



SCHOOL OF COMPUTING, ENGINEERING AND BUILT
ENVIRONMENT

WEB DESIGN TECHNOLOGIES

COURSEWORK 2: INDIVIDUAL REPORT

Link to hosted website: <https://wdt-coursework1.vercel.app/>

Link to Github File: <https://github.com/Mairee14/WDT-Coursework1.git>

Author:

Mary Nene Odeh

Student Number:

S2111000

Table of contents

1. Discussion of Development Approach	3
Justification	5
2. Accessing data from external API	7
3. Alternative ways to display data	14
4. Advantages Of Bespoke Approach	17
5. Disadvantages Of Bespoke Approach	17
6. References	19

1. Discussion of Development Approach

The approach of my website development is having clear and structured layout . I used Figma tool to design the interface layout. However, I was able to create a layout and provide a design that is visually appealing, functional and easy to navigate.

After making designs to the website using figma tool, i implemented it using bootstrap framework,Bootstrap is a framework used in creating responsive web designs, it is a combination of HTML, CSS and javascript that can be used to develop responsive websites quickly. It also have several components and styles that made it consistent to create easily. The grid system for bootstrap is used for designing responsive layouts that makes adjustments to several screen sizes.

LAYOUT

The website contains various pages such as the home page,about us, country page where a user can select a country to display certain information of the country and the cost of living . Contact page and cost of living between cities. The approach utilized in creating this website is a combination of HTML,CSS and javascript.The website consist of multiple HTML pages that were styled using bootstrap framework.The approach involves developing the home page first then the rest of the pages were developed one after the other.

The home page was developed using bootstrap grid system. The grid system provided a way to properly organize the content on the page into column card in rows. The content included on the landing page is a header section with a logo icon at the left hand side and a navigation bar with a dropdown menu that expands to show additional links at the right hand side that provides a user friendly way to navigate through different pages and sections of the website and it contains links. Also, a main section with a medium image and call to action button were it can direct a user to another page for more information, and a footer section with social media links and copyright information

About Page was developed using the same grid system as the home page. The content included a header section with a logo icon and navigation bar with dropdown menu's, a main section with small image on the middle content and a section with information about the website and its founder or developer, and a footer section with a social media links and copyright information

Cost of living on the country facts page

The Country page was developed using the same grid system as the home page ,navigation bars,. Also used a form element and javascript. The content included is a header section with navigation bar ,a main section with a a form for selecting a country, and a search bar on top of main image to search for countries, when a country is selected , the user is redirected to the country page that displays the data of information of the country and map of latitude and longitude shows along of the specific location. Cost of living on the country facts is developed using JQuery,javascript and interactive map and chart. The content include a header section with a navigation bar, with content image a main section with displaying data from external API. It displays the cost of living and along with the map API for any country been selected ,display with the (piechat). Which emphasis on the expenses prices of the chart.

Compare page

Contact Page was developed using a form using the same grid system as the home page. The content included a header section with a logo icon and navigation bar with , a main section with small image on the middle content and two search bar to search and cities and options of the

cities pops on the search bar, when the are searched the data of each cities popup with camparison and a section with the form and a footer section with a social media links and copyright information

Justification

The code starts by selecting the required elements from the DOM using the **getElementById method**. The variable **searchBtn** refers to the search button on the website and the variable **countryInp** refers to the input field where the user can enter the name of the country where he/she wants to search. An event listener is added to the **searchBtn** button that triggers a callback function when the button is clicked

In the callback function, the variable **countryName** is initialized with the value entered by the user in the **countryInp** input field. This value is then used to construct the API **call URL**. The constructed URL is passed to the **fetch** function, which makes an HTTP request to the API.

The API returns data in JSON format, so **the response.json()** method is used to process the response to a JavaScript object. The parsed object is then passed as a **data** parameter to the second **.then()** method.

The data retrieved from the API is then used to update the HTML content of the website. The code extracts relevant information from the **data** object, such as B. Country Name, Capital, Population, Currency and Languages. These values are then used to update the content of certain HTML elements on the website.

For example, the country flag image is displayed using the SVG URL from the API data and the country name is displayed in an **<h2>** element. The country's capital, continent, population,

currency and common languages are also displayed in the corresponding HTML elements.

If an error occurs during an API request, the **.catch()** method is called. The code checks whether the **countryName** variable is empty or not. If it is empty, an error message is displayed informing the user that the input field cannot be empty. If the **countryName** variable is not empty, an error message is displayed prompting the user to enter a valid country name.

In summary, the provided code fetches data from an external API using the Fetch API, parses the JSON data, and updates the HTML content of the web page with the relevant information.

Justification

Firstly, creating a clear design plan using figma helped me understand the clear website's design and layout .Using bootstrap and figma helped to ensure that the website's design was consistent visually appealing by using same styles and components across all pages.

The development approach used in creating the website was effective because it allowed the creation of responsiveness to the website that was easy to navigate and visually appealing. Additionally, the use of form elements and javascript allowed for interactivity and user input. The use of the Map API allowed for the creation of a unique feature that will provide value to the user.

Bootstrap is a great framework for developing responsive websites. It offers several components and styles that make it consistent to create easily, and the grid system helps in designing responsive layouts that can adjust to different screen sizes. Utilizing the Bootstrap framework in creating the website ensured that the website was responsive and had a consistent design across all pages, resulting in an overall cohesive design.

The development approach used in creating the website is also important, as it determines how the website will function and interact with users. By using HTML, CSS, and JavaScript, it is possible to create a dynamic website with interactive features, such as forms and maps, that enable user input and engagement.

In conclusion, the combination of my clear and structured layout using Figma, the use of the Bootstrap framework to ensure responsiveness and consistency, and the utilization of HTML,

CSS, and JavaScript in creating a dynamic website with interactive features, is an effective approach to website development. The approach adopted in creating the website ensures that it is visually appealing, functional and easy to navigate, providing users with a positive experience.

2. Accessing data from external API

The API documentation typically includes information about API endpoints, request parameters, response formats, and authentication requirements. It is important to read and understand the documentation before using the API. To access the information in APIs, authorization is necessary. Typically, to do this, getting an API key or code from the API source and add it to the request's headers or parameters. Usually, the API includes a description of the verification procedure.

The API request construction process comes next. Here, the API addresses any necessary specifications are specified. Examples to build a request are typically provided in the API instructions. Any computer language that supports making HTTP queries, such as the JavaScript `fetch()` method, can be used to create the request. An HTTP request can be used to transmit the API request once it has been created and sent to the API server. An HTTP request library or package should be available in the computer language being used. The API service will respond with an HTTP answer, which is what will be displayed. The program or script can now use the data it has received from the API. This may entail altering the data, showing it in a user interface, keeping it in a database, or using it to drive logic in the program.

Furthermore, this code is a JavaScript snippet that retrieves information about two cities provided by the user through input fields on a web page. The code consists of two parts:

```

// Get the values of the input fields
const country1 = city1Input.value.toLowerCase().split(",")[1];
const city1 = city1Input.value.toLowerCase().split(",")[0];
const country2 = city2Input.value.toLowerCase().split(",")[1];
const city2 = city2Input.value.toLowerCase().split(",")[0];

let city1Info = await fetchCityInfo(city1, country1);
let city2Info = await fetchCityInfo(city2, country2);
console.log(city1Info);
console.log(city2Info);

const inputOne = {...Object.values(city1Info)};
const inputTwo = {...Object.values(city2Info)};

```

Figure 1.

Part 1: The first part retrieves the values of the input fields and splits them into city and country. It does this by using the `toLowerCase()` method to convert the input to lowercase letters and the `split()` method to separate the city and country by the comma (,) character. The resulting variables are `country1`, `city1`, `country2`, and `city2`.

Part 2: The second part makes use of two asynchronous functions, `fetchCityInfo()` and `await`, to fetch information about the two cities from some external source. The `fetchCityInfo()` function takes two arguments, city and country, and returns a promise that resolves to an object containing various information about the city, such as its latitude, longitude, population, and timezone. The `await` keyword is used to wait for the promise to resolve before moving on to the next line of code.

After fetching the information for the two cities, the code logs the resulting objects to the console using `console.log(city1Info)` and `console.log(city2Info)`. Finally, the code creates two new objects, `inputOne` and `inputTwo`, by spreading the values of the `city1Info` and `city2Info` objects into new objects using the spread operator (`{...}`).

Overall, this code retrieves information about two cities from user input fields, fetches additional information about the cities from an external source, and creates two new objects containing the

city information.

However, breaking it down into Accessing data from external API

```
// Get the values of the input fields
const country1 = city1Input.value.toLowerCase().split(",")[1];
const city1 = city1Input.value.toLowerCase().split(",")[0];
const country2 = city2Input.value.toLowerCase().split(",")[1];
const city2 = city2Input.value.toLowerCase().split(",")[0];
```

Figure 2.

In this code, the value property of the city1Input and city2Input elements is accessed. This property contains the text entered by the user into the input fields.

The toLowerCase() method is called on the input field values to convert the text to lowercase letters. This ensures that the city and country values are consistent, regardless of whether the user entered them in uppercase or lowercase letters.

Next, the split() method is used to separate the city and country values based on the comma (,) character. The result of this method call is an array containing two elements: the city and the country.

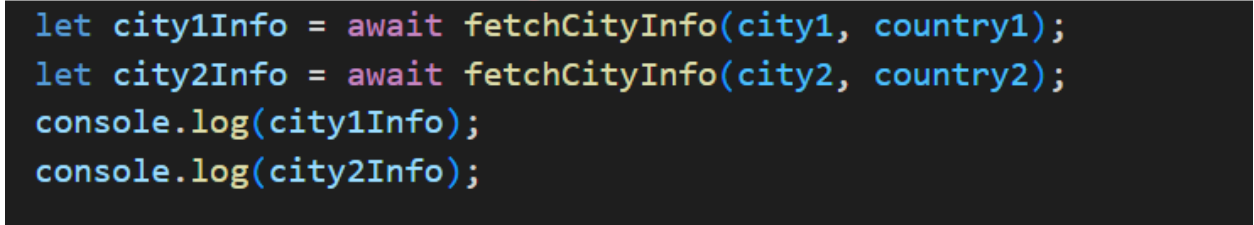
By accessing the elements of the resulting array using array indexing (e.g. [0] for the city and [1] for the country), the code is able to extract the city and country values for each input field separately.

The resulting city1 and city2 variables contain the names of the cities, while the country1 and country2 variables contain the names of the countries in which the cities are located.

This information is then passed as arguments to the fetchCityInfo() function, which retrieves information about the specified cities from the external API. The await keyword is used to pause the execution of the program until the city information has been retrieved, ensuring that the

program does not attempt to use this information before it is available.

Finally, the `city1Info` and `city2Info` objects are logged to the console using the `console.log()` function. This allows to see the information retrieved from the external API and verify that the API is working correctly.



```
let city1Info = await fetchCityInfo(city1, country1);
let city2Info = await fetchCityInfo(city2, country2);
console.log(city1Info);
console.log(city2Info);
```

Figure 3.

After obtaining the city and country names, the code is calling an asynchronous function `fetchCityInfo()` to retrieve information about each city from an external API. This function likely uses the `fetch()` method to make an HTTP request to the API endpoint and returns the response in a JSON format. The `await` keyword is used to wait for the response to be returned .

There are different options available for accessing data from an external API, each with its own advantages and disadvantages. Here are some of the most ways to access data from an external API:

<p>Fetch API: The Fetch API is a modern, built-in JavaScript API that provides a simple interface for making HTTP requests. It supports promises and <code>async/await</code> syntax for handling asynchronous requests.</p> <p>The Fetch API is widely supported in modern browsers, and it is easy to use, making it a popular choice for accessing data from external APIs. To use the Fetch API, you simply call the <code>fetch()</code> function with the URL of the API endpoint and any additional parameters, such as headers or request type. Once the request is complete, you can parse the response data and use it in your application. In the example code provided in the question, the Fetch API is used to access data from the external API.</p>	<p>Server-side requests: In some cases, it may be necessary to access external APIs from the server-side rather than the client-side. This can be useful for adding additional security or caching to API requests, or for accessing APIs that require authentication. Server-side requests can be made using server-side languages like Node.js or PHP, or using cloud services like AWS Lambda or Google Cloud Functions.</p>
<p>Third-party libraries: There are many third-party libraries available for making HTTP requests, such as Axios, jQuery, and Superagent. These libraries often provide more features and customization options than the Fetch API or XHR. However, they require additional code and dependencies and can increase the size of your codebase. Using a third-party library can also introduce security vulnerabilities, so it's important to choose a reputable library and keep it up to date.</p>	<p>XMLHttpRequest (XHR): XMLHttpRequest is an older API that has been widely used in the past for making HTTP requests. It supports both synchronous and asynchronous requests and provides more fine-grained control over the request and response than the Fetch API. However, it requires more verbose code than the Fetch API and is not as easy to use. XHR is still supported in modern browsers, but it has been largely superseded by the Fetch API.</p>

Dynamic Display

This code is designed to display information about cities, their prices and other details on a web page. It starts by defining some constants and a function to display the contents of a list. Then it uses the `innerHTML` property to dynamically generate the content of the web page.

The code is making use of jQuery, a JavaScript library designed to simplify DOM manipulation. However, the code sample does not use any jQuery-specific syntax. It is just using vanilla

JavaScript to get DOM elements by their id and set their innerHTML property.

The function displayContent takes a list and loops through it using forEach. It destructures the Cost and Value properties of each item and adds them to a template literal that represents a table row. It returns the concatenated list items.

```
<div>
  <h4>Resturant Prices</h4>
  <table>
    <thead>
      <tr>
        <th>Item</th>
        <th>Cost</th>
      </tr>
    </thead>
    <tbody>
      ${displayContent(inputOne[8])}
    </tbody>
  </table>
</div>
```

Figure 4.

The main section of the code uses to generate the HTML for the web page. It sets the innerHTML property of the resultDiv element to a long string that represents the entire page. This string includes div elements, h4 headings, and spans to display the details of each city. It also includes tables to display the prices of various items in each city, and it uses the displayContent function to populate the rows of each table with data.

The resultDiv.innerHTML property is being used to set the HTML content of a div element with the class cityinfo. This div element contains two child div elements with the classes inputone and inputtwo. The data for each city is represented in the variables inputOne and inputTwo, which are assumed to be arrays containing specific data in a specific order. The resultDiv variable references an HTML element where the comparison table will be displayed. The HTML markup generated by the code is added to the innerHTML property of this element, effectively replacing any existing content in the element with the new markup.

The code makes use of jQuery, specifically the \$() function, to simplify DOM manipulation. For

example, `searchCity.addEventListener('click', function() { ... })` could be replaced with `$(searchCity).click(function() { ... })` for a more concise and readable syntax.

Each `div` element contains multiple child `div` elements that represent different types of information such as country, capital, restaurant prices, etc. Each of these child `div` elements contains an `h4` element with a heading and a `span` element that contains the corresponding data from the `inputOne` or `inputTwo` arrays.

For example, the first child `div` element in the `inputone` `div` represents the country and contains an `h4` element with the heading "Country" and a `span` element that contains the country name from `inputOne[1]`.

There are also several table elements that represent different types of prices such as restaurant prices, market prices, etc. Each table element contains a `thead` element with a header row and a `tbody` element that contains the corresponding data from the `inputOne` or `inputTwo` arrays.

For example, the table element with the heading "Restaurant Prices" contains a `thead` element with a header row that contains two `th` elements for "Item" and "Cost". The `tbody` element contains the data from `inputOne[8]` using a helper function called `displayContent` to create the rows of the tables.

In addition to the interactivity When a user selects a country using the search input field and clicks the search button, the code fetches information about the country from the url <https://restcountries.com/v3.1/> API and displays it on the page. It also calls the `fetchCityInfo()` function, passing in the name of the country and its capital city as parameters.

The `fetchCityInfo()` function uses the `fetch()` method to send a GET request to the <https://cities-cost-of-living-and-average-prices-api.p.rapidapi.com/> API, passing in the country and city parameters as query parameters in the URL. It then extracts the required data from the response using `await` and `json()` methods.

Once the data is retrieved, the code creates a pie chart using the `createPieChart()` function,

passing in the data as parameters. This function uses the Chart.js library to create a pie chart that shows the user the breakdown of expenses in various categories like restaurants, markets, transportation, etc. for the city in that country.

This chart is displayed below the other information about the country that was fetched from the restcountries API. The interactivity comes in the form of the ability to hover over different segments of the chart and see the exact percentage and value associated with that segment. The code accomplishes this using the Chart.js library's built-in tool tip functionality.

The interactivity also includes error handling, which displays appropriate messages when the user enters invalid input or when there is an error retrieving data from the API.

And also, for the Map interactivity initializing a new map instance using the Mapbox API, and define an array of cities with their respective latitude, longitude, population, and flag URL. I also then use the `forEach()` method to loop through the country array and create a new marker element for the country. I then, added a popup to the marker that displays information about the city, population, and flag, and attach it to the marker using the `setPopup()` method. Finally, we add the marker to the map using the `addTo()` method.

With this code, users can interact with the marker on the map to display information about the country. When a user clicks on the marker, a popup will appear that displays the the country, city name, population, and flag.

3.Alternative ways to display data

Alternative ways to display data to various techniques that can be used to present information to users beyond the table.. Some of the alternative approaches that can be used include:

Custom JavaScript: One alternative approach for displaying data is to write custom JavaScript code from scratch. This approach involves manually coding the HTML, CSS, and JavaScript to create the user interface and display the data. This approach provides complete control over the code and allows for a highly customizable user interface. However, it requires a deep understanding of web development technologies and can be time-consuming to implement.

One of the advantages of using custom JavaScript is the ability to optimize the code for performance. Developers can use best practices such as minimizing the use of DOM manipulation, reducing the number of HTTP requests, and optimizing code for faster execution to create fast-loading and efficient applications. However, custom JavaScript also has its disadvantages. It requires a high level of technical expertise, and developers need to have a deep understanding of HTML, CSS, and JavaScript to implement this approach. Moreover, custom JavaScript code can be time-consuming to develop, especially for large and complex applications. It is also harder to maintain and update the code, as it may not have the support of a community of developers who can contribute to the codebase

jQuery: One alternative approach for displaying data is to write jQuery code from scratch is a JavaScript library that simplifies the process of manipulating HTML documents and handling events. It provides a concise syntax for selecting and manipulating HTML elements, making it easier to create dynamic user interfaces. jQuery also provides a set of built-in functions for

With jQuery, developers can easily traverse the Document Object Model (DOM) and manipulate the HTML elements using CSS selectors. The library provides a vast range of methods and functions that can be used to modify the content and appearance of web pages. For example, developers can use the `.addClass()` method to add a CSS class to an element, the `.hide()` method to hide an element, or the `.attr()` method to get or set the value of an attribute.

Another advantage of using jQuery is that it provides a set of built-in functions for handling common tasks, such as making AJAX requests and animating elements. With the `$.ajax()` method, developers can easily send and receive data from a server without reloading the page. And with the `.animate()` method, developers can create smooth animations and transitions for elements on the page.

handling common tasks, such as making AJAX requests and animating elements. Using jQuery can save time and simplify the code, but it requires learning a new library and can add

additional overhead to the page.

React: One alternative approach for displaying data is to write jQuery code from scratch building user interfaces that focuses on component-based architecture. It allows developers to build complex UIs using a set of reusable components that can be easily combined and updated. React uses a virtual DOM to optimize performance and minimize the number of updates required to the actual DOM. React also has a large and active community with many third-party libraries and tools available. However, it requires learning a new library and can add additional overhead to the page.

One of the key advantages of React is its component-based architecture, which allows developers to break down complex user interfaces into smaller, reusable parts. This makes it easier to manage and update large applications, as well as to collaborate with other developers. React components can also be easily combined and composed to create new UI elements, which can be reused throughout the application.

React's use of a virtual DOM is another major advantage, as it allows React to update the UI more efficiently than traditional DOM manipulation. The virtual DOM is a lightweight representation of the actual DOM, which React uses to compare the current state of the UI with the desired state. When changes are made to the UI, React updates the virtual DOM first and then applies the necessary changes to the actual DOM, reducing the number of updates required and improving performance.

Vue: Vue is a progressive JavaScript framework for building user interfaces. It focuses on the declarative rendering of data and provides a set of reactive data bindings to simplify the process of creating dynamic user interfaces. Vue also has a modular architecture, which allows developers to use only the parts of the framework that they need. It has a small learning curve and can be easily integrated into existing projects. However, it requires learning a new framework and can add additional overhead to the page.

However, like other frameworks and libraries, Vue requires learning a new set of tools and concepts. Additionally, the use of a framework like Vue can add additional overhead to the page, which can affect performance. It is important for developers to carefully consider the trade-offs and choose the right tools for their specific project requirements.

4. Advantages Of Bespoke Approach

Making use of a bespoke approach can produce a greater result in a unique and optimized site that is tailored to the needs of the user. However, an analysis for developing the website using a bespoke approach that includes the advantage and disadvantage. The advantages of a bespoke approach include the following:

1. This bespoke approach allows for complete customization of the website, as every aspect of the website is designed and crafted from scratch, it allows this approach for full control over the website's functionality, features and design.

2. This approach allows for optimized performance and faster page load times.

3. This approach provides flexibility as it is not limited by the features and constraints of prebuilt CMS templates. If there are any changes that are required, they can be made quickly and easily.

4. Unique designs, since this approach requires starting from scratch, the designer can create a unique and original design that reflects the user.

5. Better security: Bespoke websites are less vulnerable to security threats because hackers cannot exploit known vulnerabilities in popular CMS platforms like WordPress. The bespoke approach allows for more robust security measures to be put in place, reducing the risk of cyber attacks.

6. Scalability: A bespoke website can be developed with scalability in mind, allowing for easy expansion and upgrades as the business grows. In contrast, pre-built CMS platforms may require additional plugins or third-party services to achieve the same level of scalability, which can lead to compatibility issues and slower performance.

7. Greater control over SEO: Bespoke websites allow for greater control over search engine optimization (SEO), as the developer can optimize every aspect of the site for better visibility in search engine results pages (SERPs). This can be especially important for businesses that rely heavily on organic traffic.

5. Disadvantages Of Bespoke Approach

1. Developing a website from scratch can take longer than using a prebuilt CMS. Every

aspect of the website needs to be designed and developed , which can take more time rather than using pre-built templates.

2. To maintain a bespoke website approach it can be more complex than CMS website, as every aspect of the website has been custom built. It requires more time, effort to maintain.
3. This does not require prebuilt support, with a bespoke approach.
4. Higher cost: Developing a bespoke website from scratch is generally more expensive than using a pre-built CMS platform like WordPress. This is because it requires more time and resources to design, develop, and test.
5. Longer development time: A bespoke website takes longer to develop than a pre-built CMS website, as every aspect of the site needs to be created from scratch. This can delay the launch of the website and potentially impact the business's online presence.
6. Greater dependence on developers: A bespoke website requires a team of experienced developers to design, develop, and maintain the site. This means that the business may become overly reliant on these developers and struggle to make changes or updates if they are not available.
7. Lack of community support: Pre-built CMS platforms like WordPress have large and active communities that provide support, plugins, and updates. With a bespoke website, the business will need to rely on the developers to provide ongoing support and maintenance.
- 8.

Compared to making use of content management system such as WordPress

Advantages of using a CMS such as WordPress:

1. Cost-effective: Using a pre-built CMS like WordPress can be a cost-effective solution for building a website. There are many free or low-cost templates and plugins available, and you do not need to hire a team of developers to create your website.
2. Ease of use: CMS platforms like WordPress are designed to be user-friendly, even for those with little to no coding experience. You can easily create and manage content, add functionality, and customize your website using pre-built templates and plugins.
3. Community support: CMS platforms like WordPress have large and active communities that provide support, updates, and new features regularly. This can help ensure that your website stays up-to-date and secure.

Disadvantages of using a CMS such as WordPress:

1. Limited customization: While CMS platforms like WordPress allow for some customization, they are limited in terms of design and functionality. Your website may look similar to others using the same templates, and you may not be able to add certain features without extensive coding.
2. Security risks: Because CMS platforms like WordPress are widely used, they are also targeted by hackers. You will need to ensure that your website is secure and regularly updated to prevent security breaches.
3. Dependence on third-party plugins: To add functionality to your website, you will likely need to use third-party plugins. These plugins may not always be reliable, and their quality may vary, which can impact the performance and security of your website.

4. Performance issues: WordPress sites can be slow to load, especially if they are not optimized properly. This can lead to a poor user experience and negatively impact search engine rankings.

5. Complexity: While WordPress is generally user-friendly, it can still be complex for non-technical users to manage. Learning how to use WordPress can take time and effort, which may not be feasible for all website owners.

6. References

Silkalns, A. (2023) *10 best bootstrap contact forms 2023*, *AdminLTE.IO*. Available at: <https://adminlte.io/blog/bootstrap-contact-forms/> (Accessed: April 13, 2023).

Country guide app | javascript API project | HTML, CSS & javascript (2022) *YouTube*. YouTube. Available at: <https://www.youtube.com/watch?v=QDCmQHO8F8Q&t=636s> (Accessed: April 13, 2023).

HTML snippets for Twitter bootstrap framework (no date) *Bootsnipp.com*. Available at: <https://bootsnipp.com/snippets/K0vRX> (Accessed: April 13, 2023).

Harsh (2022) *Country guide app using HTML, CSS & JavaScript*, *Code With Random*. Available at: <https://www.codewithrandom.com/2022/12/02/country-guide-app-usin> (Accessed: April 13, 2023).

Majestic skyscrapers in City · free stock photo - PEXELS (no date). Available at: <https://www.pexels.com/photo/majestic-skyscrapers-in-city-14295949/> (Accessed: April 13, 2023).