**INTRODUCTION TO WEB DEVELOPMENT**

The "Interactive Web Elements" project is designed to help kids understand the foundational components of web development using HTML, CSS and JavaScript through creative and relatable analogies. Instead of building an actual webpage, this project focuses on explaining key concepts in an engaging way, making web development accessible and fun for kids. At the end of the session, they will be able to recreate a similar project, an interactive website telling a little bit about themselves.

* **DAY 1 : Introduction to Web and HTML**
* **DAY 2 : Introduction to CSS and JS**
* **DAY 3 : Project day - Kids create their own webpage**

**a.) Introduction to the Web**

* Explain that the internet is like a giant network of information and services.
* Describe what a webpage is - a document that can be viewed in a web browser.
* Introduce the roles of HTML (structure), CSS (presentation) and JavaScript (interactivity) in building webpages. - Show them a web page they are familiar with like YouTube or the Webpage we’ll build to explain these concepts.

**HTML (Structure):**

* Start by explaining that HTML is like the backbone of a webpage, just as the framework of a building provides structure and support.
* Using YouTube as an example, point out how HTML is responsible for organizing the page's content. You can mention that it defines the layout and structure of the YouTube page.
* Identify key HTML elements on the YouTube page, such as headings (titles and video descriptions), lists (video recommendations) and links (video titles and navigation).

**CSS (Presentation):**

* Explain that CSS is like the interior design of a webpage, making it visually appealing and attractive.
* Using YouTube, demonstrate how CSS is responsible for styling elements like fonts, colors, backgrounds, and layout.
* Point out the design aspects of YouTube, such as the color scheme, fonts used for text, and the layout of videos and recommendations.
* Mention that CSS styles are applied to HTML elements, making them look consistent and appealing to the user.

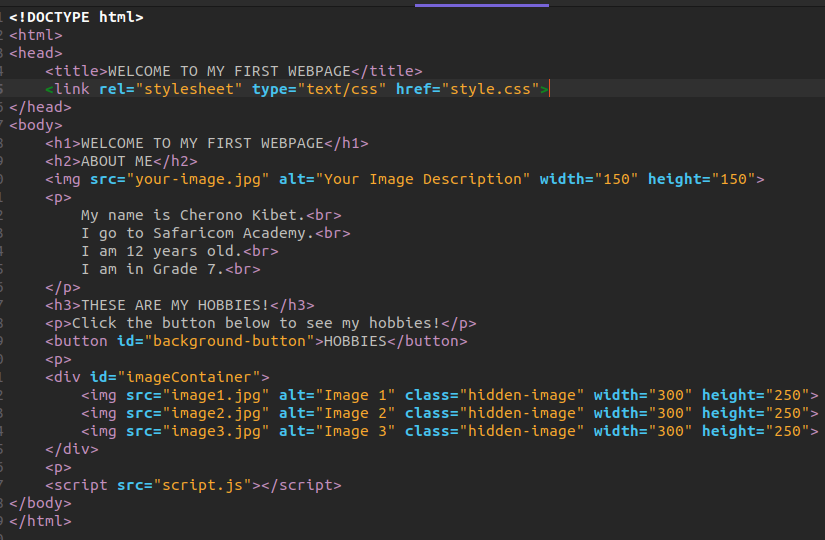
**JavaScript (Interactivity):**

* Describe JavaScript as the interactive and dynamic part of a webpage, providing functionalities and actions to make the page responsive.
* Use YouTube as an example to showcase JavaScript's role in features like video playback, comments, likes and user interactions.
* Highlight how JavaScript allows for actions like clicking on a video to play it, leaving comments, and liking videos.
* Explain that JavaScript can also control elements such as dropdown menus and interactive buttons on YouTube.

**b.) HTML Basics**

This breakdown below will help the kids understand the different parts of HTML and how they come together to create a webpage. It's like explaining the pieces of a puzzle that make up a beautiful picture!

First, create an index.html file in your project directory and add the code below piece by piece, explaining each step as listed below. Run the html script after adding each line of code so that they can understand how the webpage comes to life step by step.

****

**DOCTYPE Declaration (<!DOCTYPE html>):**

This line tells the web browser that we're using HTML, just like saying, "Hey, we're going to speak in the language of webpages!"

**HTML Tags (<html></html>):**

These tags are like a giant envelope that wraps around everything on our webpage, keeping it together.

**Head Section (<head></head>):**

Think of this part as the recipe card for your webpage. It's where you write down all the important instructions and notes for making the webpage taste and look just right. The audience doesn't see part of this card; it's like the secret ingredient list kept behind the scenes. You might write down things like what spices to use (like CSS), how long to bake (like JavaScript) and a title that tells everyone what the dish is (like the webpage's title at the top of the browser).

**Title Tag (<title>WELCOME TO MY FIRST WEBPAGE</title>):**

This is like the title of a book. It tells the browser what to show at the top of the webpage's tab.

**Link Tag (<link rel="stylesheet" type="text/css" href="style.css">):**

This tag tells the browser to go look at a file called "style.css" to find out how to make our webpage look nice.

**Body Section (<body></body>):**

This is like the pages inside the storybook. This is where the actual story unfolds. It's where you find all the characters, dialogues, and the exciting plot. Just like when you open a book and dive into the story on its pages, the **<body>** of the webpage is where we put all the words, pictures, and interactive elements that make up the exciting story of our webpage. It's where the adventure begins and you can see and interact with everything, just like turning the pages of a thrilling storybook.

**Heading Tag (<h1>WELCOME TO MY FIRST WEBPAGE</h1>):**

This is the big title of our webpage, like the title of a storybook.

**Heading Tag (<h2>ABOUT ME</h2>):**

This is like a smaller title for a special section about me.

**Image Tag (<img src="your-image.jpg" alt="Your Image Description" width="150" height="150">):**

This is like a picture. **Image Source (src="your-image.jpg"):** This is like telling the book where to find the picture. It's like saying, "Look on page X to find the picture." **Alt Attribute (alt="Your Image Description"):** This is like a caption or a description under the picture in a book. It helps readers understand what's in the picture, especially if they can't see it. It's like having a small note that tells you about the picture in case you can't see it clearly. For example, it might say, "A happy dog playing in the park." **Width and Height Attributes (width="150" height="150"):** These are like instructions for how big the picture should be on the page. Imagine you're putting a photo in your book and you want to make sure it fits nicely on the page. These attributes tell the book how wide and tall the picture should be. It's like choosing the right size for your picture so that it looks just perfect in your storybook.

**Paragraph Tag (<p>...</p>):**

This is where we write sentences and stories. It tells people more about me.

**Line Break (<br>):**

This is like pressing "Enter" on the keyboard. It moves the text to a new line.

**Heading Tag (<h3>THESE ARE MY HOBBIES!</h3>):**

Another smaller title, this time about my hobbies.

**Paragraph Tag (<p>Click the button below to see my hobbies!</p>):**

This is where we tell people to click a button to see something fun.

**Button Tag (<button id="background-button">HOBBIES</button>):**

This is like a magic button. When you click it, something special happens.

**Image Container (<div id="imageContainer">...</div>):**

This is like a box that holds more pictures.

**Hidden Images (`<img src="image1.jpg" alt="Image 1" class="hidden-image" width="300" height="250">):**

These are more pictures, but they're hidden for now. When you click the button, they'll appear.

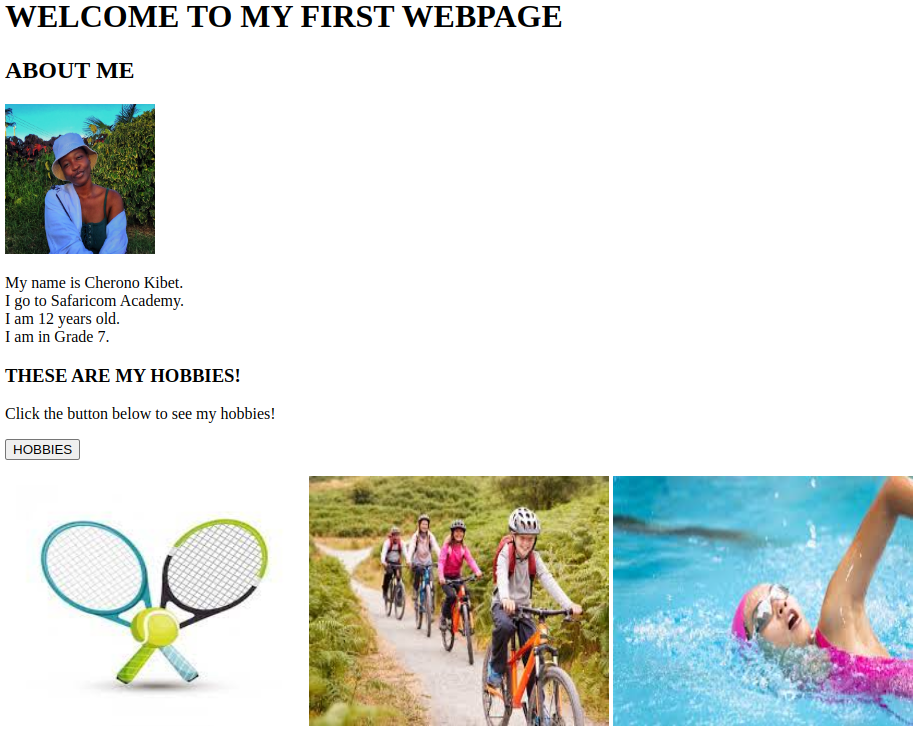
**Script Tag (<script src="script.js"></script>):**

This is where we include JavaScript, a language that makes our webpage do amazing things. The script.js file is like a recipe book for our webpage, telling it what to do when the button is clicked.

**NOTE:**

**Remove the lines calling the CSS and JS script and run the index.html as is. Let them see how the website looks like without CSS(Presentation) and JS(Interactivity)**

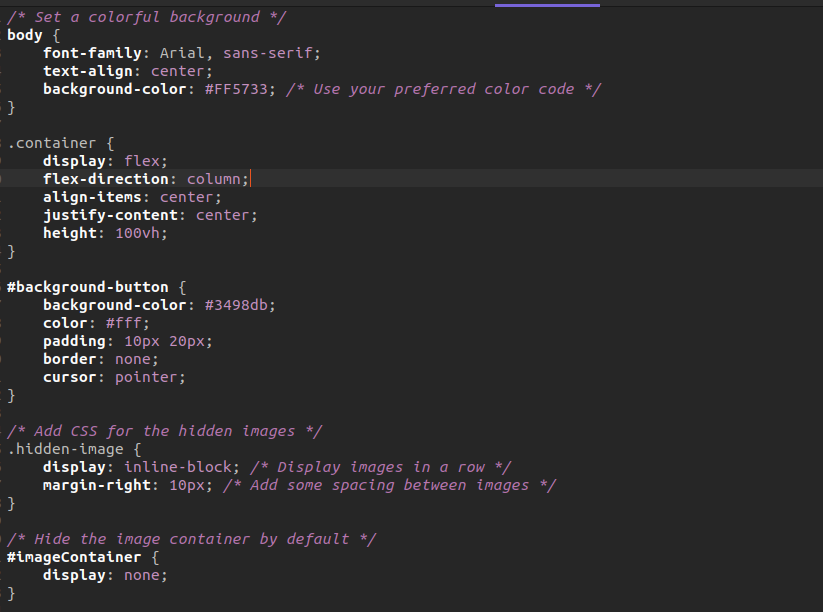
**See Below:**

****

**c.) CSS Basics**

This breakdown below will help the kids understand the different parts of CSS and how they come together to create a webpage. It's like explaining the interior design of a webpage, making it visually appealing and attractive by adding colours and giving it a nice shape.

First, create a style.css file in your project directory and add the code below piece by piece, explaining each step as listed below.

****

**body {**

**font-family: Arial, sans-serif;**

**text-align: center;**

**background-color: #FF5733; /\* Use your preferred color code \*/**

**} :**

**Choosing the Book Cover (Styling the Book Cover):** Just like we design a book cover to make it attractive, the body here represents the "book cover" of our webpage. We're selecting a special font (Arial) for the text on our cover, centering the text neatly, and giving the cover a background color (#FF5733) that's like painting the cover with our favorite color.

**Picking the Font (Choosing the Text Style):** It's similar to selecting a nice font for the text on the book cover to make it visually appealing.

**Centering the Title (Text Alignment):** This is like ensuring that the title on the book cover is right in the center, making it look balanced.

**Painting the Cover (Background Color):** The background-color is akin to painting the book cover with a vibrant and captivating color. You can choose your favorite color code just as you might choose a cover design for a book.

**NOTE:**

**Run the webpage with only the body in the css so that they can note the changes this makes on the webpage.**

**.container {**

**display: flex;**

**flex-direction: column;**

**align-items: center;**

**justify-content: center;**

**height: 100vh;**

**} :**

**Creating Chapters (Styling a Chapter):** Think of .container as a chapter in our book. We're giving this chapter a specific style.

**Flexible Book Layout (Flex Display):** **With display: flex**, it's like using a flexible book layout that can be adjusted to make the chapter look nice. We can arrange the content vertically **(flex-direction: column)**, align it neatly and ensure it's visually appealing **(justify-content: center).**

**Determining Chapter Length (Chapter Height):** The height: 100vh is like specifying how long this chapter should be. It's as tall as the entire book, making it occupy the whole page.

**NOTE:**

**Run the webpage with only the body and the .container in the css so that they can note the changes this makes on the webpage.**

**#background-button {**

**background-color: #3498db;**

**color: #fff;**

**padding: 10px 20px;**

**border: none;**

**cursor: pointer;**

**} :**

**Decorating a Special Page (Styling a Page):** This part is like decorating a special page in our book. We're making it look inviting and engaging.

**Selecting Page Colors (Background and Text):** We're choosing the background color (#3498db) for the page, just like deciding on the paper color for a page in a book. We're also picking the text color (white) to ensure it's easy to read.

**Adding Page Comfort (Padding):** The padding is like adding a comfortable margin around the page to make it user-friendly and visually appealing.

**Page Border (Border):** We're saying the page doesn't need any borders, making it look clean and sleek.

**Inviting Interaction (Cursor):** The cursor: pointer is like telling readers that when they hover over something on the page, it responds, just like a clickable link in a book.

**NOTE:**

**Run the page again with these changes for the kids to note the changes that happen. Also play around with different colors for the background and text.**

**.hidden-image {**

**display: inline-block; /\* Display images in a row \*/**

**margin-right: 10px; /\* Add some spacing between images \*/ :**

**Revealing Hidden Illustrations (Styling Hidden Images):** This is like preparing to reveal some hidden illustrations in our book, which are initially kept behind a curtain.

**Arranging Illustrations (Display and Spacing):** We're making the hidden images display in a row **(inline-block)**, just like pictures within a storybook. We're also adding some space **(margin-right)** between the pictures to make them visually appealing.

**NOTE:**

**Again, run the page to show them how the photos appear.**

**#imageContainer {**

**display: none;**

**} :**

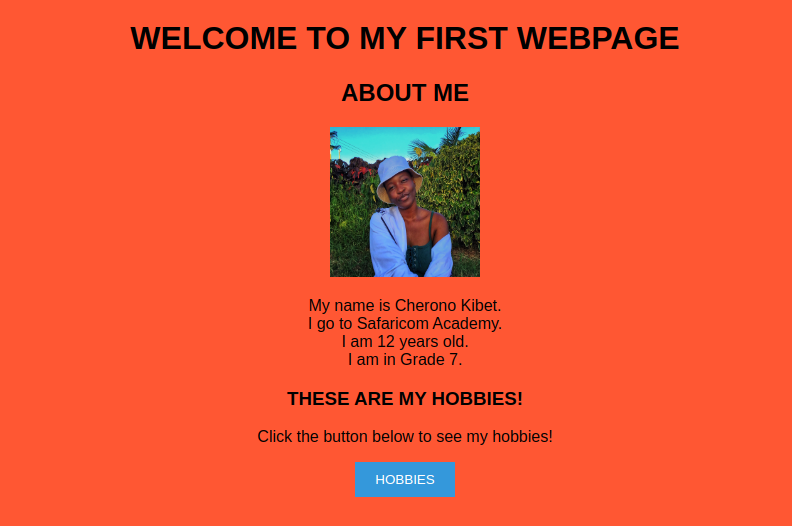
**Hiding a Secret Compartment (Hiding the Image Container):** Imagine this as a secret compartment within a bookshelf. The #imageContainer is like the name of the hidden compartment, and we're saying it should stay hidden (display: none) until we decide to reveal it, just like a hidden treasure in a storybook.

**NOTE:**

**Again, run the page to show them how the photos hide after adding this piece of code.**

**Also, try tapping the button to show them that the page is not interactive because of lack of JS(Interactivity)**

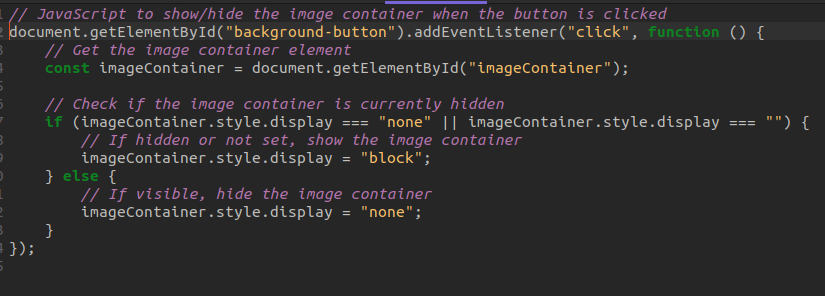
**See below the curent website:**

****

**c.) JS Basics**

This breakdown below will help the kids understand the different parts of JS and how they come together to create a webpage. JS acts as the set of instructions that responds to your actions, like operating a light switch, making things appear or disappear in your webpage.

First, create a script.js file in your project directory and add the code below, explaining each step as listed below.



**document.getElementById("background-button").addEventListener("click", function () :**

**The Light Switch (document.getElementById("background-button")):** Think of document.getElementById("background-button") as the light switch in your room. This part tells the code to find the specific light switch with the **id "background-button."**

Turning On the Light **(addEventListener("click", function () {):** Just like you press the light switch to turn on the light, **addEventListener("click", function () {** tells the code to "listen" for a click on the light switch. When someone clicks on the light switch (just like clicking the button), the action inside the curly braces ({}) is what happens. It's like turning on the light when you flick the switch.

So, **document.getElementById("background-button").addEventListener("click", function ()** is like saying, "Find the light switch with the id 'background-button,' and when someone clicks on it, do the following action," just as you would turn on the light when you press the switch.

**const imageContainer = document.getElementById("imageContainer"); :**

The Hidden Room **(The Image Container)**: Think of the image container as a hidden room in your house. It's usually closed, and you can't see what's inside.

**if (imageContainer.style.display === "none" || imageContainer.style.display === "") { imageContainer.style.display = "block";**

**} else {**

**imageContainer.style.display = "none";**

**} :**

Checking the Light **(Conditions):** The if and else parts are like checking whether the room is currently dark (lights off) or not.

Is the Light Off? **(Display Property)**: The imageContainer.style.display is like checking if the lights in the room are turned off (display: none).

Turning On the Light **(Showing):** When you press the switch (click the button), and the room is dark, you turn on the lights by setting display to "block."

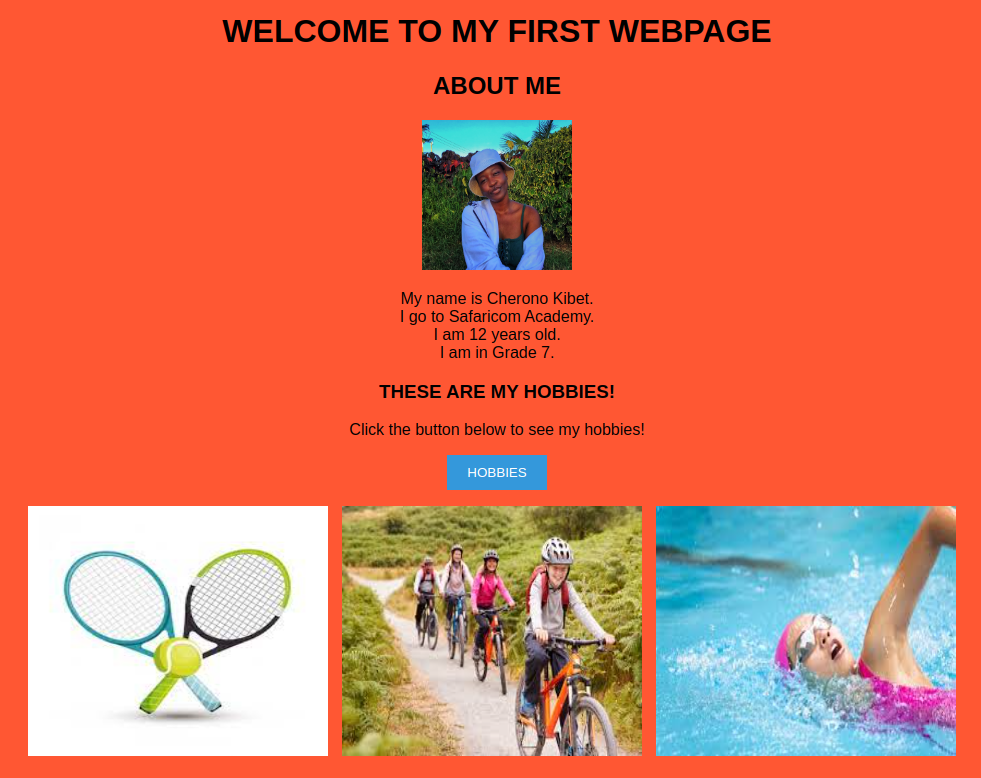
Turning Off the Light **(Hiding)**: If the lights are already on, you turn them off by setting display to "none."

User as the Switch Operator **(User Interaction)**: Just like you operate a light switch at home, you, as the user, become the switch operator. Clicking the button is like flicking the light switch, making the hidden room **(image container)** appear or disappear, similar to turning the lights on or off in your house.

**NOTE:**

**Now run the index.html on local to see the final outcome of your webpage. Let the kids see the changes that happen when the button is clicked. You can add more images to see how they align.**

**See below for final output.**

****

**d.) DAY 3 PROJECT**

They will now create their own project. Let them change the title, use different fonts, background colour, different images and align their content in different ways. (With the help of the mentors of course! 😀)