Modern Application Development II - Detailed Study Notes

Table of Contents

JavaScript Fundamentals

Basic Syntax and Data Types

Variables and Declarations

```
// let - block scoped variable
let count = 0;

// const - immutable reference
const PI = 3.14;

// var - function scoped (avoid using)
var oldWay = "legacy";
```

Data Types

- **Primitive Types**:
    - Number: `let age = 25;`
    - String: `let name = "John";`
    - Boolean: `let isActive = true;`
    - Undefined: `let x;`

- Null: `let empty = null;`

- **Complex Types**:
  - Objects: `let person = {name: "John", age: 25};`
  - Arrays: `let numbers = [1, 2, 3, 4];`
  - Functions: `function greet() { return "Hello"; }`

Control Flow

```
// If-else
if (condition) {
    // code
} else {
    // code
}

// Loops
for (let i = 0; i < 5; i++) {
    console.log(i);
}

// While loop
while (condition) {
    // code
}
```

Advanced JavaScript Concepts

Functions and Scope

```
// Function Declaration
function add(a, b) {
    return a + b;
}

// Arrow Function
const multiply = (a, b) => a * b;

// Function Expression
const divide = function(a, b) {
    return a / b;
}
```

Promises and Async/Await

```
// Promise Example
const fetchData = new Promise((resolve, reject) => {
    setTimeout(() => {
        const data = {id: 1, name: "Test"};
        resolve(data);
    }, 2000);
});

// Async/Await
async function getData() {
    try {
        const result = await fetchData;
        console.log(result);
    } catch (error) {
        console.error(error);
    }
}
```

Vue.js Framework

Component Structure

```vue
<!-- MyComponent.vue -->
<template>
  <div class="component">
    <h1>{{ title }}</h1>
    <p>{{ message }}</p>
  </div>
</template>

<script>
export default {
  name: 'MyComponent',
  data() {
    return {
      title: 'Hello Vue!',
      message: 'Welcome to my component'
    }
  }
}
</script>
```

Directives

```vue
<!-- Common Vue Directives -->
<template>
  <div>
    <!-- Conditional Rendering -->
    <p v-if="showMessage">Visible when true</p>

    <!-- List Rendering -->
    <ul>
      <li v-for="item in items" :key="item.id">
        {{ item.name }}
      </li>
    </ul>

    <!-- Event Handling -->
    <button v-on:click="handleClick">Click Me</button>

    <!-- Two-way Binding -->
    <input v-model="inputText">
  </div>
</template>
```

Frontend Development

Best Practices

1. **Responsive Design**

```
/* Mobile-first approach */
.container {
    width: 100%;
    padding: 15px;
}

@media (min-width: 768px) {
    .container {
        width: 750px;
        margin: 0 auto;
    }
}
```

2. **Component Organization**

```
src/
├── components/
│   ├── common/
│   │   ├── Button.vue
│   │   └── Input.vue
│   └── features/
│       ├── UserProfile.vue
│       └── Dashboard.vue
├── views/
└── App.vue
```

State Management

Types of State

1. **Local State**

```
// Component-level state
data() {
    return {
        count: 0,
        userInput: ''
    }
}
```

2. **Global State (Vuex)**

```
// Store definition
const store = new Vuex.Store({
    state: {
        user: null,
        isAuthenticated: false
    },
    mutations: {
        setUser(state, user) {
            state.user = user;
            state.isAuthenticated = !!user;
        }
    }
});
```

Practice Questions

1. **JavaScript Basics**
    - Explain the difference between `let`, `const`, and `var`.
    - What is hoisting in JavaScript?
    - How does closure work in JavaScript?
2. **Vue.js Concepts**
    - What is the Vue instance lifecycle?
    - Explain the difference between `v-if` and `v-show`.
    - How does two-way binding work in Vue.js?
3. **Coding Challenges**

```
// Challenge 1: Implement a function that returns a Promise
// which resolves after a given number of milliseconds

function delay(ms) {
    // Your code here
}

// Challenge 2: Create a Vue component that displays a countdown timer
```

4. **State Management**
    - What are the different types of state in a Vue application?
    - When should you use Vuex vs local component state?
    - Design a state management solution for a shopping cart feature.

Sample Solutions

```
// Delay function solution
function delay(ms) {
    return new Promise(resolve => setTimeout(resolve, ms));
}

// Usage
async function example() {
    console.log('Starting');
    await delay(2000);
    console.log('After 2 seconds');
}
```

```
<!-- Countdown Timer Component -->
<template>
  <div>
    <h2>Countdown: {{ timeLeft }} seconds</h2>
    <button @click="startTimer">Start</button>
  </div>
</template>

<script>
export default {
  data() {
    return {
      timeLeft: 60,
      timerId: null
    }
  },
  methods: {
    startTimer() {
      this.timerId = setInterval(() => {
        if (this.timeLeft > 0) {
          this.timeLeft--;
        } else {
          clearInterval(this.timerId);
        }
      }, 1000);
    }
  },
  beforeDestroy() {
    if (this.timerId) {
      clearInterval(this.timerId);
    }
  }
}
</script>
```

Remember to:

- Practice writing code regularly
- Understand concepts through practical implementation
- Review and test your code
- Follow Vue.js best practices and conventions
- Consider performance implications when managing state