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

- Instructions
- How you'll be graded

In this project, you'll use JavaScript to enhance an interactive registration form for a fictional conference called "FullStack Conf."

Using the supplied HTML and CSS files, you'll add your own JavaScript to make the form more user-friendly by:

- adding customized and conditional behavior and interactivity
- validating user input and providing helpful error messages when the user enters invalid information into the form fields.

Instead of plain "vanilla" JavaScript, you'll use the popular jQuery library to complete this project. While it's important for a developer to have a solid understanding and familiarity of plain vanilla JavaScript, it's also important to be able to understand and work with jQuery since it is so common and prevalent on the web.

The benefits of using jQuery are:

- Shorter and simplified syntax
- Help with cross-browser compatibility since jQuery deals with a lot of that stuff under the hood
- A strong familiarity with jQuery will only make you a better developer since it enjoys a strong market share on the web

After completing this project, you'll have a solid foundaitonal knowledge of using forms in your projects. This is tremendously valuable since forms are a major aspect of most web sites and web applications, which makes this project a key addition to your portfolio, and a crucial step on your journey to becoming a professional full stack web developer.

Before you start

To prepare for this project you'll need to make sure you complete and understand these steps.

4 steps

- **Review the policy on Reusing Code in a Techdegree project.**
- **GitHub**
  - Have a GitHub account and create a new repo for this project.
  - Create a README.md file for your repo that explains what the projects is and anything your user or fellow developers might need to know to use the project.
- **Download the project files**
  - You've been supplied several files for this project:
    - `index.html` - contains the initial HTML page. Ideally, you want to make as few changes to this file as possible. You'll have to add the necessary script tags to the link your JS, and you'll need to modify the HTML for the "Job Role" task below. But other than that, try to do everything you can with JavaScript, rather than modifying the HTML.
    - `css/style.css` - contains the styles for this project. Feel free to play around with some of the styles here to make it your own. You can experiment with things like the background color, font, transitions, animations, box shadows and text shadows. But the basic layout and positioning of elements should not be changed.
    - A `js` folder where you can store the JS file you need to create for this project, and the jQuery source files if you intend to include them in the project directory instead of linking to them with a CDN (Content Delivery Network) link.
- **Follow the Instructions Below**
  - Be sure to reach out on Slack if you get stuck or run into difficulties.

Project Instructions

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

15 steps

- **Add the necessary files**
  - `js/script.js` - Create a JavaScript file inside the "js" folder and link it to index.html
  - `jquery` - Link index.html to the latest version of jQuery either by including the jQuery files directly in the "js" folder, or by linking to a jQuery CDN. You should be able to find both with a good Google search, but if you get stuck or have questions, be sure to reach out on Slack.

    - `css/reset.css` or `css/nomralize.css` - This is not a project requirement, but using a reset or normalize CSS file is an important common standard in web development as it helps zero out the initial styles that different browsers add to their page by default. So you are encouraged to take this extra step. You should be able to find what you need with a Google search.

- **jQuery**

    - Add jQuery to your project by either including the jQuery source files in your directory or by using a CDN link.
    - Utilize jQuery while coding the functionality for your form.

  Note: You don't have to use jQuery for every single aspect of your project, but at the very least, your project should include and utilize at least some jQuery. Although, you should keep in mind that consistency is important. Typically, projects are built with either a jQuery or a "vanilla" JavaScript approach. And it's not a great idea to continuously bounce back and forth between the two in a single project.

- **Set focus on the first text field**

    - When the page first loads, the first text field should be in focus by default.

- **”Job Role” section**

    - Include a text field that will be revealed when the "Other" option is selected from the "Job Role" drop down menu.
    - Give the field an id of “other-title,” and add the placeholder text of "Your Job Role".

  Note: You'll need to add the "Other" job role input directly into the HTML and hide it initially with JS in order to get this feature to work when JS is disabled, which is a requirement below.

- **”T-Shirt Info” section**

    - For the T-Shirt "Color" menu, only display the color options that match the design selected in the "Design" menu.
    - 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."
    - When a new theme is selected from the "Design" menu, the "Color" field and drop down menu is updated.

- **”Register for Activities” section**

    - Some events are at the same day and time as others. If the user selects a workshop, don't allow selection of a workshop at the same day and time -- you should disable the checkbox and visually indicate that the workshop in the competing time slot isn't

available.
- When a user unchecks an activity, make sure that competing activities (if there are any) are no longer disabled.
- As a user selects activities, a running total should display below the list of checkboxes. For example, if the user selects "Main Conference", then Total: $200 should appear. If they add 1 workshop, the total should change to Total: $300.

- **"Payment Info" section**

  - Display payment sections based on the payment option chosen in the select menu.
  - The "Credit Card" payment option should be selected by default. Display the `#credit-card` div, and hide the "PayPal" and "Bitcoin" information. Payment option in the select menu should match the payment option displayed on the page.
  - When a user selects the "PayPal" payment option, the PayPal information should display, and the credit card and “Bitcoin” information should be hidden.
  - When a user selects the "Bitcoin" payment option, the Bitcoin information should display, and the credit card and “PayPal” information should be hidden.

  NOTE: The user should not be able to select the "Select Payment Method" option from the payment select menu, because the user should not be able to submit the form without a chosen payment option.

- **Form validation**

  - If any of the following validation errors exist, prevent the user from submitting the form:
    - Name field can't be blank.
    - Email field must be a validly formatted e-mail address (you don't have to check that it's a real e-mail address, just that it's formatted like one: `dave@teamtreehouse.com` for example.
    - User must select **at least** one checkbox under the "Register for Activities" section of the form.
    - If the selected payment option is "Credit Card," make sure the user has supplied a Credit Card number, a Zip Code, and a 3 number CVV value before the form can be submitted.
      - Credit Card field should only accept a number between 13 and 16 digits.
      - The Zip Code field should accept a 5-digit number.
      - The CVV should only accept a number that is exactly 3 digits long.

  NOTE: Don't rely on the built in HTML5 validation by adding the `required` attribute to your DOM elements. You need to actually create your own custom validation checks and error messages.

  NOTE: Avoid using snippets or plugins for this project. To get the most out of the experience, you should be writing all of your own code for your own custom validation.

  NOTE: Make sure your validation is only validating Credit Card info if Credit Card is the selected payment method.

- **Form validation messages**
    - Provide some kind of indication when there's a validation error. The field's borders could turn red, for example, or even better for the user would be if a red text message appeared near the field.
    - The following fields should have some obvious form of an error indication:
        - Name field
        - Email field
        - Register for Activities checkboxes (at least one must be selected)
        - Credit Card number (Only **if** Credit Card payment method is selected)
        - Zip Code (Only **if** Credit Card payment method is selected)
        - CVV (Only **if** Credit Card payment method is selected)

  Note: Error messages or indications should not be visible by default. They should only show upon submission, or after some user interaction.

- **Form works without JavaScript - Progressive Enhancement**
    - The user should still have access to all form fields and payment information if JS isn't working for whatever reason. For example, when the JS is removed from the project:
        - The "Other" text field under the "Job Role" section should be visible
        - All information for Bitcoin, PayPal or Credit Card payments should be visible.

- **CSS styles**
    - It is not required, but you are encouraged to experiment with things like the color, background color, font, transitions, animations, box shadows and text shadows. So dive into the CSS file and see if you can make this project your own with a few adjustments to the styles. But the basic layout and positioning of elements should not be changed.

- **Add good code comments**

- **Cross-Browser consistency**
    - Get in the habit of checking your project in multiple browsers. But to pass, the project only needs to work in Chrome.

- **Review the "How you'll be graded" section.**
    - Check out the "How you'll be graded" section, located above, next to the instructions tab, just below the project title. This section lists specifically what will be considered and checked when your project is being reviewed, and your project grade is being determined.

- **Quality Assurance and the Student Project Submission Checklist**
    - Web development work bears the need for a high level of precision. We're talking about

an industry that measures performance by the pixel, kilobyte, and millisecond. So it's very important to pay attention to the details and take the time to do your own thorough quality assurance testing on all your own projects. Before you submit your project for review, please do be sure to check off all of the items on the Student Project Submission Checklist. The checklist is designed to help make sure you've met the grading requirements, that your project is complete and ready to be submitted, and that you are developing good habits as a developer!

NOTE: Sometimes just getting started is the hardest part.

- It's not uncommon to feel overwhelmed and confused when beginning to build a project. If you feel this way, try not to get too focused on seeing the project as a whole. Instead, just take it one small piece at a time. After familiarizing yourself with the instructions, start by downloading the source files, and creating a GitHub repo to store them. That is always a great place to start. Then just start tackling the project one small step at a time. Remember, your first attempt isn't likely to be perfect, and it doesn't have to be. As coders, we focus first on creating something that works. And then we refactor and optimize on later iterations.

NOTE: Seeking assistance

- If you're feeling stuck or having trouble with this project
  - Reach out to the team on Slack.
  - Review material in the unit.
  - Practice your Google skills by finding different ways to ask the questions you have, paying close attention to the sort of results you get back depending on how your questions are worded.

NOTE: What you submit is what will get reviewed.

- When you submit your project, a snapshot is taken of your repository, and that is what the reviewer will see. Consequently, any changes you make to your repo after you submit will not be seen by the reviewer. So before you submit, it's a smart idea to do a final check to make sure everything in your repo is exactly what you want to submit.

Extra Credit

To get an "exceeds" rating, you can expand on the project in the following ways:

3 steps

- **T Shirt Section**
  - Hide the "Color" label and select menu until a T-Shirt design is selected from the "Design" menu.
- **Conditional Error Message**

- Program at least one of your error messages so that more information is provided depending on the error. For example, if the user hasn't entered a credit card number and the field is completely blank, the error message reads "Please enter a credit card number." If the field isn't empty but contains only 10 numbers, the error message reads "Please enter a number that is between 13 and 16 digits long."

- **Real-time Error Messages**

  - Program your form so that it provides a real-time validation error message for at least one text input field. Rather than providing an error message on submit, your form should check for errors and display messages as the user begins typing inside a text field. For example, if the user enters an invalid email address, the error appears as the user begins to type, and disappears as soon as the user has entered a complete and correctly formatted email address. **You must accomplish this will your own JavaScript code**. Do not rely on HTML5's built-in email validation.

NOTE: If you implement the above exceeds requirements in your form, make sure you detail in your submission notes which input will have different error messages depending on the error, and which input will have "real time" validation messages, so your reviewer won't miss them by accident.

- **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

Workshop

Share Your Techdegree Projects with Github Desktop

Reference

Your First HTML Form

Reference

Forms in HTML

Reference

Form Validation

External Link

Clientside Validation

Reference

The HTML <input> field

Need Help?

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

Get Help