Outline

Checkpoint 4

Web page template

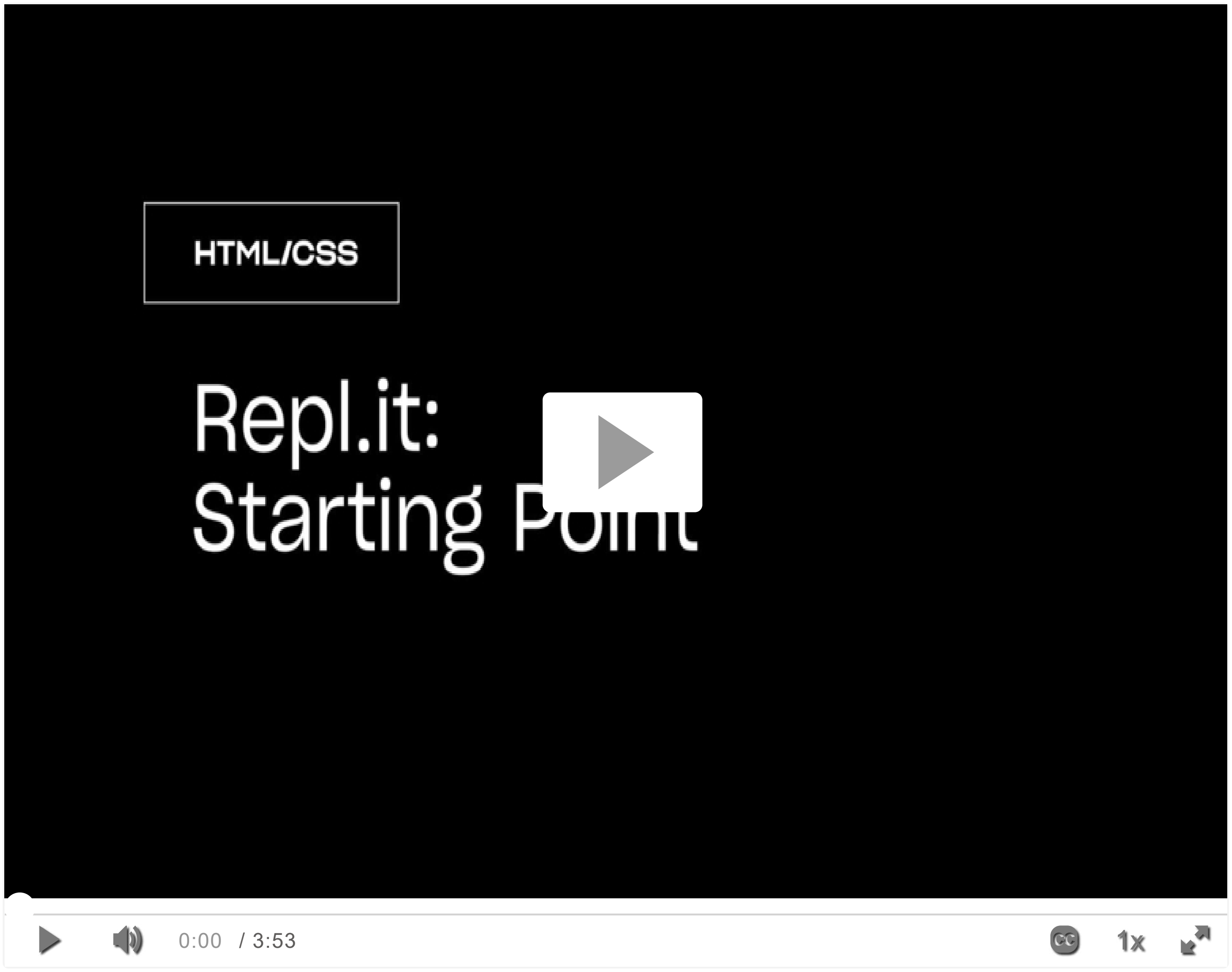
In this module, you've focused on the basics of HTML and CSS code to get yourself up to speed. Now, you're ready to learn about the supporting code that is needed to build a complete web page. In this checkpoint, you'll explore the core HTML elements that are required within every HTML web page. As you'll see, this is also referred to as a *web page starting-point template*.

By the end of this checkpoint, you will be able to do the following:

- Explain the core HTML elements that are part of every HTML web page
- Describe the organization of a multipage website

Creating a new web project in Repl.it

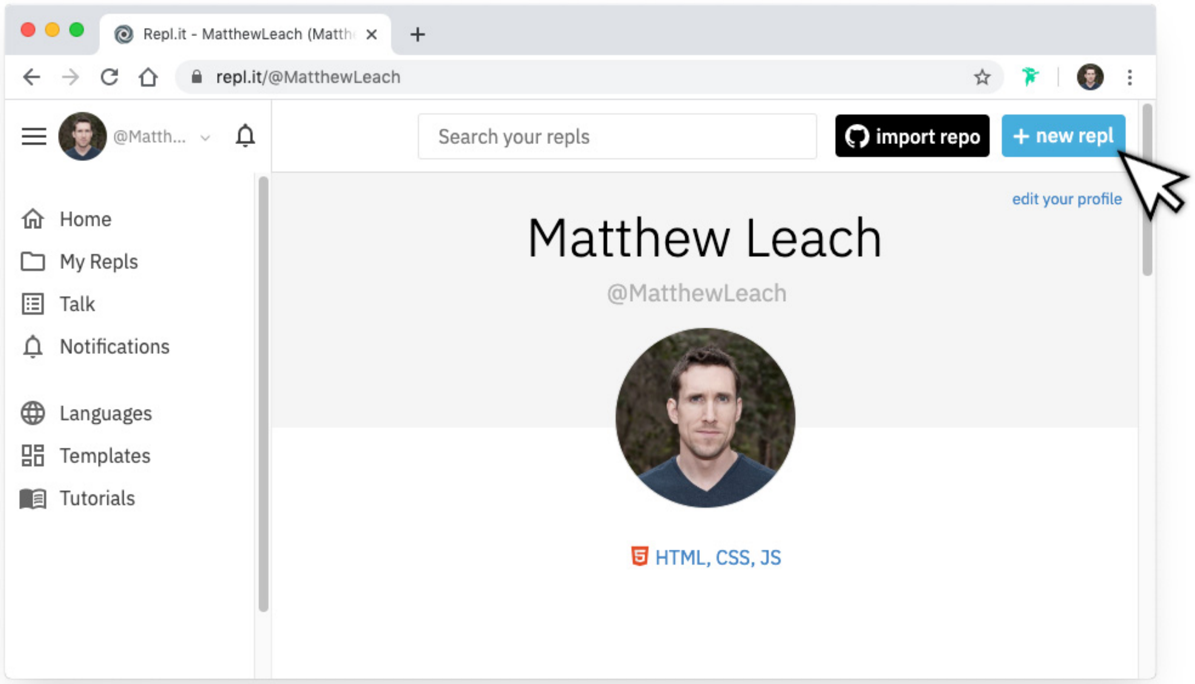
Thus far, you've been working with REPLs that have been built for you. But if you were to create a new HTML, CSS, and JS project in R[illegible], you'd notice that there's a lot of additional code in the HTML page to begin with. Watch the video below to get a better understanding of this foundational, or *starting-point*, HTML code.



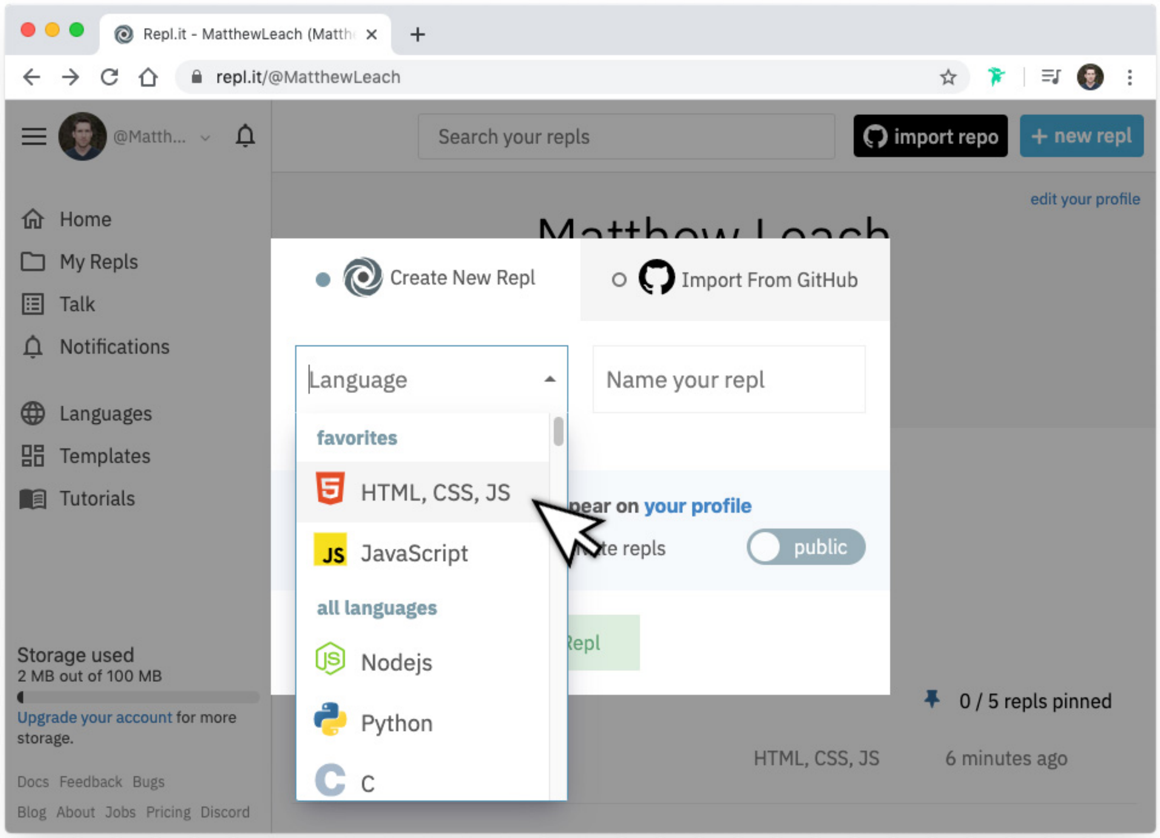
Outline

It's your turn to create a new project in Repl.it. This will let you dig into the code to properly understand how it works for all HTML, CSS, and JS projects. Follow the directions below.

1. Start by creating a new REPL by clicking the + **new repl** icon.

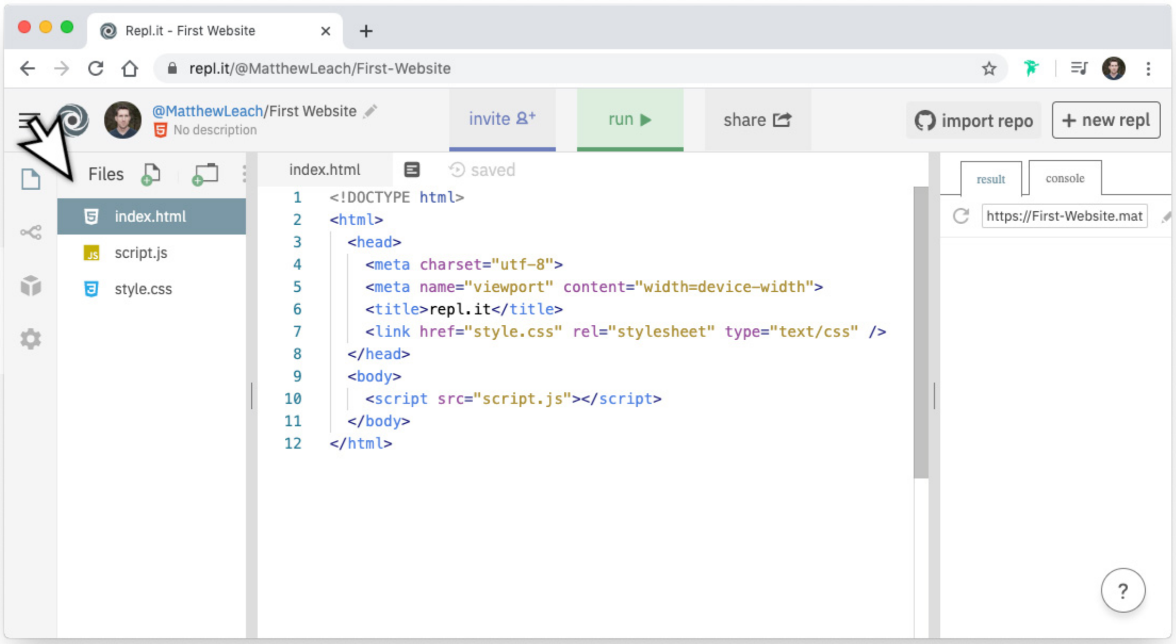


2. Then, select the **HTML, CSS, JS** framework option from the **Language** drop-down menu. Give your REPL project a name, such as *First Website*.



Outline

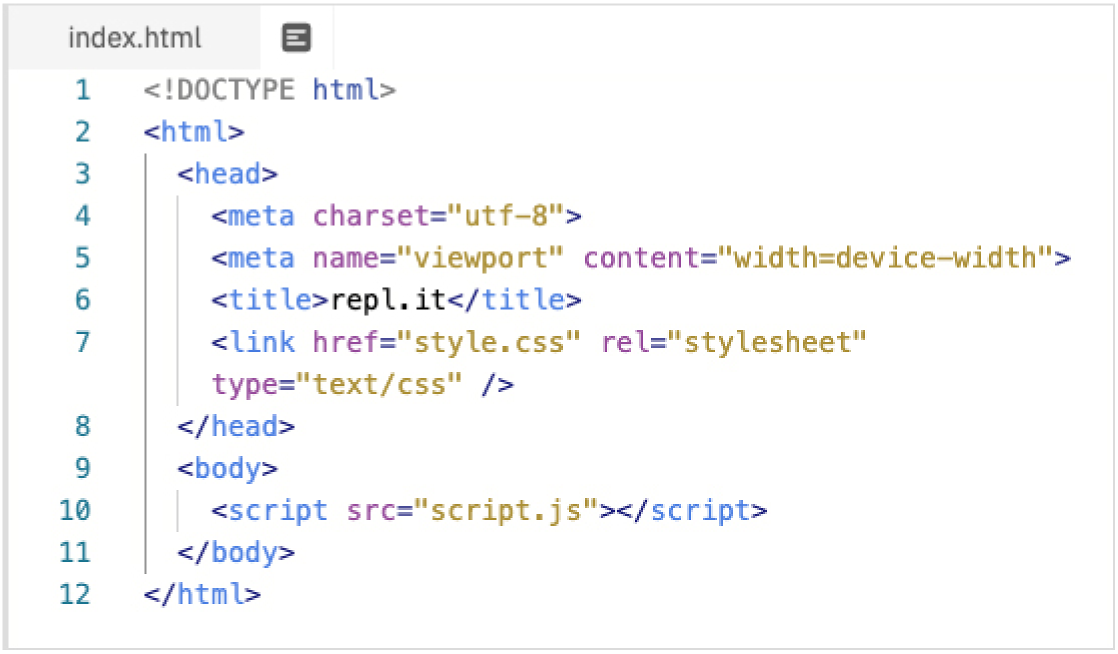
3. If you click **Files**, you'll see that your new project in Repl.it provides you with three files in your project directory:
   1. `index.html`: All your HTML code will be written here.
   2. `style.css`: All your CSS code will be written here.
   3. `script.js`: All your JavaScript code will be written here. (However, this step comes later.)



```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>repl.it</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <script src="script.js"></script>
  </body>
</html>
```

The index

A nice feature of Repl.it is that it provides the basic *framework* for a website built with HTML, CSS, and JavaScript. As mentioned above, this is useful because this code is used by every HTML page on the internet. In Repl.it, this foundational code can be found in the `index.html` file, or the index. This will help you get started.



index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width">
    <title>repl.it</title>
    <link href="style.css" rel="stylesheet"
    type="text/css" />
  </head>
  <body>
    <script src="script.js"></script>
  </body>
</html>
```

Take a moment to explore and become familiar with the different parts of this core code. The pieces of the index are defined below.

DOCTYPE

```
<!DOCTYPE html>
```

As the name `DOCTYPE` suggests, this line of code declares this document as an HTML5 web page. This tag is the first line of HTML code, and it is *required* to be the first line of code in every HTML5 web page. It's important to note that this is the only HTML tag written in capital letters—all other HTML tags should be written in lowercase.

HTML

```
<html>
  ...
</html>
```

The `<html>` element identifies and contains the HTML code in HTML web pages. The opening and closing tags, `<html>` and `</html>`, wrap around all the HTML code that is related to the web page.

Head

```
<head>
  ...
</head>
```

Every HTML web page has a `<head>` element. The `<head>` contains all the important information web browsers and search engines need regarding a web page. In a way, this element is the brains of the web page. Although the `<head>` holds vital information about the web page, nothing within it is displayed on the actual web page. It'll include several of the components you'll read about below, like meta tags, the `<title>` element, and the `<link>` element.

Outline

Meta tags

```
<meta charset="utf-8" />
```

The *meta tag* holds important information related to the data within the web page. The meta tag above is declaring the `charset`, or *character set*, to tell browsers how to process the characters and code within the file. What do you think this next meta tag does?

```
<meta name="viewport" content="width=device-wid      />
```

The meta tag in this example sets the width of the web page to follow the screen-width of whichever device a user is looking at. For instance, the width of a computer monitor will be larger than the width of a phone screen, and this meta tag ensures that the web page displays properly across both devices. This meta tag will be particularly useful when you start building web pages that are responsive to different screen-widths.

Title

```
<title>repl.it</title>
```

The `<title>` element defines the web page title, like `repl.it` above. You can see this title in the web page tab at the top of a web browser, but you won't be able to see it display on the web page itself. In fact, it might be a slightly different (or extended) version of the displayed name or title of the web page. The `<title>` is also the name that will be used when a web page is bookmarked on a web browser.

Link

```
<link href="style.css" rel="stylesheet" type="text/cs
```

The `<link>` element is used to connect and reference resources on the internet. In fact, it should seem familiar. You've used it in previous Repl.it projects to connect an external `style.css` file with the HTML web page.

Outline

Using an external CSS page can make things easier. With a separate `style.css` file, you can keep all the core CSS code written in one place, and then multiple HTML pages can reference that CSS file. That way, you can update the CSS code on all your web pages simultaneously and consistently simply by changing that one CSS file. And although this link doesn't need to be in the `<head>` element of the HTML page, it is a best practice to put it there.

The attribute `rel` stands for *relationship*—the relationship between the HTML document and the CSS file. The `type` specifies the media type of the linked file, and in this case, it's labeled as `text/css`. However, it's worth noting that the `type` attribute is no longer required in web pages, but it tends to stick around because, well, it doesn't hurt to have it.

Outline

Body

```
<body>
  ...
</body>
```

The `<body>` element contains all the HTML code for the text, images, links, and containers used for the web page structure. All of the content-based HTML code you've worked with in previous checkpoints would be placed within the `<body>`.

Script

```
<script src="script.js"></script>
```

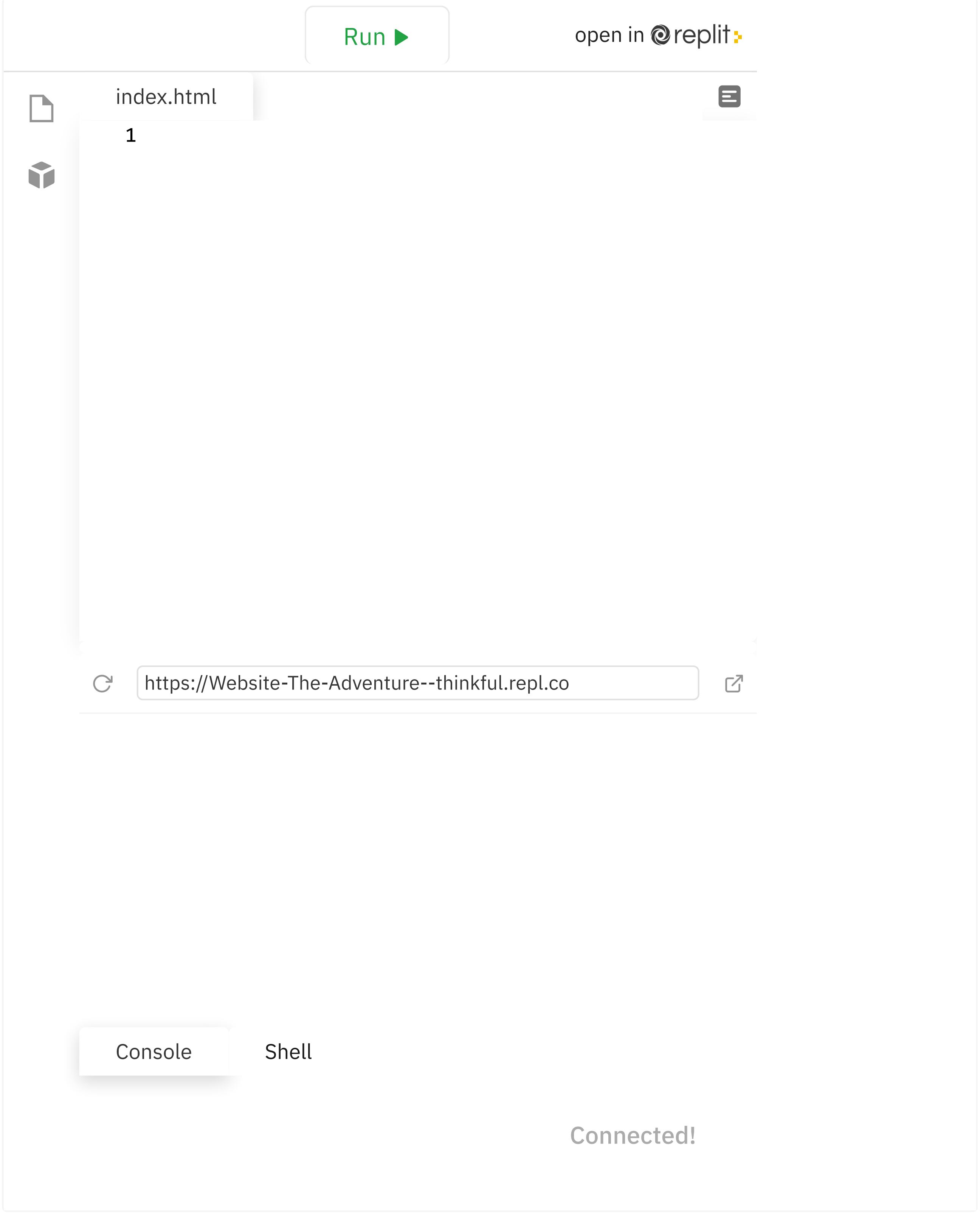
Similar to how the `<link>` element connects the HTML code to the CSS file, this `<script>` element pulls in the JavaScript code that is written in the `script.js` file. And just like it's a best practice to keep your CSS file in the `<head>`, it's a good idea to keep the `script` line of code at the bottom of the HTML page, below the other HTML code but just before the closing tag of the HTML element. This will allow the web page to load first (and faster) because the (heftier) JavaScript code will load last.

You won't be adding JavaScript to your HTML and CSS projects just yet. For now, you can just leave this where it is (or you can delete it).

Web project

Take a moment to look at what a multipage website looks like using Repl.it.

Outline



Outline

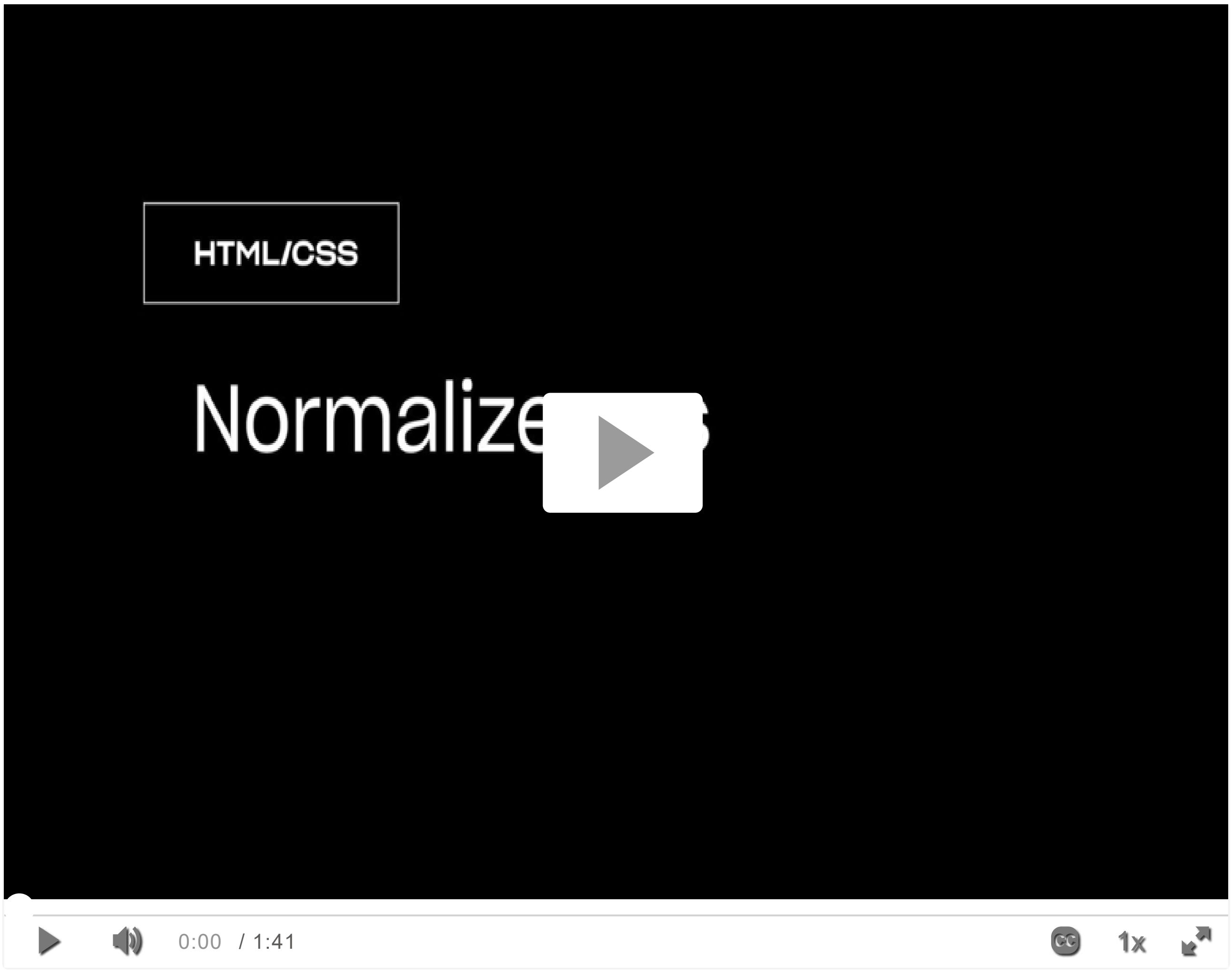
As you can see, the `index.html`, `style.css`, and `script.js` files are all still there. But in this project, there are three additional HTML files, as well as an *images* folder. The images are all organized w[illegible] subfolders so that they're easier to find.

Both the HTML code and CSS code are still pretty advanced at this stage. But don't worry! All the checkpoints within these modules will help give you a better foundation for writing, and reading, HTML and CSS. Believe it or not, within the first week of the course, you'll be coding and designing websites just like this one. And what's more, you'll understand what every line of code actually does!

The `normalize.css` file

Believe it or not, but web browsers have their own default styles for displaying HTML, which can lead to some unexpected or problematic changes to your website. So an `<h1>` element on one page may look slightly different in Chrome and Firefox, even if they are the same code. Fortunately, there's a solution: `normalize.css`. Watch the video below to learn more.

Outline



Outline

Nicolas Gallagher's normalize.css is a CSS library that sets all HTML elements to display consistently across all supported web browsers. It's a small file that styles and formats headings, paragraphs, blockquotes, and other common HTML elements so that they appear identical (or very similar) on Chrome, Firefox, Safari, etc.

Although you can download the `normalize.css` file and manage it locally, it tends to work best if you link to the file from a *content delivery network*, or CDN. It's a good idea to load the `normalize.css` file first in the code, before applying your own style rules using your own `style.css` file. You can see this in the code sample below.

```
<head>
  <meta charset="utf-8" />
```

```
    <meta name="viewport" content="width=device-width"
    <title>Template: Starting Point</title>
    <link
      href="https://cdnjs.cloudflare.com/ajax/libs/norm
      rel="stylesheet"
      type="text/css"
    />
    <link href="style.css" rel="stylesheet" type="text/
  </head>
```

Why is it better to reference `normalize.css` before your `style.css` file? Well, CSS will apply styles in the order that they appear in the code. This means that if you add the `normalize.css` file last, the styles in the `normalize.css` file may actually overwrite your styles! This concept will be covered in more depth in a later checkpoint. At this point, it's just important to remember which file to put in first.

Outline

One other thing to keep in mind: the only difference between a `normalize.css` file and a `normalize.min.css` file is that the `.min` version has all the spaces and visual formatting removed. This makes the code harder to read, but it creates a smaller file size—which is very helpful when millions of websites are referencing it.

A note on `index.html` files

Return for a moment to the ever-important `index.html` file. The filename `index.html` is significant. When a web browser opens a folder with multiple HTML files, it will always display the `index.html` page first, without that page needing to be referenced. The video below provides a bit more information.

Outline

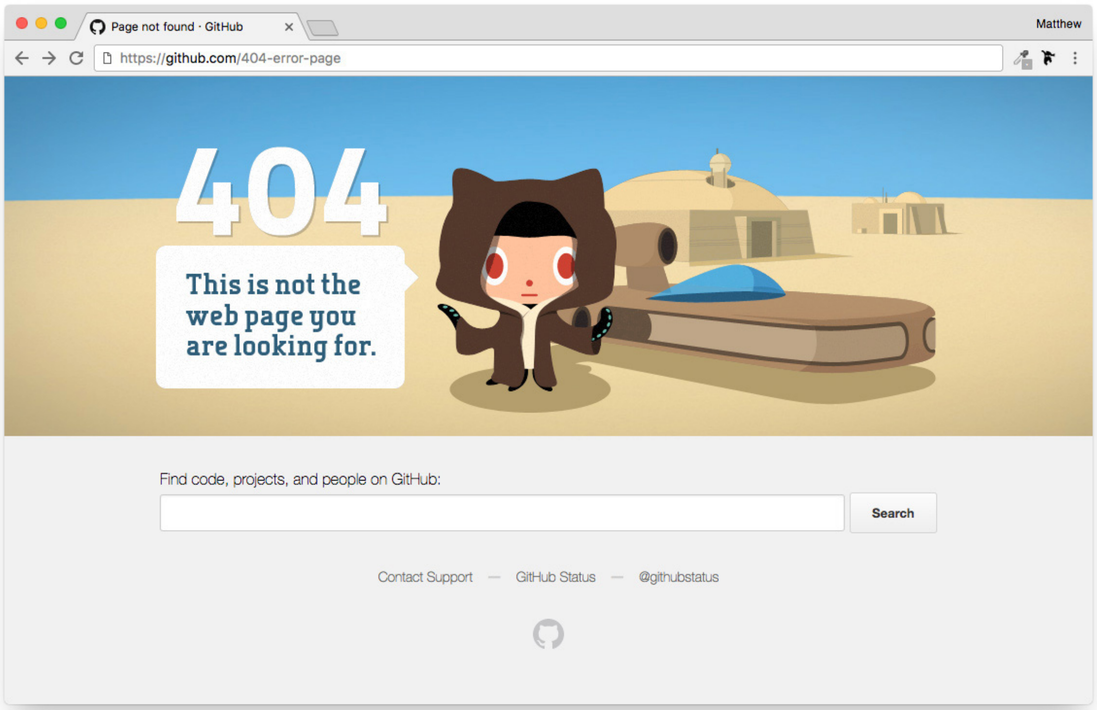
As you've seen, Repl.it tries to reinforce the importance of the `index.html` file by requiring you to have it in the main directory and prohibiting you from renaming or deleting it. Keep the following in mind when working with `index.html` files:

- Every website's home page will be named `index.html`.

- Naturally, working with multiple projects that all have the web page `index.html` can be confusing. This is why making a logical file organization, with proper project folder names, is essential.
- The `index.html` file needs to be written in lowercase letters. Web file names tend to only use lowercase letters to prevent simple mistakes.

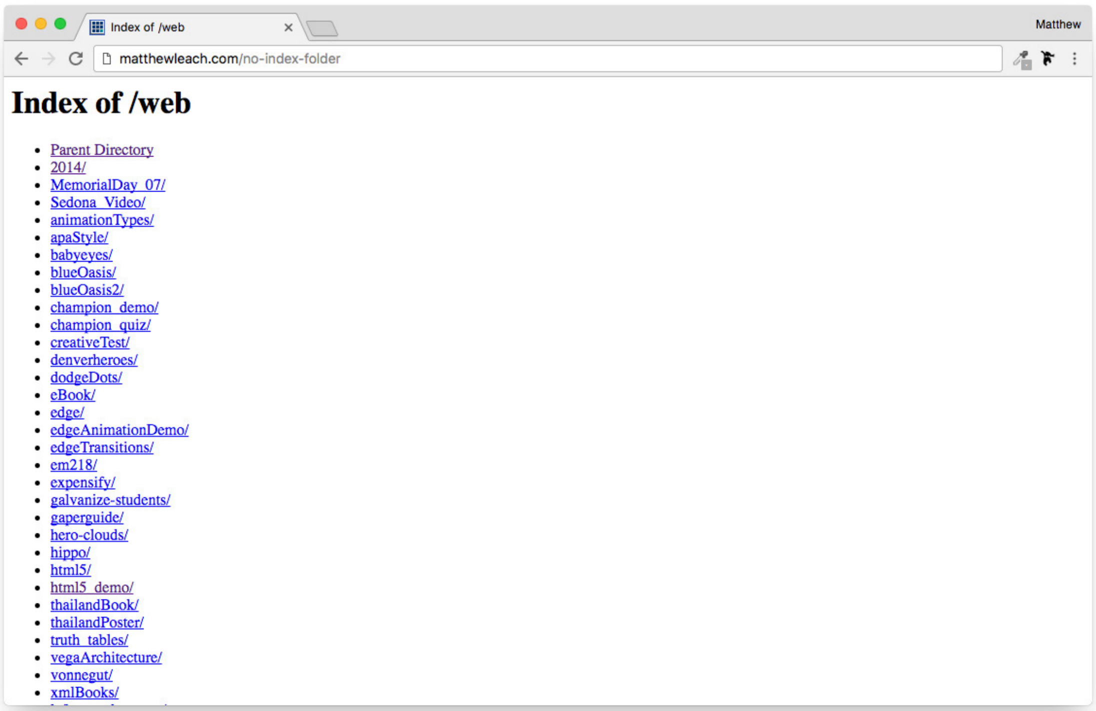
But imagine if an `index.html` file is missing from a web directory. In this case, two things could happen, based on the web server's preferences:

1. The web page opens a **404: File not found** error, as seen below.


Page not found · GitHub
Matthew
https://github.com/404-error-page
404
This is not the web page you are looking for.
Find code, projects, and people on GitHub:
Search
Contact Support — GitHub Status — @githubstatus

2. The web page displays a list of all the files in that directory, as seen below. However, this can be dangerous, as any file from thi directory can be viewed and then downloaded. In this situ n,

the web page display will depend on your web hosting provider and how they handle directories without `index.html` files.



Index of /web

matthewleach.com/no-index-folder

Index of /web

- Parent Directory
- 2014/
- MemorialDay_07/
- Sedona_Video/
- animationTypes/
- apaStyle/
- babyeyes/
- blueOasis/
- blueOasis2/
- champion_demo/
- champion_quiz/
- creativeTest/
- denverheroes/
- dodgeDots/
- eBook/
- edge/
- edgeAnimationDemo/
- edgeTransitions/
- em218/
- expensify/
- galvanize-students/
- gaperguide/
- hero-clouds/
- hippo/
- html5/
- html5_demo/
- thailandBook/
- thailandPoster/
- truth_tables/
- vegaArchitecture/
- vonnegut/
- xmlBooks/

Outline

Tips for naming folders and files

Now that you've got a bit more background, it's time to dive into some tips for naming files. Watch the video below to learn more.

Take a moment to carefully review these filenames and folder names. Can you determine why some of these names are good and others are bad?

Good	Bad
index.html	Index.html
about.html	145724.html
puppy-blog (folder)	puppy_blog (folder)
best-puppy-lessons.html	best puppy lessons.html
puppy-sleeping.png	Image-345.png

The following tips are best practices for naming files and folders. Using these tips will help you be a more professional, and more deliberate, developer.

Outline

- **Avoid spaces in your filenames.** Never use spaces in any web file or folder names. Every space will be converted into a *%20*, which can make the names of your folders and files unreadable. For instance, the filename `bad web page.html` will look like this when viewed online: `http://www.example.com/bad%20web%20page.html`. Without spaces, that URL could be much cleaner and clearer.
- **Use hyphens to separate your words.** Use hyphens rather than spaces. Turns out, *search engine optimization*, or SEO, appreciates hyphens. (And SEO helps determine how websites get ranked in users' search queries.) Here's a good use of hyphens: `http://www.website.com/sub-folder/file-name.html`.
- **Avoid underscores to separate your words.** Underscores, on the other hand, are not preferable. SEO considers filenames with underscores as one full name, so underscores are not as good as hyphens when it comes to search engine ranking. Here's an example of what not to do: `http://www.website.com/sub_folder/really_bad_topic.html`.
- **Use only lowercase letters.** Web servers are case sensitive. For this reason, it's important to write both filenames and folder names in lowercase letters; this prevents confusion regarding what is or isn't capitalized.
- **Use descriptive words.** Use specific, descriptive words to explain what the file is, does, or shows. Using good names helps [illegible] you organized, too. Consider these examples:

    - **Bad:** `image1.jpg`
    - **Good:** `black-lab-puppy.jpg`
    - **Bad:** `page.html`
    - **Good:** `contact.html`
  - **Do not use special characters:** Rely on the letters of the alphabet (A through Z), the numbers 0 through 9, and hyphens (-). But web servers will not link properly to filenames or folders with special symbols or characters. For that reason, don't use these: ; / ? : @ = + \ $, < > # % " { } | \ ^ [ ].

Drill: Files and frameworks

The code sample in the REPL below already has various web files and images in it. Your goal is to rename each file and image file with a better, more appropriate name, if necessary.

Outline

When you have completed your practice project, feel free to compare your code with this completed one.

Filename Practice: Complete

Optional practice

This practice is not required, but it will benefit you. It will get you writing code on your own. Although the HTML, CSS, and JS template in Repl.it is an easy and helpful starting point, this is an *optional* opportunity for you to code the page and organize the files from scratch.

Why is that important? Writing code from scratch can help you focus on remembering what each line does. Plus, additional practice typing code within a code editor (especially on a fairly simple project) can help you feel more comfortable with all this new material.

If you'd like to take on this opportunity, create a new HTML project and delete all the files and content. Then, start once again. Your simple framework should include the following:

- DOCTYPE
- HTML
- Head
- Meta tags
- Page title
- Link to `normalize.css`
- Link to `style.css`
- Body

Outline

Your work

03.04.21 ✓

Completed

Next checkpoint

How would you rate this content?

Report a typo or other issue

Outline

Go to Overview