CS 571 - Data Visualization & Exploration

HTML and CSS

Instructor: Hamza Elhamdadi

UMassAmherst

Upcoming Dates

Feb 7: ~~Group~~ Activity Due (Individually)

Feb 7: Homework 0 Due

Feb 10: Quiz 1 Due (will be posted at 3pm today)

Feb 13: Announce Your Project

Feb 14: Homework 1 Due

Other Administrative Details

Office Hours will begin next week
(see the syllabus for times)

Lectures will be recorded via Echo360
(the first two are already up!)

Make sure at least one member from your project team attends class on Feb 14

Visual Studio Code & Live Server Tutorial

Hypertext Markup Language

Hypertext Markup Language

Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML)

The name used to describe everything HTML could do:

Hypertext Markup Language (HTML)

The name used to describe everything HTML could do:

- It allows you to define links to other websites (hypertext)

Hypertext Markup Language (HTML)

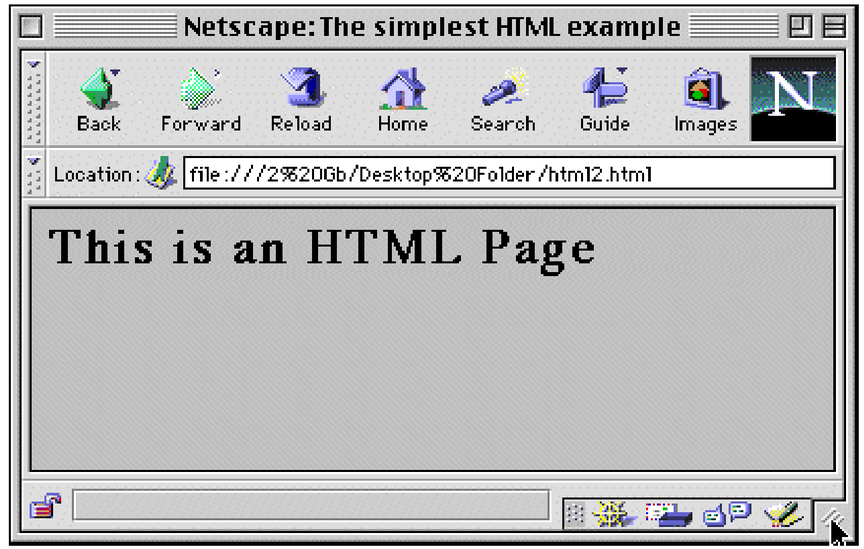
The name used to describe everything HTML could do:

- It allows you to define links to other websites (hypertext)
- It is a markup language
  - i.e., it uses tags to define the structure of the website

Hypertext Markup Language (HTML)

The name used to describe everything HTML could do:

- It allows you to define links to other websites (hypertext)
- It is a markup language
  - i.e., it uses tags to define the structure of the website



Source: https://users.cs.cf.ac.uk/dave/PERL/node259.html

HTML5

Now, HTML5 can do lots more:

HTML5

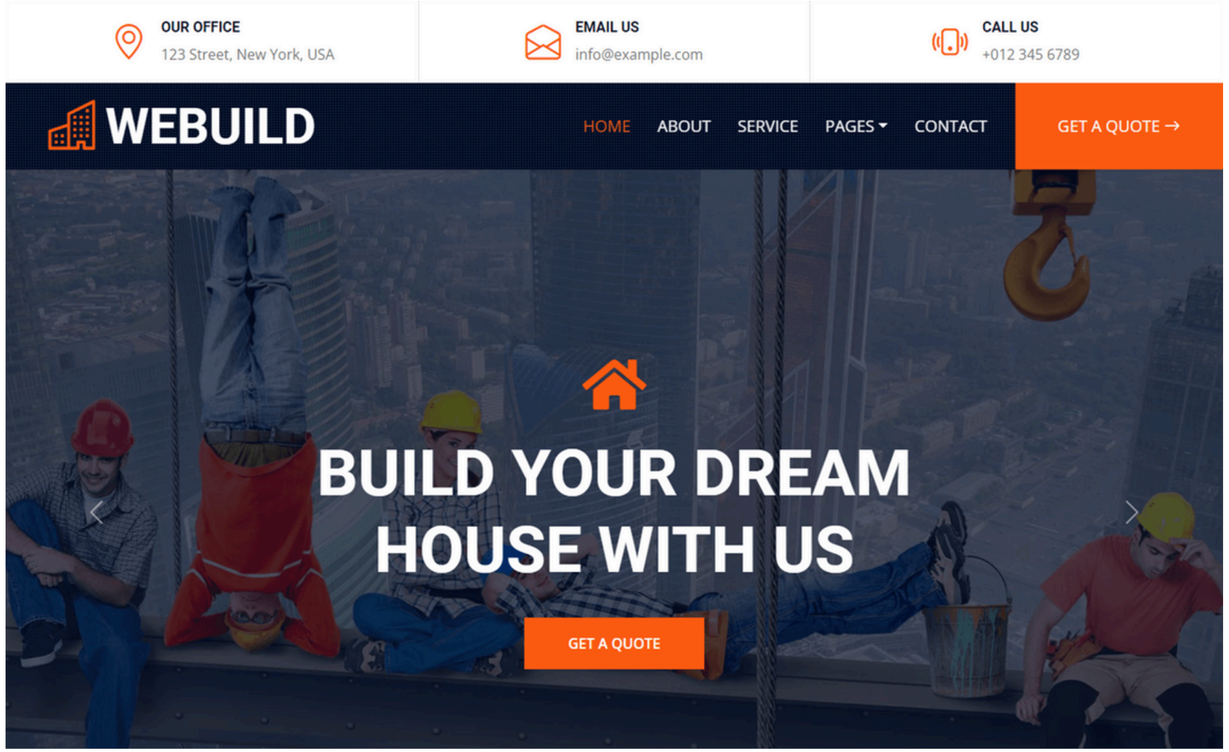
Now, HTML5 can do lots more:

- video, audio, graphics, etc...

HTML5

Now, HTML5 can do lots more:

- video, audio, graphics, etc...



Source: https://users.cs.cf.ac.uk/dave/PERL/node259.html

Basic HTML elements

An HTML element is surrounded by a pair of tags like:

```
<tag></tag>
```

Basic HTML elements

An HTML element is surrounded by a pair of tags like:

```
<tag></tag>
```

For example:

```
1 <strong>This is an HTML element.</strong>
```

Closing elements

Most elements require a closing tag like to surround the relevant text. Closing tags start with a /.

Anything outside of the tags is not part of the element:

```
<strong>This is bold (strong).</strong> This is not.
```

Self-Closing Elements

Some HTML elements (like images) do not need a closing tag.

These tags can self-close using a trailing /.

```
1 <img/>
```

In HTML5, you are allowed to use tags without closing elements. So, this is also valid:

```
1 <img>
```

Nesting HTML elements

HTML elements can be nested within other HTML elements.

```
<strong>This is strong and <u>this is underlined and strong.</u>
Just strong again.</strong>
```

Attributes

In addition to the element name, opening tags can contain extra information about the element. These pieces of information are called attributes.

```
1 <a href="https://www.cics.umass.edu"> A link to UMass CICS.</a>
```

We are using an anchor element <a>, which creates a link.
We specify the “HTML reference” attribute href to define the destination of the link.

Attributes

```
1 <strong title="More information on hover!">Use the title attribute
to reveal more information when hovering over the element.
This is called a tool-tip.</strong>
```

Really Important Attributes

The most important attributes to know are **id**, **class**, and **style**.

The **id** attribute gives an HTML element a unique identifer, which we can reference in Javascript.

```
1 <strong id="unique"></strong>
```

Think of it as making the element accessible via a global variable.

We will cover the **class** and **style** attributes in the section on CSS.

HTML Metadata and Basics

HTML documents require **metadata**, contained within specific tags that are not visually represented when loading the website.

- <html> creates the entire HTML container
- <head> creates the header
  - where we include the title info and links to CSS and JS
- <meta> is used to provide meta-information
  - like character encoding or other browser instructions
- <script> links to a script (we will use this often for JS files)
- <style> embeds style information
- <link> allows us to reference external CSS files
- <body> creates the container for the content of the website

Starter HTML Document

Just copy and paste this for now (change the orange parts):

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
</head>
<body>
    Hello World!
</body>
</html>
```

More Useful Tags

```
<body>
    <p>This is a paragraph!</p>
    <p>We can use the <br> tag to break a line</p>

    <img src="./cat.png" alt="picture of cat"/>

    <h1>Top-level header</h1>
    <h2>Second-level header</h2>
    <!-- This is a comment. It won't display. -->
</body>
```

Lists and Tables

Lists and Tables are useful to add structure

Unordered Lists

Use unordered lists to create bullet points

```
<ul>
    <li>Elements in</li>
    <li>an unordered</li>
    <li>(bulleted) list.</li>
    <li>Note: In HTML5, we don't need closing
    <li>tags to be valid
</ul>
```

Ordered Lists

Use ordered lists to number your list items

```
<ol>
    <li>Elements in</li>
    <li>an ordered</li>
    <li>(numbered) list.</li>
</ol>
```

Tables

Tables structure your info in rows and columns

```
<table border="1">
    <tr>
        <th>Column 1 header</th>
        <th>Column 2 header</th>
    </tr>
    <tr>
        <td>First Cell in Column 1</td>
        <td>First Cell in Column 2</td>
    </tr>
</table>
```

Structure / Layout

We often structure an HTML page with several “panels”
(one for the page header, one for the navigation bar, etc.)

To achieve this we can use the and tags.

There are also newer tags in that might be useful for sectioning your HTML (e.g., <article>, <section>, <footer>, and <nav>)

These are all block-level elements, meaning they break the line if not otherwise specified in CSS. Except for , which is an inline element that does not break the line. Hence, a is useful for styling a few words within a section of flowing text.

Structure / Layout

```
<nav>Navigation links would go here.</nav>
<main>
    <h1>Structuring HTML</h1>
    <section>Content for a section.</section>
    <div>div and other block-level tags</div>
    <div>add structure and add line breaks</div>
    <br>
    <span>span is like an inline version</span>
    <span>of div</span>
</main>
```

Note the lack of formatting when you run this.

Forms

```
<form>
 <label for="textfield">Enter text:</label>
 <input type="text" id="textfield"/>
 <br><br>
 <label for="longtext">Enter long text:</label>
 <textarea id="longtext">Default text</textarea>
 <br><br>
 <label for="slider">Choose a number:</label>
 <input type="range" id="slider" name="slider"
        min="0" max="10" >
 <input type="submit">
</form>
```

The Document Object Model (DOM)

You might have noticed that an HTML document looks a lot like a tree.

The root of the tree is the <html> tag, and each node may have children that may contain nodes themselves.

The DOM is a programming interface for HTML documents

We will use “DOM” to mean the tree that is created by web browsers to represent the HTML document, as well as the API browsers provide to access this tree

Inspecting the DOM of a website

Cascading Style Sheets

Cascading Style Sheets

Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS)

Plain HTML doesn’t tell the browser much about how the page should look

We can use CSS to declare rules about how specific elements will be rendered by the browser.

We can define these rules using <style> tags in the HTML header

CSS Element Selectors

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Simple CSS</title>
   <style>
       strong {
          background-color: steelblue;
          font-size: 150%;
       }
   </style>
</head>
<body>
   <p>Here is some <strong>strong text.</strong></p>
</body>
</html>
```

CSS Element Selectors

```
6     <style>
7         strong {
8             background-color: steelblue;
9             font-size: 150%;
10        }
11    </style>
```

We select all elements using the element selector **strong**

Each line inside the curly braces is a **declaration**

Here, we are saying, for every DOM element with the strong tag:

- make its background color steelblue
- and make its font size 150% of the default size

CSS Element Selectors

```
6     <style>
7         strong {
8             background-color: steelblue;
9         }
10
11        strong {
12            font-size: 150%;
13        }
14    </style>
```

CSS rules are applied in the order they are defined

If multiple rules exist for the same selector, they will be all be applied. So the code above is functionally the same as the previous slide

CSS Class Selectors

CSS selectors are capable of more than just selecting the names of HTML elements

We can create user-defined “classes” of styles

Classes allow us to declare that certain elements have a specific type that is then formatted consistently

On the next page, we’ll see an example.
Note that CSS class selectors are identified using a leading period **.** (for example, **.important**)

CSS Class Selectors

```
<head>
    <meta charset="UTF-8">
    <title>Simple CSS</title>
    <style>
        .important {
            font-weight: bold;
        }
        .footnote {
            font-size: 75%;
        }
    </style>
</head>
<body>
    <div class="important">Some text</div>
    <div class="footnote">Some text</div>
    <div class="important footnote">An important footnote!</div>
</body>
```

CSS ID Selectors

ID selectors are similar to class selectors, but IDs may only be applied to a single element in the DOM

Note: HTML doesn't enforce this, but you definitely shouldn't use the same ID more than once

You can use ID selectors to apply CSS styles as well.

On the next page, we'll see an example.
Note that CSS ID selectors are identified using a leading hashtag **#** (for example, **#first-line**)

CSS ID Selectors

```
<head>
    <meta charset="UTF-8">
    <title>Simple CSS</title>
    <style>
        #first-line {
            font-weight: bold;
            color: steelblue;
        }
        #second-line {
            text-shadow: 2px 2px #8b8b8b;
        }
    </style>
</head>
<body>
    <div id="first-line">Some text</div>
    <div id="second-line">Some other text</div>
</body>
```

CSS Relationship Selectors

```
<head>
   <meta charset="UTF-8">
   <title>Simple CSS</title>
   <style>
       p > b { color: blue; }

       div b { color: green; }
   </style>
</head>
<body>
   <p>b tags directly in a paragraph will be <b>blue.</b></p>
   <p>This is not <span><b>blue because of the span.</b></span></div>
   <div>
      b tags that are descendants of divs will be <span><b>green</b>
      </span> even if they are not direct children of the div
  </div>
</body>
```

Multiple Rules in CSS

When multiple rules apply to the same element in CSS, then different declarations may conflict with each other.

If multiple declarations conflict, the more specific declaration wins.

The rules for specificity are really complicated and not very intuitive for beginners.

A good rule of thumb when your CSS isn't working is to create a new class for the element you are trying to style, and add the declarations to this class.

Note: Stick with simple declarations when possible,
and use the Developer Tools to debug your CSS

More CSS rules

There are many more rules, including pseudo-class selectors and grid layouts that we will not cover in class.

Refer to the supplementary material on the course website for more details

External CSS

You can provide CSS stylesheets as an external file in your HTML

Let's say you have a file called **style.css** in the same directory as your HTML file and style.css contains your CSS styles.

You can add the styles from style.css to your HTML by including the following element in the <head> section:

```
1 <link rel="stylesheet" href="style.css" />
```

Note: Do not include the <style> tag in your external CSS files. Only write the CSS selectors and declarations.

Inline CSS

It is also technically valid to write styles directly as an attribute for a single element:

```
<p style="color: blue;">Blue paragraph</p>
```

This is not recommended when writing manual CSS or HTML, because this mixes content and presentation, and makes it hard to reuse these styles later.

However, when we start adding styles with Javascript and D3, you will often see this kind of styling in the Developer Tools. In this case, the reusability is achieved by the Javascript.

FIN