Front-End Web Development

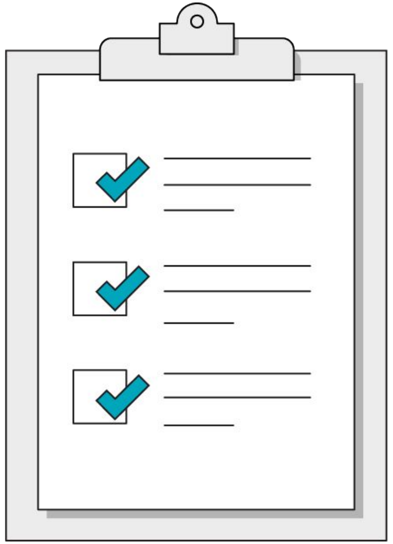
Semantic HTML and Introduction to CSS

GENERAL ASSEMBLY

Today's Learning Objectives

In this lesson, you will:

- Choose semantic HTML tags to define and organize content.
- Use CSS to apply style to webpages.
- Learn the basics of CSS syntax, including selectors and style rules.



Front-End Web Development

Semantic HTML

Guided Walk-Through:

HTML Review

10 minutes

Let's review tagging content with a quick walk-through. The example solution contains plenty of tags you haven't seen yet — don't worry, they'll come later! In the meantime, it's good to challenge yourself by analyzing unfamiliar content.

Starter code:

https://codepen.io/GAmarketing/pen/JjjBWYd

Solution code:

https://codepen.io/GAmarketing/pen/XWWBMmL

Semantic Elements

Element	Description
`main`	Indicates the main content of a page.
`header`	Content placed above and contextualizing the main content.
`nav`	A navigation section, typically containing links or tabs.
`section`	Typically indicates one of several, equally placed segments of content.
`aside`	A sidebar of content placed next to and supplementing an element.
`div`	A “division” of the page; a very generic, non-semantic element.

Discussion:

Semantic HTML

10 minutes

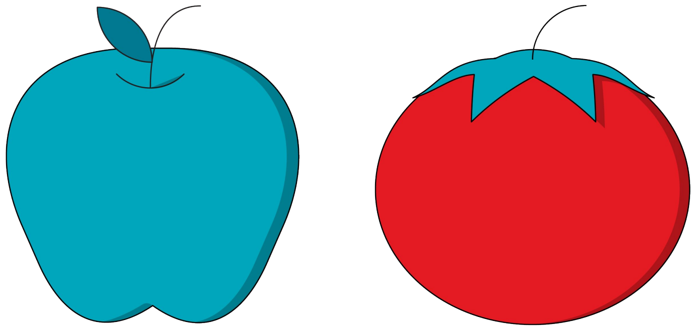
How are the elements in the following HTML nested, and why would a developer write HTML this way?

```
<nav>
    List of Links
</nav>
<main>
   <section>
      <div>
         Content A
      </div>
      <div>
         Content B
      </div>
   </section>
</main>
```

Cheat Sheet: Semantic Elements

It can be confusing to know when to use one element versus another, because they all do the same thing. This flowchart can help you decide.

There isn't "one right way" to structure HTML. It's subjective, so do your best.



Partner Exercise:

Meatless Burgers Article

15 Minutes

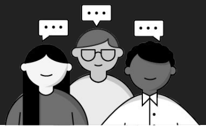
Let's recreate a section of a real website using semantic HTML tags.

Starter code:

https://codepen.io/GAmarketing/pen/OJJwpMO

Solution code:

https://codepen.io/GAmarketing/pen/KKKBWVG



Discussion:

HTML Basics

10 minutes

Before moving on to CSS, let’s talk about what we’ve learned about HTML so far.

What are the most important points?

There’s a lot of new information already and lots more to come — it’s important to prioritize and prevent information overload!

Front-End Web Development

CSS Basics

CSS: Let’s Talk Basics

Guided Walk-Through:

CSS Basics

As we introduce CSS, have this handy reference to the most basic selectors, properties, and values open in a separate browser window.

Reference code:

https://codepen.io/GAmarketing/pen/QWWBpbB

The Anatomy of CSS

Selector

(What we are styling)

```
p {
    color: blue;
    font-size: 20px;
    font-style: italic;
}
```

Declaration Block

(Our styles)

Discussion:

What Styles Look Like

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

- What are the individual parts of this CSS rule?
- How do you think we could change or extend them?
- What other style changes do you think we could make?

Selectors and Specificity: Elements < Classes < IDs

Selectors tell the browser which elements should be impacted by the style. CSS rules are applied in order of **specificity**.

- **Element selectors**
  - The least specific selectors; apply to all elements of the given tag type.
  - Syntax is simply the name of the element: **`h1`**.
- **Class selectors**
  - Apply to all elements with the chosen class attribute.
  - Syntax starts with a dot, followed by the name of the class: **`.className`**.
- **ID selectors**
  - Apply to all elements with the given ID attribute.
  - Syntax starts with a hashtag, followed by the name of the ID: **`#idName`**.

Multiple Selectors

```
p, .class-name {
  color: white;
  text-decoration: underline;
}
```

You can have...

- Two selectors together, mixing HTML elements and custom classes if you like. The example above targets **`p`** tags and any element with the class name **`.class-name`**.
- An unlimited number of selectors can go together — just separate each new selector with a comma.

Parent Selectors

```
.sidebar p {
  color: white;
  text-decoration: underline;
}
```

You can target elements according to their specific parent element by separating two selectors with a space. The example above targets all **`p`** tags contained within an element with the class of **`.sidebar`**.

There are also other selectors, called **combinators**, that target elements based on their specific relationships to other elements.

General Selectors

Cascades Throughout HTML

Using selectors that are generic, i.e., represent a “trunk” of the tree, will result in styles that cascade and affect many elements.

Good uses: Color themes, font styles, and other stylistic elements repeated throughout the site.

Specific Selectors

Hit Limited Amount of Elements

Classes and IDs are useful for styles that are only applied in small, limited sections of the website.

Good uses: The spacing and placement of individual elements and styling of non-repeated elements.

Selectors Targeting Multiple Attributes

```
div.my-class {
  color: white;
  text-decoration: underline;
}
```

You can also combine several selectors together in order to target only the elements that match all of the selectors included. The example above targets only **`div`** elements with the **`.my-class`** class.

Notice that there is no space between **`div`** and **`.my-class`**.

Properties

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

Properties are parts of the CSS spec that control style behavior. Most are logically named, but not all.

Properties will always be followed by a colon, then the value.

Properties + Values

```
p {
  color: white;
}
```

In this example, all **`p`** tags are having their **`color`** property set to the value of **`white`**. This changes their font color to the given value.

You must have at least one property/value pair per selector (otherwise, nothing happens). And they **MUST** end with a semicolon — always!

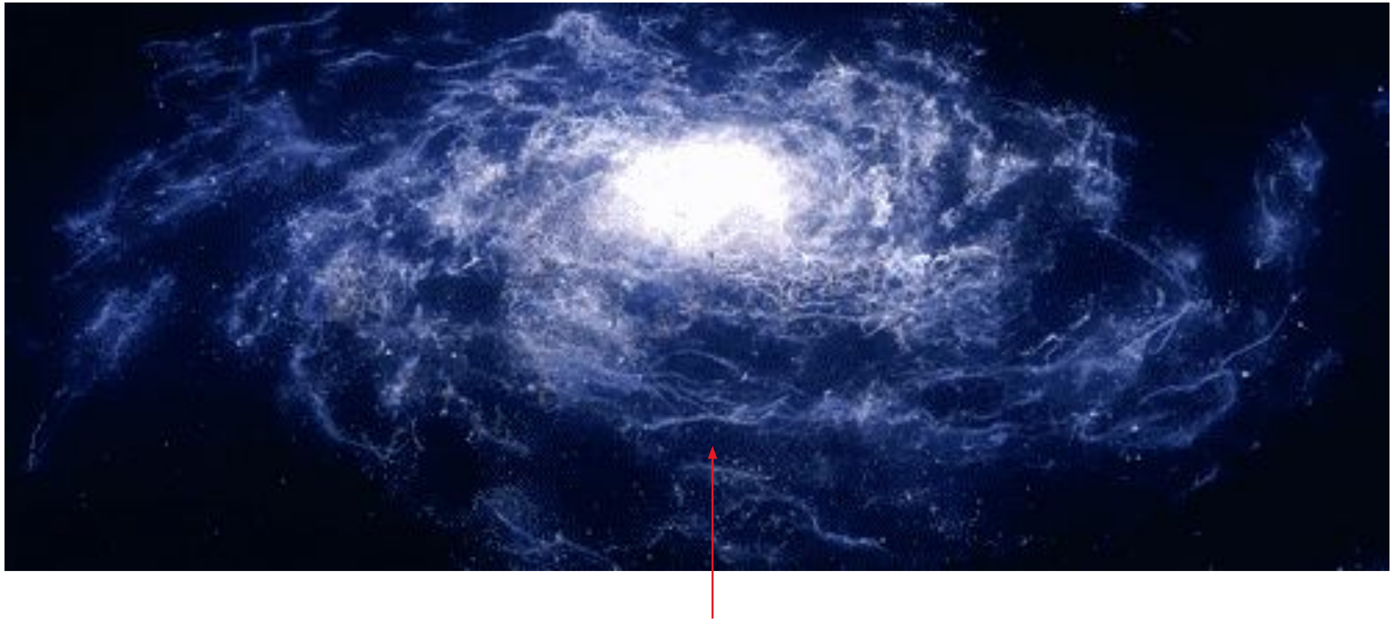
Properties + Values

```
p {
  color: white;
  font-weight: bold;
  padding: 15px;
  text-decoration: underline;
}
```

You can have an unlimited amount of property/value pairs in a style declaration.

If you find that many declarations share the same property/value pairs, consider declaring that property/value pair in a more broadly targeted selector.

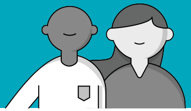
There Is a Whole Universe of CSS Properties and Values



A good dev knows what planet they are on.

Important CSS Properties

Property Name	Description
`background`, `background-color`	The style behind an element; can be images, colors, or gradients.
`border`	Styles the edges around an element.
`color`	Sets the color of text, NOT the background color!
`font`	Controls font family, size, style, and weight.
`height, width`	Sets how tall or wide an element should be in px, %, or ems.
`margin`	Creates space outside of an element.
`padding`	Creates space on the inside border of an element.
`text-align`	Defines the direction that text lines up in an element.



Partner Exercise:

Explaining CSS Selectors

15 Minutes

Walk through the following examples and take turns explaining what the selector, property and values are for each item.

Starter code:

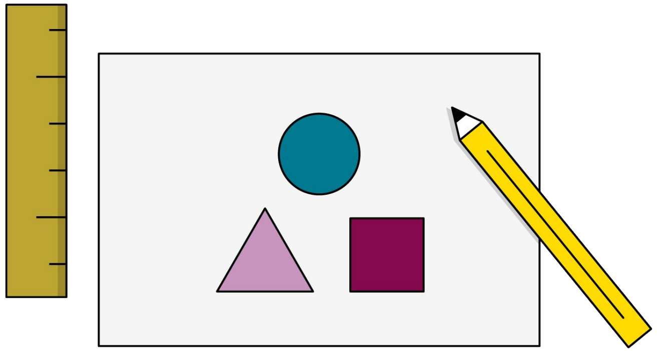
https://codepen.io/GAmarketing/pen/ZEEjeWZ

Solution code:

https://codepen.io/GAmarketing/pen/vYYaxKx

Quick Styling Tips

- Always put a new style on a new line.
- Try to alphabetize your styles or organize them logically by section.
- Use external style sheets that are linked in the head of your HTML document.
- Use whitespace so people can read your code.
- Comments are welcome in your CSS.



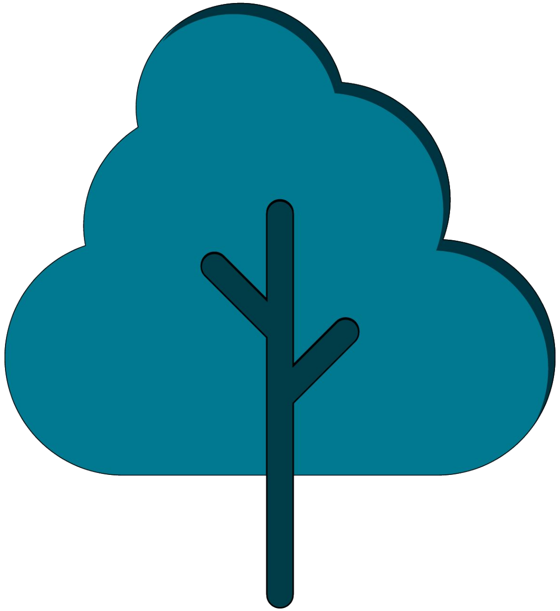
Front-End Web Development

Cascading: What Does It Mean?

Think of the Document Tree as a Real Tree

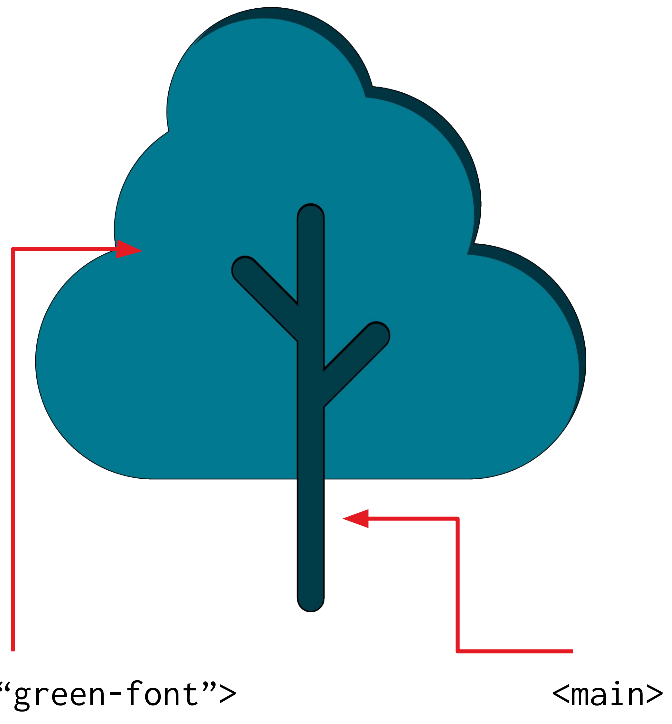
```
<main>
  <section>
    <div class="green-font">
      Content A
    </div>
    <div>
      Content B
    </div>
  </section>
</main>
```

DOM Tree

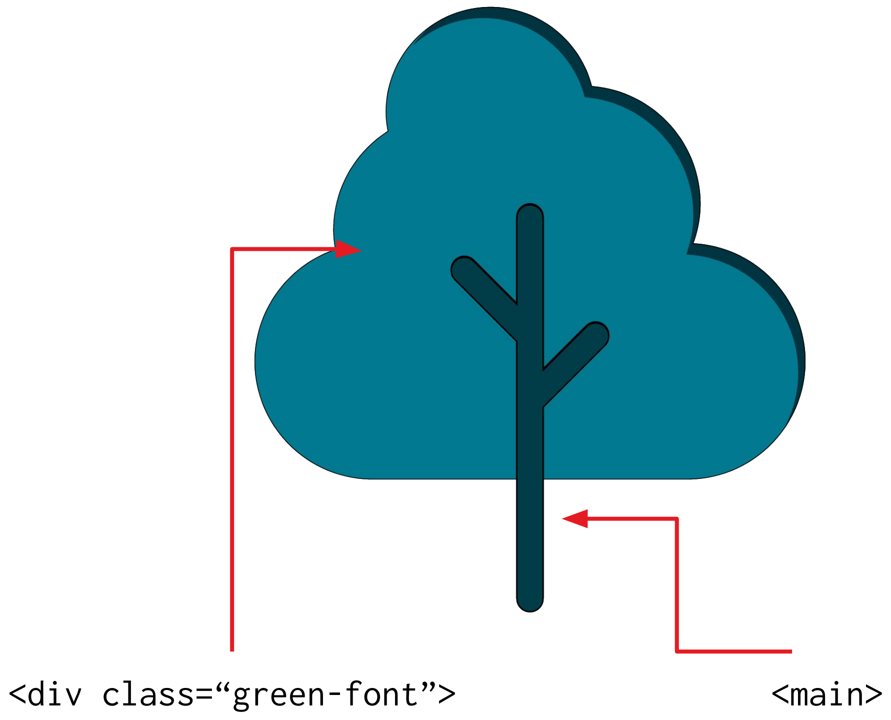


Real Tree

What If I Apply My Styles at Each of These Spots?



`<main>` Styles the Whole Tree — `.green-font` Styles a Small Leaf



Specific selectors override general selectors:

ID > Class > Element

`!important`

```
.make-it-white {
  color: white !important;
  text-decoration: underline;
}
```

- Using **`!important`** makes this style jump out of the DOM tree order and take precedence over all other styles.
- If you use this in your work regularly, **code reviewers will not be happy**. This is considered a brute-force, last-resort approach when all other selectors have failed.

Computers Out:

CSS Exploration

30 minutes

This example contains a real piece of a website. Apply all you've learned with CSS and experiment. Remember, there's more than one right answer!

Starter code:

https://codepen.io/GAmarketing/pen/xxxJqRw

Solution code:

https://codepen.io/GAmarketing/pen/rNNrZLg

Key Takeaways

CSS Cascades Through the DOM

- Style is defined in terms of **properties** and **values**.
- Using the right selectors can apply consistent styles across the whole page.
- Semantic HTML helps coordinate with CSS.

For Next Time

The CSS Box Model

- Complete Homework No. 1
- The box model defines spacing and sizing of elements.