Put it on the Web: An Intro to HTML/CSS

ZACHARY LLOYD

GC DIGITAL FELLOW

Presented by…

The Graduate Center Digital Initiatives brings together the work of scholars and technologists at the CUNY Graduate Center to pioneer new modes of inquiry that integrate digital tools and methods into the research, teaching, and service missions of the university. We do so by fostering communities of creative and critical practice among students, faculty, and staff through programs, events, workshops, grant-funded projects, consultations, and more.

https://gcdi.commons.gc.cuny.edu/

Getting Involved

Join the GCDI Group on the CUNY Academic Commons for updates on how to be involved. **cuny.is/group-gcdi**

Check the Event Calendar for upcoming events and workshops. **cuny.is/workshops**

Request a one-on-one consultation. **cuny.is/gcdfconsults**

Join the R Users Group (RUG) at **cuny.is/rug**

Join the GIS Working Group at **cuny.is/group-gis-working-group**

Join the Data Visualization Group at : **cuny.is/dvg**

Join the Python Users Group (PUG) at **cuny.is/pug**

Getting Started

Before we begin, make sure you have the following available to you on your computer:

- ✓ VSCode
- ✓ The base HTML file: “index.html”
- ✓ The HTML and CSS cheat sheets

Workshop Goals

1) Familiarize yourself with the anatomy of a webpage.

2) Understand the basics of the HTML and CSS markup languages.

3) Use HTML, CSS, and a text editor (VSCode) to build a small locally-hosted website.

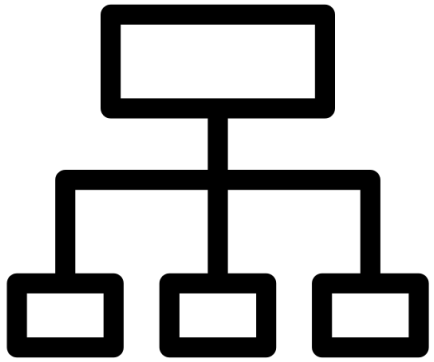
What are HTML and CSS?

- **HTML**, or *HyperText Markup Language*, is a markup language used to write web-based documents. It enables us to provide web browsers with information about the content of a document. We can, for example, indicate that some part of our document is a paragraph, image, heading, or link. The browser uses this information when displaying the document for users.

- **CSS,** or *Cascading Style Sheets*, is often used in conjunction with HTML. While HTML tells the browser what the different parts of a document are, CSS tells the browser what the parts of the document should look like. It is essentially a set of rules that are applied when rendering an HTML document. Its name—Cascading Style Sheets—refers to the fact that there is an order of precedence in how the browser applies CSS rules to a document. More specific rules overwrite less specific rules.

In general, keep in mind…

HTML provides the *structure* of a webpage.



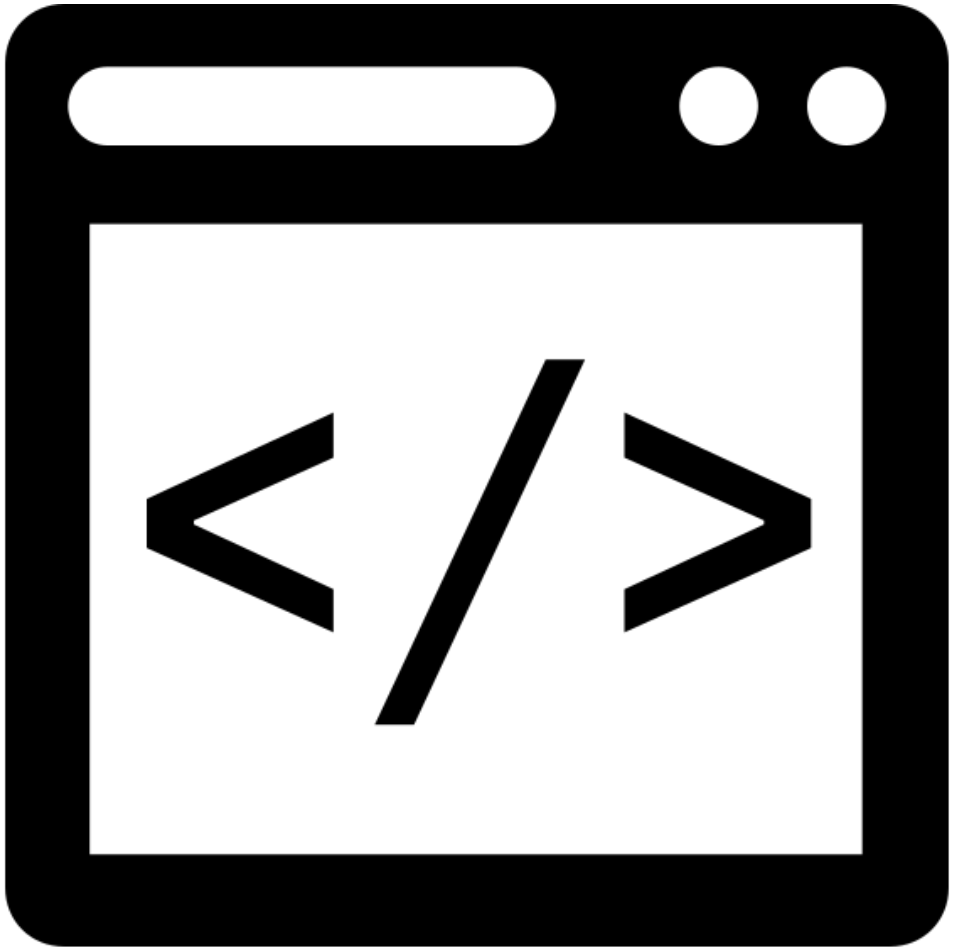
CSS provides the *style* of a webpage.

Local Hosting

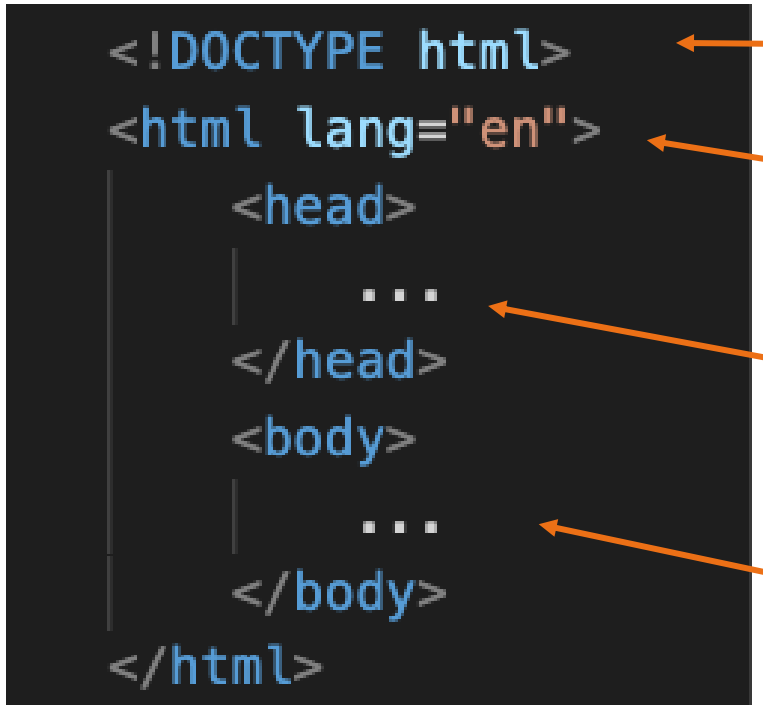
Together, these languages can be used to write and style a website using a text editor (such as VSCode) directly from your computer (or *locally*). No internet access is needed. (However, internet access is necessary if you plan on making your website available to the public.)

In our activities during this workshop we will focus on building locally-hosted websites using a simple LiveServer extension in VSCode. Again, these are websites that you can open on your web browser; however, they only exist on your own device and are only accessible to you. Locally-hosted websites are not yet on the public-facing Internet.

HTML



Basic Template for HTML



DOCTYPE tells the browser how to read the code

Root **HTML** Element: all content goes inside

Head contains basic info: title, authors, description, & link to CSS stylesheet

Body contains the content of the page (paragraphs, images, links, and more)

Tags

You probably noticed that most elements were wrapped in both **opening** and **closing tags:** <></>

Tags enclose content, and indicate what that content is.

For example, here is a paragraph element tag, with the content in between:

```
<p>This is a paragraph.</p>
```

Other elements are **self-closing**, meaning they don't need a closing tag, like this image element:

```
<img alt="This is an image" src="image.jpeg" />
```

HTML in VSCode:

index.html > html

```
1   <!doctype HTML>
2   <html lang="en">
3       <head>
4           <title>This is the page title.</title>
5       </head>
6   <body>
7       <h1>Here's an example large heading.</h1>
8       <p>This is an example paragraph of text.</p>
9   </body>
10  </html>
```

HTML rendered in the browser:

Demonstration!

Run a Live Server

1) Save the **index.html** page to a designated folder for this workshop.

2) Go to **File > Open Folder** in **VSCode** and open the workshop folder.

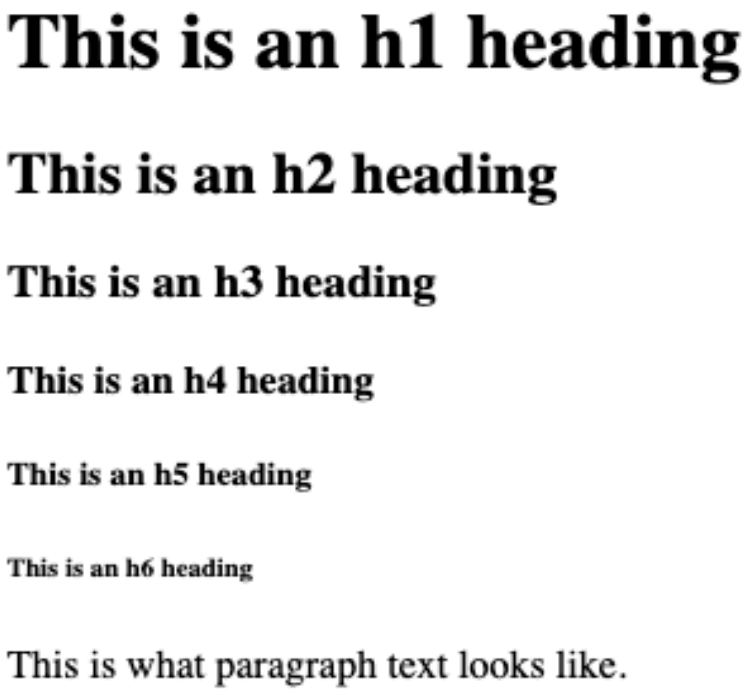
3) Install the **Live Server** extension for VSCode.

4) View **page source** in your browser.

Common HTML Elements

Headings

```html
<!DOCTYPE html>
<html lang="en">
    <head>
    </head>
    <body>
        <h1>This is an h1 heading</h1>
        <h2>This is an h2 heading</h2>
        <h3>This is an h3 heading</h3>
        <h4>This is an h4 heading</h4>
        <h5>This is an h5 heading</h5>
        <h6>This is an h6 heading</h6>
        <p>This is what paragraph text looks like.</p>
    </body>
</html>
```



Lists

Unordered List

```
<body>
    <ul>
        <li>Dogs</li>
        <li>Cats</li>
        <li>Hamsters</li>
    </ul>
</body>
```

- Dogs
- Cats
- Hamsters

Ordered List

```
<body>
    <ol>
        <li>Dogs</li>
        <li>Cats</li>
        <li>Hamsters</li>
    </ol>
</body>
```

1. Dogs
2. Cats
3. Hamsters

Links

Links are created using the <a> (anchor) tag, followed by href="" and the source. The link text is displayed in between the opening and closing tags.

Links can be **relative**, meaning they are pointing to a local file in your project…

```
<body>
    <a href="about.html">About</a>
</body>
```

…or they can be **absolute**, meaning they link to a source's full path (like a URL).

```
<body>
    <a href="https://www.google.com">Link to Google.</a>
</body>
```

Images

Images are embedded with the <img> tag, with two attributes: src (the path to the image) and alt (alt text for accessibility). The <img> tag is self-closing (it doesn't need a closing tag).

Linking to a local image:

```
<img alt="This is a dog" src="dog.jpeg" />
```

Linking to an external image via URL:

```
<img alt="This is a dog" src="https://www.thesprucepets.com/thmb/3-kxAtZmAchP9y
7PVFH2h1dKxqY=/941x0/filters:no_upscale():max_bytes(150000):strip_icc():format(
webp)/adorable-white-pomeranian-puppy-spitz-921029690-5c8be25d46e0fb000172effe.jpg" />
```

Line Breaks

Line breaks can be inserted with the
 tag.

Before line break:

```
<a href="https://www.google.com">Google</a>
<a href="about.html">About</a>
```

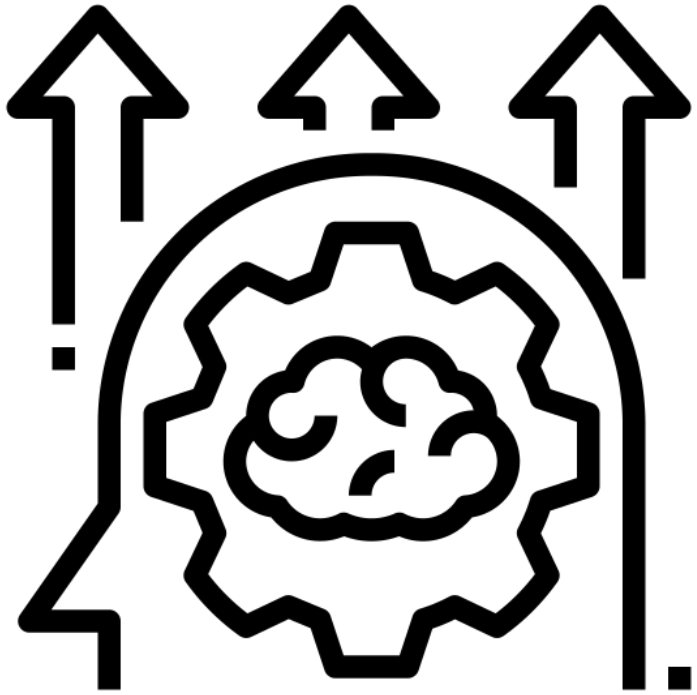
Google About

After line break:

```
<a href="https://www.google.com">Google</a>
<br>
<a href="about.html">About</a>
```

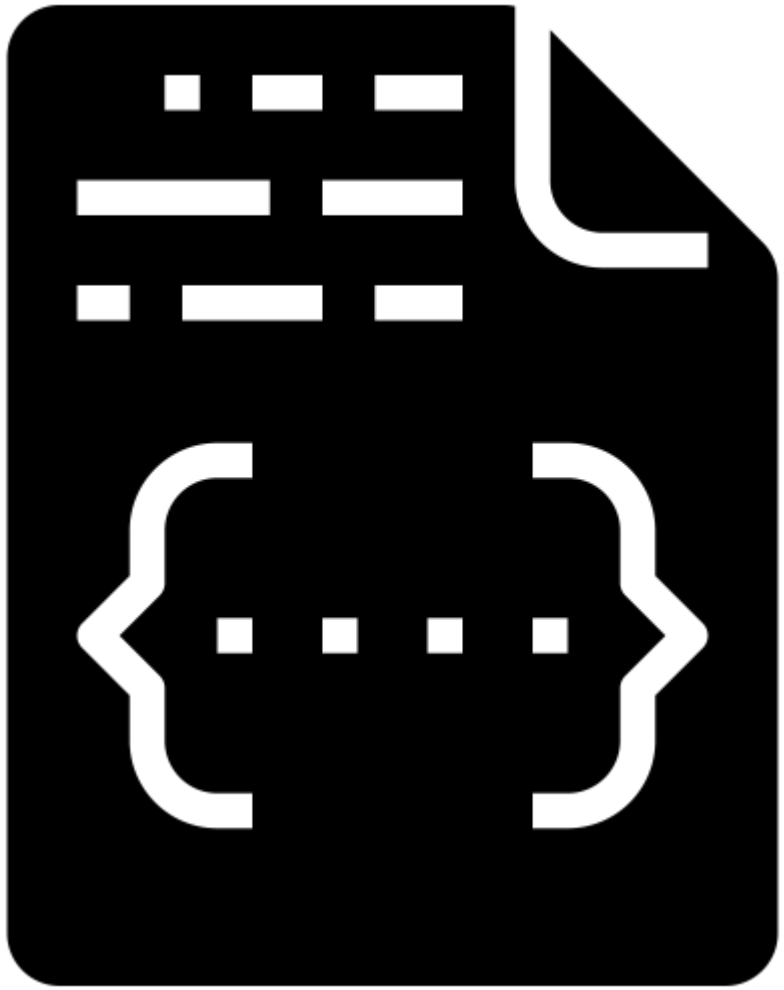
Google
About

Practice Time!



1. Change the <title></title> of your <head> section to “My Personal Website”
2. Change the <h1></h1> heading to your <body> section to “About me”
3. Insert an image of yourself or of a stock person , using relative or absolute links to your <body> section
4. Change the paragraph to a sentence or two about yourself (or your bio)
5. Add an <h2></h2> heading to your <body> section, “Publications” OR “Articles I like”
6. Add a link to a publication of yours (or an article you like)
7. Challenge: add an unordered list with various publications (or articles you like) with hyperlinks

CSS



Styling With CSS

Examples of what CSS can help you determine include:

- What background color you want to use for the page or a paragraph.
- What font or font size you want for your headings or your normal text.
- How large you want the images, and whether you want them aligned center, left, or right.
- Where elements appear on the page.
- Whether elements are visible to a user or not.
- Animations and hover effects.

CSS Integration

CSS can be integrated into your HTML in several ways (inline, internal, and external), but we will focus on **external** integration.

External styling creates a *completely separate document* (a stylesheet) for your CSS that will be linked to your HTML in the head section of your HTML document using the code below:

```
<head>
    <link rel="stylesheet" href="style.css">
    <title>This is the page title.</title>
</head>
```

CSS Rulesets

CSS is based on selectors and declarations, which together form rule sets (or just “rules”). Rule sets comprise an external styling file with a .css extension.

Here are the contents of a typical .css file:

```css
h1 {
    color: orange;
    font-style: italic;
}
p {
    font-family: sans-serif;
    font-style: normal;
}
#navbar {
    background-color: yellow;
    padding: 80px;
}
.intro {
    font-family: arial;
    background-color: grey;
    color: dark-grey;
}
```

Styling by Element

Styling by element allows you to change all elements of a given type on a page.

```
h1 {
    color: blue;
    font-size: 30px;
    font-family: 'Roboto', sans-serif;
    font-weight: bold;
    text-align: center;
}
```

Here's an example large heading.

This is an example paragraph of text.

Styling by ID

Styling by ID allows you to target a **single** element on a page. IDs are unique and should not be applied to multiple elements.

Let's say we wanted to style one single header element, without applying the style to all of them:

CSS:

```css
#headerExample {
    color: red;
    font-size: 50px;
}
```

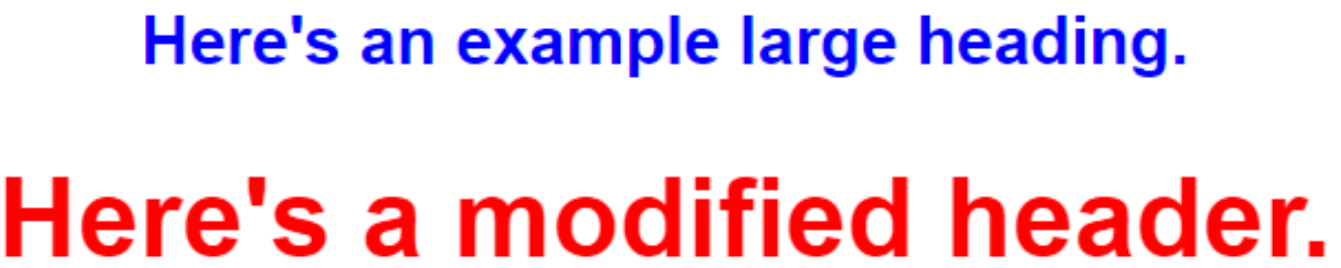
HTML:

```html
<h1>Here's an example large heading.</h1>
<h1 id="headerExample">Here's a modified header.</h1>
```

Styling by ID: The Result

```css
h1 {
    color: blue;
    font-size: 30px;
    font-family: 'Roboto', sans-serif;
    font-weight: bold;
    text-align: center;
}

#headerExample {
    color: red;
    font-size: 50px;
}
```



In short, although all <h1> tags receive styling, we can override this styling with IDs!

Styling by Class

Styling by class allows you to change all elements of a given class on a page. This effectively allows you to modify multiple elements on a page of a variety of types. Let's make a few green elements using a class:

CSS:

```css
.makeGreen {
    color: green;
    font-size: 20px;
    font-family: 'Roboto', sans-serif;
    font-weight: bold;
    text-align: center;
}
```

HTML:

```html
<p class ="makeGreen">This is an example paragraph of text.</p>
<h3 class ="makeGreen">A green header</h3>
```

Styling by Class: The Result

Here's an example large heading.

Here's a modified header.

This is an example paragraph of text.

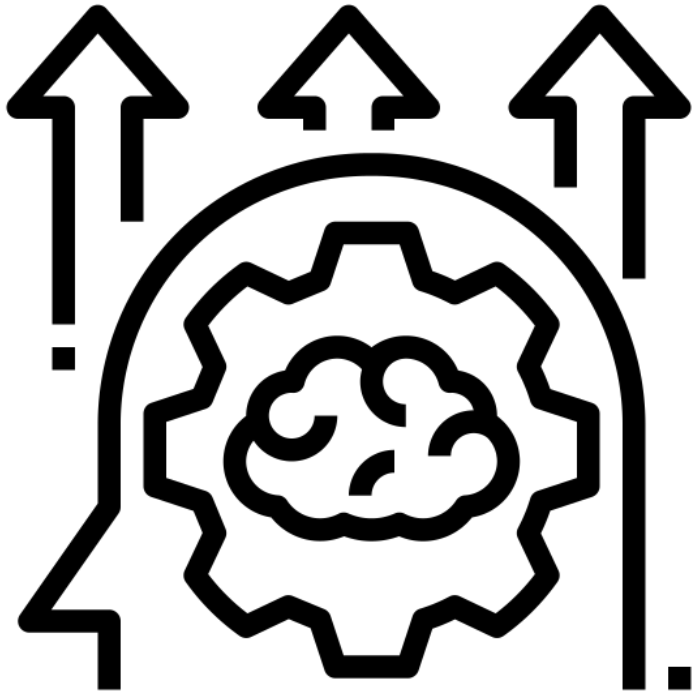
A green header

Filtering/Overriding

Keep in mind, the most specific rule in CSS always takes precedence, regardless of where it appears in the stylesheet. In order of most to least specific:

ID selectors > Class selectors > Element/Type selectors

Practice Time!



Getting Started

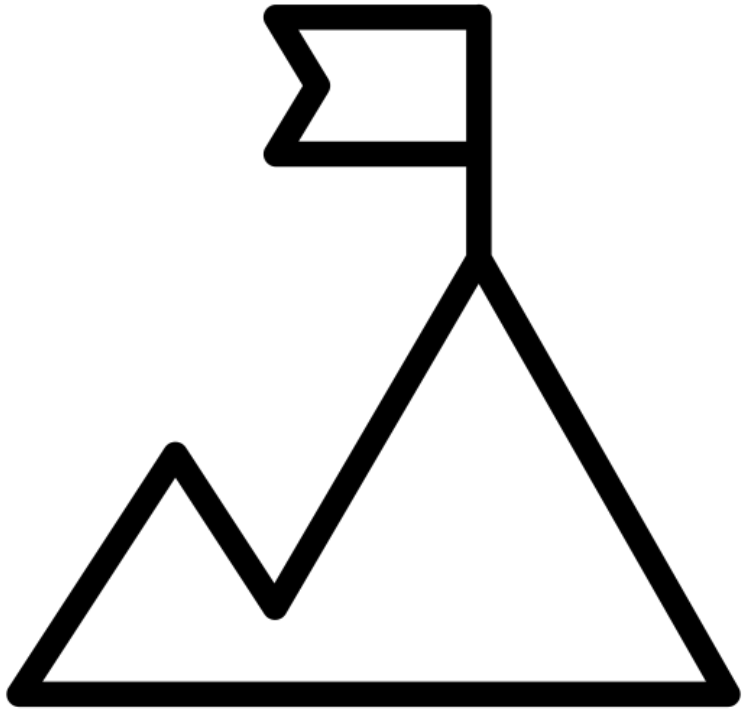
1. Create a new file in VSCode called **style.css** (saved in your workshop folder).
2. Link this stylesheet to all HTML documents that you want this styling to apply to. You can do so with the <link> tag like you saw before:

```
<head>
    <link rel="stylesheet" href="style.css">
    <title>This is the page title.</title>
</head>
```

Style Your Page

1. Use text-align to center your h1, and use font-family to change its font to “Tahoma”
2. Use color to make the color of your h2 text red
3. Change the entire background-color of your page using an id
4. Give your image a class and center it
5. Use a:hover to change the color of your links when the mouse hovers over them

Final Challenge!



Use the HTML tag

The tag acts like a **container** for other elements. Pairing with classes or ids is a particularly powerful technique.

Let's say we wanted to change the background color of each of the elements on our page, to set them off from the rest of our background. We also want to center align all of the text elements that we might write.

Instead of rewriting over and over the same changes for each element individually, let's use and an id to style them all at once.

In HTML

Inside the <body>, we can “contain” all of our elements inside a single with an id:

```
<div id="allElements">
    <h1>Here's an example large heading.</h1>
    <h1 id="headerExample">Here's a modified header.</h1>
    <p class ="makeGreen">This is an example paragraph of text.</p>
    <h3 class ="makeGreen">A green header</h3>
    <a href = "https://www.google.com">This is a link to Google.</a>
</div>
```

In CSS

Next, we simply need to create a new class that applies these changes to our id:

```
#allElements {
    background-color: peachpuff;
    text-align: center;
}
```

Now we have a background color for just our elements. Also, now that we have specified that all text elements should be centered, we can go delete the redundant centering bits in our other rulesets!

The Result


Here's an example large heading.
Here's a modified header.
This is an example paragraph of text.
A green header
This is a link to Google.

What Now?

With your newfound knowledge of HTML and CSS, you might now…

- Design a new website!
- Make edits to a pre-existing website (e.g. on Wordpress or SquareSpace)
- Add interactivity and functionality to your website by incorporating programming languages such as Javascript, SQL, and Python
- Learn how to deploy your creation to the wider web!

Join our next workshop!

This workshop has been the first in a series of workshops devoted to web development.

Please consider joining our next workshop:
Create Elegant Webpages with Bootstrap
Friday, Nov. 4th @ 1:00pm

Thank you for coming!

PLEASE, take a moment to fill out a short evaluation:

cuny.is/GCDI-webevals