1	**Compare a server-rendered application (e.g. PHP) - having its presentation layer in server-side logic - with a Rich-web approach (having is presentation layer as client-side logic). Mention two advantages of each approach** Server-side rendering advantages: • Search Engine Optimisation (SEO) – historically, search engine crawlers see websites that are server-side. • Performance – server-side rendering can be very fast under optimal conditions, i.e. fast data center or a user with a fast network connection. Client-side rendering advantages: • No full page reloading – Client-side rendering avoids making requests for a full page, a single page is changed my calling for the necessary components needed. • Enforced separation of concerns – With client-side rendering, separation of concerns is programmatically enforced. Call must be made to various services and the helps to adopt a service-oriented mindset on development.

2	**Explain how the box model works in CSS using example code in your answer**
	
	Each element in the CSS box model is represented as rectangular box, containing the box's padding, margin, and border. As a browser renders a web page, it figures the styles that need to be applied to the content of each box.
	
	• Make all <article> tags have a top relative padding
	`article { padding-top: 2%; }`
	• Make a border around all <p> with a width of 1px and color grey
	`p { border: 1px solid grey; }`
	• Build a margin around all of the <div> tags with class="box"
	`div.box { margin: 10px 5px 10px 5x; }`

3	**Suppose your web App has to run on an older browser that does not have built-in support for the fetch() API. Sketch an outline of how the fetch() API could be poly-filled in Javascript based on the `XMLHttpRequest()` API**
	
	The global fetch function is an easier way to make web requests and handle responses than using an XMLHttpRequest. This polyfill is written as closely as possible to the standard Fetch specification.
	
	`fetch('/users.html')`
	`  .then(function(response) {`
	`    return response.text()`
	`  }).then(function(body) {`
	`    document.body.innerHTML = body`
	`  })`
	
	Reference: npmjs.com - install fetch-polyfill

4	**CSS allows the reuse of code for styling DOM elements. Javascript functions can all be used for element styling and support code reuse. Compare the two code sharing approaches** • CSS style sheets can be loaded before the is fully loaded, allowing them to apply instantly before the DOM objects are displayed. • JavaScript styling is often applied after the DOM objects are displayed. • Presentation styling is often processed by CSS. • Behavioral styling is controlled by CSS
5	**Describe the Web Component abstraction and explain its perceived advantages as a separation-of-concerns mechanism over other single responsibility approaches** Web components are a set of features that allow for the creation of reusable components in a web file and application. The advantages as a separation-of-concerns mechanism over other single responsibility approaches: • Component composition is a powerful tool for managing complexity in web app development by eliminating duplication and singularity of the individual components. • Horizontal separation – an application is split into logical layers of functionality that ultimately fulfill the same role. (Presentation, business and resources access). This allows for a faster and more flexible application.
6	**Describe how a React JS stateful class component is created and how its instantiated life-cycle can be managed** • A React JS stateful class component is declared as a JavaScript class using the ES6 class keyword. To become a React component this class must extend the built-in ReactJS component class. • Its instantiated life-cycle can be managed by tapping into a series of hooks that were provided for each phase of the life cycle 1. Birth / mounting 2. Growth / update 3. Death / unmount

7	**In asynchronous programming, we have three approaches to handling data which may or may not arrive at some point in the future, namely callbacks, promises and streams. Describe each of these approaches. Are there any significant drawbacks of each in your opinion?** • **Callbacks** can be used whenever the code needs to make a request which will take a relatively long time. One of the problems with callbacks – if a further async request needs to be made on foot of a response from a previous one, then the second async request must be made with the first request's callback logic. This leads to a pattern of nested callbacks. • **Promises** are values which are a contract for delivering a value in the future i.e. the time that the response arrives and the value will be a success value. Drawback – deeply nested call chains are inefficient and difficult to reason about. • **Streams** behave like time-ordered lists of data similar to arrays, they implement the observer pattern where data is realised using the subscribe operation. Drawback – currently, support for streams in the browser only available through third part libraries.
8	**What are your reflections on how the Rich Web technologies stack up as a development environment based on your experience in this module or elsewhere?** Based on my experience in this module and elsewhere (3rd year internship), I believe that rich web technologies stack up well as a development environment because their modularity by componentising various functionalities.
9	**You will have encountered many bugs and issues during your code development in this module's labs. Describe your approach to debugging your javascript application, mentioning any tools or techniques you have used** My approach for debugging JavaScript was primary through the Chrome browser built-in debugger by pressing the F12 key on my keyboard. Generally, when I write a JavaScript function or line of code, I'd display the process or error via the console log. However, I've also used my preferred text editor (Visual studio code) for debugging JavaScript.
10	**Complete the Q6A survey for the module through webcourses** Completed