Question 1.
Explain what is meant by Rich Web Application Development. Distinguish it from traditional web development?

Ans:
In Rich Web Applications a web application where the client has most, if not all of the responsibility for implementing the presentation layer logic. Rich web applications see the significa use of javascript programming in the client browser often employing a lean data transfer interface with the server. Web application is a client server application but only if it compromises a certain set of standard based technologies.

The primary purpose of a website is to convey information. A web site content can be statically constructed in advance and served to the users browsers. The primary technologies used for web development are HTML and CSS to build the content and present it in a particular style. Javascript may be used to enhance the user experience but, increasingly, CSS is capable of achieving most of this.

Compared to web apps, web apps is more akin to a desktop application, they are normally designed to fulfil some data processing function involving a server.
Eg: an online booking engine or email or and social messaging application or a content management system. HTML and CSS are often used for building and styling the content and data views, but this is slowly disappearing and other trends are been used like javascript.Javascript used to process user inout, validate the input and fetch data from the server or save data back to the server.

Question 2.
What is the Document Object Model? Explain, giving a couple of examples, how to interact with the DOM in Javascript?

Ans:
The DOM is the browses internal representation of the page content., The DOM is a tree structure representing the hierarchal relationship between enclosing elements and their enclosed elements. Javascript can be used with or conjunction with HTML and CSS or a replacement in Web application construction/
In javascript, you can read and update property values or call methods on the nodes which may also mutate node's state. Elements can can be added or removed from the DOM.

To get access to the DOM, we reference to one or more nodes by one of the native accessor methods.

Example 1:
```
let el = document.getElementById(“myPara”); // => HTMLParagraphElement
let content = el.innerHTML;                 // => read
el.innerHTML = “new content”;               // => update
```

Example 2: Create a text node.
```
let text = document.createTextNode("Hello, world!");
el.appendChild(text);
```

Example 3: Create a new DOM element
```
let el = document.createElement("p");
let text = document.createTextNode("Hello, world!");
el.appendChild(text);

let where = document.getElementById("mySection");
```

```
document.body.insertBefore(el, where);
```

Example 4: Remove a DOM element.

```
let el = document.getElementById("gone");
if (el.parentNode) {
  el.parentNode.removeChild(el);
}
// or
let prev = el.previousElementSibling; // => Left neighbour
if (prev) {
  prev.remove()
}
```

Question 3.
What does it mean for a data structure to be described as a functor? Give a code example in Javascript in your explanation
Ans:

Question 4.
What is the de facto standard for data serialisation in the web app world? Give an example

In the rich web style of web apps. client requests and server responses are serialised into some data forma which can be translated by the client and server in some language independent way. The main requirement of a serialisation format is support for common data types and a hierarchical structural representation.

XML was the first attempt to create a language independent data serialisation format that was not HTML but shared many of the ideas of HTML.

Another attempt was Javascript object literal syntax . JS was the principal consumer of the wire format on the client.
Object literlas supports hierarchical structurele representation.
Easier tor read and generate.

JSON documents have become the de facto standard serialisation format in the web app ecosystem.
JSON is easy to read and parse and combines most of the advantages of XML without the weight.
JSON is a developer friendly format when combined with white spaces indentation but also compresses well when being deployed in production applications.

Example 1:

```
let event = {
  title: 'Live Jazz Music',
  starts_on: '2016-09-28 20:00',
  images: {
     avatar_path: 'https://s3.amazon.com/23daff3/avatar.png'
  },
  owner_id: 435643,
  address: '2345 Avery Ave, Paris, France'
};
```

Question 5.

Describe how the flexbox model works in CSS?

Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.

For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

Flexbox consists of flex containers and flex items.

A flex container is declared by setting the display properly of an elements into either flex or incline fee. Inside a flex container there is one or more flex items,

Everything outside a flex container and inside a flex item is rendered as usual. Flexbox defines how flex items are laid out inside a flex container.

Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

The following example shows three flex items. They are positioned by default: along the horizontal flex line, from left to right:

Example 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
    display: -webkit-flex;
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}
</style>
</head>
<body>

<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>

</body>
</html>
```

Question 6:
Explain how you can make a network request to a server-side resource in a web app using Javascript

The AJAX facility allows the programmer to initiate explicit network requests of some HTTP type to a specific URL. The asynchronous requests requires a callback function to be separated to hand the response which will arrive in the future. The basic native DOM API mechanism uses the XMLHttpRequest object.

```
const xhr = new XMLHttpRequest();
xhr.onreadystatechange = () => {
  if (xhr.readyState === XMLHttpRequest.DONE) {
    // The response is received
  }
};
xhr.open("GET", "http://example.com/some/resource/endpoint");
xhr.send();
```

When the request completes, the XMLHttpRequest object properties can be queried to access the response in the onreadystatechange() handler.

```
let response;
const xhr = new XMLHttpRequest();
xhr.onreadystatechange = () => {
  if (xhr.readyState === XMLHttpRequest.DONE) {
    if (xhr.status === 200) {
      response = JSON.parse(xhr.responseText);
    } else {
      console.log('Error ' + xhr.statusText);
    }
  }
}
```

Question 7:
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 approaches

CSS styling allows a programmer to style headers, containers, classes or paragraphs with very little code. A programmer can reuse the CSS code to style a certain area of a web page by calling the page or the id of an element to be styled.

Comparing this to a javascript function that can use DRY. DRY is don't repeat yourself code. a function can be created. an example of a function would be to pull information from GitHub by asking a URL. To get the repositories related to that user in Github, the url would be amended and the URL would be passed to the same function that will return an JSON object. The first time the function runs, an object will be returned about information regarding a user. The second time the function runs, the JSON object returned will be information relating to the user repositories.

Question 8:
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?

Callback
A call back is a function that you provide as an argument to the I/O request function which implements the logic to be executed at the point hen the request has been fulfilled

Example 1:
```
const url = "https://s3.amazon.com/23123fd123/95fdae";
const handler = (data) => {
  // do something with data ...
};
getSomeData(url, handler);  // named callback


// inlined callback
getSomeData(url, data => {
  // do something with data ...
});
```

The call back style can be used whenever the code need s to make a request which will take a relatively long time. examples would include reading data from a network or a disk. As soon as the request is made, it is queued by the browser and the callback function is stored for subsequent executionMeanwhile control is returned back to the executing context and the program proceeds During this time, the app may handle user input events or make further async requests
When the original request is fulfilled, the callback is called.

Promise
A promise is a value which is a contract for delivering a value in the future. The future is the time The future is the time that the response arrives and the value will be a success or an error value. A promise can also be composed or passed as arguments to functions. Promises are resolved or rejected when the future value arrives or an error occurs. Promises can also be chained to provide the same functionality s nested callbacks but which is easier to reason about and debug.

Stream
Stream are an abstraction used to model asynchronous data sources. A stream is a powerful technique when processing data when you either don't now your potential size and/or you don't know when it will arrive into your application. Examples of stream include video data and log files. Processing of stream generally must be done in time separated chances in sequence or concurrently, depending on the application.
Streams behave like time ordered lists of data similar to arrays. Streams implement the observer pattern where data is realised using the subscribe function. Once a stream is created over some future data, the stream data can operated on and traditional into new streams.