FWDP 1000 – Day 2

Course:	Web Development 1
Instructor:	Gabbie Bade

Morning Review

- Download the files.
- Time permitting, open the three HTML files in the review-of-day-01 folder in your code editor.
- Read the HTML comments in the files and complete the tasks.
- If you are unable to complete this, **THAT IS OKAY**. The completed code has also been provided for your reference.

Best Practices Follow-up

- H1s only ever occur once per webpage/HTML file. It can be in the header or main, wherever it makes sense for your content.
- Refer to MDN Web Docs for terminologies and basic syntax. The more you learn coding, the more documentations you would have to read through.
- Keep your files organised. All lowercase and no spaces (use - or _).

Best Practices Follow-up

- MDN has indicated that noopener is implied when using target=“_blank”, so all you need to add is rel=“noreferrer”.

```
<a href=“index.html" target=“_blank”
rel=“noreferrer”>
```

- noreferrer = prevents the new site from seeing where you were from/original tab (helps with privacy)
- noopener = prevents new site from messing with where you were/original tab (improves security)

Agenda

- More HTML Elements
- HTML Class Attribute
- Introduction to CSS
- Git Practice: Setting up a Repo + Making branches
- Assignment #2

Images on the Web

Images on the Web

Traditionally, images on the web were in these formats:

JPG (or JPEG) for
Photographs

PNG or GIF
for Graphics

Images on the Web

In addition to JPG and PNG files for images, there are three others you are likely to use on the web as well:

- WebP
- AVIF
- SVG

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types

WebP

The **.webp** image format is a modern replacement for JPG and PNG files on the web. It can provide the same quality at a smaller file size.

It has wide support across browsers. The biggest issue is that it does not work in Safari for macOS 10.15 or earlier: https://caniuse.com/webp

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types#webp_image

Creating WebP Files

Adobe Photoshop 23.2 and higher has native support for saving WebP files.

Use File → Save As or File → Save a Copy and choose “WebP” from the dropdown.

https://helpx.adobe.com/ca/photoshop/kb/support-webp-image-format.html

AVIF

The **.avif** image format is a modern replacement for JPG and PNG files on the web as well. It provides even smaller file sizes than .webp images at the same quality.

Browser support is more limited however:
https://caniuse.com/avif

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types#avif_image

Creating AVIF Files

You cannot create AVIF files by default in Photoshop but you can install a Photoshop plugin to create them.

Follow the steps under “Installation” here: https://github.com/0xC0000054/avif-format

Once installed, use “Save As” to get the option to save as a .avif file.

WebP vs AVIF

The main reason to not use AVIF is simply the lack of support in certain browsers currently.

If you are curious to learn more about these image formats, check out this article:

https://www.smashingmagazine.com/2021/09/modern-image-formats-avif-webp/

Scalable Vector Graphics

Scalable Vector Graphics (SVGs) are generally what we use for creating basic graphics and icons on the web.

They are a great replacement for PNG and GIF files.

You will learn more about these and create them during your design courses using Adobe Illustrator.

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Image_types#svg_scalable_vector_graphics

Optimizing Images for the Web

An image from a phone/camera is meant to be printed and must be optimized before using them on the web.

This means… determining the optimal dimensions, the best file format, and saving it at the smallest file size possible without giving up quality.

Smaller file sizes = faster websites

Image Dimensions

Images should generally not be displayed larger than their dimensions (width and height in pixels).

For example: using CSS to display an image larger than the dimensions it was saved at will make it blurry and pixelated.

It is fine to shrink an image with CSS though.

Image File Sizes

Every image file should be **below 1MB** if it is being used on the web and usually much lower than 1MB.

Even large, full width photographs can be under 500KB without degrading the image quality when viewed on a screen.

In your design course you will learn more about optimizing images for the web.

Animated GIFs

Instead of using animated GIFs like the one below, you should use video files since the quality is better and file size is lower.

Videos on the Web

Video file sizes should be as small as possible when used on a website as a replacement for an image.

Consider dimensions, file size, and file format. We will discuss formats shortly.

For large video files, use YouTube or Vimeo.

https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Containers

Performance and Optimization

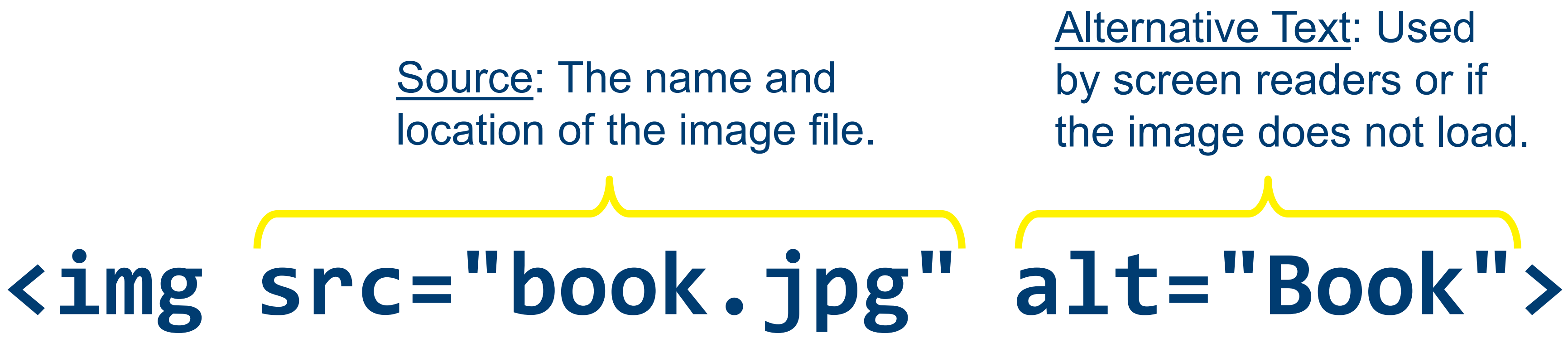
We will discuss optimizing your websites for performance in greater detail during the Capstone project at the end of the program.

But again, the basic concept is…

…the smaller your file sizes, the faster your website.

Image Element

This is the basic structure of the HTML image element.



Source: The name and location of the image file.

Alternative Text: Used by screen readers or if the image does not load.

```
<img src="book.jpg" alt="Book">
```

Image Element

Make sure you are in the day-2 branch.

Let’s add two basic image examples to our homepage:

One JPG and one PNG.

Add and commit!

https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Images_in_HTML

File Paths

Relative vs Absolute Paths

A relative path points to a file relative to the current file.

```
<img src="images/book.jpg" alt="Book">
```

An absolute path is the full URL to a file beginning with **http**.

```
<img src="https://example.com/images/book.jpg" alt="Book">
```

When to use each

Use an absolute path to load an external resource.

Use a relative path to load a resource in your files.

Note: This rule will change slightly when working with content management systems later in the program.

File Path Syntax

These two lines mean the exact same thing:

```
<img src="images/book.jpg" alt="Book">

<img src="./images/book.jpg" alt="Book">
```

The second one is using an older syntax that is not necessary for HTML, CSS, or JavaScript.

File Path Syntax

Beginning a path with a slash means “go to the root folder on the server”. These do **NOT** point to the same place:

```
<img src="images/book.jpg" alt="Book">

<img src="/images/book.jpg" alt="Book">
```

Avoid this. You will almost never need to do this.

“Live Server” in VS Code will act like these are the same but it is wrong and your code will likely break on a real server.

File Path Syntax

This path says “go up **one** folder to find the images folder, inside of the images folder load book.jpg”:

```
<img src="../images/book.jpg" alt="Book">
```

This path says “go up **two** folders to find the images folder, inside of the images folder load book.jpg”:

```
<img src="../../images/book.jpg" alt="Book">
```

More on File Paths

If that isn't completely clear yet, have a look at the links below for further explanation.

https://www.w3schools.com/html/html_filepaths.asp

https://css-tricks.com/quick-reminder-about-file-paths/

Recap

Which image formats have support in every browser?

JPG AVIF WEBP PNG GIF

True or False: The alt attribute on the <img> element is optional.

Audio & Video Elements

Audio Element

The `<audio>` element allows you to embed audio files on a webpage. You can set the audio to have controls, loop or autoplay.

Multiple file formats can be provided to allow the browser to select the file format it prefers.

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

Video Element

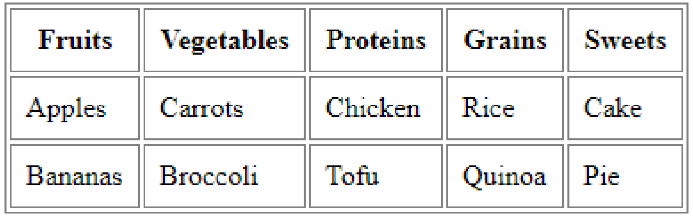
The `<video>` element allows you to embed video files on a webpage. You can set the video to have controls, loop, autoplay, be muted, and have an image display until played.

Multiple file formats can be provided to allow the browser to select the file format it prefers.

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

Table Element

The table element will create tables, similar to what you would see in a spreadsheet. Within it are HTML elements for rows, heading cells and data cells both of which are nested inside rows.



Fruits	Vegetables	Proteins	Grains	Sweets
Apples	Carrots	Chicken	Rice	Cake
Bananas	Broccoli	Tofu	Quinoa	Pie

https://developer.mozilla.org/en-US/docs/Learn/HTML/Tables/Basics

More Semantic Elements

Article Element

The `<article>` tag is used to wrap content that can be viewed independent of the rest of the webpage.

Examples: a blog post, a news article, a product, etc.

Let’s edit the news.html file to add two articles.

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

Article Child Elements

It’s common to nest other semantic elements within an article. For instance: `<header>`, `<section>`, `<footer>`.

In some cases, like a blog post or news article, you may also want to use elements like `<time>` to provide additional information.

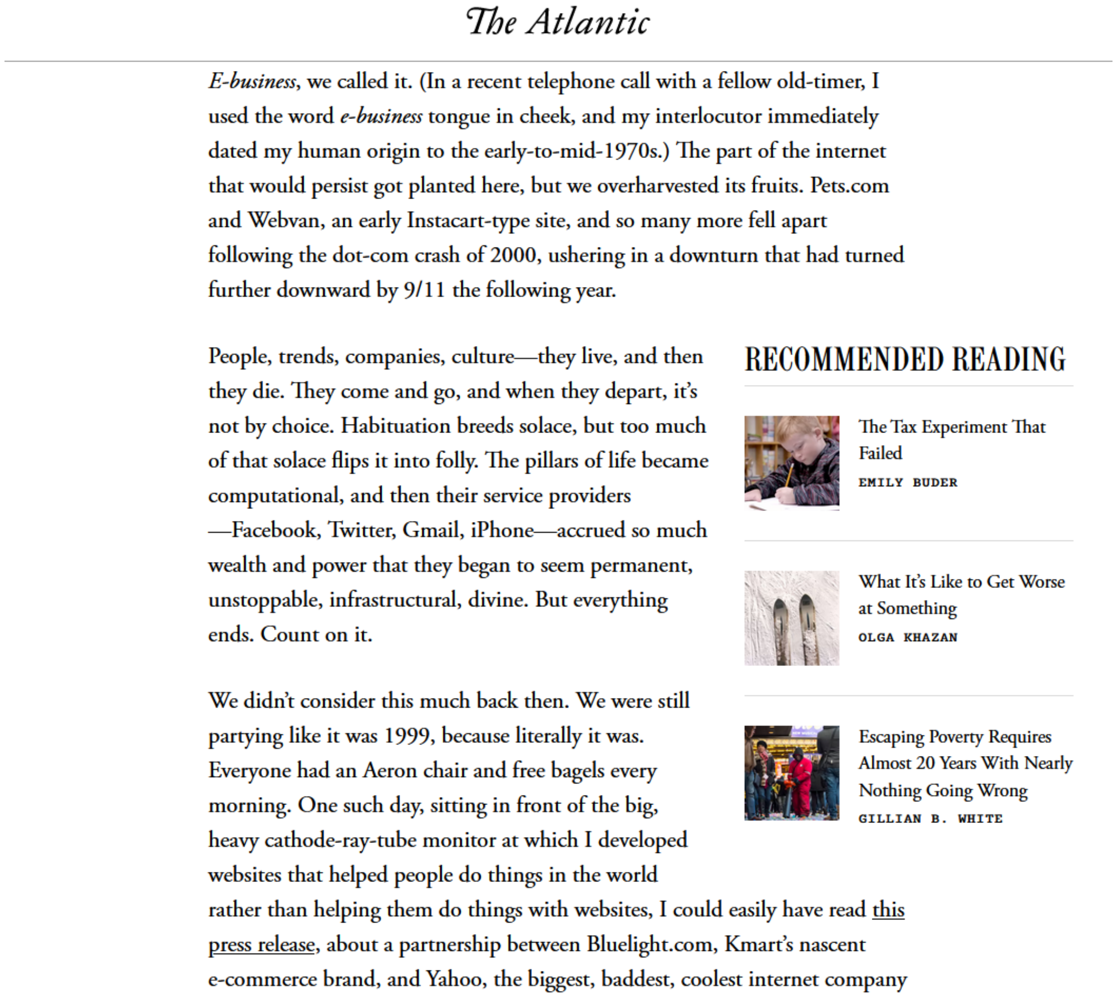
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time

Aside Element

The `<aside>` tag is used to wrap content that is indirectly related to the main content.

Examples: a sidebar, related articles, a related fact or piece of trivia in an article.

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



The Atlantic

E-business, we called it. (In a recent telephone call with a fellow old-timer, I used the word *e-business* tongue in cheek, and my interlocutor immediately dated my human origin to the early-to-mid-1970s.) The part of the internet that would persist got planted here, but we overharvested its fruits. Pets.com and Webvan, an early Instacart-type site, and so many more fell apart following the dot-com crash of 2000, ushering in a downturn that had turned further downward by 9/11 the following year.

People, trends, companies, culture—they live, and then they die. They come and go, and when they depart, it's not by choice. Habituation breeds solace, but too much of that solace flips it into folly. The pillars of life became computational, and then their service providers —Facebook, Twitter, Gmail, iPhone—accrued so much wealth and power that they began to seem permanent, unstoppable, infrastructural, divine. But everything ends. Count on it.

We didn't consider this much back then. We were still partying like it was 1999, because literally it was. Everyone had an Aeron chair and free bagels every morning. One such day, sitting in front of the big, heavy cathode-ray-tube monitor at which I developed websites that helped people do things in the world rather than helping them do things with websites, I could easily have read this press release, about a partnership between Bluelight.com, Kmart's nascent e-commerce brand, and Yahoo, the biggest, baddest, coolest internet company

RECOMMENDED READING

The Tax Experiment That Failed
EMILY BUDER

What It's Like to Get Worse at Something
OLGA KHAZAN

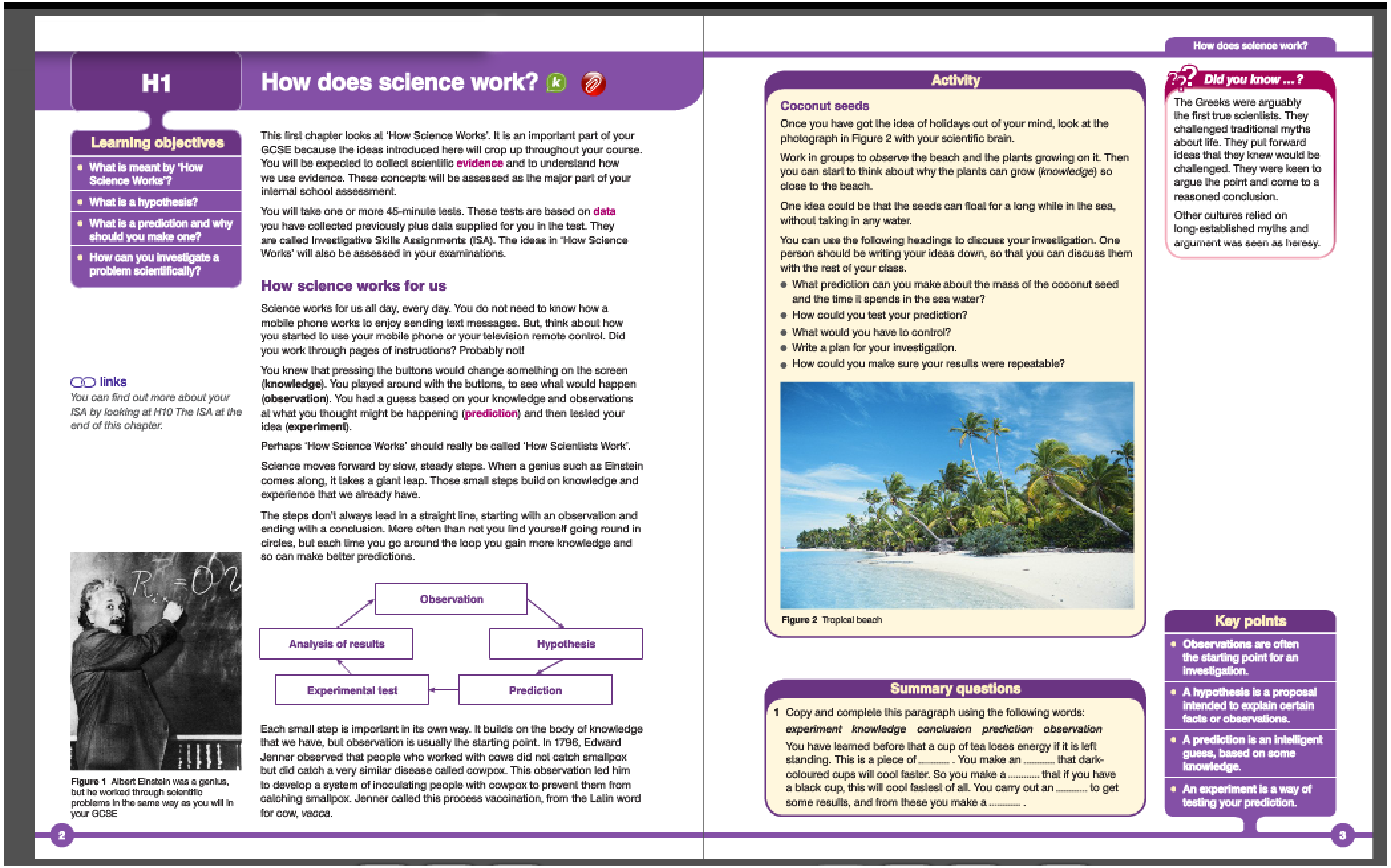
Escaping Poverty Requires Almost 20 Years With Nearly Nothing Going Wrong
GILLIAN B. WHITE

Figure Element

The `<figure>` tag is used to wrap content that is related to the article or webpage but could be moved or removed without affecting the understanding of the article or webpage.

Examples: images, diagrams, code snippets, etc.

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



H1 How does science work?

Learning objectives

- What is meant by 'How Science Works'?
- What is a hypothesis?
- What is a prediction and why should you make one?
- How can you investigate a problem scientifically?

This first chapter looks at 'How Science Works'. It is an important part of your GCSE because the ideas introduced here will crop up throughout your course. You will be expected to collect scientific **evidence** and to understand how we use evidence. These concepts will be assessed as the major part of your internal school assessment.

You will take one or more 45-minute tests. These tests are based on **data** you have collected previously plus data supplied for you in the test. They are called Investigative Skills Assignments (ISA). The ideas in 'How Science Works' will also be assessed in your examinations.

How science works for us

Science works for us all day, every day. You do not need to know how a mobile phone works to enjoy sending text messages. But, think about how you started to use your mobile phone or your television remote control. Did you work through pages of instructions? Probably not!

You knew that pressing the buttons would change something on the screen (**knowledge**). You played around with the buttons, to see what would happen (**observation**). You had a guess based on your knowledge and observations at what you thought might be happening (**prediction**) and then tested your idea (**experiment**).

Perhaps 'How Science Works' should really be called 'How Scientists Work'.

Science moves forward by slow, steady steps. When a genius such as Einstein comes along, it takes a giant leap. Those small steps build on knowledge and experience that we already have.

The steps don't always lead in a straight line, starting with an observation and ending with a conclusion. More often than not you find yourself going round in circles, but each time you go around the loop you gain more knowledge and so can make better predictions.

Observation → Hypothesis → Prediction → Experimental test → Analysis of results → Observation

Each small step is important in its own way. It builds on the body of knowledge that we have, but observation is usually the starting point. In 1796, Edward Jenner observed that people who worked with cows did not catch smallpox but did catch a very similar disease called cowpox. This observation led him to develop a system of inoculating people with cowpox to prevent them from catching smallpox. Jenner called this process vaccination, from the Latin word for cow, vacca.

links

You can find out more about your ISA by looking at H10 The ISA at the end of this chapter.

Figure 1 Albert Einstein was a genius, but he worked through scientific problems in the same way as you will in your GCSE

Activity

Coconut seeds

Once you have got the idea of holidays out of your mind, look at the photograph in Figure 2 with your scientific brain.

Work in groups to *observe* the beach and the plants growing on it. Then you can start to think about why the plants can grow (*knowledge*) so close to the beach.

One idea could be that the seeds can float for a long while in the sea, without taking in any water.

You can use the following headings to discuss your investigation. One person should be writing your ideas down, so that you can discuss them with the rest of your class.

- What prediction can you make about the mass of the coconut seed and the time it spends in the sea water?
- How could you test your prediction?
- What would you have to control?
- Write a plan for your investigation.
- How could you make sure your results were repeatable?

Figure 2 Tropical beach

Did you know ...?

The Greeks were arguably the first true scientists. They challenged traditional myths about life. They put forward ideas that they knew would be challenged. They were keen to argue the point and come to a reasoned conclusion.

Other cultures relied on long-established myths and argument was seen as heresy.

Summary questions

1 Copy and complete this paragraph using the following words:

experiment knowledge conclusion prediction observation

You have learned before that a cup of tea loses energy if it is left standing. This is a piece of .... You make an .... that dark-coloured cups will cool faster. So you make a .... that if you have a black cup, this will cool fastest of all. You carry out an .... to get some results, and from these you make a ....

Key points

- **Observations are often the starting point for an investigation.**
- **A hypothesis is a proposal intended to explain certain facts or observations.**
- **A prediction is an intelligent guess, based on some knowledge.**
- **An experiment is a way of testing your prediction.**

Figure Caption Element

The `<figcaption>` tag is used to provide an optional caption for the figure element.

This element can **only** be used if it is the first or last child inside a `<figure>` element.

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

Question #1

Which HTML element can be used to put a line through text to indicate the text is no longer accurate?

Example:

~~Doors will be open at 10am.~~

THIS EVENT IS CANCELLED!

Question #2

Which tag should be used to add a footnote number smaller and placed near the top of other text?

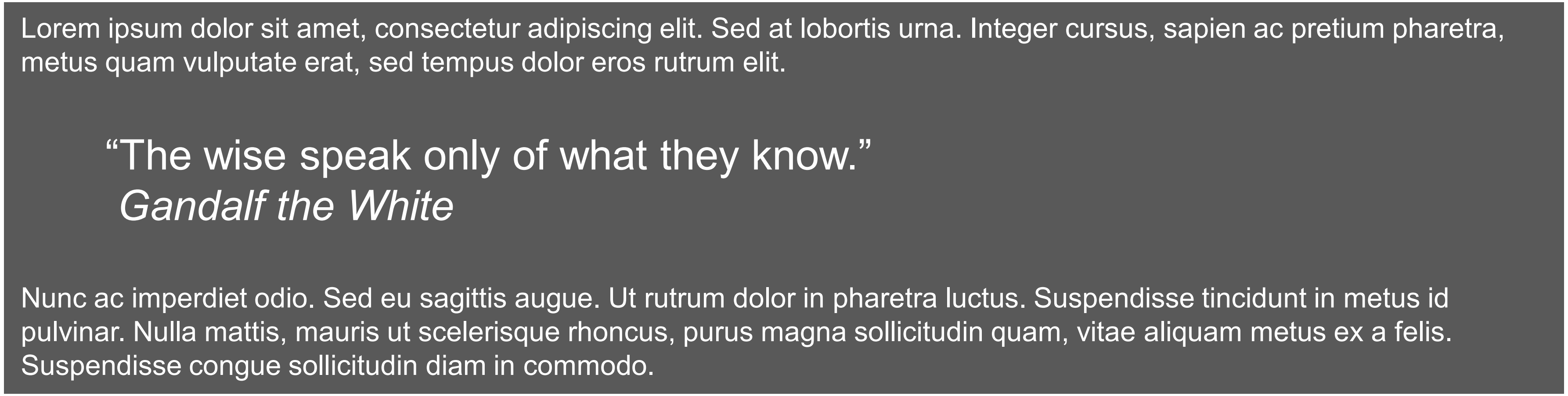
Example:

Mackenzie King was Canada's longest-serving prime minister, holding office for over twenty-one years.[2]

Question #3

What element would be used to wrap both the quotation and name? What element would be used to wrap only the name?

Example:



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed at lobortis urna. Integer cursus, sapien ac pretium pharetra, metus quam vulputate erat, sed tempus dolor eros rutrum elit.

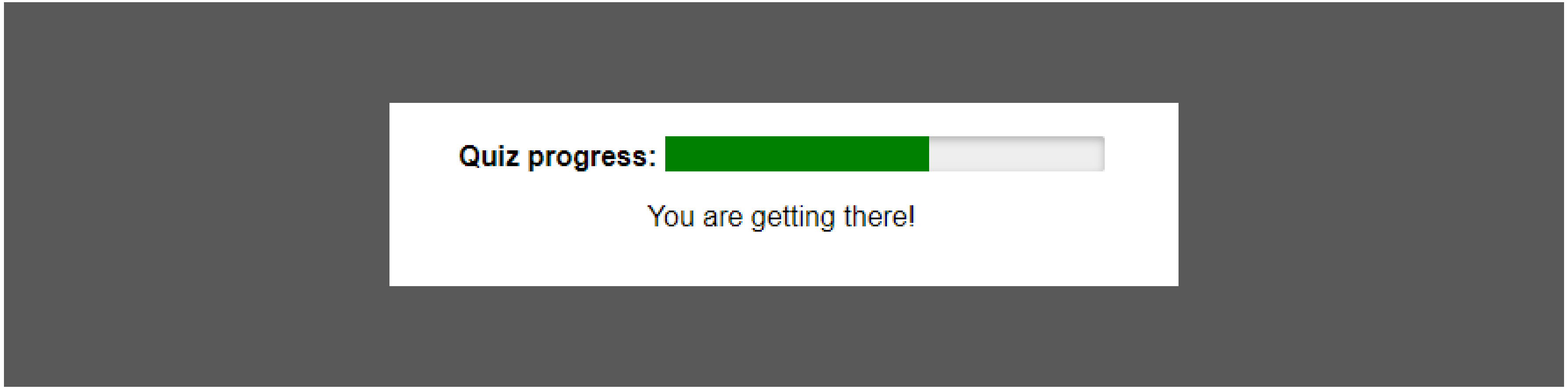
> “The wise speak only of what they know.”
> *Gandalf the White*

Nunc ac imperdiet odio. Sed eu sagittis augue. Ut rutrum dolor in pharetra luctus. Suspendisse tincidunt in metus id pulvinar. Nulla mattis, mauris ut scelerisque rhoncus, purus magna sollicitudin quam, vitae aliquam metus ex a felis. Suspendisse congue sollicitudin diam in commodo.

Question #4

What element displays an indicator showing the completion progress of a task, typically displayed as a progress bar?

Example:



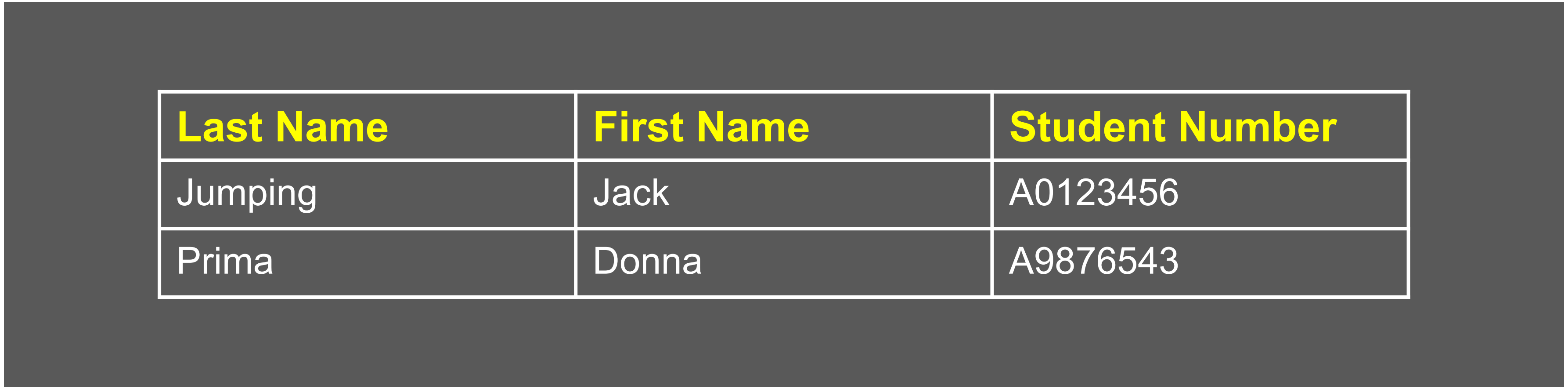
Quiz progress:

You are getting there!

Question #5

What element defines a cell as the header of a group of table cells?

Example:



Last Name	First Name	Student Number
Jumping	Jack	A0123456
Prima	Donna	A9876543

Question #6

What interactive element would be used to open and close a menu of links when the user clicks or touches the element?

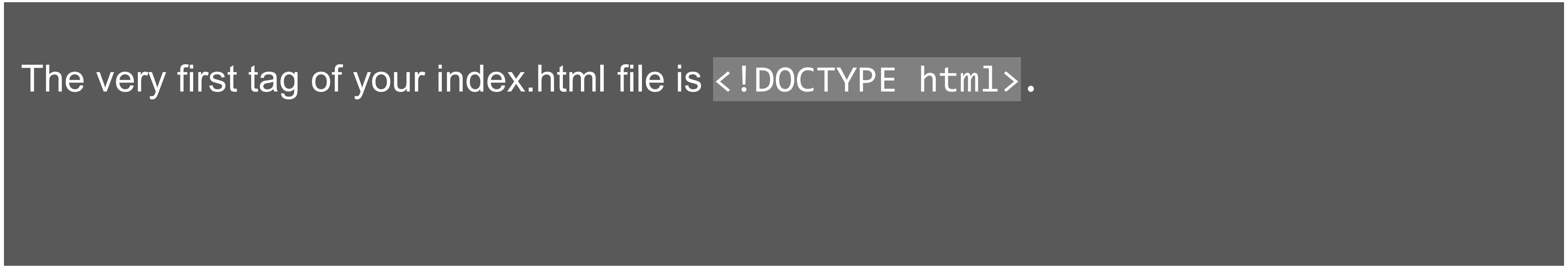
Example:

https://wp.bcitwebdeveloper.ca/assets/hamburger-menu-demo.gif

Question #7

What element displays its contents styled in a fashion intended to indicate that the text is a short fragment of computer code?

Example:



The very first tag of your index.html file is `<!DOCTYPE html>`.

Generic Elements

Div Element

The `<div>` tag is a generic tag used to create a **block-level** element for the purpose of CSS or JavaScript.

It has no semantic value and should only be used when no other semantic element is appropriate.

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

Span Element

The `<span>` tag is a generic tag used to create an **inline** element for the purpose of CSS or JavaScript.

It has no semantic value and should only be used when no other semantic element is appropriate.

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

Classes

Class Attribute

The class attribute can be set on **any** HTML tag.

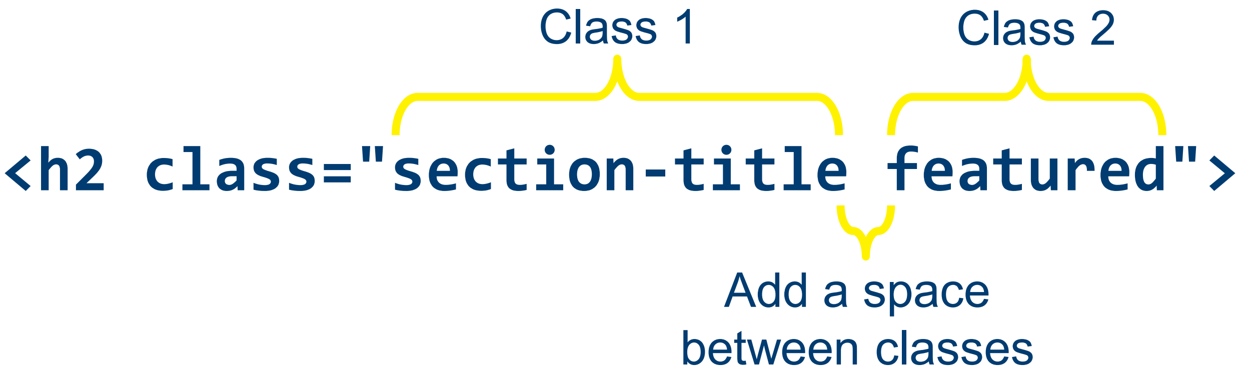
Unlike the ID attribute, which must have a unique value on a webpage, class values **can** be reused on multiple HTML elements within a webpage.

```
<h2 class="featured">
```

https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/class

Multiple Classes

In addition to being able to use the same class name on multiple elements, you can also set multiple classes on a single element…



Class 1

Class 2

```
<h2 class="section-title featured">
```

Add a space
between classes

Class Naming

There are many class naming conventions but these are some general guidelines to start with:

- Begin with a letter or number
- Use letters, numbers, dashes "-", and underscores "_"
- Use lowercase characters

Uses for Classes

Classes are primarily used for styling webpages with CSS.

Classes can also be used in JavaScript to add functionality to your webpage.

Introduction to CSS

Introduction to CSS

Cascading Style Sheets, or CSS, is how browsers control the design and layout of webpages.

CSS can be added to an HTML document in three ways:

- Saved as a **.css** file and linked from the **.html** file (most common)
- Embedded in the **.html** file (useful in some cases)
- Written inline on an HTML element (not recommended)

https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_is_structured

Inline CSS

With inline CSS, styles are written as an HTML attribute.

This is **not** recommended. HTML should contain just the content and markup, separate from the CSS.

Making changes is very tedious with this approach as well.

Inline CSS

```
<h2 style="color: red;">Section Title</h2>
```

Internal CSS

With internal CSS, styles are written inside of the `<style>` element which is in the `<head>` element of the HTML file.

There are cases where this can be useful, like in a single page website or when you have limited access in a CMS.

In general, you will only use this approach if you cannot use the external CSS approach.

Internal CSS Example

```
<head>
   <meta charset="UTF-8">
   <title>Page Title</title>
   <style>
      h2 {
         color: red;
      }
   </style>
</head>
```

External CSS

With external CSS, styles are written in a separate **.css** file and attached to the HTML file using the `<link>` element.

This is the most common way to add CSS to your webpages because one CSS file can apply to thousands of webpages on a large web site.

External CSS Example

```
<head>
   <meta charset="UTF-8">
   <title>Page Title</title>
   <link rel="stylesheet" href="styles/styles.css">
</head>
```

Within the styles.css file

```
h2 {
    color: red;
}
```

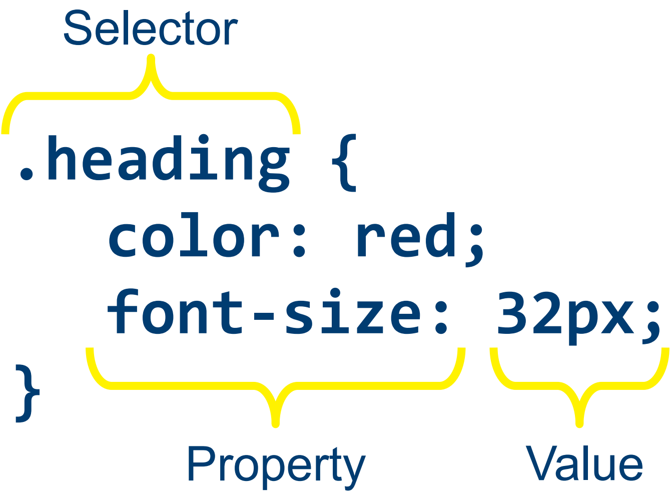
Creating a Stylesheet

Let’s create an external stylesheet for our website.

To begin with, we will add it to our home page (index.html).

CSS Syntax

CSS has a simple syntax of a **selector** followed by **curly brackets** that wrap the **property** and **value** pairs.



Selector

```
.heading {
   color: red;
   font-size: 32px;
}
```

Property

Value

CSS Syntax

Check out the blog post linked below for a detailed breakdown of the CSS syntax.

Very useful for a quick read through!

https://css-tricks.com/css-basics-syntax-matters-syntax-doesnt/

CSS Selectors

The selector is used to target an HTML element for styling.

Element selector: `h2 { }`

This will target all `<h2>` elements.

Class selector: `.col { }`

This will target all elements with the class “col”. Use a dot to tell CSS that it is a class, not an element.

ID selector: `#top { }`

This will target the element with the ID “top”. Use a hash to tell CSS that it is an ID, not an element.

CSS Descendant Selectors

Selectors can also be combined to target descendants.
Only the **last** selector is the element that will be styled.

```
section h2 { }
```

This will target all `<h2>` elements that are children, grandchildren, etc. of all `<section>` elements.

```
.featured a { }
```

This will target all `<a>` elements that are children, grandchildren, etc. of all elements with the class of “featured”.

Multiple CSS Selectors

If you want the same styles to apply to multiple selectors, you can use a comma to do so.

```
h2,
.featured {
   color: red;
}
```

This will target all `<h2>` elements **and** any element with the class “featured” and make the font red.

Combining CSS Selectors

If you want to target an HTML element **only** if it has a specific class, write your selector without a space.

```
h2.featured {
   color: red;
}
```

This will target all `<h2>` elements with the class “featured”.

Advanced CSS Selectors

We will cover more advanced CSS selectors as we go on.

The link below is a table showing all available CSS selectors.

https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors#Reference_table_of_selectors

CSS Inheritance

Inheritance: Some CSS property values set on parent elements can be inherited by their children.

For example, if we set `section { color: red; }` and have multiple `<p>` elements inside of the `<section>`, those paragraphs will display with red font.

CSS Cascade

Cascade: The order of your CSS matters. Later rules can override earlier rules. In the example below, the color of the heading will be blue.

```
h2 {
    color: red;
}

h2 {
   color: blue;
}
```

CSS Specificity

Specificity: If multiple CSS rules apply to the same element, some selectors are more powerful than others.

This is a simplified order of power:

ID Selector > Class Selector > Element Selector

This gets tricky when multiple types are used together so make your life easier and don’t nest too many selectors.

CSS Specificity Example

```
.featured {
    color: red;
}

h2 {
    color: blue;
}
```

Any <h2> element with the class “featured” will be red, even though it comes first.

Class > Element

All <h2> elements in a section will be red because two selectors is more powerful than one.

```
section h2 {
    color: red;
}

h2 {
    color: blue;
}
```

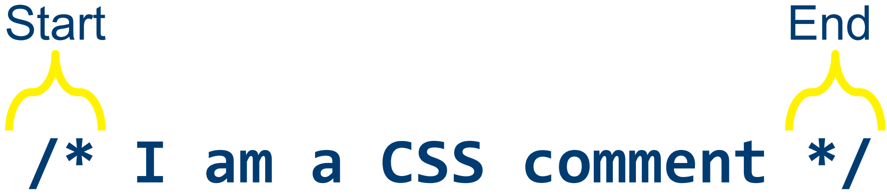
Invalid CSS

- If you have invalid CSS, the browser simply ignores it.
  - This is invalid: `..col { COLOR: RED; }`
- CSS properties and values are case sensitive.
  - This is invalid: `h2 { COLOR: RED; }`
- CSS properties and values use US spelling.
  - This is invalid: `h2 { colour: red; }`

CSS Comments

You can write comments in your CSS code that will not display on the webpage.

This is only visible when viewing the code and very helpful to explain to yourself and other developers what your code does.



Start

End

```
/* I am a CSS comment */
```

Code Comments

Writing good comments in your code is extremely helpful for yourself and other developers.

A great article with some general rules:

https://stackoverflow.blog/2021/07/05/best-practices-for-writing-code-comments/

Recap

In your CSS file all <p> elements are set to color: blue on line 6 but then they are set to color: green on line 324 so the later rule applies. Which CSS aspect does this describe?

In your CSS file all <article> elements are set to color: red and now all child elements in an <article> have red text. Which CSS aspect does this describe?

Inheritance Cascade Specificity

CSS Color Properties

Color can be applied to any HTML element. The two most common ways to set color are with these CSS properties:

CSS Property	Reference Link
`color`	Used for text.
`background-color`	Used for backgrounds.

Check out the other things that can have color set:
https://developer.mozilla.org/en-US/docs/Web/HTML/Applying_color

CSS Color Values

Colors in CSS can be defined in the following formats:

Format	Example Code	Example Output
Keyword	`red`	🟥
RGB or Hex	`rgb(255, 0, 0)` OR `#ff0000`	🟥
RGBA or Hex	`rgb(255, 0, 0, 50%)` OR `#ff000050`	🟥
HSL	`hsl(0, 225, 128)`	🟥
HSLA	`hsl(0, 225, 128, 50%)`	🟥

https://developer.mozilla.org/en-US/docs/Web/HTML/Applying_color#How_to_describe_a_color

CSS Color Values

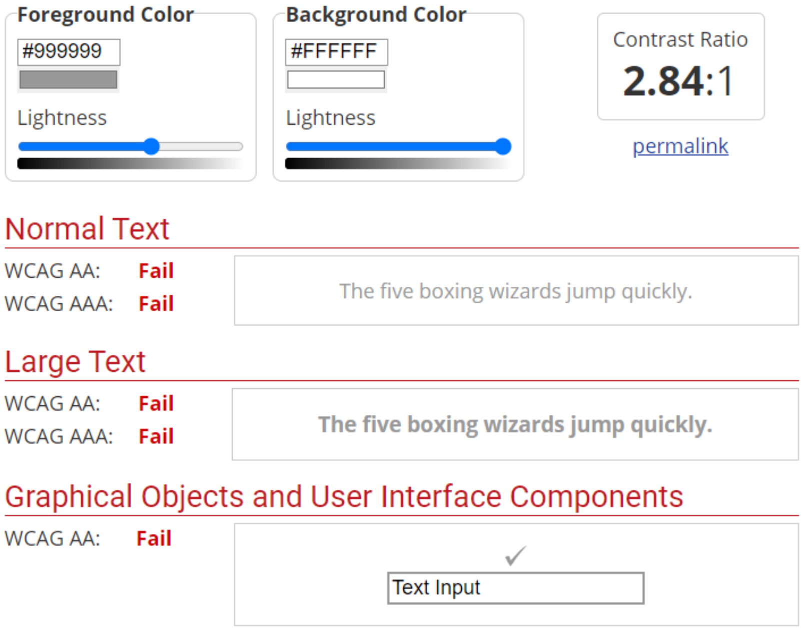
The color formats all do the same thing and browsers support them all equally so use the one that you prefer.

To reduce confusion, stick with one CSS color format on a website and don’t mix them.

Accessible Colors

For readability, have sufficient color contrast in your designs.

Example: small, gray text on a white background is bad.



Foreground Color
#999999
Lightness

Background Color
#FFFFFF
Lightness

Contrast Ratio
2.84:1
permalink

Normal Text

WCAG AA:	**Fail**	The five boxing wizards jump quickly.
WCAG AAA:	**Fail**	

Large Text

WCAG AA:	**Fail**	**The five boxing wizards jump quickly.**
WCAG AAA:	**Fail**	

Graphical Objects and User Interface Components

WCAG AA:	**Fail**	✓ Text Input

Accessible Color Tools

Contrast Checker

https://webaim.org/resources/contrastchecker/

Colorblindly (Chrome Extension)

https://chrome.google.com/webstore/detail/colorblindly/floniaahmccleoclneebhhmnjgdfijgg

Color Selecting Tools

Adobe Color Wheel

https://color.adobe.com/create/color-wheel

Color Palettes

https://coolors.co/palettes/trending

CSS Font Styling

Below are some of the common CSS properties for styling text on a webpage.

CSS Property	Reference Link
`font-family`	Change the font of the text.
`font-size`	Change the size of the text.
`font-style`	Make the text italic.
`font-weight`	Change the boldness of the text.
`line-height`	Change the height of the text.
`color`	Change the color of the text.

Web Fonts

We will cover web fonts in more detail later and specifically how to use custom fonts.

Until then, use some “web safe” fonts:

https://www.cssfontstack.com/

Font Sizes

These are the three primary units to set font sizes:

- `em` – Based on the font size of the parent element. It can be useful but is also tricky.
- `rem` – Based on the font size of the root `<html>` element. Nearly every browser defaults to 16px, so 1rem = 16px.
- `px` – A set pixel value. The old standard but it is not accessible and should be avoided.

Font Sizes

I would recommend using the `rem` unit when starting out.

Since most browsers default to a 16px font size…

```
1rem     = 16px
1.125rem = 18px
1.5rem   = 24px
2rem     = 32px
etc.
```

The math isn’t hard and your site will be accessible.

Let’s set up a Git repo!

Let’s set up a Git repo!

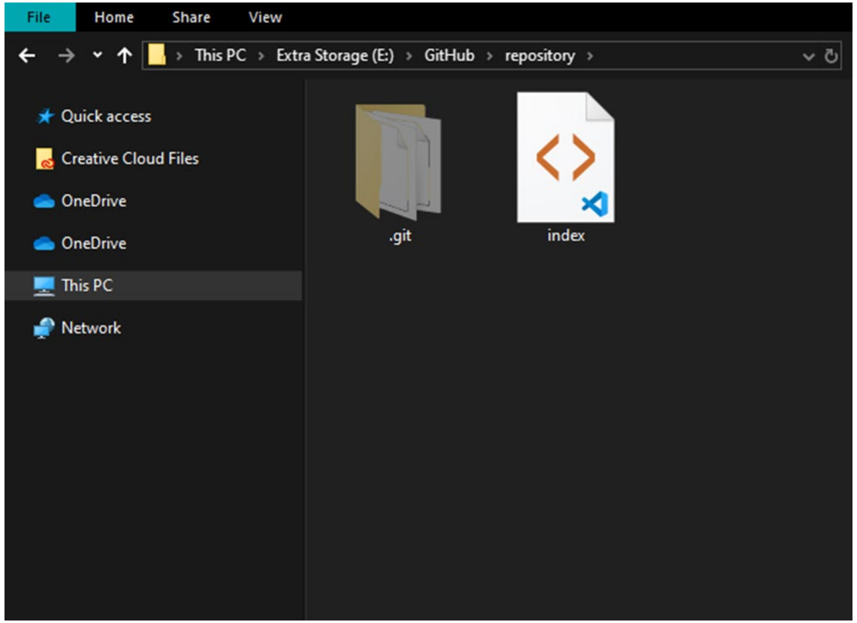
Version control is an integral aspect in development, simplistically, think of it as save points in a game.

A Repository or repo tracks all the changes you and your teammates do in your project.

Git VS GitHub

What we are doing now is a **Git repo** this is local to your device.

In the next few days, we'll use this same repo and add it on GitHub.



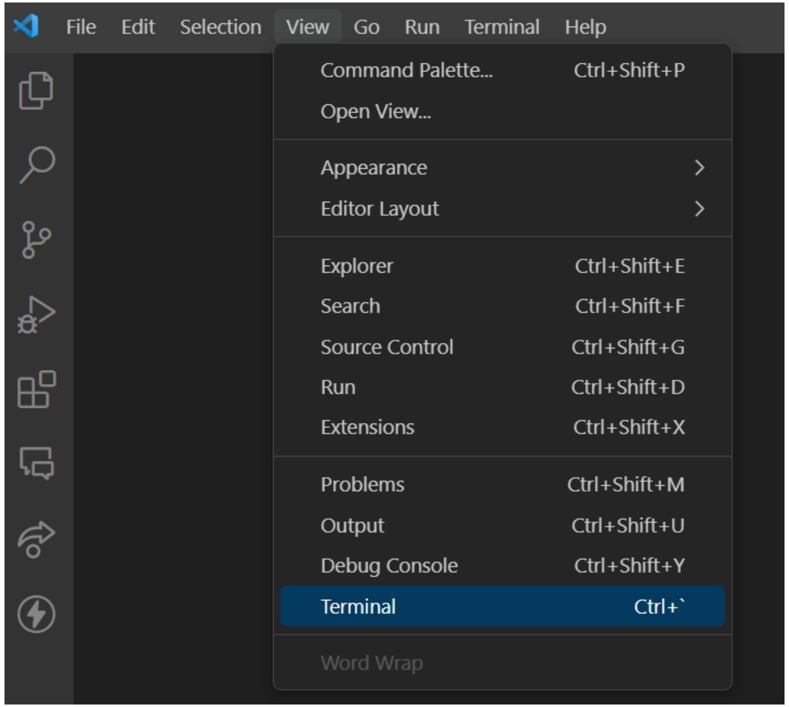
Let’s set up a Git repo!

- Open the Terminal (Ctrl + `)
- Make sure you are in the right directory, if not type:

```
cd yourproject/path
```

- Once you are in the right directory, initialise your repository.

```
git init
```



Saving your changes

Whenever you make changes, on top of just saving your files, it is best to **add** and **commit** them to your repo as frequently as you could.

This ensures that you are tracking all the changes and saves your future self from having potential headaches.

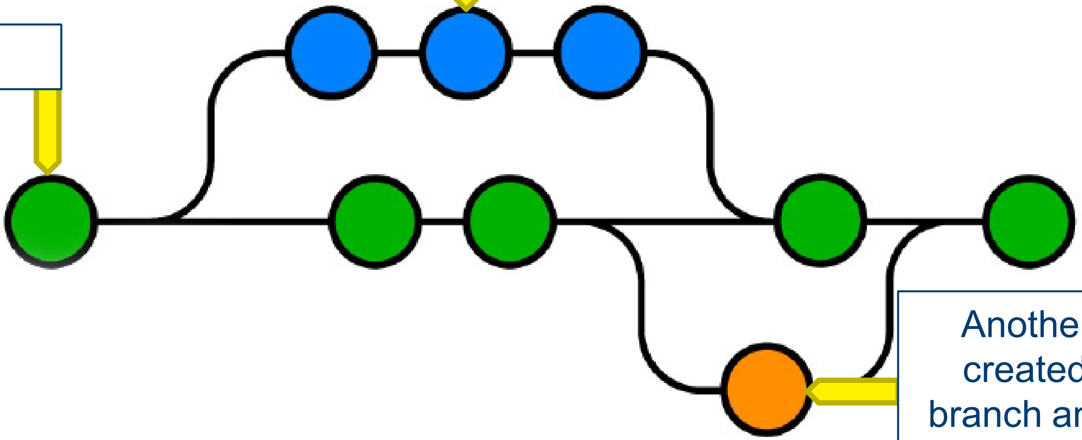
```
git add .
git commit –m “a detailed description of the change”
```

Git Practice: Making a Branch

- Open the folder of the repo we made yesterday in your code editor.
- If you haven’t added and commit, please do so now.
- Open terminal and type: `git checkout –b day-1`

This command **creates** and **switches** to the new branch you just made. This will have the exact same content as your master.

Understanding Git Branches



Feature created on new branch and merged back to master

Main branch

Another feature created on new branch and merged back to master

https://www.nobledesktop.com/learn/git/git-branches

Understanding Git Branches

- Git branches allow us to make changes to parts of our project safely outside of the master branch
- These branches are used for experimentation, making edits and changes, and must be approved before merging back to the master branch
- The master branch should be reserved as the branch that holds final production code
- We will discuss how to merge your branches on Day 3.

Git Practice: Making a Branch

- Switch back to master and make a new branch called **day-2.**
- In the terminal, regardless of which branch you're in type the following one at a time, what do you see?

```
git log
git branch
```

- Don't forget to add and commit!
- To **switch** between the branches, use: `git switch <branch name>`

Assignment #2

Assignment #2

- Please refer to Assignment #2 in the Learning Hub.
- To submit the assignment, you can do **one** of these:
  - Have me check your assignment in class before 4pm.
  - Zip today’s **folder** and upload it to the Learning Hub before next class.
- If you have questions or need guidance, just ask!

Working Together

You can work on and submit this assignment alone, in pairs, or in a group of 3.

I encourage you to work with classmates that you haven’t worked with yet.

You will have multiple group projects throughout the program, so start to get to know each other now!

Project #1

Now that we have covered the basics of HTML, you can begin to start on Project #1.

This is a single webpage and you should be able to create the HTML for it now. You can also set up your Git repository for it.

The next few classes will cover more CSS so you can style it as we go.

On Day 3, you will also learn how to connect your repo to GitHub and how to push changes. **Both projects have GitHub requirements.**

Resources

MDN Web Docs – HTML Elements List
https://developer.mozilla.org/en-US/docs/Web/HTML/Element

MDN Web Docs – HTML Attributes List
https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes

MDN Web Docs – CSS
https://developer.mozilla.org/en-US/docs/Web/CSS

MDN Web Docs – CSS Reference List
https://developer.mozilla.org/en-US/docs/Web/CSS/Reference

Video Tutorials

LinkedIn Learning – CSS Essential Training
https://www.linkedin.com/learning/css-essential-training-3/

LinkedIn Learning – Introduction to CSS
https://www.linkedin.com/learning/introduction-to-css-14934735

Tools

CSS Selectors Cheat Sheet
https://frontend30.com/css-selectors-cheatsheet/

CSS Ruler
https://katydecorah.com/css-ruler/

HTML Tags Memory Test
https://codepen.io/plfstr/full/zYqQeRw

QUESTIONS & ANSWERS