svinfo6150 / **student-keepgoing2996** Private

Code Issues Pull requests 1 Actions Projects Wiki Security

main student-keepgoing2996 / **project1** /

swiftone Adds Oct 11 Notes + Project1 ... 6 days ago **History**

..	
public	6 days ago
README.md	6 days ago
package.json	6 days ago
server.js	6 days ago

Project 1: Multi-page Website

** Due Mon Oct 17, 11:59pm PT **

Goal

README.md

You will create a website made of several HTML pages and related CSS to demonstrate the skills learned in class so far.

The website will be about an animal of your choice (real or not, but make sure you will be able to get any images needed following the restrictions in this README!)

This webpage can be viewed and used by running:

- `npm install` (only required once on any given computer per project/assignment)
- `node server.js` (starts the server, Ctrl-C to stop running the server)
- Visiting `http://localhost:3000/` in the browser

Functional Requirements

The website will consist of these pages (created inside the `public/` directory):

- `index.html` (a "Homepage")
- `about.html` (an "About" page)
- `register.html` (a Registration page to register a detail about an animal)

Each page will consist of a `<header>`, `<main>` and `<footer>` that are semantically correct. You may include other elements in/around these (for example, a `<div>` might contain one or all of the above elements) as long as these elements are used in a semantically correct way (those elements must contain content that matches the meaning of the element type - see the element's definitions on MDN if you are uncertain as to what a given element should contain).

Each page will have a `<title>` element that names the page.

- Each page will have an `<h1>` element with text that matches the text in the `<title>`
  - Exception: Each page might have an `<h1>` element that matches the name of the site. If so, the `<h1>` will not change per page, and instead there will be an `<h2>` element that will match the `<title>` and change per page.

The page may contain links to pages that do not exist

- Example: you can have a link to a privacy policy page that does not exist

Every link should use a filename that could exist

- Example: you can link to a non-existent "privacy.html", but you should not have multiple links to "/fake" or "fake.html", because "fake" does not say what the page would be

Header/Footer Requirements

- The visual header/footer for each page is identical
  - Hint: There is no way to automatically do this with just HTML, so you will have repeated content in each HTML file
  - Exception: If the page identifying header (`<h1>` or `<h2>`) is visually in the header, it will be unique on each page
- The header/footer is visually distinct from the main content of the page
- The header/footer have the same color background from each other (distinct from the main content of the page)

- The header will include a "logo" image (of your choice, subject to the image restrictions given in this file)
  - The logo image will be a link to the home page
  - The logo image will have an alt attribute of ""

Navigation requirements

- The page will include a `<nav>` that includes links to each of these pages:
  - "Home"
  - "About"
  - "Register"
- The navigation is wherever in your HTML structure you want, but should appear on every page
  - Example: The navigation may be inside the `<header>` or it may not be, either is acceptable
- You may choose different text for the links
  - They must link to those pages
  - The text you use must make it clear that they link to those pages
    - Example: "About Us" or "About Dragons" are fine, but "Ponder the Inscrutable" is not clear that it takes you to the about page
- You may link to the pages with absolute or relative paths, your choice
  - You must NOT use fully qualified URLs in these links (Example no mention of "localhost")

The navigation must be shown in at least one of the following ways:

- A horizonal menu with dropdowns
- A vertical menu that does slide-in/slide-out

Content Requirements

Homepage Contents

The home page will include at least 3 UI Cards.

Each card will contain at least:

- a visible and semantic heading title
  - Hint: Remember not to skip numbers in h1-h6 elements!
- A call-to-action link

Each card will have at least one of:

- A background color that makes it visually unique from the other cards
- A image that makes the card visually unique from the other cards

The links in the cards to not need to link to actual pages

The Home Page should make it obvious what Animal the site is about

About Contents

The About page will include at least 3 paragraphs of text that each contain at least 3 sentences.
At least 1 sentence must be real text saying in more detail what animal the site is about and something about that animal. The remaining text may be lorem-ipsum like text.

Registration Contents

The Register page will include a form

The form will submit via POST to `/register`

The form will ask for (at minimum):

- name
- email
- one piece of info using a checkbox
- one piece of info using a dropdown

You should use 1 or 2 column layout (your choice)

- The form must be usable at the required viewport sizes, but there is no specific requirement for adaptive behavior of the form

At least 1 field is required and should be visibly marked as required

The page/form should make it clear what information it is requesting and why

Responsive/Adaptive Requirements

The page is primarily laid out according to a 12-column grid

- Exceptions are allowed (example: a horizontal menu bar), but the page should mostly align to the 12 column grid with no exceptions that take up significant space

The page must be pleasant to read from 360px+ (at standard font-size), with no cut-off content, overlapping text, or awkwardly wrapped text.

- You may contain the page contents with expanding gutters, but the page should be responsive until 1000px
- There must be at least one adaptive breakpoint (exact size is your choice) on at least one page
- Adaptive breakpoints should use rem as their unit
- You must have text/boxes that are easily seen to be responsive (wrapping text as the browser resizes)

Accessibility Requirements

Reasonably resizing text/zooming does not cause your display to become difficult to use

All forms, menus, buttons, and links must be usable with keyboard as well as mouse

No information should be conveyed through color differences only

All form fields should be associated with a `<label>` element that contains useful text

Any form fields that are required should be visibly and textually marked as required

All images will have descriptive `alt` attributes

- These alt attributes should describe the contents of the picture

Any icons are not required to understand and use the page with a screen reader

- This means there must be text to explain what happens, icons can only add to the experience, not be the only means to understand it

Any transitions/animations must be set in a media query so that they don't occur for users that have indicated they want reduced-motion

Demonstrated Skills Requirements

These are skills that I/TA(s) must be able to confirm you have an understanding of based only on seeing you use the skills in the HTML/CSS

- Semantic HTML
- Semantic Class names
- CSS Grid

- CSS Flexbox
- 12 column grid layout
- A non-static position property
- Responsive HTML
- Adaptive HTML
- HTML Forms

Additional Requirements

- Follow the best practices from the course
- Do not use any Javascript (JS) (soon!)
- Do not use float unless wrapping text around an image
- Do not use tables or table-layouts unless showing a table of data
- Do not use iframes or otherwise load external content except as explicitly allowed/directed by this README
- I/TA(s) must be able to examine and evaluate your understanding of the course lessons
  - Meaning: We have to be able to look at your HTML/CSS and be able to tell if you have learned the lessons based one what we read
- Do not use any CSS preprocessor (SASS/Less/etc)
- Do not use external CSS
  - Exception: You may use icons from css.gg by copying the CSS snippet(s) into your own css file
    - You MUST include a comment in your CSS saying "below CSS adapted from https://css.gg under the MIT License"
    - You MAY modify/adapt/rename the CSS from css.gg. The comment will still apply
- Any images must be ones you personally own or from `http://unsplash.com`
  - Include an `images.txt` file in your repo (at the same level as package.json, not in `public/`) and list the unsplash.com url of each image you downloaded from unsplash
  - For any images that you personally own and used, list (in `images.txt`) the filename of that image as you have it in this project. We can recognize which images are yours and which were downloaded from unsplash by looking at the images and this `images.txt`.
  - Do NOT download and use random images from the internet/web. Only your own images or from unsplash.

Submission Instructions

- start from the up-to-date main branch (`git checkout main; git pull origin main`)
- Create a feature branch named 'project1' (`git checkout -b project1`)
- Add the necessary files to have the required features
  - running `npm install` will create a `package-lock.json` file and a `node_modules` directory. These should NOT be added to your commit.
- add, commit, and push the branch to github
- Create a PR to merge to main
- Be sure to include the TA(s) and myself reviewer(s)