🎓 JavaScript DOM & Events - Comprehensive Practice Questions

Complete Practice Set for Midterm Preparation

This is a practice resource, not a timed exam. Work through all questions to master the concepts!

📚 Table of Contents

1. Part A: Multiple Choice Questions
2. Part B: Guess the Output
3. Part C: Code Implementation
4. Part D: Debug the Code
5. Part E: Advanced Challenges
6. Answer Key

Part A: Multiple Choice Questions

🟢 Easy Level (Questions 1-20)

Question 1. Which method returns the FIRST element that matches a CSS selector?

- A) getElementsByClassName()
- B) querySelector()
- C) querySelectorAll()
- D) getElementById()

Question 2. What does `element.classList.toggle('active')` do?

- A) Adds the 'active' class
- B) Removes the 'active' class
- C) Adds the class if it doesn't exist, removes it if it does
- D) Checks if the class exists

Question 3. Which event fires when a user releases a key?

- A) keydown
- B) keyup

- C) keypress
- D) input

Question 4. What is the difference between `textContent` and `innerHTML`?

- A) textContent is faster
- B) innerHTML parses HTML tags, textContent treats everything as plain text
- C) They are the same
- D) innerHTML only works with div elements

Question 5. Which method prevents a form from submitting?

- A) event.stopPropagation()
- B) event.preventDefault()
- C) event.stopImmediatePropagation()
- D) event.cancel()

Question 6. What does `event.target` refer to?

- A) The element that has the event listener
- B) The element that triggered the event
- C) The parent element

- D) The window object

Question 7. Which of the following creates a new HTML element?

- A) document.createElement('div')
- B) document.newElement('div')
- C) document.addElement('div')
- D) new Element('div')

Question 8. What is event delegation?

- A) Removing event listeners
- B) Adding one event listener to a parent to handle events from children
- C) Creating custom events
- D) Preventing default behavior

Question 9. Which array method creates a NEW array with all elements that pass a test?

- A) forEach()
- B) map()
- C) filter()
- D) reduce()

Question 10. How do you store data in localStorage?

- A) localStorage.save('key', 'value')
- B) localStorage.setItem('key', 'value')
- C) localStorage.add('key', 'value')
- D) localStorage.store('key', 'value')

Question 11. What will `querySelectorAll('.item')` return?

- A) An Array
- B) A NodeList
- C) An HTMLCollection
- D) A single element

Question 12. Which event phase occurs FIRST?

- A) Bubble phase
- B) Target phase
- C) Capture phase
- D) Event phase

Question 13. What does `element.remove()` do?

- A) Removes all child elements

- B) Removes the element from the DOM
- C) Removes all classes from the element
- D) Removes all event listeners

Question 14. Which method adds an element as the last child?

- A) appendChild()
- B) insertBefore()
- C) insertAfter()
- D) addChild()

Question 15. How do you retrieve JSON data from localStorage?

- A) localStorage.getItem('key')
- B) JSON.parse(localStorage.getItem('key'))
- C) localStorage.parse('key')
- D) JSON.get(localStorage.getItem('key'))

Question 16. Which method removes the LAST element from an array?

- A) pop()
- B) shift()

- C) splice()
- D) slice()

Question 17. What does `addEventListener()` return?

- A) The event object
- B) The element
- C) A reference to the listener
- D) undefined

Question 18. Which property gets/sets the HTML content of an element?

- A) textContent
- B) innerText
- C) innerHTML
- D) content

Question 19. How do you select an element with id="demo"?

- A) document.querySelector('#demo')
- B) Both A and C
- C) document.getElementById('demo')

- D) document.getElement('demo')

Question 20. What does `array.length` return?

- A) The last element
- B) The number of elements
- C) The first element
- D) undefined

🟡 Medium Level (Questions 21-40)

Question 21. What's the difference between `event.target` and `event.currentTarget`?

- A) They are the same
- B) target is where the event originated, currentTarget is where the listener is attached
- C) currentTarget is always the window
- D) target is always the parent element

Question 22. Which is TRUE about `NodeList` from `querySelectorAll()`?

- A) It's a live collection
- B) It's a static collection
- C) It automatically updates when DOM changes
- D) You can't use forEach on it

Question 23. What happens if you call `preventDefault()` on a non-cancelable event?

- A) Error is thrown
- B) Nothing happens

- C) Event stops propagating
- D) Page reloads

Question 24. Which method would you use to insert an element BEFORE another element?

- A) appendChild()
- B) insertBefore()
- C) insertAfter()
- D) prepend()

Question 25. What does `element.closest('.container')` do?

- A) Selects all containers
- B) Selects the first child with class container
- C) Selects the nearest ancestor (including self) with class container
- D) Creates a new container element

Question 26. Which is NOT a valid way to add multiple classes?

- A) element.classList.add('class1', 'class2')
- B) element.className = 'class1 class2'

- C) element.classList.add('class1 class2')
- D) element.setAttribute('class', 'class1 class2')

Question 27. What does `array.map()` return?

- A) The modified original array
- B) A new array with transformed elements
- C) The first matching element
- D) undefined

Question 28. How do you remove an event listener?

- A) element.removeEventListener('click', functionName)
- B) element.deleteEvent('click')
- C) element.off('click')
- D) element.removeListener('click')

Question 29. What's the difference between `==` and `===`?

- A) No difference
- B) === checks type and value, == only checks value
- C) == is faster
- D) === only works with numbers

Question 30. Which array method modifies the original array?

- A) map()
- B) filter()
- C) push()
- D) slice()

Question 31. What does `element.cloneNode(true)` do?

- A) Clones element without children
- B) Clones element with all children
- C) Creates a new element
- D) Copies element styles

Question 32. Which is TRUE about `let` vs `const`?

- A) const variables cannot be reassigned
- B) let variables are global
- C) const is faster
- D) let cannot be used in functions

Question 33. What does `array.reduce()` do?

- A) Filters elements
- B) Reduces array to a single value
- C) Removes duplicates
- D) Sorts the array

Question 34. How do you check if an element has a specific class?

- A) element.hasClass('active')
- B) element.classList.contains('active')
- C) element.className === 'active'
- D) element.checkClass('active')

Question 35. What's the correct way to loop through a NodeList?

- A) for...in loop only
- B) forEach() or for...of loop
- C) while loop only
- D) NodeLists cannot be looped

Question 36. Which method stops event propagation to parent elements?

- A) event.cancel()

- B) event.stopPropagation()
- C) event.stop()
- D) event.preventDefault()

Question 37. What does `JSON.stringify()` do?

- A) Converts JSON to object
- B) Converts object to JSON string
- C) Validates JSON
- D) Parses a string

Question 38. Which is the correct syntax for a template literal?

- A) 'Hello ${name}'
- B) "Hello ${name}"
- C) `Hello ${name}`
- D) 'Hello ' + ${name}

Question 39. What does `array.slice(1, 3)` return?

- A) Elements at index 1 and 2
- B) Elements at index 1, 2, and 3
- C) Removes elements

- D) Returns element at index 1

Question 40. How do you get the parent element?

- A) element.parent
- B) element.parentNode or element.parentElement
- C) element.getParent()
- D) element.ancestor

🔴 Hard Level (Questions 41-50)

Question 41. What's the output of `typeof null`?

- A) "null"
- B) "object"
- C) "undefined"
- D) "number"

Question 42. What happens with multiple `stopPropagation()` calls in the same event?

- A) Error is thrown
- B) Only first one works
- C) All work but have same effect
- D) Last one overrides others

Question 43. Which is TRUE about event capturing?

- A) It's the default phase
- B) Events travel from window to target
- C) It happens after bubbling
- D) It's not supported in modern browsers

Question 44. What's the difference between `childNodes` and `children`?

- A) No difference
- B) childNodes includes text nodes, children only element nodes
- C) children includes text nodes
- D) childNodes is deprecated

Question 45. What does `element.dataset.userId` access?

- A) The element's ID
- B) The data-user-id attribute
- C) A JavaScript property
- D) The element's class

Question 46. What happens if you try to access `localStorage` in private/incognito mode?

- A) It works normally
- B) May throw an error or have reduced quota
- C) Returns undefined
- D) localStorage is always available

Question 47. What's the difference between `Array.from()` and spread operator `[ ... ]`?

- A) Array.from() can take a mapping function as second argument
- B) No difference
- C) Spread is faster always
- D) Array.from() only works with arrays

Question 48. What's the performance benefit of event delegation?

- A) No benefit
- B) Fewer event listeners means less memory usage
- C) Events fire faster
- D) Only works in modern browsers

Question 49. What does `event.stopImmediatePropagation()` do differently from `stopPropagation()`?

- A) They're the same
- B) Also prevents other listeners on the same element from executing

- C) Only works on parent elements
- D) It's deprecated

Question 50. In event flow, what is the order of phases?

- A) Target → Capture → Bubble
- B) Bubble → Target → Capture
- C) Capture → Target → Bubble
- D) Target → Bubble → Capture

Part B: Guess the Output

● Easy Level (Questions 51-60)

Question 51. What will be logged?

```
const numbers = [1, 2, 3, 4, 5];
const result = numbers.filter(num => num % 2 === 0);
console.log(result);
```

Question 52. What will be displayed in the paragraph?

```
// HTML: <p id="demo"></p>
const p = document.getElementById('demo');
p.textContent = '<b>Hello</b>';
```

Question 53. What will be logged?

```
const arr = [10, 20, 30];
const doubled = arr.map(x => x * 2);
console.log(doubled);
```

Question 54. What happens when you click the button?

```
// HTML: <button id="btn">Click</button>
const btn = document.getElementById('btn');
btn.addEventListener('click', function() {
    console.log('Clicked!');
});
btn.addEventListener('click', function() {
    console.log('Clicked again!');
});
```

Question 55. What is the final value?

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((total, num) => total + num,
console.log(sum);
```

Question 56. What classes will the div have?

```
// HTML: <div id="box" class="container"></div>
const box = document.getElementById('box');
box.classList.add('active');
box.classList.remove('container');
box.classList.toggle('highlight');
```

Question 57. What will be the length?

```
let fruits = ['apple', 'banana', 'orange'];
fruits.push('grape');
fruits.shift();
console.log(fruits.length);
```

Question 58. What will be logged?

```
const items = ['a', 'b', 'c'];
items.forEach((item, index) => {
    console.log(index + ': ' + item);
});
```

Question 59. What is the result?

```
const user = { name: 'John', age: 25 };
const json = JSON.stringify(user);
console.log(typeof json);
```

Question 60. How many elements are selected?

```
// HTML: <p class="text">1</p> <p>2</p> <p class="text"
const paragraphs = document.querySelectorAll('.text');
console.log(paragraphs.length);
```

🟠 Medium Level (Questions 61-75)

Question 61. What will be logged?

```
const arr = [1, 2, 3];
arr.forEach(num => {
    if (num === 2) return;
    console.log(num);
});
```

Question 62. What is the output?

```
const numbers = [5, 10, 15, 20];
const result = numbers.find(n => n > 12);
console.log(result);
```

Question 63. What will be in the array?

```
const arr = [1, 2, 3, 4, 5];
arr.splice(2, 1);
console.log(arr);
```

Question 64. What gets logged?

```
const obj = { a: 1, b: 2, c: 3 };
console.log(Object.keys(obj));
```

Question 65. What is displayed?

```
// HTML: <div id="container"><span>Hello</span></div>
const div = document.getElementById('container');
console.log(div.children.length);
console.log(div.childNodes.length);
```

Question 66. What will be logged?

```
let x = 5;
let y = x;
x = 10;
console.log(y);
```

Question 67. What happens?

```
const arr = [1, 2, 3];
const newArr = arr;
newArr.push(4);
console.log(arr);
```

Question 68. What is the output?

```
const students = [
    { name: 'Alice', age: 20 },
    { name: 'Bob', age: 22 },
    { name: 'Charlie', age: 20 }
];
const result = students.filter(s => s.age === 20).map(s
console.log(result);
```

Question 69. What gets logged?

```
console.log('5' + 3);
console.log('5' - 3);
```

Question 70. What will be the output?

```
const arr = [1, 2, 2, 3, 3, 3];
const unique = [...new Set(arr)];
console.log(unique);
```

Question 71. What happens?

```
// HTML: <input type="text" id="name" value="John">
const input = document.getElementById('name');
console.log(input.value);
console.log(input.getAttribute('value'));
```

Question 72. What is logged?

```
const arr = ['a', 'b', 'c'];
const result = arr.slice(1);
console.log(result);
console.log(arr);
```

Question 73. What will be the output?

```
const obj = { x: 10 };
function modify(o) {
    o.x = 20;
}
modify(obj);
console.log(obj.x);
```

Question 74. What gets logged?

```
const arr = [1, 2, 3, 4, 5];
const result = arr.filter(n => n > 2).map(n => n * 2);
console.log(result);
```

Question 75. What is the output?

```javascript
let count = 0;
const btn = document.createElement('button');
btn.addEventListener('click', () => {
    count++;
    console.log(count);
});
btn.click();
btn.click();
```

🔴 Hard Level (Questions 76-85)

Question 76. What will be logged?

```javascript
const arr = [1, 2, 3];
arr[10] = 10;
console.log(arr.length);
console.log(arr[5]);
```

Question 77. What happens?

```javascript
// HTML: <div id="parent"><div id="child">Click</div></
document.getElementById('parent').addEventListener('cli
    console.log('Parent');
});
document.getElementById('child').addEventListener('clic
    e.stopPropagation();
    console.log('Child');
});
// User clicks on child div
```

Question 78. What is the output?

```
const arr = [1, 2, 3];
const result = arr.reduce((acc, val) => {
    acc.push(val * 2);
    return acc;
}, []);
console.log(result);
```

Question 79. What gets logged?

```
console.log([] + []);
console.log([] + {});
console.log({} + []);
```

Question 80. What will happen?

```
const div = document.createElement('div');
div.innerHTML = '<p>Test</p>';
console.log(div.children.length);
console.log(div.childNodes.length);
```

Question 81. What is the output?

```
const obj = { a: 1, b: 2 };
const obj2 = obj;
obj2.a = 10;
console.log(obj.a);
```

Question 82. What gets logged?

```
let a = [1, 2, 3];
let b = a.map(x => {
    if (x === 2) return;
    return x * 2;
});
console.log(b);
```

Question 83. What is the result?

```
const arr = [1, [2, [3, [4]]]];
console.log(arr.flat(2));
```

Question 84. What happens?

```
const p = document.createElement('p');
p.textContent = 'Hello';
console.log(p.textContent);
console.log(document.body.contains(p));
```

Question 85. What is logged?

```
const obj = { a: 1 };
Object.freeze(obj);
obj.a = 2;
console.log(obj.a);
```

Part C: Code Implementation

🟢 Easy Level (Questions 86-90)

Question 86. Create a Counter Application (10 points)

HTML is provided:

```html
<div id="counter">0</div>
<button id="increment">+</button>
<button id="decrement">-</button>
<button id="reset">Reset</button>
```

Requirements:

- Increment counter when clicking +
- Decrement counter when clicking -
- Reset to 0 when clicking Reset
- Counter should not go below 0

Question 87. Input Validator (10 points)

HTML is provided:

```
<input type="text" id="username" placeholder="Username"
<div id="message"></div>
```

Requirements:

- Listen for input events
- Display "Valid" if username length >= 5
- Display "Too short" if length < 5
- Change message background to green for valid, red for invalid

Question 88. Toggle Visibility (8 points)

HTML is provided:

```
<button id="toggleBtn">Toggle</button>
<div id="content">This is some content</div>
```

Requirements:

- Toggle the visibility of the content div when button is clicked
- Use `display: none` to hide and `display: block` to show

Question 89. Change Background Color (8 points)

HTML is provided:

```
<button id="redBtn">Red</button>
<button id="blueBtn">Blue</button>
<button id="greenBtn">Green</button>
<div id="box" style="width: 200px; height: 200px; borde
```

Requirements:

- Change box background to red when clicking Red button
- Change box background to blue when clicking Blue button
- Change box background to green when clicking Green button

Question 90. Display Array Items (8 points)

Given:

```
const fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
```

HTML is provided:

```
<ul id="fruitList"></ul>
```

Requirements:

- Loop through the fruits array
- Create an `<li>` for each fruit
- Append each `<li>` to the `<ul>`

🟡 Medium Level (Questions 91-95)

Question 91. Dynamic List Manager (12 points)

HTML is provided:

```
<input type="text" id="itemInput" placeholder="Enter it
<button id="addBtn">Add Item</button>
<ul id="itemList"></ul>
```

Requirements:

- Add new list item when clicking Add button
- Each item should have a "Delete" button next to it
- Clicking Delete removes that specific item
- Clear input after adding
- Use event delegation for delete buttons
- Don't add empty items

Question 92. Grade Calculator (12 points)

HTML is provided:

```
<input type="number" id="score" placeholder="Enter scor
<button id="calculateBtn">Calculate Grade</button>
<div id="result"></div>
```

Requirements:

- Calculate letter grade based on score:
  - A: 90-100
  - B: 80-89
  - C: 70-79
  - D: 60-69
  - F: 0-59
- Display the grade in the result div
- Validate that score is between 0-100
- Show error message for invalid input

Question 93. Array Manipulation (10 points)

Given:

```
const students = [
    { name: 'Alice', grade: 85 },
    { name: 'Bob', grade: 92 },
    { name: 'Charlie', grade: 78 },
    { name: 'David', grade: 95 },
    { name: 'Eve', grade: 88 }
];
```

HTML is provided:

```
<div id="result"></div>
```

Requirements:

- Filter students with grade >= 90
- Get an array of just their names
- Sort names alphabetically
- Display in the result div as a comma-separated string

Question 94. Form Validation (12 points)

HTML is provided:

```
<form id="registrationForm">
    <input type="text" id="username" placeholder="Usern
    <input type="email" id="email" placeholder="Email">
    <input type="password" id="password" placeholder="P
    <button type="submit">Register</button>
</form>
<div id="errors"></div>
```

Requirements:

- Prevent form submission
- Validate username (min 4 characters)
- Validate email (must contain @)
- Validate password (min 6 characters)
- Display all errors in the errors div
- If all valid, show success message

Question 95. Character Counter (10 points)

HTML is provided:

```
<textarea id="message" maxlength="100" placeholder="Typ
<div id="charCount">0 / 100</div>
```

Requirements:

- Update character count as user types
- Display format: "current / max"
- Change color to red when > 80 characters
- Use the `input` event

🔴 Hard Level (Questions 96-100)

Question 96. Filterable Table (15 points)

HTML is provided:

```
<input type="text" id="searchInput" placeholder="Search
<table id="studentTable">
    <thead>
        <tr><th>Name</th><th>Age</th><th>Grade</th></tr
    </thead>
    <tbody id="tableBody"></tbody>
</table>
```

Given data:

```
const students = [
    { name: 'Alice Johnson', age: 20, grade: 85 },
    { name: 'Bob Smith', age: 22, grade: 92 },
    { name: 'Charlie Brown', age: 21, grade: 78 },
    { name: 'Diana Prince', age: 23, grade: 95 }
];
```

Requirements:

- Populate table with student data initially
- Filter table rows in real-time as user types in search

- Search should be case-insensitive
- Show all rows if search is empty

Question 97. Tab Component (15 points)

HTML is provided:

```
<div class="tabs">
    <button class="tab-btn active" data-tab="tab1">Tab
    <button class="tab-btn" data-tab="tab2">Tab 2</butt
    <button class="tab-btn" data-tab="tab3">Tab 3</butt
</div>
<div class="tab-content">
    <div id="tab1" class="tab-pane active">Content 1</d
    <div id="tab2" class="tab-pane">Content 2</div>
    <div id="tab3" class="tab-pane">Content 3</div>
</div>
```

Requirements:

- Only one tab should be active at a time
- Clicking a tab button shows its content and hides others
- Add 'active' class to active button and content
- Remove 'active' from inactive ones
- Use event delegation

Question 98. Local Storage To-Do List (18 points)

HTML is provided:

```
<input type="text" id="todoInput" placeholder="New todo
<button id="addTodo">Add</button>
<ul id="todoList"></ul>
```

Requirements:

- Add new todo items
- Each todo has a checkbox and delete button
- Checkbox toggles completed state (use strikethrough)
- Save todos to localStorage
- Load todos from localStorage on page load
- Delete removes from both DOM and localStorage
- Store as array of objects: `[{text: 'Task', completed: false}]`

Question 99. Drag and Drop Sorter (18 points)

HTML is provided:

```
<div id="container">
    <div class="draggable-item" draggable="true">Item 1
    <div class="draggable-item" draggable="true">Item 2
    <div class="draggable-item" draggable="true">Item 3
</div>
```

Requirements:

- Make items draggable
- Allow reordering items by dragging
- Use dragstart, dragover, drop events
- Prevent default on dragover
- Swap positions of dragged and target items

Question 100. Live Form Preview (20 points)

HTML is provided:

```
<form id="profileForm">
    <input type="text" id="name" placeholder="Name">
    <input type="email" id="email" placeholder="Email">
    <select id="country">
        <option value="">Select Country</option>
        <option value="USA">USA</option>
        <option value="UK">UK</option>
        <option value="Canada">Canada</option>
    </select>
    <textarea id="bio" placeholder="Bio"></textarea>
</form>
<div id="preview">
    <h3>Live Preview</h3>
    <p><strong>Name:</strong> <span id="previewName"></
    <p><strong>Email:</strong> <span id="previewEmail">
    <p><strong>Country:</strong> <span id="previewCount
    <p><strong>Bio:</strong> <span id="previewBio"></sp
</div>
```

Requirements:

- Update preview in real-time as user types
- Use `input` or `change` events appropriately
- Handle all form fields (text, email, select, textarea)
- If field is empty, show placeholder text in preview

Part D: Debug the Code

Find and fix the errors in the following code snippets.

Question 101. What's wrong with this code?

```
const button = document.getElementById('myButton');
button.addEventListner('click', function() {
    console.log('Clicked!');
});
```

Question 102. What's the issue here?

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(function(num) {
    num * 2;
});
console.log(doubled);
```

Question 103. Fix this code:

```
const items = document.getElementsByClassName('item');
items.forEach(item => {
    item.style.color = 'red';
});
```

Question 104. What's wrong?

```
const data = localStorage.getItem('user');
console.log(data.name);
```

Question 105. Debug this:

```
const form = document.getElementById('myForm');
form.addEventListener('submit', function() {
    console.log('Form submitted!');
    // Form still submits to server
});
```

Question 106. What's the problem?

```
for (var i = 0; i < 3; i++) {
    setTimeout(function() {
        console.log(i);
    }, 1000);
}
// Expected: 0, 1, 2
// Actual: 3, 3, 3
```

Question 107. Fix this code:

```
const numbers = [1, 2, 3];
const sum = numbers.reduce((total, num) => total + num)
console.log(sum); // NaN
```

Question 108. What's wrong?

```
const div = document.createElement('div');
div.innerHTML = '<p>Hello</p>';
console.log(div.parentElement); // null - why?
```

Question 109. Debug:

```
const obj = {
    name: 'John',
    age: 25
};
localStorage.setItem('user', obj);
// Later...
const user = localStorage.getItem('user');
console.log(user.name); // undefined - why?
```

Question 110. Fix this:

```javascript
const students = [
    { name: 'Alice', grade: 85 },
    { name: 'Bob', grade: 92 }
];
students.sort();
console.log(students); // Not sorted by grade
```

Part E: Advanced Challenges

Question 111. Implement a debounce function (20 points)

Create a `debounce` function that delays the execution of a function until after a specified time has elapsed since the last time it was invoked.

```
function debounce(func, delay) {
    // Your code here
}

// Usage:
const searchInput = document.getElementById('search');
searchInput.addEventListener('input', debounce(function
    console.log('Searching for:', e.target.value);
}, 500));
```

Question 112. Create a modal system (25 points)

HTML is provided:

```html
<button id="openModal">Open Modal</button>
<div id="modal" class="modal">
    <div class="modal-content">
        <span class="close">&times;</span>
        <h2>Modal Title</h2>
        <p>Modal content goes here</p>
    </div>
</div>
```

Requirements:

- Show modal when clicking the button
- Hide modal when clicking X
- Hide modal when clicking outside the modal content
- Prevent body scroll when modal is open
- Add fade-in/fade-out effect with CSS

Question 113. Implement event delegation for dynamic content (20 points)

Create a comment section where:

- Users can add comments
- Each comment has a "Reply" button
- Clicking Reply shows a reply form under that comment
- All functionality must use event delegation
- Comments can be nested (replies to replies)

Question 114. Create a shopping cart (30 points)

Requirements:

- Display list of products (name, price, image)
- Add to cart button for each product
- Cart shows added items with quantity
- Increase/decrease quantity buttons
- Remove item button
- Show total price
- Save cart to localStorage
- Load cart on page refresh

Question 115. Build a live search with API simulation (25 points)

Create a search feature that:

- Shows results as user types (with debouncing)
- Highlights matching text in results
- Shows "Loading..." while searching
- Shows "No results" if nothing found
- Allows clicking a result to select it
- Use this mock data:

```
const users = [
    { id: 1, name: 'Alice Johnson', email: 'alice@examp
    { id: 2, name: 'Bob Smith', email: 'bob@example.com
    { id: 3, name: 'Charlie Brown', email: 'charlie@exa
];
```

Answer Key

Part A: Multiple Choice Answers

Easy Level (1-20)

1. **B** - querySelector()
2. **C** - Adds if doesn't exist, removes if it does
3. **B** - keyup
4. **B** - innerHTML parses HTML, textContent is plain text
5. **B** - event.preventDefault()
6. **B** - Element that triggered the event
7. **A** - document.createElement('div')
8. **B** - One listener on parent for children
9. **C** - filter()
10. **B** - localStorage.setItem('key', 'value')
11. **B** - A NodeList
12. **C** - Capture phase
13. **B** - Removes element from DOM
14. **A** - appendChild()
15. **B** - JSON.parse(localStorage.getItem('key'))

16. A - pop()

17. D - undefined

18. C - innerHTML

19. B - Both A and C

20. B - Number of elements

Medium Level (21-40)

21. B - target is origin, currentTarget is listener location

22. B - Static collection

23. B - Nothing happens

24. B - insertBefore()

25. C - Nearest ancestor with class

26. C - Can't add multiple classes as one string with add()

27. B - New array with transformed elements

28. A - element.removeEventListener('click', functionName)

29. B - === checks type and value

30. C - push()

31. B - Clones with all children

32. A - const cannot be reassigned

33. B - Reduces to single value

34. B - element.classList.contains('active')

35. B - forEach() or for...of

36. B - event.stopPropagation()

37. B - Object to JSON string

38. C - `Hello ${name}`

39. A - Elements at index 1 and 2

40. B - parentNode or parentElement

Hard Level (41-50)

41. B - "object" (JavaScript quirk)

42. C - All work but same effect

43. B - Events travel window to target

44. B - childNodes includes text nodes

45. B - data-user-id attribute

46. B - May throw error or reduced quota

47. A - Array.from() takes mapping function

48. B - Fewer listeners, less memory

49. B - Prevents other listeners on same element

50. C - Capture → Target → Bubble

Part B: Output Answers

Easy Level (51-60)

51. `[2, 4]`

52. `<b>Hello</b>` (as plain text, not bold)

53. `[20, 40, 60]`

54. Both "Clicked!" and "Clicked again!" are logged

55. `10`

56. `"active"` and `"highlight"` (container removed)

57. `3`

58. `"0: a"`, `"1: b"`, `"2: c"` (on separate lines)

59. `"string"`

60. `2`

Medium Level (61-75)

61. `1` and `3` (return skips only current iteration)

62. `15` (first element > 12)

63. `[1, 2, 4, 5]` (removed index 2)

64. `["a", "b", "c"]`

65. `1` (one element child) and `1` or `3` depending on whitespace
66. `5` (primitives are copied by value)
67. `[1, 2, 3, 4]` (arrays are reference types)
68. `["Alice", "Charlie"]`
69. `"53"` (concatenation) and `2` (coercion to number)
70. `[1, 2, 3]` (Set removes duplicates)
71. `"John"` (both) or might differ if user changed it
72. `["b", "c"]` and `["a", "b", "c"]` (slice doesn't modify original)
73. `20` (objects passed by reference)
74. `[6, 8, 10]` (chained methods)
75. `1` then `2` (count increments each click)

Hard Level (76-85)

76. `11` and `undefined`
77. Only "Child" (stopPropagation prevents bubble)
78. `[2, 4, 6]`
79. `""` (empty string), `"[object Object]"`, `"[object Object]"` or `0`
80. `1` and `1`

81. `10` (objects are references)

82. `[2, undefined, 6]` (no explicit return gives undefined)

83. `[1, 2, [3, [4]]]` (flattens 2 levels)

84. `"Hello"` and `false` (not appended to DOM)

85. `1` (Object.freeze prevents modification)

Part C: Implementation Solutions

Question 86: Counter

```javascript
let count = 0;
const counter = document.getElementById('counter');
const incBtn = document.getElementById('increment');
const decBtn = document.getElementById('decrement');
const resetBtn = document.getElementById('reset');

incBtn.addEventListener('click', () => {
    count++;
    counter.textContent = count;
});

decBtn.addEventListener('click', () => {
    if (count > 0) {
        count--;
        counter.textContent = count;
    }
});

resetBtn.addEventListener('click', () => {
    count = 0;
    counter.textContent = count;
});
```

Question 87: Input Validator

```
const username = document.getElementById('username');
const message = document.getElementById('message');

username.addEventListener('input', (e) => {
    const value = e.target.value;
    if (value.length >= 5) {
        message.textContent = 'Valid';
        message.style.backgroundColor = 'lightgreen';
    } else {
        message.textContent = 'Too short';
        message.style.backgroundColor = 'lightcoral';
    }
});
```

Question 91: Dynamic List Manager

```javascript
const itemInput = document.getElementById('itemInput');
const addBtn = document.getElementById('addBtn');
const itemList = document.getElementById('itemList');

addBtn.addEventListener('click', () => {
    const value = itemInput.value.trim();
    if (value) {
        const li = document.createElement('li');
        li.innerHTML = `${value} <button class="delete-
        itemList.appendChild(li);
        itemInput.value = '';
    }
});

// Event delegation
itemList.addEventListener('click', (e) => {
    if (e.target.classList.contains('delete-btn')) {
        e.target.parentElement.remove();
    }
});
```

Question 93: Array Manipulation

```
const result = students
    .filter(s => s.grade >= 90)
    .map(s => s.name)
    .sort();

document.getElementById('result').textContent = result.
// Output: "Bob, David"
```

Part D: Debug Solutions

101. Typo: `addEventListner` → `addEventListener`

102. Missing return: `return num * 2;` or `num => num * 2`

103. HTMLCollection not array: `Array.from(items).forEach( ... )` or use for loop

104. Need to parse JSON: `JSON.parse(localStorage.getItem('user'))`

105. Missing preventDefault: `e.preventDefault();`

106. Use let instead of var: `for (let i = 0; i < 3; i++)`

107. Missing initial value: `reduce((total, num) => total + num, 0)`

108. Element not in DOM yet (not appended)

109. Need to stringify: `localStorage.setItem('user', JSON.stringify(obj))`

110. Need compare function:

```
students.sort((a, b) => a.grade - b.grade)
```

Good luck with your exam! 🚀