16.0 hours

Project Status
Ready for Submission

Project 3

Interactive Form

Are you ready to have your project reviewed? Before you submit your project, make sure you have double-checked your work. We want you to pass the first time!

If you get your project back and it needs work, you'll have to wait 1 day before you can resubmit your project.

Submit for Review

Project Instructions

To complete this project, follow the instructions below. If you get stuck, ask a question on Slack or in the Treehouse Community.

11 steps

1 Review Your Resources

Before you start the project, be sure to review the material and resources available to you.

- The "**Project Resources**" section on the right-hand side of this page contains links to optional project resources that can help you complete the project. Here is a description of some of the resources you will find there:
  - **Project Study Guide** - A list of links to relevant Treehouse videos, Practice Sessions, Project Warm Ups, and other resources.
  - **Project Walkthrough** - An optional step by step description of how to accomplish each phase of the project.
  - **Project Warm Ups** - Code exercises that will help prepare you for writing the code for this project.
  - **Practice Sessions** - Mini-workshops with guided practice opportunities for techniques relevant to the project.
- **Google** is a developer's best friend. Even the pros use it every day to check on any number of things. If you get stuck, try using your Google skills to find the information you need. If you aren't finding the information you need, try rephrasing your search.
- Your friendly **Slack** team is a helpful, friendly community of student developers of all levels working together to support one another on this journey. This is a great place to find guidance, advice, and encouragement. The "**Get Help**" button just below the "**Project Resources**" section will lead directly to your Slack team.

> **Pro Tip:** A good "Help!" post in Slack begins with a friendly greeting, details any important info related to the problem you're having, what you've tried so far to fix it, and most importantly, a link to your GitHub repository.

2 Getting Started with the Project Files

- If you haven't already, download the project's starter files, unzip them, add them to the root of your project folder.
- At the root of the project folder, create a folder named `js` and add a `script.js` file to it. This is where you'll write your JavaScript. Carefully review the `script` tag near the bottom of the `index.html` file. Check that the value in the `script` tag's `src` attribute matches the name of the folder and file that you created. This is necessary to connect the JavaScript to the HTML.
- Open the project folder in your text editor. Open and view the `script.js` file that you created. Load the `index.html` file in Chrome, and open the Chrome DevTools Console.
- To test that you have everything hooked up correctly, add `console.log('Test');` to the `script.js` file, save it, and refresh the page in the browser. You should now see `Test` printed in the console.
- Commit/push the updated project files to your GitHub repo.
- If you have trouble with any of the above steps, be sure to reach out to your Slack team.

3 The "Name" field

When the page first loads, the first text field should have the focus state by default to prompt the user.

- Set the `focus` property to `true` on the <input type="text"> element for the "Name" field.

4 "Job Role" section

The "Job Role" section has an <input type="text"> field where users can enter a custom job role. If the user selects "Other" in the "Job Role" drop down menu, they can enter info into the "Other job role" text field. But this field should be hidden by default and only displayed once users select "Other" in the drop down menu, and be hidden if the user selects any other option.

- Hide the "text field" with the `id` of "other-job-role" so it is not displayed when the form first loads.
- Program the "Job Role" <select> element to listen for user changes. When a change is detected, display/hide the "text field" based on the user's selection in the drop down menu.

5 "T-Shirt Info" section

The options in the "Color" drop down menu are not available for each t-shirt design. So the user shouldn't be able to see or choose a color option until they have chosen a design.

- Disable the "Color" <select> element.
- Program the "Design" <select> element to listen for user changes. When a change is detected:
  - The "Color" <select> element should be enabled.
  - The "Color" <select> element should display an available color.
  - The "Color" dropdown menu should display only the color options associated with the selected design. For example:
    - If the user selects "Theme - JS Puns" then the "Color" menu should only display "Cornflower Blue," "Dark Slate Grey," and "Gold."
    - If the user selects "Theme - I ♥ JS" then the "Color" menu should only display "Tomato," "Steel Blue," and "Dim Grey."

> **Important Note:** A `select` element is used for the color selection. There are two parts to a select element display: the element field and the drop down menu which opens after clicking on the field. Both the "Color" field **and** drop down menu must correctly update when the user selects a new theme. Neither should be empty or display unavailable colors.

> **Project Warm Up:** For some experience with the techniques you'll use in this section, complete this short exercise - **Selects and Options**.

> **Pro Tip:**
> - The `selected` attribute can determine which `option` element is displayed in the `select` field.
> - The `hidden` attribute can prevent `option` elements from being displayed in the drop down menu.

6 "Register for Activities" section

The "Total: $" element below the "Register for Activities" section should update to reflect the sum of the cost of the user's selected activities.

- Program the "Register for Activities" `fieldset` element to listen for user changes. When a change is detected:
  - If an activity is checked, the total cost should increase by the value in the `data-cost` attribute of the activity's <input type="checkbox"> element.
  - If an activity is unchecked, the total cost should decrease by that amount.
  - The <p> element with the id of "activity-cost" below the activities section should update to reflect the chosen activities' total cost.

> **Project Warm Up:** For some experience with the techniques you'll use in this section, complete this short exercise - **Checkboxes**.

7 "Payment Info" section

The credit card payment option should be selected for the user by default. So when the form first loads, "Credit Card" should be displayed in the "I'm going to pay with" <select> element, and the credit card payment section should be the only payment section displayed in the form's UI. And when the user selects one of the payment options from the "I'm going to pay with" drop down menu, the form should update to display only the chosen payment method section.

- Program the "I'm going to pay with" <select> element to listen for user changes. When a change is detected, hide all payment sections in the form's UI except the selected one.

8 Form validation

Users shouldn't be able to submit a form without the required information, or with invalid information. To prevent that from happening, **avoid using plugins, libraries, snippets or the built-in HTML5 validation, and create your own custom form validation**.

> **Note:** Form submission behavior will differ depending on whether you're running the project with a local server, or just viewing the files in the browser. It is recommended that you **view the files in the browser** instead of serving them locally. This helps facilitate development and testing, and this is how the project will be reviewed.

- Program the `form` element to listen for the `submit` event. When the form submission is detected, each required form field or section should be validated, or checked to ensure that they have been filled out correctly. If any of the following required fields is not valid, the form's submission should be prevented.
  - The "Name" field cannot be blank or empty.
  - The "Email Address" field must contain a validly formatted email address. The email address does not need to be a real email address, just formatted like one. For example: `dave@teamtreehouse.com`. A few characters for the username, followed by "@", followed by a few more characters and a ".com" for the domain name. You don't have to account for other top-level domains, like .org, .net, etc.
  - The "Register for Activities" section must have at least one activity selected.
  - **If and only if credit card is the selected payment method:**
    - The "Card number" field must contain a 13 - 16 digit credit card number with no dashes or spaces. The value does not need to be a real credit card number.
    - The "Zip code" field must contain a 5 digit number.
    - The "CVV" field must contain a 3 digit number.

> **Project Warm Up:** For some experience with the techniques you'll use in this section, complete this short exercise - **Form Input Validation**.

> **Note:**
> - Avoid using snippets, libraries or plugins.
> - Only validate the three credit card fields if "credit card" is the selected payment option.
> - Only call `preventDefault` on the `event` object if one or more of the required fields is invalid.

> **Pro Tip:**A recommended approach is to create helper functions for each of the required fields to be validated. For example, for the "Name" field, a function could check the "Name" field's value. If it equals an empty string or only blank spaces, the function could log out a helpful statement and return false. Otherwise it would return true. And then in the `submit` event listener, you could call that helper function and check what it returns: if it returns false, you would prevent the form from submitting. Otherwise, you would avoid preventing form submission, and allow the `submit` handler to either submit or move onto checking the next required field.

9 Accessibility

To craft web experiences that are accessible to all users, reach the largest possible audience, and are in compliance with web regulations, it's important to consider the needs of users with disabilities when presenting content. Some users require the help of assistive technologies like screen readers, and others aren't able to recognize prompts that are indicated by color changes alone. Complete the steps below to make your form more accessible.

- **Make the focus states of the activities more obvious to all users.** Pressing the tab key on your keyboard moves the focus state from one input to the next, but the focus indicators in the "Register for Activities" section aren't very obvious.
  - Program all of the activity `checkbox input` elements to listen for the `focus` and `blur` events.
    - When the `focus` event is detected, add the ".focus" className to the `checkbox input`'s parent `label` element.
    - When the `blur` event is detected, remove the `.focus` className from the `label` element that possesses it. It can be helpful here to directly target the element with the className of `.focus` in order to remove it.
- **Make the form validation errors obvious to all users.** With the custom form validation checks you've already written, invalid form fields will prevent the form from submitting, but all users should be presented with clear notifications of which fields are invalid.
  - When the user tries to submit the form, if a required form field or section is invalid:
    - Add the '.not-valid' className to the parent element of the form field or section. For the activity section, the parent element would be the `fieldset` element for the activity section. For the other required inputs, the parent element would be a `label` element for the input.
    - Remove the '.valid' className from the parent element of the form field or section.
    - Display the `.hint` element associated with the form field or section, which will be the last child of the parent element of the form field or section. The `parentElement` and `lastElementChild` properties will be helpful here.
  - If a required form field or section is valid:
    - Add the '.valid' className to the parent element of the form field or section.
    - Remove the '.not-valid' className from the parent element of the form field or section.
    - Hide the `.hint` element associated with that element.

> **Project Warm Up:** For some experience with the techniques you'll use in this section, complete this short exercise - **Input Validation Error Indications**.

> **Note:**
> - Error messages should not be visible by default or when the form first loads.
> - JavaScript alerts and prompts should not be used in your form validation error indications.
> - If your user tries to submit an empty form, all form validation error indications should be displayed at once, rather than one at a time.

> **Pro Tip:** A recommended approach to this part of the project is to create helper functions that accept an argument for the element that is being validated. For example, the function could accept an argument for the text input element that was checked. Then the function would update the styles for that element's parent element and the last child of that parent element. One function could update the styles when errors are detected. And another function could update the styles when errors are resolved.

10 Finishing the Project

The final stage of the project is perhaps the most important. This is where your developer skills really shine as you carefully double-check that you've accomplished all requirements and that your project is ready for submission.

- **Code Comments** - It's a best practice for development code to be well commented. Replace provided comments with your own to briefly describe your code and what it does.
- **Code Readability** - Readability is second only to functionality. Double-check your code to ensure the spacing and indentation is consistent and in keeping with best practices.
- **Quality Assurance Testing** - This is a key step in the development process.
  - Open and run your app.
  - Open the Chrome DevTools console and ensure that there are no errors displayed when the app is being used and tested.
  - Pretend to be a user and test all aspects of functionality and every possible state of the app, while monitoring the console for bugs and resolving any that arise.
- **Cross Browser Consistency** - To pass, your project only needs to work in Chrome but it's common for developers to test their projects in multiple browsers to know how they will perform out in the wild.

11 Before Submitting the Project

Before you submit your project, check off each item in the project submissions checklist below.

- [ ] I have read all of the project instructions, including the "**How you'll be graded**" section for this project.
- [ ] I understand what is needed to receive a Meets or Exceeds Expectations grade and asked for clarification about grading requirements on Slack if necessary.
- [ ] My GitHub repo for this project contains only this project, only files needed to make this project run, and a `README.md` file providing details about my project.
- [ ] I wrote all of my own code for this project. Any code included in my project that I did not write myself is appropriately attributed to its source.
- [ ] I have completed all of the project requirements and believe the project is ready to receive a Meets or Exceeds Expectations grade.
- [ ] I have received a preliminary review of my project in Slack to catch anything I might have missed.
- [ ] I understand that in order to receive an Exceeds Expectations grade, I must complete all extra credit items.
- [ ] I understand that what I submit is what will get reviewed and that when I submit my project, any changes I make after the submission won't be seen by my reviewer.

Extra Credit

To get an "exceeds" rating, complete all of the steps below:

3 steps

1 Prevent users from registering for conflicting activities

Ideally, we want to prevent users from selecting activities that occur at the same time.

- When a user selects an activity, loop over all of the activities, check if any have the same day and time as the activity that was just checked/unchecked, and as long as the matching activity is not the activity that was just checked/unchecked, disable/enable the conflicting activity's `checkbox input` and add/remove the '.disabled' className to activity's parent `label` element.

2 Real-time error message

Providing form validation error indications at the moment they occur better serves your user.

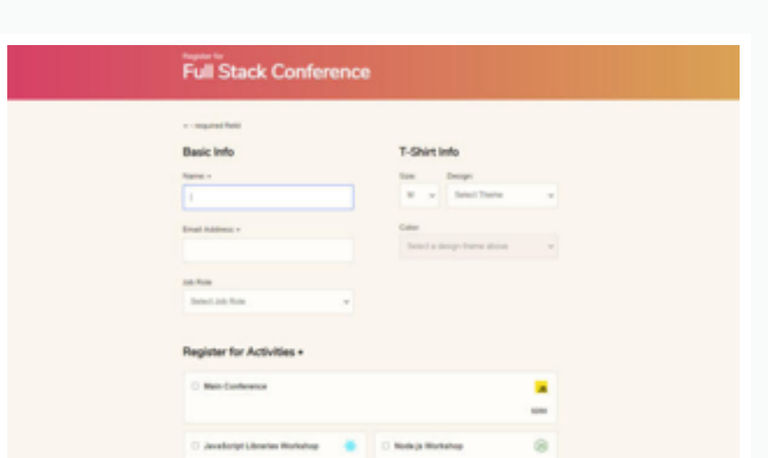
- Program at least one of the required fields to listen for user interaction like a keyup. When then user interaction occurs, run the validation check for that input. If you created helper functions to validate the required form inputs and sections, you can call those helper functions inside of a field's event listener.
- Detail this specific feature in your `README.md` file.

3 Conditional error message

Providing additional information for certain types of errors can be very helpful to your user. For example, if the email address field is empty, it would be enough to inform the user that they should add an email address. But if they've already added an email address, but formatted it incorrectly, that message wouldn't be helpful.

- For at least one required form section, provide one error message if the field fails on one of its requirements, and a separate message if it fails on one of its other requirements.
- Detail this specific feature in your `README.md` file.

> **NOTE: Getting an "Exceed Expectations" grade.**
> - See the rubric in the "**How You'll Be Graded**" tab above for details on what you need to receive an "Exceed Expectations" grade.
> - Passing grades are final. If you try for the "Exceeds Expectations" grade, but miss an item and receive a "Meets Expectations" grade, you won't get a second chance. Exceptions can be made for items that have been misgraded in review.
> - Always mention in the comments of your submission or any resubmission, what grade you are going for. Some students want their project to be rejected if they do not meet all Exceeds Expectations Requirements, others will try for all the "exceeds" requirement but do not mind if they pass with a Meets Expectations grade. Leaving a comment in your submission will help the reviewer understand which grade you are specifically going for



Download files
Zip file

Project Resources

Instruction
Project Study Guide

Instruction
Project Warm Ups

Reference
Your First HTML Form

Reference
Forms in HTML

Reference
Form Validation

External Link
Client-side Validation

Instruction
Project Walkthrough

Need Help?

Have questions about this project? Start a discussion with the community and Treehouse staff.

Get Help