COMPLETE WEB DEVELOPMENT NOTES

TABLE OF CONTENTS

---

1. INTRODUCTION TO WEB DEVELOPMENT

What is Web Development?

Web development refers to the creating, building, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management.

The Two Words:

- **Web**: Refers to websites, web pages, or anything that works over the internet
- **Development**: Building the application from scratch

Types of Web Development

Type	Description	Technologies
Frontend	User interface, what users see and interact with	HTML, CSS, JavaScript, React, Vue, Angular
Backend	Server-side logic, databases, APIs	Node.js, Python, PHP, Java, C#, Ruby
Full Stack	Both frontend and backend	Combination of above
DevOps	Deployment, hosting, infrastructure	Docker, AWS, CI/CD pipelines

Web Development Process

1. **Planning** - Define goals, target audience, features
2. **Design** - Create wireframes, mockups, prototypes
3. **Development** - Write code, build features
4. **Testing** - Debug, test functionality, cross-browser testing
5. **Deployment** - Launch to production
6. **Maintenance** - Updates, bug fixes, improvements

---

2. HTML (HYPERTEXT MARKUP LANGUAGE)

What is HTML?

HTML stands for **HyperText Markup Language**. It is the standard language used to create and design web pages on the internet. It was introduced by Tim Berners-Lee in 1991 at CERN.

Key Concepts:

- **Hypertext**: Links between web pages
- **Markup Language**: Text document within tags
- HTML provides the **structure/skeleton** of a website

Basic HTML Document Structure

```
html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Page Title</title>
</head>
<body>
    <h1>Main Heading</h1>
    <p>This is a paragraph.</p>
</body>
</html>
```

Common HTML Tags

Document Structure Tags:

Tag	Description
`<!DOCTYPE html>`	Declares HTML5 document type
`<html>`	Root element of HTML page
`<head>`	Contains metadata, title, links
`<title>`	Sets page title (browser tab)
`<body>`	Contains visible page content
`<meta>`	Provides metadata (charset, viewport, description)

Text Formatting Tags:

Tag	Description
`<h1>` to `<h6>`	Headings (h1 largest, h6 smallest)
`<p>`	Paragraph

Tag	Description
`<strong>`	Bold text (semantic - important)
`<em>`	Italic text (semantic - emphasis)
` `	Line break (self-closing)
`<hr>`	Horizontal rule (thematic break)
`<span>`	Inline container for styling
`<div>`	Block-level container

List Tags:

```
html
<!-- Unordered List (bullets) -->
<ul>
    <li>Item 1</li>
    <li>Item 2</li>
</ul>

<!-- Ordered List (numbers) -->
<ol>
    <li>First item</li>
    <li>Second item</li>
</ol>

<!-- Description List -->
<dl>
    <dt>Term</dt>
    <dd>Description</dd>
</dl>
```

Link and Image Tags:

```
html
<!-- Anchor (link) -->
<a href="https://www.example.com">Click here</a>
<a href="about.html" target="_blank">About Page (opens new tab)</a>
```

```html
<!-- Image -->
<img src="image.jpg" alt="Description of image" width="300" height="200">
```

Table Tags:

```
html
<table border="1">
    <thead>
        <tr>
            <th>Header 1</th>
            <th>Header 2</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>Row 1, Cell 1</td>
            <td>Row 1, Cell 2</td>
        </tr>
    </tbody>
</table>
```

Form Tags:

```
html
<form action="/submit" method="POST">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email">

    <label for="password">Password:</label>
    <input type="password" id="password" name="password">

    <label for="message">Message:</label>
    <textarea id="message" rows="4" cols="50"></textarea>

    <select name="country">
        <option value="us">United States</option>
        <option value="uk">United Kingdom</option>
    </select>

    <input type="submit" value="Submit">
</form>
```

Semantic HTML

Semantic elements clearly describe their meaning to both browser and developer.

Semantic Tag	Description
`<header>`	Introductory content or navigation links
`<nav>`	Navigation links section
`<main>`	Main content of the document
`<section>`	Thematic grouping of content
`<article>`	Self-contained composition
`<aside>`	Sidebar content (tangentially related)
`<footer>`	Footer content (copyright, contact)
`<figure>`	Illustrations, diagrams, photos
`<figcaption>`	Caption for figure

HTML Attributes

- `id` - Unique identifier for an element
- `class` - Class name for CSS styling (can be multiple)
- `src` - Source URL for images, scripts, iframes
- `href` - Hyperlink reference for anchor tags
- `alt` - Alternative text for images (accessibility)
- `title` - Tooltip text
- `style` - Inline CSS styles
- `target` - Where to open link (`_blank`, `_self`, `_parent`, `_top`)

---

3. CSS (CASCADING STYLE SHEETS)

What is CSS?

CSS (Cascading Style Sheets) is used to style and layout web pages — to alter the font, color, size, and spacing of content, split it into multiple columns, or add animations and other decorative features.

Three Ways to Add CSS:

1. **Inline CSS** - Inside HTML elements using `style` attribute
2. **Internal CSS** - Inside `<style>` tag in `<head>` section
3. **External CSS** - Separate `.css` file linked via `<link>` tag

```html
<!-- Inline CSS -->
<p style="color: red;">Red text</p>

<!-- Internal CSS -->
<style>
    p { color: blue; }
</style>

<!-- External CSS -->
<link rel="stylesheet" href="styles.css">
```

CSS Syntax

```css
selector {
    property: value;
    property: value;
}
```

CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

Selector	Syntax	Example	Description
Element	`element`	`p`	Selects all `<p>` elements
ID	`#id`	`#header`	Selects element with id="header"

Selector	Syntax	Example	Description
Class	`.class`	`.btn`	Selects all elements with class="btn"
Universal	`*`	`*`	Selects all elements
Grouping	`,`	`h1, h2, p`	Selects multiple elements
Descendant	(space)	`div p`	Selects `<p>` inside `<div>`
Child	`>`	`ul > li`	Direct child only
Adjacent	`+`	`h1 + p`	Element immediately after h1
General	`~`	`h1 ~ p`	All siblings after h1

Attribute Selectors:

```css
input[type="text"] { width: 100%; }
a[target="_blank"] { color: red; }
```

Pseudo-classes:

```css
/* Link states */
a:link { color: blue; }      /* Unvisited link */
a:visited { color: purple; } /* Visited link */
a:hover { color: red; }      /* Mouse over */
a:active { color: green; }   /* Clicking */

/* Input states */
input:focus { border-color: blue; }
input:checked { background: lightblue; }

/* Structural */
p:first-child { font-weight: bold; }
p:last-child { margin-bottom: 0; }
li:nth-child(odd) { background: #f0f0f0; }
```

Pseudo-elements:

```css
```

```css
p::first-line { font-weight: bold; }
p::first-letter { font-size: 200%; }
h1::before { content: "★ "; }
h1::after { content: " ★"; }
```

CSS Colors

```css
/* Named colors */
color: red; blue; green; white; black;

/* Hexadecimal */
color: #FF0000;  /* Red */
color: #00FF00;  /* Green */
color: #0000FF;  /* Blue */
color: #FFFFFF;  /* White */
color: #000000;  /* Black */

/* RGB */
color: rgb(255, 0, 0);
color: rgba(255, 0, 0, 0.5);  /* 50% opacity */

/* HSL */
color: hsl(0, 100%, 50%);
color: hsla(0, 100%, 50%, 0.5);
```

CSS Box Model

Every element on a web page is a rectangular box.

```text
+-----------------------------------+
|            MARGIN (transparent)   |
|  +-----------------------------+  |
|  |         BORDER              |  |
|  |  +-----------------------+  |  |
|  |  |      PADDING          |  |  |
|  |  |  +-----------------+  |  |  |
|  |  |  |    CONTENT      |  |  |  |
|  |  |  +-----------------+  |  |  |
|  |  +-----------------------+  |  |
|  +-----------------------------+  |
```

```
+-----------------------------------+
```

Box Model Properties:

```
css
.element {
    width: 300px;
    height: 200px;
    padding: 20px;           /* Space inside border */
    border: 2px solid black; /* Border around padding */
    margin: 30px;            /* Space outside border */
    box-sizing: border-box;  /* Includes padding/border in width */
}
```

CSS Typography (Text Styling)

```
css
body {
    font-family: 'Arial', sans-serif;
    font-size: 16px;
    font-weight: 400;        /* 100-900, normal=400, bold=700 */
    font-style: normal;      /* normal, italic, oblique */
    line-height: 1.5;        /* Space between lines */
    text-align: left;        /* left, center, right, justify */
    text-decoration: none;   /* none, underline, overline, line-through */
    text-transform: none;    /* uppercase, lowercase, capitalize */
    letter-spacing: 0.5px;
    word-spacing: 2px;
    color: #333;
}

/* Custom fonts */
@import url('https://fonts.googleapis.com/css2?family=Open+Sans&display=swap');
@font-face {
    font-family: 'MyFont';
    src: url('myfont.woff2') format('woff2');
}
```

CSS Layout

Display Property:

Value	Description
`block`	Takes full width, starts new line (div, p, h1)
`inline`	Takes only needed width, no line break (span, a)
`inline-block`	Inline but can set width/height
`none`	Hides element (removed from layout)
`flex`	Enables flexbox layout
`grid`	Enables CSS grid layout

Position Property:

Value	Description
`static`	Default, follows normal flow
`relative`	Positioned relative to normal position
`absolute`	Positioned relative to nearest positioned ancestor
`fixed`	Positioned relative to viewport (stays on scroll)
`sticky`	Switches between relative and fixed

CSS Flexbox (Modern Layout):

```
css
.container {
    display: flex;
    flex-direction: row;        /* row, column, row-reverse */
    justify-content: center;    /* flex-start, center, space-between, space-around
*/
    align-items: center;        /* flex-start, center, stretch */
    flex-wrap: wrap;            /* nowrap, wrap */
    gap: 20px;                  /* Space between items */
}
```

```css
.item {
    flex: 1;                    /* Grow equally */
    order: 2;                   /* Change order */
    align-self: flex-start;     /* Override container alignment */
}
```

CSS Grid:

```
css
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);  /* 3 equal columns */
    grid-template-rows: auto 200px auto;
    gap: 20px;
}

.grid-item {
    grid-column: span 2;        /* Span 2 columns */
    grid-row: 1 / 3;            /* Rows 1 to 3 */
}
```

CSS Animations

```
css
/* Transitions */
.button {
    background: blue;
    transition: background 0.3s ease, transform 0.2s;
}
.button:hover {
    background: red;
    transform: scale(1.05);
}

/* Keyframe Animations */
@keyframes slideIn {
    from { transform: translateX(-100px); opacity: 0; }
    to { transform: translateX(0); opacity: 1; }
}

.element {
    animation: slideIn 0.5s ease-out forwards;
}
```

CSS Media Queries (Responsive Design)

```css
/* Mobile first approach */
.container {
    width: 100%;
}

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

/* Desktop */
@media (min-width: 1024px) {
    .container {
        width: 960px;
    }
}

/* Print styles */
@media print {
    .no-print {
        display: none;
    }
}
```

---

4. JAVASCRIPT

What is JavaScript?

JavaScript is a lightweight, cross-platform, single-threaded, interpreted programming language. It is well-known for the development of web pages, providing dynamic behavior to websites.

Key Features:

- Client-side scripting language
- Weakly typed (dynamically typed)
- Both imperative and declarative
- Event-driven
- Supports object-oriented and functional programming

Adding JavaScript to HTML

```html
<!-- Inline (not recommended) -->
<button onclick="alert('Hello')">Click</button>

<!-- Internal -->
<script>
    console.log("Hello World");
</script>

<!-- External (recommended) -->
<script src="script.js"></script>
```

JavaScript Variables

```javascript
// var (function-scoped, avoid using)
var oldWay = "Don't use this";

// let (block-scoped, can be reassigned)
let name = "John";
name = "Jane";  // Allowed

// const (block-scoped, cannot be reassigned)
const PI = 3.14159;
// PI = 3.14;  // Error!
```

Data Types

```javascript
// Primitive Types
let string = "Hello";
let number = 42;
let boolean = true;
```

```javascript
let undefinedValue;           // undefined
let nullValue = null;         // null
let symbol = Symbol('id');    // Symbol

// Reference Types
let array = [1, 2, 3];
let object = { name: "John", age: 30 };
let function = function() { return "Hello"; };
```

Operators

```javascript
// Arithmetic
+ - * / % ** ++ --

// Assignment
= += -= *= /= %=

// Comparison
==  // Equal (value only)
=== // Strict equal (value and type)
!=  // Not equal
!== // Strict not equal
> < >= <=

// Logical
&&  // AND
||  // OR
!   // NOT

// Ternary
condition ? valueIfTrue : valueIfFalse
```

Control Flow

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

```
} else {
    // code
}

// Switch
switch (value) {
    case 'a':
        // code
        break;
    case 'b':
        // code
        break;
    default:
        // code
}

// Loops
for (let i = 0; i < 10; i++) { }
while (condition) { }
do { } while (condition);
for (let item of array) { }     // Iterate values
for (let key in object) { }     // Iterate properties
```

Functions

javascript

```
// Function declaration
function greet(name) {
    return `Hello, ${name}!`;
}

// Function expression
const greet = function(name) {
    return `Hello, ${name}!`;
};

// Arrow function (ES6+)
const greet = (name) => `Hello, ${name}!`;

// Default parameters
function greet(name = "Guest") {
    return `Hello, ${name}!`;
```

```
}

// Rest parameters
function sum(...numbers) {
    return numbers.reduce((a, b) => a + b, 0);
}
```

Arrays

```
javascript
// Creation
let arr = [1, 2, 3];
let arr2 = new Array(5);

// Common methods
arr.push(4);           // Add to end
arr.pop();             // Remove from end
arr.unshift(0);        // Add to beginning
arr.shift();           // Remove from beginning
arr.indexOf(2);        // Find index
arr.includes(3);       // Check existence
arr.slice(1, 3);       // Extract portion
arr.splice(1, 1);      // Remove/replace elements
arr.map(x => x * 2);   // Transform each element
arr.filter(x => x > 2); // Filter elements
arr.reduce((a, b) => a + b); // Reduce to single value
arr.forEach(x => console.log(x)); // Iterate
arr.sort((a, b) => a - b); // Sort
arr.reverse();         // Reverse order
```

Objects

```
javascript
// Object literal
const person = {
    name: "John",
    age: 30,
    greet() {
        return `Hello, I'm ${this.name}`;
    }
};
```

```javascript
// Access properties
person.name;        // Dot notation
person["name"];     // Bracket notation

// Adding properties
person.email = "john@example.com";

// Object methods
Object.keys(person);    // Get keys array
Object.values(person);  // Get values array
Object.entries(person); // Get [key, value] pairs
```

DOM Manipulation

```javascript
// Selecting elements
document.getElementById('id');
document.getElementsByClassName('class');
document.getElementsByTagName('div');
document.querySelector('.class');      // First match
document.querySelectorAll('.class');   // All matches

// Creating elements
const div = document.createElement('div');
div.textContent = "Hello";
div.innerHTML = "<strong>Hello</strong>";
div.classList.add('my-class');
div.setAttribute('data-id', '123');

// Manipulating DOM
parent.appendChild(child);
parent.removeChild(child);
element.remove();
element.style.color = 'red';
element.style.backgroundColor = 'blue';

// Events
element.addEventListener('click', () => {
    console.log('Clicked!');
});
```

```
// Common events: click, mouseover, keydown, submit, load, change
```

Events

javascript

```
// Mouse Events
onclick, ondblclick, onmouseover, onmouseout, onmousemove

// Keyboard Events
onkeydown, onkeyup, onkeypress

// Form Events
onsubmit, onchange, onfocus, onblur, oninput

// Window Events
onload, onresize, onscroll

// Adding event listeners
button.addEventListener('click', function(event) {
    console.log(event.target);  // Element that triggered event
    event.preventDefault();     // Prevent default behavior
    event.stopPropagation();    // Stop event bubbling
});
```

ES6+ Features

javascript

```
// Template literals
const name = "John";
const greeting = `Hello, ${name}!`;  // Backticks

// Destructuring
const [a, b] = [1, 2];
const { name, age } = person;

// Spread operator
const arr1 = [1, 2];
const arr2 = [...arr1, 3, 4];
const obj2 = { ...person, city: "NYC" };

// Classes
```

```
class Person {
    constructor(name) {
        this.name = name;
    }
    greet() {
        return `Hello, ${this.name}`;
    }
}

// Promises
fetch('https://api.example.com/data')
    .then(response => response.json())
    .then(data => console.log(data))
    .catch(error => console.error(error));

// Async/Await
async function getData() {
    try {
        const response = await fetch('https://api.example.com/data');
        const data = await response.json();
        console.log(data);
    } catch (error) {
        console.error(error);
    }
}
```

JavaScript Best Practices

- Use `const` by default, `let` when you need to reassign
- Avoid `var`
- Use strict equality `===` instead of `==`
- Name variables and functions meaningfully
- Keep functions small and focused
- Handle errors with try/catch
- Comment complex code
- Use modern ES6+ syntax

---

5. FRONTEND FRAMEWORKS & LIBRARIES

What are Frameworks?

JavaScript frameworks are an essential part of modern front-end web development, providing developers with tried and tested tools for building scalable, interactive web applications. Many modern companies use frameworks as a standard part of their tooling.

Popular Frameworks

Framework	Developer	Key Features	Learning Curve
React	Meta	Virtual DOM, JSX, Hooks, Component-based	Medium
Vue.js	Evan You	Progressive, Easy to learn, Two-way binding	Low
Angular	Google	Full-featured, TypeScript, Dependency injection	High
Svelte	Rich Harris	Compiler-based, No virtual DOM	Medium

Bootstrap (CSS Framework)

Bootstrap is a popular CSS framework for faster and easier web development, especially for creating responsive designs.

```
html
<!-- Include Bootstrap -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/css/bootstrap.min.css" re
l="stylesheet">

<!-- Responsive grid -->
<div class="container">
    <div class="row">
        <div class="col-md-6">Left column</div>
        <div class="col-md-6">Right column</div>
    </div>
```

```
</div>

<!-- Components -->
<button class="btn btn-primary">Primary Button</button>
<div class="card">
    <div class="card-body">Card content</div>
</div>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark"></nav>
```

---

6. BACKEND DEVELOPMENT

What is Backend?

Backend development handles server-side logic, database interactions, authentication, APIs, and business logic that users don't directly see.

Popular Backend Technologies

Technology	Language	Best For
Node.js	JavaScript	Real-time apps, APIs, Full-stack JS
Django	Python	Rapid development, Data-driven apps
Flask	Python	Microservices, Small projects
Spring Boot	Java	Enterprise applications
ASP.NET Core	C#	Microsoft ecosystem
Laravel	PHP	Web applications
Ruby on Rails	Ruby	Startups, Prototypes

REST API Basics

```javascript
// GET - Retrieve data
fetch('/api/users')
    .then(response => response.json());

// POST - Create data
fetch('/api/users', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ name: "John", email: "john@example.com" })
});

// PUT - Update data
fetch('/api/users/1', {
    method: 'PUT',
    body: JSON.stringify({ name: "John Updated" })
});

// DELETE - Remove data
fetch('/api/users/1', { method: 'DELETE' });
```

HTTP Status Codes

Code Range	Category	Examples
2xx	Success	200 OK, 201 Created
3xx	Redirection	301 Moved, 304 Not Modified
4xx	Client Error	400 Bad Request, 401 Unauthorized, 404 Not Found
5xx	Server Error	500 Internal Server Error, 503 Service Unavailable

---

7. DATABASES

Types of Databases

SQL (Relational Databases)

- Structured data with predefined schemas
- Uses tables with rows and columns
- Uses SQL (Structured Query Language)

Database	Best For
MySQL	Web applications, WordPress
PostgreSQL	Complex queries, Data integrity
SQLite	Embedded, Mobile apps
Microsoft SQL Server	Enterprise, .NET apps

NoSQL (Non-Relational Databases)

- Flexible schemas
- Horizontal scaling
- Better for large volumes of unstructured data

Type	Examples	Use Cases
Document	MongoDB, CouchDB	Content management, Catalogs
Key-Value	Redis, DynamoDB	Caching, Session storage
Graph	Neo4j	Social networks, Recommendations
Column	Cassandra, HBase	Analytics, Time-series data

Basic SQL Commands

```sql
-- SELECT (Read)
SELECT * FROM users;
SELECT name, email FROM users WHERE age > 18;

-- INSERT (Create)
INSERT INTO users (name, email) VALUES ('John', 'john@example.com');
```

```sql
-- UPDATE (Modify)
UPDATE users SET email = 'new@example.com' WHERE id = 1;

-- DELETE (Remove)
DELETE FROM users WHERE id = 1;

-- JOIN tables
SELECT users.name, orders.total
FROM users
JOIN orders ON users.id = orders.user_id;
```

---

8. VERSION CONTROL (GIT & GITHUB)

What is Version Control?

Version control tools are an essential part of modern workflows, for backing up and collaborating on codebases.

Basic Git Commands

bash

```bash
# Configuration
git config --global user.name "Your Name"
git config --global user.email "your@email.com"

# Initialize repository
git init

# Cloning
git clone https://github.com/user/repo.git

# Basic workflow
git status                    # Check current state
git add filename              # Stage file
git add .                     # Stage all files
git commit -m "Message"       # Commit changes
git push                      # Push to remote
git pull                      # Pull from remote
```

```
# Branching
git branch                    # List branches
git branch new-feature        # Create branch
git checkout new-feature      # Switch branch
git checkout -b new-feature   # Create and switch
git merge new-feature         # Merge branch

# History
git log                       # View commit history
git diff                      # See changes
git reset --hard HEAD         # Discard changes
```

Git Workflow (GitFlow)

1. `main` branch - Production-ready code
2. `develop` branch - Integration branch
3. Feature branches - New features
4. Release branches - Prepare releases
5. Hotfix branches - Emergency fixes

---

9. WEB PERFORMANCE & SEO

Web Performance Best Practices

- Optimize images (compress, use WebP format)
- Minify CSS, JavaScript, HTML
- Enable compression (Gzip, Brotli)
- Use browser caching
- Load CSS in `<head>`, JS at end of `<body>`
- Use CDN (Content Delivery Network)
- Lazy load images
- Reduce HTTP requests
- Use critical CSS inline

Core Web Vitals (Google Metrics)

Metric	Description	Good Score
LCP	Largest Contentful Paint (loading)	< 2.5s
FID	First Input Delay (interactivity)	< 100ms
CLS	Cumulative Layout Shift (visual stability)	< 0.1

SEO (Search Engine Optimization)

On-Page SEO:

```
html
<!-- Title tag (50-60 characters) -->
<title>Web Development Notes - Complete Guide 2024</title>

<!-- Meta description (150-160 characters) -->
<meta name="description" content="Comprehensive web development notes covering HTM
L, CSS, JavaScript, frameworks, and best practices.">

<!-- Heading structure (one H1 per page) -->
<h1>Main Topic</h1>
<h2>Subtopic</h2>
<h3>Details</h3>

<!-- Semantic HTML -->
<article>, <section>, <nav>, <header>, <footer>

<!-- Image alt text -->
<img src="image.jpg" alt="Descriptive text">

<!-- Canonical URL -->
<link rel="canonical" href="https://example.com/page">

<!-- Open Graph (Social Media) -->
<meta property="og:title" content="Page Title">
<meta property="og:description" content="Page description">
<meta property="og:image" content="thumbnail.jpg">
```

Technical SEO:

- Use HTTPS (SSL certificate)
- Create XML sitemap
- robots.txt file
- Mobile-friendly design
- Fast page speed
- Clean URL structure
- Structured data (Schema.org)

---

10. WEB SECURITY

Common Vulnerabilities (OWASP Top 10)

1. Injection (SQLi, XSS, Command Injection)

```javascript
// Vulnerable code
const query = `SELECT * FROM users WHERE email = '${userEmail}'`;

// Secure (parameterized query)
const query = 'SELECT * FROM users WHERE email = ?';
db.query(query, [userEmail]);
```

2. Cross-Site Scripting (XSS)

```html
<!-- Vulnerable -->
<div>Hello, {{ userInput }}</div>

<!-- Secure -->
<div>Hello, {{ escapeHtml(userInput) }}</div>
```

3. Cross-Site Request Forgery (CSRF)

- Use anti-CSRF tokens
- SameSite cookie attribute
- Verify referer header

4. Broken Authentication

- Implement MFA
- Strong password policies

- Secure session management
- Account lockout after failed attempts

Security Headers

```
html
<!-- HTTP Security Headers -->
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
Content-Security-Policy: default-src 'self'
Strict-Transport-Security: max-age=31536000
Referrer-Policy: strict-origin-when-cross-origin
```

HTTPS & SSL/TLS

- Encrypts data in transit
- Required for modern web features
- Boosts SEO ranking
- Builds user trust

Security Best Practices

- Validate all user input
- Sanitize output
- Use prepared statements for database queries
- Hash passwords (bcrypt, argon2)
- Implement rate limiting
- Keep dependencies updated
- Use environment variables for secrets
- Regular security audits
- Backup data regularly

---

11. ACCESSIBILITY

What is Web Accessibility?

Access to web content such as public services, education, e-commerce sites, and entertainment is a human right. No one should be excluded based on disability, race, geography, or other human characteristics.

WCAG Principles (POUR)

Principle	Description
Perceivable	Users can perceive content (alt text, captions)
Operable	Users can operate interface (keyboard navigation)
Understandable	Users can understand content (clear language)
Robust	Content works with assistive technologies

Accessibility Best Practices

```html
<!-- Alternative text for images -->
<img src="chart.png" alt="Sales chart showing 20% growth">

<!-- Semantic HTML over divs -->
<button>Click me</button>  <!-- Better than <div onclick=""> -->
<nav>...</nav>             <!-- Better than <div class="nav"> -->

<!-- ARIA attributes (when HTML isn't enough) -->
<div role="button" tabindex="0" aria-label="Close menu">X</div>

<!-- Focus management -->
<a href="#main" class="skip-link">Skip to main content</a>

<!-- Color contrast -->
<!-- Minimum contrast ratio: 4.5:1 for normal text -->

<!-- Form labels -->
<label for="name">Name:</label>
<input id="name" type="text">

<!-- Keyboard navigation -->
```

```
<!-- All interactive elements should be reachable with Tab key -->
```

Accessibility Testing Tools

- WAVE (Web Accessibility Evaluation Tool)
- Lighthouse (Chrome DevTools)
- axe DevTools
- Screen readers (NVDA, VoiceOver, JAWS)

---

12. RESPONSIVE DESIGN

What is Responsive Design?

Designing websites that work well on different devices, screen sizes, and resolutions.

Viewport Meta Tag

```
html
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Responsive Units

Unit	Description
`%`	Percentage of parent
`vw`	Viewport width (1vw = 1% of viewport)
`vh`	Viewport height
`rem`	Relative to root font size (16px default)
`em`	Relative to parent font size
`fr`	Fractional unit (CSS Grid)

Responsive Techniques

```css
/* Fluid images */
img {
    max-width: 100%;
    height: auto;
}

/* Fluid typography */
body {
    font-size: clamp(16px, 4vw, 24px);
}

/* Flexbox for responsive layouts */
.container {
    display: flex;
    flex-wrap: wrap;
}
.item {
    flex: 1 1 300px;  /* Grow, shrink, basis */
}

/* CSS Grid for responsive grids */
.grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
}

/* Mobile-first media queries */
/* Mobile (default) */
.container { width: 100%; }

/* Tablet */
@media (min-width: 768px) {
    .container { width: 750px; }
}

/* Desktop */
@media (min-width: 1024px) {
    .container { width: 960px; }
}
```

```css
/* Orientation */
@media (orientation: landscape) { }
@media (orientation: portrait) { }
```

---

13. DEPLOYMENT & HOSTING

Hosting Options

Type	Examples	Best For
Shared Hosting	Bluehost, HostGator	Small websites, Blogs
VPS	DigitalOcean, Linode	Medium traffic
Cloud Hosting	AWS, Google Cloud, Azure	Scalable applications
Static Hosting	Netlify, Vercel, GitHub Pages	Static sites, JAMstack
Platform as a Service	Heroku, Render	Full-stack apps

Deployment Process

1. Build/compile your application
2. Configure environment variables
3. Set up database (if needed)
4. Upload files (FTP, Git, or CLI)
5. Configure domain and SSL
6. Test production environment

CI/CD (Continuous Integration/Deployment)

- Automatically test code changes
- Automatically deploy when tests pass
- Tools: GitHub Actions, GitLab CI, Jenkins

---

14. EXAM CHEAT SHEET

Quick Reference

Frontend Technologies:

- HTML = Structure/Skeleton
- CSS = Style/Presentation
- JavaScript = Behavior/Interactivity

CSS Box Model (inside to outside):
CONTENT → PADDING → BORDER → MARGIN

CSS Selectors Priority:
Inline Style > ID > Class/Attribute > Element > Universal

JavaScript Data Types:

- String, Number, Boolean, Undefined, Null, Symbol (Primitive)
- Object, Array, Function (Reference)

HTTP Methods:

- GET = Read
- POST = Create
- PUT/PATCH = Update
- DELETE = Delete

HTTP Status Codes:

- 2xx = Success
- 3xx = Redirect
- 4xx = Client Error
- 5xx = Server Error

Git Commands:

```bash
git init          # Initialize
git add .         # Stage all
git commit -m ""  # Commit
git push          # Upload
```

```
git pull          # Download
git branch        # List branches
git checkout -b   # Create & switch branch
```

Responsive Design Breakpoints:

- Mobile: < 768px
- Tablet: 768px - 1024px
- Desktop: > 1024px

Security Headers:

- CSP = Content Security Policy
- HSTS = HTTP Strict Transport Security
- X-Frame-Options = Prevent clickjacking

SEO Meta Tags:

- Title = Page title (50-60 chars)
- Description = Page description (150-160 chars)
- Keywords = (Less important now)

Accessibility (WCAG):

- POUR = Perceivable, Operable, Understandable, Robust
- Alt text for images
- Semantic HTML
- Keyboard navigation
- Color contrast (4.5:1)

---

15. LEARNING RESOURCES

Free Resources

- **MDN Web Docs** - Comprehensive documentation
- **W3Schools** - Interactive tutorials
- **freeCodeCamp** - Full curriculum with projects
- **The Odin Project** - Full-stack curriculum
- **JavaScript.info** - Modern JavaScript tutorial

Tools

- **Code Editors**: VS Code, Sublime Text, Atom
- **Browser DevTools**: Chrome DevTools, Firefox Developer Tools
- **Design**: Figma, Adobe XD
- **Version Control**: Git, GitHub, GitLab
- **Testing**: Jest, Mocha, Cypress
- **Performance**: Lighthouse, PageSpeed Insights

Practice Platforms

- LeetCode
- Codewars
- Frontend Mentor
- CodePen
- GitHub (open source contributions)

END OF NOTES

These notes cover the essential topics for web development. For advanced topics, refer to specialized resources.