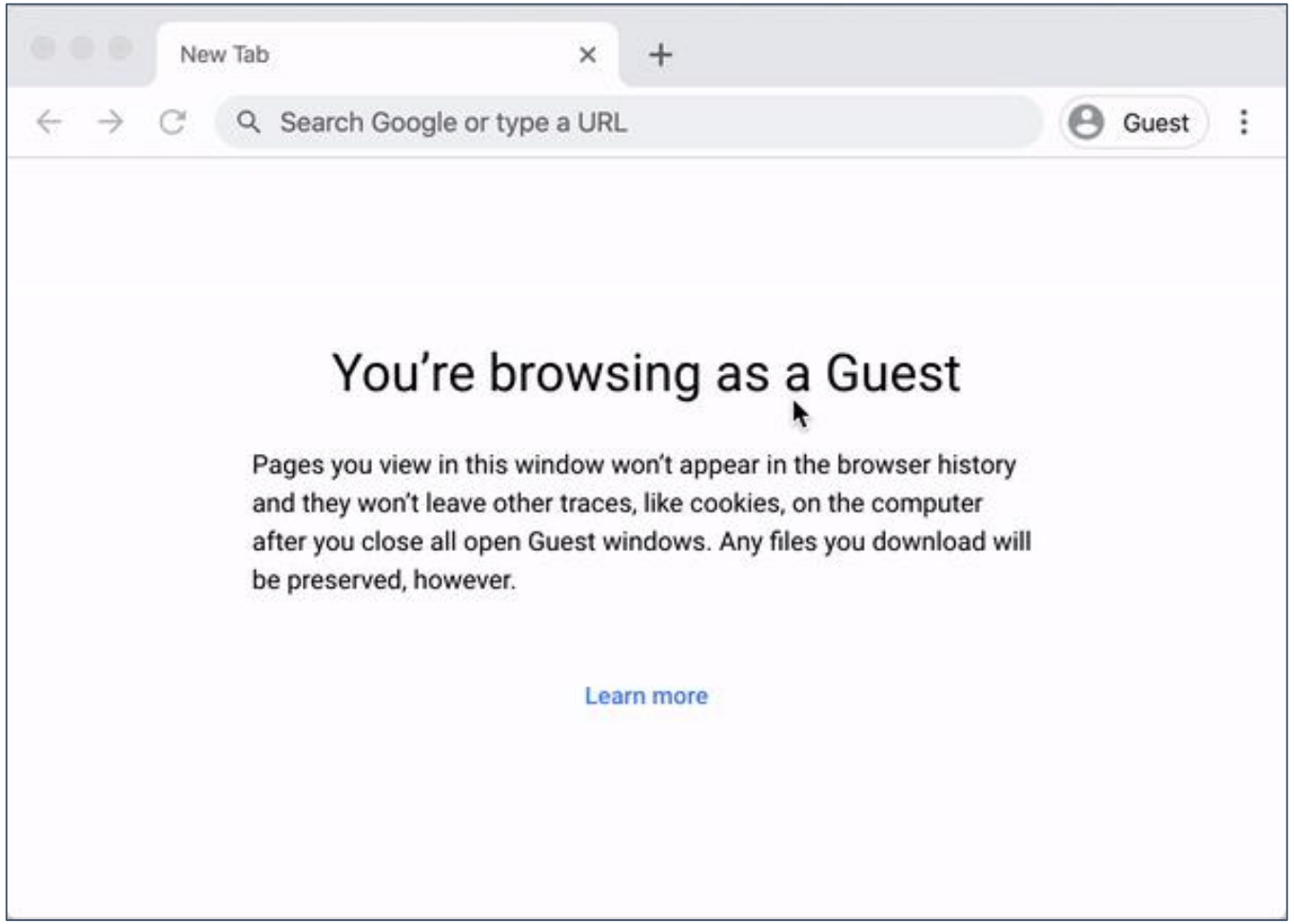
STYLE

BEHAVIOR

What's everything involved here?

1. Decide on URL...
2. Type it in...
3. Hit enter...
4. Website loads!

What happens between 3 and 4?

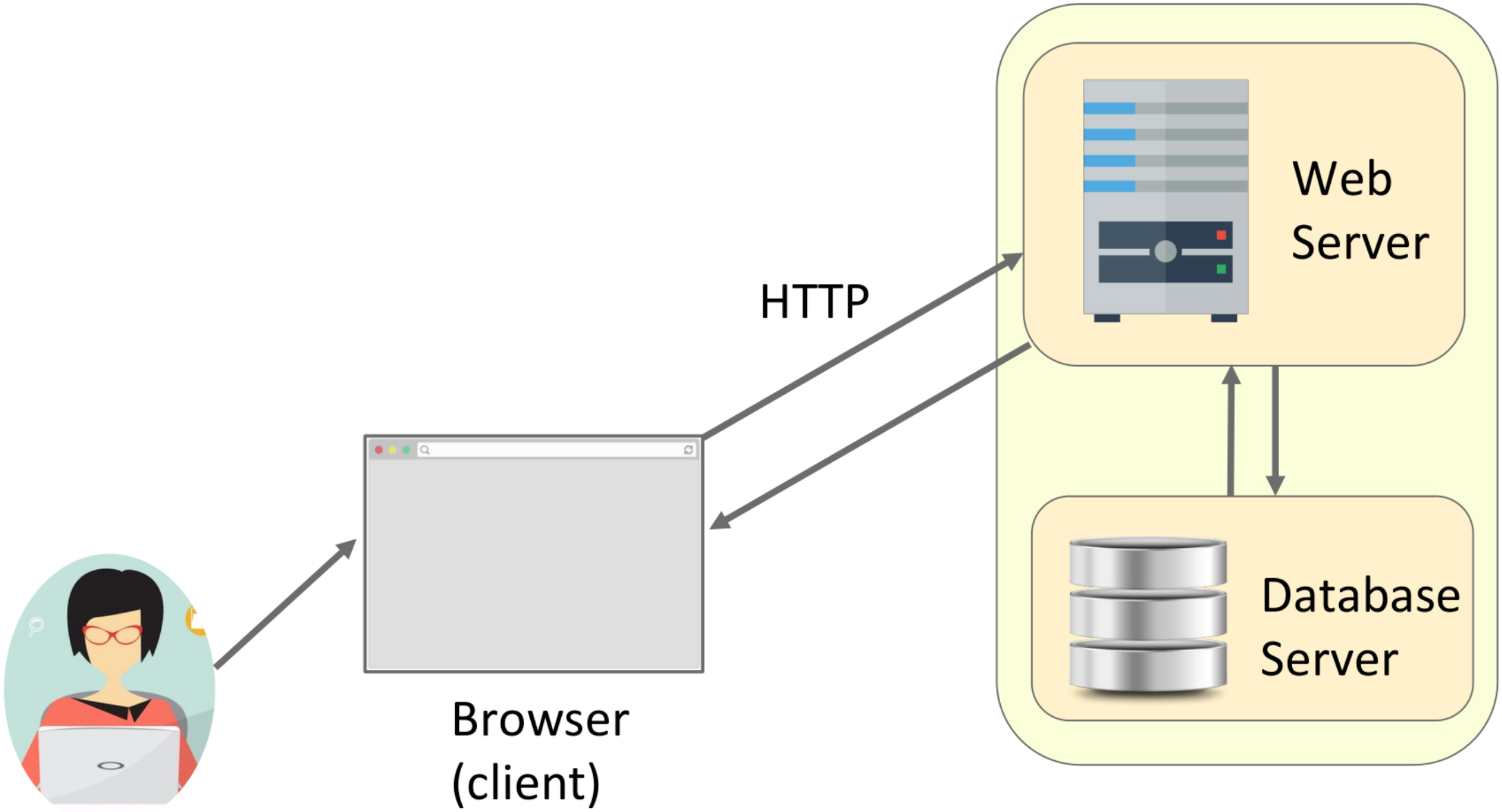


What's everything involved here?

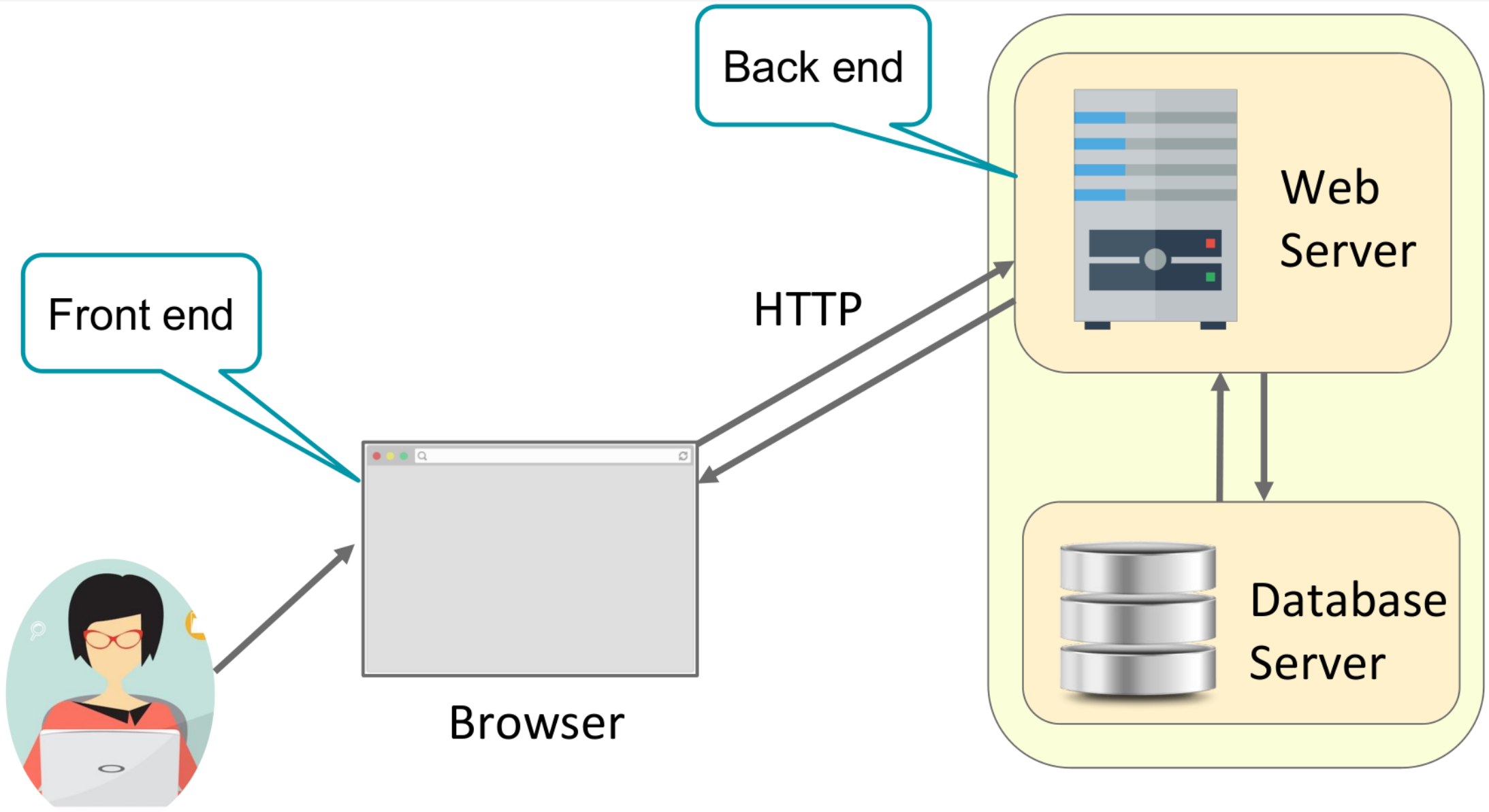
You don't have google.com on your computer. So, where does it come from?

1. DNS - Figure out where it is
2. HTTP Request - Ask for it to be sent to us
3. Browser - Check and verify what we get
4. Browser - Show it

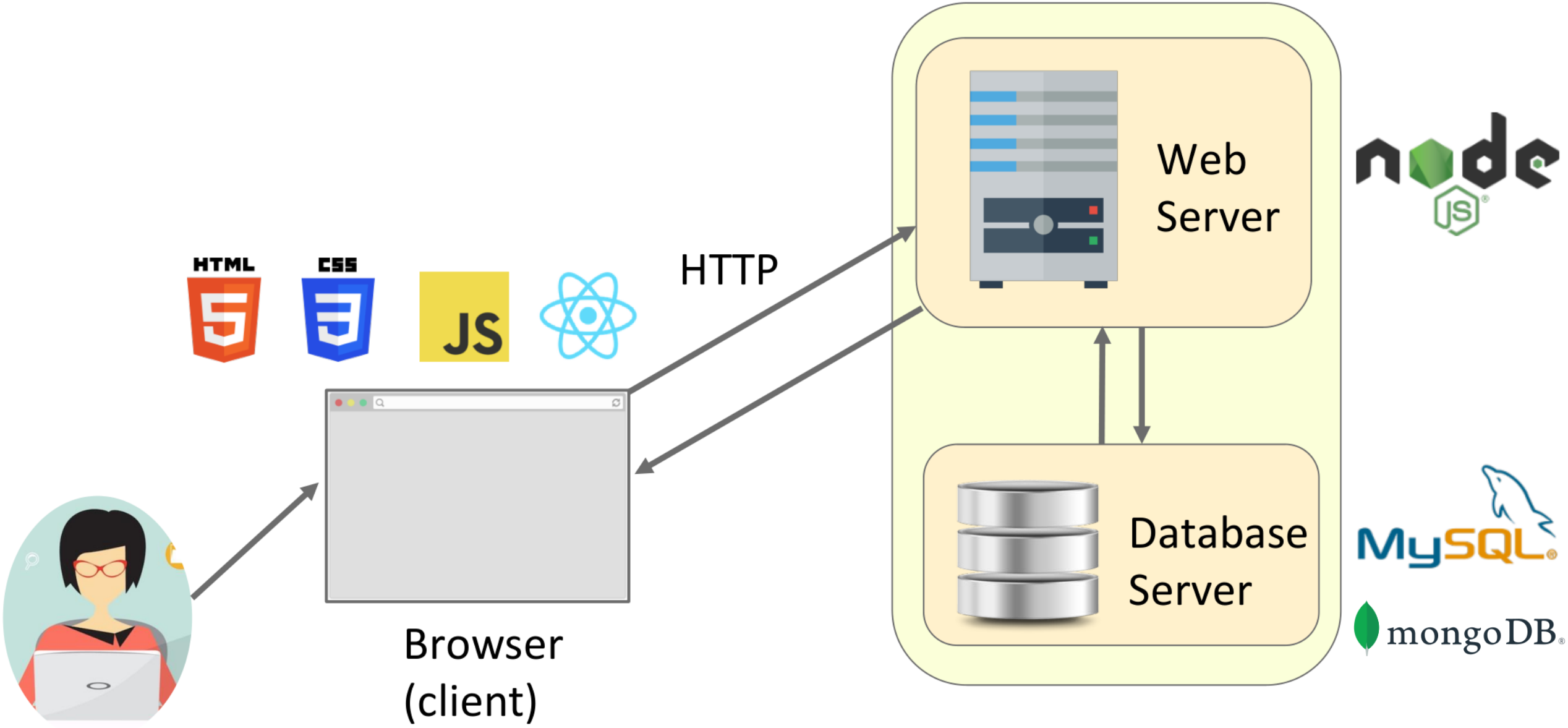
The web's general architecture



Front-end vs Back-end



This course's technology stack



The thing in the address bar. Where is the website?

URL (Uniform Resource Locator) is an identifier for the location of a document.

An example of basic URL:

The Internet

A connection of computer networks built on the Internet Protocol (IP).

Every computer has an "IP" address:

- E.g.,: 34.215.139.216
- Find yours with WhatsMyIP.org

So what's the difference between the Internet and the World Wide Web (WWW)?

Internet VS. 'The Web'

INTERNET

Computers (servers) connected to each other via a series of networks

Powered by layers upon layers:

- Physical: The cables between them
- Data & Network: The [small] packets of information
- Transport (TCP/IP): Providing connections and reliability
- Application: Tying everything together to be useful

THE WEB

- Collection of pages of information
- Text... but with some "Hyper" around it
- Pages can link to each other
- Pages have style and interactivity

Web Browser

- A software application that allows users to access and view web content.
- Examples of web content include websites, images, and videos.

Remember that URL? (https://google.com/)

Need to go out to the Internet to get the webpage.
Internet is low-level: based on numbers (IP addresses), not names.

Domain Name System (DNS)
A Domain Name System translates human-readable names to IP addresses

- Example: fit.hanu.vn → 103.238.69.140
  - Hostname of fit.hanu.vn (which we might put into the browser's address bar)
  - ... has IP address of 103.238.69.140 (which will be used to contact the server via the internet)

How to find out a host's IP address?

- ping, nslookup, online tools

What’s more about the URL?

`https://fit.hanu.vn/mod/book/view.php?id=12`

We've handled the host to IP address (so we know who to ask for the web page)

The "protocol" tells us *how*:

- HTTP: HyperText Transfer Protocol
- Gives us the instructions (protocols) for how to share (transfer) web content ("hypertext")

And the rest tells us *what*:

- From the `fit.hanu.vn` server (aka **host**)...
- I'd like the thing called `mod/book/view.php?id=12`... (aka **path** or resource)

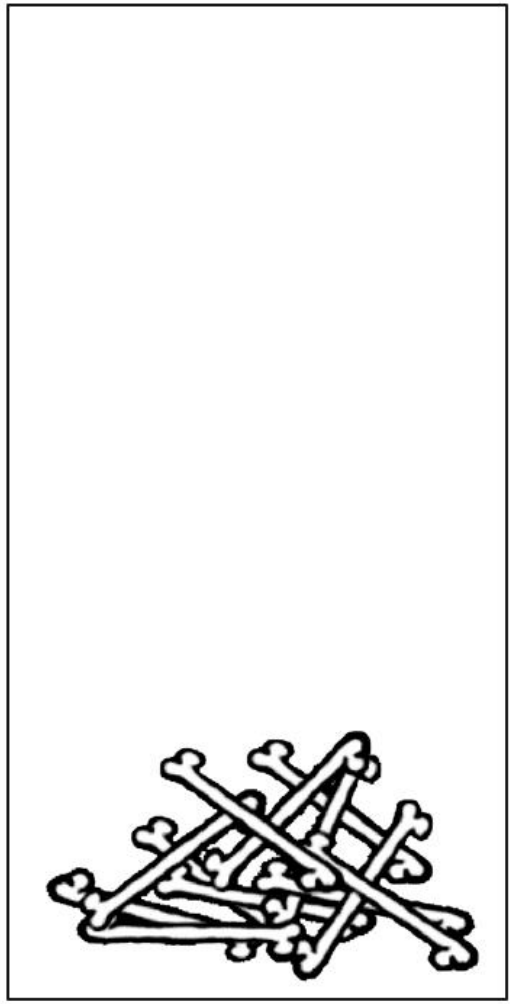
HTTP codes

Within all responses that a web server sends, there is a response code which signifies the status of the response

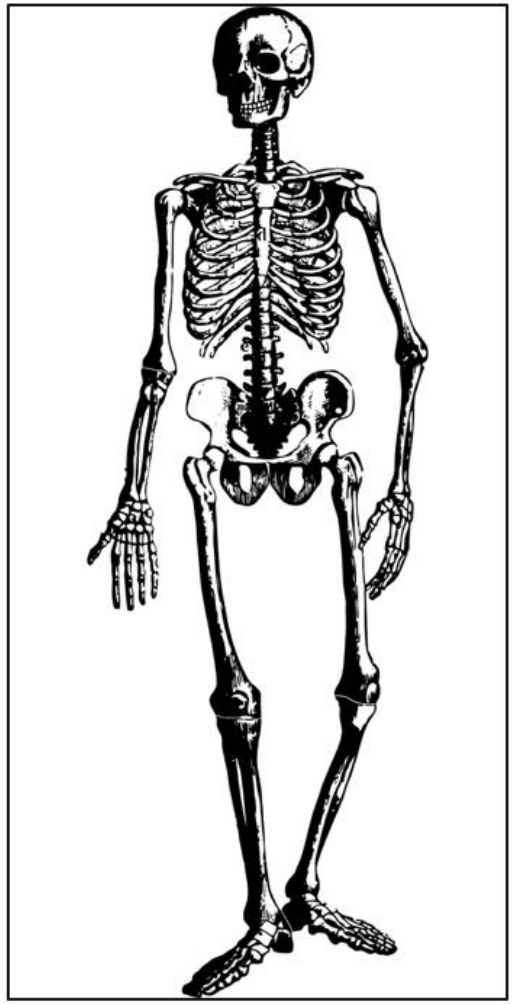
Common Codes:

Number	Meaning
200	OK
301-303	Page has moved (permanently or temporarily)
403	You are forbidden to access this page
404	Page not found
500	Internal server error

What’s in a web page?



WORDS + IMAGES



HTML

CSS

JAVASCRIPT

What’s in a web page

Hypertext Markup Language (HTML): semantic markup for web page content

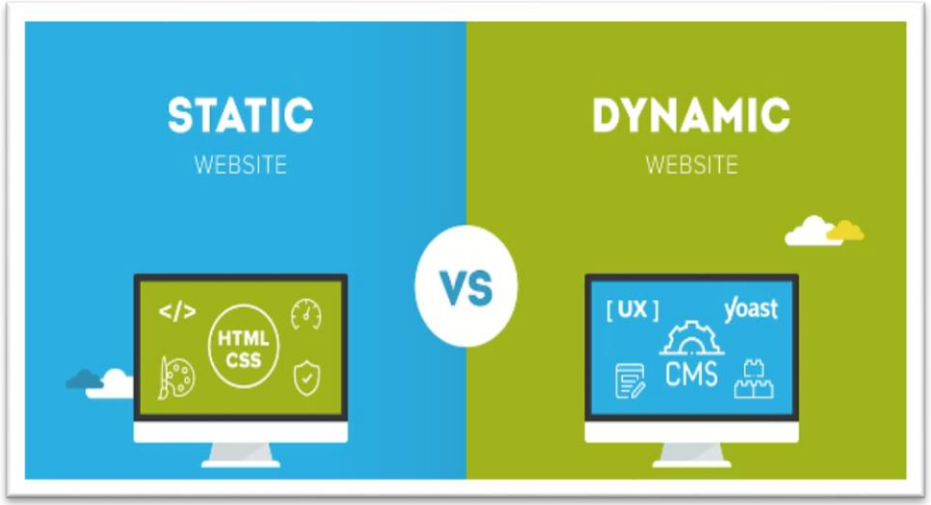
Cascading Style Sheets (CSS): styling web pages

Client-side Javascript: adding programmable interactivity to web pages

Asynchronous Javascript and XML: fetching data from web services using JavaScript fetch API

JavaScript Object Notation (JSON): file format for organizing human readable data

Static Website & Dynamic Website



1

Static Website

A website with content that does not change unless the code is manually updated. Typically built using HTML, CSS, and JavaScript.

2

Dynamic Website

A website with content that is generated dynamically, often based on user input or data from a database. Typically built using server-side technologies like PHP, ASP.NET, or Node.js.

HTML

Hyper-

prefix: over/above/beyond

Text

noun: words and/or alphanumeric characters

Markup

noun: the result of preparing text or indicating the relationship between parts of text before displaying

Language

noun: a system of symbols used to communicate ideas

In other words… HTML is the set of symbols used to give text (and images and other content) *additional meaning*.

Hypertext Markup Language (HTML)

- Describes the content and structure of information on a web page
- Not the same as the presentation (appearance on screen)
- Surrounds text content with opening and closing tags
- Most whitespace is insignificant in HTML (ignored or collapsed to a single space)
- We will use a newer version called HTML5

```
                 element
SYNTAX: <tag>content</tag>

EXAMPLE: <p>This is a paragraph</p>
```

Structure of an HTML page

An HTML page is saved into a file ending with extension `.html`

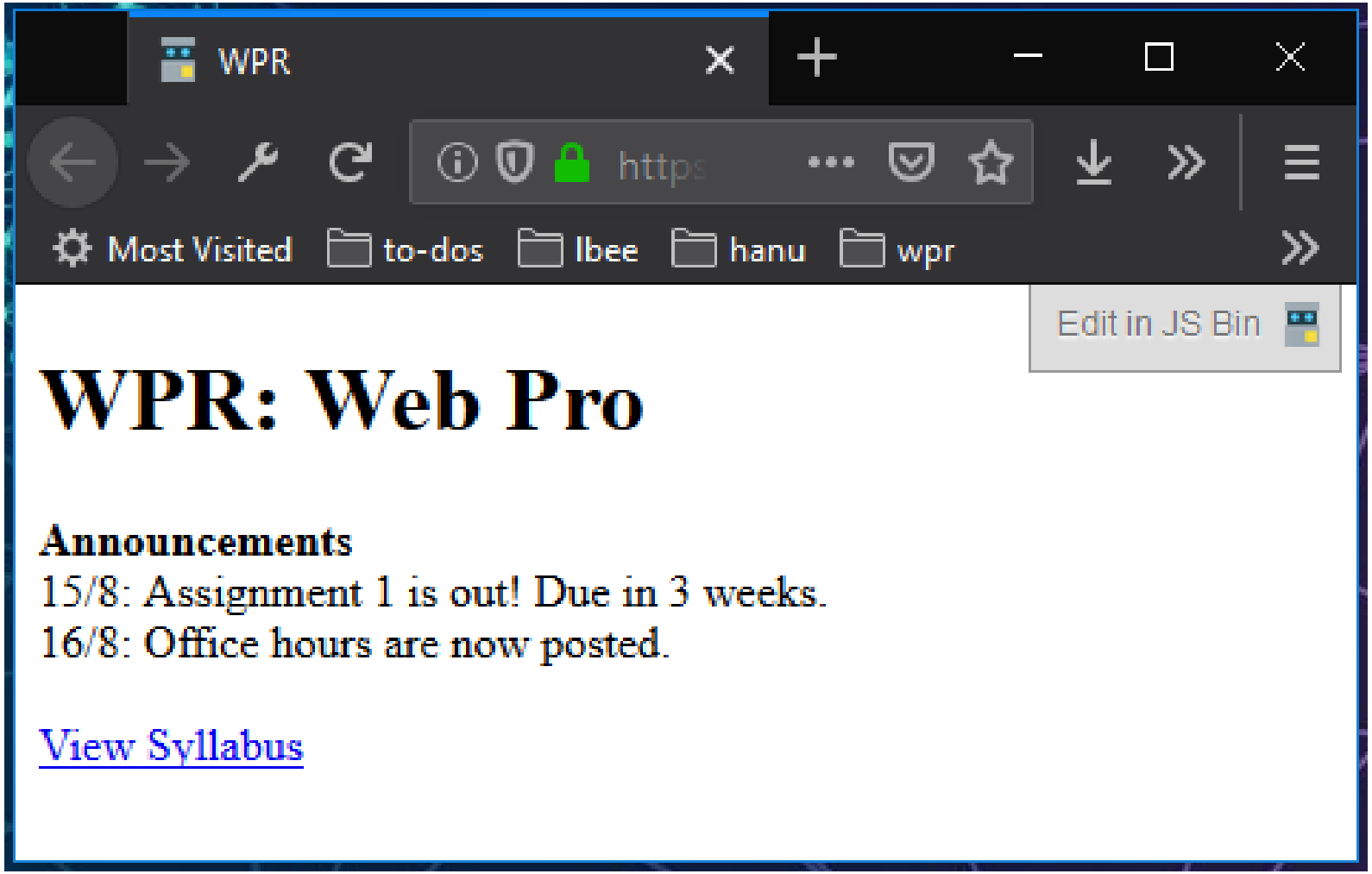
The `<head>` tag describes the page and the `<body>` tag contains the page's content

The `DOCTYPE` tag tells the browser to interpret our page's code as HTML5, the latest/greatest version of the language

```
<!DOCTYPE html>
<html>
  <head>
    information about the page
  </head>
  <body>
    page contents
  </body>
</html>
```

Exercise: Course web page

Let's write some HTML to make the following page:



Exercise: Course web page

Let's write some HTML to make the following page:

HTML Boilerplate

```
<!DOCTYPE html>
<html>

<head>
    <title>WPR Pro</title>
</head>

<body>
    ...
</body>

</html>
```

Plaintext content of the page

WPR: Web Pro
Announcements
15/8: Assignment 1 is out!
Due in 3 weeks.
16/8: Office hours are now posted.
View Syllabus

Exercise Solution

```html
<!DOCTYPE html>
<html>
<head>
    <title>WPR</title>
</head>

<body>
    <h1>WPR: Web Pro</h1>
    <strong>Announcements</strong><br />
    15/8: Assignment 1 is out! Due in 3 weeks.<br />
    16/8: Office hours are now posted.<br />
    <br />
    <a href="http://fit.hanu.vn/mod/resource/view.php?id=5778">
        View Syllabus
    </a>
</body>
</html>
```

Some points are worth noting from HTML code

(1) Multiple continuous whitespaces in HTML code collapse into one space…

(2) The <h1> heading was on a line of its own, and was not.

Block and Inline Elements

Block elements contain an entire large region of content

- Examples: paragraphs, lists, table cells
- Block elements typically take up an entire line (i.e., insert a line break).
- The browser also (usually) places a margin of whitespace between block elements
- Can be overridden with CSS

Inline elements affect a small amount of content

- Examples: bold text, code fragments, images, anchors (aka "links")
- The browser allows many inline elements to appear on the same line
- *Must* be nested inside a block element
- *Only* take up as much space as the content inside of them
- Can also be overridden with CSS

Block and Inline Elements: example

```
<em>text</em>
<em>text</em>
<em>text</em>
<p>text</p>
<p>text</p>
<p>text</p>
```

code

text text text

text

text

text

output

Rules and Exceptions

Block vs. inline:

- Some block elements can contain only other block elements: `<body>`, `<form>`
- `<p>` tags can contain only inline elements and plain text
- Some block elements can contain either: `<div>`, `<li>`

Some elements are only allowed to contain certain other elements

- `<ul>` is only allowed to contain `<li>` (but <li> can contain <ul> for nested lists!)

Some elements are only allowed once per document:

- `<html>`, `<body>`, `<head>`, `<main>`

Nesting Tags

Tags can “nest” inside of other tags

```
<body>
  <p>
    This is a <em>really, <strong>REALLY</strong></em> awesome paragraph.
    And here's a neat list:
  </p>
  <ol>
    <li>with one list item</li>
    <li>with another list item</li>
  </ol>
</body>
```

code

This is a *really*, ***REALLY*** awesome paragraph. And here's a neat list...

1. with one list item
2. and another list item!

output

Nested Lists

A list can contain another list:

```
<ul>
  <li>Koalas are marsupials</li>
  <li>Koalas like to eat Eucalyptus plants
    <ul>
      <li>
        They take up to 100 hours to digest
their food!
      </li>
    </ul>
  </li>
  <li>
    The latin name for koalas is
    <em>Phascolarctos cinereus</em>
    ("ash-colored pouch bear")
  </li>
</ul>
```

Fast Facts

- Koalas are marsupials
- Koalas like to eat Eucalyptus plants
  - They take up to 100 hours to digest their food!
- The latin name for koalas is *Phascolarctos cinereus* ("ash-colored pouch bear")

Incorrectly Nesting Tags

```
<p>
  HTML is <em>really,
  <strong>REALLY<em class="bad"></em></em> lots of</strong> fun!
</p>
```

incorrect

Tags must be correctly nested

- A closing tag must match the most recently opened tag
- The browser may render it correctly anyway, but it is invalid HTML

How would we get the above effect in a valid way?

```
<p>
  HTML is <em>really,
  <strong>REALLY lots of</strong><em class="good"></em></em> fun!
</p>
```

correct

Links (Anchors): <a>

Links, or “anchors”, to other pages (inline)

```
<p>
   Search for it on <a href="https://www.google.com/">Google</a>!
</p>
```

code

Search for it on Google!

output

Uses the `href` (Hypertext REFerence) attribute to specify the destination URL

- This can be absolute (to another web site) or relative (to another page on this site)
  - Absolute example: `"https://www.google.com/"`
  - Relative example: `"/img/myface.jpg"`

Anchors are inline elements and must be placed in a block element such as a `<p>` or `<h1>`

Images: `<img>`

Inserts a graphical image onto the page (inline)

```
<img src="img/koala-with-leaf.png" alt="A Koala with a leaf"
title="Logo">
```

code

The `src` attribute specifies the image URL

output

Motivating `alt` text

HTML5 also requires an `alt` attribute describing the image, which improves accessibility for users who can't otherwise see it.

The value of the `alt` attribute is also what you see if the image is not successfully loaded.

```
<img src="img/koala-with-leaf-broken.png" alt="A Koala with a leaf" title="Logo">
```

code

output

More about Images

If placed in an <a> anchor tag, the image becomes a link

```
<a href="https://fit.hanu.vn/courses/fall2022/wpr/">
  <img src="img/courselogo.png" alt="WPR Course Logo" title="Logo">
</a>
```

code

output

Relative vs. Absolute Paths for Links and Images

Relative: paths are relative to the document linking to the path.

- Linked files within the same directory: “filename.jpg”

```
<a href="my-other-page.html">Check out my other page!</a>
```

- Linked files within a subdirectory (e.g. “img”): “img/filename.jpg”

```
<img src="img/koala-with-leaf.png" alt="A Koala with a leaf" title="Logo">
```

Absolute: paths refer to a specific location of a file, *including the domain and protocol.*

- Typically used when pointing to a link that is published online (not within your own website).
- Example: "https://validator.w3.org/"

HTML Character Entities

A way of representing any Unicode character within a web page

character(s)	entity
< >	< >
é è ñ	é è ñ
™ ©	™ ©
π δ Δ	π δ Δ
И	И
" &	" &

- A complete list of HTML entities: http://en.wikipedia.org/wiki/List_of_XML_and_HTML_character_entity_references
- How you you display the text `&` on a web page?

General Outline with HTML5

General outline of a document body

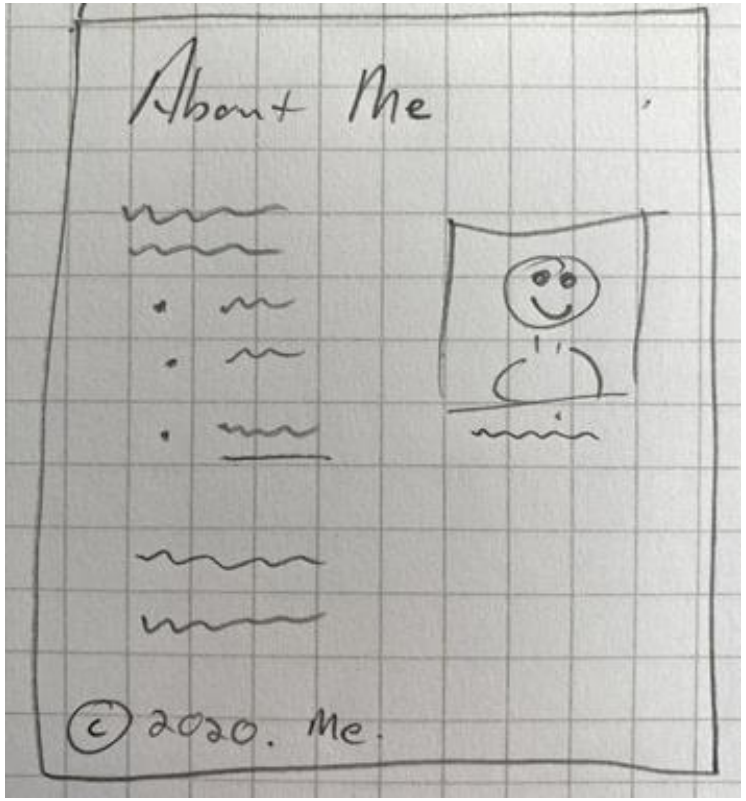
```
<body>
  <header>
    <!-- Header of the webpage body (e.g. logo, navigation bar) -->
  </header>
  <main>
    <!-- Main section of the webpage body (where most content is) -->
  </main>
  <footer>
    <!-- Footer of the webpage body (e.g. copyright info) -->
  </footer>
</body>
```

For different types of pages, you may have more elements but there are the ones you should follow as a guide for most of your web pages.

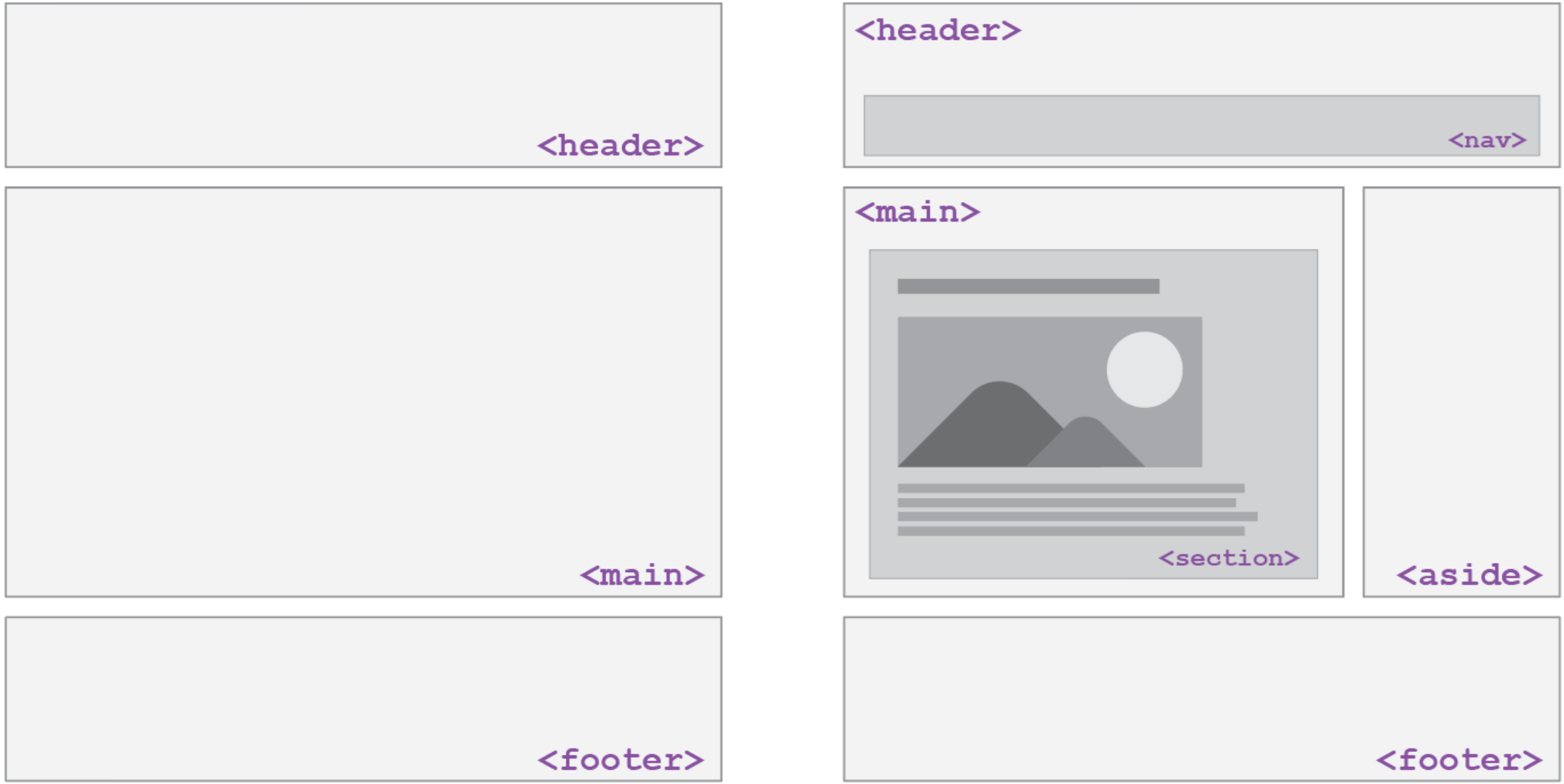
Tips when drafting HTML/CSS web pages

- Identify the meaning of the different things that will be on your page.
- Start with a sketch/wireframe before jumping into code.
  - This will give you a good idea of what kinds of things (and why) will show up in your HTML.
- A great resource on getting started with wireframes can be found here:

https://jesmingonzales.wordpress.com/2012/02/27/assignment-3-wireframes-basics-of-the-internet-design-methodology-assignment-2-review/



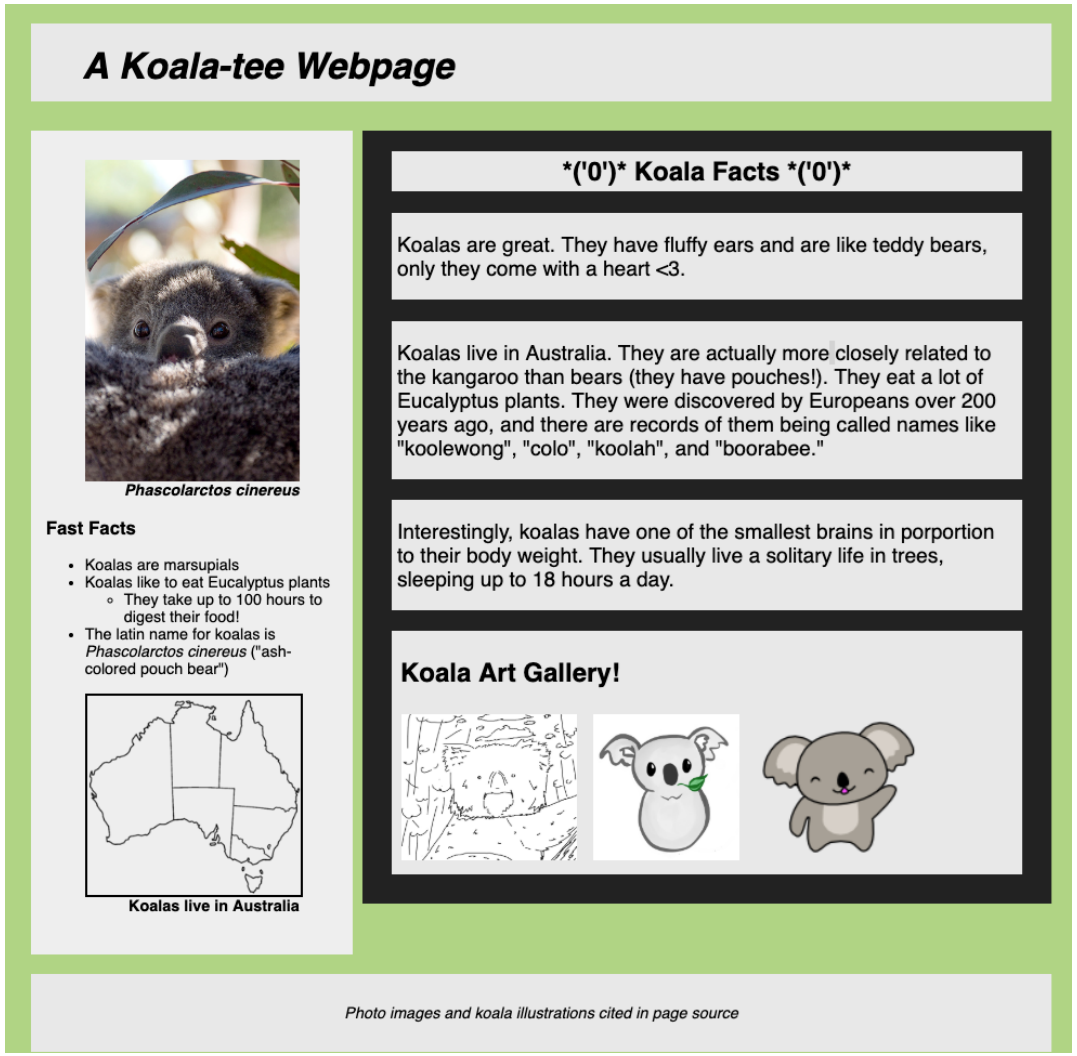
HTML5 Semantic Tags to Define Structure



HTML vs. Rendered Web Page

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Koala Fan Page</title>
  </head>
  <body>
    <header>
      <h1>A Koala-tee Webpage</h1>
    </header>
    <main>
      <aside>
        <!-- Left sidebar -->
      </aside>
      <section>
        <!-- Koala facts (header and paragraphs)-->

        <article>
          <!-- Koala art gallery -->
        </article>
      </section>
    </main>
    <footer>
      <!-- Image citations -->
    </footer>
  </body>
</html>
```



HTML5 and Semantic Tags

<main>

Main content of the document - unlike <header> and <footer> tags, there can only be one main element in the <body>. The content inside should be unique and not contain content that is repeated across pages (e.g. sidebars, nav link, search bars, etc.)

<header>

Header element - contains header information for page body or section/article, including logos, navigation bar, etc.

<footer>

Footer element - contains footer information for page body or section/article, including copyright information, contact, links, etc. Also often used with block quotes to cite sources (see CP1 about.html for an example!).

`<article>` vs. `<section>`

`Articles` are complete, standalone content.
`Sections` are pieces of a greater whole.

And: `div` has no semantic meaning, should only be added for selecting content in CSS/JS, and should be your “last resort”.

Validating your HTML

W3C's online HTML validator:

https://validator.w3.org

Why use valid HTML5 and it is important to write proper HTML code and follow proper syntax standards?

- More interoperable across different web browsers
- More likely that our pages will display correctly now and in the future
- To ensure **accessibility**

Cascading Style Sheets (CSS): `<link>`

```html
<head>
  ...
  <link href="filename" rel="stylesheet">
  ...
</head>
```

html

- **CSS** describes the appearance and layout of information on a web page (as opposed to HTML, which describes the content)
- Can be embedded in HTML (bad) or placed into separate `.css` file (much better)

Basic CSS Rule Syntax

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

template

```
p {
  color: red;
  text-decoration: underline;
  font-family: sans-serif;
}
```

example

- A CSS file consists of one or more rulesets containing one or more rules
- ***Selectors*** designate exactly which element(s) to apply styles to
- ***Properties*** determine the styles that will be applied to the selected element(s)
- Each property has a set of **values** that can be chosen
- There are currently over 200 possible style properties to choose from - use the Chrome inspector for useful autocomplete of property values!
- Together, a selector and the list of properties that follows is a ***rule***.

Type selector

```
section {
      background-color: #AED581;
      font-size: 15pt;
}
```

Selects an HTML element. To apply a style to all `section`s of the pages the CSS file is linked to, write a selector for the `section` tag.

CSS pseudo-classes

```
a:link    { color: #FF0000; } /* unvisited link */
a:visited { color: #00FF00; } /* visited link */
a:hover   {  /* mouse over link */
  color: #FF00FF;
  cursor: pointer; /* can set new pointer icon, usually a "hand" */
}
```

Class	Description
:hover	An element that has the mouse over it
:visited	A link that has already been visited
:first-child	An element that is the first one to appear inside another
:nth-child(N)	Applies to every Nth child of a given parent

There are *many* more, but these tend to be the most common.

Grouping selectors

```
section, p, h3 {
      color: green;
      font-size: 14pt;
}

p {
  text-decoration: underline;
}
```

CSS allows us to reduce redundancy by grouping rules into rulesets to style multiples types of elements with common styles.

What will the `<p>` tag look like here?

`id` and `class`

`id`

- Unique identifier for an element
- Each unique `id` can only appear once per page
- Each element can only have one `id`

`class`

- Non-unique grouping attribute to share with many elements
- Many elements (even of different types) can share the same `class`
- Each element can have many different `classes`

Why use `classes` and `ids` when we can select by the tag name?

Consider the following situation...

```html
<article>
  <p>hello 154 students</p>
  <p>week 2, woohoo!</p>
  <p>can you just style me?</p>
  <p>but not me, haha!</p>
  <p>don't forget me</p>
  <p>I need my own styles</p>
  <section>
      <p>hello from inside the section</p>
  </section>
</article>
```

html

- What CSS selector would allow you to style the contents of *only* the 6th `<p>` tag without changing the HTML?
- What CSS selector would allow you to style the contents of the 3rd and 5th `<p>` tags without changing the HTML?

Consider the following situation...

```html
<article>
  <p>hello 154 students</p>
  <p>week 2, woohoo!</p>
  <p class="demo2">can you just style me?</p>
  <p>but not me, haha!</p>
  <p class="demo2">don't forget me</p>
  <p id="demo1">I need my own styles</p>
  <section>
      <p>hello from inside the section</p>
  </section>
</article>
```

html

Uses the # to indicate demo1 is an id

```
#demo1 {
   ...
}
```

Uses the . (dot) to indicate demo2 is a class

```
.demo2 {
   ...
}
```

Adding ids and classes to html

Ids

- Unique identifier for an element
- Only allowed one id value per page
- Each element can only have one id

Class

- Non-unique grouping attribute to share with many elements
- Many elements (even of different types) can share the same class
- Each element can have many different classes

```
<section id='intro'>
     …
</section>
```

```
<p class='content'>hi</p>
<p>howdy</p>
<p class='content'>bye</p>
```

Combinator (Hierarchy) Selectors

Combinators use the HTML document (DOM) hierarchy to select elements

Classification	Description	Example
Descendent Combinator: `A B`	Selects all B descendants somewhere inside A elements	`ul li`
Child Combinator: `A > B`	Selects all B descendants directly inside A elements	`ul > li`
Adjacent Sibling: `A + B`	Selects B *only if* it shares a parent with A and is *immediately* after it in the DOM.	`#first + #second`
General Sibling: `A ~ B`	Selects all B elements that share a parent with A and appear *after* A in the DOM.	`p ~ img`

CSS Selectors: Simple selectors

Classification	Description	Example
Type selector	Selects an element type	`p`
Class selector	Selects all elements with the given class attribute value	`.koala-face`
ID selector	Selects the element with the unique id attribute value	`#scientific-name`

Given the HTML & CSS below what color will the <p> be?

```html
<p id="text">
  Very pretty, very stylish
</p>
```

html

```css
p {
  color: blue;
}

p {
  color: red;
}
```

css

The “Cascade” of CSS

All styles "cascade" from the top of the sheet to the bottom

- When you add a second file, conflicting rules are overwritten
- Take care not to override styles if possible
- If you have styles specific to a page, use a second sheet (usually linked after the shared one)

Styles "cascade" together when there are conflicting selectors but different properties.

* The term "cascade" comes from waterfalls: at the top of a waterfall, every drop of water is at the same height. As the water "cascades" down, some of those drops reach the bottom earlier (or later) than others.

“Conflicting” Rulesets?

This is a big part of the “cascade” in CSS:

Like water falling off an edge, it all ends up in a pool at the bottom, but some drops of water get there before others.

```html
<p id="text">
  Very pretty, very stylish
</p>
```

html

```css
p {
  color: blue;
}

p {
  text-decoration: underline;
}
```

css

CSS Specificity

Given an element, and two (or more) different selectors that would "select" it, which rules would actually apply?

```html
<p id="text">
  Very pretty, very stylish
</p>
```

html

```css
#text {
  color: blue;
}

p {
  color: red;
}
```

css

Among different selectors, which one is more specific?

As the browser reads through your CSS source, it calculates a score for each selector. You'll rarely, if ever, see these numbers, but they might look something like 0223 or 1000.

When looking at an HTML element and trying to decide what it should look like, it gathers all the selectors that might apply, sorts by their specificity score, and applies some style in order!

Specificity Scores*

Example Selector	Thousands	Hundreds	Tens	Ones	**Total Score**
`h1`	0	0	0	1	**0001**
`h1 + p::first-letter`	0	0	0	3	**0003**
`li > input[value="name"] > .form`	0	0	2	2	**0022**
`#text`	0	1	0	0	**0100**
`style="" attribute`	1	0	0	0	**1000**

***** You don't have to memorize these, but it's useful to know that there are rules for how the browser decides which selector is more specific than another.

CSS Specificity Calculator: https://specificity.keegan.st

General Specificity Rules

- Inline style gets a specificity value of 1000, and is always given the highest priority!
- Each ID value adds 100 to the score
- Each class value adds 10
- Each element selector adds 1
- If you use `!important`, it will override even inline styles!!!

- If two styles with the <u>same specificity</u> come into conflict, the later one will take precedence.

General Specificity Rules

- In the below example, rule C wins because it’s an inline style.

A: `h1`

B: `h1#content`

C: `<h1 id="content" style="color: pink;">Heading</h1>`

General Specificity Rules

- Equal specificity: the latest rule wins!

```
h1 {background-color: yellow;}
h1 {background-color: red;}
```

Answer: background will be red.

General Specificity Rules

- ID selectors have a higher specificity than attribute selectors

```
div#a {background-color: green;}
#a {background-color: yellow;}
div[id=a] {background-color: blue;}
```

Answer: background will be green.

General Specificity Rules

- Contextual selectors are more specific than a single element selector

```
From external CSS file:
#content h1 {background-color: red;}

In HTML file:
<style>
#content h1 {background-color: yellow;}
</style>
```

Answer: embedded rule in HTML file wins over rule in external CSS file.

General Specificity Rules

- If you use `!important`, it will override even inline styles!!!

```
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

p {
  background-color: red !important;
}
```

CSS Inheritance

Some properties, like `color`, are *inheritable*. That is: if there's no conflict, an element will inherit the value for that property from its parent.

What will the text “Hello World!” look like now?

<u>Bonus question</u>: Does specificity apply with inherited properties?

```html
<div>
  <p id="text">
    Hello World!
  </p>
</div>
```

html

```css
#text {
  color: blue;
}

div {
  color: red;
  text-decoration: underline;
}
```

css

Note: `text-decoration` is *not* inheritable, but takes on the color of the element it's applied to. This leads to unintuitive behavior, like having a red underline with blue text.

The Bad Way to Produce Styles

```
<p>
  <font face="Arial">Welcome to Greasy Joe's.</font>
  You will <b>never</b>, <i>ever</i>, <u>EVER</u> beat
  <font size="+4" color="red">OUR</font> prices!
</p>
```

code

Welcome to Greasy Joe's. You will **never**, *ever*, EVER beat OUR prices!

output

Tags such as `b, i, u` and `font` are discouraged in modern HTML

They are bad because of:

- Semantics
- Code organization
- Difficulty in changing style

Layout with Box Model

Why page layout is important

- **Example 1**: See gatesnfences.html for an example of poor HTML tags, layout, and accessibility (try resizing the page). The “old days” of layout.
  - Take a look at the HTML? table, table, table, table…

- As a user, have you ever left a website (that may be useful) because of the layout or accessibility? (e.g. some buttons, inputs, menus are not accessible)

Layout Techniques in CSS

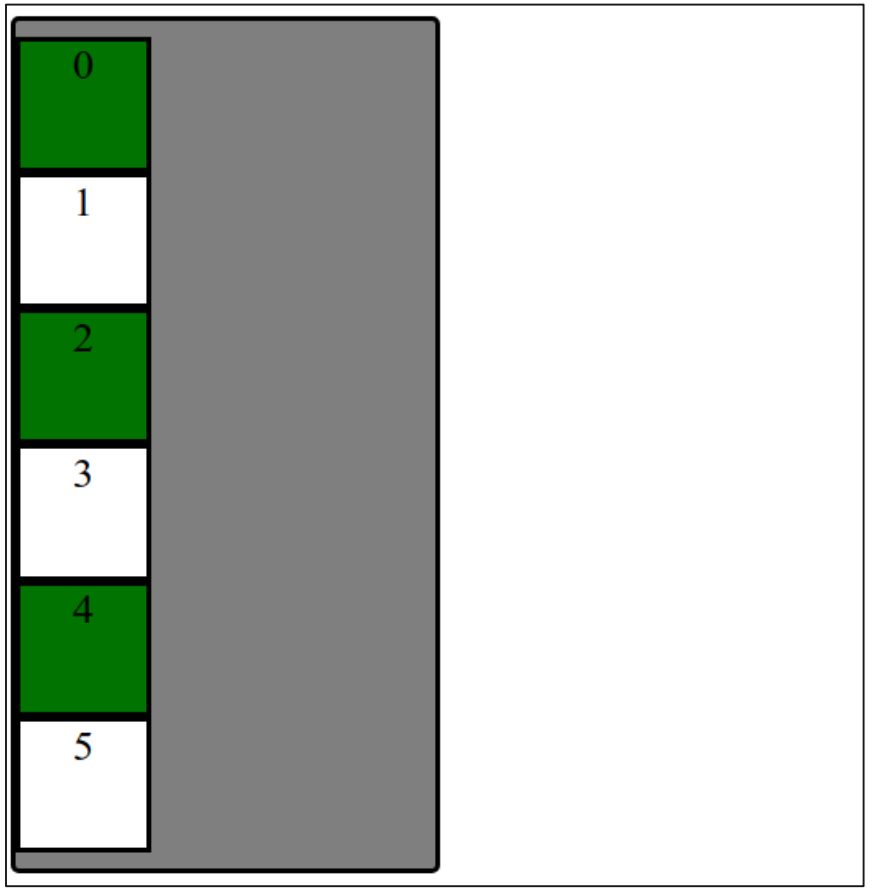
- Block vs. Inline and nesting in HTML
- Box Model (margin/padding/border)
- Flex
- Positioning
- Float (less common today, but still good to know about)

These are what we expect you to focus on, roughly in order of prioritization

Starting with Building Blocks

Question: What does this look like on a webpage by default(ish)?

```
<div id="container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

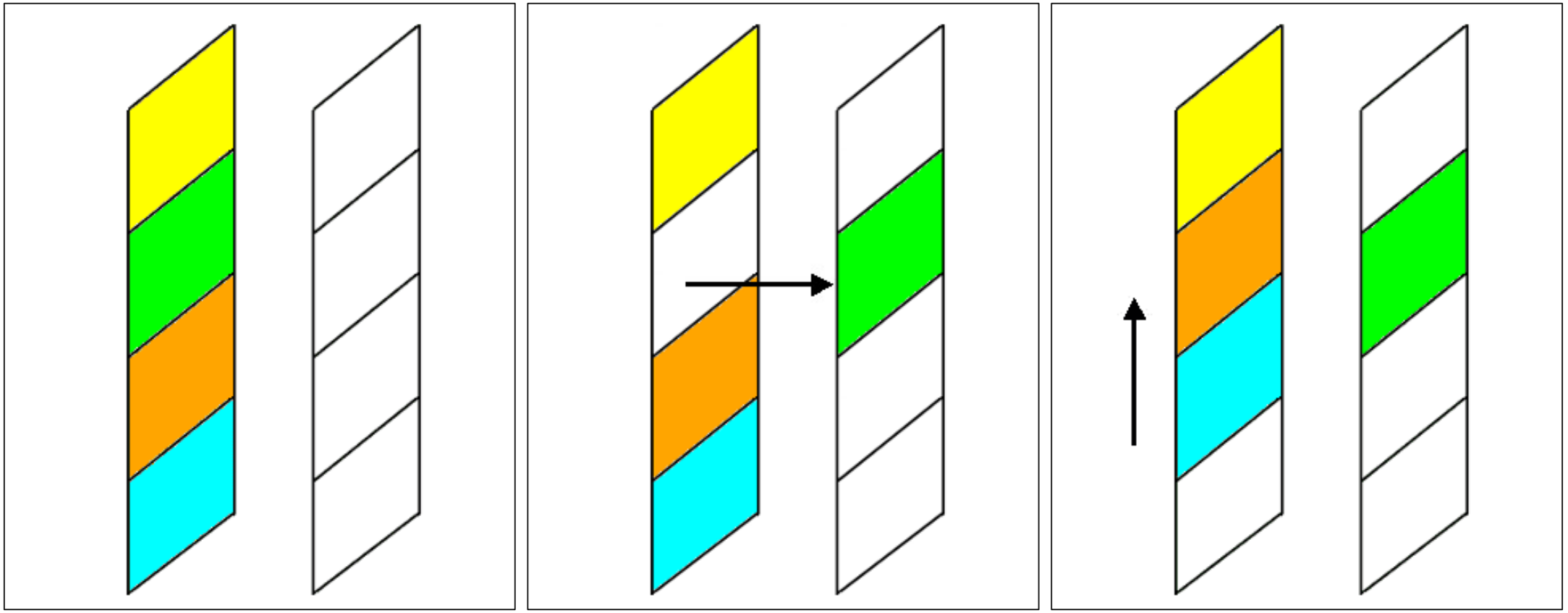


(Older) Layout Method: Floating Elements

A way to remove elements from the normal document element flow, usually to get other elements to "wrap" around them.

Can float `left`, `right` (no `center`)

An analogy: Page Layers as Sheets of Paper



Example with Float

There is a blue block has the `float` CSS property set to `left`. This tells the browser to give the element as much space as it needs, and then start bringing the next content up from below and fill in remaining space.

See the Pen - Box Float by @mehovik on Codepen.

Add `overflow: auto;` to make the parent of a floating element contain the floating element.

And that’s a wrap on float :)

This is not an exhaustive introduction to `float`. There is **SO** much more to learn about `float` as well as some other good use cases for `float` as well. However, our focus for layout will be on the box model and using `flex`.

Default Dimensions of Block and Inline Elements

Height:

- Both block and inline elements normally have the height of their content

Width:

- Inline elements have the width of their content
- Block elements have a width that spans the width of their parent and generally cause a line-break

```
<p id="css">
  CSS is <strong>really</strong> great!
</p>

<p id="html">HTML is pretty cool too.<p>
```

```
#css { background-color: lightblue; }

#html { background-color: lightgreen; }

strong { background-color: white; }
```

CSS is **really** great!

HTML is pretty cool too.

Box model properties

Margin: (*outside*) space between different elements

Border: (*optionally visible*) line that separates elements

Padding: (*inside*) space between element content and border