Introduction to Programming

Unit 2: Web Programming

In this unit we will start learning about web technologies that are used to build webpages and programs that use the web.

Unit outline

- Why are we learning this?
- The Internet, aka "the web"
- Browsers, servers
- Webpages, HTML, and CSS
- Backend, frontend
- Networking basics
- Python and Flask

Why are we learning this?

Much of our world is powered by the web.

Why are we learning this?

Even when we're not browsing *the web* on our *browsers*, we're likely on the web.

Why are we learning this?

Everything is connected to the web: your phone, your watch, even your fridge might even be connected to the web.

Why are we learning this?

But the primary use of the web is still the usage of webpages, and this is what we'll be learning about.

Why are we learning this?

Being able to create programs that rely on *the web* or *networking* is an important part of being a software engineer.

The Internet

When we talk about "the web" or "the net", we're referring to The Internet.

What is The Internet?

The Internet is a global network of billions of computers and electronic devices that are able to talk to each other.

Talking to each other

What is meant by "talking to each other" is simply the act of sending and receiving messages.

Computers talking to each other

Computers and other electronic devices are able to talk to each other in different ways.

Networks

When two or more computers talk to each other, they form a network.

Networking

When two or more computers form a network they are said to be *networking*.

Networking

We'll talk more about the basics networking in the near future, but know that networking is its own area of study in software engineering.

Let's talk about how we use the web now

1. Let's say you want to learn more about Ringo Starr, so you look him up on Google.
2. On your phone's **browser**, you navigate to Google's **webpage** by going to www.google.com. This is known as a **URL**.
3. The webpage's URL allows your phone to find Google's computers which can show you the results. Computers that your phone talks to are referred to as **servers**.

Browser

A browser is a computer program that is able to talk with other computers (or *servers*), ask for the contents of a *webpage*, and display them to you.

Server

Servers are simply computers. They're a lot like the computer you're using right now but they're usually optimized to heavy usage.

URL

URL stands for Uniform Resource Locator.

URL

URL are addresses for a resource (a webpage is a resource) on the web.

Webpage

A webpage, or a website, is all of the content that you see when using a browser.

Webpage

Webpages are made with code.

Webpage code

HTML, CSS, and JavaScript.

HTML and CSS

HTML and CSS let you define the structure and style for your webpage.

JavaScript

JavaScript is a programming language (just like Python is a programming language) that is able to interact with the webpage.

JavaScript

Webpages are static, meaning that they won't change once they are rendered, but JavaScript is able to change the content of the webpage.

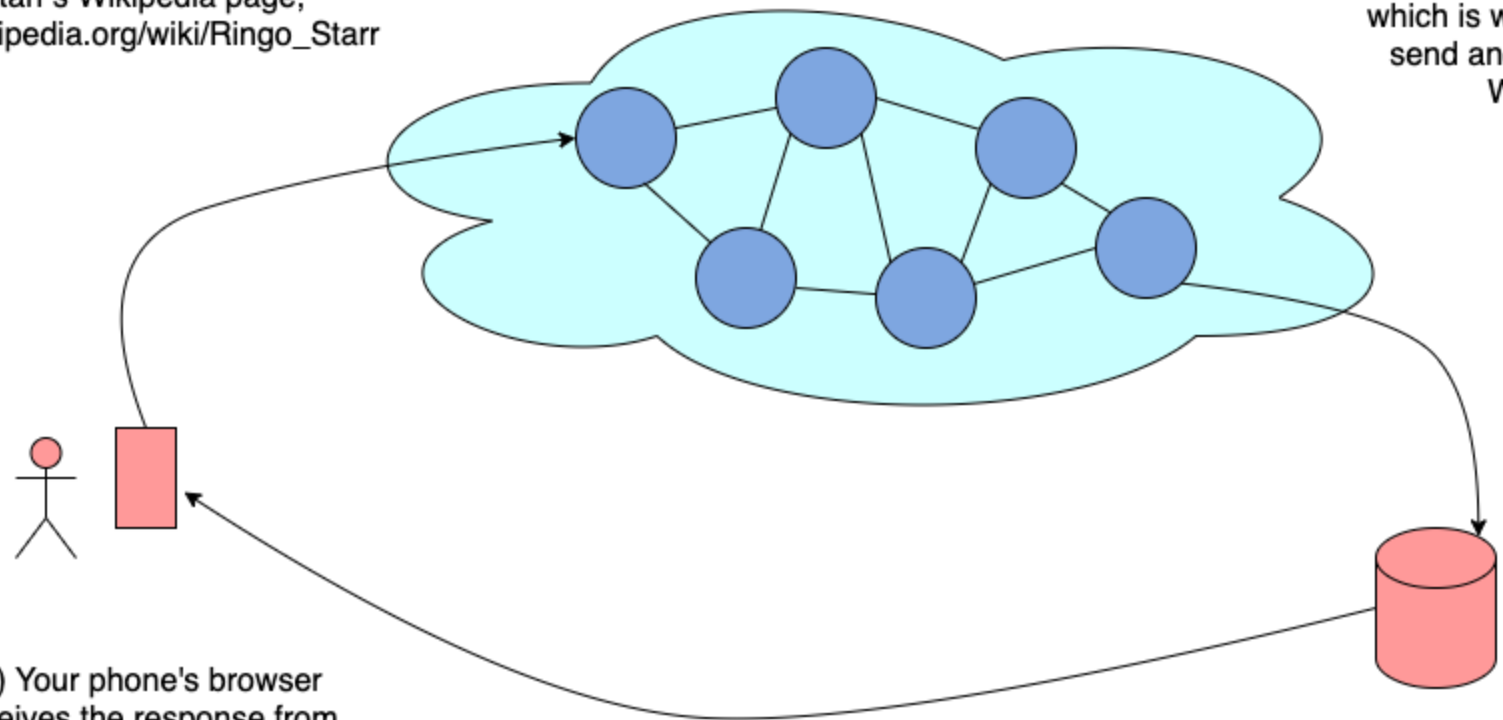
JavaScript

JavaScript allows users to interact with your webpage.

(1) You're on your phone and you go to Ringo Starr's Wikipedia page, https://en.wikipedia.org/wiki/Ringo_Starr

(2) Your phone is connected to the Internet, so its able to send a request saying that it needs the content for https://en.wikipedia.org/wiki/Ringo_Starr

(3) When your phone sends the request, it goes out to the Internet which is what enables your phone to send and receive messages from Wikipedia's servers



(4) Wikipedia's servers receive the message from your phone and respond with the HTML/CSS/JavaScript used to render Ringo Starr's Wikipedia page

(5) Your phone's browser receives the response from Wikipedia's servers and is able to render the HTML/CSS/JavaScript and show it to you

What are we going to continue learning about?

We're going to learn about HTML, CSS, and servers so that we can make our own webpages and programs that run on the web.

HTML: HyperText Markup Language

What is HTML for?

HTML is used to build webpages.

What is HTML for?

HTML is a markup language.

What is HTML for?

HTML lets us create the structure and the content of our webpages.

HTML tags

HTML is made up of *HTML tags*.

HTML example

```
<!DOCTYPE html>
<html>
    <head>
        <title>Programming Class</title>
    </head>
    <body>
        <h1>Welcome to class</h1>
    </body>
</html>
```

Different versions of HTML

HTML was released in the early 90's and there have been a few changes between then and now.

Different versions of HTML

Because older versions of HTML are still supported, we have to tell the browser which version we're using.

<!DOCTYPE html>

We tell the browser that we're using the latest version by putting `<!DOCTYPE html>` at the beginning of the HTML file.

<!DOCTYPE html>

```
<!DOCTYPE html>
<html>
    <body> ... </body>
</html>
```

HTML tags

Different HTML tags do different things.

HTML tags

For example, the `body` tags contains the webpage's body/content, the `table` tag renders a table, the `img` tag renders an image.

Some HTML tags

- `html` contains all of your webpage's HTML.
- `head` contains information that your browsers uses to render the page. Users don't see this.
- `meta` tells the browser how it should read your HTML file.
- `title` sets the page's title in the browser tag.
- `body` contains all of the content that the user sees.
- `h1` the largest text heading, there's also `h2`, `h3`, `h4`, `h5`, and `h6` which are each smaller than the previous one.

HTML tags

Tags have a name (for example, `table`), and are surrounded by angle brackets: `<table>`

HTML tags

There are start tags, `<table>`, and end tags `</table>`.

Start and end tags

Start and end tags look the same except for the slash: `<table>`, `</table>`.

Content

HTML tags can contain content.

Content

The content goes between the start and end tag.

Tags can contain text

```
<h1>Welcome to class</h1>
```

Tags can contain other tags

```
<body>
    <h1>Welcome to class</h1>
</body>
```

Self-closing tags

Some HTML tags don't have an end tag and they are said to be "self-closing": `<img />`

Self-closing tags

Self-closing tags don't have content, but they do have *attributes*.

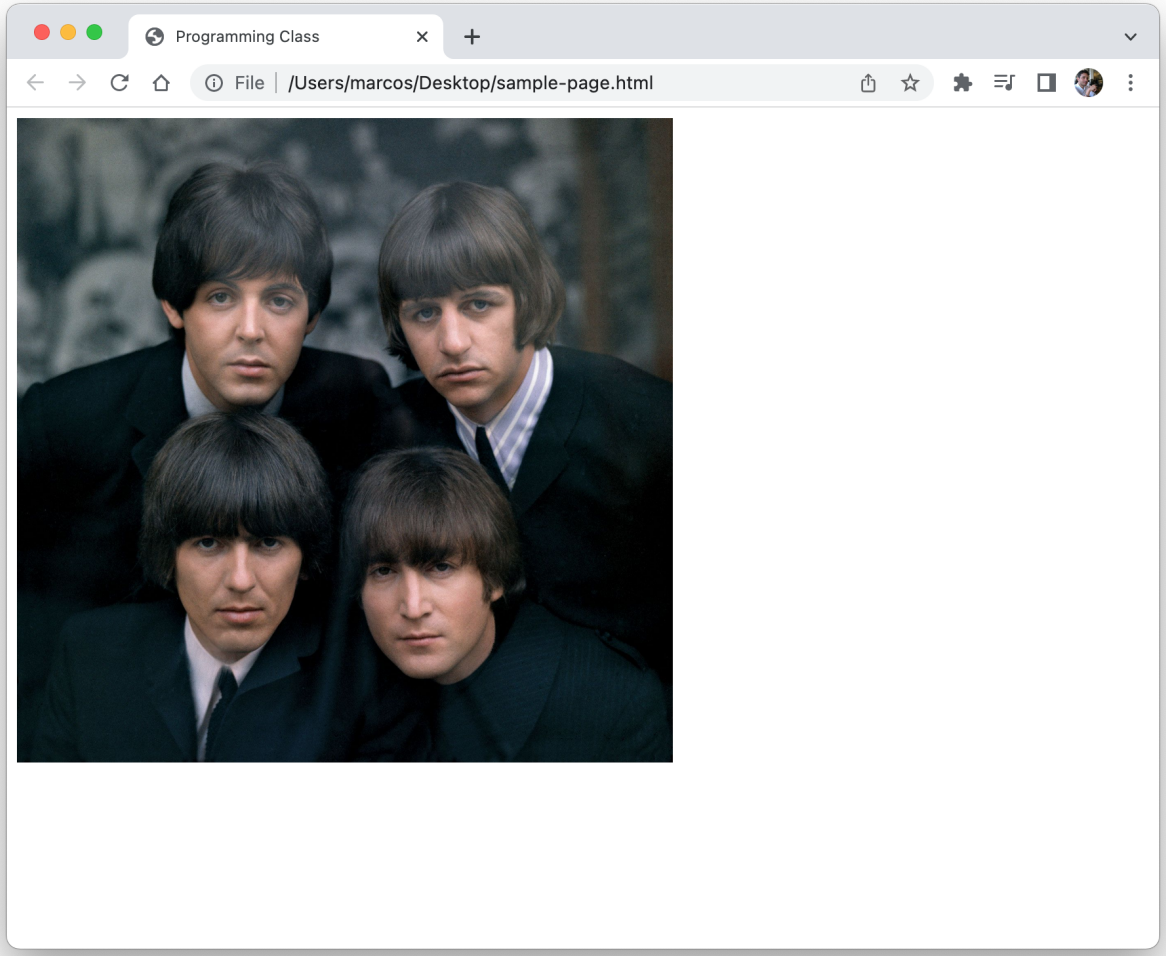
Attributes

Attributes are settings used by the browser when rendering a tag.

Example attribute

```
<!DOCTYPE html>
<html>
    <head>
        <title>Programming Class</title>
    </head>
    <body>
        <img src="TheBeatles.jpg" width="500px" />
    </body>
</html>
```

```
<img src="TheBeatles.jpg" width="500px" />
```



Attributes are settings used by the browser when rendering a tag.

For example, in `<img src="TheBeatles.jpg" width="500px" />`, `src` tells the browser where the image file can be found and `width` tells the browser how wide to render image.

A tag's attributes

All tags can have attributes.

A tag's attributes

But not all attributes work for all tags.

A tag's attributes

`src` works in tags like `img`, but won't work in an `h1` tag.

Where do tag attributes go?

Attributes go in the start tag. They *never* go in the end tag.

Common HTML tags (1)

- `html` contains all of your webpage's HTML.
- `head` contains information that your browsers uses to render the page. Users don't see this.
- `meta` metadata used by the browser when processing your code.
- `title` sets the page's title in the browser tag.
- `style` contains CSS code.
- `link` link to an asset file, such as a CSS file.
- `script` link to JavaScript file.

Common HTML tags (2)

- `body` contains all of the content that the user sees.
- `table` a table that contains different parts of a table, such as rows.
- `tr` a table row that contains table cells.
- `td` a table cell that contains any content.
- `img` used to show an image on the page.
- `span` an inline element used as a container for content.
- `div` a block element used as a container for content.
- `center` an element that centers is content.

Common HTML tags (3)

- `h1` the largest text heading, there's also `h2`, `h3`, `h4`, `h5`, and `h6` which are each smaller than the previous one.
- `p` used to render a paragraph of text.
- `ol` ordered list of items. Contains `li` tags.
- `ul` un-ordered list of items. Contains `li` tags.
- `li` list item, goes inside an `ol` or `ul`.
- `a` link to another page or another part of the current page.
- `input` renders an input element, such as a text field.

But there are is so much more!

MDN Web Docs is the best resource for anything related to web development

MDN Web Docs is the best resource for anything related to web development

Links can be found at the end of the slides.

CSS: Cascading Style Sheets

What is CSS for?

CSS is used to style webpages.

What is CSS for?

This means that CSS is able to change the look and feel of a webpage.

What is CSS for?

For example, it can change the size of an element, the color of a header, the font in your page, and many other visual aspects.

Sample CSS (whole page)

```
<!DOCTYPE html>
<html>
    <head>
        <title>Programming Class</title>
        <style>

        img {
            width: 500px;
            margin: 10px 30px;
            border: 1px solid red;
        }

        h1 {
            color: green;
            font-family: "Times New Roman";
        }

        </style>
    </head>
    <body>
        <h1>Welcome to The Beatles fan club page.</h1>
        <img src="TheBeatles.jpg" />
    </body>
</html>
```

Sample CSS (just the CSS)

```css
img {
    width: 500px;
    margin: 10px 30px;
    border: 1px solid red;
}

h1 {
    color: green;
    font-family: "Times New Roman";
}
```

Where does my CSS code go?

CSS can go in your `.html` file, but it must be inside a `style` tag.

Where does my CSS code go?

CSS can also be included in other ways but we'll come back to this.

CSS code

Let's talk about how CSS is written.

CSS rules

You write CSS by writing *CSS rules* where you can specially what properties you what to change for a set of elements.

Here's what that means

```
img {
    margin: 10px;
    width: 500px;
}

h1 {
    color: green;
    font-family: "Times New Roman";
    font-size: 40px;
}
```

CSS properties

CSS properties style an aspect of an HTML element.

Sample CSS properties

```
h1 {
    color: green;
    font-family: "Times New Roman";
    font-size: 40px;
}
```

Syntax

CSS properties are written in the following way: `<property name>: <value>;`

CSS selectors

CSS selectors determines which HTML elements the style changes are intended for.

Sample CSS selector

```
h1 {
    color: green;
    font-family: "Times New Roman";
    font-size: 40px;
}
```

CSS rules

CSS rules are the groupings of CSS selectors and the CSS properties.

Sample CSS rule

```
h1 {
    color: green;
    font-family: "Times New Roman";
    font-size: 40px;
}
```

CSS selectors

You can have one or more CSS selectors in a CSS rule.

Example of multiple selectors in a rule

```
h1, h2, h3, p {
    color: orange;
}
```

CSS selectors

- Tag selectors
- Class selectors
- ID selectors

CSS selectors

There are more CSS selectors but those are the most common ones and what we'll learn about in this class.

Tag selectors

CSS selectors can target elements by their tag name.

Tag selectors

```
h1, h2, h3, p {
    color: orange;
}
```

Class selectors

CSS selectors can target elements by their class.

Class selectors

HTML elements can have a `class` attribute:

```
<img class="large-image" src="TheBeatles.jpg" />
```

Class selectors

The class can then be used to target those HTML elements in your CSS:

```
.large-image {
    width: 1000px;
}

.medium-image {
    width: 500px;
}

.small-image {
    width: 250px;
}
```

Class selectors, syntax

Notice the dot (`.`) in `.large-image` is only used in CSS:

```
.large-image {
    width: 1000px;
}
```

Don't include it in your HTML:

```
<img class="large-image" src="TheBeatles.jpg" />
```

Matching HTML elements

CSS selector will target *all* elements that match the criteria.

Matching HTML elements

This means that a rule like the one below will change how all images in your page look:

```
img {
    width: 200px;
}
```

Matching HTML elements

Class selectors can let you target specific elements.

ID selectors

HTML elements can have an `id` attribute:

```
<img id="main-image" src="TheBeatles.jpg" />
```

ID selectors

The ID can then be used to target those HTML elements in your CSS:

```
#main-image {
    width: 1000px;
}
```

Matching HTML elements

IDs are unique.

Matching HTML elements

An ID corresponds to a single element in the page.

Matching HTML elements

This means that a rule like the one below will change how a single image in your page looks:

```
#main-image {
    width: 200px;
}
```

ID selectors, syntax

Notice the pound sign (`#`) in `#main-image` is only used in CSS:

```
#main-image {
    width: 200px;
}
```

Don't include it in your HTML:

```
<img id="main-image" src="TheBeatles.jpg" />
```

CSS selectors

- Tag selectors, example: style all `h1` elements.
- Class selectors, example: style all `p` elements with a class of `quote`.
- ID selectors, example: style single `img` element with an ID of `main-image`.

CSS selectors, syntax

- Tag selectors, just use the tag name: `p`, `img`, `span`, `table`.
- Class selectors, put a dot before the class name: `.red`, `.large`, `.active`.
- ID selectors, put a pound sign before the ID: `#intro`, `#footer`.

CSS inheritance

Styles trickle down from parent elements to child elements.

Where does my CSS code go?

CSS can go in your `.html` file, but it must be inside a `style` tag.

Where does my CSS code go?

CSS can also be placed into a separate `.css` file which can then be imported by your `.html` file with a `link` tag.

Import CSS files

```
<!DOCTYPE html>
<html>
    <head>
        <title>Programming Class</title>
        <link rel="stylesheet" href="text-styles.css" />
    </head>
    <body>
        <h1>Welcome to The Beatles fan club page.</h1>
        <img src="TheBeatles.jpg" />
    </body>
</html>
```

Where does my CSS code go?

Finally, CSS can also be inlined in an element's `style` attribute.

Common CSS properties

- `color` is used to change the color of text.
- `background-color` is used to change the background color of an element.
- `font-size` is used to change the font size.
- `font-family` is used to change the font used.
- `height` changes the height.
- `width` changes the width
- `border` is used to apply a border line around an element.
- `margin` is used to add spacing around (outside) the element.
- `padding` is used to add spacing inside the element.

But there are is so much more!

MDN Web Docs is the best resource for anything related to web development

Links can be found at the end of the slides.

Web Development

Static vs. dynamic content

Static content is content that does not change.

Static vs. dynamic content: static example

For example, if you create an HTML file and serve it on your website, any time you need to display new content you would have to modify your HTML file.

Static vs. dynamic content

Dynamic content is content that is generated by a program, allowing it to change depending on logic written by a programmer.

Static vs. dynamic content: dynamic example

For example, a web server that generates an HTML page with the current date and time.

MDN Web Docs

- https://developer.mozilla.org
- https://developer.mozilla.org/en-US/docs/Web/HTML, HTML tutorial
- https://developer.mozilla.org/en-US/docs/Web/CSS, CSS tutorial