ANSWER SHEET (DO NOT write in right hand column)

MODULE CODE:	**COMP6251**		Student ID:	**31990401**
Question number:	2a	This is page number:	1	of my answers

HTML is a set of rules for configuring the appearance and content of a web page. In essence, it is not a programming language, but a language for describing the properties of the elements that make up a web page. Using special tags, instructions are given on how to manage the content of an HTML document and how to display it to the user in a web browser.

CSS (Cascading Style Sheets) is a technology for defining the display of data on a web page that comes and goes along with that of HTML which in turn determines the structure of a page. CSS determines the formatting of the content of an HTML page, such as the colours and position of the page elements. CSS can be saved to an external file with a .css extension and shared on more than one .html page, thus saving implementation time and effort. This practically means that a change in this file will update all the HTML pages to which it links.

Differences

- HTML is the standard markup language for creating web pages and web applications. CSS is a style sheet language used to describe the presentation of documents written in a markup language such as HTML. So, this explains the basic difference between HTML and CSS
- The Hypertext markup language is the full HTML format, while the Cascading Style Sheet is the full CSS format
- HTML consists of tags that surround the content, while CSS consists of selectors that are achieved by a declaration block
- CSS can be used in HTML files while HTML cannot be used in CSS files
- There is a difference between HTML and CSS based on their usage. HTML is used to build the structure of the website while CSS is used to make the website more visible

MODULE CODE:	COMP6251		Student ID:	31990401
Question number:	2b	This is page number:	2	of my answers

Semantic markup in HTML is a specific HTML tag that introduces meaning to a web page element and therefore to the whole web page.

Examples:

- <p>: indicated that the enclosed text is a paragraph
- <a>: indicated that the enclosed text is a hyperlink
- <b>: specifies bold text
- <button>: defines a clickable button
- : defines a division or a section in an HTML document

Why is it essential to use them?

1. The only way for search engines to understand which elements on the page are more important than others is through the markup. If the page name is within a <p> tag it is likely to get lost in other content and not be returned in search engine results. But if we put it within a <h1> tag the search engine will give it more importance. Thus, search engine optimization is the first reason to use semantic markup.
2. It is easier to edit the style of a document if it is marked up correctly. If all our section headings are <h2> tags, we only need to specify one CSS class and all section headings are styled. If in the future, we want to change our design entirely a well-marked up document will make the exercise simple.
3. Many other user agents won't understand the code, whether it be a mobile device, or a web app. HTML is a common language and if we take the time to understand it and write documents semantically, the code will stand the test of time. If you choose to write poorly formed HTML it is like a dialect and therefore some will understand the code very well, but others won't understand it at all.

Inline

We use inline CSS to style an HTML element by adding the style attribute to each HTML tag, without using selectors. Each HTML tag needs to be styled individually and therefore managing a website may become too hard. However, inline CSS in HTML can be useful if we don't have access to CSS files or need to apply styles for a single element only.

Pros

- We can easily and quickly insert CSS rules to an HTML page. Thus, we can test or preview the changes and performing quick fixes to a website
- We don't need to create and upload a separate document as in the external style

Cons

- Adding CSS rules to every HTML element is time-consuming and makes the HTML structure too complex
- Styling multiple elements can affect the page's size and therefore the download time

Internal

When we use internal CSS, we have to add the <style> tag in the <head> section of the HTML document. This is an effective method of styling a single page but using this style for multiple pages is time-consuming as we have to put CSS rules to every page of your website

Pros

- We can use class and ID selectors in this style sheet in order to style an element
- We only add the code within the same HTML file, and therefore we don't need to upload multiple files

Cons

- Adding the code to the HTML document can increase the page's size and therefore the loading time

External

By using external CSS, we can link a web pages to an external .css file. This CSS type is the more efficient method, especially for styling a big and complex website. By editing one .css file, we can change the entire site at once.

Pros

- CSS code is in a separate document. Thus, the HTML files will have a cleaner structure and are smaller
- We can use the same .css file for multiple pages

Cons

- Pages may not be rendered correctly until the external CSS is loaded
- Uploading or linking to multiple CSS files can increase the site's download time

React Context API is a component structure provided by the React framework, which is used in order to enable the sharing of specific forms of data across all levels of the application, and it is implemented to solve the problem of prop drilling.

We can create and manage a global application state by using the React context API. We can share global data such as data related to a specific user or a language preference or the mode (dark, light) of the app without having to pass down props along the React component tree.

The only thing we have to do is to define a context object where createContext returns a context provider component that allows consuming component to subscribe to changes. This object accepts a value props that will be made available to child components. Any children component of the myContext will be able to access the value provided. We can use the useContext(context) hook to access the value. The argument to useContext is the context object we created.

The useReducer hook is an alternative hook to useState to manage the state. The first argument is a callback function that receives the state and action as arguments and returns the new state: (state, action) => newState. The second argument is the initial state. If you want to set the initial state lazily, we can pass an init function as a third argument to useReducer. useReducer returns the state and a dispatch function as an array that we can destructure.

MODULE CODE:	COMP6251		Student ID:	31990401
Question number:	3b	This is page number:	5	of my answers

In react destructuring is a JavaScript feature that allows us to extract multiple pieces of data from an array or object and assign them to their own variables. When used, destructuring does not modify an object or array but rather copies the desired items from those data structures into variables. These new variables can be accessed later on in a React component.

State 1

Consider we are having the following car object (left image) which contains some attributes. By using destructuring we can access the car object's (right image) attributes by using:

```
const car = {
  brand: "Audi",
  color: "Red",
  type: "Big"
}
```

```
const { firstName, lastName, city } = person;
```

State 2

Destructing enables faster access to nested data. We no longer have to iterate over an Array or object multiple times to retrieve properties of its nested children. In this case, errMsg and errorHappened are now available to use via destructuring without having to extract this nested data with a loop (right image). Some React components have multiple ternary operators within its render function. Having single word destrucutured variables vastly improves a component's readability. This can be as simple as better describing loading and error states:

```
const {error: {errorMsg, errorHappened}, loading } = this.props;
    return (
      <section>
        {loading ? (
          <div className="loader">
            <ClipLoader color={'white'} loading={loading} />
          </div>
        ) : (
          <div className="results">
            {errorHappened ? (
              <h1>{errorMsg}</h1>
            ) : (
                <div>No Error </div>
          </div>)
    </section>
```

```
<section>
        {loading ? (
          <div className="loader">
            <ClipLoader color={'white'} loading={loading} />
          </div>
        ) : (
          <div className="results">
            {errorHappened ? (
              <h1>{errorMsg}</h1>
            ) : (
                <div>No Error </div>
          </div>)
</section>
```

This is also useful as there is both easy access to the error object and its nested properties errorMsg and errorHappened. It's simple but implementing it throughout React components enables a project to improve its maintainability and readability. Improving readability is more useful than one may initially think. Essentially, the developer is eliminating an additional helper to loop through nested data in React's props.

MODULE CODE:	COMP6251		Student ID:	31990401
Question number:	3a	This is page number:	6	of my answers

By using class-base components we can add more functionality to them but at the same time we should consider the following drawbacks:

1. When we use class-based components it is more difficult to read and test because they have not the same architecture as functional components. Instead, functional components are plain JavaScript functions without state or lifecycle-hooks
2. Class based components increase the code and the app becomes more complex for developers
3. It is difficult to separate container and presentational components. When using functional components instead we only need to think more about the component's state if we don't have access to setState() in our component

A SPA app reloads only the data necessary for the user. In the case of a multi-page app, the entire web page content is refreshed. When the user launches a SPA, the server loads the entire page. Later on, responding to the request, only the necessary data is transferred in the form of JSON files.

Single Page Application (SPA)

Pros:

1. The app performance is increased as all resources are loaded during one session, and then, when interacting with the page, only the necessary data is changed
2. SPA provide a more understandable linear experience and by using AJAX and JavaScript frameworks, allows building a more flexible and responsive interface
3. After the first request to the server, all the necessary local data is stored in the cache, and that provides users with the possibility to work in an offline mode

Cons:

1. Any web app runs in JavaScript, and the data is loaded without reloading the page. Thus, there are no separate URLs optimized for search engines, and search engines do not see the content
2. If the platform is complex, large, and poorly optimized, the users' browsers will take more time to load the content
3. Without JavaScript, we cannot fully use the complete functionality of a certain app. If users disable JS in their browser (modern browsers like Brave), they will not be able to use the app to its fullest

Multi Page Application (MPA)

Pros:

1. The app has multiple pages, and each of them can be optimized for a specific group of requests
2. This architecture type allows creating as many new pages as possible for each product or service and implementing any changes in them
3. MPA development requires a smaller technology stack, and besides, a wide range of CMS are available.

Cons:

1. Performance is low in case of many requests and the necessity to reload many pages (especially true for projects with high website traffic, many pages, and multiple functions)
2. The components of a web app are deeply integrated, and that is why it can take longer to develop and test them
3. It can be a daunting task to provide technical support to websites with a lot of pages (security issues)

Before starting to work with a framework for a project we need to make sure that the framework we are going to use will be useful and will meet all our requirements. The five criteria we should consider before we chose a web framework are the following:

1. The first thing to take into account when choosing a framework is the direct and indirect costs. If the application is large (like a web app for a big company), costs can be increased. Many modules are payable, and some open-source frameworks are developed by companies that sell support services, which can help mitigate risk
2. Every application has to access a database in order to read and store data. While choosing a framework we should choose one which lets the application become database agnostic. Another thing we will need to think about is the framework's ORM capabilities. ORM will let us express data as an object and see how it relates to other objects
3. Documentation of the framework is very important and not all frameworks come with good documentation. Therefore, before selecting any framework, we need to check out the attendant documentation and make sure that it is reasonably complete
4. By using a big framework, lots of developers sifting through the code and putting it through its paces on a daily basis. When a bug is found need to be fixed as soon as possible. Thus, we need to choose a framework which is secure. The framework should be updated often, and fixes the bugs people come across as soon as possible
5. While choosing a framework, we should choose one that has the smallest possible learning curve. If we don't know the language the framework is written in, we have to make a note of including the language itself to the learning curve

A Serverless Application Model (SAM) is a cloud native platform which is used for the development of short running, stateless and event-driven applications. Such a model is scaled up and down instantly. It is also automatically and charges for actual usage at a millisecond granularity. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. Furthermore, It can be used to define the application we want to build and model it using.

Main motivation to use it

Serverless computing offers a number of advantages over traditional cloud-based or server-centric infrastructure. For many developers, serverless architectures offer greater scalability, more flexibility, and quicker time to release, all at a reduced cost. With serverless architectures, developers do not need to worry about purchasing, provisioning, and managing backend servers. However, serverless computing is not a magic bullet for all web application developers.

Pros:

1. No server management is necessary: Developers never have to deal with the servers because they are managed by the vendor. This can reduce the investment necessary in DevOps, which lowers expenses, and it also frees up developers to create and expand their applications without being constrained by server capacity
2. Developers are only charged for the server space they use, reducing cost. Code only runs when backend functions are needed by the serverless application, and the code automatically scales up as needed.

Cons:

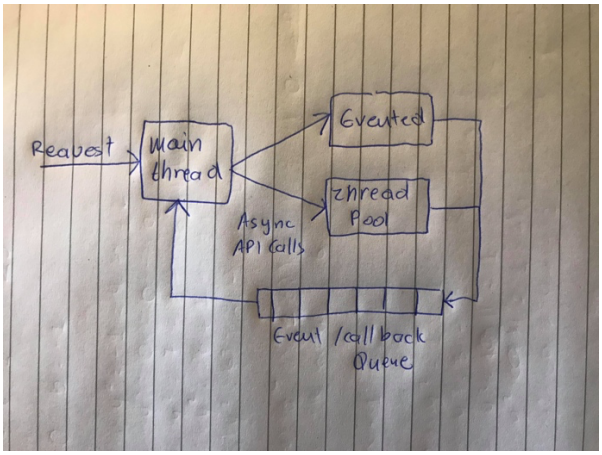
1. Testing and debugging become more challenging: It is difficult to replicate the serverless environment in order to see how code will actually perform once deployed. Debugging is more complicated because developers do not have visibility into backend processes, and because the application is broken up into separate, smaller functions
2. Serverless architectures are not built for long-running processes: This limits the kinds of applications that can cost-effectively run in a serverless architecture. Because serverless providers charge for the amount of time code is running, it may cost more to run an application with long-running processes in a serverless infrastructure compared to a traditional one

MODULE CODE:	COMP6251		Student ID:	31990401
Question number:	4b	This is page number:	10	of my answers

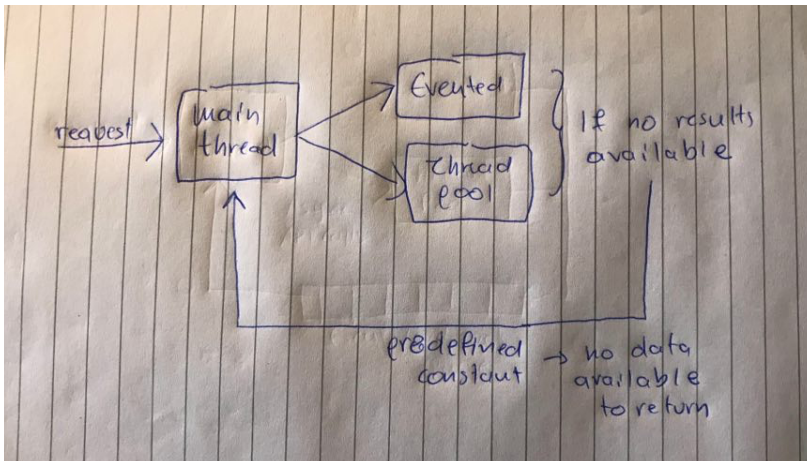
Multi-threaded request-response" architecture is an event loop that is much slower and unable to handle multiple concurrent threads at a time. The platform does not follow a similar request-response multi-threaded stateless model; instead, it goes by a simplified single-threaded event loop model. As per Node.js developers, a specific library provides this mechanism known as an event loop. This Node.js processing model is majorly based on JavaScript event-based model along with the callback mechanism.

In Node.js when an HTTP request hits the server, node calls the request handler function with a few handy objects for dealing with the transaction, request and response. In order to actually serve requests, the listen method needs to be called on the server object.

The following diagram shows how the request handling works:



The following diagram shows how non-blocking I/O works:



Middleware

Express middleware containing functions that execute during the request-response cycle and have access to both the request object and the response object. Middleware is executed during the time that a server receives a request and when it sends a response. Express middleware includes application-level, router-level, and error handling functionality and can be built-in or from a third party. Since Express.js has limited functionality of its own, an Express app is largely comprised of multiple middleware function calls.

Routing

Express routing refers to how a server-side application responds to a client request to a particular endpoint. This endpoint consists of a URI and an HTTP method (GET, POST, PUT, DELETE, etc). Routes can be either web pages or REST API endpoints. Routers are helpful in separating concerns such as different endpoints and keep relevant portions of the source code together. They help in building maintainable code. All routes are defined before the function call of app.listen(). In a typical Express application, app.listen() will be last function to execute.

MODULE CODE:	COMP6251		Student ID:	31990401
Question number:	1c	This is page number:	12	of my answers

The main reasons SPA applications use the MVVM architecture are the following:

- We don’t need to mess with VMs or any other complicated local configuration
- We can swap or add new UI engineers if we have to, without any ramp up time for server-side setup
- The same server as a back end, independently on a static, or locally
- Back end can run anywhere. Even better, we can simulate one with static JSON files
- Once we download all the assets (HTML/CSS/JS/images), our only HTTP traffic is data
- With this architecture, API server is already in place if we want to build a mobile app