HTML SYNTAX

1. Intro <https://www.youtube.com/watch?v=Lvbk6vyhglY>
2. Html structure <https://www.youtube.com/watch?v=UjCbXQ8Coic>
3. Make your first element <https://www.youtube.com/watch?v=uUknlMoWMSY>
4. Environments <https://www.youtube.com/watch?v=o6RrK6LaIZQ>
5. Text editors (cf plus bas)
6. Browsers (cf plus bas)
7. Workflow <https://www.youtube.com/watch?v=A9MZzKYGd_0>
8. Trees <https://www.youtube.com/watch?v=OVeJU18ADmw>
9. Html and trees <https://www.youtube.com/watch?v=766JMEtZCPE>
10. Spot the bug <https://www.youtube.com/watch?v=ssgUHYg5Zh8>
11. Html research <https://www.youtube.com/watch?v=WNipw86jmMc>
12. Wrap up <https://www.youtube.com/watch?v=GX9zg7Ggcdk>

HTML SYNTAX PROBLEM SET

1. Make a button (cf plus bas)
2. Chrome Extension (cf plus bas)
3. Make all the headers (cf)
4. Make a list (cf)
5. Tree to HTML (cf)
6. Constructing links (cf)
7. Create an image (cf)
8. A guide to paths (cf)
9. Quizz Figures (cf)
10. Quizz Supportig materials (cf)
11. Quizz Format a simple article (cf)

**Text editors**

There are many editors that can write plain text. In fact, you have one installed already regardless of the kind of computer you're using.

If you're on a PC, you have Notepad.

If you're on a Mac, you have TextEdit.

If you're using a Linux distribution, you have at least a command line text editor like [**Nano**](https://en.wikipedia.org/wiki/GNU_nano) or [**Vi**](https://en.wikipedia.org/wiki/Vi). Some distributions might have a [**GUI**](https://en.wikipedia.org/wiki/Graphical_user_interface) text editor installed as well.

While regular text editors work, they are slow because you have to type each and every character manually. When writing code, you'll notice pretty quickly that code structure is repetitive - you'll be writing code that looks the same over and over and over.

A regular text editor, like TextEdit or Notepad, isn't going to help you out. But text editors made for code will help you out with colorful syntax highlighting, auto-completing common blocks of code, automatic formatting and other ways you can't even imagine yet.

There is no "best" text editor. It's all a matter of personal preference. I want you to download at least two of these and try them out to see what you like.

### Sublime Text 3



**Windows, Mac, Linux**

**Cost**: $70USD (but free to try indefinitely with pop-ups)

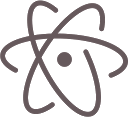
**Style**: GUI

[**Website**](http://www.sublimetext.com/3)

Sublime Text is fast, lightweight and easily customizable. With tons of keyboard shortcuts, multiple cursors and a huge ecosystem of plugins, you can create the perfect workflow.

However, updates to Sublime Text have recently started to slow, which makes some developers concerned. Many are flocking to...

### Atom



**Windows, Mac, Linux**

**Cost**: Free

**Style**: GUI

[**Website**](https://atom.io/)

Made by GitHub and a relative newcomer on the text editor scene, Atom looks and feels similar to Sublime Text. Both feature multiple cursors and they share many of the same keyboard shortcuts. Also much like Sublime Text, Atom has a rich ecosystem of packages that you can use to customize your editor to your heart's content.

Both Atom and Sublime Text are hugely popular and the majority of web developers that you'll meet will probably be using one or the other.

### Notepad++



**Windows**

**Cost**: Free

**Style**: GUI

[**Website**](https://notepad-plus-plus.org/)

A classic lightweight text editor, Notepad++ has been around since 2000.

Notepad++'s plugin ecosystem is not as large as Atom's or Sublime Text's, but many people like its clean (perhaps old school?) presentation.

### Emacs



**Windows, Mac, Linux**

**Cost**: Free

**Style**: Command-Line or GUI

[**Website**](https://www.gnu.org/software/emacs/) or [**Aquamacs**](http://aquamacs.org/) (for OS X)

Emacs is an open source text editor that's been around since the 1970s. Along with Vim, it's one of the most popular Linux text editors.

Emacs is often described as an operating system because even in a clean install there are several included applications that you wouldn't expect inside a text editor, like a news reader, several calculators, a number of games, file encryption/decryption, and a package manager for plugins written in [**Emacs Lisp**](https://en.wikipedia.org/wiki/Emacs_Lisp).

All Emacs commands exist in the same namespace so it's not uncommon to differentiate commands by having chains of keystrokes like C-x C-f (Control - x, Control - f) to open a file. Because of the incredible customizability, it's among the editors with the steepest learning curves. You will most like want to customize it: installing plugins, trying them, testing for conflicts, uninstalling the ones that have conflicts, and repeating.

### Vi/Vim



**Windows, Mac, Linux**

**Cost**: Free and open source

**Style**: Command Line or GUI

[**Website**](http://www.vim.org/index.php)

Vim, or Vi IMproved is the other text editor in the [**Unix Editor Wars**](https://en.wikipedia.org/wiki/Editor_war).

Vim runs anywhere that standard C can run and is often in the base install for most Linux and non-Windows systems including Mac OS X. It also offers a fairly robust [**tutorial**](http://www.openvim.com/) to learn how to use it. Learn it once and you can use it everywhere.

For experienced students (everyone else, it's ok if you don't understand this right now): Vim relies on modes, or scopes, when certain commands are applicable. In the command mode, the user can move around a file or execute commands. For instance, in insert mode, you can edit a file. While you are creating a HTML file (and are in HTML mode), you might be able to expand html:5 into the boilerplate for an empty HTML file.

Websites run in browsers, which means you need some browsers to run websites!

Your users could be using **any** browser to visit the websites that you build for them. There are slight differences in the way that different browsers will render your websites, which means that you'll need to test your sites in all of them.

Now, you don't need to download every browser right now, but I want to make sure that you install the two most popular cross-platform browsers: **Google Chrome** and **Mozilla Firefox**.

Other browsers, like Apple Safari and Microsoft Edge, are fantastic as well and you should continue to browse and test your sites with them. However, both are only available on a single operating system and neither have quite as robust a set of tools for developers (ie. "Developer Tools" or "DevTools") as Chrome and Firefox.

In most of the lessons on Udacity, you'll see us using Chrome. Chrome has an amazingly powerful set of Developer Tools that you'll be using to learn more about your websites than you can imagine. More on Developer Tools in later lessons.

(That's not to disparage Firefox, which also has an awesome set of Developer Tools.)

Right now, I want you to install Chrome and Firefox if you haven't already.

[**Install Chrome**](https://www.google.com/chrome/browser/desktop/)

[**Install Firefox**](https://www.mozilla.org/)

**Quiz: HTML Research**

Check out the [**MDN HTML Element Reference page**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element).

Once you're there, search for bold and emphasis. You can search text on a page using keyboard shortcuts cmd + F for Mac, orCtrl + F for Windows!

### Welcome to the HTML syntax problem set! MAKE A BUTTON

For this quiz, all I want you to do is type out the HTML to create a button element. The purpose of this quiz is to make sure that you're using the correct syntax. As you will realize when you build websites, you can write HTML incorrectly - by forgetting a closing tag, for example - and the browser will try to correct your mistake. Sometimes the browser will guess correctly and your website will look fine. But most of the time, it will be wrong and strange bugs will start appearing.

This is a good time to double check that you're creating elements correctly because you'll be writing more complicated HTML soon!

a button

Buttons may look different based on browser and operating system.

You create button elements the same way you've been creating other elements: <tag>content</tag>. In this case, the tag name isbutton and the content that comes between the tags will be the text displayed inside the button. Here's more about buttons on the[**Mozilla Developer Network (MDN)**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/button).

When you create your button, make sure you include some text content so that the button says something - it can be whatever you want!

If you want to see what your button looks like (you don't have to for this quiz), simply create an HTML file with a button element and open it in your browser!

When you're ready to create a button, click Continue to Quiz!

### Udacity Front End Feedback Chrome Extension

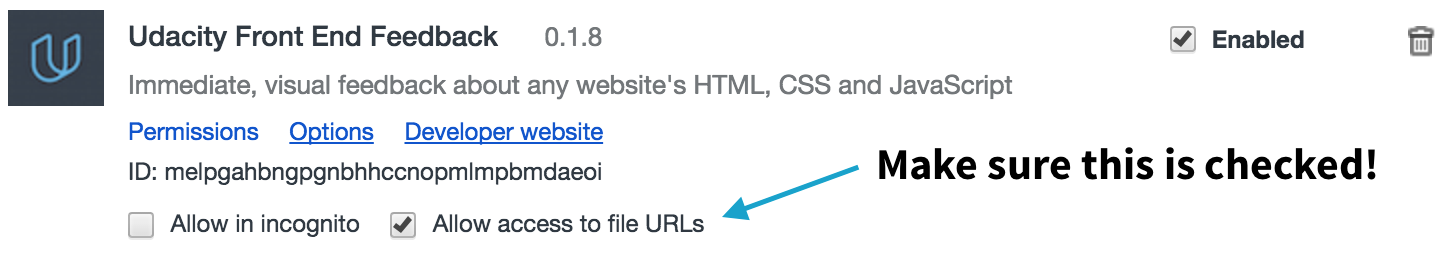
While writing code here in the Udacity classroom is a great way to learn web development, we think it's really important that you practice working with a text editor and a browser on your own computer. But we also think it's incredibly important that you get feedback on your code as you write it. So, we created the Udacity Front End Feedback Chrome Extension to give you feedback on your sites as you work on your own computer.

For the following questions and others, we expect that you'll be writing code in your text editor (doesn't matter which one) and then loading your sites in Google Chrome with this extension enabled. For some quizzes, the extension will give you a code that you'll copy and paste back into the classroom to let us know that you've finished the quiz successfully.

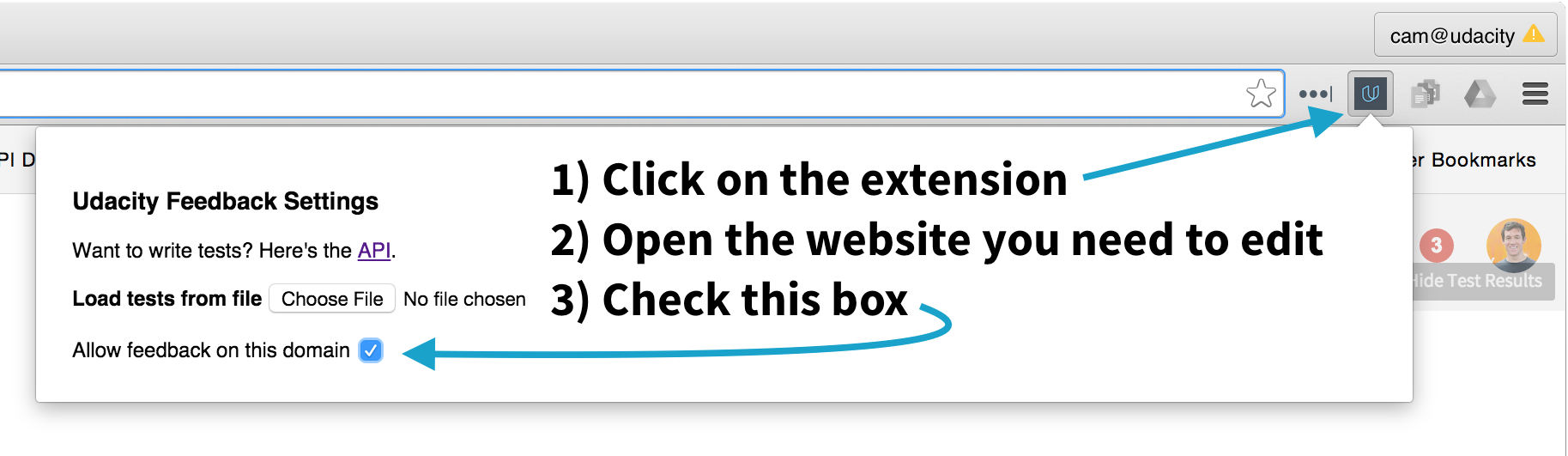
### How to Load the Extension

**1. Install the**[**Udacity Feedback Chrome Extension from the Chrome Web Store**](https://chrome.google.com/webstore/detail/udacity-front-end-feedbac/melpgahbngpgnbhhccnopmlmpbmdaeoi)**(it's free).**

**2. Go to the Chrome extensions menu (chrome://extensions) and give the extension permission to run against file URLs.**



**3. Open the website you want to edit, click on the extension icon, and check "Allow feedback on this domain":**



**4. Edit your website! You'll see the feedback change as you make changes :)**

Interested in learning more about the extension? [**Click here**](http://labs.udacity.com/udacity-feedback-extension/) for a walkthrough and more information!

**MAKE ALL THE HEADER**

Websites have different ways to show you, the user, what is important. Take for example...

## Big Important Words!

This style of text (Big Important Words!) is called a **header**. This should look familiar because almost every website uses headers.

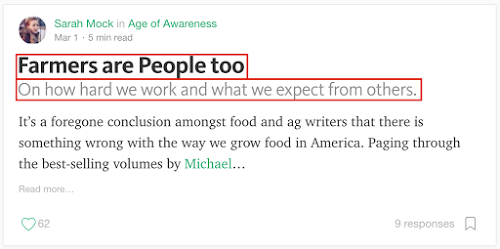
The New York Times mobile site uses them. (You can read the[***sad story here***](http://mobile.nytimes.com/2016/03/08/technology/raymond-tomlinson-email-obituary.html?_r=0).)



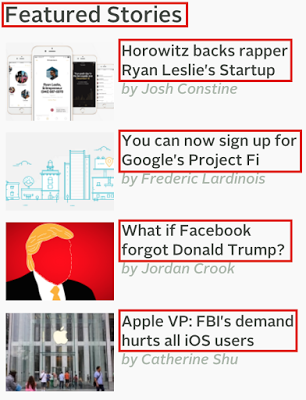
CNN sometimes puts headers over images.



It's also pretty common to see more than one kind of header being used. Here's an example of two headers on Medium.



And this is TechCrunch using headers to show the titles of featured stories.



For this quiz, I want you to experiment with four levels of headers (h1-h4).

### How to Complete this Quiz

See look, I just used a header to help you figure out why this section is here :)

* Download the zip file called headers.zip from the downloadables section. You'll find index.html inside.
* Turn on the [**Udacity Feedback Chrome Extension**](https://chrome.google.com/webstore/detail/udacity-front-end-feedbac/melpgahbngpgnbhhccnopmlmpbmdaeoi) if you haven't already.
* Edit index.html with your text editor and load your website in Chrome with the feedback turned on. When you've add all four levels of headers, you'll get a code. Copy the code into the box in the next screen to finish the quiz.

Click "Continue to Quiz" when you're ready to go!

SKIP TO QUIZ

MAKE A LIST

Did you know that web developers spend 90% of their time looking things up?

Ok, I made up that number.

But seriously, making sense of documentation and looking up new techniques and technologies is a huge part of any web developer's work. And that's what I want you to do for this quiz.

For this quiz, I want you to:

* Notice that what you're reading right now is an unordered list :) An unordered list usually displays with bullet points.
* Use the [**Mozilla Developer Network (MDN)**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element) to research **unordered lists** (<ul>).
* Make an unordered list with the three web languages:
  + HTML
  + CSS
  + JavaScript

There are two kinds of lists - ordered lists and unordered lists. Both of them require two kinds of elements. One is going to be nested inside the other (there will be one parent and multiple children).

### How to Complete the Quiz

Another unordered list!

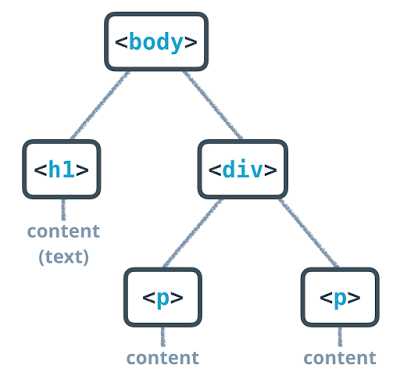
* Download the zip file called lists.zip in the downloadables section.
* Edit the index.html file inside by adding the unordered list described above.
* Turn on the Udacity Feedback Chrome Extension (if you haven't already).
* When you've successfully created the unordered list, a code will appear in the extension. Paste the code into the classroom in the next screen.

Click "Continue to Quiz" when you're ready to start!

SKIP TO QUIZ

**TREE TO HTML**

You've been learning about the relationship between data trees and HTML. Here's a sample tree that you'll need for this quiz:



I want you to build this tree with HTML.

**How to Complete this Quiz**

* Take a close look at the relationships in the tree. Pay attention to parents, children and siblings. Notice that some elements have text content.
* Download the file called tree-to-html.zip in the downloadables section. Inside you'll find index.html and an image of the tree from above.
* Turn on the Udacity Feedback Chrome extension and edit the index.html file so that the HTML has the same structure as the tree.
* When your HTML has the same structure as the tree, the Chrome extension will give you a code. Paste it in the classroom to complete the quiz!

Click "Continue to Quiz" when you're ready to get started!

SKIP TO QUIZ

**CONSTRUCTING LINKS**

"It's dangerous to go alone! Take this." Cameron hands you a hyperlink to MDN...

Whoops! That's the wrong kind of [**Link**](https://en.wikipedia.org/wiki/It%27s_dangerous_to_go_alone!).

### Hyperlinks

The power of the web is that pages can lead to other pages. If you click on the blue text above that says "Link," your browser will open up a page about a classic video game. This is called a **hyperlink**.

Hyperlinks are created with [**anchor elements**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/a), which generally look like:

<a href="https://www.udacity.com">Udacity</a>

and render on the page like this: [**Udacity**](https://www.udacity.com/).

Inside the opening a tag there is href, which stands for "reference." This is called an **attribute**. Attributes like href describe the properties of HTML elements. In this case, the href attribute is the target URL that the link will open. The content inside the anchor element is the text that users see displayed on the page.

This is the format that you must use when you make hyperlinks! Note:

* There is a space between a and href.
* There are no spaces around the =.
* The website has two " around it.
* There are no spaces between the href attribute and the > of the opening tag.

### How to Complete this Quiz

For this quiz, I want you to make your first hyperlink! On the next page, I'll give you a URL and the text that needs to be displayed. It will be your job to write the link!

Click "Continue to Quiz" when you're ready to give it a shot!

SKIP TO QUIZ

**CREATE AN IMAGE**

IMAGES! Images on the web are awesome. Time to make one.

For this quiz, you'll be creating an image here in the Udacity classroom. I'll give you the URL of an image and it will be your job to make it display.

An image is made with an  element. It looks like so:

<img src="http://somewebsite.com/image.jpg" alt="short description">

The source attribute, src, is like the href of a link - it is the URL of the image you want to display. For now, your images will need to be hosted online, which means that the URL will need to start with http:// or https://. You'll learn about another way to set image source in the next quiz. The alt attribute stands for "alternative description," which is important for people who use screen readers to browse the web. This is text that will show up in lieu of the actual image.

An image element is a little different than the elements you've seen before. Images do not need closing tags!

Here's the [**MDN reference about images**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/img).

Click "Continue to Quiz" when you're ready to go!

[**Puppy image by dbking**](https://commons.wikimedia.org/wiki/File%3ASt._Bernard_puppy.jpg) (originally posted to Flickr as 272\_7242) [**CC BY 2.0**](http://creativecommons.org/licenses/by/2.0), via Wikimedia Commons

[**Kitten image by Steve-h**](https://commons.wikimedia.org/wiki/File%3AGreen_eyes_kitten.jpg) (Flickr: Maggie) [**CC BY-SA 2.0**](http://creativecommons.org/licenses/by-sa/2.0), via Wikimedia Commons

[**Earth image by Norman Kuring**](http://visibleearth.nasa.gov/view.php?id=78314), NASA GSFC, using data from the VIIRS instrument aboard Suomi NPP.

**A GUIDE TO PATHS**

In a moment, you'll be making a website that displays an image that is stored locally on your computer. In order to display a local image, you need to be able to write a **path**.

**tl;dr** If there is a file called index.html in a directory and there is another directory calledexample/ in the same directory, you can access any files in example/ from index.html with the URL (path) example/filename.html, e.g. <a href="example/filename.html">Example Path</a>.

## Paths

A path is a way of describing where a file is stored.

Think of it like this:

Anyone in the world can use the address 1600 Pennsylvania Ave NW, Washington DC, USA 20006 to find the White House. A street address is an absolute path to a location.

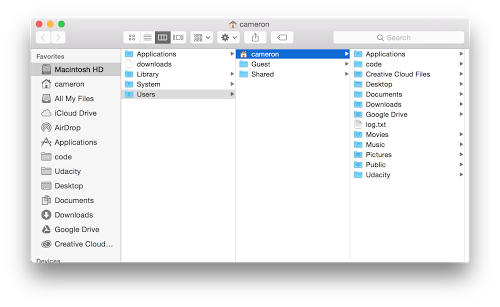
But, if you were at the [**Eisenhower Executive Office**](https://www.google.com/maps/place/Eisenhower+Executive+Office+Building/@38.8974712,-77.0390948,17z/data=!4m7!1m4!3m3!1s0x89b7b7bcdec17ee3:0xf920b148b3d45e45!2s1600+Pennsylvania+Ave+NW,+Washington,+DC+20500!3b1!3m1!1s0x0000000000000000:0x054470506cffbeb3), you could also use the phrase "next door" to find the White House. "Next door" is a relative path because it depends on your current location.

There are essentially two domains for paths that you'll need to consider as a web developer: paths to find files on your computer, **local** files, and paths to find files on other computers, **external**files.

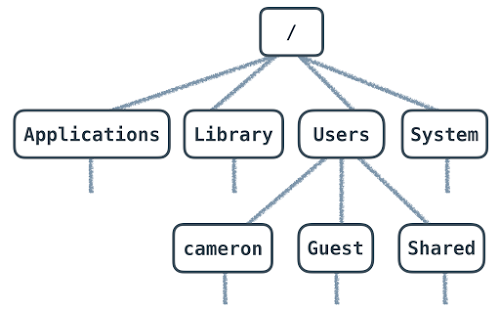
### Local Paths

Computers have folders (also called "directories"). Operating systems like Windows, Mac and Linux organize all of your files into a tree of directories called a **file system**. There's a top-most directory, often called the **root**, that contains all of the other directories. Within the root, there are files and directories. Within those directories are more files and more directories. And within those directories are even more files and directories, and so on.

Compare this part of the file system on my computer:



to a tree diagram showing the same directory structure:

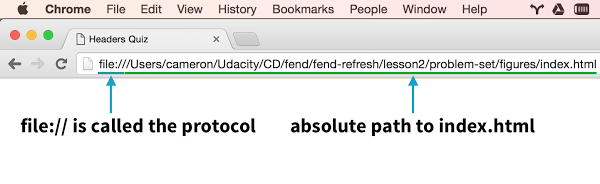


Every file has an address, which we call the "path." An absolute path is written in relation to the computer's root directory. For instance, a file in the Documents folder on a Mac has a path that looks like this:

/Users/cameron/Documents/**file**.txt

file.txt is stored inside Documents/. Users/, cameron/ and Documents/ are all names of directories. Documents/ lives inside cameron/ and cameron/ lives inside Users/. Users/ is inside the root directory, which is represented by the first /. The rest of the / are used to separate directories.

When you open an HTML file in your browser, you're seeing the absolute path to the file on your computer.



This URL will only work for you on your computer. As no one else has your file system, this URL is unique to your computer. If you want other people to be able to access it, then you need an...

### External Paths

The process of loading a website from a URL like https://www.udacity.com mimics opening an HTML file that you've written and saved to your computer. Every website starts with an HTML file. It just so happens that when you want to visit a website, the HTML file that you want to open lives on a different computer. The computer responsible for giving you a website's files is called a**server**.

Pointing your browser to https://www.udacity.com sends a request to Udacity's server for the HTML file (and others) that your computer needs to load the Udacity website. You can think ofudacity.com as the root path of Udacity's server (computer) that anyone can access (the reality of the situation is actually much more complicated but the general idea is true). Unlike your personal computer (for now!), Udacity's servers run software that **expose** files to the web, which means that they make them available to anyone who wants them. Servers have an **external path**that anyone can access and is the reason why the web works.

Different websites are just different collections of files. Every website is really just a server (or many servers) with an external address, which we call a URL. Servers store files and send them to computers who request them (the requesting computers are called **clients**).

There are different **protocols** for serving files, the most common of which on the web are HTTP and HTTPS. When you open a file on your own computer, you're using the file protocol. You don't need to know much more about protocols for now, but if you're interested in learning (a lot!) more about HTTP, check out [**Networking for Web Developers**](https://www.udacity.com/course/networking-for-web-developers--ud256).

### Relative Paths

The relative path is similar to the absolute path, but it describes how to find to a file from a directory that is not the root directory. Like using the phrase "next door" to tell someone how to find the White House from the Eisenhower Executive Office, a relative path takes advantage of the location of one file to describe where another file can be found.

Relative paths work the same for both local and external paths. Let's break down two examples of absolute paths to see how relative paths work.

**External**:

<a href="http://labs.udacity.com/fend/example/hello-world.html">Hello, world!</a>

**Local**:

<a href="/Users/cameron/Udacity/etc/labs/fend/example/hello-world.html"> Hello, world!</a>

href is really just a path to a file.

Both examples are links to the same file using absolute paths, but the external example would work for anyone and only my computer can use the link in the local example.

Pay attention to the fend/example/hello-world.html portion of both paths - they mean the same thing.

Imagine that you are editing /Users/cameron/Udacity/etc/labs/fend/test.html.test.html can reference hello-world.html by describing how to get from it's location infend/ to hello-world.html. The relative path would look like:

example/hello-world.html

This relative path takes advantage of the fact that test.html and example/ are in the same directory.

But what if I'm editing a file in /Users/cameron/Udacity/etc/labs/ and I want to write a path to hello-world.html? In that case, the relative path would be:

fend/example/hello-world.html

Now that I'm in labs/, not fend/, I have to include fend/ in a relative path to hello-world.html.

To finish this off, let's imagine there are two files:

http:*//labs.udacity.com/science/sciences.html*

and

http://labs.udacity.com/science/physics/relativity.html

In order to write a relative path from sciences.html to relativity.html, I only need to include the part of the path that describes how to get from science/ to relativity.html:

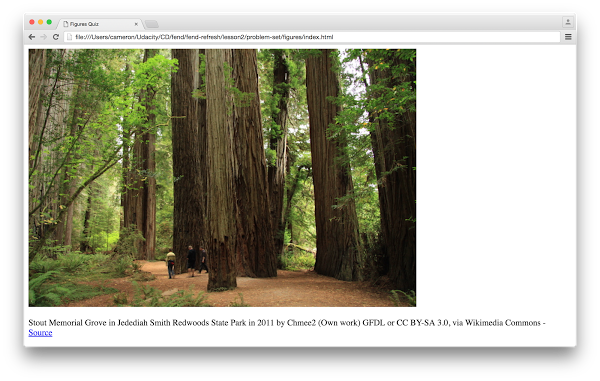
<a href="physics/relativity.html">Einstein's Special Relativity</a>

And that's it! You'll be using relative paths in the next quiz.

**QUIZZ FIGURES**

It's important to respect copyright and attribution on the web. When you use someone else's work, you need to give the author credit. With that in mind, for this quiz you're going to create an image with a caption underneath. I want you to give credit to the image's photographer by attributing them in the image's caption and providing a link to the source material.

This is what it should look like in your browser:



### How to Complete this Quiz

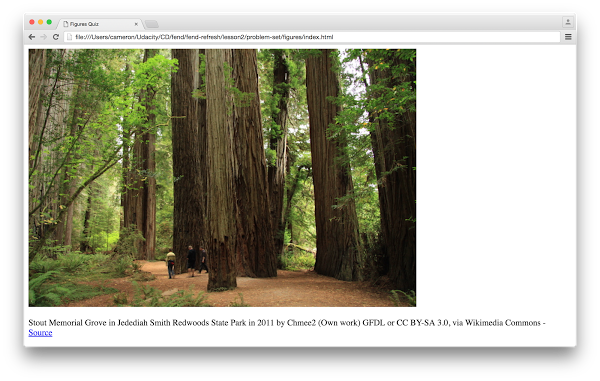
* Download the file called figures.zip from the downloadables section. You'll find index.html and an image inside.
* Edit index.html so that the caption displays below the image. I want you to use a **relative** path to the image, which is going to be in the same directory as index.html. Remember, a relative path points to the file **from the current directory** - it does not start with the root. (Your relative path should not start with /, file://, C:\, or http://).
* There's more than one option for completing this quiz **but you only have to complete one option!** You'll get a different code depending on how you complete it. Type the code into the box in the next screen to pass the quiz!

Hint: There is an HTML element made specifically for this situation! I recommend checking out the [**figure element on MDN**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/figure).

SKIP TO QUIZ

**Quiz: Figures**

This is how the website should look in your browser:



Notice that there is a link to the source in the caption below the image.

#### Supporting Materials

**[figures.zip](https://www.udacity.com/api/nodes/7265894385/supplemental_media/figureszip/download" \t "_blank)**

**QUIZZ FORMAT A SIMPLE ARTICLE**

### Mockup to Website

It's common for web developers to work with designers who focus on creating user interfaces and user experiences. Designers use software like Adobe Photoshop to mock up - draw - websites. The mockups that they create are usually just images of websites with some annotations and descriptions.

As a web developer, it's often your job to translate the mockups that designers create into live websites. I chose the word "translate" because the process of going from mockup to website is similar to the process of translating between natural languages. Just as you can create the same meaning using different words and phrases with a natural language, you can create the same website design using different HTML elements.

I want you to practice the process of going from a mockup to a website now! Here is a website mockup (note: I zoomed in for the screen shot):



You can find a copy of this image in web-dev-blog.zip in the instructor notes. You'll also find a file to start editing (index.html) and a copy of the mockup.

There are many ways to turn this mockup into website. As such, you won't be getting feedback on your site using the Chrome extension. Rather, I want you to compare your website to the mockup. You'll know you've finished this exercise when your site looks the same :)

### How to Complete this Quiz

* Download and unzip web-dev-blog.zip, which you can find in the downloadables section.
* Open the mockup and decide how you want to create the article.
* Edit index.html until your website looks identical to the mockup.
* Practice indenting children elements. I'll show you how I indented my HTML in the solution.

Just to be clear, there is no feedback from the Chrome extension. It's up to you to decide when your website looks identical to the mockup.

Click "Continue to Quiz" when you're ready to get started!

SKIP TO QUIZ

**Quiz: Format A Simple Article**

#### Supporting Materials

**[web-dev-blog.zip](https://www.udacity.com/api/nodes/7231514902/supplemental_media/web-dev-blogzip/download" \t "_blank)**