Week 2: HTML Part 1

ATLS 2200 (Web)
Spring 2022

roadmap

TODAY…

1. Week 01 Debrief
2. A History of HTML and the WWW
3. The HTML Language
4. HTML Document Structure
5. Directory Structures
6. Wrap-up + Next Steps

WHILE YOU’RE GETTING SETTLED

Make sure to check in via Canvas (“Week 2 Lecture Check-in”).

week 01 debrief

How was your first week in Web?

daily note, notes

WHAT’RE YOU EXCITED ABOUT?

“Creating actual websites. In my other classes involving coding, there wasn't much that excited me, however being to create something you can actually interact with and use is something I'm looking forward to.”

WHAT QUESTIONS DO YOU HAVE?

“Is this class manageable if you have no working knowledge of these languages prior to the start of the semester?”

“Is it ok if we don't have experience in these programs and starting out new?”

“I'm wondering if we'll cover Bootstrap or solely use HTML/CSS/JS? Same with jQuery?”

daily note, notes

WHAT'RE YOU NERVOUS ABOUT?

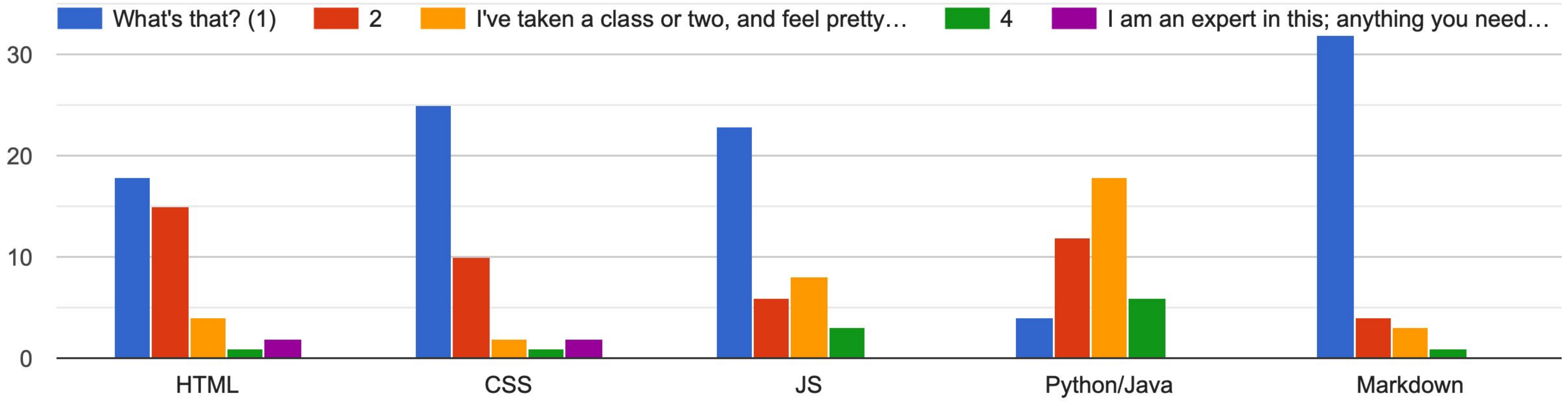
"Becoming more comfortable with coding. **It always makes me really anxious when I can't seem to get it right, as I'm more of a design/art oriented person.** I think that in learning something that will lead to me setting up my own web page will be really rewarding, and I can include design aspects into it, which I haven't really been able to do in previous coding classes."

"Working with three new languages. I have experience with Python and R, but had a lot of trouble. Hopefully I can keep up with all three."

"I took a coding class last summer and it moved so quickly that I just lost out on understanding, and I didn't even know where I didn't understand. **So I don't have a great experience with coding,** however, I am looking to change that perspective."

"I'm worried that so many students are already comfortable with these languages and I have no knowledge of any of them. I am worried that I will quickly fall behind and get overwhelmed quickly."

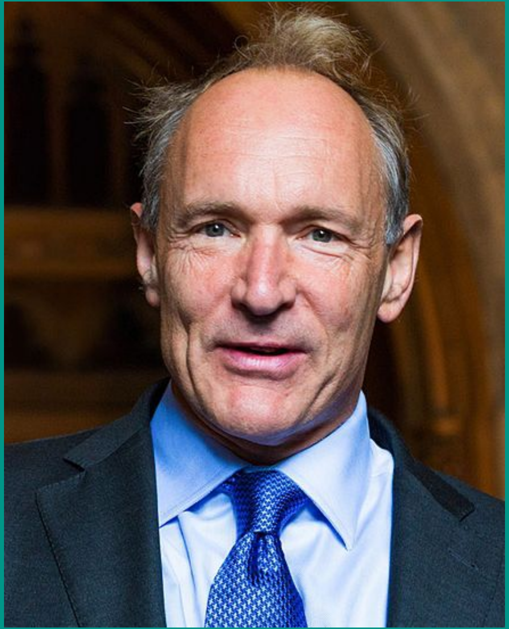
What type of experience do you have with "programming"?



a history of the html and www

To know where we're going, we should know where we've been.

a history lesson



THE INVENTION OF HTML AND THE WORLD WIDE WEB

Tim Berners-Lee (pictured) is the inventor of HTML and of the World Wide Web (WWW).

Berners-Lee, a physicist, worked at CERN (yes, that CERN, particle accelerator and all) as a software engineer.

While at CERN, he wrote memos that evolved into HTML in late **1990**. The WWW was invented in **1989;** while Berners-Lee wrote the first browser in **1991.**

The World Wide Web is a information system consisting of pages with **uniform resource locators (URLs)** that are interlinked by **hyperlinks**. Berners-Lee imagined the WWW being particularly useful for an encyclopedia-type application (hello, Wikipedia!).

HTML has since gone through several revisions – the current being **HTML5**.

the html language

Let’s start building something.

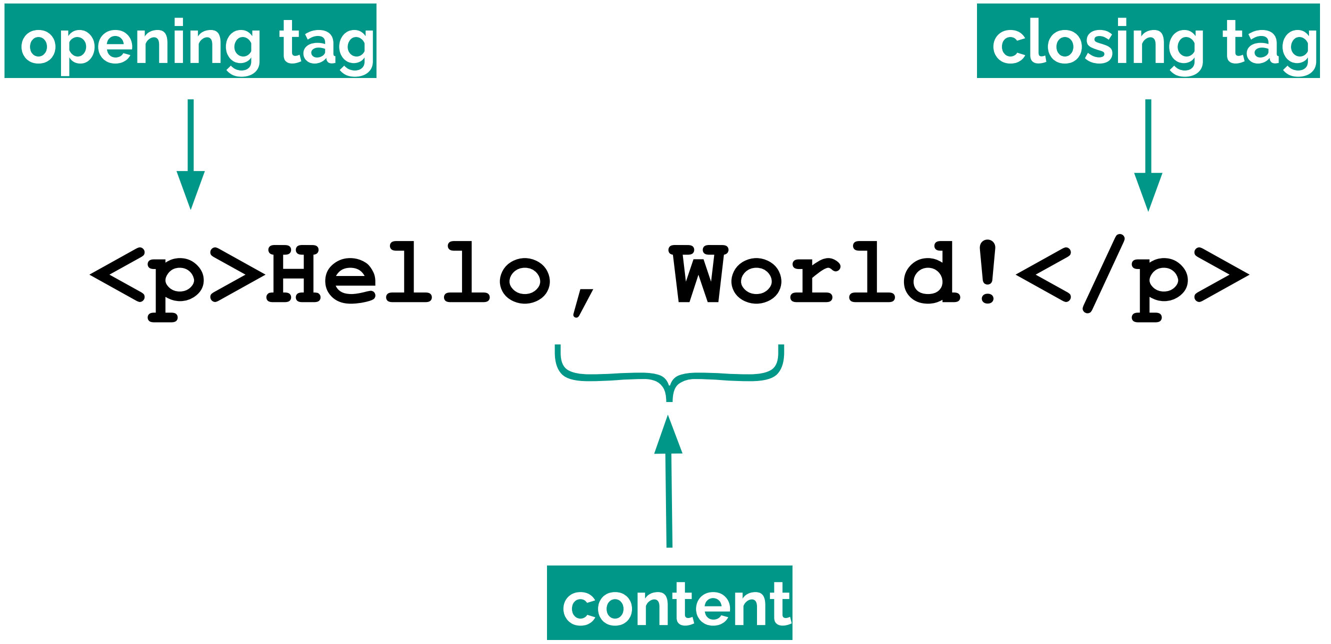
hypertext markup language

WHAT’S HTML?

Hypertext Markup Language (HTML) tells web browsers how to structure a web page when you visit it.

Contrary to how it is often presented, HTML is not a programming language. As the name suggests, it is a **markup language.**

HTML consists of **elements.** Elements are made up of **content** and **tags**.



```
<p>Hello, World!</p>
```

html

CONTENT AND TAGS

Content is simply that – the content of the element. In the previous slide, our content was just two words and some punctuation.

Tags come in two varieties: **opening** and **closing tags.** As their names suggest, **opening tags** open an element; **closing tags** close an element.

We'll learn more about the differences between opening and closing tags later. For now, just remember that closing tags have a **forward slash (/)** at the beginning of the tag.

NOTE: forgetting to close an element with a closing tag or forgetting the forward slash in the closing tag is a common "beginner" mistake. So always check carefully!

html

NESTING

Like loops in programming, we can **nest** elements within each other.

```
<p>Hello, <em>World!</em></p>
```

Be sure to close elements in the same order as you open them (just like programming – close the innermost loop first).

As we’ll see in a bit, nesting is not a trivial matter. Turns out, the entire HTML document is a giant nested list of elements!

TYPES OF ELEMENTS

There are two types of elements that are important to know and understand: **block-level elements** and **inline elements.**

Block-level elements form a visible block on the page. They start on a new line, and are generally the structural elements of the page.

The <p> tag we saw earlier is an example of a block-level element. Other examples are headings, lists, and footers.

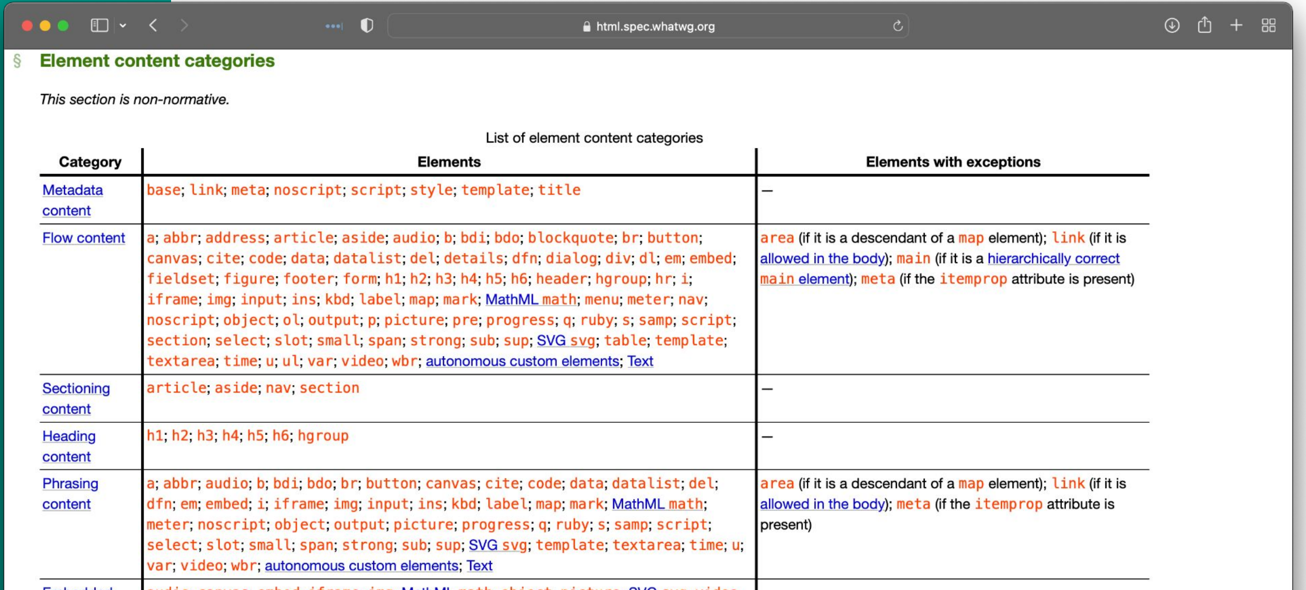
Inline elements are contained within block-level elements, don't create new lines, and are typically used to modify specific bits of content.

The <em> tag is an example of an inline element. It renders the content it surrounds as italicized.

html

TYPES OF ELEMENTS

NOTE: The TAs are probably looking at me funny because in the newest form of HTML – HTML5 – there are many more categories of elements that are far more specific. For the moment, let's stick with the deprecated block vs. inline.



html.spec.whatwg.org

§ Element content categories

This section is non-normative.

List of element content categories

Category	Elements	Elements with exceptions
Metadata content	base; link; meta; noscript; script; style; template; title	—
Flow content	a; abbr; address; article; aside; audio; b; bdi; bdo; blockquote; br; button; canvas; cite; code; data; datalist; del; details; dfn; dialog; div; dl; em; embed; fieldset; figure; footer; form; h1; h2; h3; h4; h5; h6; header; hgroup; hr; i; iframe; img; input; ins; kbd; label; map; mark; MathML math; menu; meter; nav; noscript; object; ol; output; p; picture; pre; progress; q; ruby; s; samp; script; section; select; slot; small; span; strong; sub; sup; SVG svg; table; template; textarea; time; u; ul; var; video; wbr; autonomous custom elements; Text	area (if it is a descendant of a map element); link (if it is allowed in the body); main (if it is a hierarchically correct main element); meta (if the itemprop attribute is present)
Sectioning content	article; aside; nav; section	—
Heading content	h1; h2; h3; h4; h5; h6; hgroup	—
Phrasing content	a; abbr; audio; b; bdi; bdo; br; button; canvas; cite; code; data; datalist; del; dfn; em; embed; i; iframe; img; input; ins; kbd; label; map; mark; MathML math; meter; noscript; object; output; picture; progress; q; ruby; s; samp; script; select; slot; small; span; strong; sub; sup; SVG svg; template; textarea; time; u; var; video; wbr; autonomous custom elements; Text	area (if it is a descendant of a map element); link (if it is allowed in the body); meta (if the itemprop attribute is present)

html

TAGS

There are many elements of both block and inline type. Here are a few relevant ones:

- Paragraph – <p> </p>
- Headings – HTML defines **six (6)** levels of headings
  - H1 – <h1> </h1>
  - H2 – <h1> </h2>
  - H3 – <h3> </h3>
  - … and so on (hopefully you get the idea).
- Bolding – <strong> </strong>
- Italics – <em> </em>
- Underline – <u> </u>

html document structure

Turning to the real world...

html document structure

HTML DOCUMENT STRUCTURE

The individual elements of HTML are great to understand by themselves, but it takes a whole document to see how they work.

Let’s look at an example HTML file. Notice any nesting?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

html document structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

<!DOCTYPE> is a historical artifact that has to be included for everything else to work correctly.

<!DOCTYPE html> is the shortest doctype that is valid. Be sure to include it on the first line of every HTML document.

html document structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

<html></html> is… the **html element** (surprising, I know).

The **html element** wraps all of the content on this page. You might hear it called the **root element**.

The html element is always the highest or outermost element in our nested HTML document.

html document structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

<head> </head> is the **head element**. The head element contains all the meta information about the page – all the stuff you don't want the viewer to see, but might impact the page in some way.

<meta charset=”utf-8”> specifies the character set for the document. UTF-8 contains most of the characters for written human language.

html document structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

<title> </title> is the **title element**. The **title element** sets the title of the page – what is displayed on the browser tab when you load the page.

Sometimes, the title element describes the page when you bookmark it (browser dependent).

html document structure

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>My test page</title>
  </head>
  <body>
    <p>This is my page</p>
  </body>
</html>
```

<body> </body> is the **body element**. The **body element** contains all of the content that displays on the page.

directory structure and organization

Do future you a favor.

directory structure

DIRECTORIES, FILES, AND OTHER RELEVANT TERMS

Files are units of information stored on a computer. Files take the form many different formats, denoted by their file extension.

File extensions define the type of file. You're likely familiar with many different types of files and file extensions, but in this class we'll become familiar with **.html**, **.css**, and **.js** file types.

A **directory** is a collection of files, aka a **folder**. A GitHub repo is an example of a directory.

A directory can have **subdirectories**, or directories within directories (folders within folders). A directory containing other directories is called a **parent directory**.

directory structure

DIRECTORY AND FILE PATHS

Each directory and file on your computer has a **path**, the unique address of that item on your computer.

You could think of the path of the file as being similar to the URL of a specific webpage.

When referencing other files in HTML, we’ll provide links that give the path of that other file. There are two types of links:

Absolute links are the full address of the file. We’ll use absolute links often for external links from our websites (i.e., to other websites).

Relative links are paths that shorter, and depend on path of the file linking to it.

directory structure

RELATIVE LINKS

There are three broad categories of relative links to be concerned about.

File Location	**Relative Link**
The linked page is in the same directory	Equal levels, which means we just list the file name we want to link.
The linked page is in a parent directory	We need to move up a level, so we list "../" the file name.
The linked page is in a subdirectory	We need to move down a level, so list the subdirectory name.

roadmap

WHAT'S NEXT?

In recitation this week – you'll be creating a new index.html page on your website… this is the real start of your very own website!

Quiz 1 opens at 10:45 AM; due at the start of your recitation. **Assignment 1** opens at 10:45 AM; start in recitation; due Sunday by 11:59 PM.

DAILY NOTE

We'll post the Daily Note in Slack in **#atls-2200-web-spring-2022.**

Make sure to do it now!