WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Simon Feist

Email: feistsp@ucmail.uc.edu

Figure 1: Simon's headshot

Project 1

Overview

I created and deployed a personal website on github.io with information including my resume, name, headshot, and contact information. This includes a link to another HTML page with information about the course, Web Application Programming and Hacking (WAPH). The personal website includes the use of Bootstrap, React, and a page tracker. It also includes basic JavaScript code, JQuery, Web APIs, and cookies.

https://github.com/spf125/spf125.github.io

Creating and Deploying Webpage

Using github.io, I created an index.html file that similar to the HTML file created in Lab 2. I then created a waph.html file containing information about this course and its assignments. I committed these to the github repo, so they are now accessible as a webpage at https://spf125.github.io/.

CSS template (Bootstrap)

To implement Bootstrap, I found a template from BootstrapMade. I then incorporated this into my current index.html file.

Page Tracker

To add a page tracker to my website, I decided to use https://analytics.with google.com. This involved creating an account and creating a property for the website. I then had to add the google tag inside my all of my HTML files in the website which is in the form of a JS script.

Basic JavaScript code (JQuery and React)

Digital Clock This feature is implemented by using JavaScript code within the `index.html` file. The function `displayTime()` is called every 500ms which sets the current time using the `Data()` function. This is placed on the webpage using the `getElementById()` function.

Analog Clock The analog clock is implemented by using an external JavaScript file from https://waph-uc.github.io/clock.js. By including this script and using a canvas tag in the `index.html` file, I was able to create the analog clock by calling the different JS functions within the external `clock.js` file.

Show/Hide Email To implement this feature, I created a JavaScript file titled `email.js` that has a function called `showhideEmail()`. This function either shows or hides email when the text is clicked. This is then placed in a `<div>` with the onclick attribute calling the `showhideEmail()` function.

Using React to Implement Counter For the final feature, I implemented a counter that increments with a button press and decrements with a different button press using React. This was accomplished by adding the lenks for the React development. I then used React with basic JavaScript code to create a class Counter that extends the React.Component. Then, using the React `render()` function, I was able to create the buttons the increment and decrement the counter using event handlers.

Web API Integration

jokeAPI To implement the joke of the day API, I used JQuery. JQuery gets the JSON from the webpage https://v2.jokeapi.dev/joke/Any. I then check if it's a two part joke or a single, and then put this response at the joke ID. I use setInterval to call the function to get the joke of the day every 1 minute.

API with Graphic Similar to the joke of the day, this API is also implemented with JQuery. However, since my access to https://xkcd.com/info.0.json is blocked

CORs, I had to use an external tool, `https://api.allorigins.win/get?url=`, to pull the contents from the API.

JavaScript Cookies

To implement JavaScript cookies, I had to create three functions: `SetCookie()`, `GetCookie()`, and `CheckCookie()`. The `SetCookie()` function creates a cookie with a given name, value, and days until it expires. The `GetCookie()` function either returns blank or the value of the cookie given the cookie's name. Then, `CheckCookie()` has the logic to check if a user has visited the website or not, displaying the correct message to the HTML. For displaying the cookie, I chose to make it a banner that extends from the top of the screen using HTML and CSS with a button to close the banner.