SSDP 1100 – Day 2

Course:	Front-End Web Development Essentials
Instructor:	Gabbie Bade

BRITISH COLUMBIA
INSTITUTE OF TECHNOLOGY

BCIT

Best Practices Recap

- 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 _).

Agenda

- More HTML Elements
- HTML Class Attribute
- HTML Forms
- Introduction to CSS
- Final Project

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

Videos on the Web

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

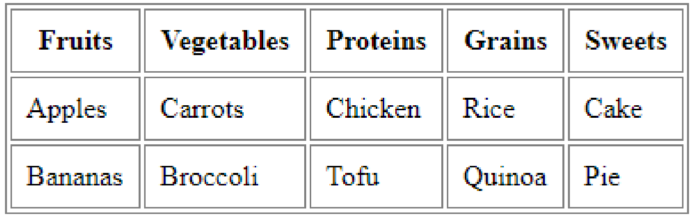
Consider dimensions, file size, and file format. We will discuss formats shortly.

For large video files, use YouTube or Vimeo.

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

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.

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.

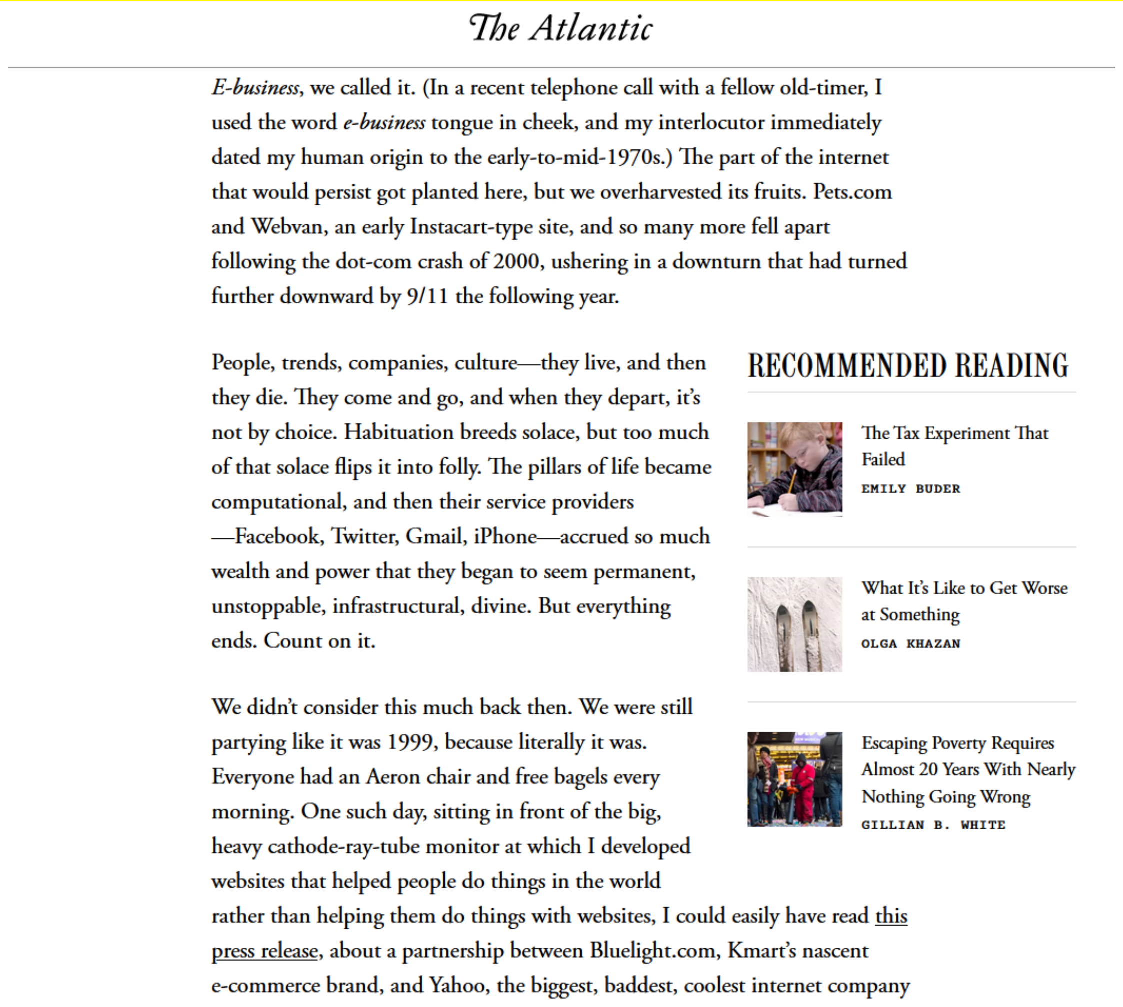
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/time

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

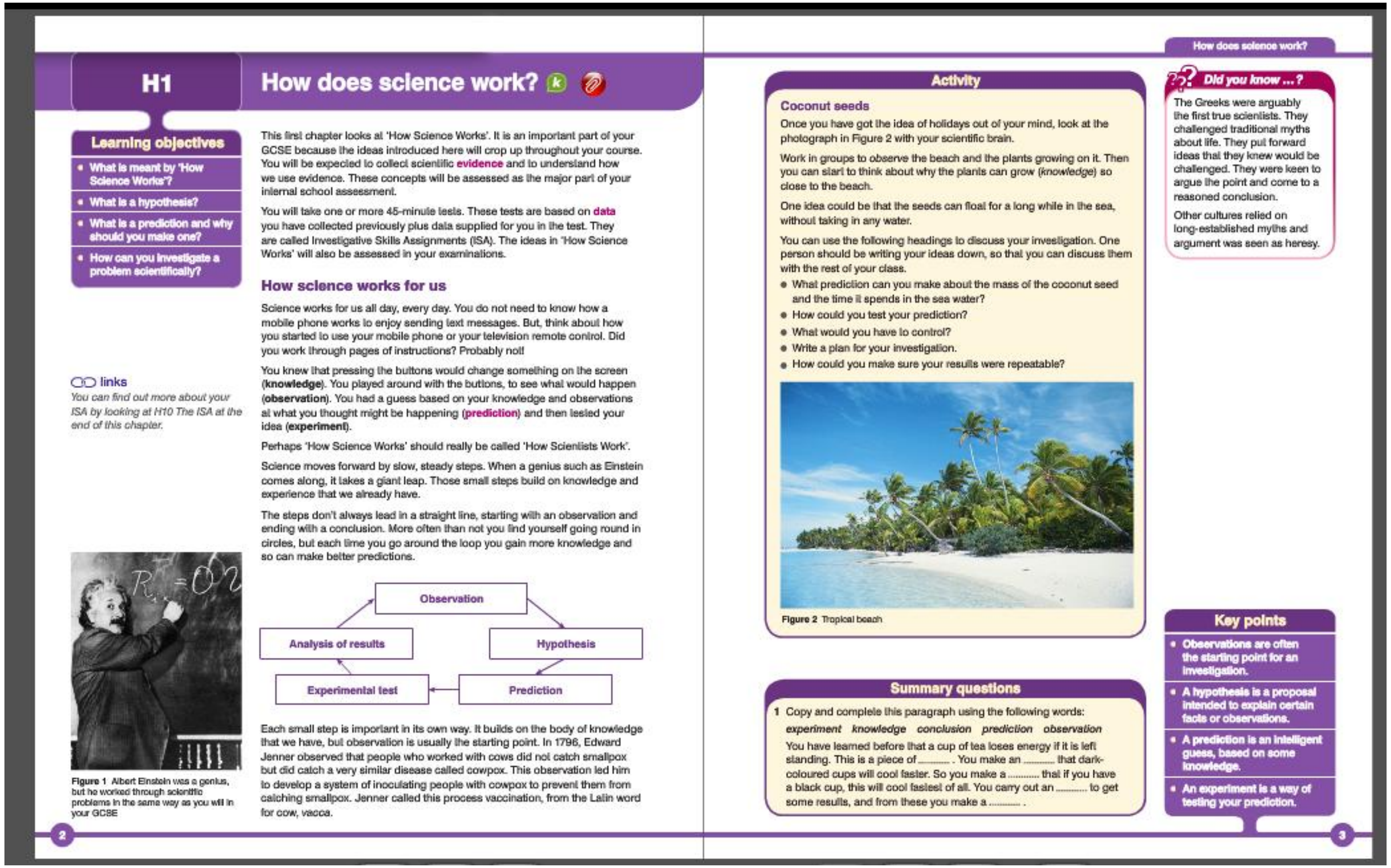
Escaping Poverty Requires Almost 20 Years With Nearly Nothing Going Wrong
GILLIAN B. WHITE

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.

$$\begin{tikzcd}
 & \text{Observation} \arrow[dr] & \\
\text{Analysis of results} \arrow[ur] & & \text{Hypothesis} \arrow[d] \\
\text{Experimental test} \arrow[u] & & \text{Prediction} \arrow[ll]
\end{tikzcd}$$

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

2

How does science work?

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.

3

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

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…

$$\texttt{<h2 class="}\overbrace{\texttt{section-title}}^{\text{Class 1}}\underbrace{\texttt{ }}_{\substack{\text{Add a space}\\ \text{between classes}}}\overbrace{\texttt{featured}}^{\text{Class 2}}\texttt{">}$$

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.

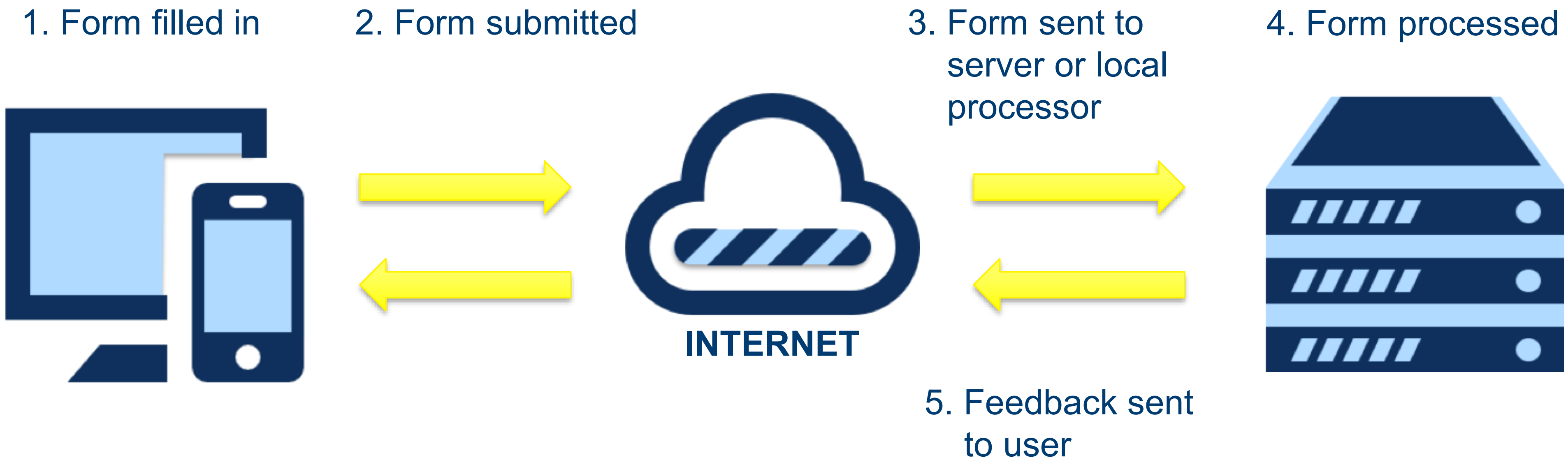
Classes can also be used in JavaScript to add functionality to your webpage.

HTML Forms

Form Tag

- Forms allow web sites users to interact with your web site
- The form tag wraps form elements and enables the sending of data inputted into a form to a web server
- You can use forms to do the following:
  - Collect data from users
  - Logging into a site
  - To perform calculations
  - To contact you
  - To pay for items online
  - To search for something online

How a Form Works



All images are from iconmonstr.com

How a Form Works

1. User fills in the form
2. User submits the form
3. Data from the form is sent to the form processor which is a server script that runs on a web server
   - Not all forms are sent to the server, some forms such as calculators can be processed locally and never require data to be sent to a server
4. The form is processed
5. Feedback can optionally be given to the user to assure them that their information has been received

The Form Tag

The form tag wraps form elements and enables the sending of data inputted into a form to a web server

The action attributes tells the browser where to send the form data

The method element determines how the data is sent to the server.

The name element provides a way to identify the form to the server.

```
<form action="processor.php" method="post” name=“contact”>
      <!– Form elements go here... -->
</form>
```

Form Tag

- The form tag can include the following attributes:
  - **action** - The location of the form processing file
    - Form processing is usually done on the server side (to be covered in PHP)
    - Not all forms require submission to a server
  - **method** - Determines how the data is sent from the client to the server
    - The values are **GET** (useful for search forms) and **POST** (the most common method for most forms).
  - **name -** Provides a way to identify the form to the server

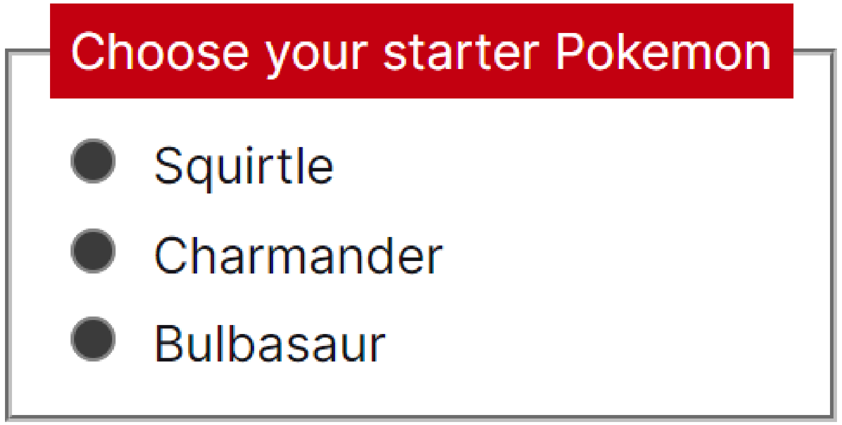
Should I use GET or POST?

- Setting the method attribute to “get” means the form data will be sent out in the open in the URL address
  - Do not use “get” for sensitive form data like credit cards, usernames, emails, passwords and other personal information
  - Use ”get” for search forms if you want to allow the user to bookmark a search query
- Use “post” for most forms
  - Post data is stored in the request body of the HTTP request[1]

1 https://www.w3schools.com/tags/ref_httpmethods.asp

Fieldset Tag

- The fieldset tag creates an element that **optionally** groups form controls and labels under a common name or caption.
- It nests a <legend> that captions the <fieldset>.
- Draws a box around the grouped controls and labels.


Choose your starter Pokemon

● Squirtle

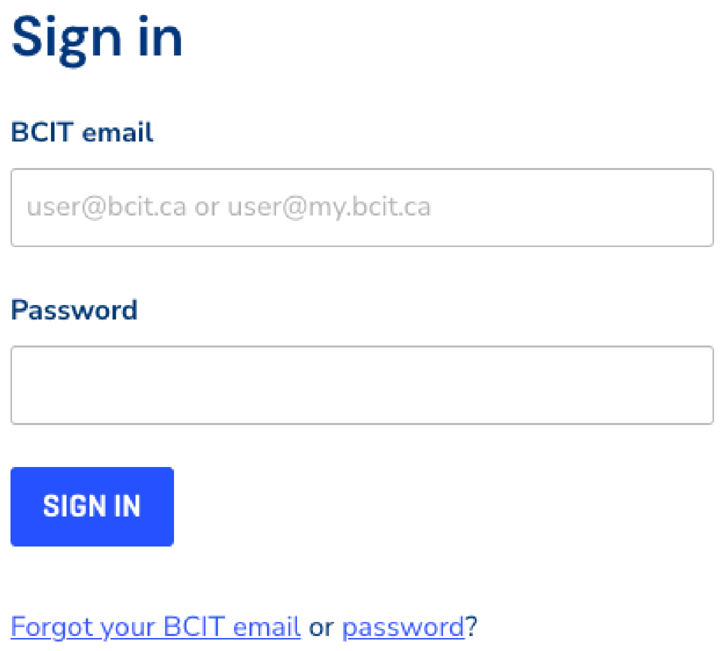
● Charmander

● Bulbasaur

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

Input Tag

- The input tag creates an element that can be used to input data by a user of a web site
- The input tag can have several “types” which determine the type of data that should be entered into the input element
- Some input types:
  - checkbox, email, tel, radio, text, password, number, button and submit


Sign in

BCIT email

user@bcit.ca or user@my.bcit.ca

Password

Forgot your BCIT email or password?

Input Tag

- Some common input tag attributes include:
  - **type** - Determines the input type
  - **name** - Identifies the input for the server
    - Used by radio and checkbox input types to group them together
  - **disabled** - Used to disable an input and to not submit any value to the server
  - **readonly** - Used to make an input read only and to send a value to the server
  - **required** - Used to tell the browser to require and validate that this input has been filled in

Read more about input tag attributes

The Input Tag

The "type" attribute determines the type of input

Used to identify the input to the server. It is also used radio and checkbox input types to group them together

```html
<input type="text" id="firstname" name="fn">
```

The "id" attribute is used to link a label to an input

Label Tag

- The label tag provides a label for an input element
- The label tag can be linked to a form element by using the “for” attribute

  ⚠️ The “for” attribute value should match the “id” attribute value of the input element that it is a label of

Read more about input tag attributes

The Label Tag

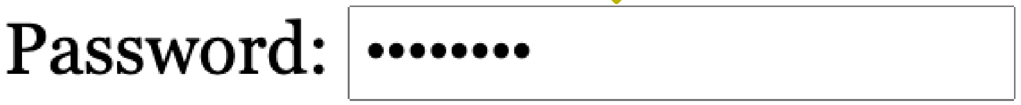
```
<label for="firstname">Firstname: </label>
<input id="firstname" type="text" name="fn">
```

The “for” attribute value must match the “id” of the input element that it is a label of

Input Type - Password

- Used for password inputs
- Text entered by the user is hidden visually by the browser

```
<label for="pw">Password: </label>
<input id="pw" type="password" name="pw">
```



Password: •••••••••

Textarea Tag

- The “textarea” tag is used for inputs where you need the give the user the ability to enter multiple lines of text or a large amount of text.
- Useful for comments

```
<label for="comments">Comments: </label>
<textarea id="comments" name="comments"></textarea>
```

Input Type - Submit

- An input with the “type” of “submit” is used to submit the form data
- When the “submit” input is clicked on the browser will send the form data to the location specified by the “action” element on the ”form” element
- The “value” attribute determines what text appears in the button

```
<input type="submit" value="Submit">
```

Input Type - Checkbox

- Allows the user to select multiple options with a ”checkbox” interface
- Use “checkbox” when you want to give the user the ability to select **multiple options**

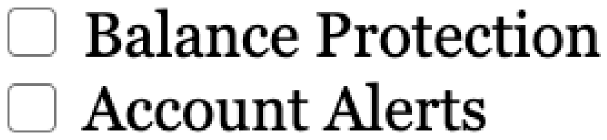
⚠️ To create a “checkbox” group make sure the each “checkbox” input has the same “name” attribute value

Input Type - Checkbox

To create a “checkbox” group make sure the each “checkbox” input has the same “name” attribute value

```
<input type="checkbox" name="a-features" id="b-pro" value="balance protection">
<label for="b-pro">Balance Protection </label>

<input type="checkbox" name="a-features" id="a-alerts" value="account alerts">
<label for="a-alerts">Account Alerts </label>
```



Input Type - Radio

- Allows the user to select one option from a group of options with a ”radio” button interface

- Use “radio” when you want to give the user the ability to **select only one option** from a group of options

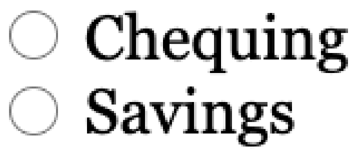
⚠️ To create a “radio” group make sure the each “radio” input has the same “name” attribute value

Input Type - Radio

To create a “radio” group make sure the each “radio” input has the same “name” attribute value

```
<input type="radio" name="account-type" id="chequing" value="chequing">
<label for="chequing">Chequing </label>

<input type="radio" name="account-type" id="savings" value="savings">
<label for="savings">Savings </label>
```



Input Type - Hidden

- An input with the type of “hidden” will not be displayed visually by the browser
- Hidden inputs form data is sent to the server for processing
- Useful to send form data to a server that you do not need or want the user to enter into a form

```
<input type="hidden" name="user-id" value="abc12345">
```

Select List

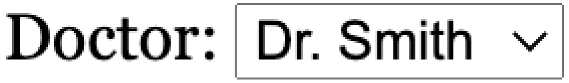
- Creates a select list form element
- Can be configured to allow the user to select only one option or multiple options
- The size attribute allows you to configure the number of options that are visible to the user before they activate the dropdown

⚠️ Will look different depending on the browser and the operating system

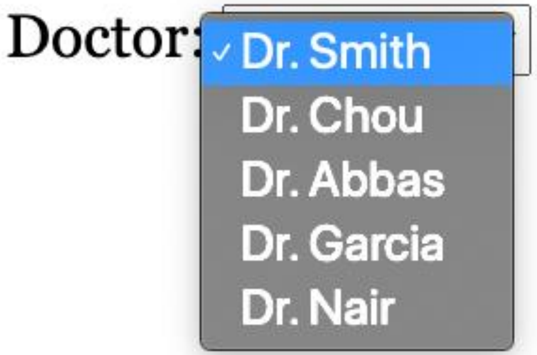
⚠️ Difficult to fully customize the appearance, but some styling is possible

Select List

Select list in the closed state (Chrome Browser, macOS)



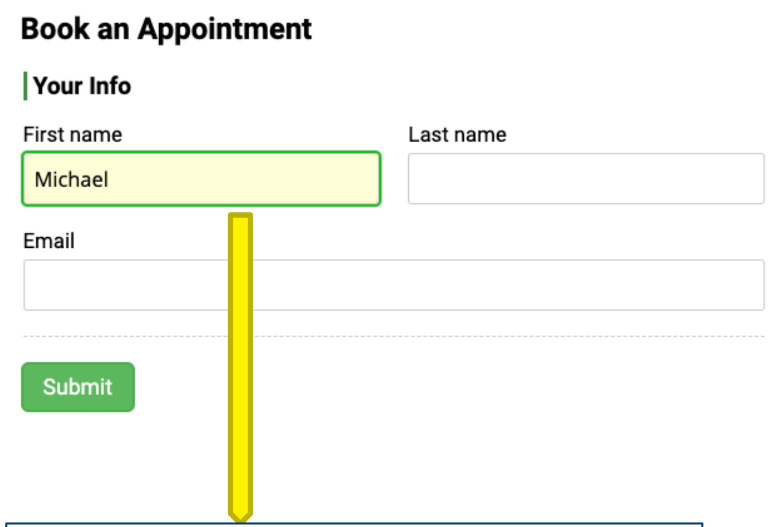
```
<label for="doctor">Doctor: </label>
<select id="doctor" name="doctor">
  <option value="Smith">Dr. Smith</option>
  <option value="Chou">Dr. Chou</option>
  <option value="Abbas">Dr. Abbas</option>
  <option value="Garcia">Dr. Garcia</option>
  <option value="Nair">Dr. Nair</option>
</select>
```



Select list in the open state (Chrome Browser, macOS)

Focus State

- One use of the ":focus" CSS pseudo-selector is style form elements when a user clicks into or focuses into a form input element
- Changing the style of the input that the user has focused into helps inform the user where they are in the form



This input has additional styles applied when the input is focused using the ":focus" pseudo selector

Read more about pseudo classes

Focus State

```
input[type="text"]:focus,
input[type="email"]:focus {
  background-color: #fdfdd6;
  outline: none;
  box-shadow: 0 0 1px 2px #13b613;
  border-radius: 3px;
}
```

The ":focus" pseudo selector applies when a user clicks into or focuses into an input element

Placeholder

- The placeholder attribute should be used to provide a short hint (a word or short phrase) intended to aid the use with data entry when the control has no value[1]
- Use the placeholder attribute with caution (see next slide)
- The placeholder attribute is NOT a substitute for label elements (see next slide)
- Give the below linked article a read to learn about some of the issues with the placeholder attribute
  - https://www.smashingmagazine.com/2018/06/placeholder-attribute/

Placeholder Attribute as Label

Note: Hiding labels and using placeholders inside form input elements as labels should be used with caution. Read the Nielsen Norman Group article Placeholders in Form Fields Are Harmful for reasons to perhaps not use this technique.

Placeholder Attribute as Label

Warning: Never omit label tags in your markup, even if you are using this placeholder as label technique. People using screen readers rely on labels to use forms properly. **Placeholders are NOT a substitute for labels in your HTML.** Use CSS to visually hide your labels (DO NOT use display: none;). Use a visually hidden style such as that used by WebAim "sr-only" class. See the next slide for a CSS code snippet that will allow you visually hide an element but still make the element accessible to screen readers.

Placeholder Attribute as Label

DO NOT BE A LAZY DEVELOPER!!! Always use label tags and tie them to their corresponding inputs with a ***for*** attribute on the label tag that matches the ***id*** attribute on the corresponding input element, even if your labels are not visible.

WebAIM sr-only Class

You can also use this code snippet from WebAIM to visually hide screen reader only elements using CSS.

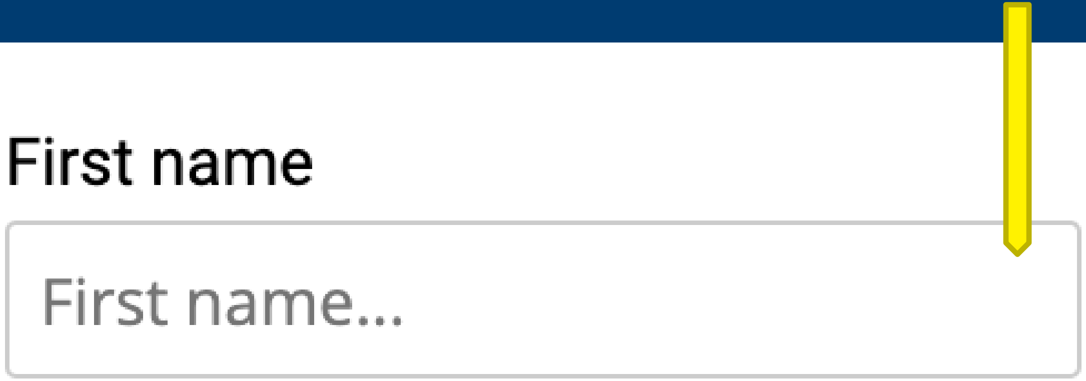
WebAIM is an excellent resource for accessibility tools and tips.

```
.sr-only {
  position: absolute;
  width: 1px;
  height: 1px;
  top: auto;
  left: -10000px;
  overflow: hidden;
  border: 0;
}
```

Code snippet from this article:
https://webaim.org/techniques/css/invisiblecontent/

Placeholder Attribute

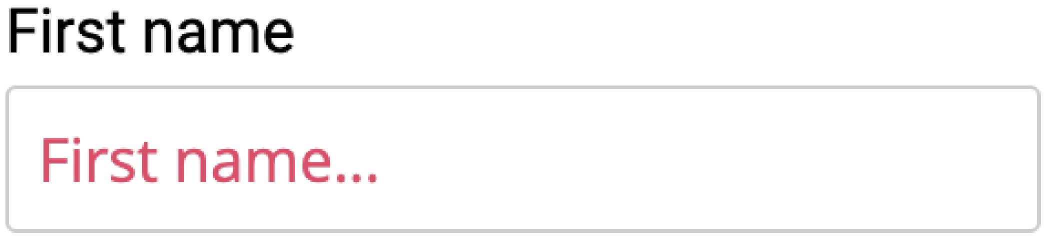
```
<label for="fn">First name</label>
<input type="text" id="fn" name="fn" placeholder="First name...">
```



BCIT

Styling the Placeholder Attribute

Styling the Placeholder can be tricky as different browsers support a slightly different syntax for the "::placeholder" pseudo selector


First name

First name...

HTML input element with a placeholder attribute styled to with a red text color

Styling the Placeholder Attribute

CSS syntax for styling the placeholder attribute for Chrome, Firefox and Chromium based Microsoft Edge

```
input::placeholder {
  color: #d84f69;
  /* Firefox fades the text using opacity
    - If you want a consistent look between
      browsers, set the opacity to 1
  */
  opacity: 1;
}
```

Styling the Placeholder Attribute

CSS syntax for styling the placeholder attribute for non-Chromium based Microsoft Edge

```
input::-ms-input-placeholder {
  color: #d84f69;
}
```

Styling the Placeholder Attribute

CSS syntax for styling the placeholder attribute for non-Chromium based Microsoft Edge

```
input::-ms-input-placeholder {
  color: #d84f69;
}
```

Styling the Placeholder Attribute

CSS syntax for styling the placeholder attribute for IE 10+

```
input:-ms-input-placeholder {
  color: #d84f69;
}
```

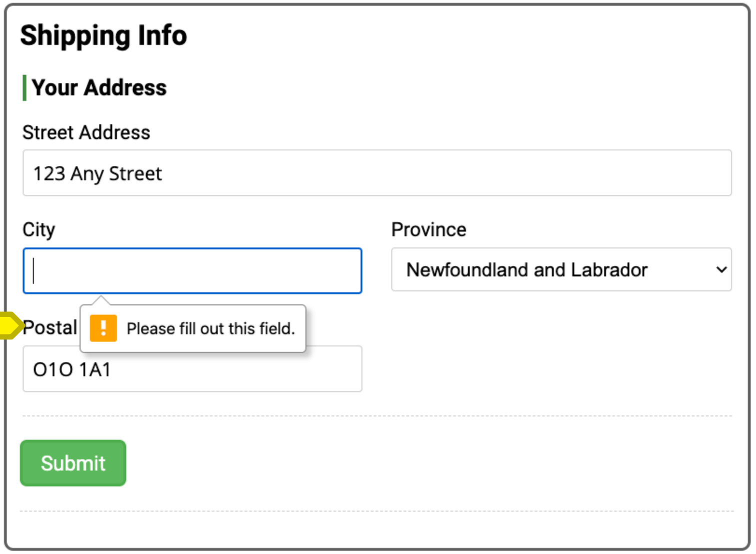
BCIT®

HTML Form Validation

- HTML5 introduced simple HTML form validation without any need for JavaScript
- You can use the ”required” attribute on an input to make sure the user enters data on an input element
- If you want more custom looking form validation then you will need to turn to JavaScript
- **Note:** HTML form validation using the built-in HTML validation or using custom JavaScript validation is not secure. Always validate the data a 2nd time on the server when receiving data from an HTML form

HTML Form Validation

```
<label for="city">City</label>
<input type="text" id="city" name="city" required>
```



Shipping Info

Your Address

Street Address

123 Any Street

City

Province

Newfoundland and Labrador

Postal

Please fill out this field.

O1O 1A1

Submit

Form Validation with the Pattern Attribute

- Using the required attribute alone allows for basic validation to make sure the user at least inputted something into an input
- If you require something a bit more custom, you can use the “pattern” attribute to make the browser validate an input against a regular expression
  - A **regular expression** is a pattern that the computer uses to test a string of text against
  - A handy reference for RegEx
- With a pattern attribute you can validate for Credit Card numbers, postal codes, phone numbers and other common types of data
- Use the title attribute to provide helper text to the user if they input invalid data

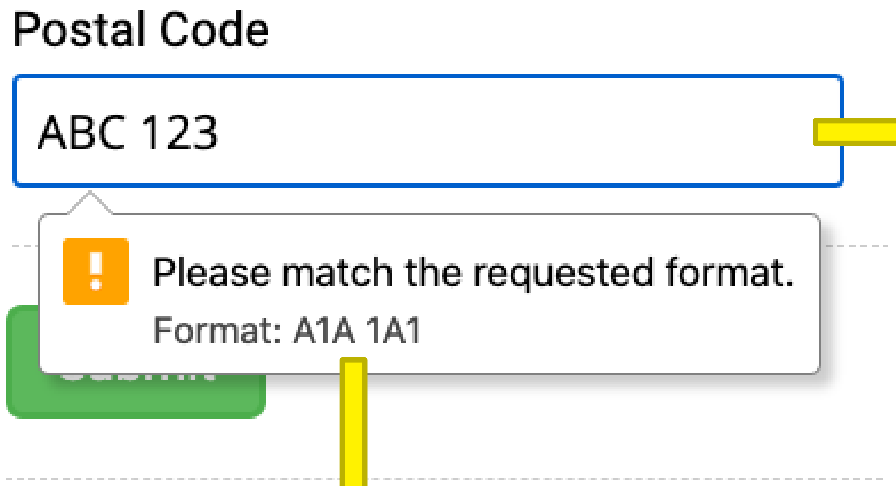
Form Validation with the Pattern Attribute

The pattern attribute tells the browser to test the input entered by the user against the regular expression set in the pattern attribute. The pattern here will test for a Canadian postal code (A1A 1A1)

```
<input type="text"
    id="postal-code"
    name="postal-code"
    pattern="[A-Za-z][0-9][A-Za-z](\s)?[0-9][A-Za-z][0-9]"
    title="Format: A1A 1A1"
    required>
```

You can use the title attribute to display a helper message to the user if they enter incorrectly formatted data

Form Validation with the Pattern Attribute


Postal Code

ABC 123

Please match the requested format.

Format: A1A 1A1

This input is being validated against a custom pattern set via the “pattern” attribute on the input element. This input is being validated for a Canadian postal code (A1A 1A1).

The text “Format: A1A 1A1” comes from the title attribute on the input element

- Visit http://html5pattern.com/ to get some common pattern values for things such as telephone numbers, credit cards, postal codes and other common types of information

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.

$$\overbrace{\texttt{style="color: red;"}}^{\text{Inline CSS}}$$

```
<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

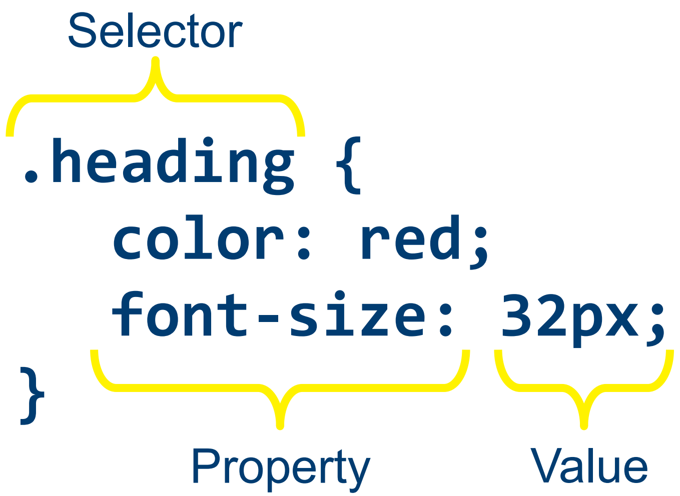
```
<head>
   <meta charset="UTF-8">
   <title>Page Title</title>
   <link rel="stylesheet" href="styles/styles.css">
</head>
```

Within the styles.css file

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

CSS Syntax

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



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

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.

$$\overbrace{\texttt{/*}}^{\text{Start}}\texttt{ I am a CSS comment }\overbrace{\texttt{*/}}^{\text{End}}$$

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/

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

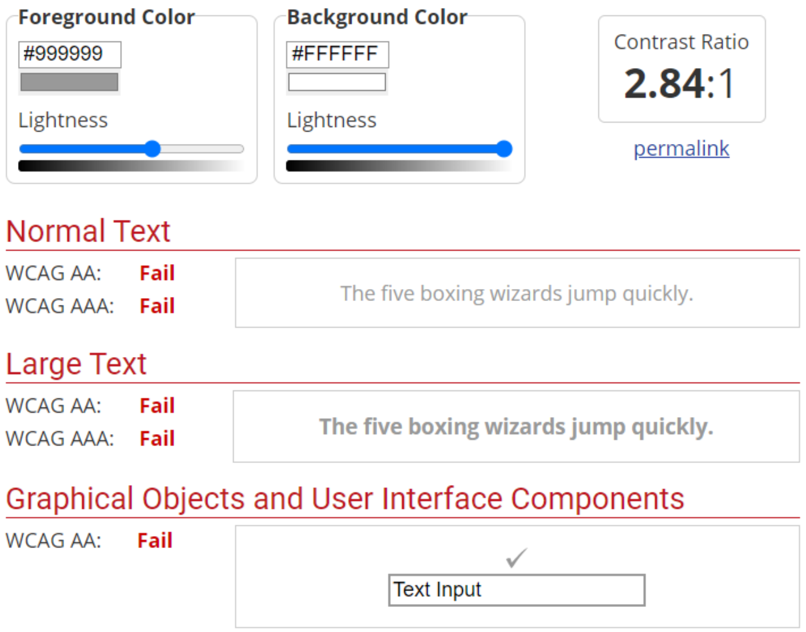
The color formats all do the same thing and browsers support them all equally so use the one that you prefer.

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
WCAG AAA:	Fail

The five boxing wizards jump quickly.

Large Text

WCAG AA:	Fail
WCAG AAA:	Fail

The five boxing wizards jump quickly.

Graphical Objects and User Interface Components

WCAG AA:	Fail

✓
Text Input

Accessible Color Tools

Contrast Checker

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

https://chromewebstore.google.com/detail/wcag-color-contrast-check/

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.

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.

Final Project

Now that we have covered the basics of HTML, you can begin working on your final project.

This is a multi-page website and you should be able to create the HTML for it now.

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

Resources

Forms (MDN)

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

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

Pseudo Classes (MDN)

https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes

Resources

Accessibility of Placeholder Attribute

https://www.boia.org/blog/is-placeholder-text-essential-for-form-accessibility

https://webaim.org/techniques/css/invisiblecontent/

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