



Girls Who Code At Home

Virtual Hike
Interactive Web App with Javascript

Activity Overview

With Earth Day happening this week, we want to encourage you to think about some of the wild and outdoor places you'd like to explore. Where would you go? What would you want to do and see? In this tutorial, you'll learn how to use **arrays**, or lists in JavaScript, to create a virtual hike! You'll create a website that can take someone on a trip to visit a national park, your favorite museum, or travel to a different country! The possibilities are endless! Before you start working on your itinerary, we recommend checking out the featured Woman in Tech Spotlight, Gitanjali Rao.

Note: When developing a website, a lot of programmers use HTML to build the skeleton of the website, CSS to style the page, and Javascript to add interactivity. In this activity we will mainly be working in Javascript. If you want a quick refresher on HTML and CSS we highly suggest you check out our third activity: [Share Your Skillz](#) before starting this activity.

Materials

- [Glitch](#) or the text editor of your choice
- [Example Virtual Hike Project \(with Extensions\)](#)
- [Example Virtual Hike Project \(no Extensions\)](#)
- [Virtual Hike Starter Code](#)
- Optional: Planning Guide
- Optional: Pen/Pencil/Markers

Women in Tech Spotlight: Gitanjali Rao



Image Source: [BBC](#)

In 2014 there was a public health crisis in Flint, Michigan. Flint's water supply contained lead which is extremely dangerous if consumed, especially for young children. After learning about this health crisis, Gitanjali took action and created Tethys, a device that detects if there is contamination in water.

Gitanjali was only 11-years old when she created Tethys. She went on to win the 2017 [Discovery Education 3M Young Scientist Challenge](#) and was invited to be a TEDx speaker!

Her device uses a carbon nanotube to detect the flow of electrons in lead, while also using an Arduino-based signal that connects to a smartphone app through Bluetooth.

Watch the [video](#) about Gitanjali Rao and her invention, Tethys. If you have time we recommend you learn more about Gitanjali by exploring [this article](#) from Bustle, [this article](#) from Young Scientist Lab, or [this article](#) from NPR. You can also catch her TEDx talk [here](#).

Reflect

There's more to being a computer scientist than just being great at coding. Take some time to reflect on how Gitanjali and her work relates to the strengths that great computer scientists focus on building - bravery, resilience, creativity, and purpose.



Gitanjali has an interest in the medical field, but still learned to code and build technology. What other fields are you interested in that you could possibly use computer science for?

Share your responses with a family member or friend. Encourage others to read more about Gitanjali to join in the discussion!

Step 1: Explore (5 mins)

In this tutorial you will create a website that presents a series of pictures to take the user on a virtual hike. Take 5 minutes to explore some of the features of this example [website](#) we created about places of nature in Japan!

- **Project Theme:** Nature
- **Audience:** People who want to learn more about places to view nature in Japan
- **Goal:** Share an experience outside of the home and plan a nature focused itinerary around Japan

As you explore the website think about the following questions:

- What happens when you click each button?
- Do the photos chosen relate back to the theme and goal? What photos would you like to incorporate?
- What do you think are the steps when you click the “Next” button? Try to break down the steps into small steps.
- When does the “Next” button stop working? When does the “Back” button stop working? Why might this happen?

Step 2: Brainstorm Features and Plan Your Project (10 mins)

Now that you have gotten the chance to explore a sample project, it’s a good idea to take some time to make a game plan first. Use this time to figure out what you want your project to do and what your goal will be. You may want to use the planning document at the end of this activity to help you capture and organize your ideas.

1. Choose a Theme, Audience, and Goal for your virtual hike.

In honor of Earth Day (April 22, 2020), we encourage you to pick an outdoor place you want to explore but you can choose any topic you’d like. If you are feeling stuck, here are some alternative project ideas:

- A virtual gallery of the activities/hobbies you like to do at home.
- A virtual walk through your favorite museum.
- A virtual hike to your favorite locations.

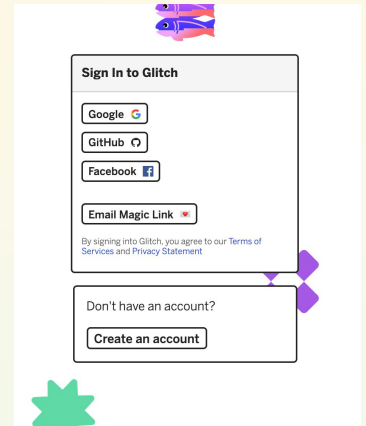
When choosing the audience of your virtual hike think about who your target audience will be. Who might be excited to use your website? Consider what they are interested in knowing and how you will capture their attention.

2. Gather at least three images.

You’ll want to include at least three images to display on your virtual journey. You can use your own images or images that you find online. As you gather your images also think about the title or text blurb you want to display with each image.

Step 3: Get Started on Glitch (10 mins)

Glitch is a simple tool for creating web apps. It comes with a text editor that allows you to see edits to your webpage in real time. It also allows you to easily publish your project for the world to see! If you find a cool project someone else made, you can look at their code and remix it.

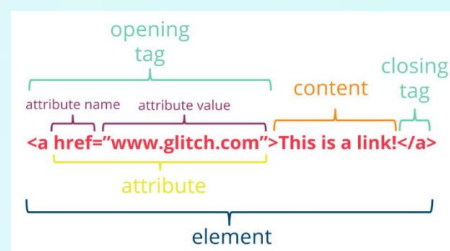
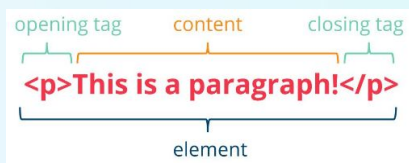


1. **Sign into [Glitch](#) using your Google, Facebook, or GitHub account.**
 - To save and share your work on Glitch, you will need to sign in. Glitch allows you to sign in using a Google, Facebook, or GitHub account. If you are under 13, you'll need your guardian's permission and email address to sign up.
2. **Remix this [starter code](#) for your virtual hike website.** Watch this [video](#) for more detailed instructions.
3. **Explore the Glitch Interface.**

The first thing you will see are the contents of the **README.md** file. READMEs are documents that you will constantly see in other people's projects. They usually contain information about how to navigate through the project's files and how to run the program. Watch this [video](#) to learn more about the Glitch interface.
4. **Setup your project view.** Click the **Show** button and choose **Next To The Code** option. This will allow you to view your code and how the project looks on the web side-by-side. Watch this [video](#) for more detailed instructions.

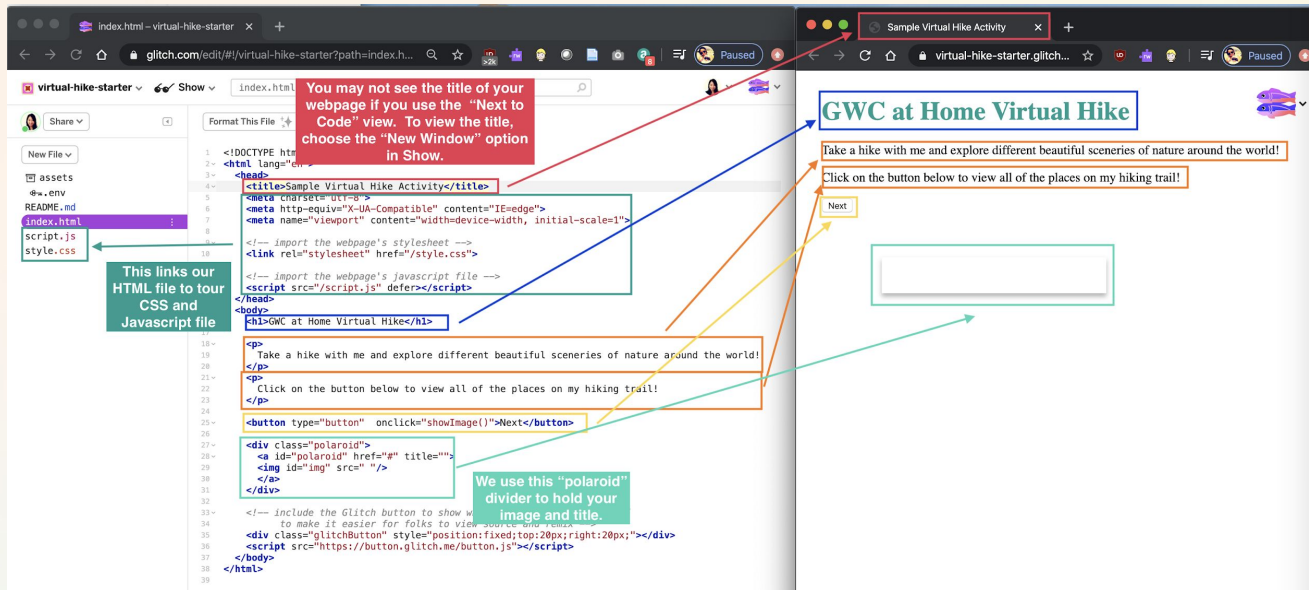
Step 4: Explore Starter Project HTML (10 mins)

Let's take a look at the [index.html](#) file first. Click on the [index.html](#) file from the left navigation menu. **HTML**, short for Hypertext Markup Language, uses **tags** to organize content for a website. All of the content that shows in your live preview are contained within a `<body>` opening tag and `</body>` closing tag. Tags are part of the anatomy of an HTML element, which are usually made up of the opening tag, content, and the closing tag. Take a look at the images below to see some examples of HTML tags for different types of elements. The `<p>` tag is used to note "paragraph" while the `<a>` tag is more of a generic tag typically assigned for links.



Step 4: Explore Starter Project HTML (Continued)

The image below maps the tags that you see in your starter code to how they look live on your website.



Let's take some time to customize the information for *your* website!

Try Coding in HTML Yourself!

- 1. Change the Title of your website.** The title of your website is the name that pops up when you load your website. Note that this is not the same as changing the URL of your website. If you want to check that the title of your webpage is correct, click **Show** and choose the **New Window** option in the menu.
 - Locate the `<title>` tag in the HTML file. (This is outlined in the red box in the image above).
 - Replace the text between the opening tag `<title>` and closing tag `</title>` with the name of your webpage.
Ex. `<title> My New Web Page Name </title>`
- 2. Change the Header of your website.** A header is the text that typically appears at the top of your page. Unlike the title of the webpage, the header will be visible on the page!
 - Locate the `<h1>` tag in the HTML file. (This is outlined in the blue box in the image above).
 - Replace the text between the opening tag `<h1>` and closing tag `</h1>` with the title of your project. This could also be the same as what you wrote for the title of your webpage.
Ex. `<h1> My New Web Page Name </h1>`

Step 4: Explore Starter Project HTML (Continued)

3. **Change the Text on your website.** In the sample code there are two paragraphs. The first contains a short description of the project while the second provides the user with instructions on how to interact with the website.

- Locate the `<p>` tag in the HTML file. (This is outlined in the orange box in the image above)
- Replace the text between the **first set** of opening `<p>` tag and closing tag `</p>` with a short description of your project. Look back at what you wrote in your planning document.

Ex. `<p>`

My New Web Page Text Description

`</p>`

- **(Optional)** Customize the instructions in the second set of `<p>` tags.

Step 5: Explore Starter Project CSS (5 mins)

CSS, short for **Cascading Style Sheets**, describes the presentation rules (or styles) that should be applied to the HTML elements. CSS allows you to change the way content on your website looks, including text color, text size, the fonts used, background colors or images, and so much more. In this tutorial we will not be going in depth on CSS. We encourage you to view [this](#) resource or try out our [Share Your Skillz](#) activity if you are interested in learning more about CSS!

In this starter project we already took care of many basic CSS style rules so you could focus on customizing and learning JavaScript. We included styling for your title, header, and polaroid. We named one of the elements **polaroid** because we added styling rules in CSS to make your photos look like polaroid photo! Styles for the polaroid were borrowed from [this](#) tutorial. You will also have time at the end of this project to customize your website to use the colors and fonts that you like!

```
1- /* CSS files add styling rules to your content */
2-
3- body {
4-   font-family: Roboto;
5-   margin: 20px;
6-   background: #ffffff;
7- }
8-
9- h1 {
10-  font-style: bold;
11-  color: #009C98;
12- }
13-
14- /*This is styling to make your photo look like a polaroid*/
15- .polaroid {
16-   background: #ffffff;
17-   display: inline-block;
18-   float: left;
19-   margin: 55px 75px 30px;
20-   padding: 15px 15px 30px;
21-   text-align: center;
22-   text-decoration: none;
23-   -webkit-box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
24-   -moz-box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
25-   box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
26-   -webkit-transition: all 0.2s linear;
27-   -moz-transition: all 0.2s linear;
28-   transition: all 0.2s linear;
29-   z-index: 0;
30-   position: relative;
31- }
32-
33- .polaroid a:after {
34-   color: #ffffff;
35-   font-size: 25px;
36-   content: attr(title);
37-   position: relative;
38-   top: 15px;
39- }
40-
41- .polaroid img {
42-   display: block;
43-   width: 250px;
44-   margin-left: auto;
45-   margin-right: auto;
46- }
47-
48- .polaroid a: hover {
49-   -webkit-transform: scale(1.2);
50-   transform: scale(1.2);
51-   transform: scale(1.2);
52-   z-index: 10;
53-   -webkit-box-shadow: 0 10px 20px rgba(0, 0, 0, 0.7);
54-   -moz-box-shadow: 0 10px 20px rgba(0, 0, 0, 0.7);
55-   box-shadow: 0 10px 20px rgba(0, 0, 0, 0.7);
56- }
```



Image Source: [Creative Market](#)

Step 6: Intro to Javascript (2 mins)

Now the fun stuff! Up until now you have been building out the skeleton of your website, but it doesn't do anything yet! That's where **Javascript** comes in. Javascript is used to make websites interactive. Let's take a look at the `script.js` file. We've added a few things to get you started, as well as a few comments that explain what each of the lines of code do.

```
1- /* If you're feeling fancy you can add interactivity
2   to your site with Javascript */
3
4 //This is an array that will hold the file names! Add at least 4 images to the images array.
5 var images = new Array();
6 images[0] = "";
7 //This is an array that holds the names of each location shown in your pictures.
8 //Make sure that the index of each location is the same as the index of the picture in the images array
9 var locations = new Array();
10 locations[0] = "";
11
12 //This is your starting index. First we start off at Home!
13 var index = 0;
14
15 //This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
16 function showImage() {
17
18 }
19
```

Note: Comments aren't read by the computer - they describe the code for others or your future self. You can add a comment on one line by typing `//` and a message.

Step 7: Intro to Variables & Arrays (2 mins)

In order to store information (like our photos and its descriptions) we need to use a **variable**. A variable is a container that stores all types of data. They can hold words, numbers, and even a list of data. Let's take a look at the key symbols (or syntax) for using variables:

```
var myVariable = value;
```

- **var** : This keyword is used to declare a variable type.
- **myVariable** : This is the name of our variable. Names usually follow [camel case](#) with no spaces and uppercase lettering signaling new words.
- **=** : The equal sign shows the assignment or reassignment of a value to the variable
- **value** : This can be any value - number, words, or even a list of data.
- **;** : All Javascript statements must end with a semicolon. This is how the computer knows that the command has ended.

An **array** is an ordered data structure that can hold multiple types of information. You can think of an array like a dresser that has many drawers, where each drawer can hold data. Let's break down the key symbols (or syntax) used to create an **empty** array (like in the starter code):

```
var myArray = new Array();
```

- **var** : We use the keyword **var** to store the array in a variable.
- **new** : This keyword is used to show that we want to create an empty object. In our case an empty array!
- **Array** : This keyword tells the computer to create an array object.
- **()** : The parentheses are used to call the array **function**. Javascript already has a built-in function (or set of commands) that helps create the array object. We will learn more about functions and how they are used later in this project. 8

Step 8: How to Use Arrays (10 mins)

Arrays are one of the most widely used data structures because it is an *ordered* data structure. An array assigns a number to each element it stores, we refer to this number as the index. Assigning each item an index value makes it easy to access, delete, or replace values that are stored in the array. In computer science we start our index count at 0! The first element in the array will be assigned the index 0 and this index increases by 1 for each element that comes after. Let's take a look at an example:

```
var programmers = ["Ada", "Grace", "Katherine", "Roya"];
```

In this example, we have the array named `programmers` that contain the names of famous women programmers: `"Ada"`, `"Grace"`, `"Katherine"`, `"Roya"`. All together this array has a length of 4. The square brackets `[]` are used to show the start and end of the contents of the array.

```
      0           1           2           3  
["Ada", "Grace", "Katherine", "Roya"]
```

Let's say we wanted to access the name "Grace" from the array, how would we do that?

We can simply use the index associated with "Grace" to get this information. We would simply write the statement: `programmers[1]` and this will return the name "Grace". We first (1) type the **name** of the array we want to access, (2) add square brackets `[]`, then (3) put the index number we want to access inside of the square brackets, since we want "Grace" we put the number 1!

If we typed `programmers[3]`, this would return the name "Roya". If we typed `programmers[4]` we would get an **error!** Arrays are **0 indexed** which means that the index starts at 0. Even though there are 4 names in the list, the name in the end has an index that is *one less than the length*.

We can also use the index to add or change a value in the array. Let's say we wanted to add the name "Gitanjali" to our array. We can type the following syntax:

```
programmers[4] = "Gitanjali";
```

Step 9: Adding Your First Image and Location (10 mins)

Take a look at the `script.js` file again. Notice that we use the syntax shown before to create two different arrays, one named `images` and one named `locations`.

```
/* If you're feeling fancy you can add interactivity
   to your site with Javascript */

//This is an array that will hold the file names! Add at least 4 images to the images array.
var images = new Array();
images[0] = "";
//This is an array that holds the names of each location shown in your pictures.
//Make sure that the index of each location is the same as the index of the picture in the images array
var locations = new Array();
locations[0] = "";

//This is your starting index. First we start off at Home!
var index = 0;

//This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
function showImage() {
}
```

- `images` will be an array that holds the image addresses, or image links for all of the photos you want to include.
- `locations` will be an array that holds the geographical locations of each image (or description if you chose an alternative theme).

As we add each photo and its title to the `images` and `locations` array, we want to make sure that the index values are aligned in both arrays. This means that if I place an image of the Northern Lights in the `images` array at index 0, I also want to include the title text “Northern Lights” in the `locations` array at index 0.

To include a photo in the array, we put the **image URL address** instead.

- **If you are using images online you want to save the image URL address in your planning guide.** To get an image address follow this [video](#).
- **If you are using your own images, be sure that your images are saved *digitally* on your computer.** To upload images to Glitch watch this [video](#).

Now that you have the image URL addresses for your images, add them to your `images` array! Navigate back to the `script.js` file. Since this is your first image, let's add the image URL at index 0. Paste your image URL address between the “ ” on line 6 in your starter code.

```
1~ /* If you're feeling fancy you can add interactivity
2   to your site with Javascript */
3
4 //This is an array that will hold the file names! Add at least 4 images to the images array.
5 var images = new Array();
6 images[0] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FNorthern%20Lights.jpg?v=1586357262455";
7 //This is an array that holds the names of each location shown in your pictures.
8 //Make sure that the index of each location is the same as the index of the picture in the images array
9 var locations = new Array();
10 locations[0] = "";
11
12 //This is your starting index. First we start off at Home!
13 var index = 0;
14
15 //This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
16~ function showImage() {
17
18 }
19
```

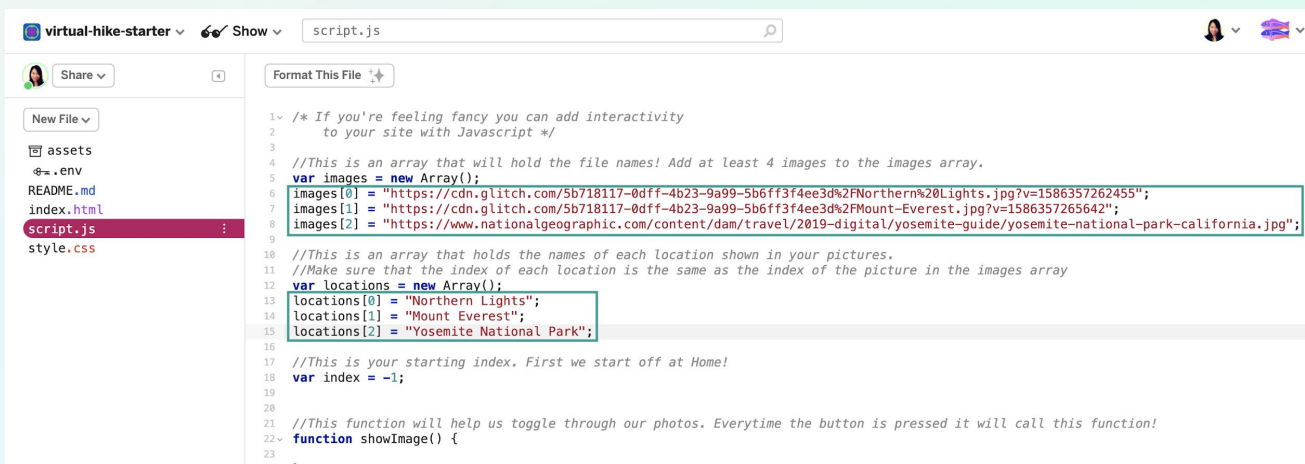
Step 9: Adding Your First Image and Location (Continued)

Next we want to add the location, or title, of the image just uploaded. Add the title of the image you added between the double quotation marks "" to the 0th index of the `locations` array.

```
1~ /* If you're feeling fancy you can add interactivity
2   to your site with Javascript */
3
4 //This is an array that will hold the file names! Add at least 4 images to the images array.
5 var images = new Array();
6 images[0] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FNorthern%20Lights.jpg?v=1586357262455";
7 //This is an array that holds the names of each location shown in your pictures.
8 //Make sure that the index of each location is the same as the index of the picture in the images array
9 var locations = new Array();
10 locations[0] = "Northern Lights";
11
12 //This is your starting index. First we start off at Home!
13 var index = 0;
14
15 //This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
16~ function showImage() {
17
18 }
19
```

Step 10: Adding More Images and Locations (10 mins)

Now that you have added your first pair of images and location title, it is time to add your remaining pictures! Follow the instructions in step 9 to add more images and text. Remember that as you add the image URL address to the `images` array, you must add its title in the `locations` array at the **same index**. Your code should look something similar to the image below after adding all of your images and locations.



```
1~ /* If you're feeling fancy you can add interactivity
2   to your site with Javascript */
3
4 //This is an array that will hold the file names! Add at least 4 images to the images array.
5 var images = new Array();
6 images[0] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FNorthern%20Lights.jpg?v=1586357262455";
7 images[1] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FMount-Everest.jpg?v=1586357265642";
8 images[2] = "https://www.nationalgeographic.com/content/dam/travel/2019-digital/yosemite-guide/yosemite-national-park-california.jpg";
9
10 //This is an array that holds the names of each location shown in your pictures.
11 //Make sure that the index of each location is the same as the index of the picture in the images array
12 var locations = new Array();
13 locations[0] = "Northern Lights";
14 locations[1] = "Mount Everest";
15 locations[2] = "Yosemite National Park";
16
17 //This is your starting index. First we start off at Home!
18 var index = -1;
19
20
21 //This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
22~ function showImage() {
23
24 }
```

Now you have added the images but nothing happens when you click the “Next” button! Although we now have a way to store these images, we haven’t set up a command to access them yet! Let’s change that.

Step 11: Intro to Functions (5 mins)

Let's first take a look at the button tag. Navigate to the `index.html` file and find this line of code.

```
<button type="button" onclick="showImage()">Next</button>
```

Notice that the attribute `onclick` is equal to `showImage()`. The attribute `onclick` is used to tell the computer what to do when the button is clicked. In this case, our button should call or run the **function** `showImage()`. The function `showImage()` is written in our Javascript file. But before we get into what `showImage()` does, let's discuss what a function is!

A **function** is a named section of a code that performs a set of instructions (or lines of code). For example, if you are told to "set the table", you already know that this means to put down a place setting on the table for the number of people in your home, then put down utensils (forks, spoons, knives, and/or chopsticks) for each setting, then place a napkin down for each setting. **Set the Table** is a **function** and the individual tasks needed to complete setting the table are included inside the function. Of course the instructions for setting the table might be different at your house. As programmers we can *specify* the tasks that are written inside a function.

Let's take a look at the `showImage()` function in the `script.js` file.

```
1~ /* If you're feeling fancy you can add interactivity
2   to your site with Javascript */
3
4 //This is an array that will hold the file names! Add at least 4 images to the images array.
5 var images = new Array();
6 images[0] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FNorthern%20Lights.jpg?v=1586357262455";
7 images[1] = "https://cdn.glitch.com/5b718117-0dff-4b23-9a99-5b6ff3f4ee3d%2FMount-Everest.jpg?v=1586357265642";
8 images[2] = "https://www.nationalgeographic.com/content/dam/travel/2019-digital/yosemite-guide/yosemite-national-park-california.jpg";
9
10 //This is an array that holds the names of each location shown in your pictures.
11 //Make sure that the index of each location is the same as the index of the picture in the images array
12 var locations = new Array();
13 locations[0] = "Northern Lights";
14 locations[1] = "Mount Everest";
15 locations[2] = "Yosemite National Park";
16
17 //This is your starting index. First we start off at Home!
18 var index = -1;
19
20
21 //This function will help us toggle through our photos. Everytime the button is pressed it will call this function!
22 function showImage() {
23
24 }
```

Using the **function** keyword lets the program know that you are declaring a function. That keyword is followed by the name of the function and any **parameters** – or pieces of information that need to be passed to the function as input – which are listed between the `()`. You might have noticed that there is nothing in between the parentheses in our function. That is because parameters are optional! All of the steps to be performed when the function is **called** are contained within a **code block** between the `{ }` that is punctuated with a final `;`.

We want function to do the following things:

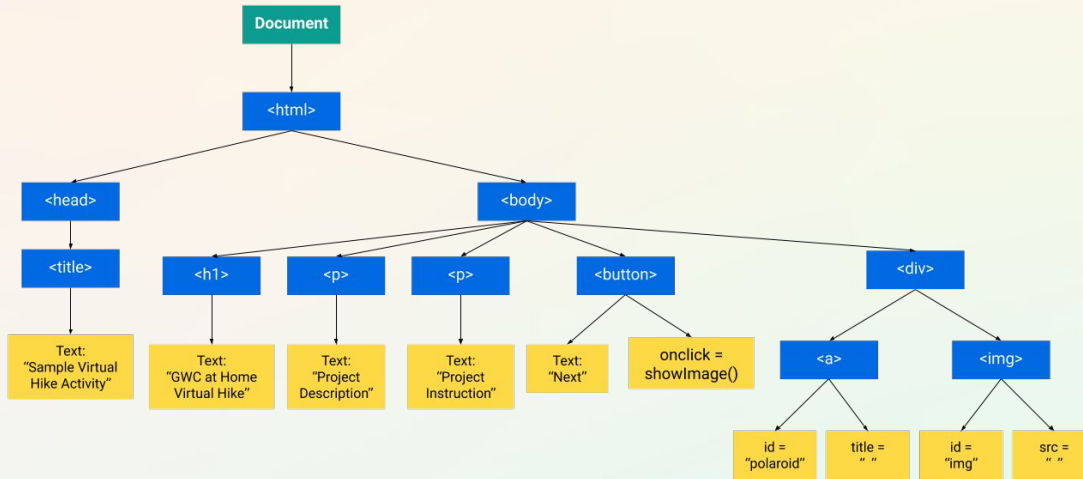
1. Update the image in the polaroid
2. Update the text in the polaroid
3. Move to the next image

In the next three steps, we will write code to add interactivity to our virtual hike.

Step 12: Update the Image in the Polaroid (10 mins)

In order to update the image in the polaroid, we need to use the DOM! A website uses a **DOM** (Document Object Model) to organize all of the content of the webpage. The DOM reflects the structure that is written in our HTML file! Here is the DOM of our website so far!

```
<div class="polaroid">
  <a id="polaroid" href="#" title="">
    <img id="img" src="" />
  </a>
</div>
```



You may notice that the **blue** boxes represent the tags in the HTML file. The information represented in the **yellow** boxes are **attributes** for each type of element. Attributes are information that are unique to each element. Just like how your hair color, eye color, and name are all attributes of you! In order for our Javascript file to communicate with the HTML skeleton, we use the DOM to help connect the elements in the `script.js` and `index.html` files. You could say the DOM provides a map for our JavaScript code to locate the elements we want to change. Using an element's **id** is the easiest way to call an element in your HTML file. Notice that the element referring to the polaroid photo was given the id name **polaroid** and the image inside the polaroid photo has a tag `` and id name **img**.

Note: The next lines of code should be written *inside* of the `showImage()` function. All code should be written between the `{}`.

1. Use the DOM to create a reference to the image tag with the id "img".

We need to tell our JavaScript file which element in the HTML file we want to change. This is where the DOM really shines.

```
var img = document.getElementById("img");
```

We use the `document` object to tell JavaScript that we want to take an action on *our* webpage. The `getElementById()` method is used to find an element on our web page by a tag's id. We use the image tag id: `"img"` to reference the `` tag. Finally, we store the reference to the image tag in a variable we named `img`.

Step 12: Update the Image in the Polaroid (Continued)

2. Update the image source to one of the pictures in the `images` array. We already created a variable called `index` that starts at 0. This will keep track of the image you want to show!

```
img.src = images[index];
```

We have our `img` object that we found through the DOM. Now we want to attach a source attribute to it by using the keyword `src`. Here we use `images[index]` to get the picture we want using the index value that is stored in the variable `index`. We use the equal sign `=` to assign the image source to the new picture!

```
//This function will help us toggle through our p  
function showImage() {  
  var img = document.getElementById("img");  
  img.src = images[index];  
}
```

3. Test that your code works! On your live website, press the “Next” button! Your first image should show up. If not, double check that you have semicolons at the end of every line of code inside the function and that your images have the correct URL address.

Step 13: Update the Text in the Polaroid (5 mins)

Now we will continue to code *inside* of the `showImage()` function, adding code to update the text shown with the image. Remember all of these steps should be written between the `{}`.

1. Create a reference using DOM to the polaroid tag, using the id “`polaroid`”.

```
var polaroid = document.getElementById("polaroid");
```

Here we call the DOM by using the keyword `document` and reference the image tag by its id, “`polaroid`”. Since we are using the id to get the tag, we use the method `getElementById()`. Finally we store this in a variable we named `polaroid`.

2. Update the title to one of the locations in the `locations` array. We already created a variable called `index` that starts at 0. This is used to help you keep track of which description you want to show!

```
polaroid.title = locations[index];
```

Here we use `locations[index]` to get the picture we want using the index value that is stored in the variable `index`. We want to make sure to use the same index to get the correct location for each picture. Next we get the title attribute of the image by using the property `title`. We use the equal sign `=` to assign the text to the right location!

```
function showImage() {  
  var img = document.getElementById("img");  
  img.src = images[index];  
  
  var polaroid = document.getElementById("polaroid");  
  polaroid.title = locations[index];  
}
```

Step 14: Move to the Next Image (2 mins)

Everytime you click the “Next” button an image should appear, but right now only one of your images shows up, why? In order to get the image we want, we used the variable `index` to get the data stored in our arrays. So far, the `index` is set to 0 but we never changed it!

After we update the image and title, we want to increase the index value by 1 so we go through each of the photos and locations.

```
index = index + 1;
```

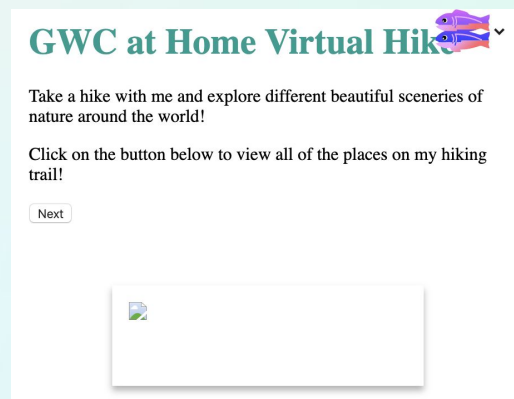
Here we re-assign the value `index` to just be the previous value plus 1. We want to add this as the **last** statement in the function.

```
function showImage() {  
  var img = document.getElementById("img");  
  img.src = images[index];  
  
  var polaroid = document.getElementById("polaroid");  
  polaroid.title = locations[index];  
  
  index = index+1;  
}
```

Let’s quickly test that our code works! On your live website, now click the “Next” button. Everytime you click the “Next” button you should see the next picture in your slide show.

Step 15: Whoops! Going Too Far! (5 mins)

You may have noticed that you get an error if you keep clicking “Next” past your last image.



Since we increase our `index` value each time we click the “Next” button, eventually our program tries to get data in our array that doesn’t exist! For that we need some way to capture if our `index` is too large. A **conditional statement** checks if a set of rules (or statement) are met and then decides which actions are performed if the rules or statement is true or false. In our case we want to know if the `index` is too large - in other words, if it is greater than the length/size of the array.

Step 15: Whoops! Going Too Far! (Continued)

Before we write out our conditional statement we want to plan what are the cases.

1. If `index` is not too large (less than the length of the array)
Then we can increase our index by 1 like before!
2. If `index` is too large
Then we want to just keep `index` as is. By not changing the value of `index` we just leave the last picture visible to our users and do not advance our images further.

But how do we determine if the `index` is too large? We can tell by the size of the array! Think back on what is the last index value in the array? How does it relate to the size? In fact, the last index of the array is always one *less* than the size/length of the array. This is because arrays are 0 index! We know that the last index can be represented by `images.length - 1`.

When writing conditionals, we use the keywords `if` and `else` to break up our rules set and use the code block `{ }` to let the computer know which lines of code to execute if the condition is true!

We want to replace our previous statement `index = index + 1;` with the following lines of code to capture when index might be too large!

```
function showImage() {  
  var img = document.getElementById("img");  
  img.src = images[index];  
  
  var polaroid = document.getElementById("polaroid");  
  polaroid.title = locations[index];  
  
  if (index < images.length - 1) {  
    index = index + 1;  
  }  
}
```

In order to see if `index` is too large, we compare it to the `length` of the array. Since we said the last index in the array is always one less than the length, we used this value in our condition (or rule). We know that if the index isn't too large, then we will increase the value of `index` by 1. We don't add more code to capture the second case if `index` is too large because we don't really want to update the value of `index` if that is the case. In other words, if `index` is too large we do nothing!

Now take a couple minutes to test your code. Be sure to **click** the Next button multiple times to test if your button stops at the last image!

Step 16: Extensions (5-20 mins)

Adding Some Fancy Fonts (5-10 mins)

In this project we set the font to “Roboto”, but you can change this for your own project! You can easily import and use fancy fonts in your website from [Google Fonts](#). Watch this [video](#) to learn how to change the fonts on your website!

Change Your Website’s Background Color (5-10 mins)

Let’s change the background colors with more interesting colors! Colors are given numerical values to make it easier for the computer to distinguish different colors. They can be represented by their hex, rgb, or hsl value. We recommend choosing one type for all colors to be consistent. W3Schools has a great [color picker](#) to help you find the values of colors you might want to add to your website! Want to add a cool gradient effect like our sample project? Use this [website](#) to help plan and generate the CSS code. Watch this [video](#) for detailed instructions on changing colors.

Add More CSS Style Rules (10-20 mins)

You can do so much with CSS to make your website unique! We compiled a list of resources that you can browse for more tips on styling your website with CSS.

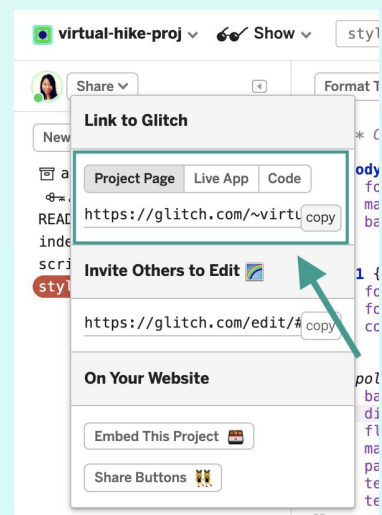
- W3Schools has a great [reference](#) section for CSS.
- Mozilla has a wealth of helpful information in their CSS module, including:
 - [How to style text](#)
 - [CSS Layout](#)
 - [How to use CSS to solve common problems](#)
 - [Debugging your CSS](#)

Step 18: Share Your Girls Who Code at Home Project! (5 mins)

We would love to see your virtual hike projects! Don’t forget to tag @girlswhocode #codefromhome and you may even get featured on our account!

To Share your Glitch Project:

1. Click the **Share** button on the top left next to your profile icon.
2. You may choose to share your Project Page link (which will allow others to see your code) or just the Live App. Click on whichever option you would like to share and copy the link provided.



Virtual Hike Project Planning Worksheet

Project Overview Planning

Theme: What topic will be covered with your virtual hike?

Audience: Who is this website going to serve? What group of people will be interested in your product?

Goal: What do you want your website to accomplish? Why is this of interest to your audience?

Image Planning

Choose at least three images that you want featured on your virtual hike. Be sure to consider the **order** of your images. Fill out the table below and include the image, the image URL address, and title.

Number	Image	Image Source <i>(Indicate if this will be your own photo or a photo from another website. If using a photo from a website, document a link to the site.)</i>	Title <i>(What text do you want to display to your audience with this picture?)</i>
0.			
1.			
2.			
3.			

Image Planning (Continued)

Number	Image	Image Source <i>(Indicate if this will be your own photo or a photo from another website. If using a photo from a website, document a link to the site.)</i>	Title <i>(What text do you want to display to your audience with this picture?)</i>
4.			
5.			
6.			
7.			