Edward Myers

CS 601

Term Project Submission

Please fill out and have ready for Final Demo. Please turn in with your final submission.

1. How would you describe the website design? Does it use a color theme?

The project uses a traditional design style across all pages, with **seven top navigation buttons** laid out horizontally across the top of the page in **fixed position**. Body content flows behind it as I manipulated the navigation’s **z-index.** Content deploys and scrolls vertically down the page. As inherited from the suggestions from assignments #1 and #2, the pages are broken down by category:

* **Main** page has the introductory content, summarizing what is to follow and containing my audio element.
* **About Me** page has information about my high-school studies, family, and hobbies.
* **My Backyard** page has an interactive map centered on my hometown, with details about my job and job description.
* **Studies** breaks down my MS CIS curriculum at Boston University and my degree goals.
* **Projects** outlines my various papers and term projects from my other classes in the MS CIS masters program of note.
* **Gallery** provides an inventory of the major pictures used throughout the site and gives opportunity for credit citation.
* **Contact Me** contains links to view my LinkedIn page, GitHub page and to send me an email.

All page bodies have a header with a lead picture, with sub headers leading up section and article content underneath.

The site had a personal preference color palette hard coded into the primary styling (main.css), but through JavaScript I employed an **overlay palette css** provided by W3C for continuity. Headers, buttons, tables and div tag content used various forms of the light and dark styling, which changed both background and font color for clear viewing contrast.

* W3C light blue palette.css locally downloaded
* Credit: <https://www.w3schools.com/lib/w3-theme-light-blue.css>

1. How would you describe the website layout? Does it flow when resizing the window?

The website was originally envisioned to be viewed from the desktop monitor, widescreen format. However, several stepdown **breakpoints** exist where media layout changes at **900px, 600px and 400px** widths.

* **Greater than 900px**
  + Main menu navigation shows as horizontal buttons in an inline flex display.
  + Main html body width is capped at 900px, providing dead space equally on each side using auto margins.
  + H3 headers are left justified.
  + Tables are centered throughout.
  + Projects displayed on the Project page are oriented in a column, with interior description and link button displayed as grid with column width percentages to line up side by side.
  + Gallery page uses a 2-column fluid flex-box layout, delineating 2 column pictures per row and letting images flow organically for visual interest.
  + The Contact Me page has horizontal grid elements that line up link buttons with a JavaScript injected aside as fade text.
* **Less than 900px**
  + The main menu navigation switches to a three-column grid layout with the 7th button on the third row centered.
  + On the Gallery page, since the viewable window is narrower the display switches from flex box to flex column. The pictures now line up vertically with side padding.
* **Less than 600px**
  + Image sources are now switched from the 1280px horizontal versions to the 640px horizontal versions for more efficient page loading.
  + The main h1 title now begins to scale with the page, decreasing from default sizing to a minimum of 10pt using a calculation ( calc(10pt +5vw) ).
  + All headers are now centered.
  + Font sizing for common tags (p, table data, project description data, etc.) are made larger.
  + On the Project page, the link button now floats below the descriptive content for easier reading.
* **Less than 400px (mobile profile view typically fixes to 400px, therefore scaling below this breakpoint is not specifically styled)**
  + The header scales down to 10pt at a quicker calculation ( calc(10pt + 4vw)) to make room for a hamburger-style menu button (clover). The main navigation buttons are hidden by default, with the clover button acting as a toggle to hide and show the flex-column oriented buttons. This was done to show more of the page content and make the menu less dominant.
  + Font sizes are increased further for mobile format reading.
  + Content on the Projects and Contact page is oriented in full column mode, with buttons more pronounced for ease of touch.
  + Side margins around pictures that are full width are removed, allowing the image to float to each edge and remain bigger.
* **Print media**
  + Unnecessary buffers are removed, and a 2cm border is created around the print page.
  + Images are capped at 500px wide so as not to dominate printouts.
  + Anchor link href urls are appended to the text in print mode so the reader can understand the link destination.
  + Article content is set to 100% of the page width to shorten the printout.
  + Paragraph font is changed to Times New Roman as serif fonts read better on print media.

Upon increasing and decreasing the browsing window manually, the content contracts fluidly and doesn’t create element collision.

1. How would you describe the text? Is it readable?

The site uses a **specific set of fonts** for specific html elements:

* **A custom font ('satisfy')** was taken from 1001fonts with free license and run through a font format generator at fontsquirrel.com. The fonts are imported upon page load to style the section, table, and legend headers. Several san-serif fonts are supplied as a failback.
* **The Courier New/Courier/monospace** san-serif font family was used for button text styling and certain descriptive headers for contrast.
* *The Good Things* title used a standout **Georgia/Times New Roman/Times** serif font with sizing and weight.
* All other text uses browser default.

About 80% of the site uses serif fonts that are very readable within the browser. I specifically chose the satisfy script font for its visual appeal and understand the potential drawbacks of using it for header content. To combat this, I’ve made the text sizing larger and bolded where I use this font to make it as readable as possible.

1. How would you describe the navigation? Is it consistent across all pages?

The navigation is consistent in naming and button appearance across all pages regardless of the viewing size. Only CSS styling is altered to economize the readable space. It is always at the top of the screen in the vicinity of the main page title within the fixed display panel.

Regarding styling, the button referencing the current page is colored differently to provide the user with a visual indication of where they are within the navigation menu. When hovering over the buttons, there is a pronounced shadow over the button in focus to reinforce the mouse pointer position.

1. Have you included an image on every page? Do they have Alt, Height, Width attributes?

Where available, duplicate images with different resolutions are provided in the page source /images folder ( 1280 width and 640 width ). The image sources are switched between the two using css within the site depending on the current screen resolution to economize loading.

All images are encapsulated within figure tags, have default set width/height parameters, have alt text and employ a figcaption. Images are allowed to expand with site content fluidly based on width, with height being set by inheriting the picture ratio.

1. Does the site have any broken links? Does it work in every browser?

Every link on every page has been tested thoroughly on several browsers and all are working. There are some that link in main page and some that open another tab to view, all of which are working.

1. Does the site use 5 – 7 new HTML5 semantic elements? Does it use a good variety of HTML elements, bold, italics, highlighting, etc….?

Throughout the pages of the site are ample use of the following tags:

* HTML 5 - header, footer, nav, figure, figcaption, article, section, table (and sub-elements). JavaScript injects add blockquote and aside.
* Inline and special elements - <b>, <strong>, &copy;, and 🙌. CSS injects 👉 and 🎓 as unordered list bullets.
* Default HTML inline and block elements - <ol><li>, <ul><li>, <a>, <p>, <h1> through <h5>, <div> sparingly, <iframe>.

1. On what pages does the site use JavaScript? Do you think it shows mastery of the topic?

The site uses JavaScript in the following places:

* All pages:
  + Navigation menu: current page button is highlighted using JavaScript which changes the background color and font color via the function setActivePage().
  + Palette: The JavaScript function setPalette() creates the link to a new palette css link per page, then searches for certain document DOM elements through querySelect to assign new styling.
* My Backyard, Studies pages:
  + Table: table cell in focus becomes highlighted using JavaScript which changes the background cell color and font color via the function highLighter();
* Contact page:
  + Central contact buttons on click fire the contactText(option) function, which writes custom content wrapped in html to the DOM in the empty div element beside it.
  + Quote API: functions getApi() and writeApi() fetch a JSON file of quotes from the site credited below and writes a random one with string manipulation to the DOM in the empty article tag.
    - https://forum.freecodecamp.org/t/free-api-inspirational-quotes-json-with-code-examples/311373
* Projects page:
  + Projects list: functions getProjects() and jsonParse() fetch my hosted JSON file of projects and uses a loop to output the contents dynamically to the DOM within the empty section tag.

I believe my use of JavaScript throughout the site demonstrates good knowledge of core programming logic (loops, conditional statements, switch/case, document and window state access and DOM manipulation. Furthermore, I’ve used several ES6 constructs such as for-of, arrow functions, inline if statements, string methods, and asynchronous functions.

1. On what pages does the site use Vue.js? Given more time do you think you would use Vue on every page?

All pages:

* + Header / navigation menu:
    - The entire navigation menu button anchor links are reactively generated through a Vue component template using a v-for loop referencing a data object containing all button ids, classes, hrefs and text values.
  + Footer:
    - All page footers content and links are reactively generated through a Vue component template using direct {{ }} variable access to individual hrefs.

With both JavaScript and Vue.js, I find the ability to change content on multiple pages at once to be very powerful. Specifically with Vue, I find the list looping to have the potential to reduce repeated element creation on a page. If given more time, I would most likely incorporate an additional Vue component to iterate through my JSON projects list.

1. How does the site update the DOM? What pages show examples of updating the DOM?

Elements copied from JavaScript response above:

* All pages:
  + Navigation menu: current page button is highlighted using JavaScript which changes the background color and font color via the function setActivePage().
  + Table: table cell in focus becomes highlighted using JavaScript which changes the background cell color and font color via the function highLighter();
* Contact page:
  + Central contact buttons on click fire the contactText(option) function, which writes custom content wrapped in html to the DOM in the empty div element beside it.
  + Quote API: functions getApi() and writeApi() fetch a JSON file of quotes from the site credited below and writes a random one with string manipulation to the DOM in the empty article tag.
    - https://forum.freecodecamp.org/t/free-api-inspirational-quotes-json-with-code-examples/311373
* Projects page:
  + Projects list: functions getProjects() and jsonParse() fetch my hosted JSON file of projects and uses a loop to output the contents dynamically to the DOM within the empty section tag.

1. Does the site include extra credit, including: PHP, Database, Flexbox, Grid, CSS Media Queries or AJAX. Please explain.

Above and beyond elements:

* + Two JSON fetches (projects list and remote API import).
  + Use of noscript tags on every page to provide functionality if JavaScript is disabled.
  + Use of custom imported font script.
  + Seven pages of content rather than the minimum of three pages.
  + Use of flexbox and grid displays in several elements, especially in navigation as described above.
  + Use of 3 benchmark media tags to stipulate changes in styling behavior.
  + Print media styling.
  + Use of keyframe animations to add slide-in and fade effects to various headers and body content.
  + Remote API used for quote feature.
  + Palette css used to unify theme.
  + Favicon.ico and .png added for multi-browser tab viewing. Clover design recycled to be the ‘hamburger’ menu mobile button.