# 1.1: Getting Started with Web Development

 Estimated Task Time: 1 - 3 hours.

In this Task, you’ll be using an Agile project management tool such Trello, Jira, or Ora to create a board that includes all the to-dos of your study and project schedule for this course. Doing so will set you up for success and allow you to plan for the tasks you’ll need to complete. It will also get you familiar with using planning tools, which is something you’ll be doing on a daily basis as a web developer.

**Directions**

1. Select a project management tool and create an account. We recommend either [Trello](https://trello.com/en?truid=tr100d25-cdfc-0612-96b7-98151e4c5470), [Jira](https://www.atlassian.com/software/jira), or [Ora](https://ora.pm/). Take a look at each one and choose whichever feels most intuitive to you. Once you've chosen, take some time to familiarize yourself with the tool and its features before moving on.
2. Create a board that includes at least these 3 columns with these statuses:
   * To-do
   * In-Progress
   * Done
3. Refer to the [project brief](https://images.careerfoundry.com/public/courses/intro-to-frontend/A1_WDv2_Project%20Brief.pdf) and take a quick look at the content coming up in future Exercises. Then, create “tickets” for your study and project schedules. (For example, you may want to create a ticket for reading through each Exercise, for doing each quiz, etc.)
4. Take a screenshot of your finished board and upload it here. Feel free to share additional thoughts or ask questions on your submission page.

**Bonus Task 1:**

Try writing a few user stories to go along with your project tickets; for example, “As a hiring manager, I want to see links to an applicant’s work examples so that I can easily evaluate them.” You can also group related tickets in various ways depending on which tool you selected. Play around with the tools a bit to find the best layout for you.

**Bonus Task 2:**

Think you already have a feel for how much effort some of your tasks will take? Give story points a try! For each ticket, estimate the time and effort for the task using story points. Don’t worry too much about estimating correctly. You’ll always be able to re-estimate as you progress through the course!

TIP!  
We suggest you save different versions of your deliverables with different file names. You can add v1, v2, or v3 to a filename (e.g., Exercise 4.2-Building a Mood Board-v1). Alternatively, you can add the completion date of a specific version of your deliverable to the filename (e.g., Exercise1.4-Building a Mood Board-09/06/19). This way, you'll be able to present the evolution of your projects in your portfolio or present iterations of your projects in an interview.

# 1.2: HTML and the Web

 Estimated Task Time: 1 - 3 hours.

Ready to create the HTML skeleton of your new portfolio site? Before you write any code, you need to find a good text editor. While any text editor would technically work, the best are those with convenient code-writing features such as color coding and auto-indenting. We recommend [Atom](https://atom.io/), but you can also check out other popular text editors such as [Sublime Text](https://www.sublimetext.com/) and [VS Code](https://code.visualstudio.com/). Choose what works best for you!

If you get stuck when working on the Task, try using the Web Inspector to see how elements are presented on another page.

**Part 1**

1. Open a new file in your text editor and set up your HTML document. Remember, you first need a declaration to tell the browser that this is an HTML document: <!DOCTYPE html>.
2. Next, add the <html>, <head>, and <body> tags. Don’t forget to close the tags! Your <head> and <body> elements should be nested inside of your <html> element.
3. Inside the <head> element, you only need to add the <title> element for now, so pick a title you like and add it to the code.
4. Within the body tag, just write “Hello World” for now.
5. Save your file as “home.html”.
6. Once you’ve saved your file, open it in your web browser:
   * To do so, double-click on the file in your file explorer, and it should open in your default web browser.
   * If Chrome isn’t your default web browser, change it to Chrome or right-click on the file and select “Open with” → “Chrome.”
7. Add a favicon to your HTML file inside of the <head> element using the code from the Exercise. (Hint: look for the <link> tag. This can be the same [favicon Wikipedia uses](http://wikipedia.com/static/favicon/wikipedia.ico) for now.)
8. After adding your favicon, make sure you save your file. Then, go back to your browser and refresh the site to see your changes. You should now see the favicon on top. Congratulations—you’ve created your first simple HTML site!

**Part 2**

1. Take the image of the [portfolio site design](https://images.careerfoundry.com/public/courses/intro-to-frontend/E2_NEW/example_homepage_plain.png) from above and annotate as many of the HTML tags as you can to create the basic HTML structure. (Tip: To annotate screens, you can use any program you like. We recommend [Skitch](https://evernote.com/products/skitch" \t "_blank) for Mac, [Jing](https://www.techsmith.com/jing-tool.html), or the built-in snipping tool for Windows.)
2. Once you’re happy with your code and annotations, save both files and compress them into a single zip file. (If you’re not sure how to compress multiple files into a single zip file, check out these [instructions on zipping files](https://www.nebraska.gov/NREC/real/realzipfiles.html).) Then, upload the file here. Feel free to share additional thoughts or ask questions on your submission page.

**Bonus Task:**

1. Want to use your own favicon instead of the Wikipedia one? Go ahead and make your own! Once your image is ready, [use Favic-o-matic](http://www.favicomatic.com/) to ensure it’s “favicon-ready.”
2. Feel like you’re already ready to move beyond annotating and want to write the HTML structure itself? Go ahead and give it a try! Add the HTML structure to your new “home.html” file and submit it along with the rest of your work. You’ll review it at the start of the next Exercise.

# 1.3: HTML Semantics

Estimated Task Time: 1 - 3 hours.

In this Task, you’ll create additional pages for your portfolio site using the recommended structure discussed in the Exercise. Remember that you can always preview your site by double-clicking your HTML file from a file explorer and opening it directly in your default browser.

**Directions**

1. Create a folder structure for your website:
   * The structure should include a folder for your images and the upcoming CSS files.
   * Rename your HTML file from the previous Exercise to “index.html”.
2. Add additional HTML pages to the new folder.
   * Create a “contact.html” page.
   * Create an “about.html” page.
3. Build a header with a logo and a set of navigation links, then copy-paste it in all your files.
   * Add the HTML5 <header></header> element to “index.html” and add your logo and navigation inside it.
   * Create a navigation using the HTML5 tag <nav></nav> inside the header and copy the whole header over to the contact and about pages.
4. Add contact information to your contact page.
   * Add a linked email address.
   * Add a linked telephone number or voice/video chat contact information.
   * Add a call-to-action link.
5. Update your footer.
   * Add SVG icons for your “Find me on” section in the footer.
   * Replace the div tags with HTML5 <footer> tags.
6. Once you’re happy with your code, save it and create a zip file of all the folders within your new website’s file structure.
7. Upload your zip file here. Feel free to share additional thoughts or ask questions on your submission page.

NOTE!  
Make sure all your HTML files are properly indented!

**Bonus Task:**

Looking for additional practice with forms? Create a [CodePen](https://codepen.io/" \t "_blank) with a contact form. Save the CodePen and add the link to your submission. You’ll be able to add the contact form to your portfolio page after Achievement 3, once you’ve learned about server-side programming.

# 1.4: Advanced HTML & Web Accessibility

 Estimated Task Time: 1 - 3 hours.

In this Task, you’ll include additional content in your about and contact pages and make your portfolio site project accessible. Remember—to preview your site, just double-click your HTML file from the Finder window or from your file explorer to open it directly in your default browser.

**Directions**

1. Add a form to your contact page:
   * Add a contact form to your “contact.html” page (you can also reuse the one from the bonus task in your CodePen).
   * Validation: Make sure your form includes HTML5 form validation.
   * Accessibility: Make sure your form includes labels for each input field.
2. Add content to your about page:
   * Add some information about yourself to the about page.
   * Add a table to your about page (e.g., regarding your skills, years of experience, etc.).
   * Accessibility: Make sure the content is structured semantically.
   * Accessibility: Make sure the data tables are using all necessary tags (e.g., thead, tbody, etc.).
3. Make your portfolio site project accessible:
   * Add the accessibility checker “[tota11y](http://khan.github.io/tota11y/)” to your site and try to remove all errors. To do so, follow the steps provided earlier in the Exercise to create a new “tota11y.min.js” file and link to it in your HTML files. *https://github.com/Khan/tota11y/releases/tag/0.1.6*
   * Add alt text to all of your images.
   * Add role attributes to your navigation.
   * Make sure you use all semantic HTML tags (e.g., <footer>, <header>, <nav>, etc.).
4. Once you’re happy with your code, save it and create a zip file of all the folders within your website’s structure.
5. Upload your zip file here. Feel free to share additional thoughts or ask questions on your submission page.

# 1.5: CSS & The Look of Your Website

 Estimated Task Time: 1 - 3 hours.

In this Task, you’ll begin styling your website by creating and linking a CSS style sheet to your HTML files. Note that if you’ve linked your CSS file properly, you should be able to double-click your HTML file to open it directly in your default browser. You can then see all the styling you’ve applied. When you add something to your CSS file, you first need to save it, then refresh your browser to see the change.

**Directions**

1. Create a new file called “styles.css” in the CSS folder of your project.
2. Link your new CSS file to the index, about, and contact page HTML files of your portfolio site.
3. Add styling to your portfolio project page:
   * Reset the default browser stylings.
   * Define your colors.
   * Pick at least one font from [Google Fonts](https://fonts.google.com/).
   * Define font sizes and line heights for default text elements (body, headers, etc.).
   * Create link stylings with different stylings for normal, hover, active, and visited states.
   * Include at least one button somewhere on your page and give it stylings for normal and hover states (e.g., decrease its opacity and have the mouse turn into a hand pointer on hover).
4. Once you’re happy with your code, save it and create a zip file of the folders within your new website structure.
5. Upload your zip file here. Feel free to share additional thoughts or ask questions on your submission page.

# 1.6: Advanced CSS Layout

Estimated Task Time: 4 - 6 hours.

In this Task, you’ll be using CSS to style the layout of your HTML file. Remember that you can preview your site by double-clicking on your HTML file to open it directly in your default browser. You’ll then be able to see all the styling you’ve done. When you add something to your CSS file, make sure you save it before refreshing your browser. Otherwise, the changes won’t take effect.

**Directions**

1. Create a layout for your portfolio page.
   * Replace the placeholder logo with a real logo and add some spacing.
   * Set the box-sizing to border-box on all your elements.
   * Create a Flexbox layout for your header bar.
   * Align your profile image and your welcome text.
2. Add a new page for your work examples and create a grid layout.
   * Copy the contents of one of your other HTML pages to use as the baseline for your “work.html” file.
   * Add a link to “work.html” to your navigation.
   * Add a grid layout to “work.html” with at least 4 grid child items.
3. Add a layout for your footer.
   * Pick any of the layout methods you just learned to align your footer.
   * Center the footer's content.
4. Once you’re happy with your code, save it and create a zip file of the folders within your new website structure.
5. Upload your zip file here. Feel free to share additional thoughts or ask questions on your submission page.

**Bonus Task:**

Play the games mentioned in the text to learn the basics of Flexbox and the new CSS grid. Take a screenshot when you reach the last level and include the screenshots in your zip file.

Rubric

Refer to the categories below to see how to meet the requirements of the approved stage

# 1.7: Advanced Topics in CSS

 Estimated Task Time: 4 - 6 hours.

In this Task, you’ll use advanced CSS to add transitions and animations to your about page as well as make it responsive. If you’d prefer to work on your home and/or contact pages, this is also fine, so long as you add transitions and animations to a minimum of 1 page on your site.

Remember that you can preview your site by double-clicking on your HTML file to open it directly in your default browser. You’ll then be able to see all the styling you’ve done. When you add something to your CSS file, make sure you save it before refreshing your browser. Otherwise, the changes won’t take effect.

**Directions**

1. Add a @media query for smaller screen sizes.
   * Adjust your 4-column grid to a 2-column or 1-column grid.
   * Make the heading 1 smaller.
   * Experiment with different screen sizes to make sure your pages are responsive.
2. Select one page of your website for more advanced CSS styling (e.g., the about page).
3. Add CSS transitions to your selected page.
   * Add a transition to your buttons.
   * Optional: Add transitions to your links.
4. Add an illustration to your page and animate it.
   * Suggestion: Pick an SVG illustration of your hometown.
   * Add one animation to the illustration with CSS.
   * Optional: Add multiple animations with rotate, scale, translate, etc.
5. Once you’re happy with your code, save it and create a zip file of the folders within your new website structure.
6. Upload your zip file here. Feel free to share additional thoughts or ask questions on your submission page.

**Bonus Task**:

Take a look at Foundation or Bootstrap and create an interactive prototype.

1. Think of a process you want to make a prototype for (e.g., registering for an online service, signing up for a newsletter, etc.).
2. The prototype should consist of at least 3 steps that are linked to each other (e.g., a homepage with a register button, a registration page, a confirmation page, etc.).
3. Create a hand-drawn sketch of the steps.
4. Use a CSS framework to create an interactive prototype for your steps.
5. Add the prototype to your zip file.

# 1.8: CSS Preprocessors & CSS Variables

 Estimated Task Time: 4 - 6 hours.

In this Task, you’ll do two things: practice working with SCSS and add CSS variables to your project.

Remember that you can preview your site by double-clicking on your HTML file to open it directly in your default browser. You’ll then be able to see all the styling you’ve done. When you add something to your CSS file, make sure you save it before refreshing your browser. Otherwise, the changes won’t take effect.

**Step 1: Rewrite the CSS in this CodePen as SCSS.**

1. Open this [SCSS CodePen](https://codepen.io/2b8efbb7-02ca-46dd-8f04-d3fa33ffa54f/pen/mKKwKq) and fork it (you can fork a CodePen by using the “Fork” button in the top-right navigation bar). This creates a link that you can upload later once you’re done with the Task.
2. Rewrite all the plain CSS in the CodePen as SCSS. The CodePen itself is already set to SCSS, so you can simply start writing your SCSS. If something’s not working or your changes aren’t visible, you might have written invalid SCSS. You can check by pasting your SCSS into SassMeister.
   * Replace all colors with color variables.
   * There should be a lighter and a darker version of the primary color, which you should use as background-colors for the boxes (see class names).
   * Add a :hover pseudo class to the boxes (in a nested style) that removes the box-shadow on hover.
   * Create a parent class (shared class defined with %) that gets extended for all boxes and includes all the properties that the boxes share.

**Step 2: Add CSS variables to your own project’s CSS file.**

1. Now, head back to your own project and the “styles.css” file for your portfolio page.
2. Replace all your colors with *native CSS color variables*.
3. Replace any other repetitive values (e.g., font-weight, font-size, etc.) as variables, as well.
4. Postprocess your CSS file once you’re done.
   * Copy the content of your CSS file and paste it into the [PostCSS playground](https://madlittlemods.github.io/postcss-css-variables/playground/" \t "_blank) for converting variables.
   * Copy the output of the PostCSS playground and paste it into the [autoprefixer](http://autoprefixer.github.io/" \t "_blank).
   * Create a new file called “style.production.css” and paste the output of the autoprefixer into the file.
   * **Important:** Make sure you keep (and turn in) not only your new “style.production.css” file but also your “styles.css” file. Any future changes to your CSS should be made through your original “styles.css” files, then this process should be repeated again to create a new “style.production.css” file.
   * Update the links to your CSS file in all your HTML files (they should now point to your “style.production.css” file).

**Step 3:**

Once you’re happy with your code, save it and create a zip file for your project.

**Step 4:**

Upload your zip file and the link to the forked CodePen here. Feel free to share additional thoughts or ask questions on your submission page.

# 1.9: Git & Version Control

 Estimated Task Time: 1 - 3 hours.

In this Task, you’ll set up a GitHub account for your project and make some changes to your website before pushing them to GitHub. You’ll also work on finishing up your portfolio website by replacing any remaining placeholder text and images with your own content.

IMPORTANT NOTE!  
If, at this point in the course, you're uncomfortable uploading your real name and photo to your portfolio site, you're free to use an alias and placeholder image. We know that many of our students are already working jobs while taking this course, and that releasing info about wanting to change careers could lead to trouble with a current employer. If you're worried about this, you're free to use an alias until you're ready to start promoting your site and begin your job search. In this case, please still follow the directions for the Task as closely as possible, but switch out any mention of your real name with an alias (and find a stock photo or other image to use as your profile picture). This way, once you're ready to release your portfolio site into the wild at the end of the course, you'll be able to quickly switch in your real information.

**Directions**

1. Set up a [GitHub account](https://github.com/).
2. Download and install [GitHub Desktop](https://desktop.github.com/).
3. Create a new project for your website in GitHub Desktop.
4. Publish your local repository to an online repository called “website-portfolio” (or something similar) on GitHub.
5. Make some real changes to your project:
   * Change the page title of your “contact.html” file to something indicating that this is a contact page, for example “Contact Lisa Gringl,” then commit that change using a reasonable commit message.
   * Change the page title of your “about.html” file in the same way.
   * If you haven’t done so yet, replace the placeholder titles, logo (text or image), and project images in all your HTML files.
   * If you haven’t done so yet, finalize the text on your pages, as well.
6. Commit the changes you’ve made to your project. Make sure to follow the standards you learned when creating commit messages.
7. Push your changes to GitHub.
8. Submit the GitHub repository link to your project here. Feel free to share additional thoughts or ask questions on your submission page.

# 1.10: Code Quality, Testing, and Web Hosting

 Estimated Task Time: 4 - 6 hours.

In this Task, you’ll add code linters for HTML and CSS to your development environment (i.e., your text editor, Atom) and fix any errors and warnings that pop up. Then, you’ll test your portfolio page in all modern browsers. Finally, you’ll upload your portfolio site to a web server to make it publically available.

**Directions**

1. Add code linters to your development environment:
   * At least one of your choice for HTML and CSS
   * The W3C code linter
2. Run your HTML code through the W3C validator.
3. Find all warnings and errors and fix them.
4. Decide on a web-hosting provider.
   * This can either be a paid provider or a free one (like GitHub Pages).
5. Upload your portfolio to your web server.
6. Cross-browser test your live website. We recommend using [Cross Browser Testing](https://crossbrowsertesting.com/), as the 100 free minutes of manual testing they provide are more than enough to test your portfolio site in the four mentioned browsers.
   * Test in all modern browsers: IE Edge, Safari, Firefox, and Chrome.
   * Compile a list of issues you want to fix.
   * Fix the issues.
7. Commit any new changes you made and push them to GitHub.
8. Submit the following documentation about cross-browser testing in a zip file:
   * A screenshot with the passed W3C validator check (0 errors).
   * A list of the linters you used for your project. Include the names of the linters in a txt or doc file (if you didn’t use Atom, specify which editor you used). If you changed any plugin settings, specify those, as well.
   * A list of issues you found during your cross-browser testing (in a txt or doc file).
9. Submit a link to your published portfolio site.