UDACITY

PROJECT

Build a Portfolio Site

A part of the Front-End Web Developer Nanodegree Program

PROJECT REVIEW

CODE REVIEW 20

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

Project is looking great and you're almost there to meeting specifications. ⭐

There is just a little bit more work to do in terms of the image responsiveness, so keep up the good work on the way to your upcoming re-submission!

As you make your updates, remember to use the device simulations available through the Google Chrome Dev Tools to help you thoroughly test your page across different viewports.

> Please let me know what you think of DevFlow and repo.

Having had a look at your README and notes on DevFlow, I think it is great that you are using such tools to organise and enhance your development workflow. ⭐ At the end of the day, it is whatever works for you best that counts, and in my experience, employers are keen to see that you are able to use solutions as they are prevalent in the industry.

> Please look at the Q&A table and the Lessons Learned.

While I did look at this as per your request (the extensive discussion-oriented questions you appear to have addressed at reviewers), please note that for questions of this nature, or that may have otherwise come up during your development process, you are encouraged to make use of the more interactive support systems available as part of your Nanodegree, all of which are backed by a supportive community:

- Classroom Mentor
- Udacity Discussion Forums
- Nanodegree Slack Channel

Happy learning! 💫

Design

The page at minimum includes all of the following:

- **at least 4 images**
- **title text (h1, h2, etc.)**
- **regular (paragraph) text**
- **a logo.**

HTML5 semantic tags such as `<header>`, `<footer>`, `<article>`, `<section>` etc. are used to add meaning to the code.

No `<div>` or `<section>` tags are without a `CSS` class or id.

Check out the W3C documentation on HTML Structural Elements to learn more!

Well done using some semantic elements to mark up appropriate content in your HTML. I have left you some additional suggestions in the code review.

Having semantic code can go a long way to making your page structure more understandable to other developers, and also in terms of accessibility.

You can see on this page the list of new semantic elements in HTML5.

Provide at least one of the following customizations:

- **Customize images and text.**
- **Customize placement of the elements on the page (grid layout) with `HTML`, `CSS` or both.**
- **Customize `CSS` styles applied at minimum to paragraph and heading elements.**

Well done customising the portfolio to make it truly your own! 🌟

Page utilizes a grid-based layout with styles making use of the `flexbox` layout or a framework like `Bootstrap`, `Foundation`, etc.
If you're using `Bootstrap`: the rows and columns of the grid must be wrapped in an element with a `container` class.

Well done implementing Flexbox preferentially for layout across your page. For further learning, I can recommend the following lengthy, but very detailed article for thoroughly understanding the Flexbox layout system.

There exists also a Flexbox Defense game, dedicated to learning Flexbox.

Responsiveness

All content is responsive and displays on all display sizes. This includes:

- Desktop
- Mobile: Google Nexus 5
- Tablet: Apple iPad

An image's associated title and text renders next to the image in all viewport sizes.

TIP: Test responsiveness with Chrome Developer Tools device emulation by right-clicking anywhere on page, selecting 'Inspect Element', clicking the rectangle to the left of the Elements tab, select Apple iPad or Google Nexus 5 from Device drop-down list, and click reload.

No issues on this iteration of the project. It is worth checking this criterion again once you address the other issues with responsiveness present. 😃

All content is rendered on all three devices. No content should be hidden on mobile devices.

Viewport `meta` tag is included in `HTML`. (i.e. `<meta name="viewport" …`)

If a CSS framework is used, classes provided by the CSS framework are used to make images responsive, otherwise media-queries are used to ensure responsiveness of images.

If we inspect the page on Desktop viewports, we can notice that some of your project images appear squashed out of their native aspect ratio, meaning that they are not yet rendering fully responsively. Please see the code review for further guidance.

Separation of Concerns

Portfolio completely separates structure (`HTML`) from design/style (`CSS`). There are no `style` attributes present in the body of the `HTML` document. There are no `<style>` elements in the document.

Note: It is acceptable to include `height` and `width` attributes in `<img>` elements.

Well done making sure all your style information is handled through CSS in your stylesheet. ⭐

Files are organized with a directory structure that separates files based on functionality. For example:
`css/` for stylesheets
`img/` for images
`js/` for JavaScript files

Code Quality

- All code (`HTML` element names, attributes, attribute values) is lowercase (except `text/CDATA`).
- Code does not have trailing white spaces.
- Indentation is consistent (either all tabs or all 2 spaces or all 4 spaces etc).
- Code uses a new line for every block, list or table element and indent every such child element (it's acceptable to put all `<li>` elements in one line).
- [Optional] When quoting attribute values, code uses double quotation marks.

HTML code quality is excellent and a pleasure to parse! 

- `HTML` documents use `HTML5` `<!doctype html>` .
- Code passes `HTML` and `CSS` validators.
  *[Optional] Code does not use entity references unless necessary e.g. characters with special meaning in `HTML` (like < and &) as well as control or “invisible” characters (like no-break spaces).
- [Optional] Code omits type attributes for style sheets and scripts.

Well done on submitting valid HTML5 and CSS3 for review!

Providing the browser with valid code to work with is an excellent habit that offers a great starting point for your page rendering as expected and in line with the respective standards.

- Code does not have trailing white spaces.
- Indentation is consistent (either all tabs or all 2 spaces or all 4 spaces etc).
- Code indents all block content, that is rules within rules as well as declarations to reflect hierarchy and improve understanding.
- Code uses a semicolon after every declaration for consistency and extensibility reasons.
- Code always uses a space after a property name's colon, but no space between property and colon, for consistency reasons.
- Code always use a single space between the last selector and the opening brace that begins the declaration block.
- Code always start a new line for each selector and declaration.
- Code always put a blank line (two line breaks) between rules.
- [Optional] Code uses double quotation marks for attribute selectors or property values. Do not use quotation marks in `URI` values (`url()`).

- Code uses meaningful or generic ID and class names that are as short as possible but as long as necessary.
- Code does not use element names in conjunction with IDs or classes.
- Code uses shorthand properties where possible.
- [Optional] Code omits unit specification after 0 values.
- [Optional] Code includes leading 0s in decimal values for readability.
- [Optional] Code uses 3-character hexadecimal notation where possible.
- [Optional] Code separate words in ID and class names by a hyphen.
- [Optional] Code avoids user agent detection as well as `CSS` "hacks"—try a different approach first.

Well done naming your CSS classes semantically. Going forward, you may further be interested in looking into the BEM CSS framework, which establishes a robust methodology for naming styles and can help you stay on top of progressively more

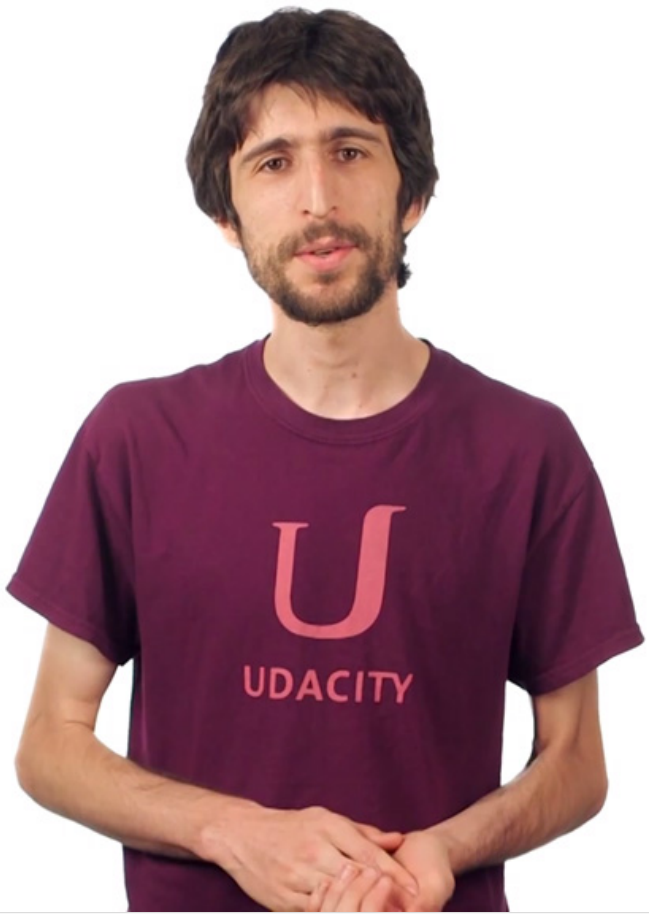
complex projects through improved modularity, re-usability and maintainability of CSS code.

- `HTML` templates and documents use `UTF-8` encoding. (no `BOM`) i.e. `<meta charset="utf-8">`.
  *[Optional] Mark todos and action items with `TODO`

RESUBMIT

DOWNLOAD PROJECT

20 CODE REVIEW COMMENTS ›



Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH

Rate this review

Student FAQ