15-112 Term Project Proposal

Harvey Zheng | harveyz

Project Description

Name: WebSketch

The project will allow you to lay out different components to mockup or design a website, and export it to a static page with actual HTML and CSS.

Competitive Analysis

Currently, there are a number of editors that allow you to lay out components for a website, and it will give you the HTML, CSS, and any JavaScript the site may need for things like carousels. These include companies like Mobirise, Bootstrap Studio, Adobe Muse, or SquareSpace.

Mobirise uses pre-existing blocks that the user can edit and maneuver in a static page. Similarly, my project will generate a static page from dragging in already existing blocks. The difference is that Mobirise uses a grid-like system, where things are placed above and below each other, never on top of each other.

Bootstrap Studio also uses pre-existing components built on top of the Bootstrap CSS framework. Since the app uses Bootstrap, they natively use Bootstrap's grid system for the arragement of components. Once again, you cannot overlay things on top of each other, like with layers.

Adobe Muse was software within the Adobe Creative Suite that generated static pages from laying out components onto a canvas, and is not built on any existing web technologies. This is probably most similar to what I aim to build, although my project will be significantly simpler and won't have as many features. Objects can free-float, and lots of custom stuff can be added.

SquareSpace is a popular site builder that uses prebuilt components in a certain order, and its components have limited edits it can make. However, its simplicity is something I'd want to replicate, since it is very easy to design and publish a website with.

Structural Plan

The project has these main parts:

1. Animating GUI for a user to lay out their website with
2. A way to store the objects onto a canvas and have it be retrievable later
3. A way to export the objects into an HTML and CSS files.

And will probably be organized into three different files with these parts.

The GUI

The GUI will include a few different tools that will create or edit objects on the canvas. This will include:

- A color palette, for the current selected color
  - User clicks on a box, inputs a hex code, and the app will store it.
- A create rectangle (div) tool, which will need to create rectangles of different sizes based on where the user drags it. You will also have the ability to enter text into the rectangle.
  - Color is based on the color in the palette. Will have a view aspect (dragging rectangle), a controller aspect (manipulating the rectangle's model from the view), and a model (where its data is stored).
- A text tool that inputs text where you want it
  - Font, color are user inputs. Will also be stored in a class.
- An align tool to align up to two selected objects
  - Probably end up just being a method that is activated when clicked.
- A cursor tool to select objects to be moved or deleted.
  - Will need to store the objects that are shift-clicked into a list.

Storage

Storing objects should be done in a way that makes it easy to export to HTML and CSS. Each type of component will get a different class: a div will get a class and text will get a class. Classes for each CSS class will also be created. Class variables will keep track of every object created.

Exporting to HTML and CSS

This will likely be a function or two that converts the stored objects into HTML and CSS based on the attributes of the objects.

Algorithmic Plan

The hardest part of the project will probably be how I convert objects laid out on a canvas into HTML and CSS from a Python animations library. The best way to do this would be to properly store objects and make sure they have the right attributes. For the purposes of the MVP, everything on the exported web page will be absolutely positioned.

A list within the model portion of the app will store each component, with the indices corresponding to their layer, the higher the index the more forward the object.

A div's attributes will include any text as a text object, its position, and its color. Text objects will store its position, the content of the text, the kind of alignment, and instantiate a CSS class object that stores its font, font size, and color. If the CSS class object already exists, per a set of CSS class objects stored as a class variable, then it will use the already existing object.

When exporting to HTML and CSS, the Python function will loop through the list and convert the objects into HTML and CSS. For the divs, style attributes will be used for their colors, heights, positions. For the text, the set of CSS objects will be converted into a CSS file, with the class of each text object assigned accordingly. HTML style attributes will be used on both to ensure `position: absolute` is enabled, and the index numbers can be converted into `z-index` numbers. The exporter function will then write it into a file.

Timeline Plan

Date	Event	Notes
11/20	Ability to create rectangles and input text	TP1
11/22	Add text to rectangles and store input text attributes as CSS classes	
11/24	Align objects and color picker	
11/26	Export objects to HTML and CSS	TP2

Version Control Plan

My plan to back up my code is to regularly push to a public GitHub repository.

Module List

- HTML

- CSS