

2019-20

Monday 9 January 2023 19:40

- 1 (a) Define the term rich web application. Briefly describe the typical technologies used and their roles in the construction of a rich web app.

(8 Marks)

A rich web application is a web application that has many of the same features and capabilities as a desktop application. This can include things like offline functionality, push notifications, and the ability to access device hardware like the camera or microphone.

Typical technologies used in the construction of a rich web application include:

- **HTML, CSS, and JavaScript:** These are the core technologies used to build the user interface and provide basic functionality.
- **Web frameworks:** These are pre-built libraries and frameworks that provide additional functionality and structure to web applications. Examples include React, Angular, and Vue.js.
- **APIs:** These allow the web application to access external data or functionality, such as the ability to access the user's location or send push notifications.
- **Server-side languages and frameworks:** These are used to build the backend of the web application and handle tasks like storing and retrieving data, authenticating users, and processing requests from the frontend. Examples include Python, Java, and Ruby on Rails.
- **Databases:** These are used to store and retrieve data for the web application. Examples include MySQL, MongoDB, and PostgreSQL.
- **Hosting:** This is where the web application is deployed and made available to users over the internet. Examples include cloud hosting platforms like Amazon Web Services and Google Cloud.

- (b) Describe what pseudo selectors are in CSS and what problem they solve. Consider the following code fragment:


```
<div id="pseudo-style">
  <p>My first paragraph</p>
  <p>My second paragraph</p>
</div>
<div>
  <p>My third paragraph</p>
  ..
  <p>My nth.. paragraph</p>
</div>
```

Make the first paragraph text colour red, the second paragraph text colour 'blue'. All other text within all other paragraphs should be black. Outline the css for this problem.

(8 Marks)

Pseudo-selectors in CSS are used to style elements that have a special state or relationship to other elements. For example, the `:hover` pseudo-selector can be used to apply styles to an element when the user hovers their mouse over it.

To solve the problem you described, you could use the `:first-of-type` and `:nth-of-type(2)` pseudo-selectors to select the first and second paragraphs within the first `div` element, like this:

 Copy code

```
#pseudo-style p:first-of-type {
  color: red;
}

#pseudo-style p:nth-of-type(2) {
  color: blue;
}

p {
  color: black;
}
```

This would apply the styles to the first and second paragraphs within the element with the `id` of `'pseudo-style'`, while the final `p` rule would apply the default black color to all other paragraphs.


- (c) What are higher order functions in Javascript? What is meant by a function closure? Give an example of declaring and using a closure.

(8 Marks)

Higher order functions are functions in JavaScript that take other functions as arguments or return them as output. These functions allow you to abstract common patterns of function use and reuse them in different contexts.

A function closure is a function that has access to the variables in its outer scope, even after the outer function has returned. This allows the inner function to "remember" the context in which it was created and continue to reference those variables.

Here is an example of declaring and using a closure in JavaScript:

 Copy code

```
function outerFunction() {
  let x = 10;

  return function innerFunction() {
    console.log(x);
  }
}

let closure = outerFunction();
closure(); // Output: 10
```

In this example, the `outerFunction` declares a variable `x` and returns the `innerFunction`. The `innerFunction` has access to the variable `x` even though the `outerFunction` has already returned. When the `closure` is called, it logs the value of `x` to the console.

- (d) What is a stateful function component in React JS? Illustrate your answer with an example declaration and component usage.

(8 Marks)

In React, a stateful function component is a function component that has a state and the ability to manage and update it. Stateful function components are created using the `'useState'` hook, which allows you to add state to function components.

Here is an example of a stateful function component in React:

```
import { useState } from 'react';

function Example() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>Click me</button>
    </div>
  );
}
```

In this example, the `'Example'` component uses the `'useState'` hook to add state to the component. The state variable is `'count'`, and the `'setCount'` function is used to update the value of `'count'`. The component renders a button and a paragraph, and the paragraph displays the current value of `'count'`. When the button is clicked, the `'setCount'` function is called with the new value of `'count'`, updating the state and re-rendering the component.

To use the `'Example'` component, you can include it in a parent component like this:

```
<Example />
```

This would render the `'Example'` component and display the button and paragraph. When the button is clicked, the `'count'` state variable would be updated and the component would re-render.

- (e) Why use Elm? What problems is it trying to address with web development? What is meant by type inference?

(8 Marks)

Elm is a functional programming language that is often used for building web applications. It is designed to make it easier to build reliable and maintainable web applications by providing features like static typing and type inference.

One of the main problems that Elm is trying to address in web development is the issue of runtime errors. In many programming languages, it is possible for an application to encounter an error at runtime, which can be difficult to debug and can lead to poor user experiences. Elm aims to eliminate runtime errors by using a static type system, which catches errors at compile time rather than runtime.

Type inference is a feature of some programming languages that allows the compiler to automatically determine the data types of variables and expressions based on how they are used. This can save developers time and effort by eliminating the need to explicitly specify the data types of variables and expressions.

In Elm, type inference is used to help catch errors and ensure that the code is correct. For example, if a developer tries to use a function that expects a number as an argument, but passes in a string instead, the compiler will be able to detect this error and alert the developer. This can help prevent issues like runtime errors and improve the overall reliability of the application.

- 2 (a) With the aid of a diagram and example code, describe the Cascading Style Sheets (CSS) Box Model and show how it can be used to space DOM elements.

(12 Marks)

The CSS Box Model is a model used to determine the layout and dimensions of elements on a web page. It consists of four main parts: the content, padding, border, and margin.

The content is the actual content of the element, such as text or an image. The padding is the space around the content, and the border is a line that goes around the padding. The margin is the space outside of the border.

Here is an example of how you can use the CSS Box Model to space DOM elements:

```
div {  
  margin: 10px;  
}  
  
p {  
  padding: 5px;  
  border: 1px solid black;  
}
```

In this example, the `div` element has a margin of 10 pixels, and the `p` element has a padding of 5 pixels and a 1 pixel solid black border. This would create space between the `div` and other elements on the page, and add padding and a border around the `p` element.

You can also specify different values for the top, right, bottom, and left sides of the box model using the `margin-top`, `margin-right`, `margin-bottom`, and `margin-left` properties, or use the shorthand `margin: top right bottom left;` syntax. Similar properties and shorthand syntax are available for the padding and border as well.

- (b) An asset bundler (like Webpack) allows CSS and JavaScript to be shared across a large project using the ES6 export and import semantics. Explain what an asset bundler does and what problems it solves. Also, describe how an asset loader can work with third-party loaders (like Babel) to allow access to future technologies beyond the currently-supported W3C and standards and vendor features.

(18 Marks)

An asset bundler is a tool that allows you to bundle your CSS and JavaScript assets into a single file or a few files, which can make it easier to manage and improve the performance of your web application.

One of the main problems that asset bundlers solve is the issue of multiple requests to the server for separate CSS and JavaScript files. When a user visits a web page, their browser sends a request to the server for each of the resources (CSS, JavaScript, images, etc.) needed to render the page. If a page has many separate CSS and JavaScript files, this can result in a large number of requests, which can slow down the page load time and impact the user experience. Asset bundlers allow you to bundle all of your CSS and JavaScript into a single file, reducing the number of requests and improving performance.

Asset bundlers can also work with third-party loaders, like Babel, to allow access to future technologies beyond the currently-supported W3C and standards and vendor features. Babel is a tool that allows you to use new JavaScript language features that may not be supported in all browsers yet. By using Babel with an asset bundler, you can write code using these new features and have it transpiled (converted) into a version that is compatible with all browsers. This allows you to use the latest and greatest technologies in your web application without having to worry about browser compatibility.

3 (a) What is the Document Object Model (DOM)? Show, using example code of your choice, how a developer can interact with DOM API to capture an input event using the following approaches

- A Callback Function
- An Observable Stream

(12 Marks)

The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the structure of a document as a tree of objects, with each object representing a part of the document (such as an element, an attribute, or a piece of text).

Here is an example of how you can use a callback function to capture an input event in the DOM:

```
// Get the input element
const input = document.querySelector('input');

// Set up the event handler
input.addEventListener('input', function() {
  console.log('Input event captured');
});
```

In this example, we first use the `querySelector` method to get a reference to the `input` element. Then, we use the `addEventListener` method to attach an event handler to the `input` event. When the user types something into the input field, the event handler function will be called, and the message "Input event captured" will be logged to the console.

Here is an example of how you can use an observable stream to capture an input event in the DOM:

```
import { fromEvent } from 'rxjs';

// Get the input element
const input = document.querySelector('input');

// Create the observable stream
const input$ = fromEvent(input, 'input');

// Subscribe to the stream
input$.subscribe(() => {
  console.log('Input event captured');
});
```


In this example, we use the `fromEvent` function from the RxJS library to create an observable stream for the `input` event on the input element. We then use the `subscribe` method to attach an event handler to the stream. When the user types something into the input field, the event handler function will be called, and the message "Input event captured" will be logged to the console.

(b) Describe the Promise API in Javascript, briefly illustrating, using code, how a function can return a promise and how a consumer uses a returned promise object. Describe, with an example, how promises can help to improve network throughput between a rich web application and the server

(18 Marks)

The Promise API in JavaScript is a way to handle asynchronous operations in a synchronous-like style. A promise represents the result of an asynchronous operation, which can either be a value (fulfilled promise) or an error (rejected promise).


Here is an example of how a function can return a promise in JavaScript:

 Copy code

```
function getData() {  
  return new Promise((resolve, reject) => {  
    setTimeout(() => {  
      resolve('Some data');  
    }, 1000);  
  });  
}
```

In this example, the `getData` function returns a new promise that will resolve with the value "Some data" after a 1 second delay.

Here is an example of how a consumer can use a returned promise object:


 Copy code

```
getData().then(data => {  
  console.log(data); // Output: "Some data"  
}).catch(error => {  
  console.error(error);  
});
```

In this example, we call the `getData` function and use the `then` method to specify a callback function that will be called when the promise is resolved. We also use the `catch` method to specify a callback function that will be called if the promise is rejected.

Promises can help to improve network throughput between a rich web application and the server by allowing the application to send multiple requests to the server in parallel, rather than waiting for one request to complete before sending the next. This can result in faster overall performance, as the application can take advantage of available network resources more efficiently.

For example, consider a rich web application that needs to retrieve data from multiple API endpoints. Without promises, the application might have to send the requests sequentially, waiting for each request to complete before sending the next:

 Copy code

```
fetch('/api/endpoint1')  
  .then(response => response.json())  
  .then(data => {  
    // Do something with the data  
    return fetch('/api/endpoint2');  
  })  
  .then(response => response.json())  
  .then(data => {  
    // Do something with the data  
  });
```

With promises, the application can send the requests in parallel, improving network throughput:

```
Promise.all([
  fetch('/api/endpoint1'),
  fetch('/api/endpoint2'),
]).then(([response1, response2]) => {
  return Promise.all([response1.json(), response2.json()]);
}).then(([data1, data2]) => {
  // Do something with the data
});
```

In this example, the **'Promise.all'** method is used to send the requests in parallel, and the responses are processed once they are all available. This can result in faster overall performance, as the requests are not blocked by each other.

- 4 (a) Describe in detail JSX (from React JS) using code fragments to illustrate your answer. Also, explain in detail the Virtual DOM in React JS.

(12 Marks)

JSX is a syntax extension for JavaScript that allows you to write HTML-like code in your JavaScript code. It is often used in React to define the structure of a user interface.

Here is an example of JSX:

```
const element = (
  <div>
    <h1>Hello, world!</h1>
    <p>This is an example of JSX.</p>
  </div>
);
```

In this example, the **'element'** variable contains a JSX expression that defines a **'div'** element with a **'h1'** and a **'p'** element inside.


JSX is transformed into regular JavaScript by a JSX transformer, such as Babel, at build time. The resulting code might look something like this:

```
const element = React.createElement(
  "div",
  null,
  React.createElement("h1", null, "Hello, world!"),
  React.createElement("p", null, "This is an example of JSX.")
);
```

The Virtual DOM (VDOM) is a virtual representation of the actual DOM in a React application. It is a lightweight in-memory representation of the DOM that allows React to make efficient updates to the actual DOM.

When a React component renders, it creates a VDOM tree that represents the structure of the UI. React then compares the current VDOM tree to the previous VDOM tree (if there was one), and calculates the minimum number of changes needed to update the actual DOM to match the new VDOM tree. This process is known as reconciliation.

For example, consider a simple React component that displays a list of items:

 Copy code

```
function List(props) {  
  return (  
    <ul>  
      {props.items.map(item => (  
        <li key={item.id}>{item.text}</li>  
      ))}  
    </ul>  
  );  
}
```

- (b) Javascript and Elm support a functional programming paradigm. Explain what is meant by functional programming, mentioning the important features that a language must support in order to be considered functional. In your opinion, why do you think this style has become so popular in rich web application development in recent years?

(18 Marks)

Functional programming is a programming paradigm that is based on the idea of treating computation as the evaluation of mathematical functions. In functional programming, functions are first-class citizens, which means that they can be treated like any other value in the language, such as being passed as arguments to other functions or being returned as results.

There are several important features that a language must support in order to be considered functional:

- Pure functions: In functional programming, functions should be pure, which means that they should not have any side effects and should always produce the same output for a given input.
- Immutability: In functional programming, it is generally best practice to use immutable data structures, which means that once a value is set, it cannot be changed. This can make it easier to reason about the code and avoid unexpected bugs.
- Referential transparency: In functional programming, expressions should be referentially transparent, which means that they can be replaced with their values without affecting the behavior of the program.
- Higher-order functions: In functional programming, functions should be able to accept other functions as arguments or return them as results, which allows for powerful abstractions and code reuse.

Functional programming has become popular in rich web application development in recent years because it can make it easier to write reliable and maintainable code. The emphasis on pure functions and immutability can help to prevent unintended side effects and reduce the risk of bugs, and the use of higher-order functions can make it easier to write reusable and composable code.

Additionally, functional programming can make it easier to reason about and debug code, as it encourages a more declarative and modular style of programming. This can be especially useful in large and complex applications, where it can be difficult to understand and manage the interactions between different parts of the codebase.

Furthermore, functional programming can be particularly well-suited for concurrent and parallel programming, as it can make it easier to write code that is free from race conditions and other concurrency-related issues. This can be particularly useful in modern web applications, which often rely on multiple asynchronous operations and need to be able to scale to meet the demands of large numbers of users.