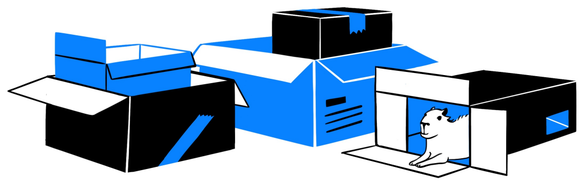
# Your First Independent Project



404

When communicating via HTTP, a server is required to respond to a request, such as a web browser request for a webpage. A 404 error is often returned when pages have been moved or deleted

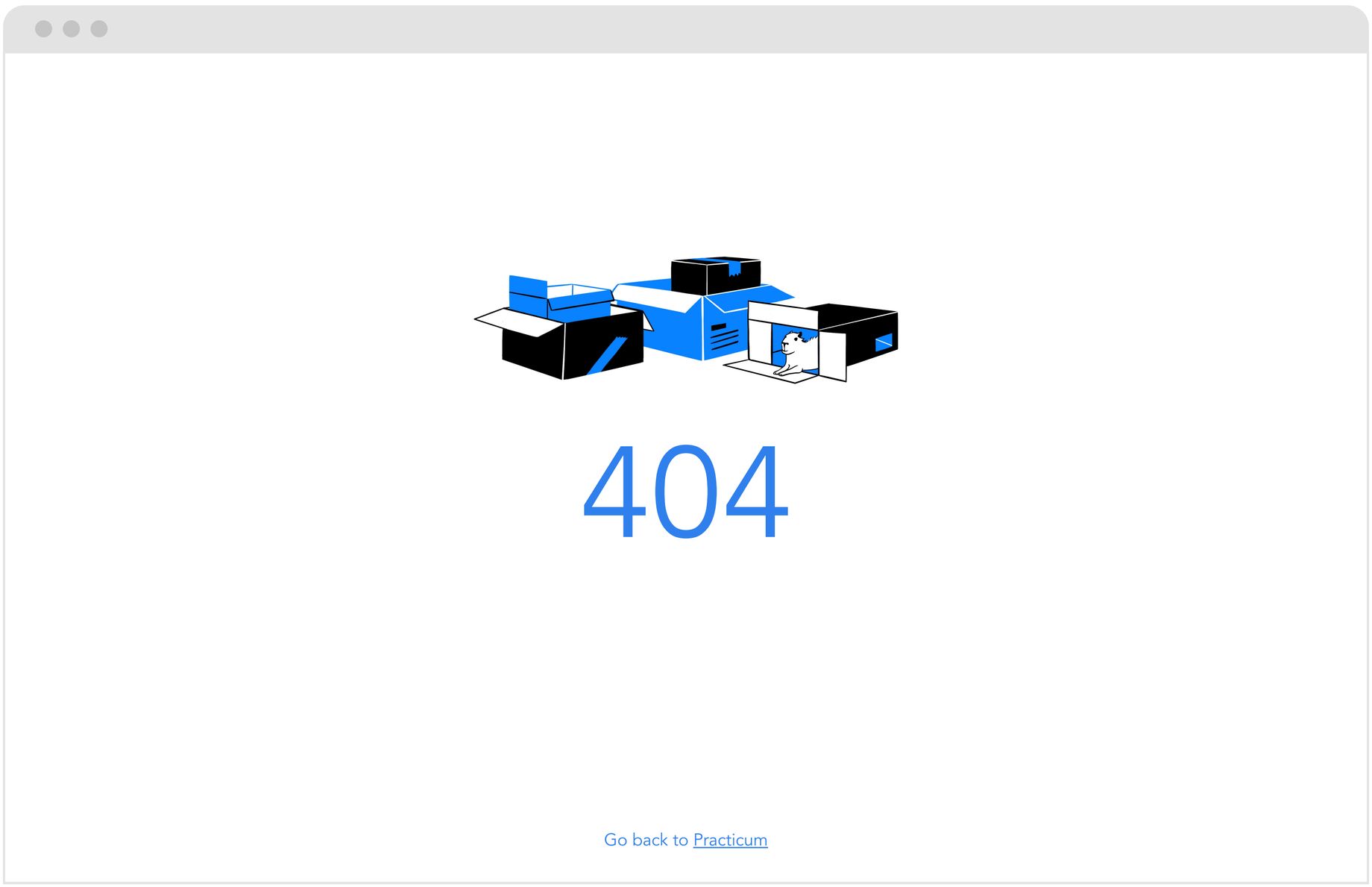
Go back to Practicum

To get started, open the `index.html` and `style.css` files from the starter kit in your code editor. You'll notice that each of these files already contains some code.

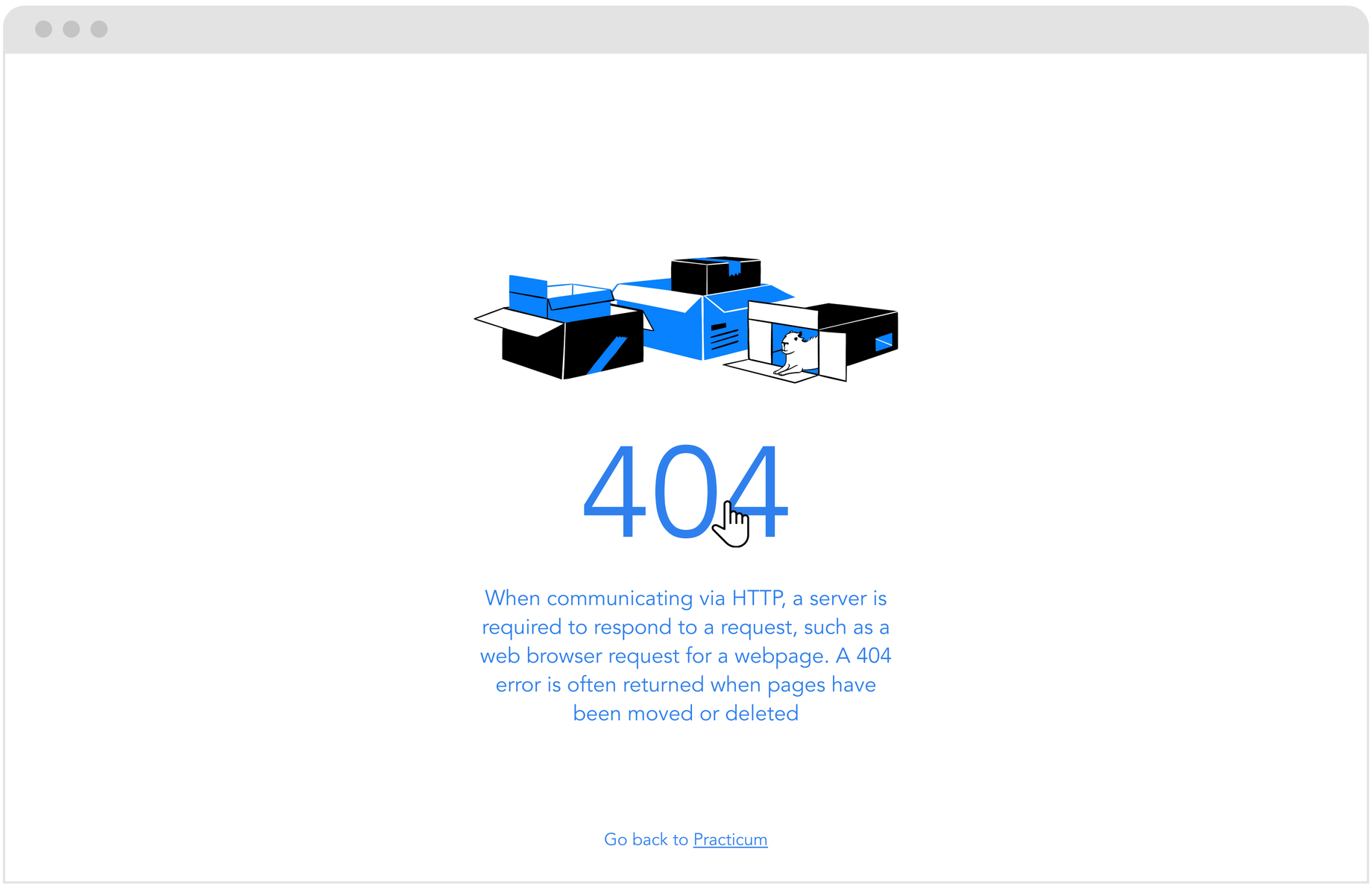
The HTML file already contains the basic structure for your webpage and includes some comments to show you where to write your code.

In the CSS file, the `color` and `font-family` properties have already been assigned to the `body` selector. This means that these styles will automatically be applied to any text elements you create inside the `<body>` element in your HTML file. The CSS file also contains a rule for the `footer` class. This will ensure that the footer text stays in one place. You'll learn more about how to position elements like this in the full program.

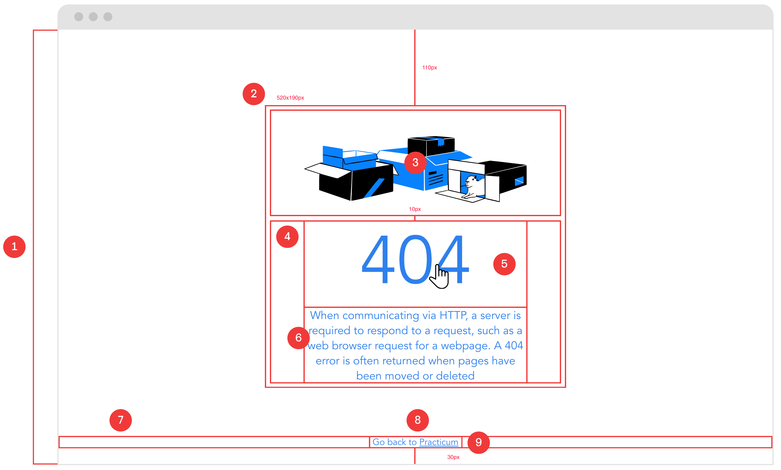
Here's what the webpage will look like in its default state:




Here's what it will look like when the user hovers over the "404" text with their mouse:



Before you start coding the elements for this page, recall that the browser automatically applies margins to some HTML elements. Override this behavior by resetting the margins of the `<body>` and `<p>` elements to zero at the top of your `style.css` file.




**1.** The `<body>` element (in CSS, the `body` selector) — this has already been coded for you in the starter kit. Add the following declarations to its CSS rule:

- `display: flex;` — this will make it easy for you to center the elements on your webpage. This declaration provides us with a wide range of options for arranging our elements. You'll learn more about these in the full program.
- `height: 100vh;` — this ensures that the height of the webpage will always equal the height of the user's display, no matter what size their screen is.

**2.** The `<div>` element with the `container` class — this has already been coded for you in the starter kit. The image and the main texts of your webpage will be nested inside this container. Create a CSS rule for the `container` class and assign it the following declarations:

- `width: 520px;`
- `margin: auto;` — this will keep your container, and all its nested elements, in the center of the page

**3.** The `<img>` element — add this element immediately after the opening tag of the `<div>` element with the container class and then do the following:

- Assign it the `image` class.
- Remember that for images, the `src` attribute is mandatory. Add this attribute and set its value to `./empty-boxes.svg` . This is the path to the file in the starter kit that contains the image shown in the design.
- The alt attribute is also mandatory for `<img>` elements. Give it a value that describes the image.
- Finally, create a CSS rule for the image class and add the `width: 100%` declaration to it. This will ensure that the image has the same width as its container, i.e. `520px` .

**4.** The `<div>` element with the `text` class — this element has already been coded for you in the starter kit's HTML file. The 404 message and the explanatory text will be nested inside this element. Create a CSS rule with the `.text` selector and assign it the following declarations:

- `width: 400px;`
- `margin: 0 auto;` — this will center the element horizontally, while getting rid of the vertical margins.
- `cursor: pointer;` — this will turn the cursor into the pointer icon (which looks like a hand) when the user hovers over this element with their mouse.

**5.** The `<p>` element with the `text-code` class (alternatively, you can use an `<h1>` element) — add this element inside the `<div>` element with the `text` class. Write "404" in between the opening and closing tags. Create a CSS rule for the `text-code` class and give it the following declarations:

- `text-align: center;`
- `font-size: 120px;`
- `font-weight: 400;`

**6.** The `<p>` element with the `text-info` class — add this element inside the `<div>` element with the `text` class immediately after the `<p>` element with the `text-code` class and add the following text between the opening and closing tags:

"When communicating via HTTP, a server is required to respond to a request, such as a web browser request for a webpage. A 404 error is often returned when pages have been moved or deleted."

Create a CSS rule with the `.text-info` selector and assign it the following declarations:

- `text-align: center;`
- `font-size: 20px;`
- `font-weight: 400;`
- `opacity: 0;` — This will ensure that this text is hidden by default
- `transition: opacity .4s ease-in-out;` — This will ensure that when the text transitions between its visible and invisible state, it does so smoothly over a period of 0.4 seconds. This type of behavior is referred to as a transformation. You'll learn all about how to implement transformations in the full program.

**7.** The `<div>` element with the `footer` class — this element has already been coded for you in the starter kit. If you want, you can change this to a `<footer>` element. Add the following declarations to its CSS rule:

- `width: 100%;`
- `display: flex;`

**8.** The `<p>` element — Add this element inside the `<div>` element with the `footer` class. The element should contain the following text: "Go back to Practicum." Create a CSS rule for this element and assign it the following declarations:

- `margin: auto;`
- `font-size: 16px;`

**Note:** You have to make sure that these declarations are only applied to the `<p>` element inside the footer, and not all `<p>` elements on the page. To do this, your CSS rule will need to have something called a nesting selector. Give your rule a selector that will only select paragraphs nested inside `footer` class elements. See if you can find out how to do this by experimenting and/or searching online.

**9.** The `<a>` element — The "Practicum" link in the footer should take you to Practicum's home page. Turn this text into a link by wrapping it in a set of `<a></a>` tags and then configure the link as follows:

- Assign it the mandatory `href` attribute with a value of `https://practicum.yandex.com`
- The default color for links is a different shade of blue to the texts on our webpage, so override this by assigning it the `color: #2f80ed;` declaration. You can either do this with another nesting selector or by assigning a class to the link element.

Once you've coded all these elements, the only thing left to do is implement hover functionality. The paragraph with the `text-info` class should become visible whenever the user hovers over the `<div>` element with the `text` class. You will have to figure out how to implement this functionality by yourself, but here are a few hints:

- You will need to create a new CSS rule
- This CSS rule will contain two classes in the selector
- You will need to append the `:hover` pseudo-class to one of these classes