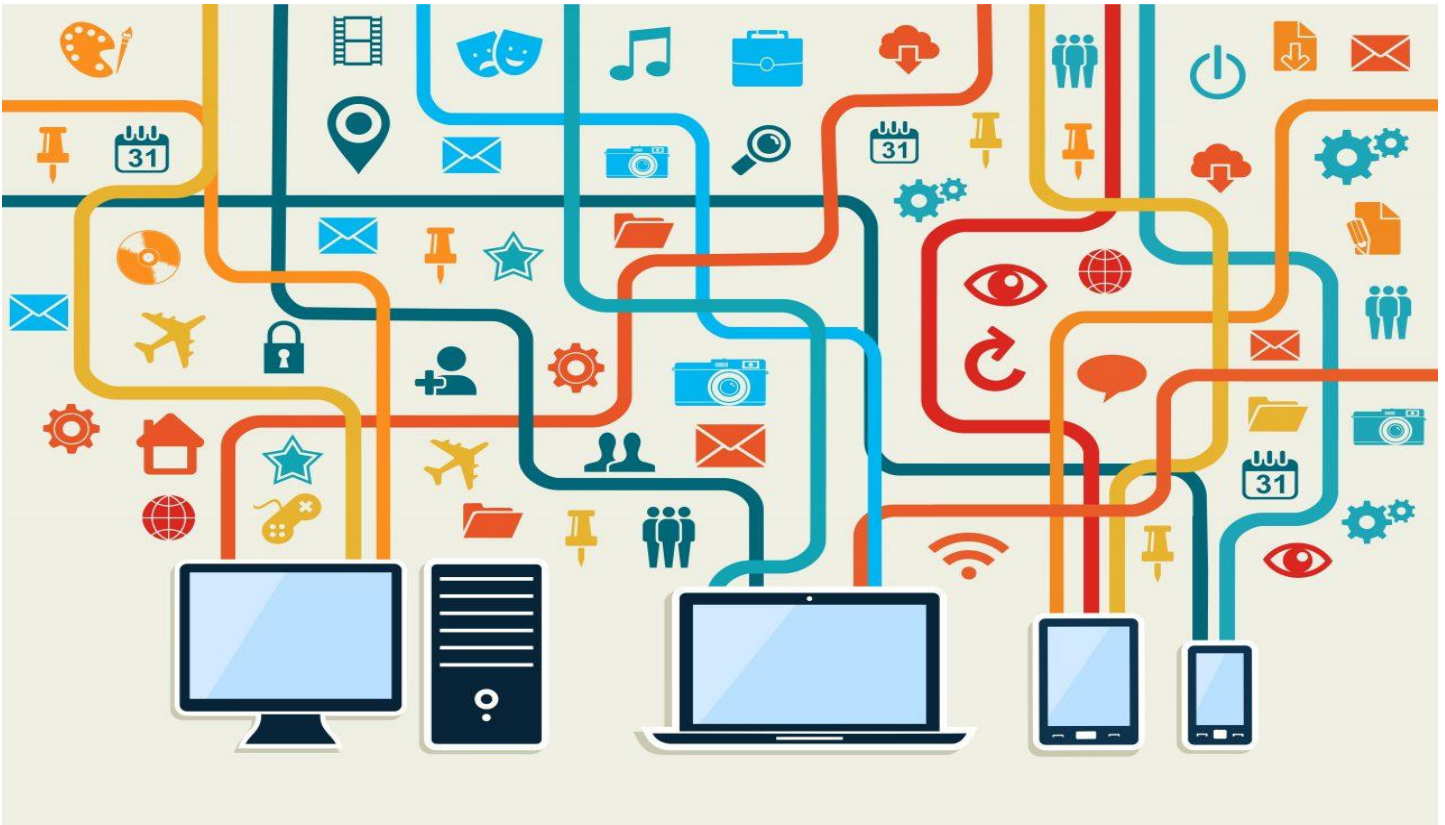


Building Your First Website

Lesson Guide



On the move... Bringing technology into classrooms



UNLV | HOWARD R. HUGHES
COLLEGE OF ENGINEERING

Table of Contents

Lesson Overview	3
Brief Description	3
Knowledge Targets	3
Standards	3
Suggested Timeline	4
Warm-up Activity: Exploring How Websites Are Made	5
A World of Websites	5
The Code Behind the Page	6
Activity Outline	8
Getting Set Up	8
Getting Started with HTML	9
Getting Started with CSS	9
Images and Attributes	10
Container Elements	13
The <head> Tag	14
Basic JavaScript	14
Sharing Your Website	16
Wrapping Up	17
Reflection	17
Further References	18
Answer Key for Warm-Up Activity	18

Lesson Overview

Brief Description

Students will learn the basics of web development with HTML, CSS, and JavaScript by coding a shareable webpage.

Knowledge Targets

- Students can identify the programming and markup languages used in web development and the differences between them.
- Students can use the basic functionality of an online code editor.
- Students understand how opening and closing tags work in HTML.
- Students can modify the text and visual content of their website.
- Students understand how to write basic style rules in CSS to modify properties such as color, margin, width/height, and border.
- Students can use CSS selectors for an element's name, ID, or class.
- Students understand how attributes affect an HTML element.
- Students understand how HTML tags can be nested.
- Students understand how to make lists on their website.
- Students can add an event listener to an element on their page.
- Students understand the basics of how callback functions change and what the event listener does.
- Students know how to access and share the website they created during the workshop.

Standards

Grade 4

- 4.ID.A.1 Demonstrate how a design process works to generate ideas, consider solutions, plan to solve a problem, or create innovative products that are shared with others.
- 4.ID.D.1 Demonstrate perseverance when working with open-ended problems.
- 4.CC.B.1 Create an original, digital work as a form of personal or group expression.
- 4.CC.C.1 Communicate information and ideas to an intended audience using digital text, images, and audio.

Grade 5

- 5.KC.D.1 Propose solutions to real-world problems using collected data and digital tools.
- 5.CC.B.1 Create original works and learn strategies for responsibly remixing or repurposing to create new artifacts.

Grades 6-8

- 6-8.EL.D.1 Navigate a variety of technologies and transfer their knowledge and skills to learn how to use new technologies.
- 6-8.KC.D.1 Explore real-world issues and problems through inquiry and analysis, develop ideas, actively create solutions for them, and evaluate and revise through the use of digital tools.
- 6-8.ID.A.1 Engage in a design process and employ it to inquire and analyze, generate ideas, create innovative products or solve authentic problems, and evaluate the process to revise if needed.
- 6-8.CT.A.1 Practice defining problems to solve by computing for data analysis, modeling, or algorithmic thinking.
- 6-8.CC.B.1 Create original works and apply strategies for responsibly remixing or repurposing to create new artifacts.

Grades 9-12

- 9-12.ID.A.1 Engage in a design process and employ it to inquire and analyze, generate ideas, create innovative products or solve authentic problems, and evaluate the process to revise if needed.
- 9-12.CT.A.1 Define complex issues, create a plan, and select appropriate technology-assisted methods such as data analysis, abstract models, and algorithmic thinking in exploring and finding solutions.

Suggested Timeline

The workshop is designed to take place over three approximately hour-long blocks. Throughout the lesson plan, look for the arrows on the side of the page that indicate how the activity is split into days:



These are just suggestions. If your session takes shorter or longer based on the number of questions students have, that is perfectly fine!

Warm-up Activity: Exploring How Websites Are Made

A World of Websites

Did you know?

There are over 1.8 billion websites on the Internet.

Question

List 5 of your favorite websites.

1. _____
2. _____
3. _____
4. _____
5. _____

Discussion

Think about why you like these websites. How do they make your life better? What do you think their creators were thinking when they made them? Talk about these questions with the class or your neighbor.

Did you know?

All of the websites you listed (and every other website on the Internet) has been created by **code**.

When **programmers** write code, they tell your Internet browser what the website should have on it, how it should look, and what it should do as you use it.

The Code Behind the Page

Let's Learn More

There are three kinds of code that are most important to making websites: **HTML**, **CSS**, and **JavaScript**.

HTML

- Stands for “Hypertext Markup Language”
- Defines what you should see on a website, like text and pictures, and their content
- Tells the website where each element should show up, what order should things be in, should they go on top or on bottom, etc.
- Example:

```
<h1>My First Heading</h1>
<p>My first paragraph.</p>
```

CSS

- Stands for “Cascading Style Sheets”
- Adds style and design to the elements on a website
- Can change the color, position, size, animation, and much more
- Example:

```
p {
  color: #333;
  font-size: 13px;
  line-height: 18px;
  font-family: Arial, Helvetica, sans-serif;
}
```

JavaScript

- Makes websites interactive
- Changes what happens when you click or tap on different parts of a website
- Can make websites dynamic, so the website may be different one time you visit it compared to the next time
- Example:

```
function sayHello() {
  alert("Hello!")
}
```

Practice Questions

Pretend you're creating a website. Next to each thing you want to change about your website, write which programming language would help you most to make that change.

1. I want the website to have a heading with my name on it. _____
2. I want my name to be blue and bold. _____
3. I want my name to be above my picture. _____
4. I want my picture to be of a dog. _____
5. I want my picture to have a red border. _____
6. I want my website to display a message when you click on my picture. _____
7. I want to have a button that makes a new picture appear. _____
8. I want to have a caption under my picture. _____
9. I want the caption to have an animation. _____
10. I want the caption to be a random color every time I go to my website. _____

Congratulations!

Now that you know a little bit about how websites are made, it's time to make your own.

By the end of this workshop, you will be able to do all of the 10 things listed above...and so much more!

Activity Outline

Getting Set Up

Day 1

1. Go to <https://www.glitch.com> and press “Sign Up” to create an account.
 - a. Make sure you keep note of your username and password so you can come back and look at your code later.
2. Go to <https://glitch.com/edit/#!/web-tutorial-base> to access the template for your website. Click “Remix” in the upper right corner to create a copy of the template on your account to edit.
3. Go to the project you remixed and look at the menu on the left. This menu can be used to switch between the different files in your project. Click through the different files and look at the contents of each one.
 - a. index.html has the HTML code that defines the layout of your website.
 - b. script.js has the JavaScript code that makes your website interactive.
 - c. style.css has the CSS code that you will edit to customize the look of your website.
 - d. LICENSE and README.md give details about your project. This is helpful for when other people want to use or understand your website.
 - e. base-styles.css has some basic CSS styles that are used in the template. Please do not edit anything in this file.
4. Switch to index.html and try typing something in the window with the text. This window is called a code editor: just like Google Docs or Microsoft Word, you can type whatever you want in the code editor window.
5. On the bottom of the screen, click the button with the magnifying glass that says “Preview”, then choose “Open preview pane”. You can see your website show up on the right side of the screen. As you make changes to your code, you will see your website change in real time in this preview.

LICENSE

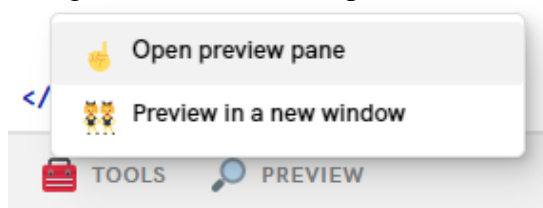
README.md

base-styles.css

index.html

script.js

style.css



For teachers:

You can access a sample of the finished project at this link: <https://glitch.com/edit/#!/web-tutorial-complete>.

Getting Started with HTML

1. HTML uses **tags** as the building blocks of a website. In your template, there is an example of an `<h1>` tag:

```
<h1>MY WEBSITE</h1>
```

Each tag defines a part, or element, of your website.

It starts with an opening tag: `<h1>`

In the middle is anything that goes inside the element: “MY WEBSITE”

And ends with a closing tag: `</h1>`

2. You can change the content of your website by changing what goes between the opening and closing tag. Change the header to say “<YOUR NAME>’s WEBSITE” instead of “MY WEBSITE”.
3. An `<h1>` is just one of many types of HTML tags. It stands for “header 1” and defines a large section header on your website. Here are some other common HTML tags:
 - a. `<h2>`, `<h3>`, `<h4>`: Smaller headers
 - b. `<p>`: paragraph
 - c. `<div>`: container (We will talk more about this later in the activity)
 - d. ``, ``: lists
 - e. `<nav>`: navigation
4. Under the `<h1>` tag, create a new `<h2>` tag and make it say whatever you want.
 - a. Remember to include the opening tag and the closing tag.

Getting Started with CSS

1. Now, we will customize the way your HTML elements look using CSS. Switch over to the `style.css` file in the left panel.
2. In CSS, you write code for rules on how different elements should look and which elements those rules should apply to.
3. In `style.css`, we will create a CSS **selector**. A selector points to what parts of our website will follow the rules we write for that selector.
 - a. One way we can select elements on our page is using the name of their tag. Here is a selector that selects all `<h1>` tags:

```
h1{
}
```

Notice how the selector starts with the name of the tag, then has an opening and closing curly bracket {}.

4. We will put our CSS **rules** inside the curly brackets. Every rule tells one part of the element how it should look: for example, there are rules for color, margin, font, and more.

- a. Here is a rule to make our header red:

```
h1{
  color: red;
}
```

The rule has three parts: the name of the property we're changing (color), a colon (:), and the value we want to give the property (red).

- b. Look at the preview of your website. What changed color? What didn't?
5. Create another selector and rule to make your <h2> tag a different color than your <h1> tag.
6. Color is far from the only thing we can change using CSS.
 - a. Try setting the background-color property of your <h1> or <h2> tags.
 - b. Try setting the font-size property of your <h1> or <h2> tags. You can tell the CSS code how big the font should be by giving it a number followed by "px", which stands for pixels.
Example: font-size: 20px

Images and Attributes

1. Now, we're going to add a new type of HTML element to our page: an image. To make our image go to the right of our headers, we are going to put it under the closing tag of the <div> named "name-box". Here it is highlighted in the template:

```
<div class="col-container" id="outer-box">

  <div class="row-container" id="inner-box">

    <div class="col-container" id="name-box">

      <h1>MY WEBSITE</h1>

    </div>

    <!-- Put the image here! -->


  </div>
</div>
```

2. Once you have found the spot, create a new tag: . When you create the tag, you will see that no image shows up yet. This is because we have to use an **attribute** to tell our HTML what image it should show.

- a. Attributes tell our HTML more details about a tag. They live in the opening tag and are placed between the name of the tag and the closing >. In fact, you can see some tags already have “id” and “class” attributes in your template:

```
<div class="col-container" id="outer-box">
```

Like a CSS rule, an attribute has three parts: the name of the attribute, an = sign, and its value in quotes.

3. The attribute we need for our tag is called “src”. Using the class and id attributes above it as an example, add an attribute called src to your tag. For now, you can set its value to anything.
 4. Now, let’s make a real image appear by fixing the value of the src attribute. Src stands for “source”, and it tells our HTML element where it should find the image to display.
 - a. Use Google Image Search to find a picture of your favorite animal or food. Once you find an image you like, right-click on the image and select “Open image in new tab”.
 - b. Once the image opens in a new tab, copy the URL of the website that opens. You can find the URL in the search bar at the top of your browser:
-  hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/dog-puppy-on-garden-roy...
- c. Paste the URL as the value of the src attribute on your tag. Remember to put it in quotes!
 - d. Check the preview of your website to make sure you can see the image you chose.
 5. Two more important attributes are “class” and “id”. Both of these attributes are used to label elements on your page.
 - a. Class is used to group similar elements together. If you have three tags on your page with the same class, this means that those tags share something in common.
 - b. Id is used to uniquely identify one element, like a nametag. Every id attribute on your page should be unique.
 - c. Take a look at the class and id attributes on the <div> tags on the page. Notice how the class “col-container” is repeated on two <div> tags, while the id attribute is unique for every <div>.
 6. The “class” and “id” attributes are important in CSS because we can use them as selectors.
 - a. Remember what we used as a selector before? We could only use the name of a tag. But what if we wanted to make only one <h1> tag red, not all of them? This is where other selectors come in.
 - b. Go back to the style.css file. We are going to make a selector that changes the background color of our page by selecting the <div> with an id attribute of “outer-box”.
 7. To select an element by its “id” attribute, we use this format:

```
#id{
  |
}
```

This tells our CSS to apply the rules in this selector to the element whose “id” attribute matches what comes after the #.

- a. Using this format, make a CSS selector for the id “outer-box”.
 - b. Add a rule for the background-color property to your new selector and see what happens to the preview of your website.
8. We can also select elements by their class using this format:

```
.class{
  |
}
```

This tells our CSS to apply the rules in this selector to any elements whose “class” attributes match what comes after the period.

Remember, we use classes to group similar elements together, so you can use class selectors to give similar elements the same CSS styles.

9. Practice what you’ve learned by adding an “id” attribute to your tag and using it to add some new styles to it.
- a. Is your image too big or too small for your website? The width and height properties can help you here.

Just like when you changed the font-size property earlier, we will use “px” (pixels) to tell our CSS how big or small the image should be. Here is an example of how to change the width of an image:

```
width: 500px;
```

This tells our website that the image should be 500 pixels wide. Try using this property along with “height” to make your picture the size you want.

- b. Try adding a border to the image using the border property. We can customize three parts of the border: the thickness, the shape, and the color.
For example:

```
border: 10px solid red;
```

10px tells our CSS how big the border should be.

Solid tells our CSS what the border should look like (other options include dashed, dotted, and inset).

Red tells our CSS what color the border should be.

- c. Is your image too close to your name? Let’s try giving it some room with a margin!
A margin is the distance between one element and the other elements next to it. Practice setting the margin CSS property. Its value is in pixels just like width and height.

Container Elements

1. Think back to when you first added text your `<h2>` tag. Up until now, we have been putting text in between the opening and closing tags just like this.
We can actually put much more than just text between the opening tags- we can nest other tags inside! When we put a tag inside another tag, we call the tag on the outside a **container**. The most common tag to use as a container for other tags is the `<div>`.
How many `<div>` tags are on your website right now? What is inside each one?
2. HTML is read top-to-bottom, but tags inside the same `<div>` can go side by side. Think of each `<div>` like a box: we can stack boxes on top of each other, but we can also arrange things inside the same box.
3. Look at the `<div>` with the id “inner-box”. There are two things inside of it: your `` tag, and another `<div>` with your `<h1>` and `<h2>` tags inside of it.
 - a. Try moving the `` tag outside the “inner-box” `<div>` and see what happens to the layout of your page.
Remember: moving it outside the `<div>` means moving it somewhere after the `<div>`’s closing tag.
 - b. Now try moving the `` tag outside the “outer-box” `<div>`. What happened this time? What’s different about your `` now that it’s outside the “outer-box” `<div>`, and why?
4. Now add your own new container `<div>` to your page. Try putting it inside the “outer-box” `<div>` but outside the “inner-box” `<div>`.
5. Let’s add another element inside your new `<div>` so it can be a proper container! This time, we’ll use a new tag: the `` tag.
 - a. `` stands for unordered list. It creates a list of bullet points. Add an opening and closing `` tag inside of your new `<div>`.
 - b. You may notice that nothing shows up in your list yet. This is because the `` tag is actually also a container. Each `` tag you nest inside of the `` tag will become a new item in your list.
 - c. Using what you’ve learned about container elements, try adding three `` tags inside of your `` tag. You can make whatever list you want: a list of your favorite animals, foods, vacation spots... your imagination is the limit!
 - d. Once you’re done creating your list, give it a label by adding a new header tag above it inside of your new `<div>`.

The <head> Tag

1. You may have noticed a special tag at the very top of your HTML file: <head>. The tags inside the <head> tag are like a nutritional label for your website: they tell the browser and people who visit it some important information about the behind-the-scenes details.
- 2.
3. Take a look at the <title> tag inside the <head> tag. Try changing it to something other than the default.
 - a. Now, go back to where you opened the preview panel for your website, but select “Preview in a new window” instead.
 - b. Look at the name of your website on the tab. How did it get there?
4. The two <link> tags tell your HTML code where to look for its CSS styles. Do you notice how the names in the “href” attribute of the <link> tag match with the names of the CSS files you have in your project?

Basic JavaScript

1. It’s time to make your website interactive! Users usually interact with websites through things they can click on, type in, or change in some other way. To start out, we’re going to be making two different buttons that people who visit your website can press.
2. Make a <button> tag underneath your tag. In the <button> tag, the text you want to show on the button goes in between the opening and closing tag.
 - a. Give the <button> tag a unique id attribute—we will need it in the next step!
3. Open the “script.js” file in your editor. You will already see a little code in this file—please don’t change it for now.
4. Look at the first line of code in the file:

```
let myBtn = document.getElementById("??")
```

This creates a **variable** named myBtn. A variable keeps track of a value in our code so we can use it later.

document.getElementById()’s job is to find an element in your HTML code based on its id attribute. It will find whatever id you put in quotes inside the parentheses. For example:

```
let myBtn = document.getElementById("button1")
```

would find the button on your page whose id attribute is button1.

Based on the id you gave your button, replace the question marks so that your JavaScript code can find the button you created.

5. Now that your code found the button, we are going to add an **event listener** to it. An event listener waits for something to happen on your website, then tells your code what it should do when it happens.

For example, we will add a “click” event listener—it will wait until someone clicks on your button, and then run the code that will tell it what to do when it sees someone click it. There are lots of different events we can listen for from our website, from clicking and pressing buttons, to typing, and even to making the window bigger or smaller.

- a. We start our event listener like this:

```
myBtn.addEventListener("click", function(){

})
```

We tell our code to use the variable “myBtn”. Then, inside the parentheses of `addEventListener()`, we tell our code what event it should listen for (“click”), and what it should do when that event happens.

Don’t worry if you don’t understand all the parentheses and brackets right now—these are just the basics, and you will still be able to make cool things happen on your website even if you don’t understand 100% of the code.

- b. We will tell our website what code to run when we click the button by putting it inside the curly brackets. Here’s an example:

```
myBtn.addEventListener("click", function(){
    // Type here!
    // And type more here!
    // Anything you want.
})
```

- c. You can run any code you want inside these curly brackets: the technical term for this code is a **callback function**. For now, we are just going to make a pop-up appear with a message to the user.
 - d. To make a pop-up, type `alert()` inside the curly brackets in the area shown above. Then, inside the parentheses, pick the message you want to show the user and wrap it in double quotes.

Here’s an example:

```
alert("Hello world!")
```

- e. Now, see what happens when you click on your button! As an extra challenge, try making your website show another pop-up after the user closes the first one.
6. We will add one last button to your website—a rainbow button that changes color every time you click it!
 - a. Go back to your HTML code and add another button. This time, make sure its id attribute is set to “colorBtn” (without the quotes!).
 - b. Now, switch to your JavaScript code. You might see that the code that changes the button’s color is already added.

Again, you will not understand all of this code right now, and that is okay! The important thing is that it picks a random color and sets the button's text to that color.

This is another reason why JavaScript is powerful—we can use it to dynamically change the HTML and CSS on our page.

- c. First, we have to find our button again. Using the first line of code as an example, make a new variable for the color button and use `document.getElementById()` to find the button.
 - i. Hint: We want to give the variable a different name than the first variable, so you will need to change what goes before the `=` sign.
Also, remember that our new button has a different id, so think carefully about what should go inside the parentheses in `document.getElementById()`.
- ci. Next, we need to add another event listener to our new button. Again, use the first event listener as an example.
 - i. Hint: The first thing inside the parentheses of `addEventListener()` is still the event we want to listen for. Is this still “click”, or should it be something different?
 - ii. We need to add this event listener to our new button variable instead of the old one. This means that the name before the dot should be the new variable's name, not the old one!
 - iii. This time, instead of writing a new function for our event listener, we are going to use the `changeColor()` function at the bottom of your `script.js` file. Here is an example of how you can tell the event listener to use this function:


```
.addEventListener("click", changeColor)
```

Notice how the second value in the parentheses matches the name of the `changeColor()` function- that is how your code knows which function to use!
- e. Test your button and enjoy seeing all the colors of the rainbow!

Sharing Your Website

1. Congratulations on building your very first website! There are still lots of things you can customize, so feel free to have fun with changing the HTML and CSS as much as you want using everything you learned today.
2. Click the “Share” button on the upper right of your screen. Look at the link called “Live site” on the bottom: this is a link that you can share with friends and family so they can all visit your website!

Wrapping Up

Reflection

Think about the activity you completed today and answer the following questions.

1. How are HTML, CSS, and JavaScript different from one another? Which did you enjoy writing the most?

2. Think of 3 different things you still want to add to your website:

- a.

- b.

- c.

For at least one of these ideas, briefly explain how you might add them to your website using what you learned today.

Remember, you can always come back to the project you created and keep adding new things, or even make new websites! That's what's great about coding: there's always more for you to create.

3. Navigate to any website you like and right click on an empty spot on the page. On the options menu that opens, click “Inspect”. This opens a panel that shows you the HTML used to create this website, along with tons of other details on the code behind the scenes.

If you can’t see the HTML, make sure you are on the tab that says “Elements”.

Looking at the HTML for the website you chose, write about at least 3 tags, attributes, or other elements that you recognize from the website you built today.

Further References

1. <https://www.w3schools.com/>
 - a. Expansive resource for tutorials and references on HTML, CSS, JavaScript, and much more.
2. <https://www.freecodecamp.org/learn/responsive-web-design/>
 - a. A lesson plan with over 300 hours of content that guides you through the ins and outs of designing websites.
 - b. FreeCodeCamp has other courses just like this for other programming topics, so feel free to explore any you’re interested in.
3. <https://flexboxfroggy.com/>
 - a. A short game that teaches you how to use flexbox, a set of CSS properties that help you arrange elements on your website.

Answer Key for Warm-Up Activity

- | | |
|---------------|----------------|
| 1. HTML | 9. CSS |
| 2. CSS | 10. JavaScript |
| 3. HTML | |
| 4. HTML | |
| 5. CSS | |
| 6. JavaScript | |
| 7. JavaScript | |
| 8. HTML | |