**Challenges in This Course**

**Challenge Types**

There are two types of challenges in this course. Each one is designed to prepare you for a scenario that you're likely to encounter as a web developer.

The two types of challenges are the following:

* **On-the-job ticket** or **feature request challenges** give you starter code in a folder called Develop, which you'll modify to complete the challenge. Odd-numbered modules follow this format.
* **Job-seeking coding assessments** or **take-home assignments** don't provide starter code. You’ll build these from scratch. Even-numbered modules follow this format.

**Challenge Elements**

Challenges adhere to a format that’s commonly used by software development teams that use **agile project management** to manage their work. Practicing this will prepare you for the workflows you’ll experience as a professional full-stack web developer.

**DEEP DIVE**

Each Challenge contains the following elements:

* **User Story:** This is a short, simple description of a feature told from the perspective of the person who is requesting the new capability, usually a user or customer of the system. This follows an AS AN / I WANT / SO THAT format. For example, "AS A shopper visiting an online store, I WANT to place items in a shopping cart, SO THAT I can purchase them."
* **Acceptance Criteria:** These are the requirements that you must meet to satisfy the scope of work. They are not exhaustive, but they do entail the minimum aspects of a working solution. Consider this a checklist of baseline requirements. Acceptance criteria can be presented in various ways. In this case, we’ll use a common format called **scenario-oriented criteria** which expresses each requirement in a WHEN / THEN format. Don’t worry if this doesn’t make sense now; it will become very familiar to you after you complete a couple of challenges.
* **Mock-up:** This is an image or animation that demonstrates the design and functionality of the web application that you'll build for the Challenge.
* **Submission:** You'll submit your completed Challenge for review. In the real world, when a developer finishes working on a project, another developer reviews the code, providing feedback on errors and making sure that all of the acceptance criteria have been met. For each Challenge, your instructional staff will serve as your team of reviewers.

**Your Task**

This week is an odd-numbered week, so your Challenge is an on-the-job ticket, which means you'll begin with starter code that you need to modify. This week's challenge involves a very important aspect of web development: **accessibility**.

One of the most common tasks for front-end and junior developers is to take existing code and refactor it (recall that to refactor code is to improve it without changing what it does) to meet a certain set of standards or implement a new technology. In this Challenge, a marketing agency has hired you to refactor an existing site to make it more accessible.

**NOTE**

Your coursework this week will prepare you with all the skills that you need to succeed on this assignment!

Web accessibility is an increasingly important consideration for businesses. It ensures that people with disabilities can access a website using assistive technologies such as video captions, screen readers, and braille keyboards. Making a website accessible is also good for business for many reasons, one of them being that accessible sites are better positioned in search engines like Google. It also helps companies avoid litigation that can occur when people with disabilities can't access their website.

Even though accessibility is a broad topic that can include complex requirements, your tech lead has given you a small list of specific criteria to satisfy the project. These criteria are documented in the Acceptance Criteria section.

**IMPORTANT**

An important rule to follow when working with someone else's code is the **Scout Rule**, which recommends that you always leave the code a little cleaner than when you found it.

To impress clients, you should always go the extra mile and improve the codebase for long-term sustainability. For example, make sure that all links are functioning correctly. Also, rework the CSS to make it more efficient by consolidating CSS selectors and properties, organizing them to follow the semantic structure of the HTML elements, and including comments before each element or section of the page.

Are you ready to jump in? Here are this week's challenge requirements:

**User Story**

AS A marketing agency

I WANT a codebase that follows accessibility standards

SO THAT our own site is optimized for search engines

**Acceptance Criteria**

GIVEN a webpage meets accessibility standards

WHEN I view the source code

THEN I find semantic HTML elements

WHEN I view the structure of the HTML elements

THEN I find that the elements follow a logical structure independent of styling and positioning

WHEN I view the image elements

THEN I find accessible alt attributes

WHEN I view the heading attributes

THEN they fall in sequential order

WHEN I view the title element

THEN I find a concise, descriptive title

**Mock-Up**

**NOTE**

This layout is designed for "desktop", so you may notice that some of the elements don't look like the Mock-Up when viewed at a resolution smaller than 768px. In future lessons, you will learn how to make elements "responsive" so that your web application is optimized for any screen size.

The following image shows the web application's appearance and functionality:

**How to Complete the Challenge**

Follow these steps to complete the challenge:

1. Clone your [starter code (Links to an external site.)](https://github.com/coding-boot-camp/urban-octo-telegram).
2. Refactor the code to meet the Acceptance Criteria.
3. Make sure your work meets the full list of grading requirements.
4. Follow the instructions to submit your Challenge for review.

**Grading Requirements**

This challenge is graded based on the following criteria:

**Technical Acceptance Criteria: 40%**

* Satisfies all of the preceding acceptance criteria plus the following code improvements:
  + Application's links all function correctly.
  + Application's CSS selectors and properties are consolidated and organized to follow semantic structure.
  + Application's CSS file is properly commented.

**Deployment: 32%**

* Application deployed at live URL.
* Application loads with no errors.
* Application GitHub URL submitted.
* GitHub repository that contains application code.

**Application Quality: 15%**

* Application resembles (at least 90%) screenshots provided in challenge instructions.

**Repository Quality: 13%**

* Repository has a unique name.
* Repository follows best practices for file structure and naming conventions.
* Repository follows best practices for class/id naming conventions, indentation, quality comments, etc.
* Repository contains multiple descriptive commit messages.
* Repository contains quality README file with description, screenshot, and link to deployed application.

**How to Submit the Challenge**

You are required to submit BOTH of the following for review:

* The URL of the deployed application.
* The URL of the GitHub repository that contains your code. Give the repository a unique name and include a README file that describes the project.