Program Development in a Graphical Environment (PDGE)

Week 1: Introduction

TEACHER: LEI
CAT: BRODIE

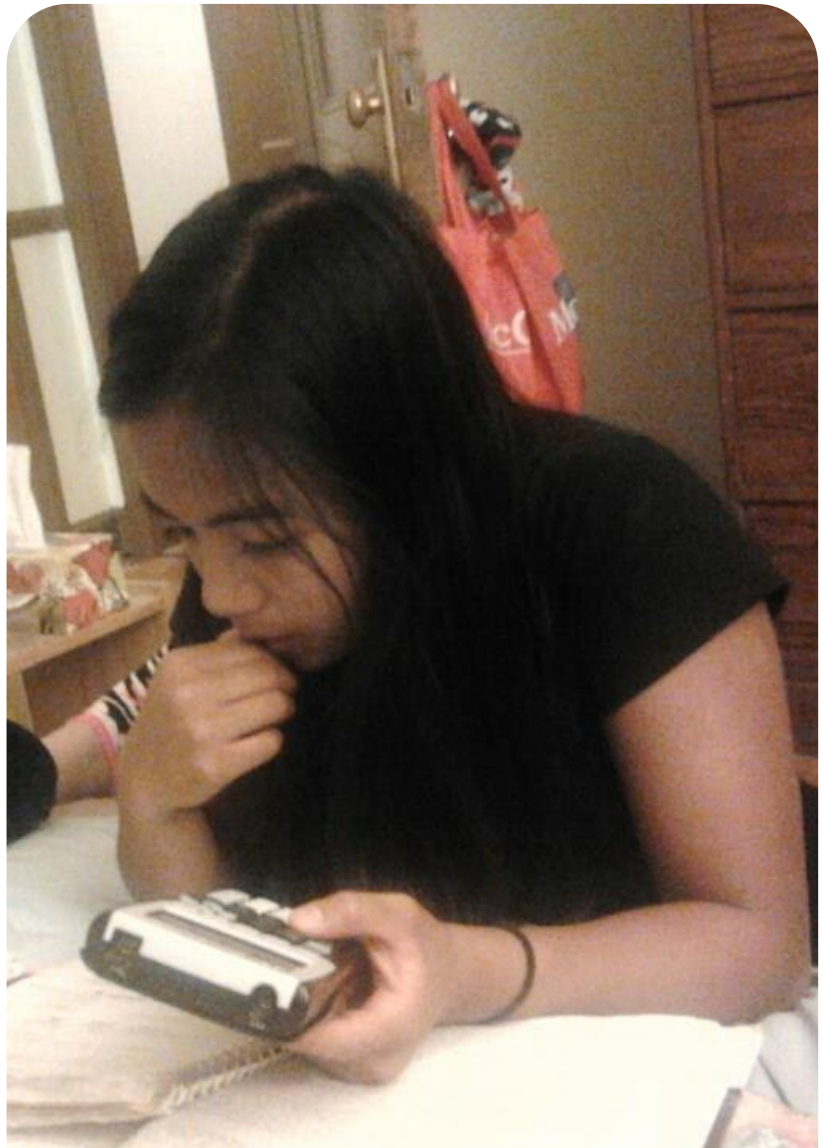
👋 Hello, it's focus time! so please put phones in your bag & keep your laptop closed unless we're doing an activity 

Agenda

- Welcome
- Course Expectations & Goals
- Program Development - in Web!
- Prior Knowledge Questionnaire

Lei’s Journey

- Me when I was 17! -->
- Went to McGill for Biology
- Oops, prefer Computer Science
- Summer job: Kids computing camp
- Longest tech job: Shopify
- Hmm, I think prefer teaching



Lei’s Journey

- Hello Dawson!

Land Acknowledgement

Learn

- 94 Calls to Action
- Read Indigenous authors
- Watch Indigenous content

Participate

- Sep. 30 Truth & Reconciliation Day
- March: Indigenous People's Week
- Attend a pow wow

Please get into pairs ✌️
(trios if need be)

1 person: notetaker (on paper)
1-2 people: notefinders (phone)

Course Outline

Course Outline Quest

1

Teacher info

- Office?
- Office hours?
- Contact preferences?

2

Evaluation

- Types of assessments?
- Weighting?
- Summative?
- Homework hours?

3

Policies

- Penalty for plagiarism?
- Name of document that describes student evaluation policies?

Kahoot!

Learning Environment

Think of a positive learning experience.

What did you, your classmates, and/or your teacher do to make it positive?

Classroom Rules

Write down a few ground rules you'd like for this course to be a positive learning environment.

Consider a negative learning experience.

What made it negative? What could have improved the situation for you?

In groups of 4-5:

- Decide on the essential expectations for this course!
- Write each one on a sticky note
- Consider:
  - What you expect of each other & your teacher
  - Communication (in class and outside)
  - Phone usage
  - Assessment submission
  - Lateness
  - Academic integrity
  - Anything else?

Lei's Expectations

- Respect
- Openness
- Participation
- Communication
- Equity
- Support

Key Information

- Course outline
- Office hours
- Contact preferences

Assessments

- Pre-assignment work
- Reflections
- Submissions

Resources

- Slides
- Links
- Test review

5min Break!

What is a web page?

- What are they made up of?
- Write down a definition!
- Definition: A web page is a file structured with HyperText Markup Language (HTML)
- The HTML gives instructions to include:
  - Text, images, videos, software components
  - Hyperlinks to allow users to navigate between web pages
  - Which layout and styles to apply to the page

What is a website?

- How do you know you’re browsing one?
- Write down a definition!
- Definition: One or more web pages linked together.
- The pages have a common domain name
- What is an example of a domain name?
  - We’ll be coming back to this!

What tools do you need to build a website?

1

Code editor

- How we write website files (like HTML)

2

Web server

- We'll be using a few different options!
- Plus: **Console** to upload to the web server

3

Web browser

- Where we interact with websites
- Recommended: Firefox, Chrome (not Safari!)

HTML Concepts

- HyperText **Markup** Language
- We use markup to indicate the structure and role of text or other elements of a web page
- This distinguishes as HTML **semantic** – it gives *meaning*
- Some roles/structures
  - Heading
  - Paragraph
  - List

HTML Elements

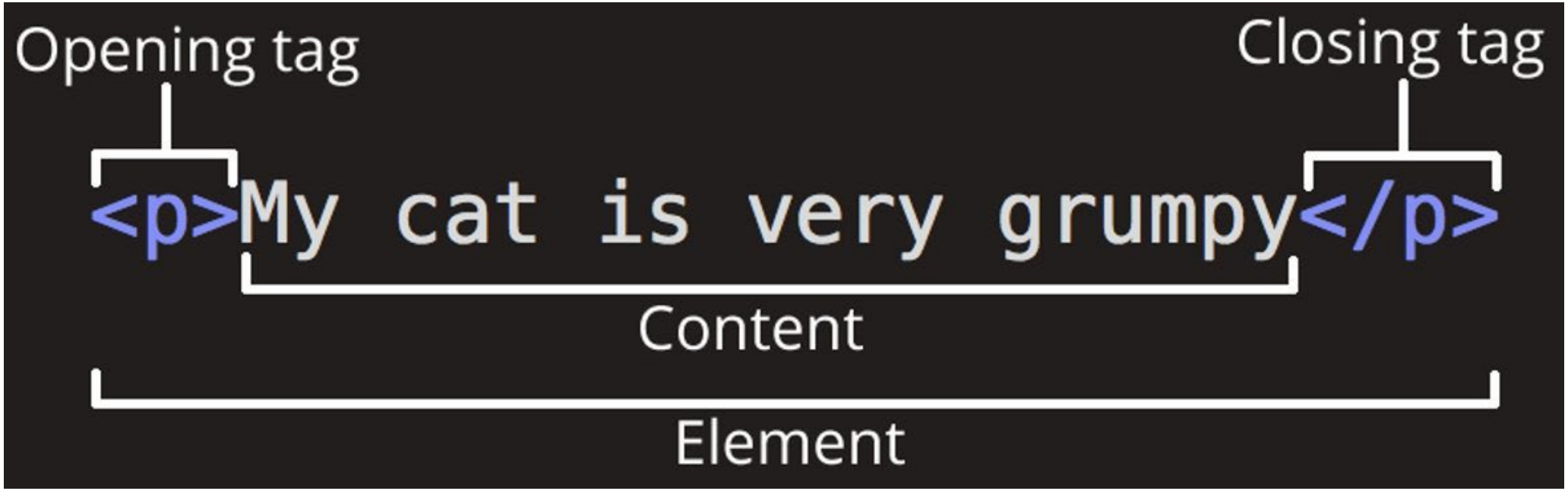
- HTML consists of **tags** that describe the type of content, and the web browser decides how to display it
- For example, if we wanted to display the following text content as a paragraph:

  My cat is very grumpy

  The HTML element would be:

```
<p>My cat is very grumpy</p>
```

HTML Elements parts - regular



Required & Recommended HTML elements – how many do you see?

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Text Basics</title>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" /
  </head>
  <body>


  </body>
</html>
```

HTML Elements parts – self-closing

- Self-closing elements don’t contain text content or other elements
- Often used to insert non-textual elements
- Do NOT have a closing tag
- Examples:

  Horizontal rule: **<hr/>**

  Meta: **<meta charset="utf-8"/>**

Class Coding Goal

Let’s make sure everyone’s got the basics!

3 min challenge! In pairs, discuss:

How many elements make up this page?

Guess the tag name for 3 of the elements!

Main heading of web page

First subtitle

Emphasized words in a paragraph. **Bold words** in a paragraph.

Subsection

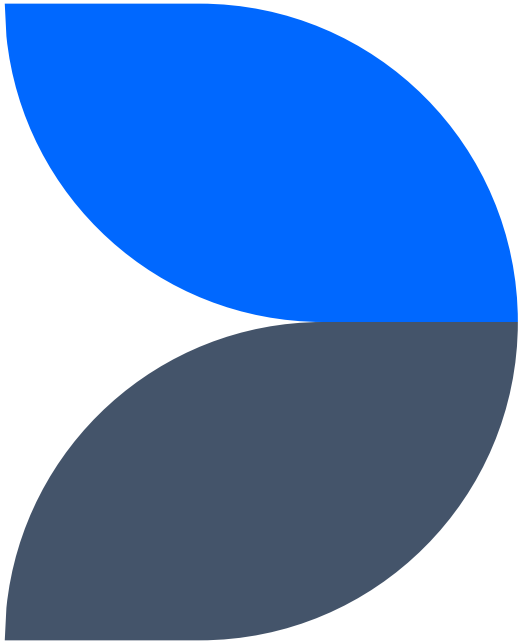
Second subtitle for lists

- First item
- Second item
- Third item

1. First item
2. Second item
3. Third item
    - Third item description

-

Why do some elements start on a new line and others don’t?



Inline vs Block elements

- When we add multiple images, they display beside each other until there's no more room
  These are examples of an **inline** element

- When we add a heading element or paragraph element, they are displayed on the next line
  These are examples of a **block** element

- CSS is where we can change these default behaviours!

Prior Knowledge Questionnaire

Please open course Moodle to find link!

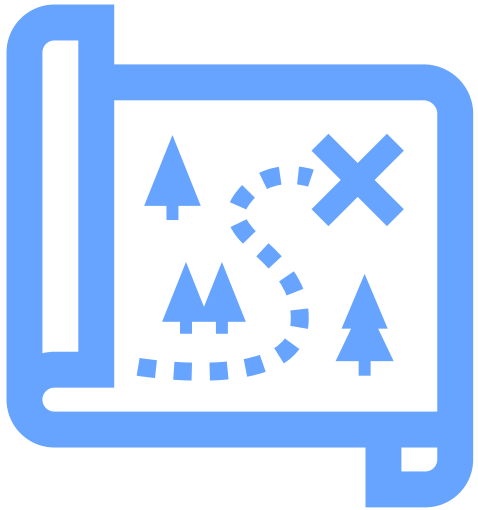
Homework

- Welcome Survey
- Prior Knowledge Questionnaire
- Watch: The Internet video
- Prepare for next class: lab activities
  - Unique website you like
  - 3 photos you have taken of things you like, MUST BE saved to your Dawson OneDrive

PDGE Week 1: Internet & Setup Lab

Agenda

- Homework check: what did you learn from the video?
- Setup Lab Activity: 5 Quests
  1. Website Investigation
  2. Sign up for GitLab
  3. Create a project on
  4. Exploring HTML/CSS
  5. Push to GitLab



Tags: Block text

Heading tags - these describe the heading level, NOT the order (think of lab report or essay with headings)

<h1></h1>

<h2></h2>

<h3></h3>

<h4></h4>

<h5></h5>

<h6></h6>

Note: No h7 or higher, it only goes up to 6!

Paragraph: ** **

Tags: Lists

List types:

Unordered list

- Item 1
- Item 2
- Item 3

Ordered list

1. First item
2. Second item
3. Third item

How to code it in HTML:

```
<ul>
<li>Item 1</li>
<li>Item 2</li>
<li>Item 3</li>
</ul>
```

```
<ol>
<li>Third item</li>
<li>Second item</li>
<li>Third item</li>
</ol>
```

Tags: Text Modifiers

Emphasis

****strong**** (usually makes text bold)

****emphasize**** (emphasize, usually makes text italicized)

Comments: **<!-- The web browser will ignore everything here -->**

Tags: Images

<img> is a tag used for inserting images to a web page, for example:
<img src="images/cat.jpg" alt="lounging gray cat">

Note: It is an self-closing element

src specifies where the image comes from

alt describes the image in case it can’t be loaded or for those who use screen readers to browse the Web

What do you remember?

Get into pairs & designate a scribe ✍️

Answer the following questions:

1. What are three devices or hop locations on the Internet?
2. Define 3 of the following terms: LAN, packet, hop count, IP address, DNS
3. What is one difference between how TCP and UDP work?

What do you remember?

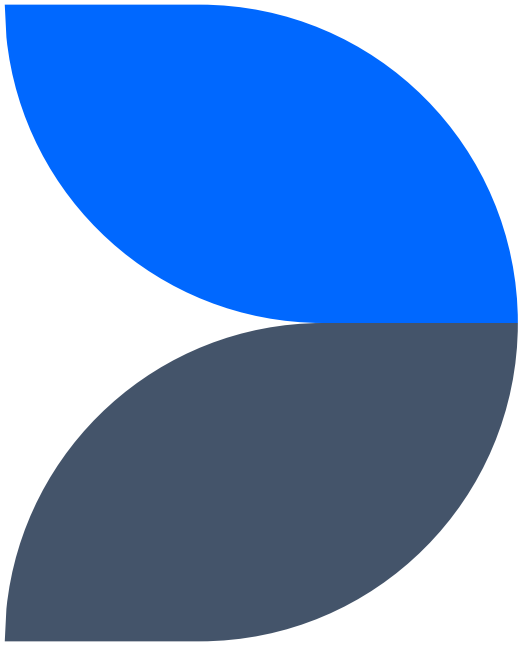
Recreate & label this diagram:

Route a packet travels from client computer to web server:

Setup Lab

5 Quests

Instructions on Moodle



Homework

- Complete any incomplete tasks from **Setup Lab**
- Watch video on the World Wide Web: https://www.youtube.com/watch?v=guvsH5OFizE (up to 7:45)
- Quiz #1 next class on Internet + Web videos!

PDGE Week 2:
HTML & Media usage

Spongetext

The code takes the function input and returns a string with the input alphabetical characters randomly capitalized, except for the first letter, which is always lowercase.

Examples:

englishToSpongecase("brodie12");
>>> bROdiE12
englishToSpongecase("patr1ck*");
>>> paTr1CK*

Discuss Internet + Web videos (5min)

Quiz #1

When you're done, hand in your sheet

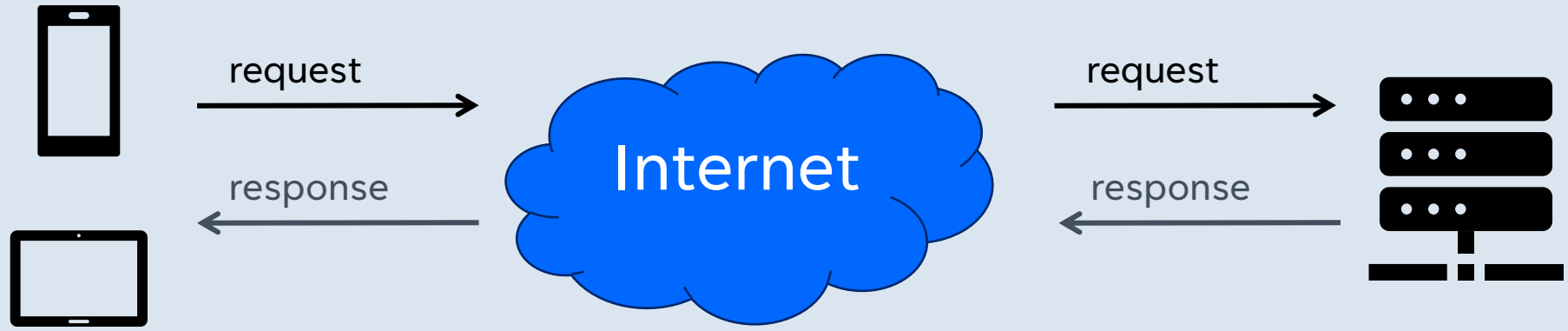
QUIETLY take a break at your seat☺

1. What is DNS for? (2 points)
2. Which came first – the Internet or the Web? (1 point)
3. Give an example of an HTTP status code. *Bonus: what does the code mean?* (1point + *1 bonus point*)
4. What is the role of the web browser? (2 points)
5. Give two examples of HTML tags and explain what each does. (4 points)

Agenda

- Quiz #1
- Client-server model
- Parts of a URL
- Relative paths
- HTML attributes & values
- Class coding
- HTML Validation
- Media use

Client-server model



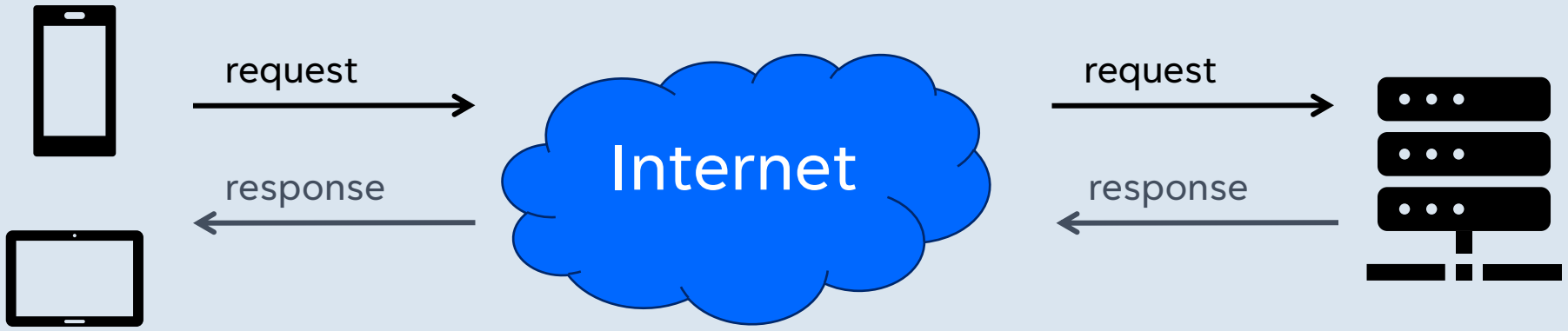
Clients (Browser):

- Requests services or files from a server
- Displays results

Web Server:

- Listens for requests
- Responds to requests

Client-server model



Clients
(Browser)

Web Server

Q: What's another example of a client & server? *(hint: Minecraft)*

Try entering this in a web browser

http://216.59.56.92/about-us

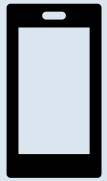
What does it open?

DNS

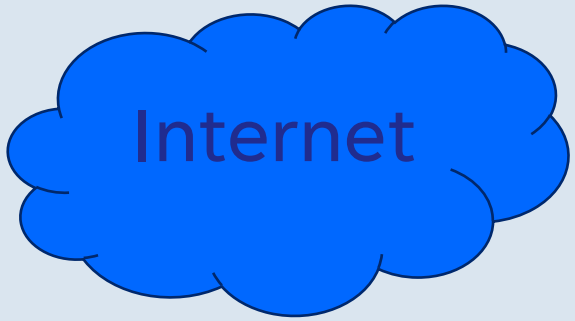
Device: Hey DNS, what is the IP address of dawsoncollege.qc.ca ?

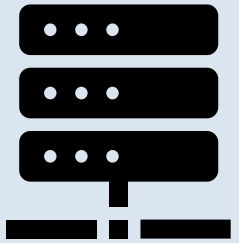
DNS: Let me go look it up! ... Here it is: 216.59.56.92

DNS



Clients
(Browser)



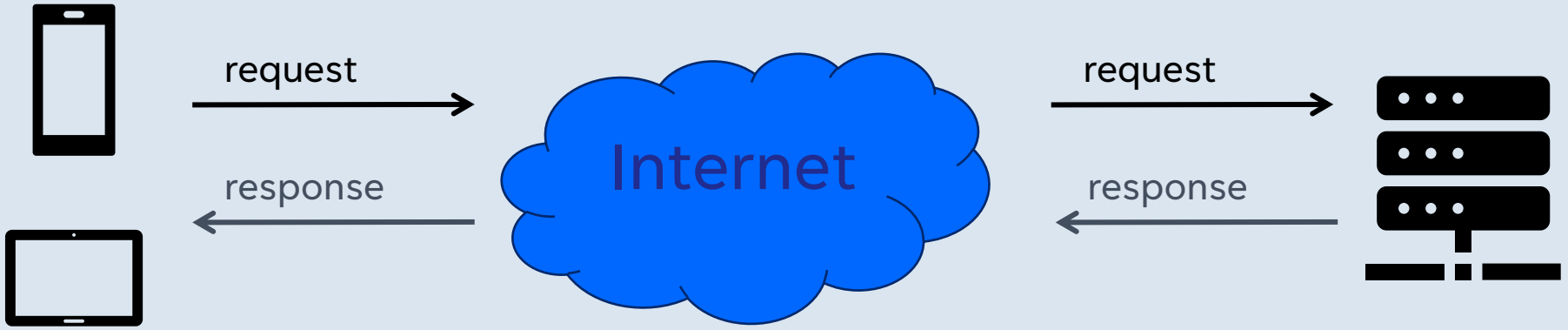


Web Server

Request-Response

Device: Hey web server at 216.59.56.92, can you get me the web page file at the path **about-us** ?

Web Server: Sure! Here are the files you need ☺



Clients
(Browser)

Web Server
216.59.56.92

What are the 4 parts of this URL?

`https://dawsoncollege.gitlab.io/introweb/1-terminology-html/basic-html.html`

What are the 4 parts of this URL?

`https://dawsoncollege.gitlab.io/introweb/1-terminology-html/basic-html.html`

Communication protocol

- A protocol describes how two entities will communicate (like Morse code!)
- **http** and **https** are the main web protocols that describe how browsers and web servers communicate
- What do you think http**s** means?
  - **secure** http

What are the 4 parts of this URL?

Domain name

- The human-readable name of the web server
- We can also think of this as the computer that our browser wants to get a file from

- Domains get more specific from right to left
  - .com, .ca, .io are examples of **top-level domains**

Top-level Domain (TLD)

- The rightmost part of the domain name
- Most common TLDs: **.com, .org, .net**
- Now there are a lot of fun ones:
  - **.art**
  - **.cool**
  - **.pizza**
  - **.quebec**
  - **.tech**

How do you get a domain name?

- A domain name **registrar** provides domain name registrations to the public
- They own the rights to domain names and lease them out
- Examples of registrars?

What are the 4 parts of this URL?

`https://dawsoncollege.gitlab.io/introweb/1-terminology-html/basic-html.html`

Path to the specific file or resource requested

- The rest of the URL indicates where and what the specific file or resource the browser has requested
- The **`.html`** at the end tells you the **file type** that has be requested
  - Here, the browser is requesting the file **basic-html.html** in the folder **1-terminology-html**, that is in folder **introweb**

Practice: Name the parts of this URL

`http://www.immigration.govt.nz/new-zealand-visas`

HTML Element attributes & values

Consider this element:

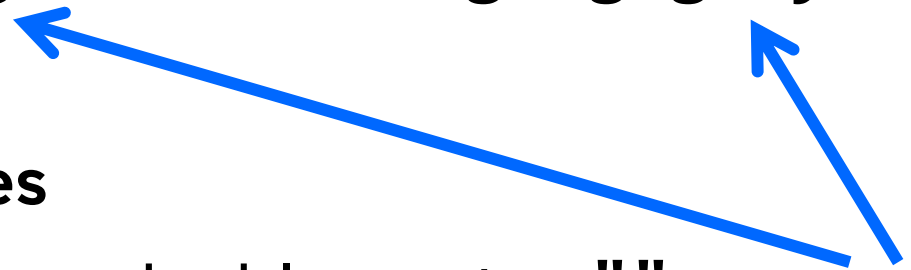
```
<img src="images/cat.jpg" alt="lounging gray cat">
```

src and **alt** are called **attributes**

HTML Element attributes & values

Consider this element:

```
<img src="images/cat.jpg" alt="lounging gray cat">
```



src and **alt** are called **attributes**

The part after the **=** and between double quotes **""** are **values**

When we get to creating forms, this is VERY important terminology to remember!!!

Draw what this code would produce:

```
<h1>Things Lei Likes</h1>
<hr/>
<h2>Food I Love 🍽️</h2>
<ul>
    <li>Mango 🥭</li>
    <li>Tonkatsu 🍗</li>
</ul>
<img src="mango.jpg" alt="mangoes">
<p>
    <em>Photo in public domain.</em>
</p>
<h2>Top 3 Fave 🏀 Players </h2>
<ol>
    <li>Charles Barkley</li>
    <li>Ricky Rubio</li>
    <li>Lu Dort</li>
</ol>
```

```
<h2>Contact 📞</h2>
<ul>
    <li>Email 📧</li>
    <li>Omnivox 🌼</li>
    <li>Phone ☎️</li>
</ul>
```

I will be calling people up to draw 1-2 elements!

Solution:

Things Lei Likes

Food I Love 🍽️

- Mango 🥭
- Tonkatsu 🍗

Photo in public domain.

Top 3 Fave 🏀 Players

1. Charles Barkley
2. Ricky Rubio
3. Lu Dort

Contact 📞

- Email 📧
- Omnivox 🌼
- Phone ☎️

Class Coding

What’s new?

Things Lei Likes

Jump to Contact

Food I Love 🍽️

- Mango 🥭
- Tonkatsu 🍗

Photo in public domain, from this source.

Top 3 Fave 🏀 Players

1. Charles Barkley
2. Ricky Rubio
3. Lu Dort

Contact 📞

- Email 📧
- Omnivox 🌼
- Phone ☎️

How do you know you're coding **correctly**?

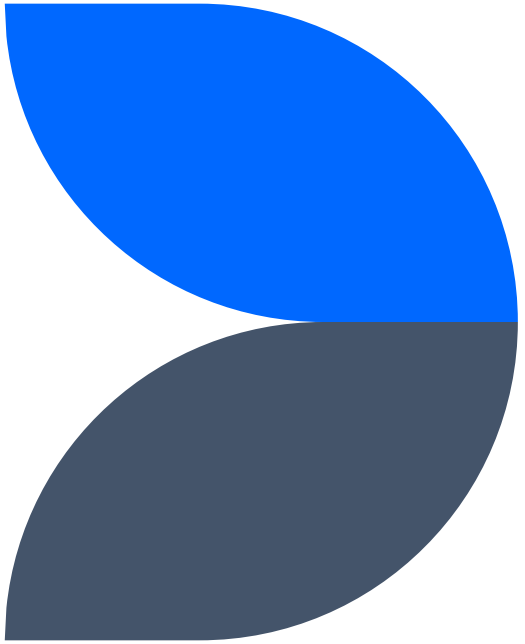
Validator!
https://validator.w3.org/#validate_by_input

This helps us code *correctly*, not just "it works".

5min

In the Setup Lab, images were in the assets folder.

Do you remember how images were referenced in the <img> tag?

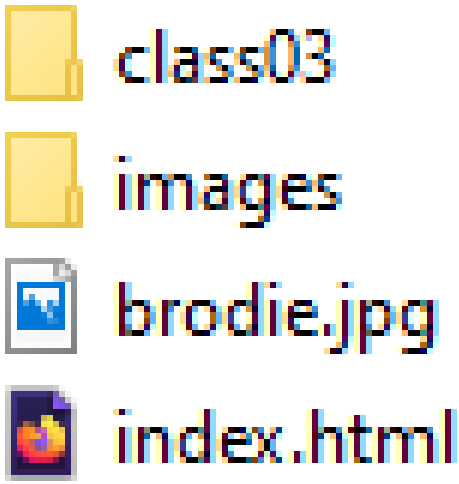


Use a relative path

Relative path: Reference to document or file *relative* to the file it is being used in, in the context of the folder structure.

Examples:

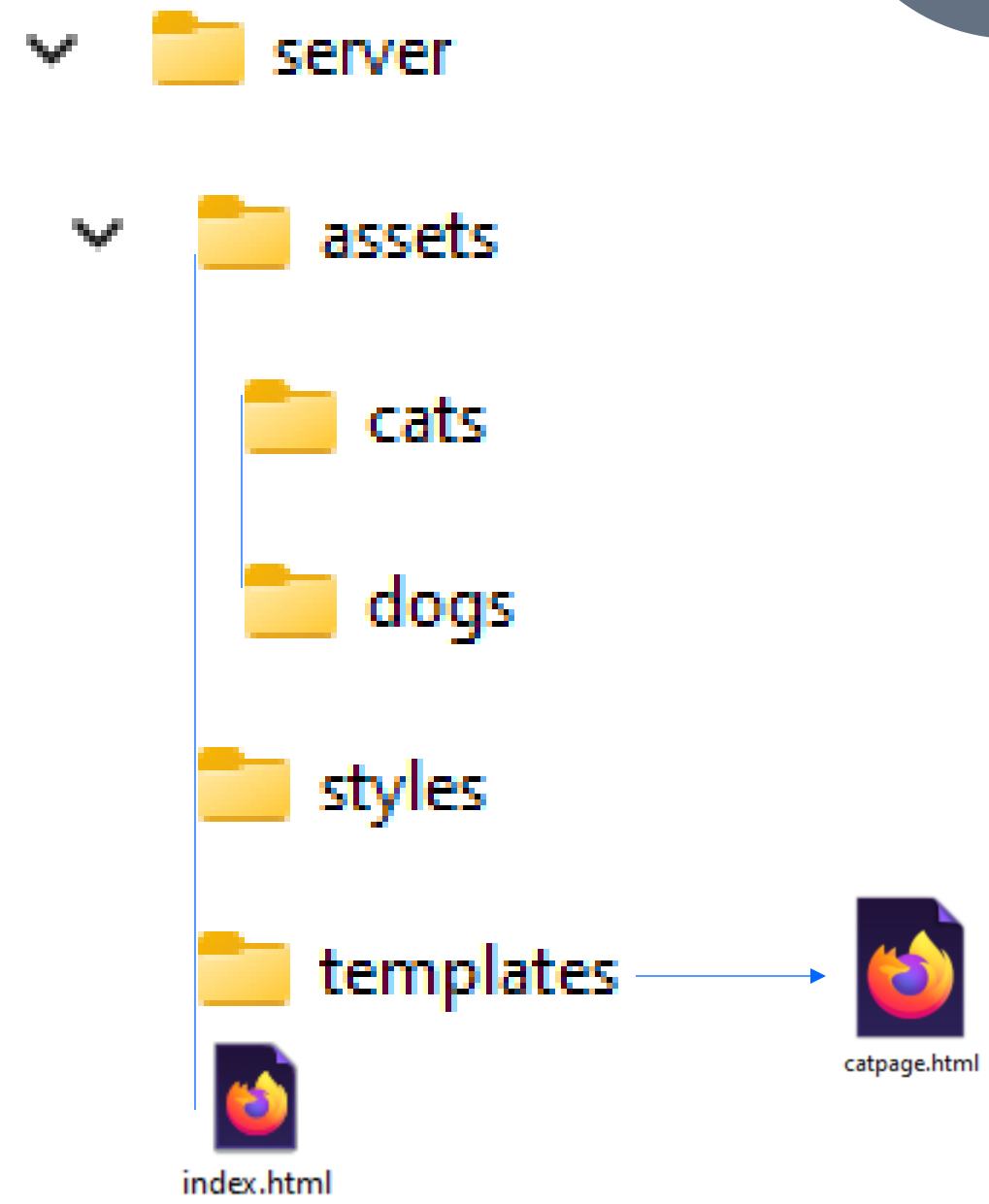
- If I am working on **index.html**, the relative path to an image called **cat.png** in the **images** folder is "**images/cat.png**"
- If I am working on a file in **class03**, then the relative path to **brodie.jpg** is **"../brodie.jpg“**
- **../** indicates going up a folder



Relative path practice

If you are working on a file called **index.html** from the **server** folder, how would you insert an image called **puppy.png** located in the **dogs** folder?

If you are working on a file called **catpage.html** in the **templates** folder, how would you insert an image called **kitty.gif** located in the **cats** folder?



What media can we use?

What images/media are developers allowed to use in a web page?

- Our own content
- Free-to-use content

Before you use content that is not yours:

Check the copyright to see if what you intend to do is allowed by the author!

Copyright types ©

- **Traditional copyright**
  Work cannot be used, adapted, copied, or published without the creator's permission.

  All original work is protected under copyright when it's created.

- **Creative Commons**
  Work may be used without permission, but only according to the creative commons license applied to it.

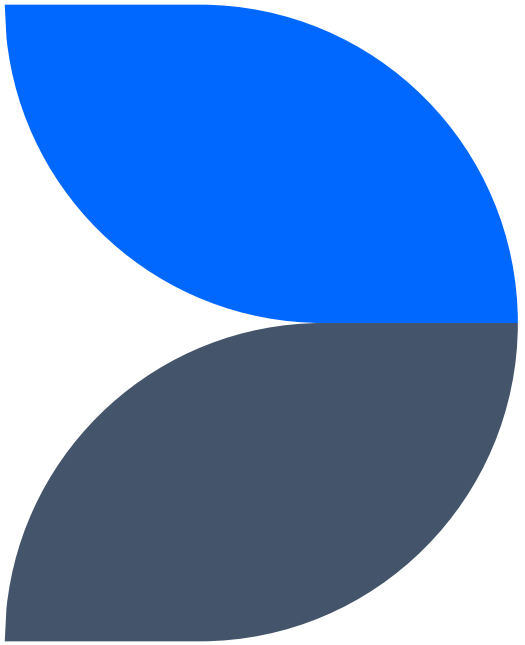
- **Public Domain**
  Work can be used, adapted, copied, or published without restrictions or permission.

Fair Dealing Rules ⚖️

- Allows for the reproduction and use of copyright-protected works for certain purposes without requiring permission, provided that use/dealing is "fair".
- What is fair?
- research, private study, **education**, parody, satire, criticism or review, and news reporting

All media used in lab assignments must be cited according to the license associated with it.

If it's your media, it's easier!



Media citation examples

Traditional
This image is from source link, used under fair dealing for educational purposes.

Creative Commons

Instructions here: https://creativecommons.org/share-your-work/use-remix/

Public Domain

This image is from source link, from the public domain.

Your own media

This image is the author's own.

Media citation in HTML

"Cat" by Tony is licensed under CC BY 2.0

Today’s Tag: Figure

A way to insert an image with a caption! Example:

```
<figure>
    <img src="images/cat.png" alt="white cat">
    <figcaption>
        <a href="https://www.flickr.com/photos/7788419@N05/15218475961">
"Cat"</a> by Tony is licensed under CC BY 2.0
    </figcaption>
</figure>
```

NOTE: This is for when you want to add a caption to an image, images without a caption only need to use the <img> tag.

Today’s tags: hyperlinks

There are three main uses for the a*nchor* tag:

1. Jumping to a specific part of a web page using id, like the Contact heading
2. External links (going to a different web page or website)
3. Opening the default email or phone app to contact a specified email or phone number.

Today’s tags: hyperlinks

```
<a href="https://example.com" target="blank">Clickable text</a>
```

a is for anchor

href attribute specifies where the link goes

target attribute tells the browser how to open the link

blank value opens the link in a new tab

Today’s tags: clickable image

We can also use images as hyperlinks, like this:

```
<a href="https://example.com">
    <img src="images/cat.jpg" alt="cat">
</a>
```

Best Web resource

https://developer.mozilla.org/en-US/

Today’s tags: Sectioning content

Use MDN Content categories to explore what these elements are for:

- <main>
- <header>
- <footer>
- <nav>
- <article>
- <aside>
- <section>

Homework

- Prepare for Quiz 2 on the content from the code today + these slides!
- Prepare for the first lab assignment: See Moodle for instructions

PDGE Week 3: Cascading Style Sheets (CSS)

Quiz Revision

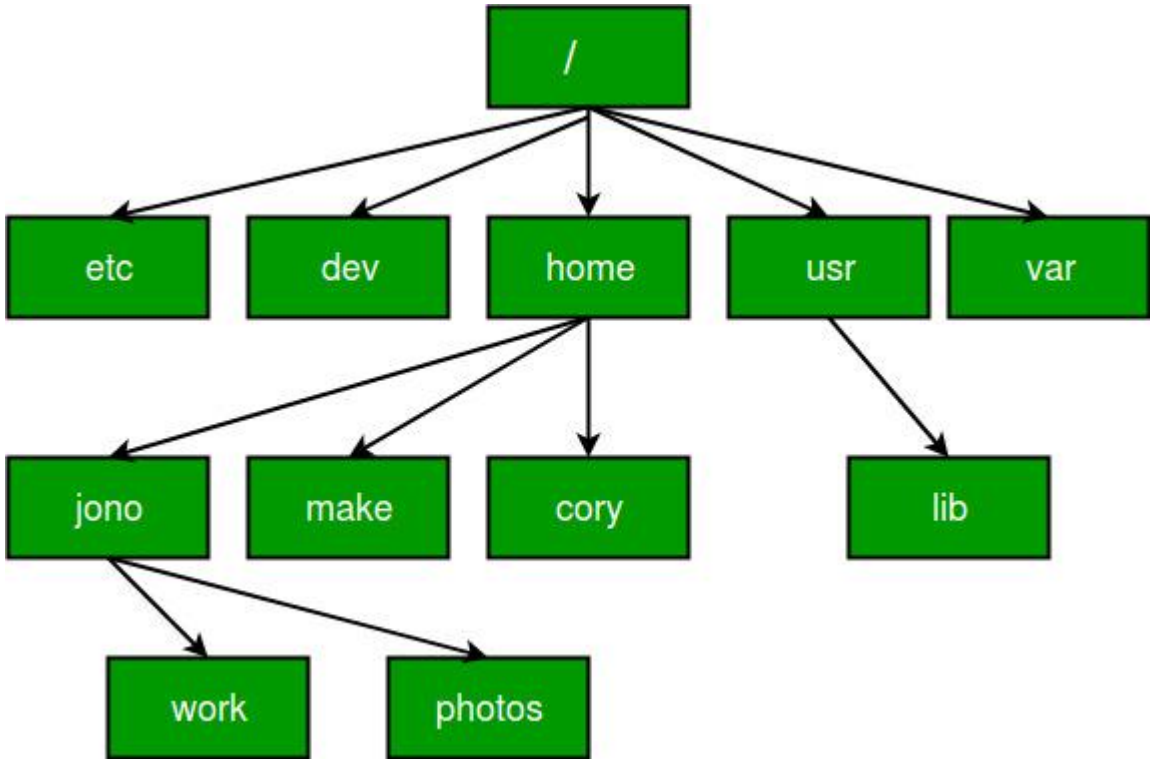
- Earn back 50% of the marks you missed
- How? Resubmit your quiz with the correct answers AND an explanation why you got the question incorrect.
- Examples:
  In question 2, I felt rushed and made this mistake: ...
  In question 5, I confused the terminology and thought that x meant y, but now I realize...
- **DUE the class/lab after the quizzes are returned**

Agenda 

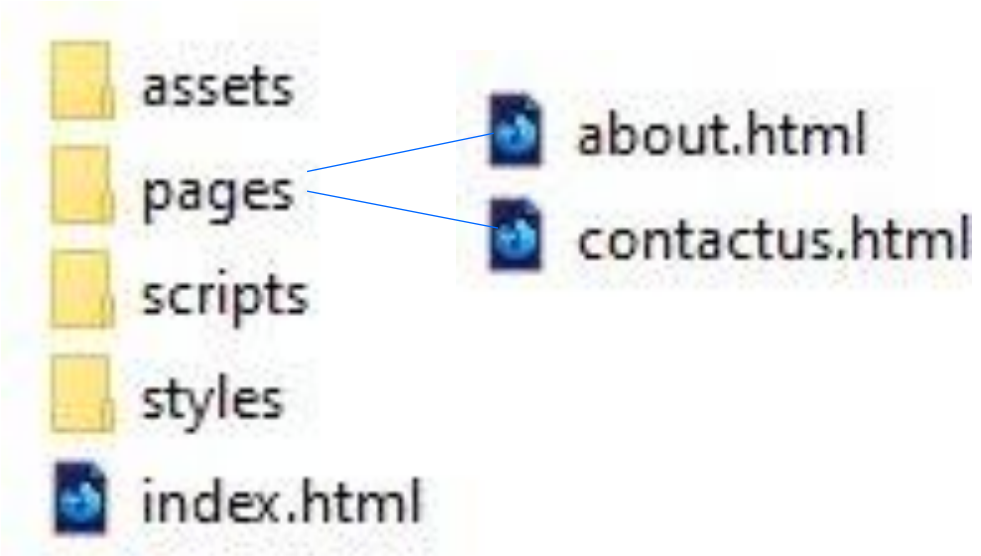
- Quiz #2
- Relative Path practice
- Multi-page websites
- <link> element
- CSS syntax
- Web colours
- Class coding: CSS properties
- Content sectioning
- Box Model
- Pseudo-selectors
- CSS Diner

Relative path practice

1. If you are working on an HTML file in the **/** (root) folder, what is the path to an image in the **work** folder?
2. If you are working on an HTML file called **cory.html** in the **cory** folder, what is the path to an image in the **lib** folder?
3. Pick two folders and ask a neighbour to give what the path between them would be.



Multi-page websites



Adding a link to another page of the same website

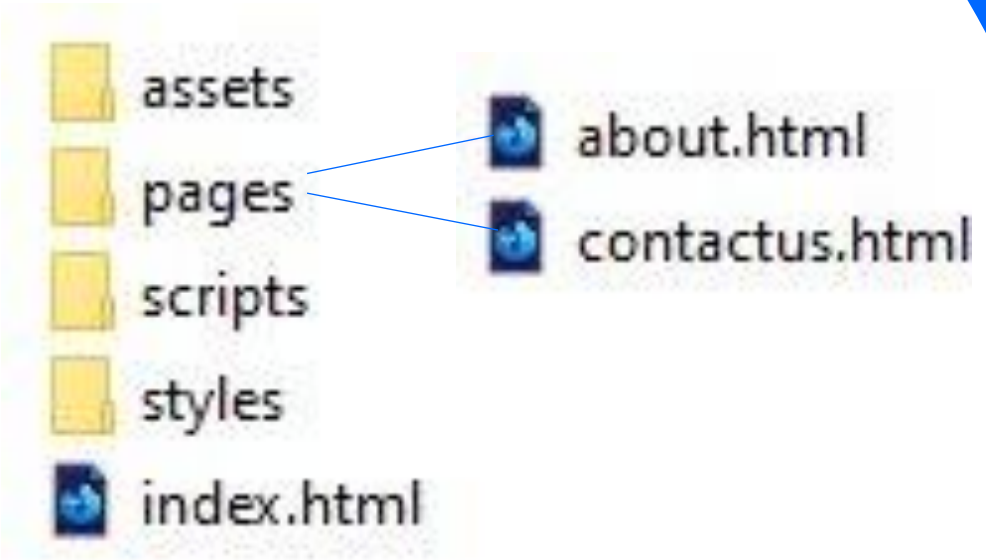
Use the **<a>** element but the **href** value is a **relative path** to the file it refers to, on the server

Practice:

From index.html, create a hyperlink to about.html

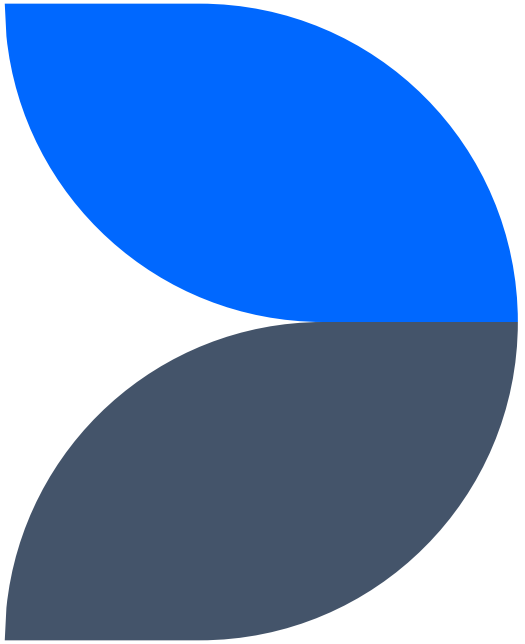
From about.html, create a hyperlink to contactus.html

From contactus.html, create a hyperlink to index.html



CSS TIME 💎

Time to STYLE!



Three ways to style

~~Inline~~ DO NOT USE!!!

- Deprecated, but here so you recognize and avoid it.
- Applies to one element; added to its opening tag: `<p style="color: blue;">`

Internal – ok for testing

- Coded *within* the HTML file within the head section using `<style> </style>` tags
- Applies to the entire HTML file

External – USE THIS

- Coded in a separate CSS file
- Applies to every HTML file it is added to via `<link>`

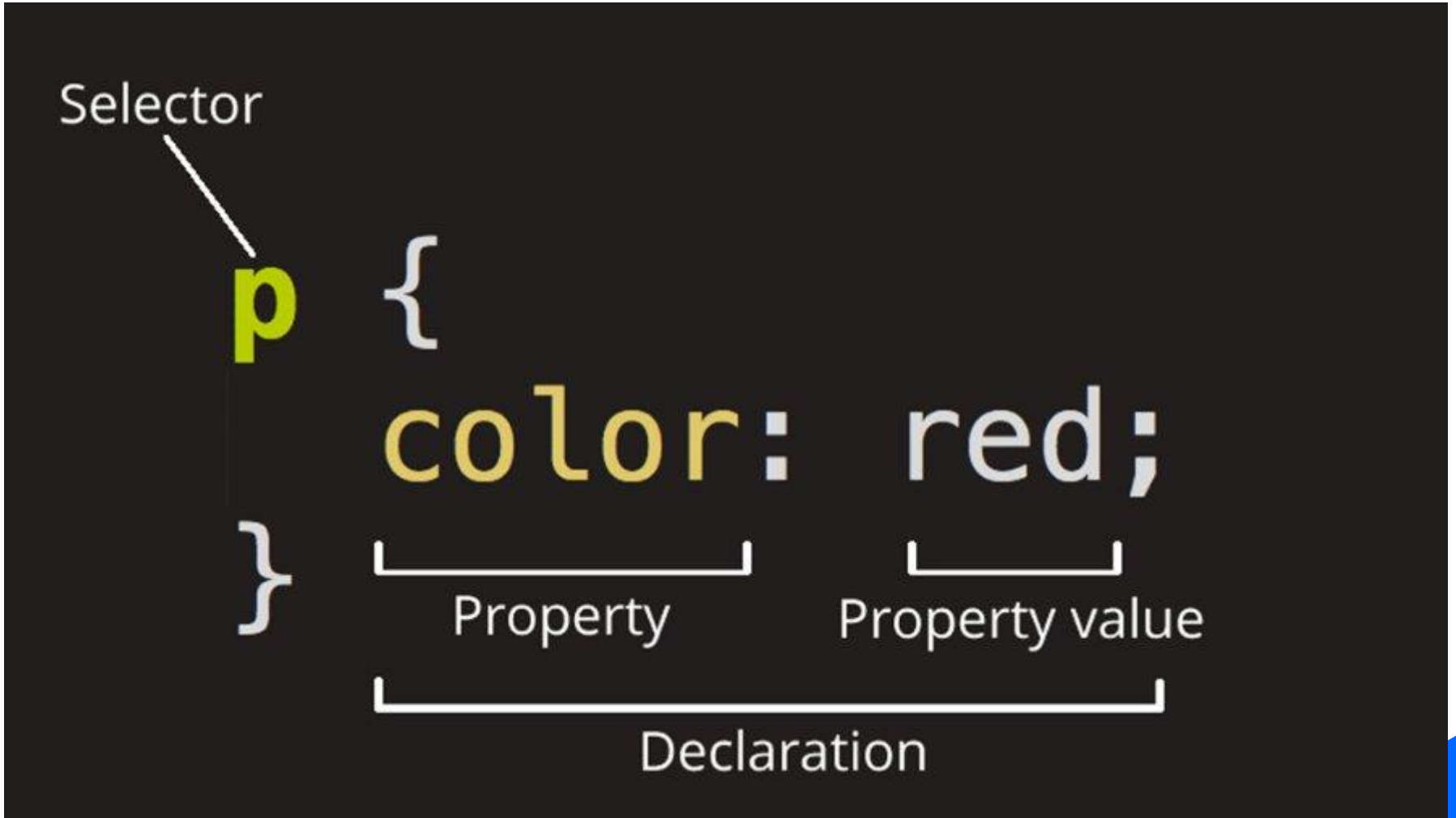
Today’s Tag: <link>

** NOT for hyperlinks!**

<link rel="stylesheet" type="text/css" href="homepage.css">

- You can name your CSS file anything as long as it ends in **.css**
- Like with **<a>** the **href** value is a relative path to your CSS file

CSS Syntax & Terminology



Representing colours in CSS

https://www.youtube.com/watch?v=15aqFQQVBWU&t=43s

In CSS, we can write colours in many ways:

- HTML Color Name
  h1 { color: red; }
- Hexadecimal
  h1 { color: #ff0000; }
- Decimal
  h1 { color: rgb(255, 0, 0); }
- Percentage
  h1 { color: rgb(100%, 0%, 0%); }

What about transparency?

We need another setting: the **alpha** channel!

Alpha also goes from 0 to 255, but represented as a number between 0 and 1

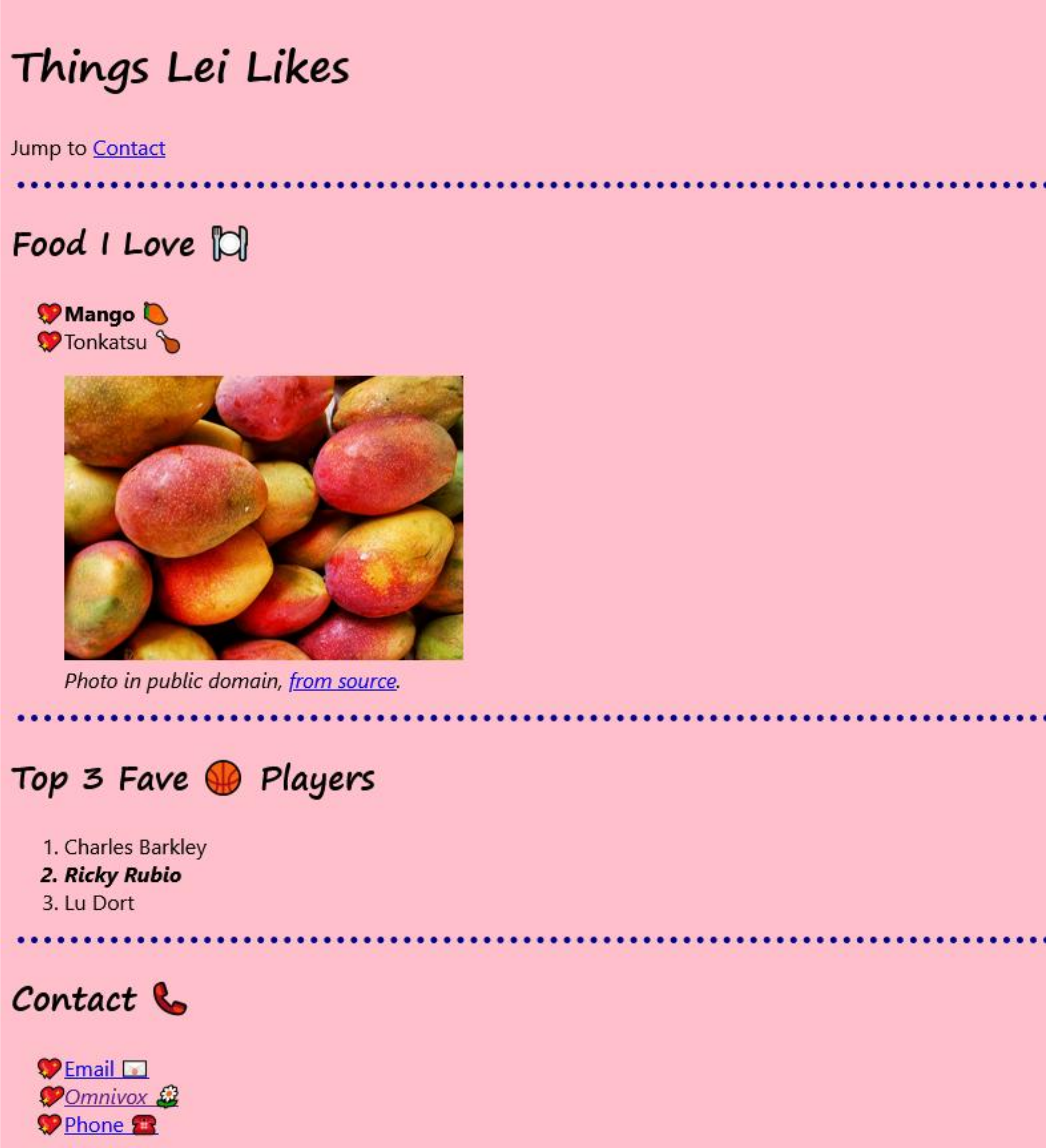
0 = transparent

1 = opaque

```
p { color: rgba(255, 0, 0, 0.5) }
```

Class Coding

What elements need to be styled?



Multiple CSS selectors

```
h1, h2 {
 font-family: 'Segoe Print', serif;
}
```

This CSS rule applies to BOTH h1 and h2 elements. How does this differ from the following?

```
h1 h2 {
 font-family: 'Segoe Print', serif;
}
```

Nested CSS selectors

```
ul li {
    list-style-type: '💖';
}
```

This CSS rule only applies to **li** elements nested inside a **ul** element.

This means it won’t apply to **li** elements nested inside **ol** elements.

What if we want to apply the same style to multiple elements?

What if we want to apply the same style to multiple elements?

1. Add class attribute + value in HTML:

```
<hr class="dashed"/>
<hr/>
<hr class="dashed"/>
```

2. Select it in CSS with the dot `.` :

```
.dashed {
    border: 5px dashed blue;
}
```

Class attributes are selected by a dot .

What if we want to apply the same style to multiple elements?

An element can have multiple classes applied to it:

```
<li class="bold italic">Ricky Rubio</li>
```

The class values are all within quotes "" separated by a space.

Today's CSS Properties

font-family

- Allows you to choose different fonts for your text

font-family: 'Verdana', sans-serif;

- Please add the generic family name at the end of the list!
- The web browser will use its default generic font in case your chosen font is not available
- Generic font family options: **serif, sans-serif, fantasy, monospace, cursive**, **system-ui**

Importing a font

1. Choose a font from Google Fonts
2. Click on the **Get font** button
3. Click on the **< > Get embed code** button
4. Select the **@import** option
5. Copy ONLY the **@import** line, like this:
   @import url('https://fonts.googleapis.com/css2?family=Asimovian&display=swap');
6. Paste what you copied to the top of your CSS file.
7. Copy the font-family line to the ruleset:

```
h1 {
    font-family: "Asimovian", sans-serif;
}
```

Today's CSS Properties

font-weight: bold;

- Allows you to style text as bold
- INSTEAD of

font-style: italic;

- Allows you to style text as italic
- INSTEAD of

font-size: 18px;

- Allows you to specify text size

Today's CSS Properties

border: 5px solid red;

- Shorthand for 3 separate properties
  - border-width
  - border-style
  - border-color
- Use this to add a border around an element
- You can use this to style **<hr>** to be a fun separator 🌟

border-top or **border-bottom**

- To style specific parts of the border

Today's CSS Properties

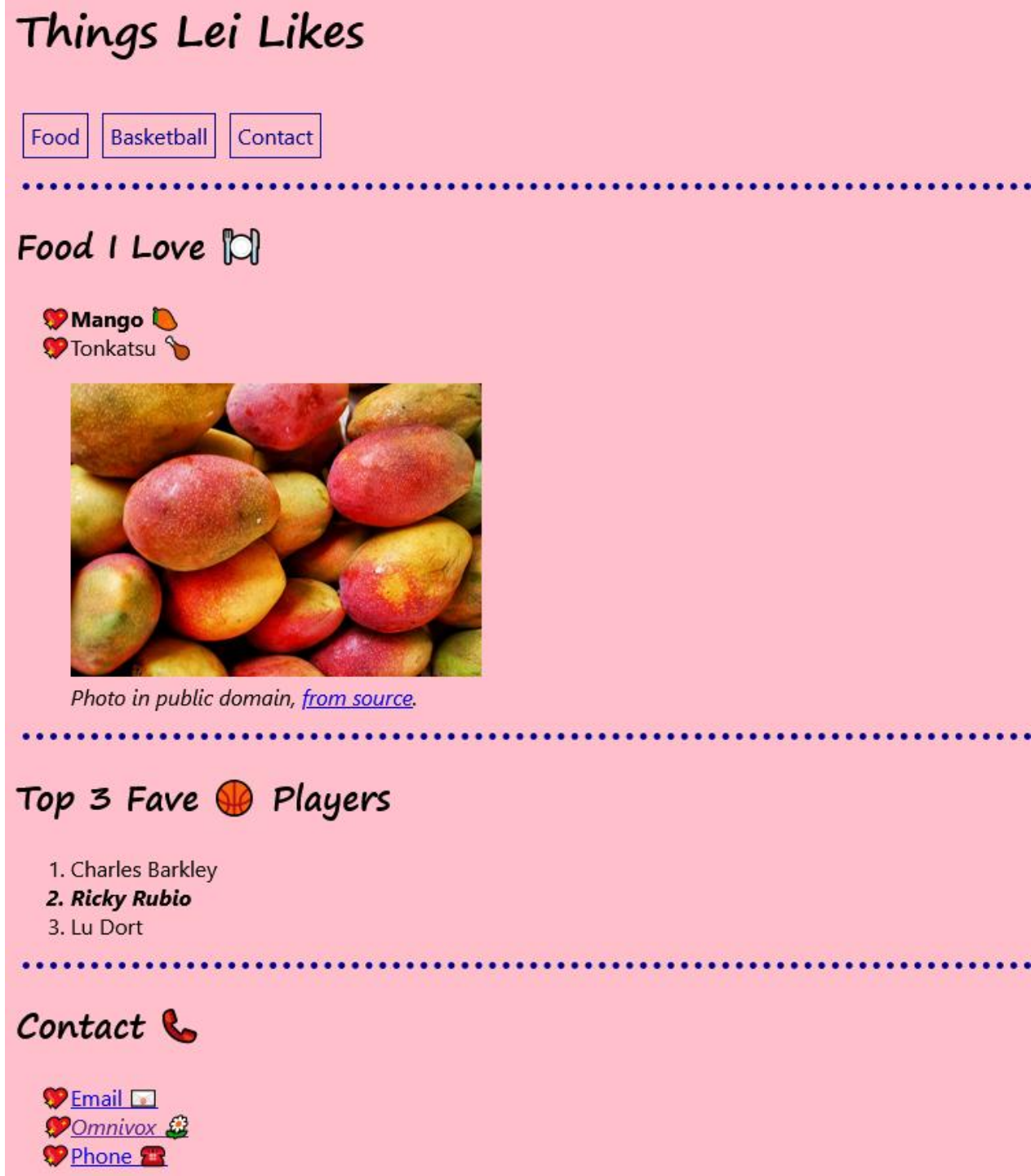
list-style-type: '💖';

- Used to change the bullet-type
- Built-in types: e.g., disc, circle, square
- We can also use emojis!

5min

What *content sectioning* would you apply to this page?

- <main>
- <header>
- <footer>
- <nav>
- <article>
- <section>



Today's Tag:

Inline element used to group content for styling

- It has no effect on content unless you style it
- Think of it like an empty, outer canvas box
- It is **non-semantic**

Before you add a span, you must be able to explain to your teacher why you need it, and no other semantic element works

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span

Today’s Tag:

Block element used to group content for styling,

- Otherwise like span, **non-semantic**

Before you add a div, you must be able to explain to your teacher why you need it, and no other semantic element works

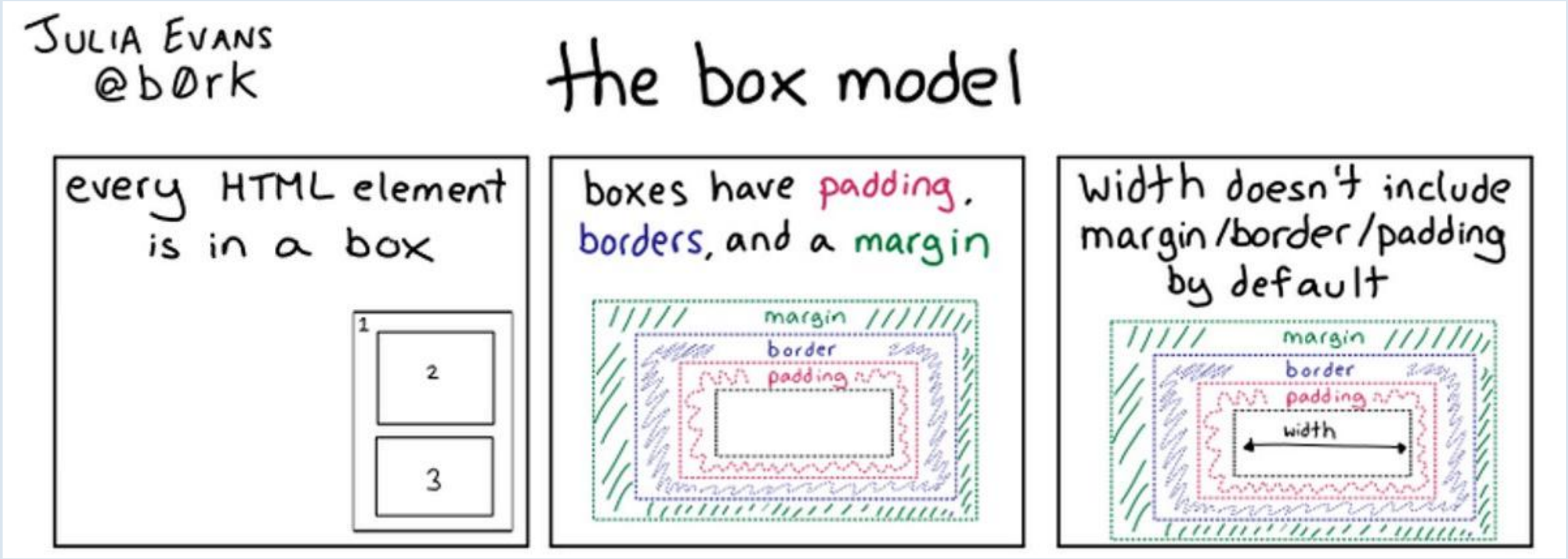
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div

Class Coding

Which elements need to be styled?

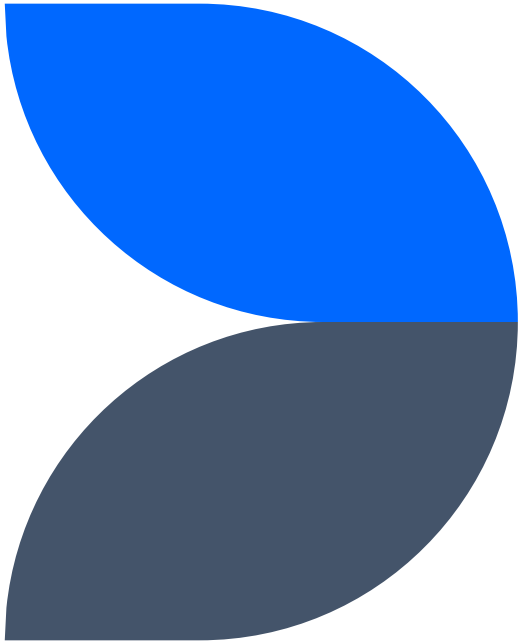
Which properties?

CSS Box Model



CSS Box Model

In action with **Inspect**



Today’s CSS: margin & padding

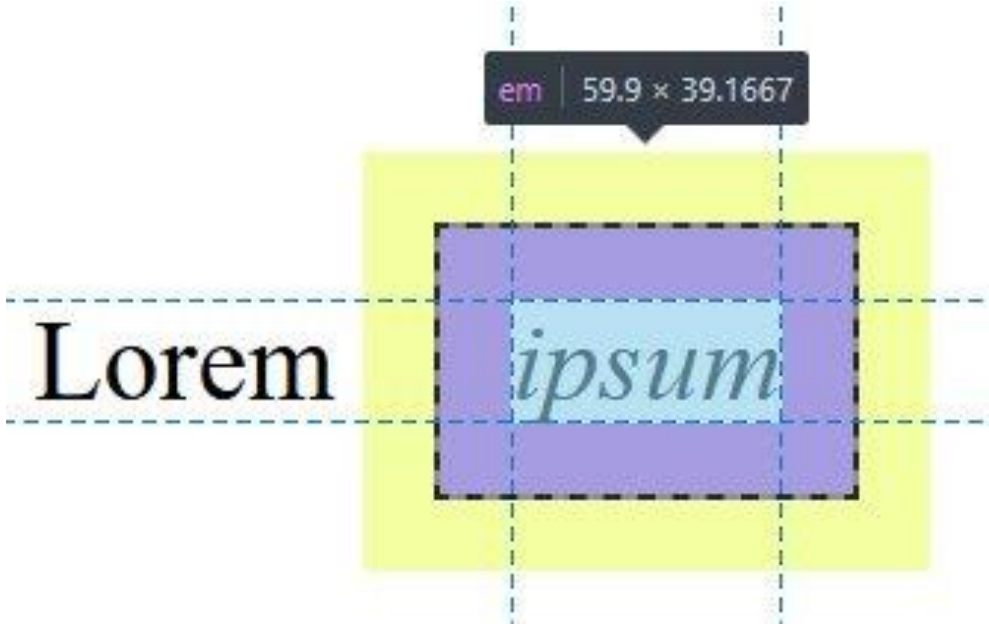
All elements can be given a border, margin, and padding.

For the element:

** Lorem ipsum **

With this CSS:

```
em {
    padding: 10px;
    border: 1px dashed black;
    margin: 10px;
}
```



If the black dashed line is the border, what is the yellow area? What is the purple?

Today's CSS Properties

display: inline-flex;

Force an element to display:

- **inline** (adds it to the page horizontally if there's room)
- **flex** (flexible layout)

Other options:

display: flex;

Same as above but **block** (adds it to the page on its own shelf)

Today's CSS Properties: Pseudoclass selectors for <a>

Different selectors based on the state of the li

- normal, unclicked/unvisited link:

```
a:link {
    color: green;
}
```

- mouse is hovering over link: **a:hover**
- mouse is clicking the link: **a:active**
- visited link: **a:visited**

Julia Evans
@b0rk

specificity

different rules can set the same property

```
a:visited {
  color: purple;
  font-size: 1.2em;
}
#start-link {
  color: orange;
}
```

which one gets chosen?

CSS uses the "most specific" selector that matches an element

In our example, the browser will use `color: orange` because IDs (like `#start-link`) are more specific than pseudoclasses (like `:visited`)

CSS can mix properties from different rules

TRY ME!

```
a:visited {
  color: purple;
  font-size: 1.2em;
}
#start-link {
  color: orange;
}
```

it'll use this font-size

but use this color because #start-link is more specific

how CSS picks the "most specific" rule

a selector with element names

```
body div span a {
    color: red;
}
```

loses to

a selector with .classes or :pseudoclasses

```
.sidebar .link {
    color: orange;
}
```

loses to

a selector with an #id

```
#header a {
    color: purple;
}
```

loses to

an inline style

```
style="color: green;"
```

loses to

an !important rule*

```
color: blue !important;
```

!important is very hard to override, which makes life hard for your future self!

https://wizardzines.com/comics/css-specificity/

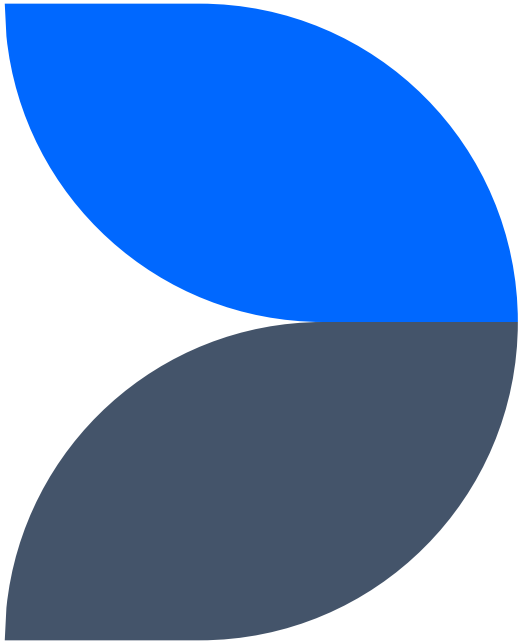
Fork trial

1. Open https://gitlab.com/dawson-scsm-cohort-2026/sf3-pdge/assignments/pre-assignment-tester
2. Click the Fork button in the top left.
3. In **Project URL**, select **dawson-scsm-cohort-2026/sf3-pdge/section#/YourName**
4. Click **Fork Project**

You should be ready for the lab!

Challenge:
CSS Diner

https://flukeout.github.io/



Homework

- Prepare for the first lab assignment: See Moodle for instructions
- CSS Diner: until at least Level 11
- Quiz #3 next week on these slides

PDGE Week 4: Forms

Spot the bugs

```
1. <!DOCTYPE html>
2. <html lang="en">
3.   <head>
4.     <title>Tab name<title>
5.     <meta charset="utf-8">
6.     <h1>Welcome to my website!</h1>
7.   </head>
8.   <body>
9.     <!-- TODO: insert more HTML here >
10.  </body>
11.</html>
```

Discuss what you studied for the quiz (3min)

Agenda

- Quiz #3
- Specificity practice
- Forms
- GET + POST
- Class coding
- Client-side validation
- Homework: Test #1 in two weeks

Specificity & cascading order – what colours will the list text be?

```
.main {
  color: red;
}
.special {
  color: orange;
  font-weight: bold;
}
li {
  color: green;
}
```

```
<ul class="main">
  <li>Item One</li>
  <li>Item Two
    <ul>
      <li>2.1</li>
      <li>2.2</li>
    </ul>
  </li>
  <li>Item Three
    <ul class="special">
      <li>3.1
        <ul>
          <li>3.1.1</li>
          <li>3.1.2</li>
        </ul>
      </li>
      <li>3.2</li>
    </ul>
  </li>
</ul>
```

Form-associated elements

<form>

<input/>

<label>

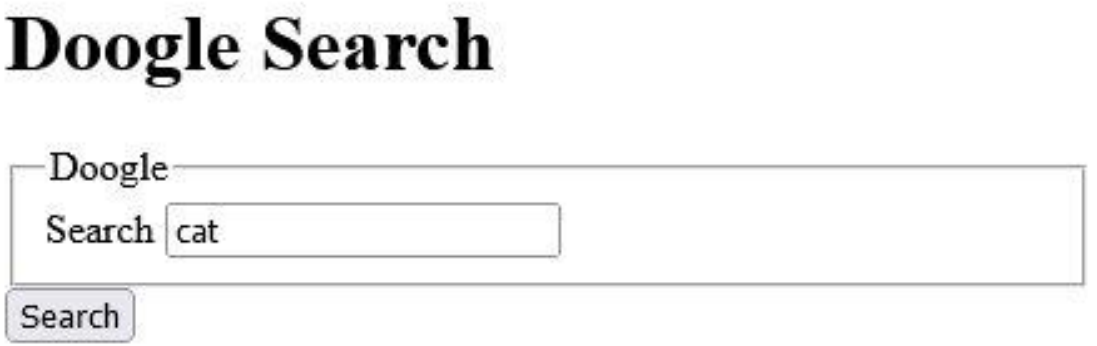
<button>

<fieldset>, <legend>

<textarea>

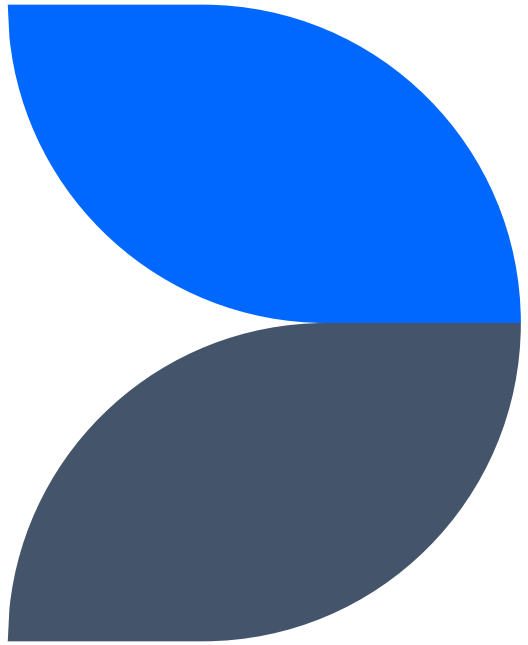
<select>, <option>

Forms
Let’s start here:



Okay but how do you make the button work?

Form submission



<form> attributes

Two important form *attributes*:

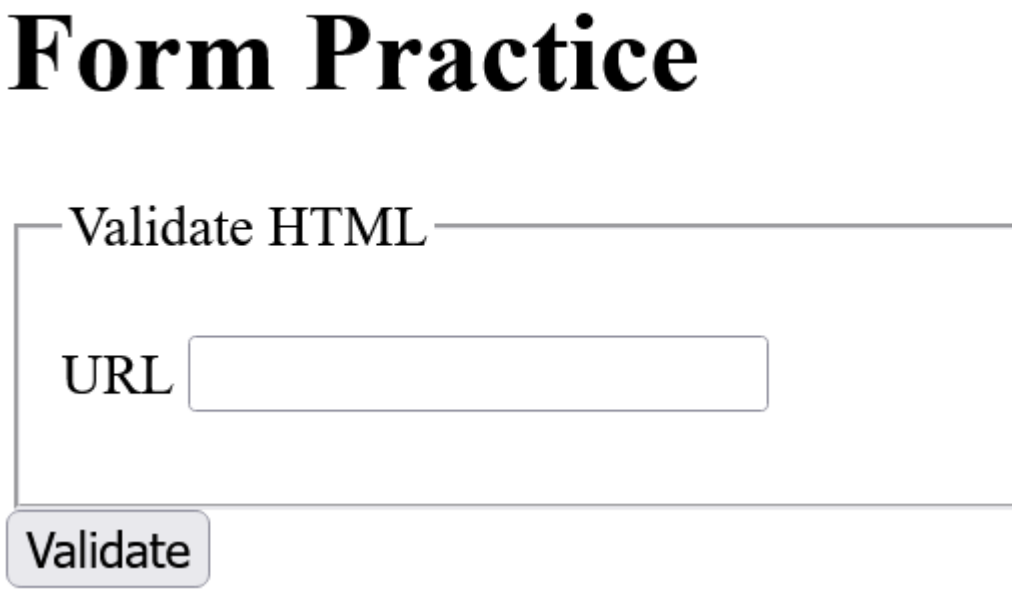
```
<form action="path/to/script" method="get">
…
</form>
```

- The **action** attribute defines which script on the web server the form data will be sent to when it is submitted
- The **method** attribute defines which type of HTTP request will be sent

So, to submit a form, you need a server!

Let’s borrow Google’s server:

Exercise: Borrow the W3C Validator server



Method: HTTP GET or POST?

GET: we are **requesting information from** a server, and our arguments go into the URL

Example HTTP request to W3 Validator:

```
GET /nu/?doc=https%3A%2F%2Fsonic.dawsoncollege.qc.ca
```

Corresponding URL:
`https://validator.w3.org/nu/?doc=https%3A%2F%2Fsonic.dawsoncollege.qc.ca`

Since arguments go into the URL, the request can be bookmarked, or the link shared

Method: GET or POST?

POST: we are **sending information to** the server, and our arguments go into the request header

Example HTTP request to Dawson's server sonic:

```
POST /users/
lastname: Lopez
firstname: Lei
```

Corresponding URL: https://sonic.dawsoncollege.qc.ca

The arguments aren't visible in the URL!

Method: GET or POST?

When does it make sense to make a GET request?

What situations require a POST request?

Let’s build an HTTP request

HTML

```
<form action="users.js" method="get">
  <input type="text" name="lastname" />
  <input type="text" name="firstname" />
</form>
```

Request

```
GET /users/?
lastname="Lopez"
firstname="Lei"
```

What’s the corresponding URL?

REQUEST

```
GET /users/?
lastname=Lopez&
firstname=Lei&
country=Canada
```

Corresponding URL, if the server is located at sonic.dawsoncollege.qc.ca

`https://sonic.dawsoncollege.qc.ca/users/?lastname=Lopez&firstname=Lei&country=Canada`

The part of the URL after the ? Is called the **query string**

What’s the corresponding URL?

REQUEST

```
GET /students/?
id=1234567&
username=brodie&
program=Fisheries&
```

What’s the corresponding URL if the server is at https://catcollege.quebec ?

What’s the corresponding URL?

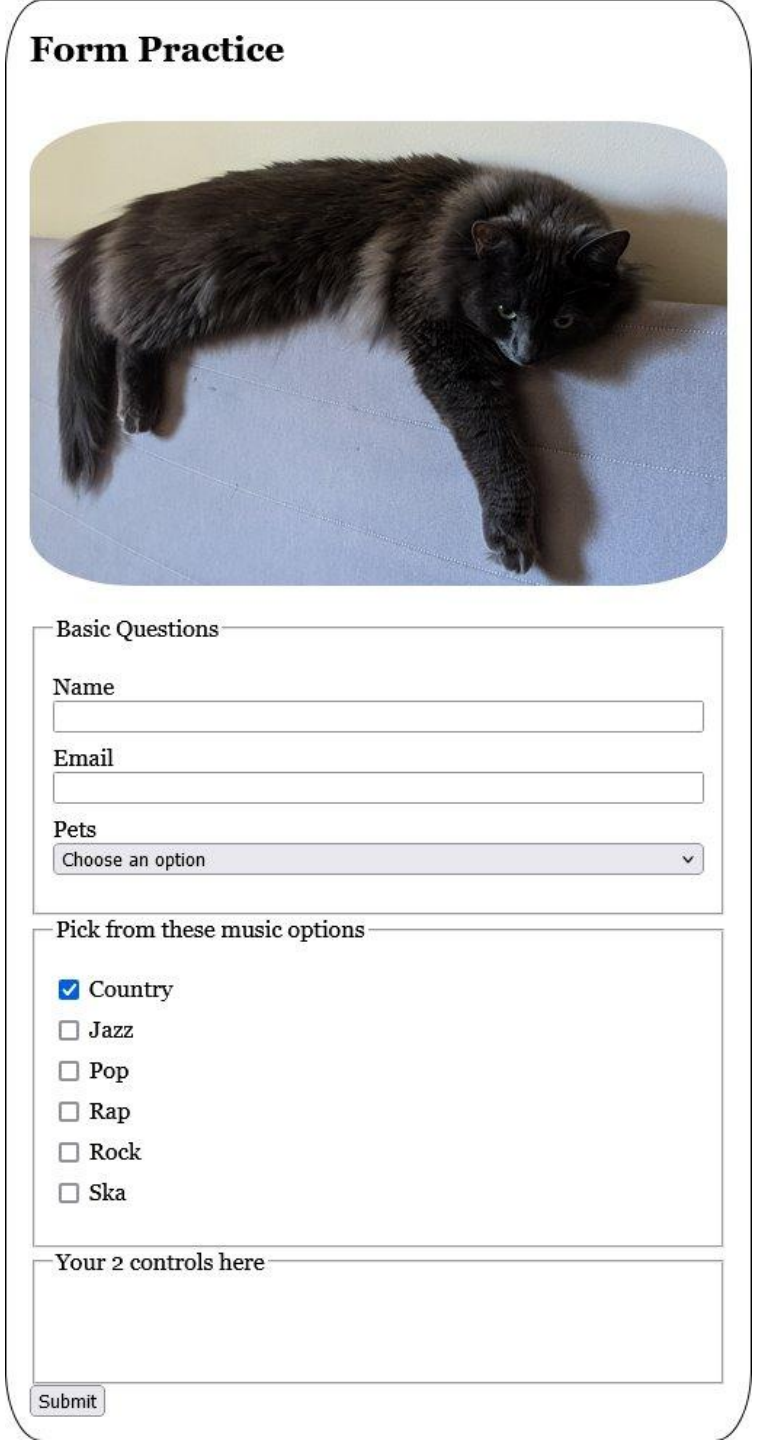
REQUEST

```
POST /students/
id=7654321
username=fitz
password=itsatrap
```

What’s the corresponding URL if the server is at https://catcollege.quebec ?

Download
big-form.html
+
brodie.png

Goal →



Form controls

Definition: Elements that take input from a user

<input/>

<textarea></textarea>

<select><option>...</option></select>

label

For usability and accessibility, we include an explicit label for each form control.

- The **value** of the for attribute on all <label> elements is the **id** of the form control with which it is associated
- When a user clicks or touches/taps a label, the browser passes the focus to its associated input

```
<label for="name">Name:</label>
<input type="text" id="name" name="user_name" >
```

Input

```
<input type="text" name="user-name" id="name" />
```

The input element is a field that the user fills/selects

There are many type attribute values. Here are some:

- text – single line of text
- email, tel
- radio, checkbox
- date, month, time, week
- number, range

General <input> tag attributes

type – the type of input requested from the user (defaults to text)

name – the name of the form control, it is submitted to the server along with the input entered by the user

id – unique identifier useful for the label **for** attribute

required – user must enter something

placeholder – text that is shown in the control before the user types anything

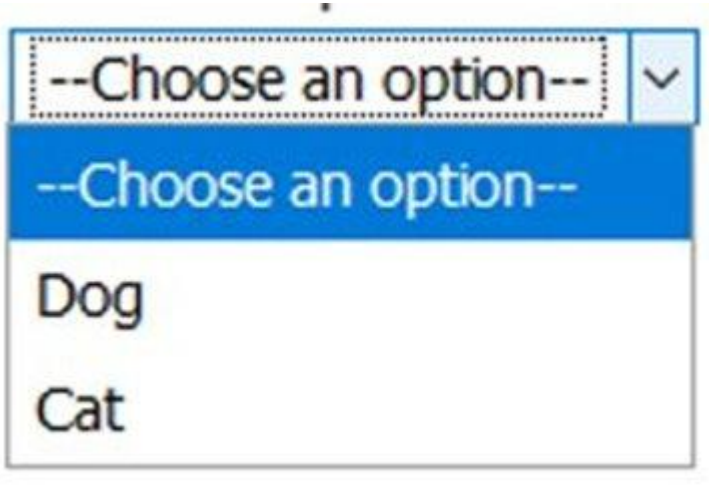
- Useful to give the user a little hint on the expected data (e.g., an email control might have "e.g., person@example.com")

select

How we provide a dropdown menu of options

Basic usage:

```
<label for="pet-select">Choose a pet:</label>
<select name="pets" id="pet-select">
  <option value="">--Choose an option--</option>
  <option value="dog">Dog</option>
  <option value="cat">Cat</option>
</select>
```



textarea

Like an **<input type="text">** but used for multiple lines of user text input

```
<label for="message">Message:</label>
            <textarea id="message" name="user_message" cols="30" rows="5"
                                                placeholder="Type here"></textarea>
```

Message: Type here

button

```
<button type="submit">Send your message</button>
```

3 possible type values: submit, reset, or button.

- **type="submit"** sends the data to the webpage specified by the form's action attribute
- **type="reset"** resets all the form controls to their default values
- **type="button"** does nothing by default! We would have to code it with JavaScript!

Structuring a form – fieldset and legend

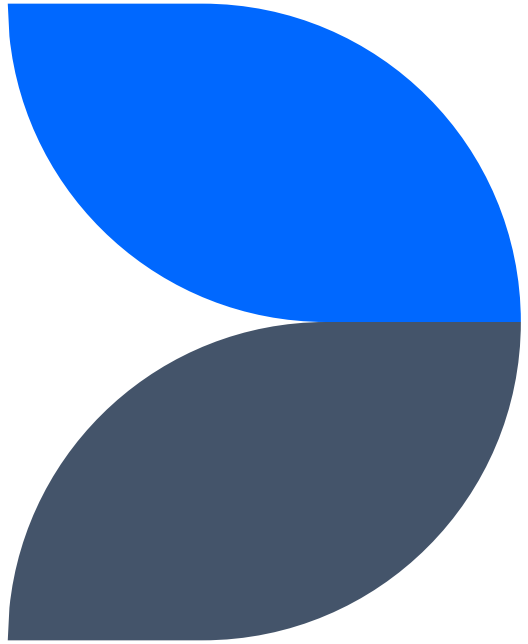
<fieldset> element creates a **group** of form controls that share the same purpose, for styling and semantic purposes.

<legend> element adds a label to the fieldset

The text content of the <legend> formally describes the purpose of the <fieldset> it is included inside.

Client-side form validation

Help the user enter the right data!



Why validate a form?

- To try to send the right data in the right format so our server-side code works
- Protect user data (require long passwords)
  - Type password with stars!
- Protect server (web security!)

Why validate a form?

required

- Forces field to have a value

minlength="8" & **maxlength="15"**

- Requires a text **input** or **textarea** to have a minimum/maximum character length

min="1" & **max="12"**

- Validation for a number input
- Greater than a **min**imum value
- Less than a **max**imum value

Form validation in CSS

:valid – Allows us to style an input that's valid

Example:

```
input:valid, textarea:valid {
    background-color: palegreen;
}
```

:invalid – Allows us to style an input that's not valid

:focus – Styles an element that's clicked on (in focus)

Example:

```
input:invalid:focus { background-color: red; }
```

Homework

- Read feedback on A1
- Study for Test 1 (in two weeks) – includes today's material
- Upload your script from SF1 Intro to Programming Assignment 1 to Moodle

PDGE Week 5 Class: Intro to JS + Review

Agenda

- Common Assignment Issues
- Intro to JavaScript

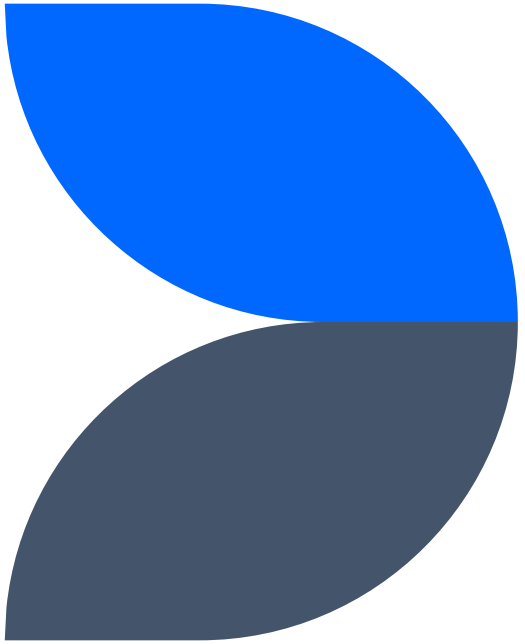
- Review Kahoot!
- Extra Practice problems
- Makeup Quizzes (for students who have booked with me)

Common Assignment Issues

- Design: differences not explained in enough detail
- Git: commit messages should describe action
- Media: some images were captioned as in the public domain when they were not!
- HTML: Vague title (tab text)
- CSS: No backup font-family

JavaScript

What was going on in **calculate.js**?



Components of a web page

HTML

Content of the page

- Information
- All the other files required

CSS

Appearance & layout of the page

- Colours, layout, font, animation

JavaScript

Behaviour of the page

- Validation, interaction, user input

Assets

Separate files included in the page

- images, videos, fonts

Consider Python... What characteristics make up a programming language?

LIST AT LEAST 5 THINGS

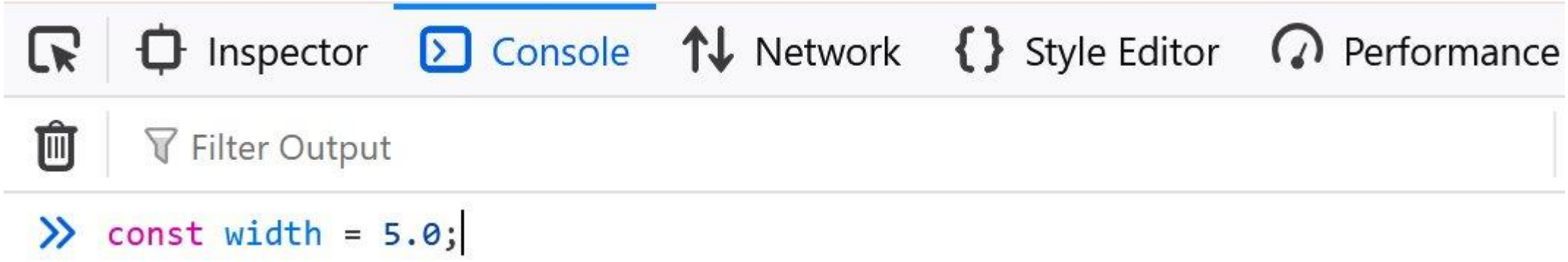
What is JavaScript?

- High-level programming language written for humans
- Similar constructs: variables, conditions, if statements, functions
- **Dynamic typing** – you do not declare variable types and variables can be reassigned something of another type
- Not pre-compiled
- **Pro:** Can easily run on the browser
- **Con:** must end lines in a semi-colon ;

How easy is it to try in the browser?

Super easy, barely an inconvenience!

Find the **Console** from **Inspect** dev tools




```
const width = 5.0;
const height = 3;
const rectangleArea = width * height;
```

Types

- What types does Python have?

In JavaScript:

- **number** – integer or floating type
- **string** – one or more characters
- **boolean** – true or false
- **undefined** – used for unassigned variables
- **object** (a more complex type)
- **null** – absence of object values

Challenge: Use the typeof() function to output each of these types.

Ways to declare a variable in JS

1. **const**
   For variables that will not change, use this by default!

2. **let**
   For variables that will change ☺

3. ~~**var**~~
   The unsafe, old version of let DO NOT use
   Most common in generated code ☹

How do you add JS to your website?

In HTML, in the **head** section:

```
<script src="script.js" defer></script>
```

When the browser receives HTML, it constructs a model of the HTML.

When it reads a **<link>** element, it loads the file (like a stylesheet) to apply to the page.

When it reads a **<script>** element, it loads the JavaScript file to apply to the page.

defer tells the browser to wait to run until the HTML and other files are all loaded

How do you add JS to your website?

In HTML, in the **head** section:

```
<script src="script.js" defer></script>
```

This is the only accepted way to use JavaScript in this course.

Do you recognize the **src** attribute?

Note: There are other ways to use JS from an HTML file (embedded within head, inline event handlers), but these ways are not recommended. It is better to have a ***separation of concerns***, with HTML and JS separated. DO NOT submit these other ways in this class.

Starting a js script

```
'use strict';
```

This declaration will make sure that code is run in strict mode.

strict mode? 🤔

- “Avoid silly mistakes” mode
- JavaScript was designed in a way that is easy to make mistakes ☹
- Strict mode makes it easier to find bugs!

Starter script.js

Using what you know of how to write functions, fill in the missing function that does this:

Calculate and return the cost of the flight, assuming that it is the input **fee** plus $2 dollars per **kilometre** (also an input).

Try the button to test if it works!

Starter script.js

What do you think these lines do?

```
const firstButton = document.querySelector("button");
firstButton.addEventListener("click", showFlightCost);
```

Starter script.js

What about these lines?

```
const fee = document.getElementById("fee").value;
const km = document.getElementById("km").value;
const cost = calculateFlightCost(fee, km);

document.querySelector("output").innerText = cost;
```

Review Kahoot!

Go to kahoot.it or QR code ☺

1. Spot & fix the bugs
2. Draw the form
3. Write the URL sent to the server

```
<form action="https://example.com/subscribe" method="get">
  <label for="choice">Choose an option:</label>
  <select name="choice" id="choices" >
    <option value="option1">Option 1</option>
    <option value="option2">Option 2</option>
    <option value="option3">Option 3</option>
  <input type="checkbox" id="filter" name="filter" value="true">
  <label for="filter">
  <button type="submit">Submit</button>
</form>
```

More drawing & labelling practice

1. Draw a form with at least 3 form inputs, then:
   Trade with a peer and have them label what elements would be needed, along with the attributes.
2. Draw out the box model of an element with the corresponding CSS declarations.
3. Label a URL with its 5 parts.
4. Label the 4 components of a CSS ruleset.
5. Label the 5 components of an HTML element

Prep for A2

Homework

- Prepare for Assignment 2
  Research your formula:
  -What inputs does it need?
  -What units are they in? Min/max values?
  -Find/create an image relating to it
- Study for Test 1 – check reference sheet
- Submit Quiz Revisions ☺

PDGE Week 7 Class: More JS

HTML docs are a (family) tree

child element

HTML elements nested directly inside another element (their parent element)

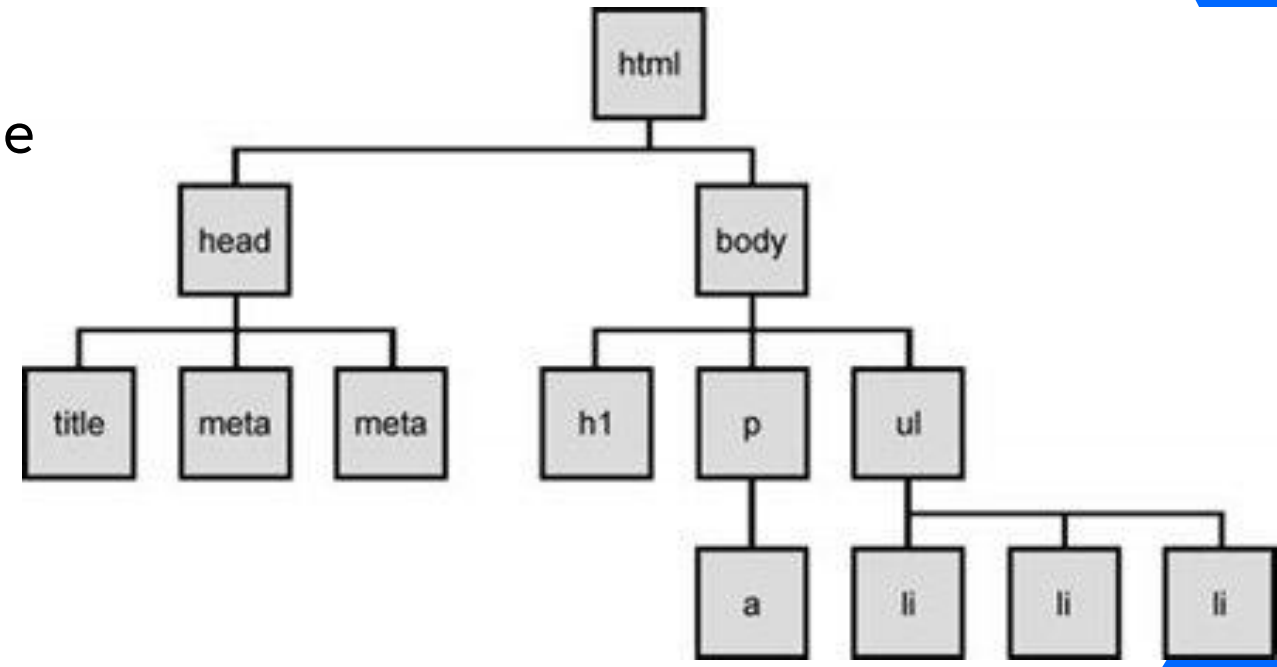
parent element

An HTML element that contains other elements (their children)

descendant

Any HTML elements that are nested in another element (can be deeper than a child element)

Q: What are the descendants of body?



Draw the Tree for this code: (3min)

```
<body>
      <header>Header</header>
      <main>
            <nav>
                  <ul>
                        <li>Home</li>
                        <li>Next</li>
                  </ul>
            </nav>
            <article>
                  <section>Section 1</section>
                  <section>Section 2</section>
            </article>
      </main>
</body>
```

Using the Document Object Model (DOM)

The DOM is the **programming interface** that JavaScript uses to get and change the contents of the HTML document

The window, document, all elements and attributes and text nodes are available as **objects**

The `document` and `window` objects are the objects you will use often

- `window` represents the browser (kind of)
- `document` represents the html document

Try it in console! (2min)

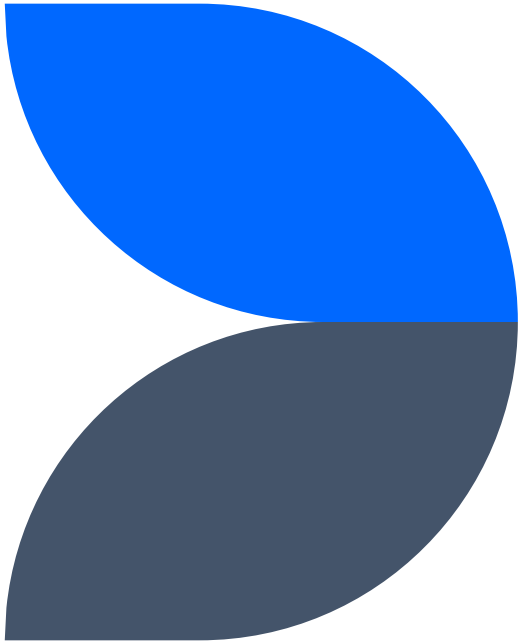
- Get information from the browser:

- `window.innerHeight` - try resizing your window, what happens?
- `window.location` - what is this?

- `document.doctype`
- `document.title`

Query Selector Practice

Please download the starter code and **assets** for Week 7!



Fun in browser Console! (10min)

Without adding new id or class attributes, write the JS to:

- Store the first image into a non-changing variable called **firstImg**
- Store the second fieldset into into a non-changing variable called **secondFieldset**
- Check if the the **pop** checkbox is checked
- Get the user input for **Name** and make it lowercase
- Change the **Pets** label text to **Animals**
- Add an exclamation point **JS Practice** h2
- Change the first image from **brodielee.jpg** to **brodie-nap.jpg**

How to get elements from the DOM

Two helpful functions:

- Find the **first** element for a given selector – use `document.querySelector('cssSelector');`
- Find a list all elements for a given selector – use `document.querySelectorAll('cssSelector');`

  You can get a specific element with array/list/collection syntax [#]

  e.g., `document.querySelectorAll('h2')[0]` -> finds the first h2 element in the page

Terminology Check!

Using properties

Accessing

let content = paragraph**.innerText;**

Assigning

paragraph**.innerText =** "new text";

Using methods/functions

Getting

let url = image**.getAttribute**('src');

Setting

image**.setAttribute**('src', 'diff.png');

Terminology Check!

Using properties

.innerText is like a variable that you can access and change its value directly

Using methods/functions

.getAttribute() and .setAttribute() are *functions*.

You need to call the functions with parameters to get or change the values.

Getting values

element.getAttribute('attributename');

➔returns the value of the given attribute

➔Try in console: getting the **alt** attribute of the image on the page

element.innerText;

➔returns the text content of an element

➔Try in console: getting the text content of the button

Setting values

element.setAttribute('attributeName', 'attributeValue');

➔Sets the attribute of an element to 'attributeValue’

➔Try in console: Set the **href** value of an **a** element to the Dawson homepage

element.innerText = 'value';

➔Sets the text content of an element to 'value';

== vs === 

There are two types of comparison operators in JS

The **equality** operator == checks if the values are equal. It will convert and compare values even if they have different types

The **strict equality** operator === checks if the values are equal AND they have the same type

Let’s try it out – What do you think?

'1' == 1

'1' === 1

'1' === "1"

5 == 5.0

5 === 5.0

0 == false

0 === false

What do you think?

'1' == 1 true

'1' === 1 false

'1' === "1" true

5 == 5.0 true

5 === 5.0 true

0 == false true

0 === false false

falsey and truthy values

A **falsey value** is a value that is treated as false when used in a boolean expression (using non-strict equality)

Examples of falsey values:

- false
- 0 and -0
- "" (empty string)
- NaN (not a number, which has the type of number 🤔)
- undefined (an uninitialized variable)

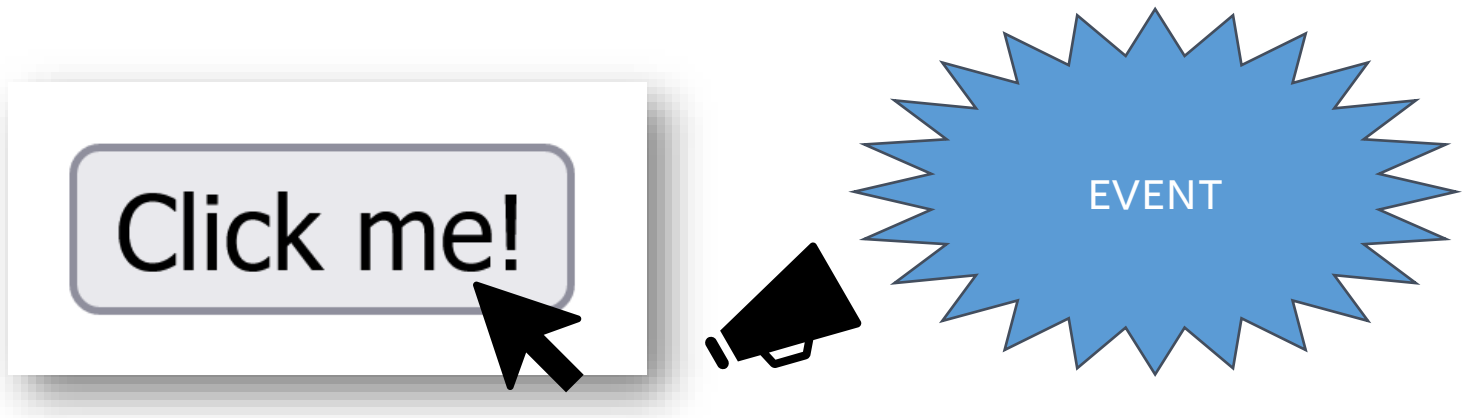
Everything that is not falsey is **truthy**!

LESSON: ALWAYS USE STRICT EQUALITY IN JAVASCRIPT!!!

Event-driven Programming

1: User interacts with page

Event-driven Programming



1: User interacts with page

2: User action produces an event

Event-driven Programming

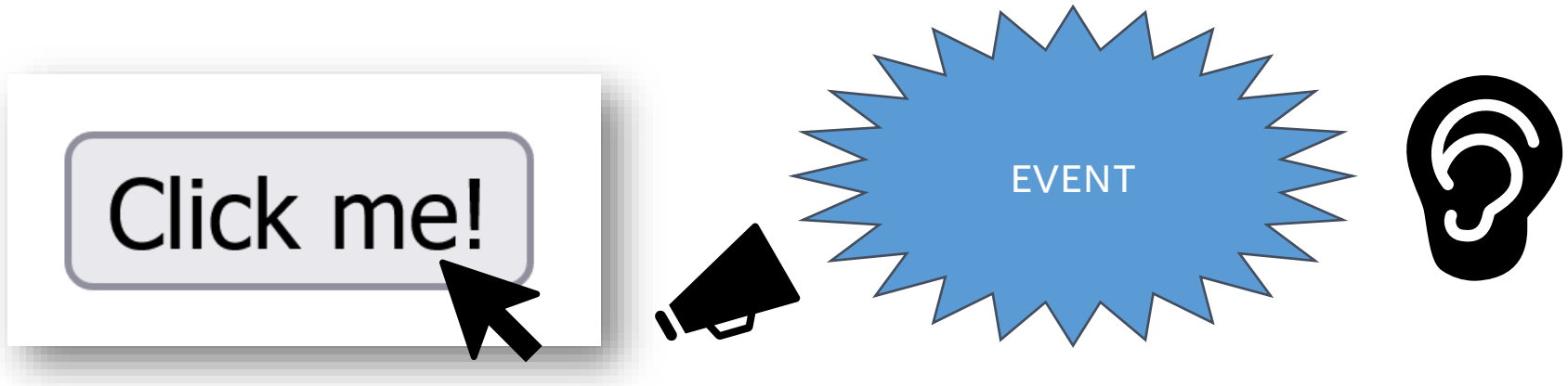
We want some sort of response that makes a change in the webpage!

1: User interacts with page

2: User action produces a browser event

Event-driven Programming

3: An event listener added by a developer “hears” & responds to the event

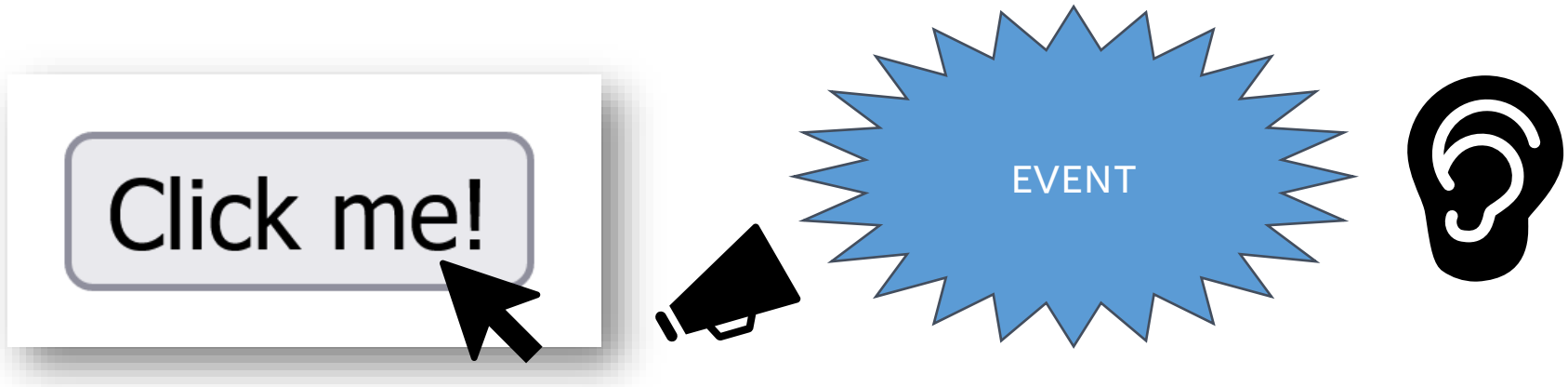


1: User interacts with page

2: User action produces a browser event

Event-driven Programming

3: An event listener added by a developer “hears” & responds to the event



1: User interacts with page

2: User action produces an event

4: The webpage changes in some way 👏

Template for handling events

```
// find the element (like a button) that will produce the event
let element = document.querySelector('selector');


// register an event listener on the element
// set the function that will handle the event
element.addEventListener('eventName', functionName);


// code the event handler function that is called when the event is
produced
function functionName() {
  …
}
```

Task: Code a click event on the JS Practice h2 heading that adds an exclamation point each time it is clicked

Unpack the addEventListener code

```
//listen for the event, and register the event handler
button.addEventListener('click', updateText);
```

The HTML element (or its ancestor) that will fire the event

Unpack the addEventListener code

```
//listen for the event, and register the event handler
button.addEventListener('click', updateText);
```

Name of the event as a string.

There are different events possible, for example:

- The user selects, clicks, or hovers over the element.
- The user chooses a key on the keyboard.
- A form is submitted.

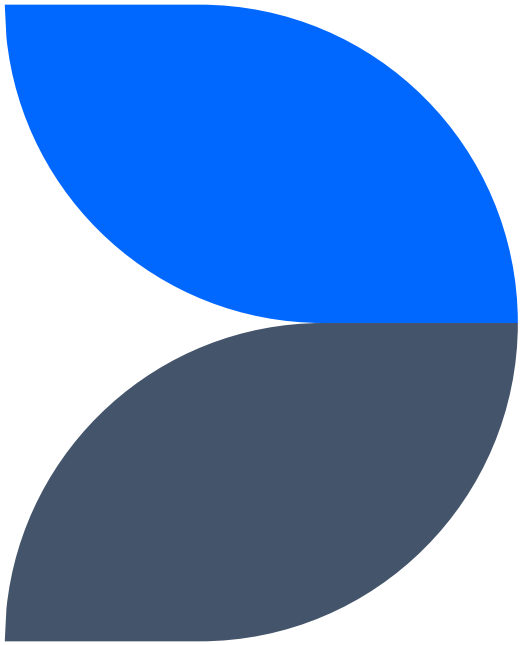
Unpack the addEventListener code

```
//listen for the event, and register the event handler
button.addEventListener('click', updateText);
```

Name of the function that will handle the event
Notice **NO** parentheses – you are not **calling** the function.

Task: Cycle images on click (5min)

Code the JS to cycle between the two Brodie pictures in assets when the image element is clicked.



Toggling a class

In JS, the **classList** is the list of **class=** attributes applied to an element ☺

Example usage:

CSS

```
.rotate {
  transform: rotate(180deg);
}
```

JS

```
<element>.classList.toggle("rotate");
```

Toggling a class

Task: Write a function that finds the first image and flips it by toggling the class **rotate** for the image's classList.

```
function flipImage() {
  const firstImg = document.querySelector("img");
  firstImg.classList.toggle("rotate");
}
```

Keyboard events + event objects!

Let’s tie it to a key press!

```
document.addEventListener('keyup', handleKey);

function handleKey(e) {
    if (e.key === "f") {
        flipImage();
    }
}
```

NOTE: e is the event object that is automatically sent to the event handler!

What else can we do with event objects?

Add the following line to one of your event handlers:

```
console.log(e.type);
```

What does it do?

Add the following line to one of your event handlers:

```
console.log(e.target);
```

What does it do?

With **e.target**, we can update the element that triggered the event!

Challenge: Add an event listener to **document.body** that toggles the **border** class on the element that was clicked on.

What other browser events exist?

And where would I ask you to find them?

Try to trigger 3 different events below:

You'll need to register a new event listener on an appropriate target (element, document, etc.) with one of the event handlers you already have, or a new one!

- `mouseover`
- `mouseout`
- `mousedown`
- `mouseup`
- `contextmenu`
- `keydown`
- `keyup`
- `dblclick`
- `click`
- `load`
- `pageshow`
- `scroll`
- `blur`
- `onfocus`
- `select`
- `wheel`
- `input`
- `copy`
- `cut`
- `paste`
- `resize`

Using a canvas to draw

The canvas is a browser built-in set of HTML and JavaScript that allows us to draw (hello math functions!)

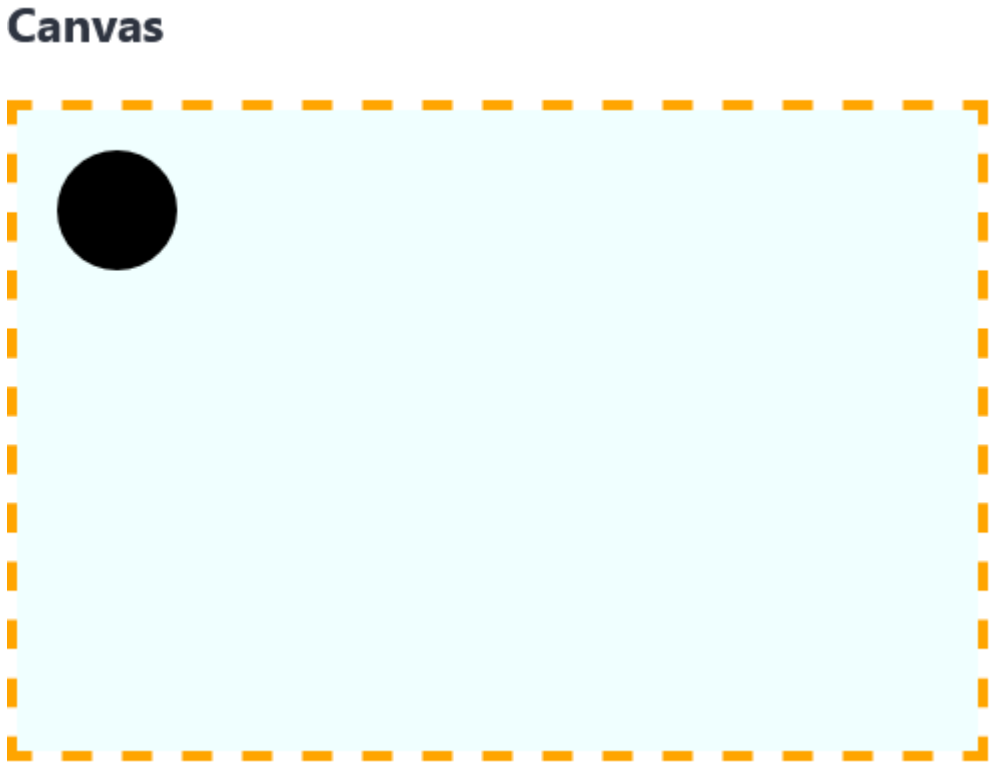
We'll stick with a circle for now ☺

Who wants to move it around??

Using a canvas to draw

1. Add an event listener that listens for the keydown event and will call the **handleKey** event handler.
2. Code the event handler function **handleKey**
   a. Each key (w, a, s, d) must adjust the **x** and **y** value according to the graphics coordinates
   b. At the end of the function, the function **drawCircle** can be called with the new x/y values ☺

Using a canvas to draw



- Add the ability to use the arrow keys in the existing if/else if statements
- Change the colour of the circle to be a gradient from one colour to another
- Try making the circle move faster or slower
- Add a key that increases/decreases the **size** of the circle
- Try changing the circle into another shape!

PDGE Week 8 Class: Backend + Flask

Agenda

- Quiz 5
- Git Feature Flow
- Virtual environments
- Backend: Flask! Jinja!

Quiz #5

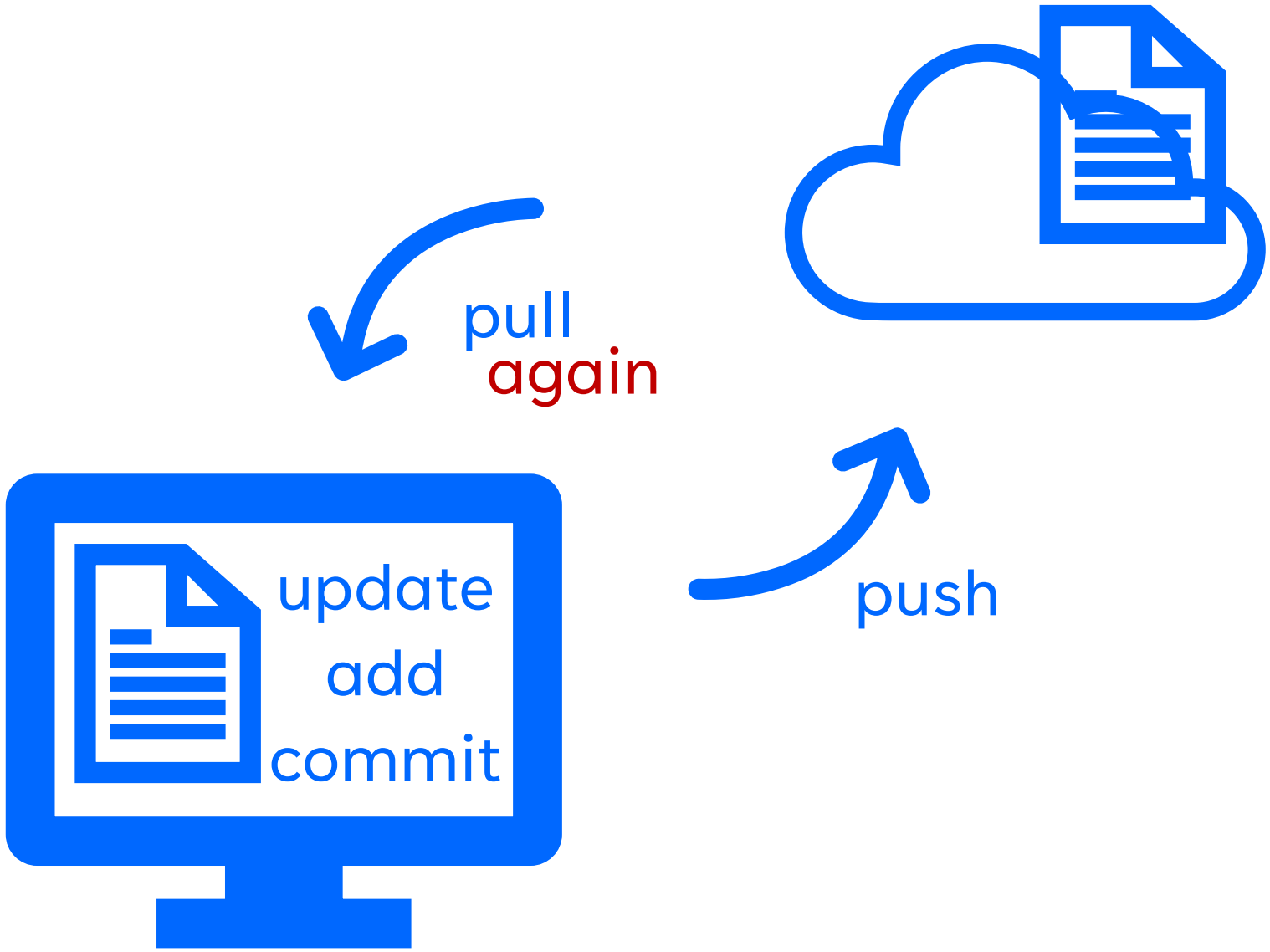
1. Name & explain 1 browser event other than click/keyup/keydown. *(2pts)*
2. Assuming that **e** is an event object, what does **e.target** return? *(1pt)*
3. Draw the DOM tree for this HTML code snippet: *(7pts)*

```
<body>
  <header>Hello World!</header>
  <main>
    <p>My <span>first</span> website.
          Hope you <span>en<span class="capital">joy</span></span>!
    </p>
  </main>
  Thanks for reading!
</body>
```

Git Feature Flow

One way to work on the same codebase together efficiently ☺

Git Workflow



Typical git sequence

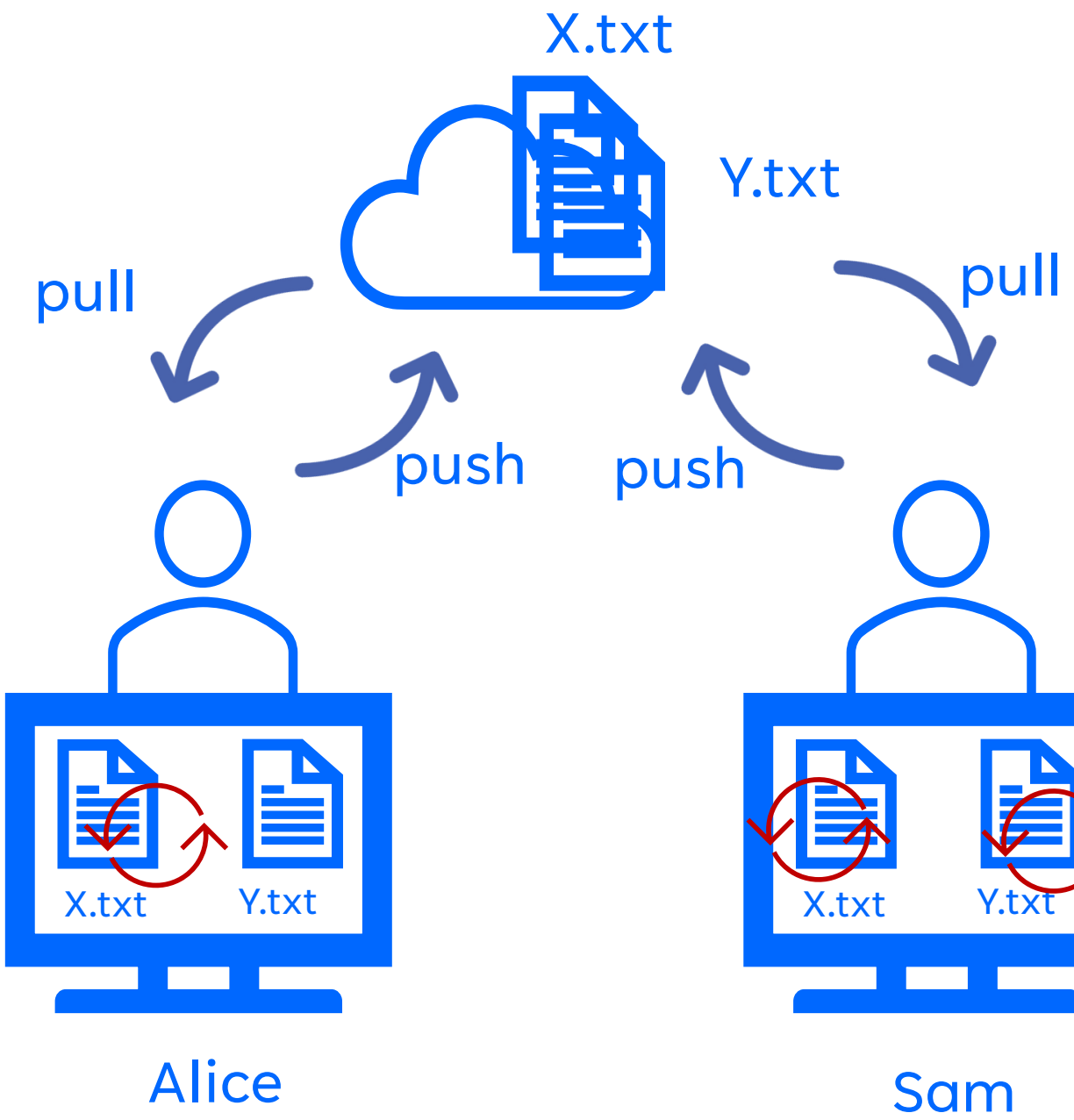
1. **git clone** → Done once per computer to connect the computer to the repository and download everything in the server into a specific folder. If this is not the first time you're working from this computer, then you would do git pull here instead (see step 6)
2. **(outside of git)** Make whatever changes you want. These changes must be within the folder cloned to git in step one
3. **git add *specificFileOrFolder* or git add .** → This adds the file/folders (or everything) to git on your local computer so that it is *tracked*
4. **git status** → To confirm that all the files you want show up as added
5. **git commit -m "Please put a useful message here describing the purpose of the latest work"** → To create a snapshot (on your local computer still!)
6. **git pull origin** → To download any updates made in the server since step 1
7. 🤞 **Cross your fingers** → Hope that there is no merge conflict!
   If so, resolve the merge conflict! (next lecture!)
8. **git push origin** → To upload *your* updates to the server.

Pull scenarios

Typically, 3 different scenarios that can happen during the second pull

- Case 1: The pull works with no changes at all (yay!)
- Case 2: The pull works but involved an automatic merge.
  - Depending on your computer settings, you may receive a confirmation dialog notifying you of this (**MERGE_MSG**)
  - Happens if developers both make *compatible* changes to the same file. For example, if they each *add* to the same file.
- Case 3: The pull requires a manual merge.
  - Happens if developers both make *incompatible* changes to the same file.
  - For example, if they each modify the same line of code – which modification is the desired one!?

Scenario 1



Alice updates X.txt

What happens when Sam “repulls” before pushing their change?

Sam updates Y.txt

Sam now has the new X.txt

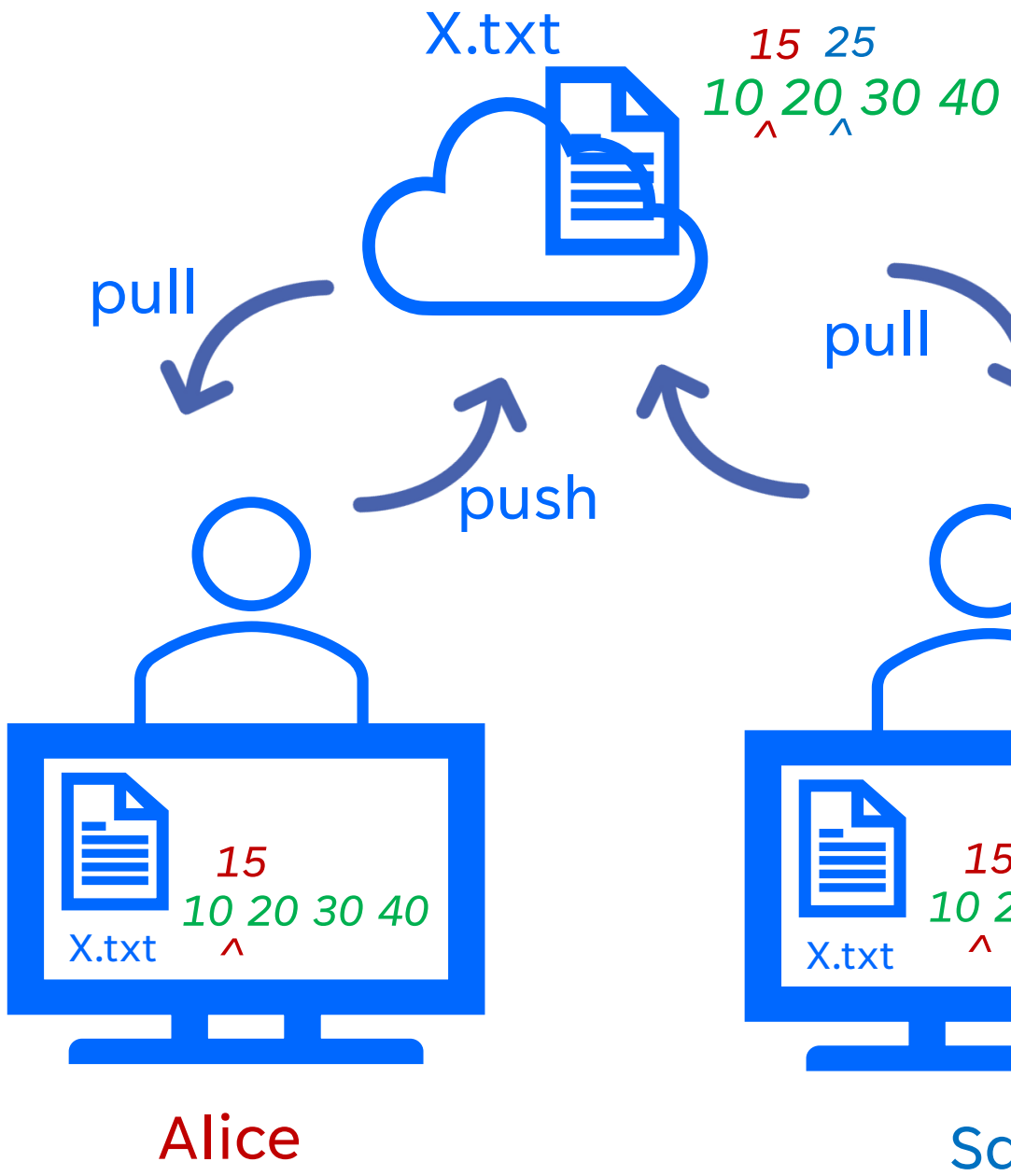
Scenario 1 – working on different files – Automatic merge

- Suppose there are 2 files in the code repository: X.txt, Y.txt
- Alice and Sam both download their individual copies at the same time.
- Alice changes X.txt and pushes the change to the remote server
- Sam changes Y.txt and plans to push the change to the remote server. Before doing so, they repull the latest code from the server.
- In this case, Sam's computer will now have a version of code including **both** changes (i.e. both X.txt and Y.txt are there)

Scenario 2

1. What happens when Sam repulls?

2. What contents would you like to show up on the server?



⚠ If Sam were allowed to push their change to the remote repository, then it would undo Alice's change!

Alice adds 15 between 10 and 20

Sam adds 25 between 20 and 30

Auto-merge!

Scenario 2 : working on the same file - Automatic Merge

Suppose there is a file X.txt in the code repository with the following contents: **10 20 30 40**

Alice and Sam both download their individual copies at the same time. So, on each of their computers the file X.txt contains text: **10 20 30 40**

1) Alice adds the *15* between 10 and 20 and pushes the change.
   - Alice's version is now: **10 15 20 30 40** (which is also the version on the server now)
2) Sam adds the *25* between 20 and 30.
   - Sam's version is now: **10 20 25 30 40**

Scenario 2 : working on the same file - Automatic Merge (continued)

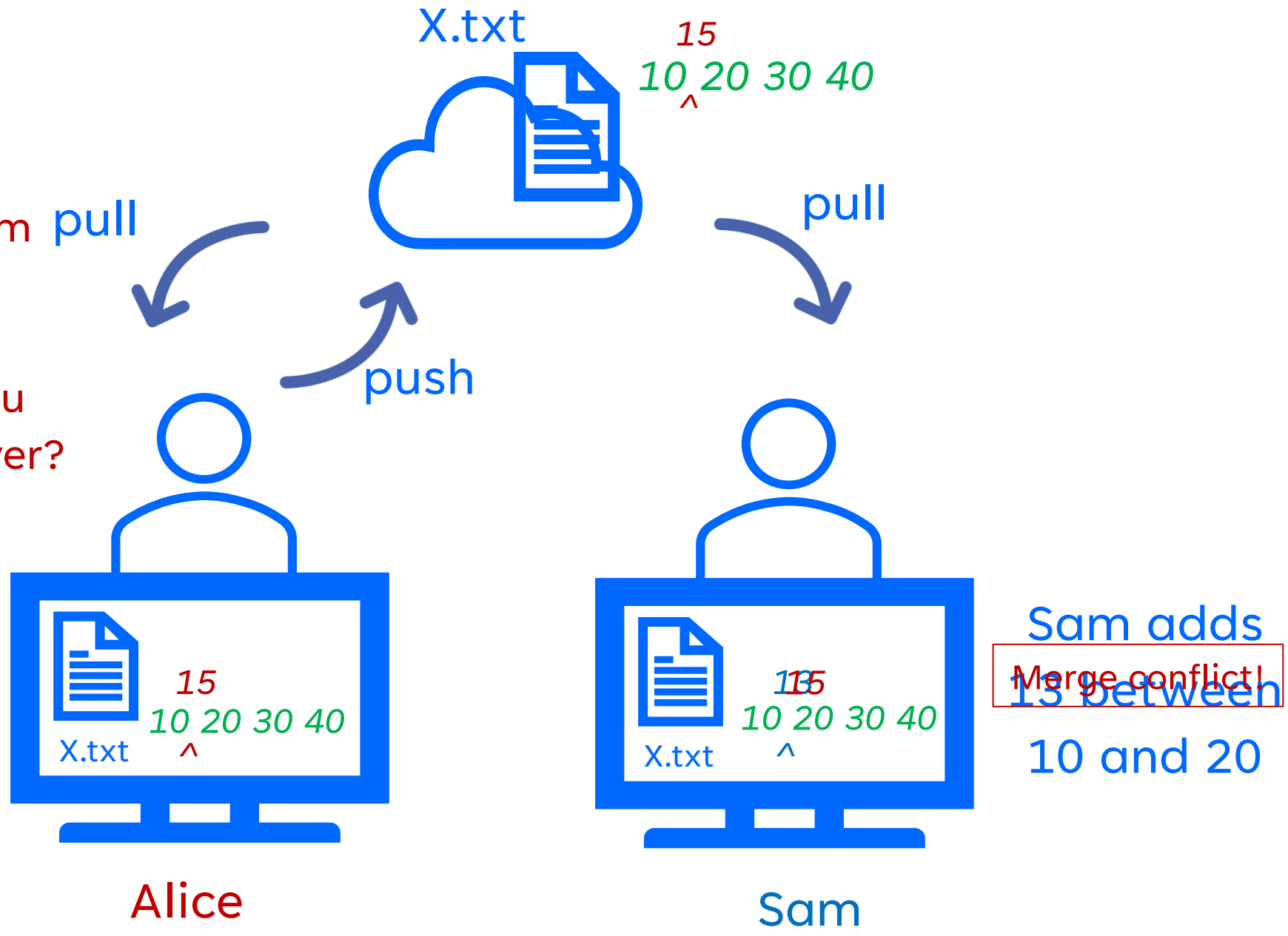
3) If Sam were allowed to push their change to the remote repository, then it would undo Alice's change! So, a warning comes that forces Sam to download the latest code. When this is done, the source repository system will *merge automatically* the two together on Sam's computer.

The source repository system is able to *infer* that the final version of the file should be (10 15 20 25 30 40) and this version will now be on Sam's computer. They can push it to the server without erasing Alice's change!

Scenario 3

1. What happens when Sam repulls?

2. What contents would you like to show up on the server?



Scenario 2 : working on the same file - Conflicting Merge

Suppose there is a file X.txt in the code repository with the following contents: **10 20 30 40**

Alice and Sam both download their individual copies at the same time. So, on each of their computers the file X.txt contains text: **10 20 30 40**

1) Alice adds the *15* between the first 10 and 20 and pushes the change.

   - Alice's version is now: **10 15 20 30 40** (which is also the version on the server now)

2) Sam adds the *13* also between 10 and 20.

   - Sam's version is now: **10 13 20 30 40**

Scenario 3: Resolving a conflicting merge

The source repository system is not able to determine whether the final file should be:

10 13 15 20 30 40

or

10 15 13 20 30 40

Sam needs to *manually* merge the files together as the computer cannot infer which edit makes more sense!

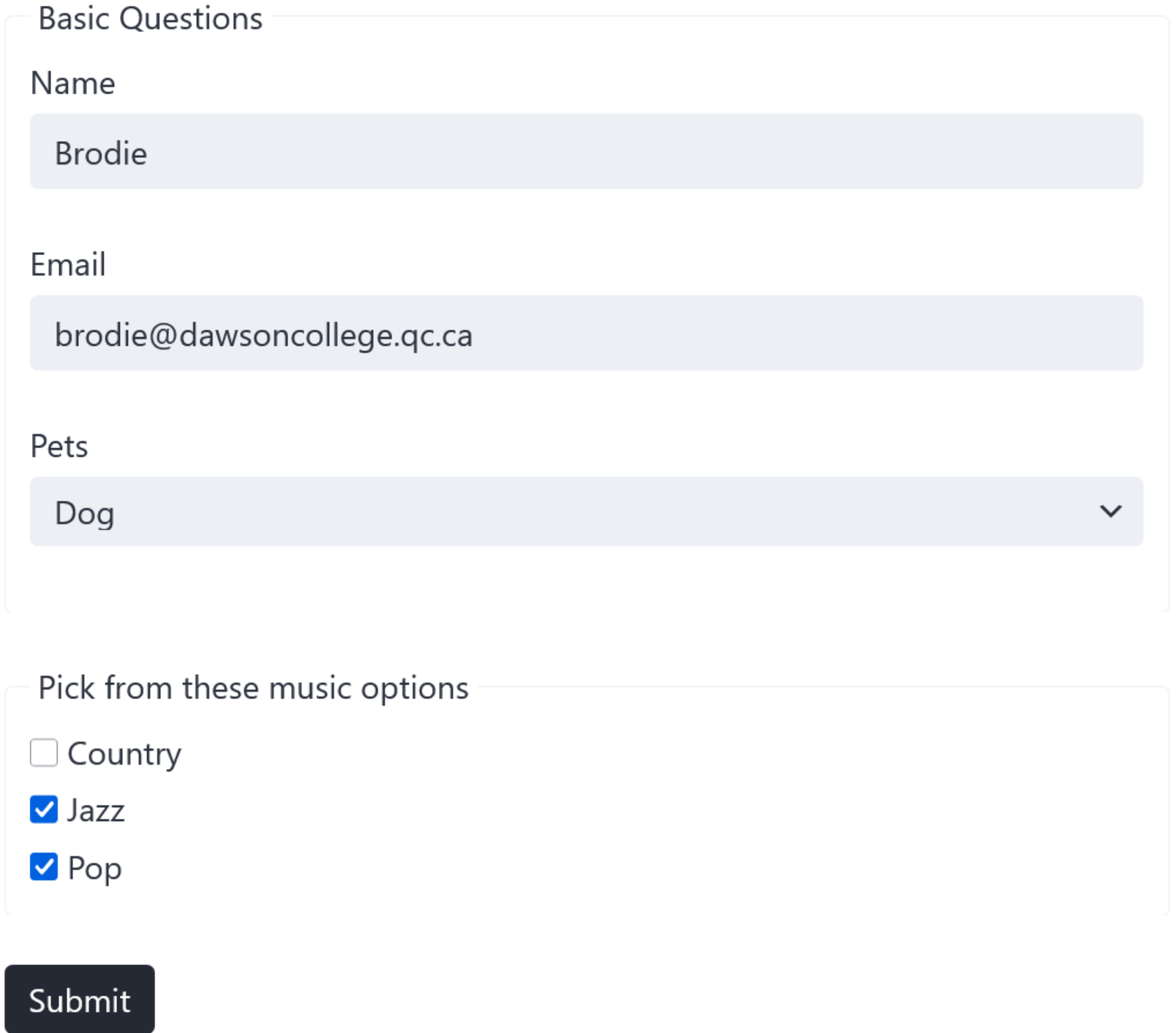
How to avoid merge conflicts

- PULL before pushing
- Small changes: If you make lots of small commits instead of a few big commits, then it is less likely you'll create a new bug. If you do create a bug, then it's easier to identify.
- Rollbacks / reverts: This is a feature within Git that will let us undo a commit that someone else (or ourselves) made (this is reactive but sometimes unavoidable)
- Code reviews: This is the idea of having someone else check over your work *before* you commit it (proactive)

How to fix merge conflicts

- Git will tell you why it could not create the merge for you
- You will need to edit the specific file(s) that it had trouble merging (you can find them by searching for **<<<**
- After editing them, you will create a *new* commit, with those edited files (by using the command git add).
- After committing the change, the merge conflict will be resolved.
- You can then push to the server. When you do this push, there will be *two* commits pushed simultaneously.

What gets sent to the server?



Time to code the backend!

Time to code the backend!

- So... we could process those HTTP requests ourselves
- OR we can use a framework!
- Examples:
  - Django (python)
  - Ruby on Rails (ruby)
  - Express.js (JavaScript)
  - Laravel (PHP)
- This course: **Flask**

Flask

A lightweight Python web framework

Virtual Environments

- A way to make sure we use the correct Python + library versions for each project we're working on!
- **venv** module in Python ☺
- Resource: https://flask.palletsprojects.com/en/stable/installation/#virtual-environments
- Don't forget to activate your environment!

Flask - Hello World

- Create a file called **hello.py**
- Minimal Flask app:

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

- In Terminal, run **flask --app hello run**

What does .route() do?

Allows us to define different pages for our users to visit ☺

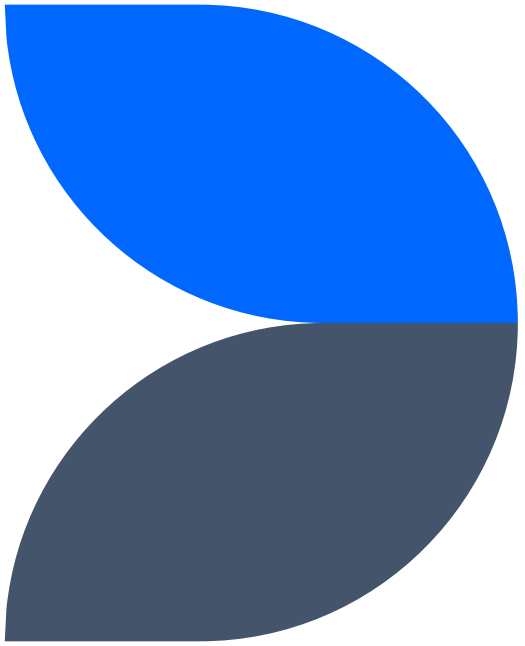
This syntax declares a **decorator**, which adds the function under it to our instance of Flask :)

What was the **bold** part of this URL called?
https://example.com/**pages**?key=value

Now we can call it a route, because we'll be coding the route to the response for a given request!

Download

index.html (bare bones HTML)



Let’s upgrade

Create a new file called **app.py** with these updates:

```
from flask import Flask, render_template


app = Flask(__name__)


@app.route("/")
def index():
    return render_template("index.html")
```

This assumes that **index.html** is in a folder called **templates/**

Let’s personalize ☺

In **templates/index.html**, add **Hi, {{ name }}** to the body text, then update:

```
from flask import Flask, render_templates, request


app = Flask(__name__)


@app.route("/")
def index():
    name = request.args["name"]
    return render_template("index.html", name=name)
```

Open the page

http://127.0.0.1:5000/?name=Brodie

What happens if you go to the plain URL?
http://127.0.0.1:5000/

Oops! **3min Task:** Write an if-else statement that uses the **name** if it's in the request args and **"world"** if not.

JK there’s a better way

```
if "name" in request.args:
    name = request.args["name"]
else:
    name = "world"
```

Becomes:

```
name = request.args.get("name", "world")
```

Let’s attach all this to a form!

In **index.html**, replace the **Hi, {{ name }}** with this form:

And now we need to add a new route to **app.py** ☺

3min Task: Add a new route to **app.py** for the path **/greet** that returns the **greet.html** file

What do we need to make this URL work?
http://127.0.0.1:5000/greet.html?name=Brodie

Client vs Server-side validation

We can add **required** and **minlength**, etc. client-side, but that doesn't prevent the user from deleting those!

Server-side validation is needed to defend against malicious users, while client-side validation is best for helping users.

What's reading {{ name }} ??

A templating library called Jinja

Jinja

Are you tired of typing HTML?

Let’s harness the power of Jinja templating!

Create a new file called **layout.html** in **templates/** with the standard content:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Flask Practice</title>
  </head>
  <body>


  </body>
</html>
```

Are you tired of typing HTML?

Let’s harness the power of Jinja templating!

Create a new file called **layout.html** in **templates/** with the standard content:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Flask Practice</title>
  </head>
  <body>
   {% block body %}{% endblock %}
  </body>
</html>
```

BURN all the boilerplate!

In **index.html**, we can now delete all these lines:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Flask Practice</title>
  </head>
  <body>

  </body>
</html>
```

BURN all the boilerplate!

So now we're left with:

```
<form action="/greet" method="get">
    <input type="text" name="name">
    <button type="Submit">Greet</button>
</form>
```

BURN all the boilerplate!

But now we need to add the Jinja:

{% extends “layout.html” %}

{% block body %}

```
<form action=“/greet” method=“get”>
    <input type=“text” name=“name”>
    <button type=“Submit”>Greet</button>
</form>
```

{% endblock %}

Your turn! Burn the boilerplate for **greet.html**

What about POST?

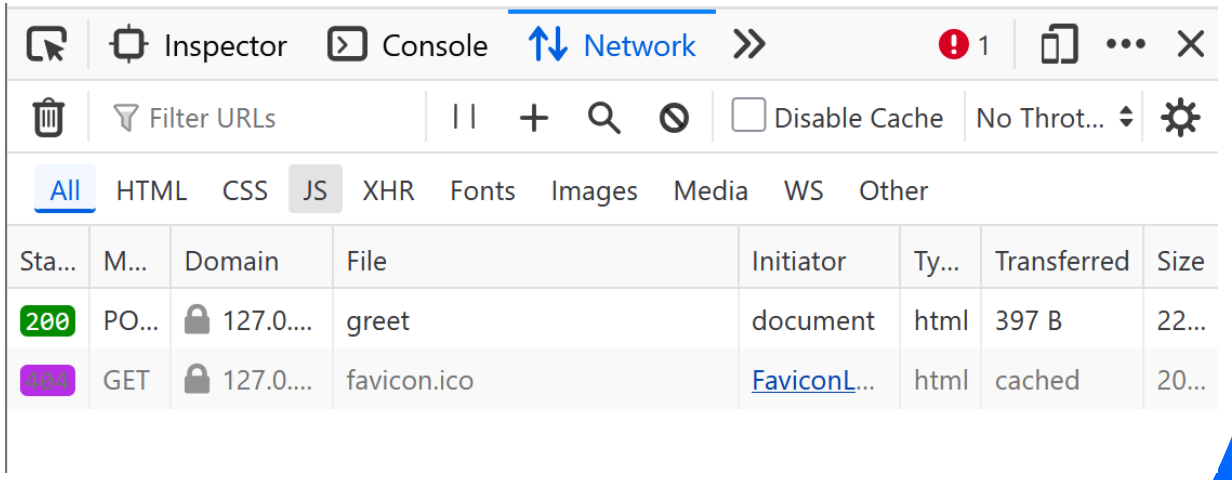
What happens when you change from **get** to **post** in **index.html**?

Oop, let's fix that in **app.py**

```
@app.route("/greet", methods=["POST"])
def greet():
    name = request.form.get("name", "world")
    return render_template("greet.html", name=name)
```

Network tab in developer tools

You can creep on your requests to any website server!



Conditionals in Jinja

We can let Jinja handle the default value!

```
@app.route(“/greet”, methods=[“POST”])
def greet():
    name = request.form.get(“name”, “world”)
    return render_template(“greet.html”, name=name)
```

Conditionals in Jinja

```
{% extends "layout.html" %}

{% block body %}

    Hi,
    {% if name %}
        {{ name }}
    {% else %}
        world
    {% endif %}

{% endblock %}
```

Challenge: Code a backend for our form!

Provided:

- Empty **app.py**
- **static/brodie-nap.jpg** (in Flask, assets are stored in **static/**)

In templates:

- **failure.html**
- Non-Jinja **index.html**
- Basic **layout.html**
- Starter **success.html**

Challenge: Code a backend for our form!

1. Code **app.py** to setup Flask
2. Make sure the **form** in HTML sends a **post** request to **/submit**
3. Code the route for **“/”** to display **index.html**
4. Code the POST to **/submit** that displays **failure.html** if **name** or **email** are missing; otherwise, it displays **success.html** sending the **request.form** to the template
5. Make **index.html** use Jinja templating from **layout.html**
6. Update **success.html** with the **request.form** name, as well as the keys and values, using a Jinja for loop 🔁
7. Update **app.py** and **index.html** to source Pets & Music from arrays setup in **app.py**

References

PDGE Week 12 Class: Presenting Data

Agenda

- Data is a file
- .gitlab-ci.yml
- Deploying a Flask app
- Lightning Talks

All code + data are files

- In A4, when we kill the server, the submitted scores go away ☹
- To store, we need a database
- In the class we'll simulate that with a **csv** file
  - In production, this would be TERRIBLE practice (security risk)
  - But this is for learning ☺
- Clone the starter code & explore, what is the website doing?
- Open **.data/scores.csv**

Save the submission to the csv

```
import csv
```

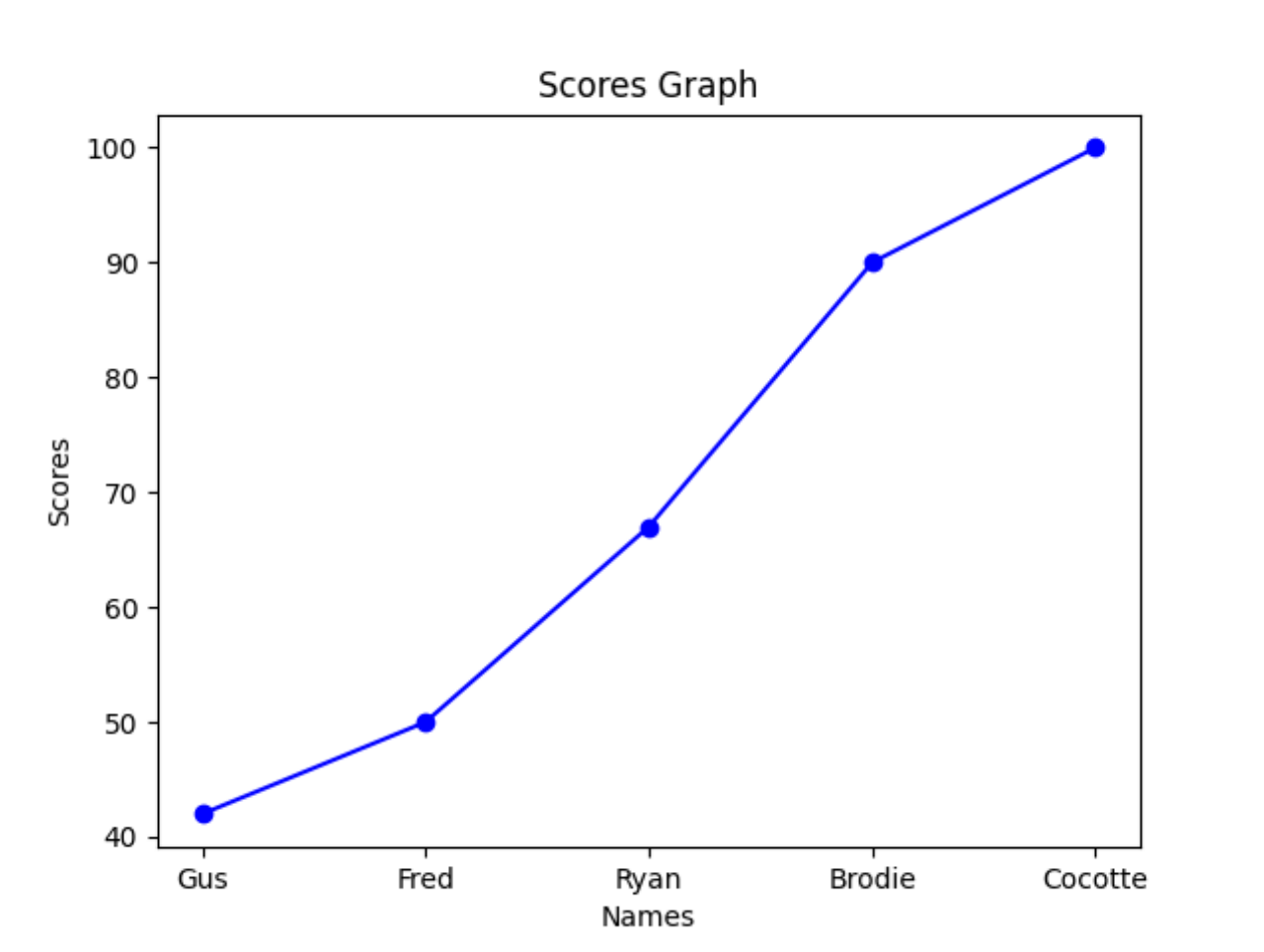
Before we return the **success.html**:

```
with open('.data/scores.csv', 'a', newline='') as csv_file:
        data = csv.writer(csv_file, delimiter=',')
        data.writerow([name, score])
```

Let’s implement the /scores page

```
with open('.data/scores.csv') as csv_file:
    data = csv.reader(csv_file, delimiter=',')
    first_line = True
    scores = []
    for name, score in data:
        if not first_line:
            scores.append({"name": name, "score": int(score)})
        else:
            first_line = False
    sorted_scores = sorted(scores, key=lambda item: item['score'])
```

How about a graph of the data?



How about a graph of the data?

```
import csv, matplotlib.pyplot as plt


def generateScoresGraph(scores):
    name_data = [data["name"] for data in scores]
    scores_data = [data["score"] for data in scores]
    plt.plot(name_data, scores_data, marker='o', linestyle='-', color='blue')
    plt.xlabel('Names')
    plt.ylabel('Scores')
    plt.title('Scores Graph')
    plt.savefig('static/graph.png')
    plt.close()
```

.gitlab-ci.yml

- Why do we need pipelines?
- To validate, test, lint, prepare code before publishing
- How?
- In GitLab, and many other systems, write the instructions in a YAML file
- Infrastructure developers write the code that reads those instructions ☺ (Thank you to them!!)

How to read .gitlab-ci.yml

```
flake8:
  stage: test
  image: python:latest
  before_script:
    - pip install flake8
  script:
    - flake8 app.py
  allow_failure: false
```

- Title of Job
- Stage of Pipeline (defaults: build, test, deploy)
- Image: Sandbox with necessary software
- Things to run before the job script
- The commands to run!
- If the job is allowed to fail for the pipeline to succeed

How to read .gitlab-ci.yml

```yaml
image: busybox
create-pages:
  pages:
    # The folder that contains the files to be expo
    publish: public
  rules:
    # This ensures that only pushes to the default
    # a pages deploy
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
  script:
    - echo 'nothing to do'
```

- busybox is a minimal container
- Title of Job: **create-pages**
- **pages** is GitLab Pages-specific
- **rules** determine when a job is run This one only runs the **create-pages** job on a push to the **main** branch
- No commands to run!

Deploying a Flask app

- What do you need to deploy a Flask app?
- Server
  - computer with flask + libraries installed
  - has your website code
  - discoverable on the Internet/Web
- Unfortunately, that's annoying to manage or expensive ☹

Render

- Free tier! With no credit card needed!
- But apps will sleep
- Cannot handle unlimited requests
- https://render.com/docs/deploy-flask
- Try it!

Lightning Talks (Nov 19/26)

Prepare 4 to 5-min talk on a web dev topic that teaches something new (& hopefully useful) for your peers.

1. See Moodle for the list of ideas (based on your answers)
2. Select your topic by signing up to the form linked on Moodle
3. Pick your timeslot on Moodle
4. Prepare presentation materials (slides/demo/Kahoot/etc.)
   - MUST be doable from the teacher station / accessible via a browser
   - Submit to Moodle: if not, you will have to present sans material
5. PRACTICE, out loud at least a couple times 🗣️
6. Present! Learn from your peers + give feedback ☺

Lightning Talks (Nov 19/26)

Points to cover:

- What is the tool/skill/software for?
- Pros and cons? Alternatives?
- How is it helpful to your peers?
- How can they learn more?

Homework

- A4 Fixup
- Select topic + prepare lightning talk
- Complete Project Phase 1 tasks

PDGE Week 15 Class: Review + Reflection

Agenda

- Mind mapping
- Kahoot!
- Reference sheet
- Self-reflection

Mind mapping by topic/skill: What comes to mind...

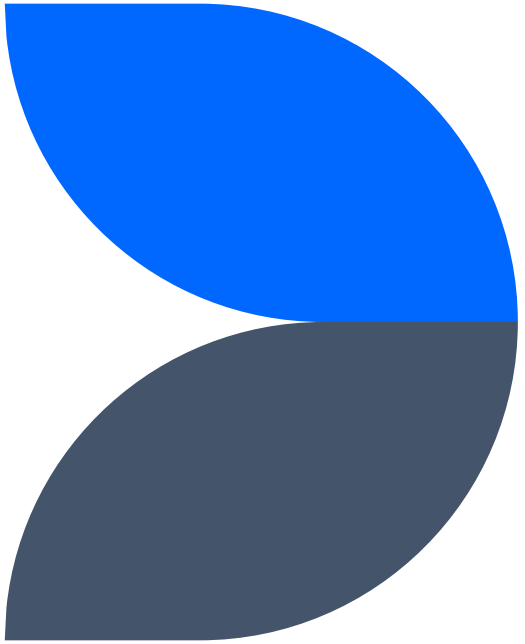
1. Internet connectivity & terminology
2. Internet vs Web
3. Roles of client vs server on the Web
4. Understand DOM tree structure
5. Determine CSS applied (source order & specificity)
6. Browser Dev Tools
7. Accessibility basic checks
8. JavaScript for event-based interactions

Mind mapping by topic/skill: What comes to mind...

9. Receiving, manipulating, displaying data in Flask
10. Git Feature Flow
11. Giving code review, spotting bugs + best/bad practices
12. Interpret .gitlab-ci.yml files
13. Requirements for deployment to a web server
14. Citing external media or content

Kahoot!

last one ☹
https://kahoot.it/challenge/05236802



Reference Sheet

Anything you'd like to add?

REFERENCE

HTML	JS
<!-- -->	.addEventListener
a href="" target=""	.body
article	.classList
body	.getAttribute
br, hr	.includes
button type=""	.parseInt
div	.innerText
em	.querySelector
fieldset, legend	.querySelectorAll
figure, figcaption	.setAttribute
form method="" action=""	.src
h1, h2, h3, h4, h5, h6	.toggle
head	.value
header, footer	.valueAsNumber
html lang=""	
img src="" alt=""	console.log
input type="" name=""	
label for=""	e.key
li	e.target
link href="" rel="" type=""	e.type
main	
meta name="" content=""	**Python/Flask**
meta charset=""	int
ol, ul	render_template
p	request.args
script src="" defer	request.form
section	url_for
select name="", option value=""	
span	
strong	
style	
textarea cols="" rows="" name=""	
title	

Self-Reflection for Project

- Review the project rubric – how would you rate yourself in each category and why? **Provide evidence** in the form of images, commit or file links from GitLab.
  - **Team Agreement & Project Proposal**
  - **Assigned Form & Interaction/server-side Feature**
  - **Final Code + Code Review**
  - **Testing & Documentation**
  - **Accessibility**
  - **Deployment**
  - **Teamwork**
  - **Presentation**
- What new skills or tools did you learn in working on this project? What would you change about the project?
- What advice would you give to future PDGE students for the project or course?