Advanced JavaScript Notes

Asynchronous JavaScript

Callbacks

javascript

Copy

```javascript
function fetchData(callback) {
  setTimeout(() => {
    const data = { name: "John", age: 30 };
    callback(data);
  }, 2000);
}

fetchData((data) => {
  console.log(data); // { name: "John", age: 30 }
});
```

Promises

javascript

```javascript
function fetchUserData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      const success = true;
      if (success) {
        resolve({ id: 1, username: "john_doe" });
      } else {
        reject(new Error("Failed to fetch user data"));
      }
    }, 2000);
  });
}

fetchUserData()
  .then(user => {
    console.log("User data:", user);
    return fetchUserPosts(user.id);
  })
  .then(posts => {
    console.log("User posts:", posts);
  })
  .catch(error => {
    console.error("Error:", error.message);
  })
  .finally(() => {
    console.log("Operation completed");
  });
```

Async/Await

javascript Copy

```javascript
async function getUserData() {
  try {
    const user = await fetchUserData();
    console.log("User:", user);

    const posts = await fetchUserPosts(user.id);
    console.log("Posts:", posts);

    return { user, posts };
  } catch (error) {
    console.error("Error:", error.message);
    throw error;
  }
}

// Call the async function
getUserData().then(result => {
  console.log("Complete data:", result);
});
```

ES6+ Features

Destructuring

javascript Copy

```javascript
// Array destructuring
const colors = ["red", "green", "blue"];
const [primaryColor, secondaryColor, tertiaryColor] = colors;

// Object destructuring with default values and renaming
const person = { name: "Alice", age: 28 };
const { name: fullName = "Anonymous", age, occupation = "Unknown" } = person;

// Function parameter destructuring
function displayPerson({ name, age, occupation = "Not specified" }) {
  console.log(`Name: ${name}, Age: ${age}, Occupation: ${occupation}`);
}
```

Spread and Rest Operators

javascript   Copy

```javascript
// Spread with arrays
const numbers = [1, 2, 3];
const moreNumbers = [...numbers, 4, 5, 6];

// Spread with objects
const baseConfig = { theme: "dark", language: "en" };
const userConfig = { ...baseConfig, notifications: true };

// Rest parameter
function sum(...values) {
  return values.reduce((total, value) => total + value, 0);
}
```

Map, Set, WeakMap, and WeakSet

javascript   Copy

```javascript
// Map
const userRoles = new Map();
userRoles.set("john", "admin");
userRoles.set("jane", "editor");
userRoles.set("bob", "subscriber");

// Iterating over Map
for (const [user, role] of userRoles) {
  console.log(`${user} is a ${role}`);
}

// Set (unique values)
const uniqueTags = new Set(["javascript", "programming", "web", "javascript"]);
console.log([...uniqueTags]); // ["javascript", "programming", "web"]

// WeakMap and WeakSet - special collections with weak references to objects
```

Advanced Functions

Higher-Order Functions

javascript

```javascript
// Function that returns a function
function createMultiplier(factor) {
  return function(number) {
    return number * factor;
  };
}

const double = createMultiplier(2);
const triple = createMultiplier(3);

console.log(double(5)); // 10
console.log(triple(5)); // 15
```

Function Currying

javascript

```javascript
// Manual currying
function curry(fn) {
  return function curried(...args) {
    if (args.length >= fn.length) {
      return fn.apply(this, args);
    } else {
      return function(...moreArgs) {
        return curried.apply(this, args.concat(moreArgs));
      };
    }
  };
}

// Usage example
function sum(a, b, c) {
  return a + b + c;
}

const curriedSum = curry(sum);
console.log(curriedSum(1)(2)(3)); // 6
console.log(curriedSum(1, 2)(3)); // 6
console.log(curriedSum(1)(2, 3)); // 6
```

Function Composition

javascript

```javascript
const compose = (...fns) => x => fns.reduceRight((acc, fn) => fn(acc), x);

const double = x => x * 2;
const square = x => x * x;
const addOne = x => x + 1;

const compute = compose(addOne, square, double);
console.log(compute(3)); // (3*2)² + 1 = 37
```

Object-Oriented JavaScript

Classes and Inheritance

javascript

Copy

```javascript
class Person {
  constructor(name, age) {
    this.name = name;
    this.age = age;
  }

  greet() {
    return `Hello, my name is ${this.name} and I'm ${this.age} years old.`;
  }

  // Static method
  static fromBirthYear(name, birthYear) {
    const age = new Date().getFullYear() - birthYear;
    return new this(name, age);
  }

  // Getter
  get description() {
    return `${this.name} (${this.age})`;
  }

  // Setter
  set fullName(value) {
    const parts = value.split(' ');
    this.name = parts.join(' ');
  }
}

// Inheritance
class Employee extends Person {
  constructor(name, age, position, salary) {
    super(name, age);
    this.position = position;
    this.salary = salary;
  }

  greet() {
    return `${super.greet()} I work as a ${this.position}.`;
  }
}

const employee = new Employee("Jane Smith", 32, "Developer", 75000);
console.log(employee.greet());
```

Prototypes

javascript

Copy

```javascript
// Constructor function
function Vehicle(make, model, year) {
  this.make = make;
  this.model = model;
  this.year = year;
}

// Adding methods to prototype
Vehicle.prototype.getInfo = function() {
  return `${this.year} ${this.make} ${this.model}`;
};

Vehicle.prototype.isAntique = function() {
  return new Date().getFullYear() - this.year > 25;
};

// Create instances
const car = new Vehicle("Toyota", "Camry", 2020);
console.log(car.getInfo()); // "2020 Toyota Camry"
```

JavaScript Modules

ES Modules

javascript

```javascript
// math.js
export const PI = 3.14159;

export function add(a, b) {
  return a + b;
}

export function multiply(a, b) {
  return a * b;
}

export default class Calculator {
  // Class implementation
}

// app.js
import Calculator, { PI, add, multiply as mult } from './math.js';
import * as math from './math.js';

console.log(PI); // 3.14159
console.log(add(2, 3)); // 5
console.log(mult(2, 3)); // 6
console.log(math.PI); // 3.14159
```

Advanced DOM Manipulation

Working with DOM Fragments

javascript

Copy

```javascript
function createList(items) {
  const fragment = document.createDocumentFragment();

  items.forEach(item => {
    const li = document.createElement('li');
    li.textContent = item;
    fragment.appendChild(li);
  });

  return fragment;
}

// Usage
const list = document.getElementById('myList');
const items = ['Apple', 'Banana', 'Cherry', 'Date'];
list.appendChild(createList(items));
```

Intersection Observer

javascript

Copy

```javascript
const options = {
  root: null, // viewport
  rootMargin: '0px',
  threshold: 0.5 // 50% visibility
};

const callback = (entries, observer) => {
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      console.log('Element is now visible');
      entry.target.classList.add('visible');
      // Optionally stop observing
      // observer.unobserve(entry.target);
    } else {
      console.log('Element is no longer visible');
      entry.target.classList.remove('visible');
    }
  });
};

const observer = new IntersectionObserver(callback, options);
const target = document.querySelector('.lazy-load');
observer.observe(target);
```

MutationObserver

javascript

Copy

```javascript
// Configure the observer
const config = {
  attributes: true,
  childList: true,
  subtree: true
};

// Callback function
const callback = function(mutationsList, observer) {
  for (const mutation of mutationsList) {
    if (mutation.type === 'childList') {
      console.log('A child node has been added or removed.');
    } else if (mutation.type === 'attributes') {
      console.log(`The ${mutation.attributeName} attribute was modified.`);
    }
  }
};

// Create an observer instance
const observer = new MutationObserver(callback);

// Start observing
const targetNode = document.getElementById('some-id');
observer.observe(targetNode, config);

// Later, you can stop observing
// observer.disconnect();
```

JavaScript Design Patterns

Module Pattern

javascript

```javascript
const calculator = (function() {
  // Private variables and functions
  let result = 0;

  function validate(num) {
    return typeof num === 'number';
  }

  // Public API
  return {
    add: function(num) {
      if (validate(num)) {
        result += num;
      }
      return this;
    },
    subtract: function(num) {
      if (validate(num)) {
        result -= num;
      }
      return this;
    },
    getResult: function() {
      return result;
    },
    reset: function() {
      result = 0;
      return this;
    }
  };
})();

// Usage
calculator.add(5).subtract(2).add(10);
console.log(calculator.getResult()); // 13
```

Factory Pattern

javascript

```javascript
function createUser(type) {
  const user = {
    type: type,
    createdAt: new Date()
  };

  if (type === 'admin') {
    user.permissions = ['read', 'write', 'delete'];
    user.accessLevel = 'full';
  } else if (type === 'editor') {
    user.permissions = ['read', 'write'];
    user.accessLevel = 'partial';
  } else {
    user.permissions = ['read'];
    user.accessLevel = 'basic';
  }

  return user;
}

const admin = createUser('admin');
const editor = createUser('editor');
const reader = createUser('reader');
```

Observer Pattern

javascript

```javascript
class EventEmitter {
  constructor() {
    this.events = {};
  }

  on(event, listener) {
    if (!this.events[event]) {
      this.events[event] = [];
    }
    this.events[event].push(listener);
    return this;
  }

  off(event, listener) {
    if (!this.events[event]) return this;

    this.events[event] = this.events[event].filter(l => l !== listener);
    return this;
  }

  emit(event, ...args) {
    if (!this.events[event]) return this;

    this.events[event].forEach(listener => {
      listener.apply(this, args);
    });
    return this;
  }

  once(event, listener) {
    const onceListener = (...args) => {
      listener.apply(this, args);
      this.off(event, onceListener);
    };

    this.on(event, onceListener);
    return this;
  }
}

// Usage
const emitter = new EventEmitter();

emitter.on('userLoggedIn', (user) => {
  console.log(`${user.name} logged in`);
});

emitter.emit('userLoggedIn', { name: 'John', id: 123 });
```

Error Handling and Debugging

Custom Error Types

javascript

```javascript
class ValidationError extends Error {
  constructor(message, field) {
    super(message);
    this.name = 'ValidationError';
    this.field = field;
  }
}

class AuthenticationError extends Error {
  constructor(message, userId) {
    super(message);
    this.name = 'AuthenticationError';
    this.userId = userId;
    this.date = new Date();
  }
}

// Usage
function validateUser(user) {
  if (!user.email) {
    throw new ValidationError('Email is required', 'email');
  }

  if (!user.password || user.password.length < 6) {
    throw new ValidationError('Password must be at least 6 characters', 'password');
  }
}

try {
  validateUser({ email: 'test@example.com', password: '123' });
} catch (error) {
  if (error instanceof ValidationError) {
    console.error(`Validation failed for ${error.field}: ${error.message}`);
  } else {
    console.error('Unknown error:', error);
  }
}
```

Debugging Techniques

javascript

```javascript
// Using console methods
console.log('Basic logging');
console.error('Error message');
console.warn('Warning message');
console.info('Informational message');

console.table([
  { name: 'John', age: 30 },
  { name: 'Jane', age: 28 }
]);

console.group('User Details');
console.log('Name: John Doe');
console.log('Role: Administrator');
console.groupEnd();

// Performance measurement
console.time('operation');
// ... some operation
console.timeEnd('operation');

// Conditional breakpoints in browser devtools
// In the browser, you can right-click on a line number and set a condition:
// e.g., i === 5

// Debugging with debugger statement
function problematicFunction(data) {
  for (let i = 0; i < data.length; i++) {
    if (data[i].isSpecial) {
      debugger; // Execution will pause here when devtools is open
      // inspect variables, call stack, etc.
    }
    // process data
  }
}
```

Web APIs

Fetch API

javascript

Copy

```javascript
// Basic GET request
fetch('https://api.example.com/data')
  .then(response => {
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    return response.json();
  })
  .then(data => {
    console.log('Data received:', data);
  })
  .catch(error => {
    console.error('Fetch error:', error);
  });

// POST request with options
fetch('https://api.example.com/users', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer token123'
  },
  body: JSON.stringify({
    name: 'John Doe',
    email: 'john@example.com'
  })
})
.then(response => response.json())
.then(data => console.log('User created:', data));

// Using with async/await
async function fetchUserData(userId) {
  try {
    const response = await fetch(`https://api.example.com/users/${userId}`);

    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }

    const userData = await response.json();
    return userData;
  } catch (error) {
    console.error('Failed to fetch user data:', error);
    throw error;
  }
}
```

Local Storage

javascript Copy

```javascript
// Store data
localStorage.setItem('user', JSON.stringify({ name: 'John', id: 123 }));
localStorage.setItem('theme', 'dark');

// Retrieve data
const user = JSON.parse(localStorage.getItem('user'));
const theme = localStorage.getItem('theme');

// Remove specific item
localStorage.removeItem('theme');

// Clear all items
localStorage.clear();

// Storage event (triggered when storage changes in other tabs)
window.addEventListener('storage', (event) => {
  console.log('Storage changed:', event.key, event.oldValue, event.newValue);
});
```

Web Workers

javascript

```javascript
// main.js
const worker = new Worker('worker.js');

worker.postMessage({
  command: 'calculate',
  data: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
});

worker.onmessage = function(event) {
  console.log('Result from worker:', event.data);
};

worker.onerror = function(error) {
  console.error('Worker error:', error.message);
};

// worker.js
self.onmessage = function(event) {
  const { command, data } = event.data;

  if (command === 'calculate') {
    // Perform intensive calculation
    const result = data.map(num => num * num)
                       .filter(num => num > 10)
                       .reduce((sum, num) => sum + num, 0);

    // Send result back to main thread
    self.postMessage(result);
  }
};
```

Modern JavaScript Tools and Practices

ES Modules vs CommonJS

javascript

```javascript
// ES Modules (ESM)
// file.mjs or .js with "type": "module" in package.json
import { readFile } from 'fs/promises';
import lodash from 'lodash';

export function helper() {
  // ...
}

// CommonJS (Node.js)
// file.cjs or .js with default Node.js setup
const { readFile } = require('fs').promises;
const lodash = require('lodash');

function helper() {
  // ...
}

module.exports = { helper };
```

Performance Optimization

javascript

Copy

```javascript
// Debouncing
function debounce(func, wait) {
  let timeout;

  return function executedFunction(...args) {
    const later = () => {
      clearTimeout(timeout);
      func(...args);
    };

    clearTimeout(timeout);
    timeout = setTimeout(later, wait);
  };
}

// Usage
const debouncedSearch = debounce(function(searchTerm) {
  // API call or intensive operation
  console.log('Searching for:', searchTerm);
}, 300);

// Throttling
function throttle(func, limit) {
  let inThrottle;

  return function executedFunction(...args) {
    if (!inThrottle) {
      func(...args);
      inThrottle = true;
      setTimeout(() => {
        inThrottle = false;
      }, limit);
    }
  };
}

// Usage
const throttledScroll = throttle(function() {
  console.log('Scroll event handled');
}, 1000);

// Memoization
function memoize(fn) {
  const cache = new Map();
  return function(...args) {
    const key = JSON.stringify(args);

    if (cache.has(key)) {
      return cache.get(key);
    }

    const result = fn.apply(this, args);
    cache.set(key, result);
    return result;
  };
}

// Usage
const expensiveOperation = memoize(function(n) {
  console.log('Computing...');
  return n * n;
});
```

Testing

javascript

Copy

```javascript
// Basic unit test with Jest
// sum.js
function sum(a, b) {
  return a + b;
}
module.exports = sum;

// sum.test.js
const sum = require('./sum');

test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});

test('adds 0 + 0 to equal 0', () => {
  expect(sum(0, 0)).toBe(0);
});

test('adds negative numbers', () => {
  expect(sum(-1, -2)).toBe(-3);
});
```

Security Best Practices

Content Security Policy

javascript

Copy

```javascript
// Setting CSP in HTML
// <meta http-equiv="Content-Security-Policy" content="default-src 'self'; script-src

// Or in HTTP headers
// Content-Security-Policy: default-src 'self'; script-src 'self'

// Handling CSP violations
window.addEventListener('securitypolicyviolation', (event) => {
  console.error('CSP violation:', event.blockedURI, event.violatedDirective);
});
```

XSS Prevention

javascript

Copy

```javascript
// Unsafe
element.innerHTML = userInput; // DANGEROUS if userInput is untrusted

// Safe alternatives
element.textContent = userInput; // Safe for text
element.setAttribute('data-info', userInput); // Safe for attributes

// If you need to accept HTML, sanitize it
// Using DOMPurify library
import DOMPurify from 'dompurify';
element.innerHTML = DOMPurify.sanitize(userInput);
```

CSRF Protection

javascript

```javascript
// Include CSRF token in forms
/*
<form action="/api/submit" method="POST">
  <input type="hidden" name="csrf_token" value="random-token-here">
  <!-- other form fields -->
</form>
*/

// Include CSRF token in AJAX requests
fetch('/api/data', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'X-CSRF-Token': document.querySelector('meta[name="csrf-token"]').content
  },
  body: JSON.stringify(data)
});
```