WAPH-Web Application Programming and Hacking

Instructor: Dr. Phu Phung

Student

Name: Ana Cedillo

Email: cedillak@ucmail.uc.edu

Short-bio: My key interests are in web and app development.

Ana Cedillo

Repository Information

Respository's URL: https://github.com/cedillak/waph-cedillak.git

This is a private repository for Ana Cedillo to store all code from the course.

Overview

This project involved creating and deploying a professional profile website using GitHub Pages. The website showcases my resume and integrates various functionalities using HTML, CSS, and JavaScript, along with public APIs. The website is accessible at https://cedillak.github.io, and the project folder is available on GitHub at https://github.com/cedillak/cedillak.github.io.

Through this project, I gained valuable experience in creating and deploying a professional website using GitHub Pages. I learned how to use a Bootstrap template to quickly design a polished look, and I enhanced my JavaScript skills by adding interactive features like digital and analog clocks, a show/hide email button, and a confetti animation. I also integrated public APIs to include dynamic content, such as a random programming joke and a dog image. Additionally, I implemented JavaScript cookies to personalize the user experience by remembering visitors and displaying welcome messages. Overall, this project improved my web development skills and taught me how to make a website both functional and engaging.

General requirements:

Creating and Deploying a Personal Website:

I created a repository on GitHub named `cedillak.github.io` to host my personal website. Using a free Bootstrap portfolio template, I replaced the placeholder content with my personal information, including my name, headshot, contact details, education background, work experience, and skills. This customized template ensured a professional and cohesive presentation tailored to potential employers. The template allowed for a clean and professional look that I believe appeals to potential employers in the computer science field.

Adding a Link to the WAPH Course:

I added a new HTML page, waph.html, which provides an overview of the "Web Application Programming and Hacking" course. I linked this page from the main website using the following code: `<a href="waph.html" target="_blank">Click Here to Access the Course Information</a>`

Non-technical requirements

Use an open-source CSS template:

To target potential employers, I selected a Bootstrap theme that matched my professional aspirations as a computer science student. I Googled "free bootstrap themes" and chose one that provided a modern and professional look. This theme was customized with my personal details to create a cohesive and attractive profile.

Include a page tracker:

I implemented a page tracker using Flag Counter to keep track of visitors to my website. The tracker was added to the index file with the following code: `<h3 class="mb-0">Page Tracker</h3> <a href="https://info.flagcounter.com/0sYk"><img src="https://s01.flagcounter.com/count2/0sYk/bg_FFFFFF/txt_000000/border_CCCCCC/columns_2/ma alt="Flag Counter" border="0"></a>` This snippet includes a header and an image link from Flag Counter, which displays visitor statistics on the website.

Technical requirements

JavaScript code:

1. **Digital Clock:** I implemented a digital clock using JavaScript that displays the current time. The digital clock function updates the inner HTML of an element with the ID `digit-clock` to show the current date and time. This function is called every 500 milliseconds using `setInterval`, ensuring the displayed time remains current.
2. **Analog Clock:** For the analog clock, I used JavaScript to create a clock face with moving hands. The analog clock is drawn on a canvas element. The code first sets up the canvas context and adjusts the center of the clock. It then periodically calls the `drawClock` function every second to update the clock's face, numbers, and hands.
3. **Show/Hide Email:** This feature toggles the visibility of an email address. Initially, the email is hidden. When the user clicks a button, the `showhideEmail` function either shows or hides the email address by modifying the inner HTML of the `email` element and updating its content accordingly.
4. **Confetti Animation:** To add a playful element, I incorporated the JS Confetti library. The confetti effect is triggered by clicking a button. The button has an event listener that calls the `addConfetti` method of the JSConfetti library, which generates a confetti animation using specified emojis.

Integrate the jokeAP:

I integrated the JokeAPI to fetch a new programming joke every minute. The code uses jQuery to fetch a joke from the JokeAPI every 60 seconds. The `fetchJoke` function makes a GET request to the API and updates the HTML of an element with the ID `joke-of-the-day` to display the joke. This ensures that the joke updates automatically at regular intervals.

Integrate public API with graphics:

1. **Random Dog Image API:** I used the Dog CEO's Dog API to fetch and display a random dog image. The fetch function makes a request to the API, retrieves the image URL, and then calls `display_image` to set the

`src` attribute of an `img` element to this URL, displaying the image on the page.

2. **Pokémon Image API:** As a bonus, I implemented functionality to fetch and display a Pokémon image based on user input. I used the PokeAPI to get data for a Pokémon specified by the user. The fetch request is made when the user submits the Pokémon name, and the image URL from the response is used to update the `src` attribute of an HTML `img` element.

Use JavaScript cookies to remember the client:

1. **Set Cookie:** I created a function to set a cookie with a specified name, value, and expiration date. This function constructs a cookie string and assigns it to document.cookie.
2. **Get Cookie:** Another function retrieves the value of a cookie by its name. It parses document.cookie, splits it into individual cookies, and returns the value of the specified cookie.
3. **Check Visit:** The main function checks for a cookie indicating the last visit. If the cookie is not found, it displays a welcome message and sets the cookie with the current date and time. If the cookie is found, it displays a message with the last visit date and time, then updates the cookie with the current date and time. This function runs when the page loads, ensuring the welcome message is displayed appropriately based on the user's visit history.