INFO1111: Computing 1A Professionalism

2023 Semester 1

Self-Learning Report

Submission number: 1

Github link: https://github.com/JerryJJJJin/INFO1111-.git

Student name	Jerry Jin
Student ID	530477150
Topic	JavaScript *Note: This must be the same as was in your topic approval*
Levels already achieved	none
Levels in this report	A,B,C

Contents

1. Level A: Initial Understanding

1.1. Level A Demonstration

1. Embedding javascript into a webpage to be executed by the web browser
2. javascript functions, build functions and pass data through them
3. Accessing webpage elements using Document Object Model (DOM)

1.2. Learning Approach

For this self-learning topic, I approached learning by following these steps:

1. Identify my goals: my goal for the course is to demonstrate level A objectives first and foremost. This guided my path of learning as everything I came across I tried to relate to my 3 goals as stated in 1.1.

2. Doing general research: As I have never done any software development before I did not know exactly what javascript was used for. So I googled about the functionality of javascript and asked a few of my older friends regarding their experience with JS.

3. Learning the basics: I went onto Codecademy and started a free introduction to javascript course to learn the fundamentals such as syntax, data types, operators and functions.

4. Expanding my knowledge: while learning the basics, I went on youtube to watch some tutorial videos and easy follow-along videos.

5. Putting it all together: By following the youtube videos, I tried to replicate their javascript project and customized it to my liking. This helped me as while I was learning I always wondered how to visualize the javascript code or what the code represented visually.

1.3. Challenges and Difficulties

The following is a list of difficulties I experienced when learning JS:

1. Understanding Syntax: When first doing some initial research about JS, I found the example codes to be really hard to understand due to two reasons. 1, I had no knowledge about JS in general and 2, the syntax used is different to languages such as Python. For example, console.log is essentially print in Python.

2.Live server: When watching the youtube tutorial videos, most assume that you know the basics of web development such as how to visualize your code in VS code for example. It was frustrating at first as I did not know what this function was called so I could not even search on google regarding the issue.

3.Html: As websites are written in Html and JS is used for its functionality, inevitably, during the learning phase I came across some Html which I did not understand.

4.DOM was the most challenging part to learn for this level, as it modifies the webpage it sometimes is difficult to follow from which step it started or which aspect of the webpage is it changing and how is it able to do that.

1.4. Learning Sources

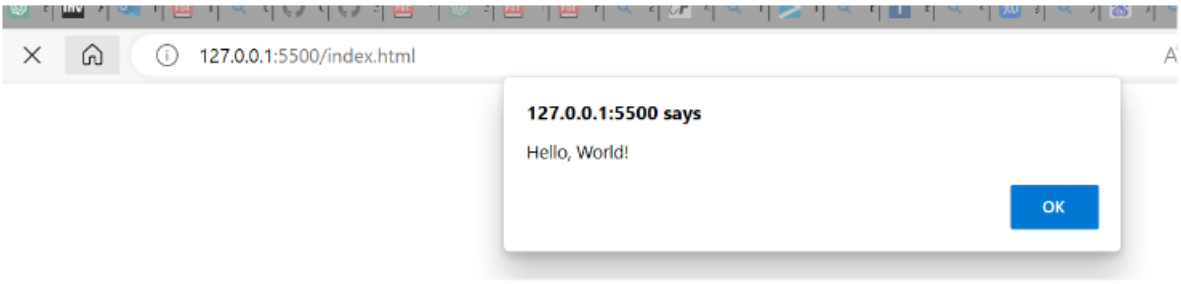
Learning Source - What source did you use? (Note: Include source details such as links to websites, videos etc.).	Contribution to Learning - How did the source contribute to your learning (i.e. what did you use the source for)?
https://www.youtube.com/watch?v=821C5aJ3SLM	Linking JS to Html
https://www.youtube.com/watch?v=W6NZfCO5SIk	basic knowledge of JS, what it is, variable, functions
https://www.youtube.com/watch?v=FOD408a0EzU	JS function
https://www.youtube.com/watch?v=DGaRuK7D92M	DOM exercise
https://www.codecademy.com/courses/introduction-to-javascript/lessons/introduction-to-javascript	introduction to JS, data types, basic syntax and built in functions

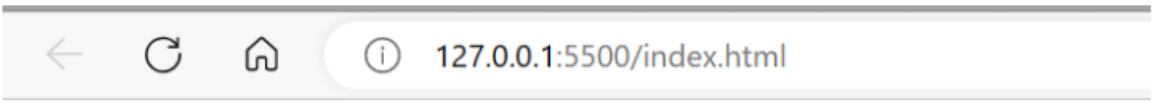
1.5. Application artifacts

Embedding javascript into a webpage to be executed by the web browser

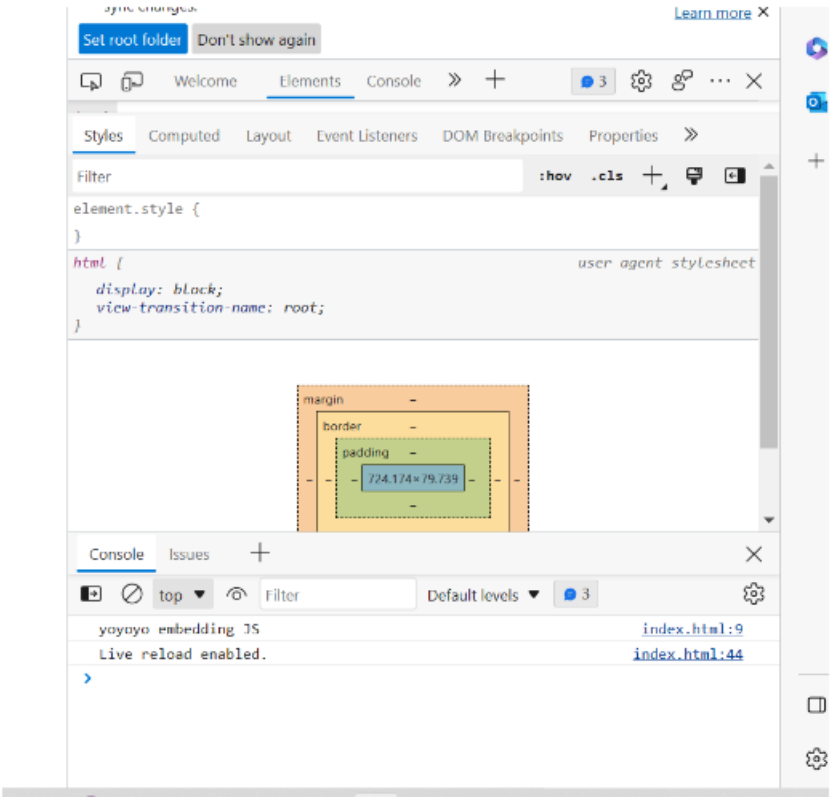
```
<!DOCTYPE html>
<html>
  <head>
    <title>Embedding javascript into a webpage to be executed by the web browser</title>
    <script src = "Hello.js">
        // inside the script tag, we can write JavaScript
    </script>
    <script>
        console.log('yoyoyo embedding JS')
    </script>
  </head>
<body>
    <h1>What's your name user!</h1>
    <h2>Nice to meet you</h2>
    <script>document.body.innerHTML = '<h1>Javascript says hello to you</h1>'</script>
</body>
</html>
```

Results





Javascript says hello to you



What does it do?

First, upon opening the webpage, an alert will pop up saying “Hello World!”
Then in the console, it will print “yoyoyo embedding JS”
Finally the webpage will display “Javascript says hello to you”

How does it work?

Generally, JS code is executed line by line hence the web browser upon loading the webpage will interpret the code from top to bottom. JS code is placed within the <script> tags of an HTML document which embeds it into the webpage for the browser to execute. However, the placement of these <script> tags within the head/body section will affect how and when the JS code is being executed.

As the head section is executed first, it can be used to run JS code before any content is loaded on to the webpage. Eg. initializing libraries or global variables. However, for demonstration purposes, I implemented the built in function alert to display a message box with an ok button to show that it is in fact executed first as the webpage is blank at that point.

<script src = "Hello.js">

// inside the script tag, we can write JavaScript

</script>

So this chunk will be executed first. It sources and run JS code from the Hello.js file which contains alert("Hello, World!"); which causes an alert window to pop up.

<script>

console.log('yoyoyo embedding JS')

</script>

Then it will execute this part which essentially prints 'yoyoyo embedding JS' into the console

<h1>What's your name user!</h1>

<h2>Nice to meet you</h2>

Then it will execute these 2 lines printing "What's your name user!" followed by "Nice to meet you" in the next line. However you will not see these two lines as →

<script>document.body.innerHTML = '<h1>Javascript says hello to

you</h1>'</script>

This code is executed next which replaces the two messages in the body and display "Javascript says hello to you" onto the webpage with h1 (heading 1) font size.

How did I create it?

Using the basic HTML file, Inside the head section, I added a script tag that referred to an external JS file "Hello.js". This script tag is using the src attribute to specify which file it is souring code from

Next, I added another script tag inside the head section that uses the console.log() function to print a message to the browser console when executed

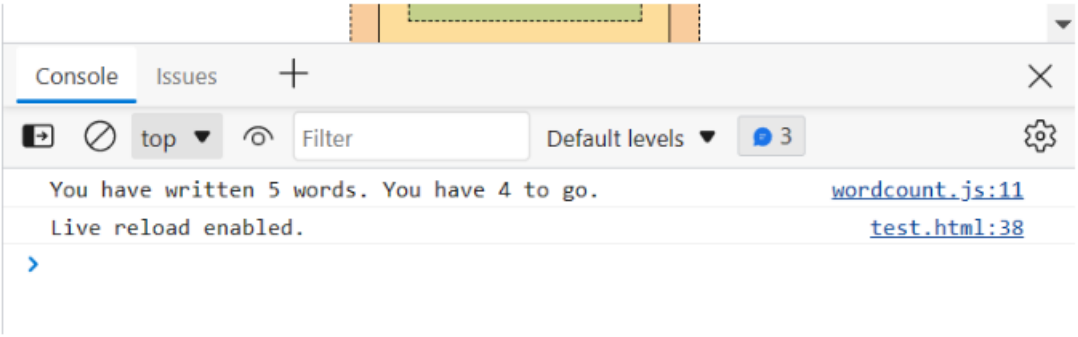
Then, I added another script tag inside the body section that uses the document.body.innerHTML to change the message of the body element to a Header message Finally opened the file with the liver server extension for visualization

Javascript functions, build functions and pass data through them

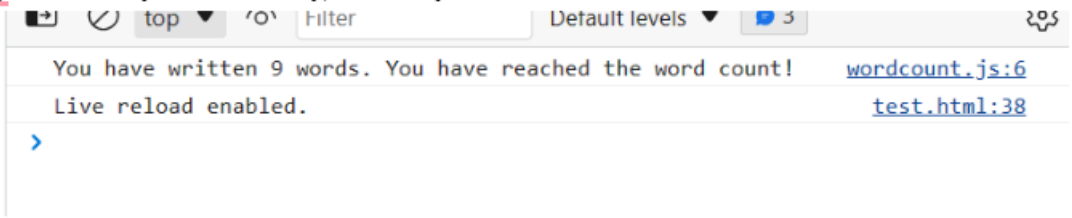
```
function wordcount(text, target) {
    let words = text.split(" ");
    let wordcount = words.length;

    if (target * 0.9 <= wordcount && wordcount <= target * 1.1) {
        console.log("You have written " + wordcount + " words. You have
reached the word count!");
    }

    else if (wordcount < target * 0.9) {
        wordsleft = target * 0.9 - wordcount;
        console.log("You have written " + wordcount + " words. You have "
+ wordsleft + " to go.");
    }

    else if (wordcount > target * 1.1) {
        wordsover = wordcount - target * 1.1;
        console.log("You have written " + wordcount + " words. You are " +
wordsover + " words over.");
    }
}

wordcount(essay, 10);
```

Results

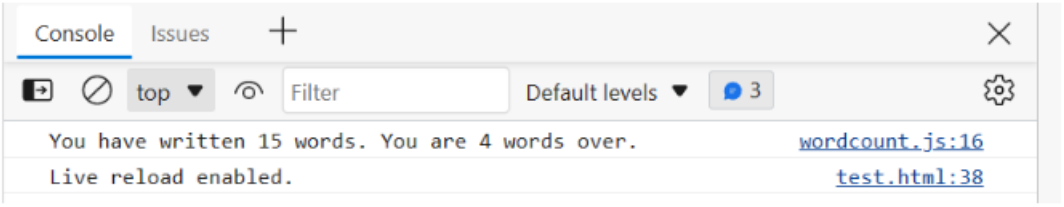
If essay = “Hello my name is Jerry!”



If essay = “Hello my name is Jerry, I am very cool!”



If essay = “Hello my name is Jerry, I love to chat and chat and chat and chat.”



What does it do?

This is a function that aims to count the number of words in an essay and if it reached a 10-word count (just for the ease of demonstration the word count is low). If the essay is within plus minus 10 percent of the word count then it is within the target. However, if the wordcount of the essay is < 9 or > 11 i.e. greater than the 10 percent range then it has not reached the 10 wordcount target.

How does it work?

function wordcount(text, target)

The function wordcount takes in two parameters: text and target

let words = text.split(" ");

let wordcount = words.length;

Then it splits the text using the split() function by removing the spaces forming an array and counts the number of words through the length function.

if (target * 0.9 <= wordcount wordcount <= target * 1.1)

else if (wordcount < target * 0.9)

else if (wordcount > target * 1.1)

Then it checks if the word count is within plus minus 10 percent of target word count using conditional statement and mathematical and logical operators.

If the word count is within the target range, it prints a message to the console indicating that it has been reached.

If the word count is less than 90 percent of the target, it calculates the number of words left to reach the target and prints a message to the console indicating how many words have been written and how many are left to reach the target.

If the word count is more than 110 percent of the target, it calculates the number of words over the target and prints a message to the console indicating how many words have been written and how many are over the target.

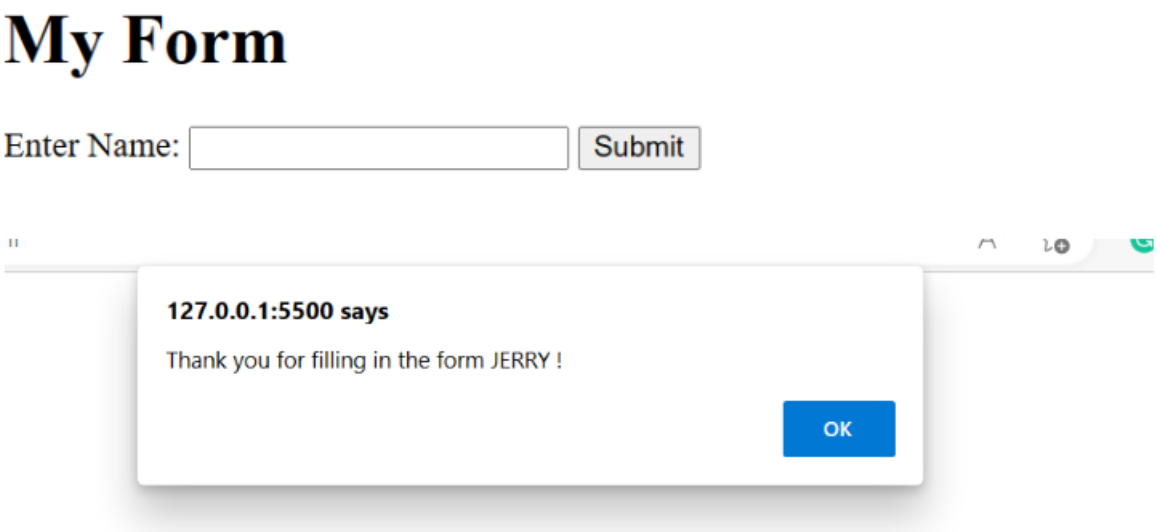
How did I create it?

1. Defining the function with parameters
2. Count the number of words
3. Write Conditional statements for the respective situations
4. Write the respective print statements
5. Call the function with essay and 10 as arguments

Accessing webpage elements using Document Object Model (DOM)

```html
<!DOCTYPE html>
<html>
<head>
    <title>My Form</title>
    <script>
        function displayGreeting() {
            let name = document.getElementById("nameInput").value;
            alert("Thank you for filling in the form " +
name.toUpperCase() + " !")
        }
    </script>
</head>
<body>
    <h1>My Form</h1>
    <label>Enter Name:</label>
    <input type="text" id="nameInput">
    <button onclick="displayGreeting()">Submit</button>
</body>
</html>
```

Results



What does it do?

The webpage has a form for which the user will enter his name. Upon clicking the submit button, the program will alert the user with a Thank you message.

How does it work

<h1>My Form</h1>

<label>Enter Name:</label>

<input type="text" id="nameInput">

This will create a Heading My Form and an empty box to enter name. Whatever the user inputs, it will be stored with an id of nameInput

<button onclick="displayGreeting()">Submit</button>

This attaches an event listener to the submit button, when clicked by the user, the displayGreeting function is called

function displayGreeting()

let name = document.getElementById("nameInput").value;

alert("Thank you for filling in the form " +

name.toUpperCase() + " !")

The displayGreeting function retrieves the name from the input element using document.getElementById("nameInput").value and displays a personalized greeting using alert().

How did I create it?

1. Write the displayGreeting function in the head section as this should be loaded first
2. Write the heading and box for the form
3. Create the variable for the user input
4. Create the submit button as an event listener to interact with the user

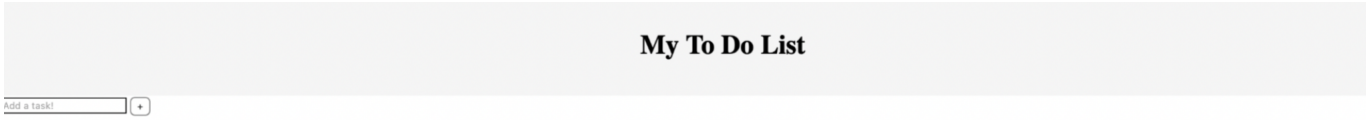
2. Level B: Basic Application

2.1. Level B Demonstration

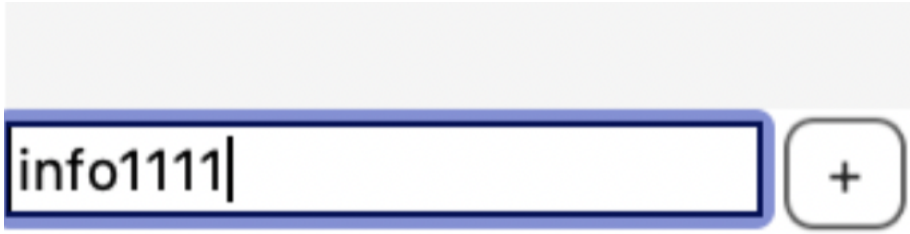
To display my level B understanding of Javascript, I decided to create a simple To-do list using event based triggers including: click event, mouse event and keyboard event. The To-do list has a heading of "My To Do List" underneath is a text box and a "+" button to add the tasks, which enlarges when hovered over. Task then appears in a gray block with a "done" button to delete the task which clicked. Users can also navigate tasks using the up and down arrow key which highlights the selected task in blue and enlarges it.

2.2. Application artifacts

What does it do? Upon opening the webpage it should look like this

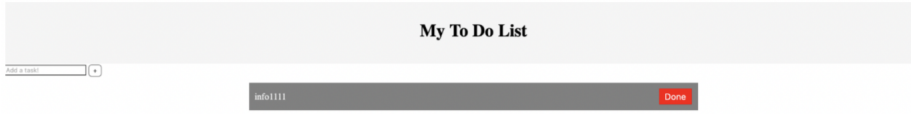


In the text box, the user can then enter any tasks to add to the to-do list



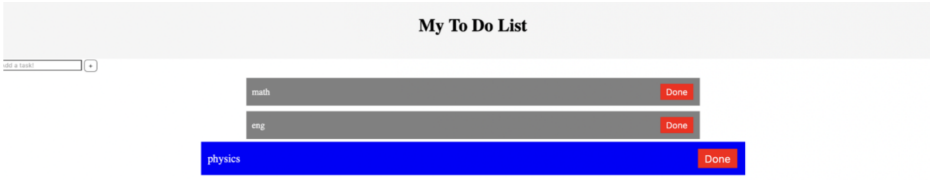
If the user hovers the mouse over the Add button, the button will enlarger

After clicking the Add button, the task will appear like so



The user can add a number of tasks and can select them using the up and down arrow keys. The selected taks will appear in blue and enlarge

The user can also delete any tasks by clicking the Done button and the task will be removed from the list.



How does it work?

Style

This section of the code is mainly used to format my code, colours, align everything, and the transition effects for the enlargement feature.

```
<input type="text" id="textInput" placeholder="Add a task!" />
<button id="addButton">+</button>
<div class="center-container">
    <div id="container"></div>
</div>
```

First creates the text box and a button to add tasks to the to-do list then this section placed the text box and button on the left and the task list in the center of the page

```
const addButton = document.getElementById('addButton');
const textInput = document.getElementById('textInput');
const container = document.getElementById('container');
const blocks = []; // store references to all blocks
let selectedBlockIndex = -1; // initially, no block is selected
```

1. addButton: This variable holds a reference to the DOM element with the
2. ID addButton. It is the button that users click to add new tasks.
3. textInput: This variable holds a reference to the DOM element with the
4. ID textInput. It is the text input field where users type their tasks.
5. container: This variable holds a reference to the DOM element with the
6. ID container. It is the container where new task blocks will be added.
7. blocks: This variable is an array that will store references to all the task blocks. It's initialized as an empty array because there are no blocks when the script first runs. As users add new tasks, references to the newly created task blocks will be pushed into this array.
8. selectedBlockIndex: This variable represents the index of the currently selected task block in the blocks array. It is initially set to -1, which means no block is selected.

```
addButton.addEventListener('click', function(event) {
  const blockContent = textInput.value;
  if (!blockContent) return; // do not create empty blocks

  const block = document.createElement('div');
  block.className = 'block';

  const content = document.createElement('span');
  content.textContent = blockContent;
  block.appendChild(content);

  const doneButton = document.createElement('button');
  doneButton.textContent = 'Done';
  doneButton.className = 'doneButton';
  doneButton.addEventListener('click', function() {
    container.removeChild(block);
    const index = blocks.indexOf(block);
    if (index > -1) {
      blocks.splice(index, 1);
      if (selectedBlockIndex === index) {
        selectBlock(-1);
      }
    }
  });
  block.appendChild(doneButton);

  container.appendChild(block);
  blocks.push(block);

  textInput.value = ''; // clear the input field
});
```

This section essentially codes for the actual functionality of the add and done button. for example adding a task to the list but not adding any empty tasks and removing tasks blocks and from the array.

```
// mouse hover over add button enlargement transition
addButton.addEventListener('mouseover', function(event) {
  addButton.style.transform = 'scale(1.2)';
});

addButton.addEventListener('mouseout', function(event) {
  addButton.style.transform = 'scale(1)';
});
```

This section codes for the effect where if the user's mouse is hovering over the Add button, the icon will become bigger. And if the user's mouse is off the icon, the icon will shrink in size back to its original size.

```
document.addEventListener('keydown', function(event) {
  // only if blocks are selected
  if (blocks.length === 0) {
    return;
  }

  // arrow key events
  if (event.keyCode === 38) { // up arrow
    selectBlock(selectedBlockIndex - 1);
  }
  else if (event.keyCode === 40) { // down arrow
    selectBlock(selectedBlockIndex + 1);
  }
});
```

This code is to allow the navigation between different tasks on the to-do list and makes sure that there is at least one block to select. Up arrow selects the task block with index 1 smaller than the current one and the down arrow selects the task block with index 1 greater or the one below.

```
function selectBlock(index) {
  // unselect the currently selected block
  if (selectedBlockIndex >= 0) {
    blocks[selectedBlockIndex].classList.remove('selected');
  }

  // make sure index is within range
  selectedBlockIndex = (index + blocks.length) % blocks.length;

  // select the new block and make it bigger
  blocks[selectedBlockIndex].classList.add('selected');
}
```

This code allows for the selection of the task block i.e. deletion, and enlargement as per the annotation.

How did I create it?

1. Went on to youtube to find inspirations and tutorials on how to create a To-do list.
2. Create separate sample files that are coded for different functions such as the text box, add button which becomes bigger, and selecting task block with up and down arrow keys.
3. Create a new html file that contains all of these sample codes put together. This was not formatted yet so everything looked really messy. Had one format section to align everything
4. Chose my favorite colors for various objects
5. Test out on live server

3. Level C: Deeper Understanding

Level C focuses on showing that you have actually understood the tool or technology at a relatively advanced level. You will need to compare it to alternatives, identifying key strengths and weaknesses, and the areas where this tool is most effective.

3.1. Strengths

Javascript being a popular tool of web development has many key strengths: Wide browser support: Javascript is supported by all major web browsers, allowing develo Pres to create cross platform applications

1. Active community: easy to get help and find inspirations, helping developers to build applications More efficiently and effectively

2. Event-based programming: allows developers to create responsive and interactive web applications That react to user's input and other events eg. keyboard and mouse commands.

3.2. Weaknesses

However Javascript is not without its disadvantages:

1. Learning curve: This is more of my personal opinion, but as someone who has never Done css nor HTML, I realized that I needed to learn some basics alongside Javascript. This is due to the 3 are required in order to make a webpage and java only allows for the Interactivity.

2. Security: Javascript code is available to be viewed at the client side which opens up opportunities for Cyber attacks and information leaks

3.3. Usefulness

Javascript could be really useful in the scenario of developing a dynamic and interactive web Application such as a weather dashboard. In this scenario, users in input their location and Javascript is able to fetch the data from a weather API and dynamically update the page with real-time temperature, Humidity and forecasts. Additionally, it can also create interactive visuals For example charts and maps to display weather patterns. Hence the dynamic and interactive Nature Javascript provides the users with a better understanding of the weather, and overall a more engaging experience.

3.4. Key Question 1

When should you use, and not use Javascript?

Use Javascript when you want to create a dynamic, interactive web application, handle user input, manipulate DOM or communicate with servers without reloading the page. Ideal for building responsive and engaging pages.

Avoid Javascript for content presentation or static websites in these cases HTML and CSS should be used to create a simple, fast-loading page. Additionally, depending on your target audience bandwidth Use Javascript if the target audience is young people with heavy usage broadband plans as Javascript frameworks may be megabytes in size.

Vice versa, avoid if the audience is a remote village with little access to the internet

3.5. Key Question 2

How does Javascript make use of frameworks?

Javascript frameworks render prewritten JS code that produces routine programming features to the development process more efficient. It is completely doable to build applications without frameworks but they provide a template so each time you write code you don't have to start from scratch but rather build upon existing feature sets. A popular Javascript framework is React which offers reusable components and users can also immediately see the applied changes with its hot load feature.

4. Level D: Evolution of skills

4.1. Level D Demonstration

This is a short description of the application that you have developed. (50-100 words). **IMPORTANT:** *You might wish to submit this as part of an earlier submission in order to obtain feedback as to whether this is likely to be acceptable for level D.*

4.2. Application artifacts

Include here a description of what you actually created (what does it do? How does it work? How did you create it?). Include any code or other related artefacts that you created (these should also be included in your github repository).

If you do include screengrabs to show what you have done then these should be annotated to explain what it is showing and what the application does.

4.3. Alternative tools/technologies

Identify 2 alternative tools/technologies that can be used instead of the one you studied for your topic. (e.g. if your topic was Python, then you might identify Java and Golang)

4.4. Comparative Analysis

Describe situations in which both your topic and each of the identified alternatives would be preferred over the others (100-200 words).