**1. What is the web component abstraction? Explain the relationship between web components and ordinary native DOM elements. How does a web component abstraction facilitate the separation-of-concerns principle in web development?**

The web component abstraction consists of a user interface that is associated with two important aspects, the presentation and the behaviour. The component presentation can be visual such as an image or video, or it can be a text input field etc.

The component behaviour is a set of actions that can be taken on behalf of the component based on user feedback. An example of this would be the response that a button component defines when pressed by the user.

Component abstraction offers the opportunity and capability to comply with good design principles. In terms of the separation-of-concerns, the abstraction allows the ability to incapsulate and hide implementation details of the component’s presentation and behaviour at the individual component level.

**2. What is JavaScript Object Notation (JSON)? What is JSON’s relationship with standard and native JavaScript data structures? In your answer, mention the key differences. Show in code, how to convert between JSON and native JavaScript data structures.**

JavaScript Object Notation is a minimal human-readable format for structuring data. It is mostly used to transmit data between server and web applications as an alternative to XML. It is a developer friendly format when combined with whitespace indentation but also compresses well when being deployed in production applications.

In terms of JSON’s relationship with standard data structures, the Javascript Object Literal syntax gave rise to the idea for a different serialisation format from XML.

Whereas JSON.parse can read JSON strings into native JS data types.

**E.g.**

JSON.parse(‘ { } ’);

JSON.parse(‘true’ );

JSON.parse(‘”foo”’ );

JSON.parse(‘[1, 5, “false”]’ );

JSON.parse(‘null’ );

JSON.parse(‘{“x”:5, “y”:6}’);

**3. In functional programming, what does the term functor mean? Can you give an example in Javascript and briefly show how it being used in code?**

A functor is a data object that can hold elements of any data type and which implements the map operation. The most common use of functors in Javascript are Arrays. Using the map() function, you are able to take another function as an argument and call that function for each element of the functor. This would result in a new functor.

**For example:**

[‘123’, ‘456’. ‘789’]

.map(s => parseInt(s))

.map(n => n/10)

**4. Modern rich web development often requires the use of build-time tools for packaging code and assets into an application bundle. Briefly describe what each of the following tools does and what problem each attempts to solve:**

* *A Bundler like Webpack*
* *A Loader like Babel*
* *An Asset Minimiser like Uglify*

1. Webpack Bundler

Webpack is a build environment that bundles everything (Javascript, CSS, Images etc.) into a dependency graph. This maps every module your project needs and generates one or more bundles.

*Problems it solves:*

1. General Maintenance – Keeps everything in the one file

2. Babel

Babel is a transpiler that takes modern Javascript and coverts it to the older version so that the code can be executed in all browsers including the ones that don’t support modern Javascript standards.

*Problems it solves:*

1. Allows the user to write better code by using modern Javascript languages. E.g. React

2. Better code reuse and better compile-time checking

3. Uglify

Uglify is a minimiser written in Javascript. It parses input files and applies any compression options. It includes an assortment of tools to help the user work with Javascript code. It contains tools such as a parser, code generator, compressor etc.

*Problems it solves:*

1. Allows the user to download fewer bytes therefore enabling the page to load faster or use less battery.

**5. Briefly describe the process that takes place when a page is loaded into a browser. In your answer, deal with each of asset types, namely HTML, CSS and JS. When is it safe to start executing JS on a page? Why? How do you control when your JS gets executed?**

The three primary technologies used when building a webpage are HTML, CSS and JS. The first HTTP request will start loading the initial assets from the server into the browser. In this case, assets are HTML and CSS which build the content for the webpage and present it in a particular style. The browser parses each of the files as they are loaded and runs any commands that may be contained. JS is then used to enhance the user experience by adding functional programming such as key presses, timers etc.

You can control where and when your JS code gets executed by using <script> tags, these are executed synchronously with the browser’s processing.

**6. This module has considered three different kinds of asynchronous programming mechanisms, namely callbacks, promises and streams. Mention one advantage and one disadvantage of each type***.*

1. Callbacks

*Adv –* Can be used when the code needs to make a request that will take a long time.

*Disadv –* Can lead to a pattern of nested callbacks. This is when a second asynchronous request must be made in response to a previous one, with the first requests callback logic.

2. Promises

*Adv –* Can provide the same functionality as nested callbacks but is easier to reason about and debug.

*Disadv –* It increases the CPU usage substantially.

3. Streams

*Adv –* Different architectures such as mouse clicks, keyboard inputs, timers can all be processed within the same logical structure using the same semantics.

*Disadv –* There is no built-in support for streams in Javascript in the browser.

***7. The JS built-in try-catch statement does not help to deal with asynchronous errors. Why is this the case? What alternatives exists for dealing with asynchronous errors?***

Try-catch exceptions do not work with asynchronous error handling as exceptions are only a synchronous mechanism. There are other alternatives for dealing with these errors such as Promises, these provide a clean, flexible way to chain multiple async operations together without having to use callbacks.

***8. Describe, JSX using code fragments to illustrate your answer.***

The DOM mark-up language in React is JSX. It’s Javascript but supports a mark-up like syntax for convenience.

**E.g.1**

<div>

<img src=”avatar.png” className=”profile” />

<h3> { [firstName, lastName].join(‘ ‘) } </h3>

</div>

**E.g.2**

React.createElement(“div”, null,

React.createElement(“img”,{ src: “avatar.png”, className: “profile”}),

React.createElement(“h3”, null, [firstName, lastName].join(“ “))

);

In the above two examples, one is a JSX fragment and the other is a <div> container containing an <img> and a <h3> element. The curly brackets is the Javascript expression expansion syntax where arbiratry Javascript code can be injected into the JSX templates.

***9. Describe the Elm language and architecture and how it is attempting to solve the problems associated with W3C web technologies.***

The Elm language was developed by Evan Czaplicki out of frustration with the existing web ecosystem. It is based on Meta Language style syntax, being cross-platform, it is purely functional and is also quite small, having only a few keywords and operators.

There are three main elements of the architecture, the model data structure, the view function and the update function.

Model 🡪 is a data structure which holds the application state.

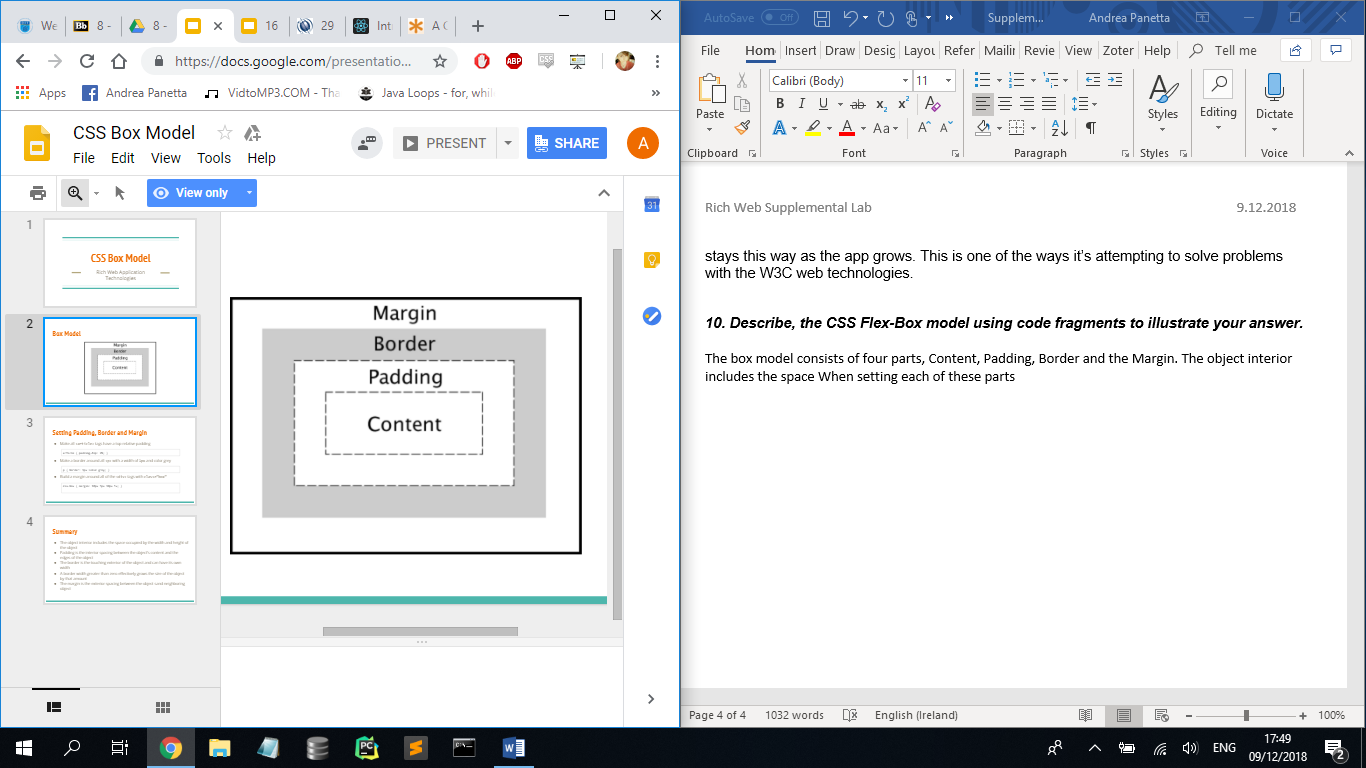
View 🡪 responsible for writing the DOM contents from the model contents.

Update 🡪 handles the asynchronous events and then updates the model.

One of the main advantages of Elm is its Friendly Compiler. Having no runtime errors, no null, no “undefined is not a function”, makes coding a lot less frustrating. Instead it has friendly error messages that help add features quicker and has well architected code that stays this way as the app grows. This is one of the ways it’s attempting to solve problems with the W3C web technologies.

***10. Describe, the CSS Flex-Box model using code fragments to illustrate your answer.***

The box model consists of four parts, Content, Padding, Border and the Margin. The object interior includes the space occupied by the width and height of the object. The padding is the interior spacing between the object’s content and the edge of the object. The border touches the exterior of the object and the margin is the exterior spacing between the object and the neighbouring object.



To set each of these parts, the following CSS code is what to use:

1. Make all <article> tags have a top relative padding:

Article { padding-top: 2%; }

2. Make a border around all <p>

p { border: 1px solid grey; }

3. Build a margin around all the <div> tags with class=”box”

div.box { margin: 10px 5px 10px 5px; }