## 

## Welcome to the freeCodeCamp curriculum

We have thousands of coding lessons to help you improve your skills.

You can earn each certification by completing its 5 final projects.

And yes - all of this is 100% free, thanks to the thousands of campers who [donate](https://donate.freecodecamp.org/) to our nonprofit.

If you are new to coding, we recommend you [start at the beginning](https://learn.freecodecamp.org/responsive-web-design/basic-html-and-html5/say-hello-to-html-elements).

#### 1. Responsive Web Design Certification (300 hours)

2.Basic HTML & HTML5

HTML, or HyperText Markup Language, is a markup language used to describe the structure of a web page. It uses a special syntax or notation to organize and give information about the page to the browser. Elements usually have opening and closing tags that surround and give meaning to content. For example, there are different tag options to place around text to show whether it is a heading, a paragraph, or a list.

For example:

<h1>Top level heading: Maybe a page title</h1>

<p>A paragraph of text. Some information we would like to communicate to the viewer. This can be as long or short as we would like.</p>

<ol>

<li>Number one on the list</li>

<li>Number two</li>

<li>A third item</li>

</ol>

# Top level heading: Maybe a page title

A paragraph of text. Some information we would like to communicate to the user. This can be as long or short as we would like.

1. Number one on the list
2. Number two
3. A third item

The HyperText part of HTML comes from the early days of the web and its original use case. Pages usually contained static documents that contained references to other documents. These references contained hypertext links used by the browser to navigate to the reference document so the user could read the reference document without having to manually search for it.

As web pages and web applications grow more complex, the W3 Consortium updates the HTML specification to ensure that a webpage can be shown reliably on any browser. The latest version of HTML is HTML5.

This section introduces how to use HTML elements to give structure and meaning to your web content.

3.Basic CSS

Cascading Style Sheets (CSS) tell the browser how to display the text and other content that you write in HTML.  
  
Note that CSS is case-sensitive so be careful with your capitalization. CSS has been adopted by all major browsers and allows you to control:

* color
* fonts
* positioning
* spacing
* sizing
* decorations
* transitions

There are three main ways to apply CSS styling. You can apply inline styles directly to HTML elements with the style attribute. Alternatively, you can place CSS rules within style tags in an HTML document. Finally, you can write CSS rules in an external style sheet, then reference that file in the HTML document. Even though the first two options have their use cases, most developers prefer external style sheets because they keep the styles separate from the HTML elements. This improves the readability and reusability of your code. The idea behind CSS is that you can use a selector to target an HTML element in the DOM (Document Object Model) and then apply a variety of attributes to that element to change the way it is displayed on the page.  
  
In this section, you'll see how adding CSS styles to the elements of your CatPhotoApp can change it from simple text to something more.

## 4.The Applied Visual Design Challenges

Visual Design in web development is a broad topic. It combines typography, color theory, graphics, animation, and page layout to help deliver a site's message. The definition of good design is a well-discussed subject, with many books on the theme.  
  
At a basic level, most web content provides a user with information. The visual design of the page can influence its presentation and a user's experience. In web development, HTML gives structure and semantics to a page's content, and CSS controls the layout and appearance of it.  
  
This section covers some of the basic tools developers use to create their own visual designs.

5.The Applied Accessibility Challenges

"Accessibility" generally means having web content and a user interface that can be understood, navigated, and interacted with by a broad audience. This includes people with visual, auditory, mobility, or cognitive disabilities. Websites should be open and accessible to everyone, regardless of a user's abilities or resources. Some users rely on assistive technology such as a screen reader or voice recognition software. Other users may be able to navigate through a site only using a keyboard. Keeping the needs of various users in mind when developing your project can go a long way towards creating an open web. Here are three general concepts this section will explore throughout the following challenges:

1. have well-organized code that uses appropriate markup
2. ensure text alternatives exist for non-text and visual content
3. create an easily-navigated page that's keyboard-friendly

Having accessible web content is an ongoing challenge. A great resource for your projects going forward is the W3 Consortium's Web Content Accessibility Guidelines (WCAG). They set the international standard for accessibility and provide a number of criteria you can use to check your work.

## 6.The Responsive Web Design Challenges

Today, there are many types of devices that can access the web. They range from large desktop computers to small mobile phones. These devices have different screen sizes, resolutions, and processing power. Responsive Web Design is an approach to designing web content that responds to the constraints of different devices. The page structure and CSS rules should be flexible to accommodate these differences. In general, design the page's CSS to your target audience. If you expect most of your traffic to be from mobile users, take a 'mobile-first' approach. Then add conditional rules for larger screen sizes. If your visitors are desktop users, then design for larger screens with conditional rules for smaller sizes. CSS gives you the tools to write different style rules, then apply them depending on the device displaying the page. This section will cover the basic ways to use CSS for Responsive Web Design.

## 7.The CSS Flexbox Challenges

A website's User Interface ("UI") has two components. First, there are the visual elements, such as colors, fonts, and images. Second, there is the placement or positioning of those elements. In Responsive Web Design, a UI layout must accommodate many different browsers and devices accessing the content.  
  
CSS3 introduced Flexible Boxes, or flexbox, to create page layouts for a dynamic UI. It is a layout mode that arranges elements in a predictable way for different screen sizes and browsers. While somewhat new, all popular modern browsers support flexbox. This section covers how to use flexbox and the different layout options it offers.

## 8.The CSS Grid Challenges

CSS Grid helps you easily build complex web designs. It works by turning an HTML element into a grid container with rows and columns for you to place children elements where you want within the grid.

9.The Responsive Web Design Projects

Time to put your newly learnt skills to work! By working on projects you would have the opportunity of applying all the skills, principles and concepts you have learnt so far HTML, CSS, Visual Design, Accessibility, etc.

In this section you get the chance to:

* Build a Tribute Page
* Build a Survey Form
* Build a Product Landing Page
* Build a Technical Documentation Page
* Build a Personal Portfolio Webpage

By the end of this, you would have 5 responsive websites under your belt that you can show off to friends, family, employers, etc. Have fun and remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

10.Responsive Web Design Projects - Build a Tribute Page

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/zNqgVx>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (just for example jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** My tribute page should have an element with a corresponding id="main", which contains all other elements.
* **User Story #2:** I should see an element with a corresponding id="title", which contains a string (i.e. text) that describes the subject of the tribute page (e.g. "Dr. Norman Borlaug").
* **User Story #3:** I should see a div element with a corresponding id="img-div".
* **User Story #4:** Within the img-div element, I should see an img element with a corresponding id="image".
* **User Story #5:** Within the img-div element, I should see an element with a corresponding id="img-caption" that contains textual content describing the image shown in img-div.
* **User Story #6:** I should see an element with a corresponding id="tribute-info", which contains textual content describing the subject of the tribute page.
* **User Story #7:** I should see an a element with a corresponding id="tribute-link", which links to an outside site that contains additional information about the subject of the tribute page. HINT: You must give your element an attribute of targetand set it to \_blankin order for your link to open in a new tab (i.e. target="\_blank").
* **User Story #8:** The imgelement should responsively resize, relative to the width of its parent element, without exceeding its original size.
* **User Story #9:** The imgelement should be centered within its parent element.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js.
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

11.Build a Survey Form

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/VPaoNP>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (just for example jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I can see a title with id="title"in H1 sized text.
* **User Story #2:** I can see a short explanation with id="description"in P sized text.
* **User Story #3:** I can see a form with id="survey-form".
* **User Story #4:** Inside the form element, I am required to enter my name in a field with id="name".
* **User Story #5:** Inside the form element, I am required to enter an email in a field with id="email".
* **User Story #6:** If I enter an email that is not formatted correctly, I will see an HTML5 validation error.
* **User Story #7:** Inside the form, I can enter a number in a field with id="number".
* **User Story #8:** If I enter non-numbers in the number input, I will see an HTML5 validation error.
* **User Story #9:** If I enter numbers outside the range of the number input, which are defined by the minand maxattributes, I will see an HTML5 validation error.
* **User Story #10:** For the name, email, and number input fields inside the form I can see corresponding labels that describe the purpose of each field with the following ids: id="name-label", id="email-label", and id="number-label".
* **User Story #11:** For the name, email, and number input fields, I can see placeholder text that gives me a description or instructions for each field.
* **User Story #12:** Inside the form element, I can select an option from a dropdown that has a corresponding id="dropdown".
* **User Story #13:** Inside the form element, I can select a field from one or more groups of radio buttons. Each group should be grouped using the name attribute.
* **User Story #14:** Inside the form element, I can select several fields from a series of checkboxes, each of which must have a value attribute.
* **User Story #15:** Inside the form element, I am presented with a textareaat the end for additional comments.
* **User Story #16:** Inside the form element, I am presented with a button with id="submit"to submit all my inputs.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

12.Build a Product Landing Page

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/RKRbwL>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (just for example jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** My product landing page should have a headerelement with a corresponding id="header".
* **User Story #2:** I can see an image within the headerelement with a corresponding id="header-img". A company logo would make a good image here.
* **User Story #3:** Within the #headerelement I can see a navelement with a corresponding id="nav-bar".
* **User Story #4:** I can see at least three clickable elements inside the navelement, each with the class nav-link.
* **User Story #5:** When I click a .nav-linkbutton in the navelement, I am taken to the corresponding section of the landing page.
* **User Story #6:** I can watch an embedded product video with id="video".
* **User Story #7:** My landing page has a formelement with a corresponding id="form".
* **User Story #8:** Within the form, there is an inputfield with id="email"where I can enter an email address.
* **User Story #9:** The #emailinput field should have placeholder text to let the user know what the field is for.
* **User Story #10:** The #emailinput field uses HTML5 validation to confirm that the entered text is an email address.
* **User Story #11:** Within the form, there is a submit inputwith a corresponding id="submit".
* **User Story #12:** When I click the #submitelement, the email is submitted to a static page (use this mock URL: <https://www.freecodecamp.com/email-submit>) that confirms the email address was entered and that it posted successfully.
* **User Story #13:** The navbar should always be at the top of the viewport.
* **User Story #14:** My product landing page should have at least one media query.
* **User Story #15:** My product landing page should utilize CSS flexbox at least once.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/full/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

13.Build a Technical Documentation Page

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/NdrKKL>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (just for example jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I can see a mainelement with a corresponding id="main-doc", which contains the page's main content (technical documentation).
* **User Story #2:** Within the #main-docelement, I can see several sectionelements, each with a class of main-section. There should be a minimum of 5.
* **User Story #3:** The first element within each .main-sectionshould be a headerelement which contains text that describes the topic of that section.
* **User Story #4:** Each sectionelement with the class of main-sectionshould also have an id that corresponds with the text of each headercontained within it. Any spaces should be replaced with underscores (e.g. The sectionthat contains the header "Javascript and Java" should have a corresponding id="Javascript\_and\_Java").
* **User Story #5:** The .main-sectionelements should contain at least 10 pelements total (not each).
* **User Story #6:** The .main-sectionelements should contain at least 5 codeelements total (not each).
* **User Story #7:** The .main-sectionelements should contain at least 5 liitems total (not each).
* **User Story #8:** I can see a navelement with a corresponding id="navbar".
* **User Story #9:** The navbar element should contain one headerelement which contains text that describes the topic of the technical documentation.
* **User Story #10:** Additionally, the navbar should contain link (a) elements with the class of nav-link. There should be one for every element with the class main-section.
* **User Story #11:** The headerelement in the navbar must come before any link (a) elements in the navbar.
* **User Story #12:** Each element with the class of nav-linkshould contain text that corresponds to the headertext within each section(e.g. if you have a "Hello world" section/header, your navbar should have an element which contains the text "Hello world").
* **User Story #13:** When I click on a navbar element, the page should navigate to the corresponding section of the main-docelement (e.g. If I click on a nav-linkelement that contains the text "Hello world", the page navigates to a sectionelement that has that id and contains the corresponding header.
* **User Story #14:** On regular sized devices (laptops, desktops), the element with id="navbar"should be shown on the left side of the screen and should always be visible to the user.
* **User Story #15:** My Technical Documentation page should use at least one media query.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

14.Build a Personal Portfolio Webpage

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/zNBOYG>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use HTML, JavaScript, and CSS to complete this project. Plain CSS is recommended because that is what the lessons have covered so far and you should get some practice with plain CSS. You can use Bootstrap or SASS if you choose. Additional technologies (just for example jQuery, React, Angular, or Vue) are not recommended for this project, and using them is at your own risk. Other projects will give you a chance to work with different technology stacks like React. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** My portfolio should have a welcome section with an id of welcome-section.
* **User Story #2:** The welcome section should have an h1element that contains text.
* **User Story #3:** My portfolio should have a projects section with an id of projects.
* **User Story #4:** The projects section should contain at least one element with a class of project-tileto hold a project.
* **User Story #5:** The projects section should contain at least one link to a project.
* **User Story #6:** My portfolio should have a navbar with an id of navbar.
* **User Story #7:** The navbar should contain at least one link that I can click on to navigate to different sections of the page.
* **User Story #8:** My portfolio should have a link with an id of profile-link, which opens my GitHub or FCC profile in a new tab.
* **User Story #9:** My portfolio should have at least one media query.
* **User Story #10:** The height of the welcome section should be equal to the height of the viewport.
* **User Story #11:** The navbar should always be at the top of the viewport.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

#### 15.Javascript Algorithms And Data Structures Certification (300 hours)

## 16. Introduction to JavaScript

JavaScript is a high-level programming language that all modern web browsers support. It is also one of the core technologies of the web, along with HTML and CSS that you may have learned previously. This section will cover basic JavaScript programming concepts, which range from variables and arithmetic to objects and loops.

17. Introduction to the ES6 Challenges

ECMAScript is a standardized version of JavaScript with the goal of unifying the language's specifications and features. As all major browsers and JavaScript-runtimes follow this specification, the term *ECMAScript* is interchangeable with the term *JavaScript*.  
  
Most of the challenges on freeCodeCamp use the ECMAScript 5 (ES5) specification of the language, finalized in 2009. But JavaScript is an evolving programming language. As features are added and revisions are made, new versions of the language are released for use by developers.  
  
The most recent standardized version is called ECMAScript 6 (ES6), released in 2015. This new version of the language adds some powerful features that will be covered in this section of challenges, including:

* Arrow functions
* Classes
* Modules
* Promises
* Generators
* letand const

**Note**  
Not all browsers support ES6 features. If you use ES6 in your own projects, you may need to use a program (transpiler) to convert your ES6 code into ES5 until browsers support ES6.

## 18. The Regular Expression Challenges

Regular expressions are special strings that represent a search pattern. Also known as "regex" or "regexp", they help programmers match, search, and replace text. Regular expressions can appear cryptic because a few characters have special meaning. The goal is to combine the symbols and text into a pattern that matches what you want, but only what you want. This section will cover the characters, a few shortcuts, and the common uses for writing regular expressions.

## 19. The Debugging Challenges

Debugging is a valuable and (unfortunately) necessary tool for programmers. It follows the testing phase of checking if your code works as intended, and discovering it does not. Debugging is the process of finding exactly what isn't working and fixing it. After spending time creating a brilliant block of code, it is tough realizing it may have errors. These issues generally come in three forms: 1) syntax errors that prevent a program from running, 2) runtime errors when code fails to execute or has unexpected behavior, and 3) semantic (or logical) errors when code doesn't do what it's meant to.  
  
Modern code editors (and experience) can help identify syntax errors. Semantic and runtime errors are harder to find. They may cause your program to crash, make it run forever, or give incorrect output. Think of debugging as trying to understand why your code is behaving the way it is.  
  
Example of a syntax error - often detected by the code editor:

funtion willNotWork( {  
  console.log("Yuck");  
}  
// "function" keyword is misspelled and there's a missing parenthesis

Here's an example of a runtime error - often detected while the program executes:

function loopy() {  
  while(true) {  
    console.log("Hello, world!");  
  }  
}  
// Calling loopy starts an infinite loop, which may crash your browser

Example of a semantic error - often detected after testing code output:

function calcAreaOfRect(w, h) {  
  return w + h; // This should be w \* h  
}  
let myRectArea = calcAreaOfRect(2, 3);  
// Correct syntax and the program executes, but this gives the wrong answer

Debugging is frustrating, but it helps to develop (and follow) a step-by-step approach to review your code. This means checking the intermediate values and types of variables to see if they are what they should be. You can start with a simple process of elimination.  
  
For example, if function A works and returns what it's supposed to, then function B may have the issue. Or start checking values in a block of code from the middle to try to cut the search space in half. A problem in one spot indicates a bug in the first half of the code. If not, it's likely in the second.  
  
This section will cover a couple helpful tools to find bugs, and some of the common forms they take. Fortunately, debugging is a learnable skill that just requires a little patience and practice to master.

## 20. Introduction to the Basic Data Structure Challenges

Data can be stored and accessed in many different ways, both in Javascript and other languages. This section will teach you how to manipulate arrays, as well as access and copy the information within them. It will also teach you how to manipulate and access the data within Javascript objects, using both dot and bracket notation. When you're done with this section, you should understand the basic properties and differences between arrays and objects, as well as how to choose which to use for a given purpose.

## 21. Introduction to Basic Algorithm Scripting

A computer algorithm is a sequence of steps that is followed to achieve a particular outcome. To write an algorithm, you must first understand a problem, and then solve it with coding.

To make solving problems easier, it can be helpful to break them down into many chunks. Then, each chunk can be solved one by one. For example, if you are building a calculator, don't try to solve the problem as a whole. First, consider how to get inputs. Then, determine each arithmetic operation one by one. Finally, display the results.

In this section we will learn to solve basic algorithm problems using JavaScript. This will help you improve your problem solving skills and prepare you to later solve more complex problems.

#### Hint: If you get stuck, try using console.log()to log variable values to the console. This will help to debug problems.

## 22. Introduction to the Object Oriented Programming Challenges

At its core, software development solves a problem or achieves a result with computation. The software development process first defines a problem, then presents a solution. Object oriented programming is one of several major approaches to the software development process.  
  
As its name implies, object oriented programming organizes code into object definitions. These are sometimes called classes, and they group together data with related behavior. The data is an object's attributes, and the behavior (or functions) are methods.  
  
The object structure makes it flexible within a program. Objects can transfer information by calling and passing data to another object's methods. Also, new classes can receive, or inherit, all the features from a base or parent class. This helps to reduce repeated code.  
  
Your choice of programming approach depends on a few factors. These include the type of problem, as well as how you want to structure your data and algorithms. This section covers object oriented programming principles in JavaScript.

23. Introduction to the Functional Programming Challenges

Functional programming is an approach to software development based around the evaluation of functions. Like mathematics, functions in programming map input to output to produce a result. You can combine basic functions in many ways to build more and more complex programs.  
  
Functional programming follows a few core principles:

* Functions are independent from the state of the program or global variables. They only depend on the arguments passed into them to make a calculation
* Functions try to limit any changes to the state of the program and avoid changes to the global objects holding data
* Functions have minimal side effects in the program

The functional programming software development approach breaks a program into small, testable parts. This section covers basic functional programming principles in JavaScript.

## 24. Introduction to the Intermediate Algorithm Scripting Challenges

This is a stub introduction

25. Introduction to the JavaScript Algorithms and Data Structures Projects

Time to put your new JavaScript skills to work! These challenges will be similar to the algorithm scripting challenges but more difficult. This will allow you to prove how much you have learned.

In this section you will create the following small JavaScript programs:

* Palindrome Checker
* Roman Numeral Converter
* Caesars Cipher
* Telephone Number Validator
* Cash Register

Have fun and remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

Good Luck!

## 26. JavaScript Algorithms and Data Structures Projects: Palindrome Checker

Return trueif the given string is a palindrome. Otherwise, return false.

A palindrome is a word or sentence that's spelled the same way both forward and backward, ignoring punctuation, case, and spacing.

**Note**  
You'll need to remove **all non-alphanumeric characters** (punctuation, spaces and symbols) and turn everything into the same case (lower or upper case) in order to check for palindromes.

We'll pass strings with varying formats, such as "racecar", "RaceCar", and "race CAR"among others.

We'll also pass strings with special symbols, such as "2A3\*3a2", "2A3 3a2", and "2\_A3\*3#A2".

Remember to use [Read-Search-Ask](http://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) if you get stuck. Write your own code.

## 27. JavaScript Algorithms and Data Structures Projects: Roman Numeral Converter

Convert the given number into a roman numeral.

All [roman numerals](http://www.mathsisfun.com/roman-numerals.html) answers should be provided in upper-case.

Remember to use [Read-Search-Ask](http://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) if you get stuck. Try to pair program. Write your own code.

## 28. JavaScript Algorithms and Data Structures Projects: Caesars Cipher

One of the simplest and most widely known ciphers is a Caesar cipher, also known as a shift cipher. In a shift cipherthe meanings of the letters are shifted by some set amount.

A common modern use is the [ROT13](https://en.wikipedia.org/wiki/ROT13) cipher, where the values of the letters are shifted by 13 places. Thus 'A' ↔ 'N', 'B' ↔ 'O' and so on.

Write a function which takes a [ROT13](https://en.wikipedia.org/wiki/ROT13) encoded string as input and returns a decoded string.

All letters will be uppercase. Do not transform any non-alphabetic character (i.e. spaces, punctuation), but do pass them on.

Remember to use [Read-Search-Ask](http://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) if you get stuck. Try to pair program. Write your own code.

## 29. JavaScript Algorithms and Data Structures Projects: Telephone Number Validator

Return trueif the passed string looks like a valid US phone number.

The user may fill out the form field any way they choose as long as it has the format of a valid US number. The following are examples of valid formats for US numbers (refer to the tests below for other variants):

555-555-5555  
(555)555-5555  
(555) 555-5555  
555 555 5555  
5555555555  
1 555 555 5555

For this challenge you will be presented with a string such as 800-692-7753or 8oo-six427676;laskdjf. Your job is to validate or reject the US phone number based on any combination of the formats provided above. The area code is required. If the country code is provided, you must confirm that the country code is 1. Return trueif the string is a valid US phone number; otherwise return false.

Remember to use [Read-Search-Ask](http://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) if you get stuck. Try to pair program. Write your own code.

## 30. JavaScript Algorithms and Data Structures Projects: Cash Register

Design a cash register drawer function checkCashRegister()that accepts purchase price as the first argument (price), payment as the second argument (cash), and cash-in-drawer (cid) as the third argument.

cidis a 2D array listing available currency.

The checkCashRegister()function should always return an object with a statuskey and a changekey.

Return {status: "INSUFFICIENT\_FUNDS", change: []}if cash-in-drawer is less than the change due, or if you cannot return the exact change.

Return {status: "CLOSED", change: [...]}with cash-in-drawer as the value for the key changeif it is equal to the change due.

Otherwise, return {status: "OPEN", change: [...]}, with the change due in coins and bills, sorted in highest to lowest order, as the value of the changekey.

Remember to use [Read-Search-Ask](http://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) if you get stuck. Try to pair program. Write your own code.

|  |  |
| --- | --- |
| **Currency Unit** | **Amount** |
| Penny | $0.01 (PENNY) |
| Nickel | $0.05 (NICKEL) |
| Dime | $0.1 (DIME) |
| Quarter | $0.25 (QUARTER) |
| Dollar | $1 (DOLLAR) |
| Five Dollars | $5 (FIVE) |
| Ten Dollars | $10 (TEN) |
| Twenty Dollars | $20 (TWENTY) |
| One-hundred Dollars | $100 (ONE HUNDRED) |

#### 31. Front End Libraries Certification (300 hours)

32. Introduction to the Bootstrap Challenges

Bootstrap is a front-end framework used to design responsive web pages and web applications. It takes a mobile-first approach to web development. Bootstrap includes pre-built CSS styles and classes, plus some JavaScript functionality. Bootstrap uses a responsive 12 column grid layout and has design templates for:

* buttons
* images
* tables
* forms
* navigation

This section introduces some of the ways to use Bootstrap in your web projects.

## 33. Introduction to jQuery

jQuery is one of the many libraries for JavaScript. It is designed to simplify scripting done on the client side. jQuery's most recognizable characteristic is its dollar sign ($) syntax. With it, you can easily manipulate elements, create animations and handle input events.

## 34. Introduction to the Sass Challenges

Sass, or "Syntactically Awesome StyleSheets", is a language extension of CSS. It adds features that aren't available using basic CSS syntax. Sass makes it easier for developers to simplify and maintain the style sheets for their projects.  
  
Sass can extend the CSS language because it is a preprocessor. It takes code written using Sass syntax, and converts it into basic CSS. This allows you to create variables, nest CSS rules into others, and import other Sass files, among other things. The result is more compact, easier to read code.  
  
There are two syntaxes available for Sass. The first, known as SCSS (Sassy CSS) and used throughout these challenges, is an extension of the syntax of CSS. This means that every valid CSS stylesheet is a valid SCSS file with the same meaning. Files using this syntax have the .scss extension.  
  
The second and older syntax, known as the indented syntax (or sometimes just "Sass"), uses indentation rather than brackets to indicate nesting of selectors, and newlines rather than semicolons to separate properties. Files using this syntax have the .sass extension.  
  
This section introduces the basic features of Sass.

## 35. Introduction to the React Challenges

React, popularized by Facebook, is an open-source JavaScript library for building user interfaces. It is used to create components, handle state and props, utilize event listeners and certain life cycle methods to update data as it changes.

React combines HTML with JavaScript functionality to create its own markup language, JSX. This section will introduce you to all of these concepts and how to implement them for use with your own projects.

## 36. Introduction to the Redux Challenges

[Redux](https://redux.js.org/) is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. While you can use Redux with any view library, it's introduced here before being combined with React.

Improve this intro on [GitHub](https://github.com/freeCodeCamp/learn/tree/master/src/introductions/front-end-libraries/redux).

## 37. Introduction to the React and Redux Challenges

This series of challenges introduces how to use Redux with React.

In a React Redux app, you create a single Redux store that manages the state of your entire app. Your React components subscribe to only the pieces of data in the store that are relevant to their role. Then, you dispatch actions directly from React components, which then trigger store updates.

Improve this intro on [GitHub](https://github.com/freeCodeCamp/learn/tree/master/src/introductions/front-end-libraries/react-and-redux/index.md).

## 38. Introduction to the Front End Libraries Projects

This introduction is a stub

Help us make it real on [GitHub](https://github.com/freeCodeCamp/learn/tree/master/src/introductions).

39. Front End Libraries Projects - Build a Random Quote Machine

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/qRZeGZ>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use any mix of HTML, JavaScript, CSS, Bootstrap, SASS, React, Redux, and jQuery to complete this project. You should use a frontend framework (like React for example) because this section is about learning frontend frameworks. Additional technologies not listed above are not recommended and using them is at your own risk. We are looking at supporting other frontend frameworks like Angular and Vue, but they are not currently supported. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I can see a wrapper element with a corresponding id="quote-box".
* **User Story #2:** Within #quote-box, I can see an element with a corresponding id="text".
* **User Story #3:** Within #quote-box, I can see an element with a corresponding id="author".
* **User Story #4:** Within #quote-box, I can see a clickable element with a corresponding id="new-quote".
* **User Story #5:** Within #quote-box, I can see a clickable <codea< code="" style="box-sizing: inherit; margin-bottom: 0px;">element with a corresponding id="tweet-quote".</codea<>
* **User Story #6:** On first load, my quote machine displays a random quote in the element with id="text".
* **User Story #7:** On first load, my quote machine displays the random quote's author in the element with id="author".
* **User Story #8:** When the #new-quotebutton is clicked, my quote machine should fetch a new quote and display it in the #textelement.
* **User Story #9:** My quote machine should fetch the new quote's author when the #new-quotebutton is clicked and display it in the #authorelement.
* **User Story #10:** I can tweet the current quote by clicking on the #tweet-quoteaelement. This aelement should include the "twitter.com/intent/tweet"path in it's hrefattribute to tweet the current quote.
* **User Story #11:** The #quote-boxwrapper element should be horizontally centered. Please run tests with browser's zoom level at 100% and page maximized.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

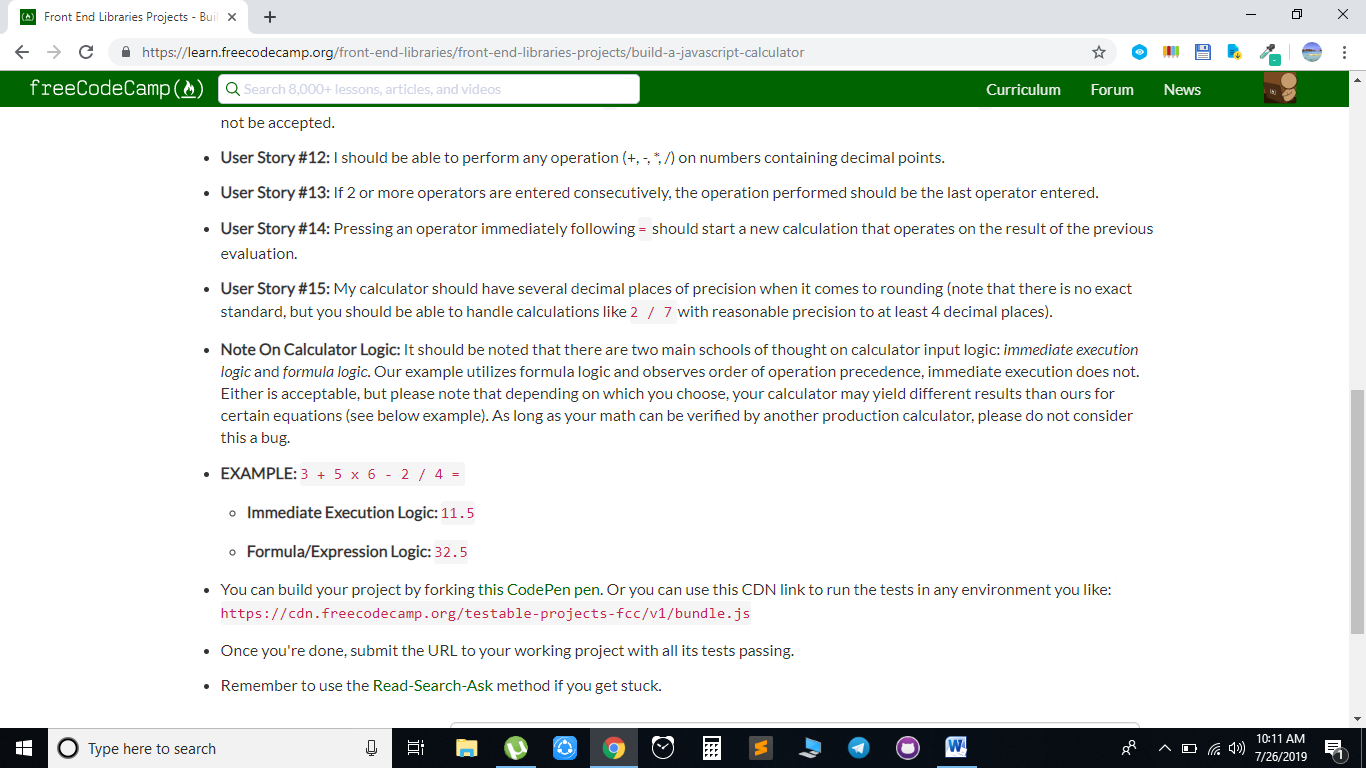
40. Front End Libraries Projects - Build a Markdown Previewer

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/GrZVVO>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use any mix of HTML, JavaScript, CSS, Bootstrap, SASS, React, Redux, and jQuery to complete this project. You should use a frontend framework (like React for example) because this section is about learning frontend frameworks. Additional technologies not listed above are not recommended and using them is at your own risk. We are looking at supporting other frontend frameworks like Angular and Vue, but they are not currently supported. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I can see a textareaelement with a corresponding id="editor".
* **User Story #2:** I can see an element with a corresponding id="preview".
* **User Story #3:** When I enter text into the #editorelement, the #previewelement is updated as I type to display the content of the textarea.
* **User Story #4:** When I enter GitHub flavored markdown into the #editorelement, the text is rendered as HTML in the #previewelement as I type (HINT: You don't need to parse Markdown yourself - you can import the Marked library for this: <https://cdnjs.com/libraries/marked>).
* **User Story #5:** When my markdown previewer first loads, the default text in the #editorfield should contain valid markdown that represents at least one of each of the following elements: a header (H1 size), a sub header (H2 size), a link, inline code, a code block, a list item, a blockquote, an image, and bolded text.
* **User Story #6:** When my markdown previewer first loads, the default markdown in the #editorfield should be rendered as HTML in the #previewelement.
* **Optional Bonus (you do not need to make this test pass):** When I click a link rendered by my markdown previewer, the link is opened up in a new tab (HINT: read the Marked.js docs for this one!).
* **Optional Bonus (you do not need to make this test pass):** My markdown previewer interprets carriage returns and renders them as br(line break) elements.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

41. Front End Libraries Projects - Build a Drum Machine

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/MJyNMd>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use any mix of HTML, JavaScript, CSS, Bootstrap, SASS, React, Redux, and jQuery to complete this project. You should use a frontend framework (like React for example) because this section is about learning frontend frameworks. Additional technologies not listed above are not recommended and using them is at your own risk. We are looking at supporting other frontend frameworks like Angular and Vue, but they are not currently supported. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I should be able to see an outer container with a corresponding id="drum-machine"that contains all other elements.
* **User Story #2:** Within #drum-machineI can see an element with a corresponding id="display".
* **User Story #3:** Within #drum-machineI can see 9 clickable drum pad elements, each with a class name of drum-pad, a unique id that describes the audio clip the drum pad will be set up to trigger, and an inner text that corresponds to one of the following keys on the keyboard: Q, W, E, A, S, D, Z, X, C. The drum pads MUST be in this order.
* **User Story #4:** Within each .drum-pad, there should be an HTML5 audioelement which has a srcattribute pointing to an audio clip, a class name of clip, and an id corresponding to the inner text of its parent .drum-pad(e.g. id="Q", id="W", id="E"etc.).
* **User Story #5:** When I click on a .drum-padelement, the audio clip contained in its child audioelement should be triggered.
* **User Story #6:** When I press the trigger key associated with each .drum-pad, the audio clip contained in its child audioelement should be triggered (e.g. pressing the Q key should trigger the drum pad which contains the string "Q", pressing the W key should trigger the drum pad which contains the string "W", etc.).
* **User Story #7:** When a .drum-padis triggered, a string describing the associated audio clip is displayed as the inner text of the #displayelement (each string must be unique).
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.

42. Front End Libraries Projects - Build a JavaScript Calculator

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/wgGVVX>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use any mix of HTML, JavaScript, CSS, Bootstrap, SASS, React, Redux, and jQuery to complete this project. You should use a frontend framework (like React for example) because this section is about learning frontend frameworks. Additional technologies not listed above are not recommended and using them is at your own risk. We are looking at supporting other frontend frameworks like Angular and Vue, but they are not currently supported. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** My calculator should contain a clickable element containing an =(equal sign) with a corresponding id="equals".
* **User Story #2:** My calculator should contain 10 clickable elements containing one number each from 0-9, with the following corresponding IDs: id="zero", id="one", id="two", id="three", id="four", id="five", id="six", id="seven", id="eight", and id="nine".
* **User Story #3:** My calculator should contain 4 clickable elements each containing one of the 4 primary mathematical operators with the following corresponding IDs: id="add", id="subtract", id="multiply", id="divide".
* **User Story #4:** My calculator should contain a clickable element containing a .(decimal point) symbol with a corresponding id="decimal".
* **User Story #5:** My calculator should contain a clickable element with an id="clear".
* **User Story #6:** My calculator should contain an element to display values with a corresponding id="display".
* **User Story #7:** At any time, pressing the clear button clears the input and output values, and returns the calculator to its initialized state; 0 should be shown in the element with the id of display.
* **User Story #8:** As I input numbers, I should be able to see my input in the element with the id of display.
* **User Story #9:** In any order, I should be able to add, subtract, multiply and divide a chain of numbers of any length, and when I hit =, the correct result should be shown in the element with the id of display.
* **User Story #10:** When inputting numbers, my calculator should not allow a number to begin with multiple zeros.
* **User Story #11:** When the decimal element is clicked, a .should append to the currently displayed value; two .in one number should not be accepted.
* **User Story #12:** I should be able to perform any operation (+, -, \*, /) on numbers containing decimal points.
* **User Story #13:** If 2 or more operators are entered consecutively, the operation performed should be the last operator entered.
* **User Story #14:** Pressing an operator immediately following =should start a new calculation that operates on the result of the previous evaluation.
* **User Story #15:** My calculator should have several decimal places of precision when it comes to rounding (note that there is no exact standard, but you should be able to handle calculations like 2 / 7with reasonable precision to at least 4 decimal places).
* **Note On Calculator Logic:** It should be noted that there are two main schools of thought on calculator input logic: *immediate execution logic* and *formula logic*. Our example utilizes formula logic and observes order of operation precedence, immediate execution does not. Either is acceptable, but please note that depending on which you choose, your calculator may yield different results than ours for certain equations (see below example). As long as your math can be verified by another production calculator, please do not consider this a bug.
* **EXAMPLE:** 3 + 5 x 6 - 2 / 4 =  
  + **Immediate Execution Logic:** 11.5
  + **Formula/Expression Logic:** 32.5
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.
* 

43. Front End Libraries Projects - Build a Pomodoro Clock

* **Objective:** Build a [CodePen.io](https://codepen.io/) app that is functionally similar to this: <https://codepen.io/freeCodeCamp/full/XpKrrW>.
* Fulfill the below [user stories](https://en.wikipedia.org/wiki/User_story) and get all of the tests to pass. Give it your own personal style.
* You can use any mix of HTML, JavaScript, CSS, Bootstrap, SASS, React, Redux, and jQuery to complete this project. You should use a frontend framework (like React for example) because this section is about learning frontend frameworks. Additional technologies not listed above are not recommended and using them is at your own risk. We are looking at supporting other frontend frameworks like Angular and Vue, but they are not currently supported. We will accept and try to fix all issue reports that use the suggested technology stack for this project. Happy coding!
* **User Story #1:** I can see an element with id="break-label"that contains a string (e.g. "Break Length").
* **User Story #2:** I can see an element with id="session-label"that contains a string (e.g. "Session Length").
* **User Story #3:** I can see two clickable elements with corresponding IDs: id="break-decrement"and id="session-decrement".
* **User Story #4:** I can see two clickable elements with corresponding IDs: id="break-increment"and id="session-increment".
* **User Story #5:** I can see an element with a corresponding id="break-length", which by default (on load) displays a value of 5.
* **User Story #6:** I can see an element with a corresponding id="session-length", which by default displays a value of 25.
* **User Story #7:** I can see an element with a corresponding id="timer-label", that contains a string indicating a session is initialized (e.g. "Session").
* **User Story #8:** I can see an element with corresponding id="time-left". NOTE: Paused or running, the value in this field should always be displayed in mm:ssformat (i.e. 25:00).
* **User Story #9:** I can see a clickable element with a corresponding id="start\_stop".
* **User Story #10:** I can see a clickable element with a corresponding id="reset".
* **User Story #11:** When I click the element with the id of reset, any running timer should be stopped, the value within id="break-length"should return to 5, the value within id="session-length"should return to 25, and the element with id="time-left"should reset to it's default state.
* **User Story #12:** When I click the element with the id of break-decrement, the value within id="break-length"decrements by a value of 1, and I can see the updated value.
* **User Story #13:** When I click the element with the id of break-increment, the value within id="break-length"increments by a value of 1, and I can see the updated value.
* **User Story #14:** When I click the element with the id of session-decrement, the value within id="session-length"decrements by a value of 1, and I can see the updated value.
* **User Story #15:** When I click the element with the id of session-increment, the value within id="session-length"increments by a value of 1, and I can see the updated value.
* **User Story #16:** I should not be able to set a session or break length to <= 0.
* **User Story #17:** I should not be able to set a session or break length to > 60.
* **User Story #18:** When I first click the element with id="start\_stop", the timer should begin running from the value currently displayed in id="session-length", even if the value has been incremented or decremented from the original value of 25.
* **User Story #19:** If the timer is running, the element with the id of time-leftshould display the remaining time in mm:ssformat (decrementing by a value of 1 and updating the display every 1000ms).
* **User Story #20:** If the timer is running and I click the element with id="start\_stop", the countdown should pause.
* **User Story #21:** If the timer is paused and I click the element with id="start\_stop", the countdown should resume running from the point at which it was paused.
* **User Story #22:** When a session countdown reaches zero (NOTE: timer MUST reach 00:00), and a new countdown begins, the element with the id of timer-labelshould display a string indicating a break has begun.
* **User Story #23:** When a session countdown reaches zero (NOTE: timer MUST reach 00:00), a new break countdown should begin, counting down from the value currently displayed in the id="break-length"element.
* **User Story #24:** When a break countdown reaches zero (NOTE: timer MUST reach 00:00), and a new countdown begins, the element with the id of timer-labelshould display a string indicating a session has begun.
* **User Story #25:** When a break countdown reaches zero (NOTE: timer MUST reach 00:00), a new session countdown should begin, counting down from the value currently displayed in the id="session-length"element.
* **User Story #26:** When a countdown reaches zero (NOTE: timer MUST reach 00:00), a sound indicating that time is up should play. This should utilize an HTML5 audiotag and have a corresponding id="beep".
* **User Story #27:** The audio element with id="beep"must be 1 second or longer.
* **User Story #28:** The audio element with id of beepmust stop playing and be rewound to the beginning when the element with the id of resetis clicked.
* You can build your project by forking [this CodePen pen](http://codepen.io/freeCodeCamp/pen/MJjpwO). Or you can use this CDN link to run the tests in any environment you like: https://cdn.freecodecamp.org/testable-projects-fcc/v1/bundle.js
* Once you're done, submit the URL to your working project with all its tests passing.
* Remember to use the [Read-Search-Ask](https://forum.freecodecamp.org/t/how-to-get-help-when-you-are-stuck/19514) method if you get stuck.