Checkpoint 5

Links and paths

By now, you've learned a bit about how web pages reference other web-based resources. Now, you'll dive a little deeper into this process. Turns out, you use the same technique to incorporate links to other web pages as you would to include images.

At its core, this technique is about creating *paths* to web pages and images. Understanding how paths work—and how to create them—is an essential skill for web developers.

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

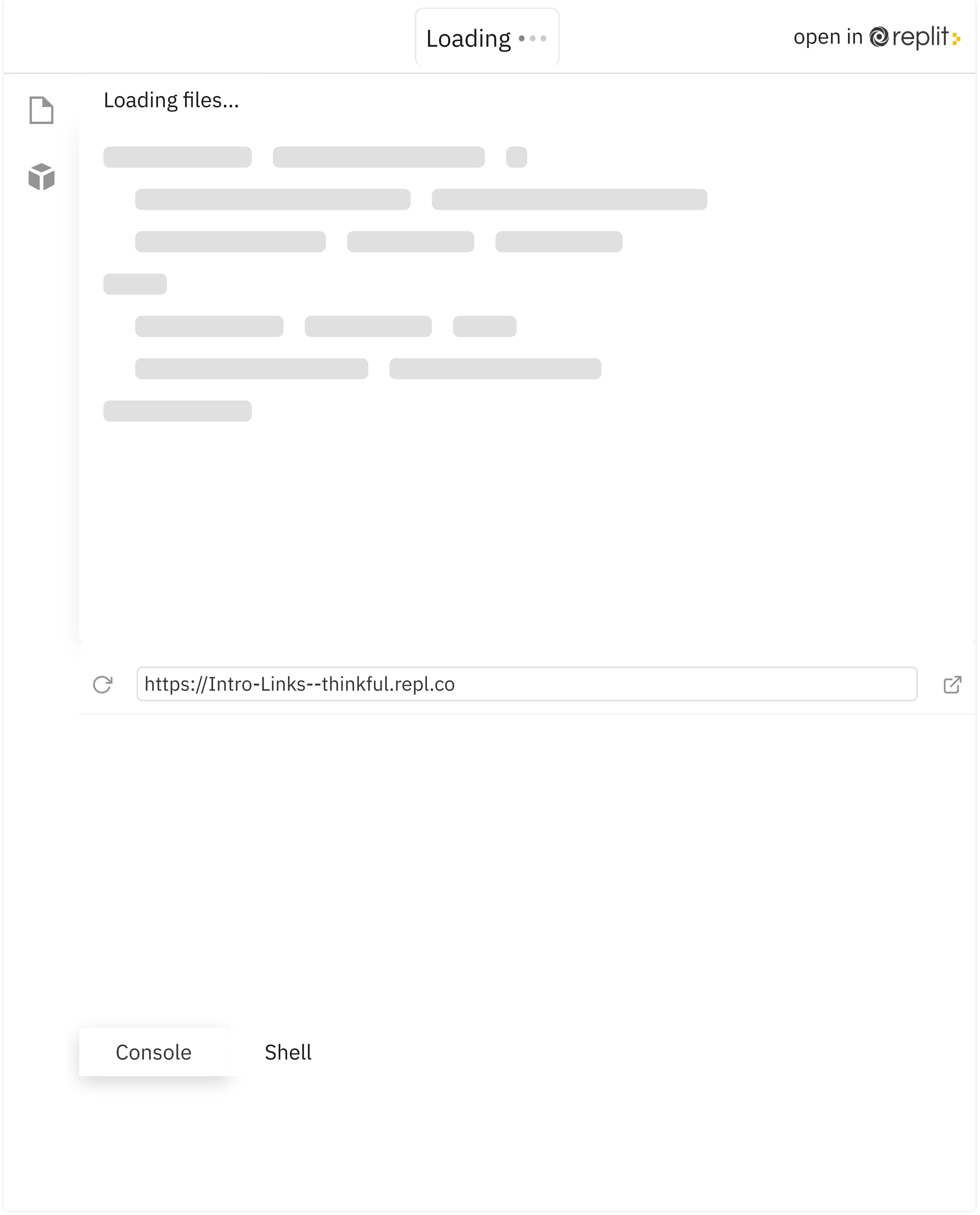
- Include proper paths for links and images in a website

Intro to links

Before diving in, take a moment to review the code sample in the REPL below. As you review, try to answer the following questions. You don't need to look up the answers or read ahead; this is just an opportunity for you to practice applying what you've learned about coding [illegible] to a new and unfamiliar situation. Give it your best shot.

- What are the opening and closing tags for links?
- What code is used to create a text link?
- What code is used to create an image link?
- What is an `href` in a link?

Outline



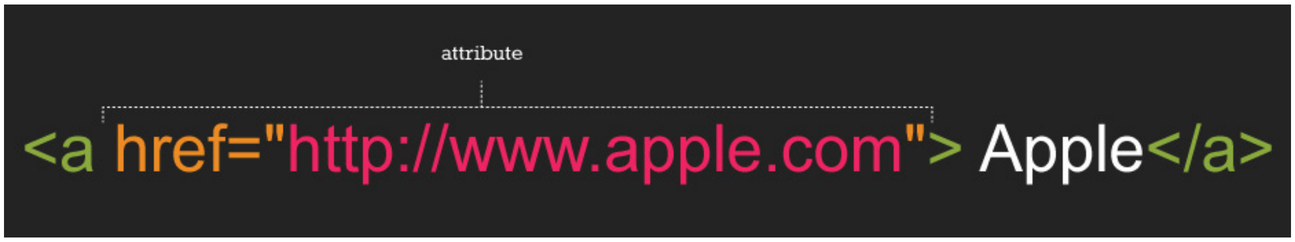
Outline

Link details: The basics

When reviewing the code above, what patterns did you see? Di
anything seem familiar from previous checkpoints?

One thing that you might've noticed is the `<a>` element. When working with HTML links, you'll use *anchor* elements, represented by the `<a>`, to wrap around text or images to create a link. The opening tag `<a>` starts at the beginning of the text or before an image element, and the closing tag `</a>` is used at the end of the text or after the image element. Any text or image that is properly referenced in between these anchor tags will be a *clickable* element on the web page.

Inside the `<a>` element, there is an attribute called `href`. The `href`, which stands for *hypertext reference*, refers to the web page that will open when the link is clicked. The web page that opens is based on the specific URL path that is provided in the `href`—in the example below, this is the Apple URL.



```
<a href="http://www.apple.com"> Apple</a>
```

A *URL path* directs the computer to the precise location of an asset or file, with each necessary step in the path separated by a forward slash `/`. Although some paths, like the one above, won't need multiple steps!

Now that you've got the gist, you're ready to go deeper. In your web page, you might want to link to resources located either *outside* your website or *within* your website. These two situations require two different kinds of links: absolute and relative links.

Absolute link paths

Imagine that you're building a web page, and you want to link to content that is located outside your website, on another website. In this case, you must use *absolute links*. Absolute links use the full URL path, which means including `http://`, and follow the URL path structure outlined below, with each component separated by a `/`.

1. **Web server name**: This is the `www.webserver.com` in the example below.
2. **Names of folder or folders**: The path could require multiple folders and subfolders. This is seen in the `/folder/subfolder/` below.
3. **Filename**: Finally, the path ends with the filename itself: `/filename.html`.

```
<a href="http://www.webserver.com/folder/subfolder/fi
```

Outline

Check out the sample paths below. What do they link to?

- An absolute link path: http://www.adobe.com/products/photoshop.html
- An absolute image path: https://images.unsplash.com/photo-1524338198850-8a2ff63aaceb

Relative link paths

Imagine, on the other hand, that you want to link to content that is located *within* your website. In this case, you should use *relative links*. Relative links use a path that connects one file to another file th on the same server. They follow the structure outlined below (which, as you might've noticed, is similar to the one used for absolute links).

1. **Names of folder or folders**: The path could require multiple folders and subfolders. This is seen in the `/folder/subfolder/` part of the code below.

2. **Filename**: The path ends with the filename itself: `/filename.html`.

```
<a href="folder/subfolder/filename.html">Link</a>
```

Relative link paths can *call*, or retrieve, files from within the same directory, or they can follow a more complicated route into various folders and subfolders until they connect the path to the web page or image file. Like with absolute links, the `/` tracks the route into multiple locations to find the desired file, even within the same website. However, relative links work differently than absolute paths. Relative links are different from absolute paths in the following ways:

- With relative paths, the web page and the referencing file must be within the same website structure.
- With relative paths, the paths are dependent on where the file is located in reference to the web page.

Here are some examples of relative link paths:

- `about.html`
- `contact.html`

- `portfolio/project1/index.html`
- `portfolio/project2/index.html`

Here are some examples of relative image paths:

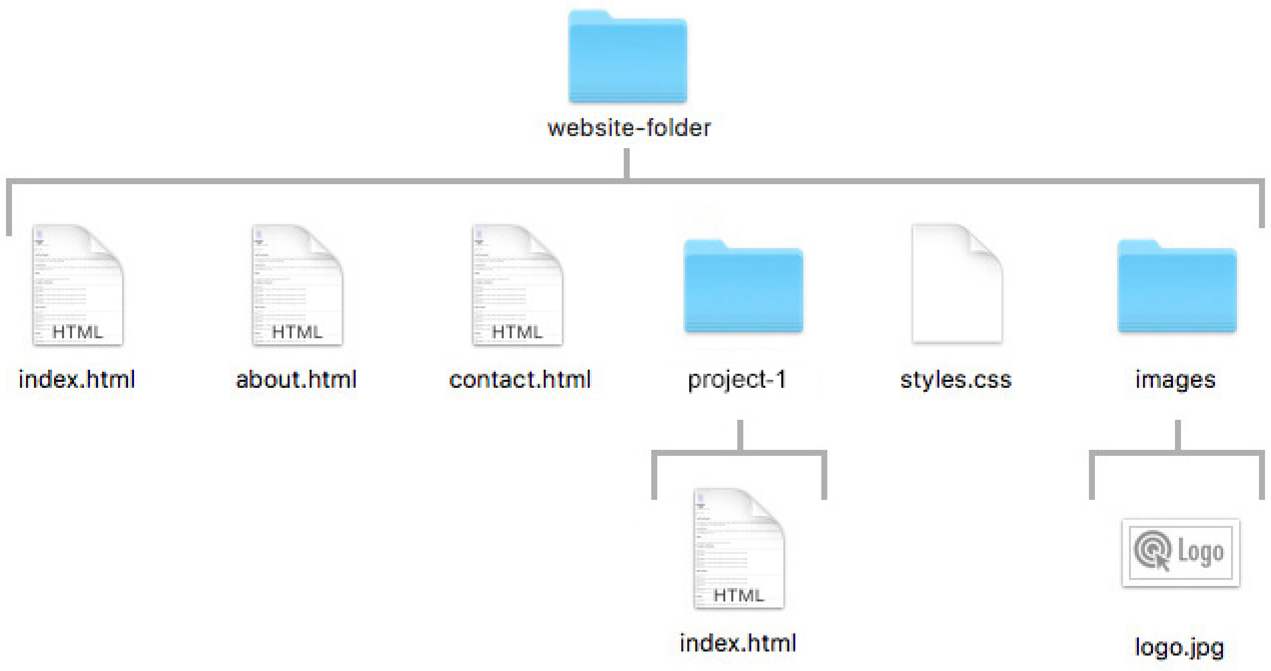
- `images/waterfalls/iceland-waterfall.jpg`
- `gallery/nature/waterfall.jpg`

Relative links: Paths up

Thus far, all the paths you've worked with have been structured from the top down, following a route from the largest, most general part of the file hierarchy to the most specific. But maybe you're already within a directory, or even subdirectory, and you want to call a file or image that's in a level *above* the current level in the website's file structure. Fortunately, you can still reference files from the bottom up.

Outline

Imagine you're on a web page that is within a subdirectory, and you want the web page to reference an image that is located in a folder above it. In this case, you'll use this structure to reference one level above the current level: `../`. As you can see, it involves two periods and a forward slash, which tells the code to move up a level to look for the desired file. This is often referred to as *referencing the parent directory*.

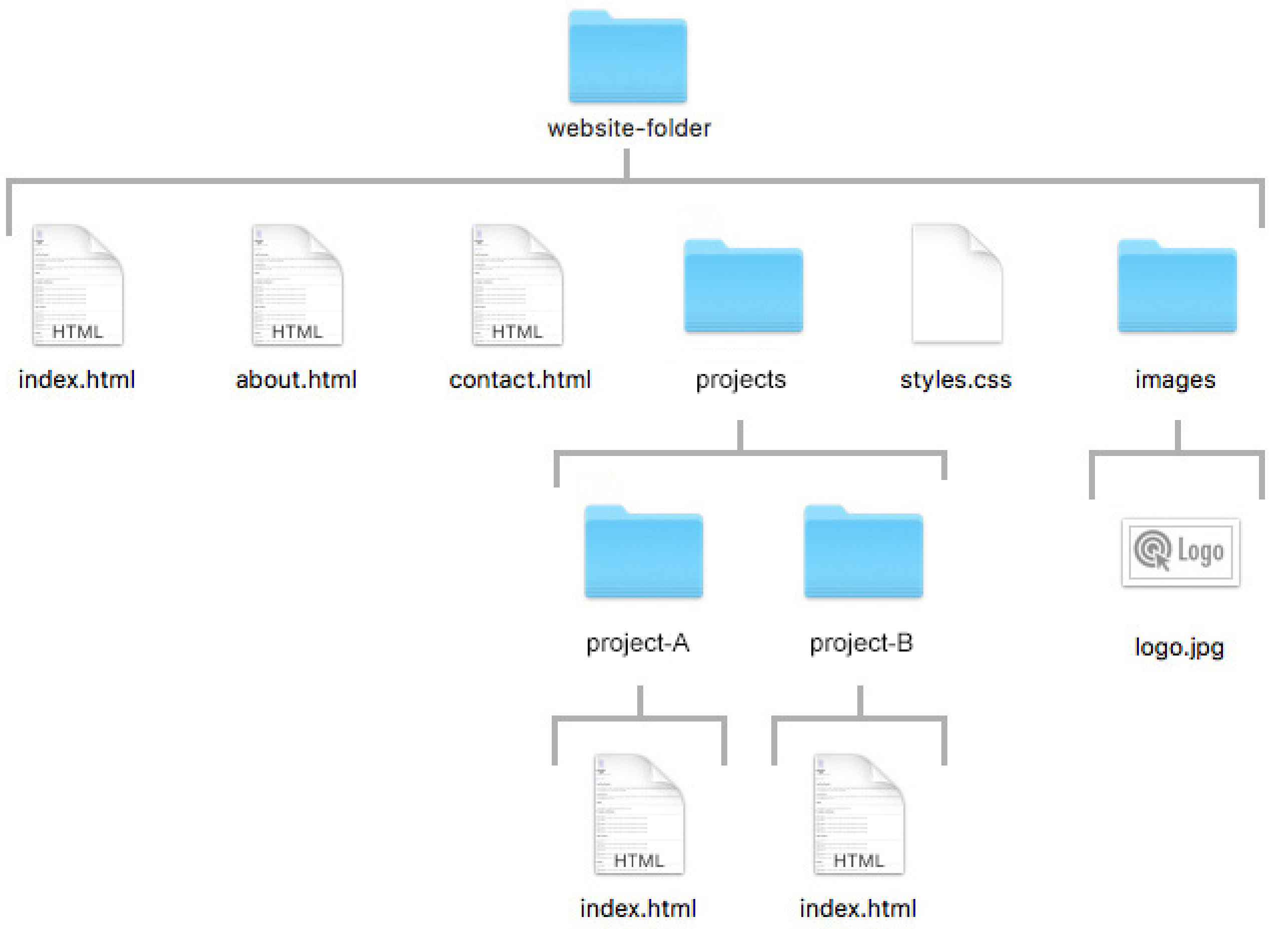


In the example above, the *project-1* folder has an `index.html` page that wants to reference the `logo.jpg` file within the image directory of the primary folder, which is the *website-folder*. To get to the correct image file, the path needs to first look one level above the `index.html` in the *website-folder*. Then, the path needs to continue back down into the *images* folder to look for the `logo.jpg` file.

The path to this file would look like this.

```
<img src="../images/logo.jpg" />
```

You can also move two or more levels up to locate files in directories even further above the web page. This is often referred to as *referencing the grandparent directory*. You'll use the `../` structure for each level you want to move up in the file structure. So if you w to move two levels up, you'll incorporate it twice into the path: `../../`.



Outline

In the above example, the *project-A* folder has an `index.html` page that wants to reference the `logo.jpg` file within the *website-folder*. To get to the correct image file, the path needs to first look one level above the `index.html` file in the *projects* folder, and then move up one more level into the *website-folder*. Finally, the path needs to continue back down into the *images* folder to look for the `logo.jpg` file.

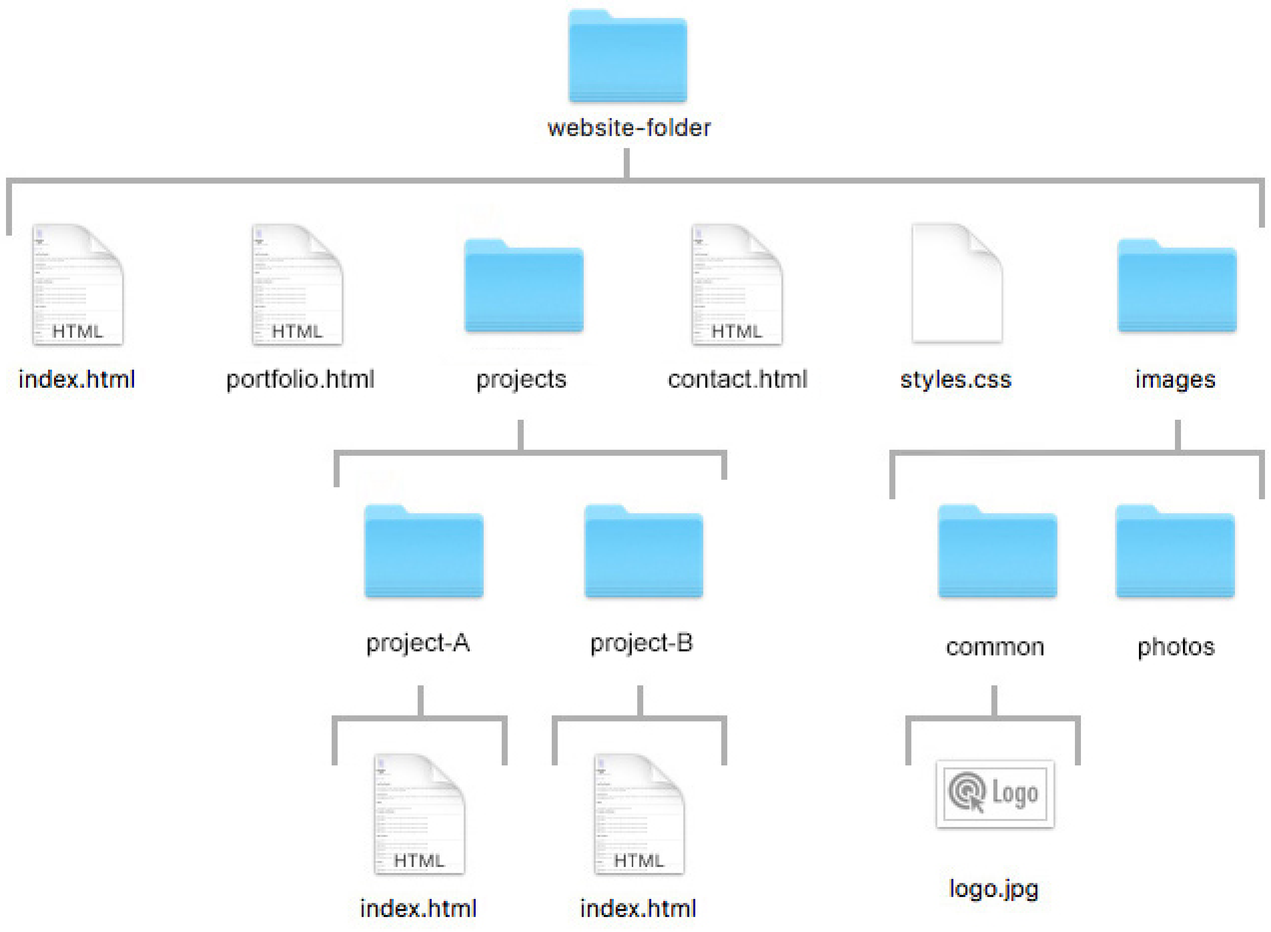
What would this path look like? Check it out below.

```
<img src="../../images/logo.jpg" />
```

Outline

Link path practice

You've been exploring the nuts and bolts of file structures. And for most basic websites, the files are arranged in the following way.



Now, it's your turn. Using the image, determine the paths from each file listed to another file in the structure above. How would you write each of these paths? Write down the answers for yourself in a separate document. (There are a couple included to guide you.)

From	To	Path
index.html	portfolio.html	href="portfolio.html"
index.html	contact.html	
portfolio.html	index.html	
contact.html	index.html	

contact.html	portfolio.html	
index.html	project-A > index.html	
index.html	project-B > index.html	
portfolio.html	project-B > index.html	
index.html	images > logo.jpg	src="images/common/logo.jpg"
portfolio.html	images > logo.jpg	
project-A	images > logo.jpg	
project-B	images > logo.jpg	
project-A > index.html	project-B > index.html	
project-B > index.html	project-A > index.html	

When you're done, feel free to compare your code with this completed code sample.

Link Path Practice Complete

Open links in new tabs

During your personal internet use, you've probably noticed that clicking a link on a web page often reroutes that same page to the new website. By default, links open up the new linked page—whether that's an image, another web page, or another web asset—within the existing browser window. But you can change this default functionality; links can be set to open within a new tab within the browser using the following attribute: `target="\_blank"` (or `target="_blank"`, as seen below). That can be pretty useful, turns out.

```
<a href="http://www.website.com" target="_blank">Lin
```

Contact links

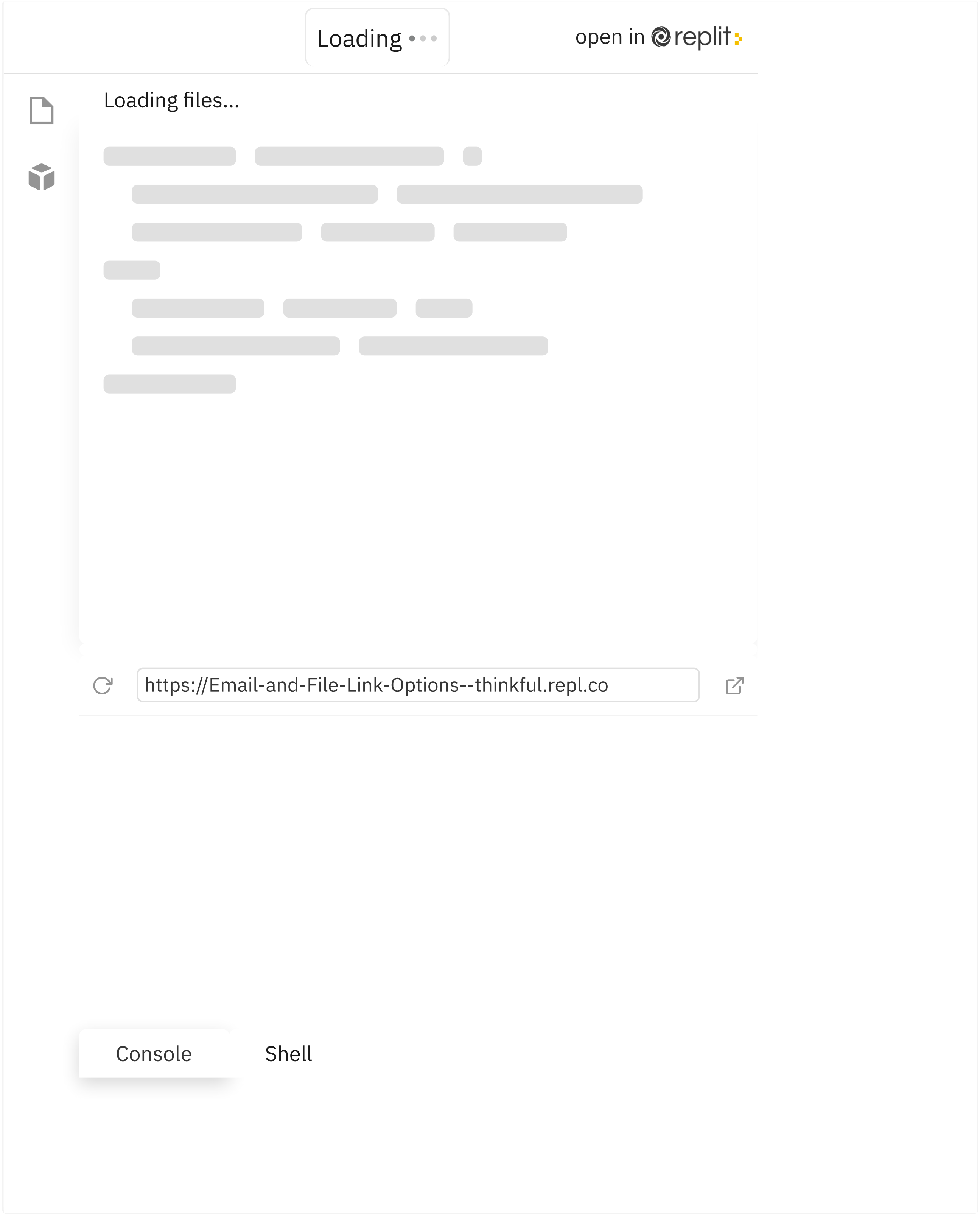
Whenever you provide an email address or phone number within your website, you can use certain HTML attributes to make the links far more user friendly. Here is the complete collection of options for your reference:

- Including a basic email link
- Including an email link with a subject
- Adding CC and BCC to an email link
- Adding body text to an email Link
- Styling email links
- Including telephone links
- Opening file links
- Downloading file links

Outline

Demo: Contact links

The Repl.it below provides all of the examples listed above, with both the code and a functional example. Review the code and play around with the attributes to learn how contact links work on websites. How would you incorporate a link to each of the examples mentioned above?



https://Email-and-File-Link-Options--thinkful.repl.co

Outline

Assignment

The Repl.it below contains the code for a four-page website ca[illegible] *The Adventure*. You'll see that the links are broken. Do the following [illegible] the website's four web pages:

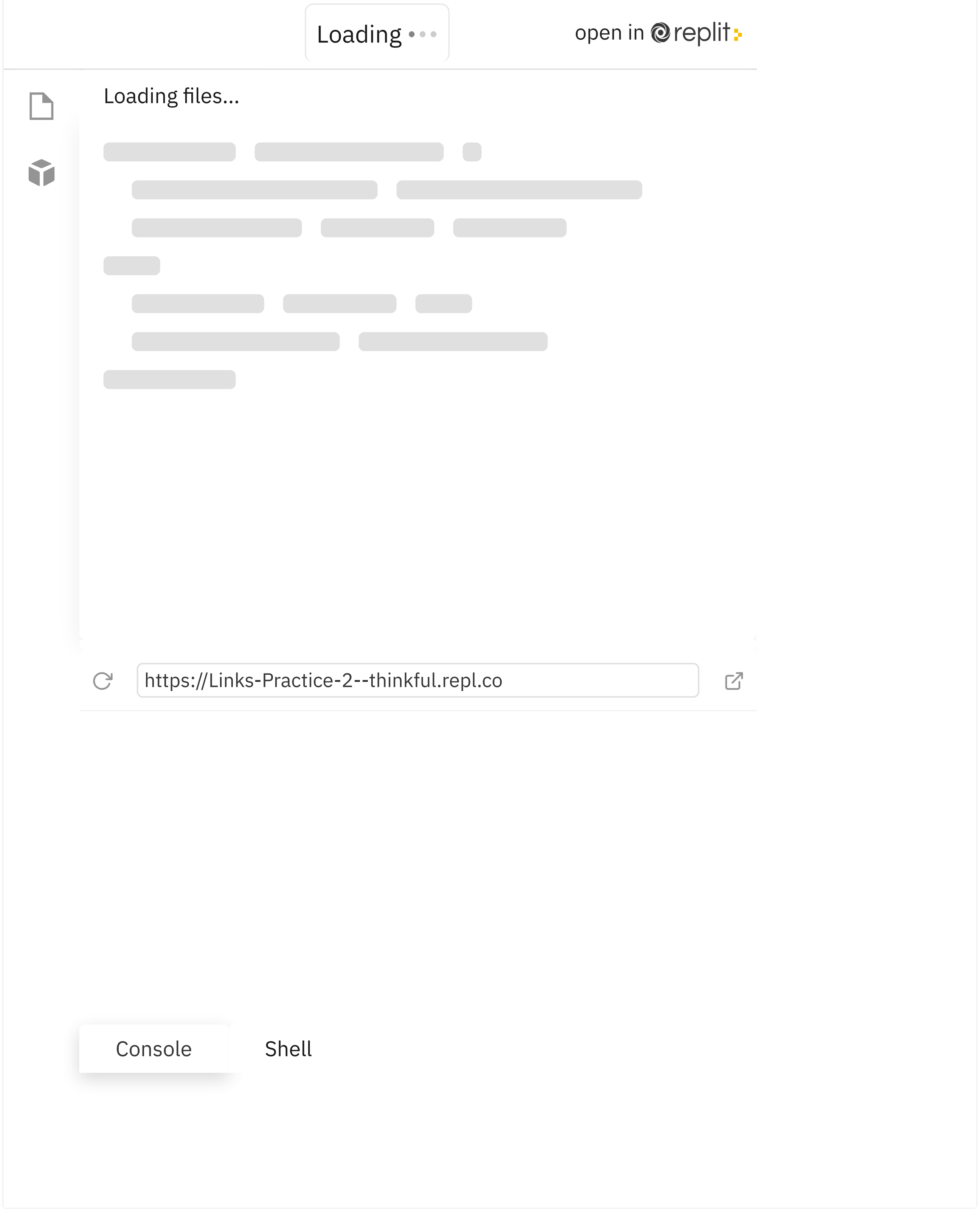
1. Fix and resolve issues with the navigation link paths.
2. Fix and resolve issues with the image paths.
3. Submit a link to your Repl.it in the box below.

However, be sure to change *only* the code within the HTML pages. There are some advanced CSS techniques used for the navigation and page layouts that will be explained in upcoming checkpoints. But for now, focus on connecting the paths for the links and images properly.

When you're done and have submitted a link to your Repl.it in the box below, feel free to compare your code with this completed one.

Practice Link Paths Complete

Outline



https://Links-Practice-2--thinkful.repl.co

Outline

Checkpoint

Submit your ideas or a link to your work here and use it as a conversation starter during your next mentor session.

This checkpoint will not be graded, but is still required.

Your work

03.04.21

Share your ideas here...

bold _italic_ `code` > quote - bullet list

Preview

Outline

Completed

Next checkpoint

How would you rate this content?

Report a typo or other issue

Go to Overview

Outline