

2.1 Web and HTML Basics

Due Monday by 10am **Points** 25 **Submitting** a website url

Release 0: Spy on Your Browser

All day long, your computer has been whispering to other computers, right under your nose.

Every time you visit a website, your computer (the **client**) reaches out to another computer (the **server**) on the Internet and asks it for the documents, images, songs, and other files that you wanted. If you're wanting to play a song, your computer might reach out to one of Spotify's servers to request the right track. If you'd like to see what your friends are up to, your computer might contact servers owned by Facebook, Twitter, or Instagram.

The conversation goes a little something like this:

CLIENT: Hey, it's me, Ashley's computer. Would you happen to have her Facebook profile page?

FACEBOOK SERVER: Yep, here you go.

CLIENT: Great, thanks, it's downloading now. Let me go ahead and display that for her -- oops, it looks like this page has some images in it. I guess I'm going to need those, too.

FACEBOOK SERVER: Sure! A picture of Ashley's feet on the beach, four pictures of her cat, and a group shot of her friends on a camping trip, coming right up.

This computer-to-computer conversation uses a protocol called **HTTP**. **Protocol** is a fancy word, but a protocol is simply a conversation that occurs according to a known standard of communication.

You use protocols all the time. For instance, you might walk into a restaurant and say to the hostess, "Two, please." The hostess will automatically understand that you'd like a table for two people, even though what you just said hardly makes any sense. That's because you're using a common protocol for requesting a table. She might then grab two menus and say "Right this way," which you understand to mean that if you faithfully follow her, you'll be enjoying a big plate of nachos in no time.



Mmm ... protocol.

HTTP requests are simply text, formatted in a certain way, that allows computers to talk to one another in order to exchange files and other information. HTTP is a communication protocol between computers, not humans, so you're usually left out of the conversation. But you can snoop on this little tea party quite easily. Or should we say ... H-Tea-Tea-Party?

(Sorry.)

You'll learn more about HTTP in later challenges, but for now, let's just get an idea of how many HTTP requests your computer might make just to load a single page.

DO THE THING

1. Run `git pull` in the `phase-0-tracks` repository to make sure you have the latest version.
2. In `phase-0-tracks/html`, use the command line to create a `web_basics` directory.
3. In Chrome, navigate to `http://www.cnn.com`, or another news site you enjoy. (Notice that the protocol is right there in the address!)
4. In the Chrome application menu, go to View > Developer > Developer Tools to open the developer tools pane.
5. Inside the developer tools pane, click the Network tab.
6. Reload the page and watch all of the HTTP requests go whizzing by. Whoa!
7. Take a screenshot of your results and put it in your `web_basics` folder. Don't forget to use Git and GitHub workflow to save your work.

When we tried this, our browser made 350 requests on our behalf. That's a lot of HTTP requests! And yet what displays in our browser appears to be a single document. Your browser does all the work of requesting files and rendering all of this information as one page. **Rendering** refers to the process of converting text and image data to a human-viewable document.

In the next release, you'll learn more about creating a document for the browser to render.

Release 1: Make Your First Webpage

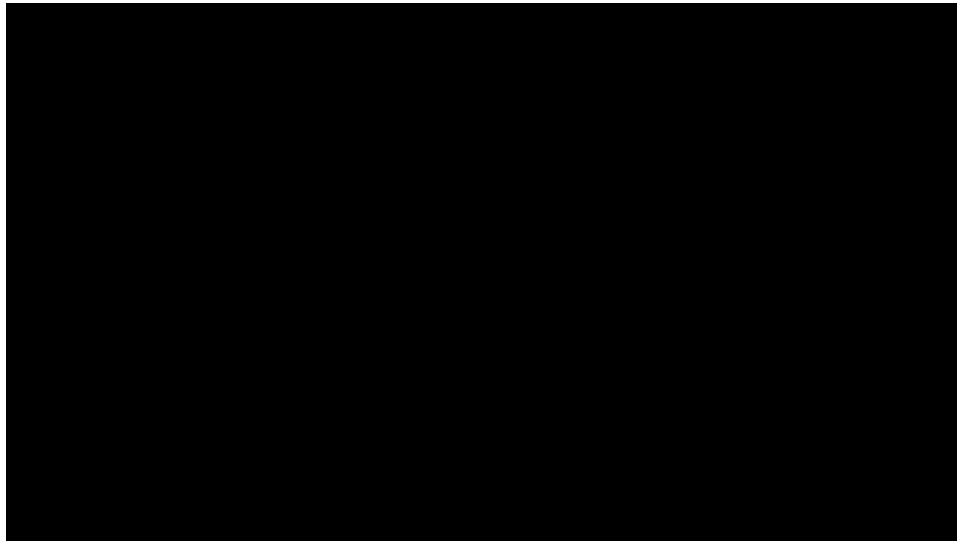
Your browser is essentially a computer-to-human translator, and one of the languages it speaks is HTML. **HTML**, or **Hypertext Markup Language** if you're feeling wordy, tells the web browser how to display the text, images, and other content of a web page.

The very page you're reading is made of HTML, in fact.

TRY THE THING

Right-click on this page in Chrome and choose "View Page Source" to see it. What you see won't make much sense to you yet, but some of it will by the end of this week.

Here's an introduction to HTML. Some of it will be relevant right away, whereas other parts are just to make you aware of the general landscape. You may need to bump up the quality setting of the video to see the code very well, and like most of our videos, it's best watched in full screen.



- Create a basic HTML page - 1:25
- Add a link - 7:38
- Add an image - 9:28
- Add a list - 10:40
- HTML5 and Semantic Tags - 13:45
- Tables - 15:40
- Meta Tags - 17:35
- Accessibility - 19:35
- ID attribute - 21:10
- View Source - 22:20

- MDN Documentation - 22:35

DO THE THING

It's time to ... drumroll ... make a webpage!

1. Add an `index.html` file to your `web_basics` folder (don't forget to use Git and GitHub workflow for this challenge). The main page inside any given folder is often called `index.html`; you'll see this convention often.
2. Open `index.html` in Sublime.
3. If you type `html` at the top of the file and then press the tab key, Sublime will provide an HTML snippet to get you started. Otherwise, you can add the following items to your page yourself, until you have the following page:

```
<!DOCTYPE html>
<html>
  <head>
    <title> Your Title Here</title>
  </head>
  <body>
    Your Content Here
  </body>
</html>
```

Note how nested tags are indented. You should follow that convention throughout your HTML. Some developers (and the Sublime snippet) don't indent the `head` and `body` tags just to save space, which is also okay.

4. Open your page in Chrome. The command `open index.html` will work if you're inside the `web_basics` folder, or you can drag and drop the file onto Chrome from your files and folders. You won't see much there yet.
5. Add an `h1` tag inside your `body` tag, and put a message there. (It's programming tradition to use "Hello world", but you do you.) Save the file.
6. Refresh the page in Chrome. You should see your message there. Magical!
7. We won't always remind you, but this would be a great time to commit your changes. How can you help yourself get into the habit of committing when you've completed a small task?

Release 2: Use the Documentation

Developers don't memorize every aspect of a given technology. It would just be too much information to keep in their heads. It's very common for developers to have to look up Ruby methods, HTML tags, and so on. So one of the most important skills you can practice as a new programmer is exploring documentation.

The [HTML documentation](https://developer.mozilla.org/en-US/docs/Web/HTML) (https://developer.mozilla.org/en-US/docs/Web/HTML) at MDN (Mozilla Developer Network) offers a list of attributes of each HTML tag, along with code examples, in the [HTML Element Reference](https://developer.mozilla.org/en-US/docs/Web/HTML/Element). (https://developer.mozilla.org/en-US/docs/Web/HTML/Element).

You won't understand everything you read when browsing through documentation -- no one does! The point is not to know all of the answers, but to feel comfortable browsing complicated documents and finding what you need.

DO THE THING




Use the documentation, and any other sources you need, to add examples of the following elements to the page you created in the previous release. Your content doesn't have to have a theme or even make a ton of sense, but it should be original content, and it should be appropriate for the tag.

If you need a theme for your content, may we take point from the video and suggest cats? They're little tiger-shaped pets. That's amazing!

- `p`
- `strong`
- `em`
- `ul` and `li`
- `ol` and `li`
- `a`

Check your code using an [HTML validator](https://validator.w3.org/#validate_by_input) (a good habit to get into), and make any needed changes. Submit the GitHub URL to your `index.html` file when finished (but make sure your screenshot from the previous release was pushed up as well).

Thanks <https://creativecommons.org/licenses/by/2.0/> to Flickr user Earls37a for the publicly licensed **photo** <https://www.flickr.com/photos/indraw/5968863751/in/photolist-a6rZeZ-pcBzud-7ZrGVx-9yRFCQ-56a8Hn-4H9cim-8J3YMw-bFezu-bcEMsZ-7XxLfv-6C6TFe-6nXmqY-4ecqZe-7XB1py-dKS7X6-gef9o-8pSZf3-5oWxPX-aT3CJi-32Qqe-5H4pyG-8iVw2-76rLQL-8d7KwU-5Ab9xB-4RzMpk-5Y6RAK-9pYYC1-c22oJ-75bENS-7CEic1-647ohP-p5kqU6-7j87w7-LCeWG-c43oJ-p4cyJZ-aowaGM-ary6WB-9qA9x-Acihv-4Qbrgu-7WNrkU-Qu2p-5W5yPT-9SpvLd-efDY7u-ze9ac-4mDDQX-aPbJo6> of nachos acquired via the magic of protocol.

HTML 1 : Introduction				
Criteria	Ratings			Pts
 create a valid HTML document template using <!DOCTYPE>, <html>, <head>, and <body> elements threshold: 3.0 pts	Exceeds Expectations 5.0 pts	Meets Expectations 3.0 pts	Does Not Meet Expectations 0.0 pts	5.0 pts
 use basic HTML container elements (p, h1, ...) threshold: 3.0 pts	Exceeds Expectations 5.0 pts	Meets Expectations 3.0 pts	Does Not Meet Expectations 0.0 pts	5.0 pts
 use nested elements threshold: 3.0 pts	Exceeds Expectations 5.0 pts	Meets Expectations 3.0 pts	Does Not Meet Expectations 0.0 pts	5.0 pts
use dev tools to track network requests	Exceeds Expectations 5.0 pts	Meets Expectations 3.0 pts	Does Not Meet Expectations 0.0 pts	5.0 pts
demonstrates good Git workflow	Exceeds Expectations 5.0 pts	Meets Expectations 3.0 pts	Does Not Meet Expectations 0.0 pts	5.0 pts
Total Points: 25.0				