Ultimate Web Development Cheatsheet

JavaScript

Core Syntax

```
// Variables
let variable = 'value';              // Block scope, can reassign
const constant = 'value';            // Block scope, cannot reassign
var oldVariable = 'value';           // Function scope, can reassign

// Data Types
const string = 'text';               // String
const number = 42;                   // Number
const decimal = 3.14;                // Number (floating point)
const boolean = true;                // Boolean
const nullValue = null;              // Null
const undefinedValue = undefined;    // Undefined
const symbolValue = Symbol('desc');  // Symbol
const bigIntValue = 9007199254740991n; // BigInt

// String operations
const template = `Value: ${variable}`;  // Template literal
const concat = 'Hello' + ' World';      // Concatenation
const sliced = string.slice(0, 4);      // Slice string
const upper = string.toUpperCase();     // Uppercase
const lower = string.toLowerCase();     // Lowercase
const replaced = string.replace('x', 'y'); // Replace first occurrence
const replacedAll = string.replaceAll('x', 'y'); // Replace all occurrences
const splitted = string.split(',');     // Split to array
const trimmed = string.trim();          // Remove whitespace
const includes = string.includes('x');  // Check if contains substring
const startsWith = string.startsWith('x'); // Check if starts with
const endsWith = string.endsWith('x');  // Check if ends with
const indexOf = string.indexOf('x');    // Find position of substring
```

```
// Number operations
const rounded = Math.round(3.14);       // Round to nearest integer
const floored = Math.floor(3.14);       // Round down
const ceiled = Math.ceil(3.14);         // Round up
const parsed = parseInt('42', 10);      // Parse string to integer
const floatParsed = parseFloat('3.14'); // Parse string to float
const fixed = number.toFixed(2);        // Format with 2 decimal places
const exponential = number.toExponential(2); // Exponential notation
const isNan = isNaN(value);             // Check if NaN

// Operators
const sum = 1 + 2;                    // Addition
const difference = 5 - 3;             // Subtraction
const product = 4 * 2;                // Multiplication
const quotient = 10 / 2;              // Division
const remainder = 10 % 3;             // Modulo
const power = 2 ** 3;                 // Exponentiation
const increment = ++variable;         // Pre-increment
const decrement = --variable;         // Pre-decrement
const postIncrement = variable++;     // Post-increment
const postDecrement = variable--;     // Post-decrement

// Comparison operators
const equal = x == y;                 // Equal (coerces types)
const strictEqual = x === y;          // Strictly equal (no type coercion)
const notEqual = x != y;              // Not equal
const strictNotEqual = x !== y;       // Strictly not equal
const greater = x > y;                // Greater than
const less = x < y;                   // Less than
const greaterEqual = x >= y;          // Greater than or equal
const lessEqual = x <= y;             // Less than or equal

// Logical operators
const and = x && y;                   // Logical AND
const or = x || y;                    // Logical OR
const not = !x;                       // Logical NOT
const nullish = x ?? y;               // Nullish coalescing
```

Control Flow

```
// Conditionals
if (condition) {
  // code
} else if (otherCondition) {
  // code
} else {
  // code
}

// Ternary operator
const result = condition ? valueIfTrue : valueIfFalse;

// Switch statement
switch (value) {
  case 'option1':
    // code
    break;
  case 'option2':
    // code
    break;
  default:
    // default code
    break;
}

// Loops
for (let i = 0; i < 10; i++) {
  // code runs 10 times
  if (condition) continue; // Skip to next iteration
  if (condition) break;    // Exit loop
}

while (condition) {
  // code runs while condition is true
}

do {
  // code runs at least once, then while condition is true
} while (condition);

// for...of loop (iterate over iterable values)
for (const item of array) {
```

```
  // code for each item
}

// for...in loop (iterate over object properties)
for (const key in object) {
  if (object.hasOwnProperty(key)) {
    // code for each own property
  }
}
```

Functions

```
// Function declaration
function functionName(param1, param2) {
  return result;
}

// Function expression
const functionName = function(param1, param2) {
  return result;
};

// Arrow function
const arrowFunction = (param1, param2) => {
  return result;
};

// Arrow function with implicit return
const shortArrow = (param1, param2) => result;

// Default parameters
function withDefaults(param1 = 'default', param2 = 42) {
  return result;
}

// Rest parameters
function withRest(param1, ...restParams) {
  // restParams is an array
  return result;
}
```

```
// Destructuring parameters
function withDestructuring({ name, age }) {
  // using name and age directly
  return result;
}

// Closures
function createCounter() {
  let count = 0;
  return function() {
    return ++count;
  };
}
const counter = createCounter();
counter(); // 1
counter(); // 2

// Immediately Invoked Function Expression (IIFE)
(function() {
  // code runs immediately
  const privateVar = 'cannot access outside';
})();

// Function methods
const obj = {
  value: 42,
  method() {
    return this.value;
  }
};
const boundFunction = obj.method.bind(obj);  // Bind 'this'
obj.method.call(otherObj, arg1, arg2);       // Call with 'this' and args
obj.method.apply(otherObj, [arg1, arg2]);    // Call with 'this' and array of args
```

Objects and Arrays

```
// Object creation
const obj = {
  property: value,
  method() {
    return this.property;
```

```
  },
  'complex key': value,
  [dynamicKey]: value  // Computed property
};

// Object methods
const keys = Object.keys(obj);                // Array of property names
const values = Object.values(obj);            // Array of values
const entries = Object.entries(obj);          // Array of [key, value] pairs
const merged = Object.assign({}, obj1, obj2); // Merge objects
const hasOwn = Object.hasOwn(obj, 'prop');    // Check own property
const frozen = Object.freeze(obj);            // Make immutable
const sealed = Object.seal(obj);              // Prevent add/delete props
const descriptor = Object.getOwnPropertyDescriptor(obj, 'prop'); // Get prop config

// Object spread
const clone = { ...obj };                     // Shallow clone
const mergedSpread = { ...obj1, ...obj2 };    // Merge objects

// Property descriptors
Object.defineProperty(obj, 'prop', {
  value: 42,
  writable: true,       // Can be changed
  enumerable: true,     // Shows up in loops
  configurable: true    // Can be deleted/redefined
});

// Arrays
const arr = [1, 2, 3];
const empty = new Array(3);   // Empty array with 3 slots
const fromArgs = Array.of(1, 2, 3); // Array from arguments
const fromIterable = Array.from('123'); // Array from iterable

// Array methods
arr.push(4);               // Add to end
arr.pop();                 // Remove from end
arr.unshift(0);            // Add to beginning
arr.shift();               // Remove from beginning
arr.splice(1, 2, 'x');     // Remove/replace elements
arr.slice(1, 3);           // Get subarray (no mutation)
arr.concat([4, 5]);        // Combine arrays
arr.join('-');             // Join elements to string
arr.includes(2);           // Check if contains
```

```
arr.indexOf(2);            // Find index of element
arr.lastIndexOf(2);        // Find last index of element
arr.reverse();             // Reverse array (mutates)
arr.sort((a, b) => a - b); // Sort array (mutates)
arr.fill(0, 1, 3);         // Fill range with value

// Array iteration methods
arr.forEach((item, index, array) => { /* code */ });    // No return value
const mapped = arr.map(item => item * 2);               // Transform to new array
const filtered = arr.filter(item => item > 2);          // Keep matching items
const found = arr.find(item => item > 2);               // Find first match
const foundIndex = arr.findIndex(item => item > 2);     // Find first match index
const foundLast = arr.findLast(item => item > 2);       // Find last match
const foundLastIndex = arr.findLastIndex(item => item > 2); // Find last match inde
const reduced = arr.reduce((acc, item) => acc + item, 0); // Combine values
const reducedRight = arr.reduceRight((acc, item) => acc + item, 0); // Right-to-lef
const every = arr.every(item => item > 0);              // Check if all match
const some = arr.some(item => item > 2);                // Check if any match
const flat = [1, [2, 3]].flat();                        // Flatten nested arrays
const flatMapped = arr.flatMap(x => [x, x * 2]);        // Map + flatten

// Array destructuring
const [first, second, ...rest] = arr;
const [a = 'default', b] = arr;  // Default value
```

Promises and Async

```
// Creating promises
const promise = new Promise((resolve, reject) => {
  if (success) {
    resolve(value);
  } else {
    reject(error);
  }
});

// Consuming promises
promise
  .then(value => {
    // Handle success
    return nextValue;
```

```
  })
  .catch(error => {
    // Handle error
    return recovery;
  })
  .finally(() => {
    // Always executes
  });

// Chaining promises
promise
  .then(value1 => {
    return value2;
  })
  .then(value2 => {
    return value3;
  });

// Promise static methods
Promise.resolve(value);             // Create resolved promise
Promise.reject(error);              // Create rejected promise
Promise.all([promise1, promise2]);  // Wait for all promises (fails fast)
Promise.allSettled([p1, p2]);       // Wait for all promises (no fail fast)
Promise.race([promise1, promise2]); // First to resolve/reject
Promise.any([promise1, promise2]);  // First to resolve

// Async/await
async function asyncFunction() {
  try {
    const result = await promise;    // Wait for promise
    return processedResult;
  } catch (error) {
    // Handle errors
  }
}

// Sequential execution
async function sequential() {
  const result1 = await asyncFunction1();
  const result2 = await asyncFunction2(result1);
  return result2;
}
```

```
// Parallel execution
async function parallel() {
  const [result1, result2] = await Promise.all([
    asyncFunction1(),
    asyncFunction2()
  ]);
  return combineResults(result1, result2);
}

// Handling timeouts
const timeout = ms => new Promise(resolve => setTimeout(resolve, ms));

// Promise with timeout
const promiseWithTimeout = (promise, ms) => {
  const timeoutPromise = new Promise((_, reject) => {
    setTimeout(() => reject(new Error('Timeout')), ms);
  });
  return Promise.race([promise, timeoutPromise]);
};

// Async generators
async function* asyncGenerator() {
  yield await asyncOperation1();
  yield await asyncOperation2();
}

// Using async generators
async function useGenerator() {
  for await (const value of asyncGenerator()) {
    console.log(value);
  }
}
```

Events

```
// Adding event listeners
element.addEventListener('click', handleClick);
element.addEventListener('click', handleClick, {
  once: true,             // Run once then remove
  passive: true,          // Promise not to call preventDefault()
  capture: true           // Fire in capture phase
```

```
});

// Removing event listeners
element.removeEventListener('click', handleClick);

// Event handler with parameters
element.addEventListener('click', (event) => {
  handleClick(event, customArg);
});

// Event object properties
function handleEvent(event) {
  event.target;           // Element that triggered event
  event.currentTarget;    // Element that listener is attached to
  event.type;             // Event type (click, keydown, etc.)
  event.preventDefault(); // Prevent default action
  event.stopPropagation(); // Stop bubbling to parent elements
  event.stopImmediatePropagation(); // Stop other listeners on same element

  // Mouse event properties
  event.clientX;          // X coordinate relative to viewport
  event.clientY;          // Y coordinate relative to viewport
  event.pageX;            // X coordinate including scroll offset
  event.pageY;            // Y coordinate including scroll offset
  event.offsetX;          // X coordinate relative to target element
  event.offsetY;          // Y coordinate relative to target element
  event.button;           // Mouse button (0=left, 1=middle, 2=right)

  // Keyboard event properties
  event.key;              // Key value (e.g. "a", "Enter")
  event.code;             // Physical key code (e.g. "KeyA", "Enter")
  event.altKey;           // Whether Alt key was pressed
  event.ctrlKey;          // Whether Ctrl key was pressed
  event.shiftKey;         // Whether Shift key was pressed
  event.metaKey;          // Whether Meta key was pressed
  event.repeat;           // Whether key is being held down
}

// Common events
// Mouse events
element.addEventListener('click', handleClick);
element.addEventListener('dblclick', handleDoubleClick);
element.addEventListener('mousedown', handleMouseDown);
```

```
element.addEventListener('mouseup', handleMouseUp);
element.addEventListener('mousemove', handleMouseMove);
element.addEventListener('mouseover', handleMouseOver);
element.addEventListener('mouseout', handleMouseOut);
element.addEventListener('mouseenter', handleMouseEnter); // Doesn't bubble
element.addEventListener('mouseleave', handleMouseLeave); // Doesn't bubble
element.addEventListener('contextmenu', handleContextMenu);
element.addEventListener('wheel', handleWheel);

// Keyboard events
element.addEventListener('keydown', handleKeyDown);
element.addEventListener('keyup', handleKeyUp);
element.addEventListener('keypress', handleKeyPress);

// Form events
form.addEventListener('submit', handleSubmit);
input.addEventListener('input', handleInput);
input.addEventListener('change', handleChange);
input.addEventListener('focus', handleFocus);
input.addEventListener('blur', handleBlur);
input.addEventListener('reset', handleReset);

// Document/Window events
window.addEventListener('load', handleLoad);
window.addEventListener('DOMContentLoaded', handleDOMContentLoaded);
window.addEventListener('resize', handleResize);
window.addEventListener('scroll', handleScroll);
document.addEventListener('visibilitychange', handleVisibilityChange);
window.addEventListener('online', handleOnline);
window.addEventListener('offline', handleOffline);
window.addEventListener('beforeunload', handleBeforeUnload);
window.addEventListener('unload', handleUnload);
window.addEventListener('error', handleError);

// Drag and drop events
element.addEventListener('dragstart', handleDragStart);
element.addEventListener('drag', handleDrag);
element.addEventListener('dragenter', handleDragEnter);
element.addEventListener('dragleave', handleDragLeave);
element.addEventListener('dragover', handleDragOver);
element.addEventListener('drop', handleDrop);
element.addEventListener('dragend', handleDragEnd);
```

```
// Touch events
element.addEventListener('touchstart', handleTouchStart);
element.addEventListener('touchmove', handleTouchMove);
element.addEventListener('touchend', handleTouchEnd);
element.addEventListener('touchcancel', handleTouchCancel);

// Animation events
element.addEventListener('animationstart', handleAnimationStart);
element.addEventListener('animationiteration', handleAnimationIteration);
element.addEventListener('animationend', handleAnimationEnd);
element.addEventListener('transitionstart', handleTransitionStart);
element.addEventListener('transitionend', handleTransitionEnd);
element.addEventListener('transitionrun', handleTransitionRun);
element.addEventListener('transitioncancel', handleTransitionCancel);

// Clipboard events
element.addEventListener('copy', handleCopy);
element.addEventListener('cut', handleCut);
element.addEventListener('paste', handlePaste);

// Media events
media.addEventListener('play', handlePlay);
media.addEventListener('pause', handlePause);
media.addEventListener('ended', handleEnded);
media.addEventListener('volumechange', handleVolumeChange);
media.addEventListener('timeupdate', handleTimeUpdate);
media.addEventListener('loadeddata', handleLoadedData);

// Creating custom events
const customEvent = new CustomEvent('myEvent', {
  detail: { data: 'value' },
  bubbles: true,
  cancelable: true
});
element.dispatchEvent(customEvent);

// Event delegation
document.getElementById('parent').addEventListener('click', (event) => {
  if (event.target.matches('.child-selector')) {
    // Handle click on child element
  }
});
```

```
// Event propagation phases
// 1. Capture phase: Down from window to target
// 2. Target phase: Event reaches the target
// 3. Bubbling phase: Up from target to window

// Event listener options
{
  capture: false,     // Use capture phase (default: false)
  once: false,        // Remove after invocation (default: false)
  passive: false,     // Never calls preventDefault (default: false)
  signal: controller.signal // AbortController signal to remove listener
}
```

Mutators, Observers, and Advanced

```
// Mutation Observer
const observer = new MutationObserver((mutations) => {
  for (const mutation of mutations) {
    if (mutation.type === 'childList') {
      console.log('Children changed');
    } else if (mutation.type === 'attributes') {
      console.log(`${mutation.attributeName} changed`);
    }
  }
});

// Start observing
observer.observe(element, {
  childList: true,      // Observe child additions/removals
  attributes: true,     // Observe attribute changes
  characterData: true,  // Observe text content changes
  subtree: true,        // Observe all descendants
  attributeOldValue: true, // Record previous attribute values
  characterDataOldValue: true // Record previous text values
});

// Stop observing
observer.disconnect();

// Intersection Observer
const intersectionObserver = new IntersectionObserver((entries) => {
```

```
  entries.forEach(entry => {
    if (entry.isIntersecting) {
      console.log('Element is visible');
      entry.target.classList.add('visible');
    } else {
      console.log('Element is not visible');
      entry.target.classList.remove('visible');
    }
  });
}, {
  root: null,           // Viewport is root
  rootMargin: '0px',    // No margin
  threshold: 0.1        // Fire when 10% visible
});

intersectionObserver.observe(element);
intersectionObserver.unobserve(element);
intersectionObserver.disconnect();

// Resize Observer
const resizeObserver = new ResizeObserver(entries => {
  for (const entry of entries) {
    const { width, height } = entry.contentRect;
    console.log(`Element size: ${width}x${height}`);
  }
});

resizeObserver.observe(element);
resizeObserver.unobserve(element);
resizeObserver.disconnect();

// Performance Observer
const performanceObserver = new PerformanceObserver(list => {
  const entries = list.getEntries();
  for (const entry of entries) {
    console.log(`${entry.name}: ${entry.startTime}ms`);
  }
});

performanceObserver.observe({ entryTypes: ['resource', 'mark', 'measure'] });
performanceObserver.disconnect();

// Object mutators and accessors
```

```
const person = {};

// Define property with getter/setter
Object.defineProperty(person, 'name', {
  get() {
    return this._name;
  },
  set(value) {
    this._name = value;
  },
  enumerable: true,    // Shows in Object.keys()
  configurable: true   // Can be deleted/redefined
});

// Proxy
const handler = {
  get(target, property) {
    console.log(`Getting ${property}`);
    return target[property];
  },
  set(target, property, value) {
    console.log(`Setting ${property} to ${value}`);
    target[property] = value;
    return true;
  }
};
const proxy = new Proxy(obj, handler);

// Reflect API
Reflect.get(target, property);         // Get property
Reflect.set(target, property, value);  // Set property
Reflect.has(target, property);         // Check if property exists
Reflect.deleteProperty(target, property); // Delete property
Reflect.construct(Target, args);       // Construct with new
Reflect.apply(func, thisArg, args);    // Call function

// Web Workers
const worker = new Worker('worker.js');
worker.postMessage({ data: 'value' });
worker.onmessage = function(event) {
  console.log(event.data);
};
worker.terminate();
```

```
// Service Workers
navigator.serviceWorker.register('/sw.js')
  .then(registration => {
    console.log('Service worker registered');
  })
  .catch(error => {
    console.error('Registration failed:', error);
  });

// Shared Workers
const sharedWorker = new SharedWorker('worker.js');
sharedWorker.port.start();
sharedWorker.port.postMessage('Hello');
sharedWorker.port.onmessage = function(event) {
  console.log(event.data);
};

// Broadcast Channel
const channel = new BroadcastChannel('my_channel');
channel.postMessage('Hello everyone');
channel.onmessage = function(event) {
  console.log(event.data);
};
channel.close();

// IndexedDB
const request = indexedDB.open('myDatabase', 1);
request.onupgradeneeded = function(event) {
  const db = event.target.result;
  const store = db.createObjectStore('customers', { keyPath: 'id' });
  store.createIndex('name', 'name', { unique: false });
};
request.onsuccess = function(event) {
  const db = event.target.result;
  const transaction = db.transaction(['customers'], 'readwrite');
  const store = transaction.objectStore('customers');
  store.add({ id: 1, name: 'John' });
};

// Web Components
class MyElement extends HTMLElement {
  constructor() {
```

```
    super();
    this.attachShadow({ mode: 'open' });
    this.shadowRoot.innerHTML = `
      <style>
        :host { display: block; }
      </style>
      <div>Custom element content</div>
    `;
  }

  connectedCallback() {
    // Element added to DOM
  }

  disconnectedCallback() {
    // Element removed from DOM
  }

  attributeChangedCallback(name, oldValue, newValue) {
    // Attribute changed
  }

  static get observedAttributes() {
    return ['my-attr'];
  }
}
customElements.define('my-element', MyElement);
```

CSS

Selectors

```
/* Basic selectors */
* {}                    /* Universal selector */
element {}              /* Type selector */
.class {}               /* Class selector */
#id {}                  /* ID selector */
[attr] {}               /* Attribute selector */
[attr="value"] {}       /* Attribute exact value */
```

```
[attr^="val"] {}         /* Attribute starts with */
[attr$="lue"] {}         /* Attribute ends with */
[attr*="alu"] {}         /* Attribute contains */
[attr~="value"] {}       /* Attribute contains word */
[attr|="value"] {}       /* Attribute starts with value- */

/* Combinators */
elem1, elem2 {}          /* Multiple selectors */
parent > child {}        /* Direct child */
ancestor descendant {}   /* Descendant */
prev + next {}           /* Adjacent sibling */
prev ~ siblings {}       /* General siblings */

/* Pseudo-classes */
:hover {}                /* Mouse over element */
:active {}               /* Element being activated */
:focus {}                /* Element with focus */
:focus-visible {}        /* Element with keyboard focus */
:focus-within {}         /* Element or child has focus */
:target {}               /* Element targeted by URL hash */
:visited {}              /* Visited link */
:link {}                 /* Unvisited link */
:any-link {}             /* Any link (:link or :visited) */

/* Form pseudo-classes */
:checked {}              /* Checked input */
:disabled {}             /* Disabled element */
:enabled {}              /* Enabled element */
:valid {}                /* Valid form element */
:invalid {}              /* Invalid form element */
:required {}             /* Required form element */
:optional {}             /* Optional form element */
:in-range {}             /* Input value in range */
:out-of-range {}         /* Input value out of range */
:placeholder-shown {}    /* Input showing placeholder */
:read-only {}            /* Element not editable */
:read-write {}           /* Element editable */
:default {}              /* Default form element */

/* Structural pseudo-classes */
:root {}                 /* Root element (html) */
:empty {}                /* Element with no children */
:first-child {}          /* First child */
```

```
:last-child {}           /* Last child */
:only-child {}           /* Only child */
:first-of-type {}        /* First of type */
:last-of-type {}         /* Last of type */
:only-of-type {}         /* Only of type */
:nth-child(n) {}         /* Nth child */
:nth-last-child(n) {}    /* Nth last child */
:nth-of-type(n) {}       /* Nth of type */
:nth-last-of-type(n) {}  /* Nth last of type */

/* Pseudo-elements */
::before {}              /* Before element */
::after {}               /* After element */
::first-letter {}        /* First letter */
::first-line {}          /* First line */
::selection {}           /* Selected text */
::placeholder {}         /* Input placeholder */
::marker {}              /* List marker */
::backdrop {}            /* Backdrop for dialog/fullscreen */
::cue {}                 /* WebVTT cue text */
::slotted(selector) {}   /* Shadow DOM slotted elements */
::part(name) {}          /* Shadow DOM part */

/* Logical selectors */
:is(sel1, sel2) {}       /* Matches any of the selectors */
:where(sel1, sel2) {}    /* Like :is() but zero specificity */
:not(selector) {}        /* Negation */
:has(selector) {}        /* Parent contains selector */

/* Other selectors */
:dir(rtl) {}             /* RTL direction */
:lang(en) {}             /* Language */
:fullscreen {}           /* Fullscreen element */
:playing {}              /* Media is playing */
:paused {}               /* Media is paused */
:defined {}              /* Custom element that's been defined */

/* Special n-values for nth selectors */
:nth-child(odd) {}       /* Odd children (1, 3, 5...) */
:nth-child(even) {}      /* Even children (2, 4, 6...) */
:nth-child(3n) {}        /* Every third child (3, 6, 9...) */
:nth-child(3n+1) {}      /* Every third child offset by 1 (1, 4, 7...) */
```

```
:nth-child(-n+3) {}      /* First three children (1, 2, 3) */
:nth-child(n+3) {}       /* All children from third on (3, 4, 5...) */
```

Box Model & Layout

```
/* Box model */
.box {
  width: 100px;              /* Content width */
  height: 100px;             /* Content height */

  padding: 10px;             /* All sides */
  padding: 10px 20px;        /* Vertical | Horizontal */
  padding: 10px 20px 30px;   /* Top | Horizontal | Bottom */
  padding: 10px 20px 30px 40px; /* Top | Right | Bottom | Left */
  padding-top: 10px;
  padding-right: 20px;
  padding-bottom: 30px;
  padding-left: 40px;

  border: 1px solid black;   /* Width | Style | Color */
  border-width: 1px;
  border-style: solid;
  border-color: black;
  border-top: 1px solid black;
  border-right: 2px dashed red;
  border-bottom: 3px dotted green;
  border-left: 4px double blue;

  border-radius: 5px;        /* All corners */
  border-radius: 5px 10px;   /* Top-left/bottom-right | Top-right/bottom-left */
  border-radius: 5px 10px 15px; /* Top-left | Top-right/bottom-left | Bottom-right
  border-radius: 5px 10px 15px 20px; /* Top-left | Top-right | Bottom-right | Botto
  border-radius: 50%;        /* Circle (if square) */
  border-top-left-radius: 5px;
  border-top-right-radius: 10px;
  border-bottom-right-radius: 15px;
  border-bottom-left-radius: 20px;

  margin: 10px;              /* All sides */
  margin: 10px 20px;         /* Vertical | Horizontal */
  margin: 10px 20px 30px;    /* Top | Horizontal | Bottom */
```

```css
  margin: 10px 20px 30px 40px; /* Top | Right | Bottom | Left */
  margin-top: 10px;
  margin-right: 20px;
  margin-bottom: 30px;
  margin-left: 40px;
  margin: 0 auto;             /* Center horizontally */

  box-sizing: content-box;    /* Default: width/height = content only */
  box-sizing: border-box;     /* width/height includes padding & border */
}

/* Display properties */
.element {
  display: block;             /* Full-width block */
  display: inline;            /* Inline with content */
  display: inline-block;      /* Inline with block properties */
  display: flex;              /* Flexbox container */
  display: inline-flex;       /* Inline flexbox container */
  display: grid;              /* Grid container */
  display: inline-grid;       /* Inline grid container */
  display: table;             /* Table behavior */
  display: contents;          /* Children only */
  display: none;              /* Remove from layout */
}

/* Position properties */
.element {
  position: static;           /* Default */
  position: relative;         /* Relative to normal position */
  position: absolute;         /* Relative to positioned ancestor */
  position: fixed;            /* Relative to viewport */
  position: sticky;           /* Hybrid of relative/fixed */

  top: 10px;                  /* Offset from top */
  right: 20px;                /* Offset from right */
  bottom: 30px;               /* Offset from bottom */
  left: 40px;                 /* Offset from left */

  z-index: 10;                /* Stack order */

  float: left;                /* Float to the left */
  float: right;               /* Float to the right */
  float: none;                /* Default */
```

```
  clear: left;                /* Clear left floats */
  clear: right;               /* Clear right floats */
  clear: both;                /* Clear both directions */
  clear: none;                /* Default */
}

/* Size properties */
.element {
  width: 100px;               /* Fixed width */
  height: 100px;              /* Fixed height */

  min-width: 100px;           /* Minimum width */
  max-width: 1000px;          /* Maximum width */
  min-height: 100px;          /* Minimum height */
  max-height: 1000px;         /* Maximum height */

  width: 50%;                 /* Percentage of parent width */
  height: 50%;                /* Percentage of parent height */

  width: 100vw;               /* 100% of viewport width */
  height: 100vh;              /* 100% of viewport height */

  width: min-content;         /* Smallest possible width */
  width: max-content;         /* Largest max-content width */
  width: fit-content;         /* Between min and max content */

  width: clamp(200px, 50%, 600px); /* Min, preferred, max */

  aspect-ratio: 16 / 9;       /* Maintain aspect ratio */
}

/* Overflow properties */
.element {
  overflow: visible;          /* Default, content overflows */
  overflow: hidden;           /* Clip overflow */
  overflow: scroll;           /* Always show scrollbars */
  overflow: auto;             /* Show scrollbars when needed */

  overflow-x: auto;           /* Horizontal overflow */
  overflow-y: auto;           /* Vertical overflow */

  text-overflow: ellipsis;    /* Show ... for text overflow */
```

```
  white-space: nowrap;        /* No text wrapping */

  overflow: hidden;
  text-overflow: ellipsis;    /* Single line ellipsis */
  white-space: nowrap;

  display: -webkit-box;
  -webkit-line-clamp: 3;      /* Multi-line ellipsis (3 lines) */
  -webkit-box-orient: vertical;
  overflow: hidden;
}

/* Visibility properties */
.element {
  visibility: visible;        /* Default, element is visible */
  visibility: hidden;         /* Invisible but takes up space */
  visibility: collapse;       /* For table rows/columns */

  opacity: 1;                 /* Fully opaque */
  opacity: 0.5;               /* 50% transparent */
  opacity: 0;                 /* Fully transparent */
}
```

Flexbox

```
/* Container properties */
.flex-container {
  display: flex;               /* Create flexbox container */
  display: inline-flex;        /* Inline flexbox container */

  flex-direction: row;         /* Left to right (default) */
  flex-direction: row-reverse;  /* Right to left */
  flex-direction: column;      /* Top to bottom */
  flex-direction: column-reverse; /* Bottom to top */

  flex-wrap: nowrap;           /* Single line (default) */
  flex-wrap: wrap;             /* Multiple lines */
  flex-wrap: wrap-reverse;     /* Multiple lines, reversed */

  flex-flow: row wrap;         /* Shorthand for direction & wrap */
```

```
  justify-content: flex-start;  /* Items at start (default) */
  justify-content: flex-end;    /* Items at end */
  justify-content: center;      /* Items at center */
  justify-content: space-between; /* Equal space between items */
  justify-content: space-around;  /* Equal space around items */
  justify-content: space-evenly;  /* Equal space all around */

  align-items: stretch;         /* Stretch to fill (default) */
  align-items: flex-start;      /* Items at start */
  align-items: flex-end;        /* Items at end */
  align-items: center;          /* Items at center */
  align-items: baseline;        /* Items along text baseline */

  align-content: flex-start;    /* Lines at start */
  align-content: flex-end;      /* Lines at end */
  align-content: center;        /* Lines at center */
  align-content: space-between; /* Equal space between lines */
  align-content: space-around;  /* Equal space around lines */
  align-content: stretch;       /* Lines stretch to fill (default) */

  gap: 10px;                    /* Gap between items */
  gap: 10px 20px;               /* Row gap | Column gap */
  row-gap: 10px;                /* Gap between rows */
  column-gap: 20px;             /* Gap between columns */
}

/* Item properties */
.flex-item {
  order: 0;                     /* Default order */
  order: 1;                     /* Higher numbers come later */
  order: -1;                    /* Negative numbers come first */

  flex-grow: 0;                 /* No grow (default) */
  flex-grow: 1;                 /* Grow to fill space */
  flex-grow: 2;                 /* Grow twice as much */

  flex-shrink: 1;               /* Can shrink (default) */
  flex-shrink: 0;               /* Cannot shrink */
  flex-shrink: 2;               /* Shrink twice as much */

  flex-basis: auto;             /* Default size */
  flex-basis: 0;                /* Start with zero size */
```

```
  flex-basis: 100px;           /* Start with 100px */
  flex-basis: 50%;             /* Start with 50% */

  flex: 0 1 auto;              /* Default (grow shrink basis) */
  flex: 1;                     /* Same as flex: 1 1 0 */
  flex: auto;                  /* Same as flex: 1 1 auto */
  flex: none;                  /* Same as flex: 0 0 auto */

  align-self: auto;            /* Inherit from container */
  align-self: flex-start;      /* Item at start */
  align-self: flex-end;        /* Item at end */
  align-self: center;          /* Item at center */
  align-self: baseline;        /* Item along text baseline */
  align-self: stretch;         /* Item stretches to fill */
}
```

Grid

```
/* Container properties */
.grid-container {
  display: grid;               /* Create grid container */
  display: inline-grid;        /* Inline grid container */

  grid-template-columns: 100px 200px;            /* Fixed width columns */
  grid-template-columns: 1fr 2fr;                /* Fractional columns */
  grid-template-columns: repeat(3, 1fr);         /* 3 equal columns */
  grid-template-columns: minmax(100px, 1fr) 2fr;  /* Min/max size */
  grid-template-columns: auto 1fr auto;          /* Content-sized columns */

  grid-template-rows: 100px 200px;               /* Fixed height rows */
  grid-template-rows: 1fr 2fr;                   /* Fractional rows */
  grid-template-rows: repeat(3, 1fr);            /* 3 equal rows */
  grid-template-rows: minmax(100px, auto);       /* Min/max size */

  grid-template-areas:                           /* Named grid areas */
    "header header header"
    "sidebar content content"
    "footer footer footer";

  grid-template: 100px 1fr 50px /                /* Shorthand (rows / columns) */
                 1fr 3fr;
```

```css
grid-auto-columns: 100px;                /* Implicit column size */
grid-auto-rows: 100px;                   /* Implicit row size */

grid-auto-flow: row;                     /* Auto-place by row (default) */
grid-auto-flow: column;                  /* Auto-place by column */
grid-auto-flow: dense;                   /* Fill in gaps */

grid: 100px 1fr / 1fr 2fr;               /* Super shorthand */

column-gap: 10px;                        /* Gap between columns */
row-gap: 10px;                           /* Gap between rows */
gap: 10px;                               /* Gap for both */
gap: 10px 20px;                          /* Row gap | Column gap */

justify-items: stretch;                  /* Horizontal stretch (default) */
justify-items: start;                    /* Horizontal start */
justify-items: end;                      /* Horizontal end */
justify-items: center;                   /* Horizontal center */

align-items: stretch;                    /* Vertical stretch (default) */
align-items: start;                      /* Vertical start */
align-items: end;                        /* Vertical end */
align-items: center;                     /* Vertical center */

place-items: center;                     /* Center both ways */
place-items: start end;                  /* Vertical | Horizontal */

justify-content: start;                  /* Grid at start (default) */
justify-content: end;                    /* Grid at end */
justify-content: center;                 /* Grid at center */
justify-content: stretch;                /* Grid fills container */
justify-content: space-between;          /* Space between tracks */
justify-content: space-around;           /* Space around tracks */
justify-content: space-evenly;           /* Equal space all around */

align-content: start;                    /* Grid at start (default) */
align-content: end;                      /* Grid at end */
align-content: center;                   /* Grid at center */
align-content: stretch;                  /* Grid fills container */
align-content: space-between;            /* Space between tracks */
align-content: space-around;             /* Space around tracks */
align-content: space-evenly;             /* Equal space all around */
```

```
  place-content: center;                    /* Center both ways */
  place-content: start end;                 /* Vertical | Horizontal */
}

/* Item properties */
.grid-item {
  grid-column-start: 1;                     /* Start at column line 1 */
  grid-column-end: 3;                       /* End at column line 3 */
  grid-column: 1 / 3;                       /* Shorthand (start / end) */
  grid-column: 1 / span 2;                  /* Start at 1, span 2 columns */
  grid-column: 1 / -1;                      /* Start at 1, end at last line *

  grid-row-start: 1;                        /* Start at row line 1 */
  grid-row-end: 3;                          /* End at row line 3 */
  grid-row: 1 / 3;                          /* Shorthand (start / end) */
  grid-row: 1 / span 2;                     /* Start at 1, span 2 rows */

  grid-area: header;                        /* Named grid area */
  grid-area: 1 / 1 / 3 / 3;                 /* row-start / col-start / row-er

  justify-self: stretch;                    /* Horizontal stretch (default) *
  justify-self: start;                      /* Horizontal start */
  justify-self: end;                        /* Horizontal end */
  justify-self: center;                     /* Horizontal center */

  align-self: stretch;                      /* Vertical stretch (default) */
  align-self: start;                        /* Vertical start */
  align-self: end;                          /* Vertical end */
  align-self: center;                       /* Vertical center */

  place-self: center;                       /* Center both ways */
  place-self: start end;                    /* Vertical | Horizontal */

  order: 0;                                 /* Default placement order */
  order: 1;                                 /* Higher numbers come later */
  z-index: 1;                               /* Stack order for overlapping it
}

/* Grid functions */
.grid {
  /* repeat() - repeat patterns */
  grid-template-columns: repeat(3, 1fr);    /* 3 equal columns */
```

```
  grid-template-columns: repeat(3, 100px 200px);  /* Repeat pattern 3 times */

  /* minmax() - set minimum and maximum sizes */
  grid-template-columns: minmax(100px, 1fr);      /* Min 100px, max 1fr */

  /* fit-content() - fit to content with max size */
  grid-template-columns: fit-content(300px);      /* Fit to content up to 300px */

  /* auto-fill - fit as many as possible */
  grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));

  /* auto-fit - fit as many as possible and expand them */
  grid-template-columns: repeat(auto-fit, minmax(100px, 1fr));
}

/* Named grid lines */
.grid {
  grid-template-columns: [start] 1fr [middle] 2fr [end];
  grid-template-rows: [top] 100px [center] 1fr [bottom];
}

.item {
  grid-column: start / end;
  grid-row: top / center;
}

/* Subgrid */
.nested-grid {
  display: grid;
  grid-column: 1 / 3;
  grid-row: 1 / 3;
  grid-template-columns: subgrid;   /* Use parent grid columns */
  grid-template-rows: subgrid;      /* Use parent grid rows */
}
```

Animations & Transitions

```
/* Transitions */
.element {
  /* Property to animate */
```

```
  transition-property: all;
  transition-property: transform, opacity;

  /* Duration */
  transition-duration: 0.3s;
  transition-duration: 300ms;

  /* Timing function */
  transition-timing-function: ease;          /* Default - slow start, fast middle,
  transition-timing-function: linear;        /* Constant speed */
  transition-timing-function: ease-in;       /* Slow start */
  transition-timing-function: ease-out;      /* Slow end */
  transition-timing-function: ease-in-out;   /* Slow start and end */
  transition-timing-function: step-start;    /* Instant at start */
  transition-timing-function: step-end;      /* Instant at end */
  transition-timing-function: steps(4, end); /* 4 discrete steps */
  transition-timing-function: cubic-bezier(0.1, 0.7, 1.0, 0.1); /* Custom curve */

  /* Delay */
  transition-delay: 0.1s;
  transition-delay: 100ms;

  /* Shorthand */
  transition: all 0.3s ease 0.1s;            /* property duration timing-function c
  transition: transform 0.3s ease, opacity 0.5s linear; /* Multiple transitions */
}

/* Keyframe animations */
@keyframes slide-in {
  from {
    transform: translateX(-100%);
  }
  to {
    transform: translateX(0);
  }
}

@keyframes pulse {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.2);
```

```
  }
  100% {
    transform: scale(1);
  }
}

.element {
  /* Name of animation */
  animation-name: slide-in;

  /* Duration */
  animation-duration: 1s;

  /* Timing function */
  animation-timing-function: ease;

  /* Delay */
  animation-delay: 0.5s;

  /* Iteration count */
  animation-iteration-count: 1;              /* Once */
  animation-iteration-count: 3;              /* Three times */
  animation-iteration-count: infinite;       /* Infinite loop */

  /* Direction */
  animation-direction: normal;               /* Default */
  animation-direction: reverse;              /* Backwards */
  animation-direction: alternate;            /* Forward then backward */
  animation-direction: alternate-reverse;    /* Backward then forward */

  /* Fill mode */
  animation-fill-mode: none;                 /* Default */
  animation-fill-mode: forwards;             /* Keep end state */
  animation-fill-mode: backwards;            /* Apply initial state during delay */
  animation-fill-mode: both;                 /* Both forwards and backwards */

  /* Play state */
  animation-play-state: running;             /* Default */
  animation-play-state: paused;              /* Paused */

  /* Shorthand */
  animation: slide-in 1s ease 0.5s 3 alternate forwards;
  /* name duration timing-function delay iteration-count direction fill-mode */
```

```css
  /* Multiple animations */
  animation: slide-in 1s ease, pulse 2s infinite;
}

/* Transform */
.element {
  /* 2D transforms */
  transform: translateX(20px);              /* Move horizontally */
  transform: translateY(20px);              /* Move vertically */
  transform: translate(20px, 20px);         /* Move horizontally and vertically */

  transform: scale(1.5);                    /* Scale both axes */
  transform: scaleX(1.5);                   /* Scale horizontally */
  transform: scaleY(1.5);                   /* Scale vertically */
  transform: scale(1.5, 0.5);               /* Scale X and Y separately */

  transform: rotate(45deg);                 /* Rotate clockwise */
  transform: rotate(-45deg);                /* Rotate counter-clockwise */

  transform: skewX(10deg);                  /* Skew horizontally */
  transform: skewY(10deg);                  /* Skew vertically */
  transform: skew(10deg, 20deg);            /* Skew both axes */

  /* 3D transforms */
  transform: translateZ(20px);              /* Move along Z-axis */
  transform: translate3d(10px, 20px, 30px); /* Move in 3D space */

  transform: rotateX(45deg);                /* Rotate around X-axis */
  transform: rotateY(45deg);                /* Rotate around Y-axis */
  transform: rotateZ(45deg);                /* Rotate around Z-axis (same as rotat
  transform: rotate3d(1, 1, 1, 45deg);      /* Rotate around vector */

  transform: scaleZ(1.5);                   /* Scale along Z-axis */
  transform: scale3d(1, 1.5, 0.5);          /* Scale in 3D */

  transform: perspective(500px);            /* Apply perspective */

  /* Multiple transforms */
  transform: translate(20px, 20px) rotate(45deg) scale(1.5);

  /* Transform origin */
  transform-origin: center;                 /* Default */
```

```
  transform-origin: top left;               /* Top left corner */
  transform-origin: 50% 50%;                /* Center (same as default) */
  transform-origin: 0 0;                    /* Top left (in pixels) */
  transform-origin: 100% 100%;              /* Bottom right */
  transform-origin: 50% 50% 20px;           /* 3D origin */

  /* Perspective properties */
  perspective: 1000px;                      /* Perspective depth */
  perspective-origin: center;               /* Perspective viewpoint */

  /* Backface visibility */
  backface-visibility: visible;             /* Default */
  backface-visibility: hidden;              /* Hide element when facing away */

  /* Transform style */
  transform-style: flat;                    /* Default */
  transform-style: preserve-3d;             /* Preserve 3D positioning */
}

/* Motion path */
.element {
  offset-path: path('M0,0 L100,100 L200,0');  /* SVG path */
  offset-distance: 50%;                       /* Position along path (0-100%) */
  offset-rotate: auto;                        /* Auto-rotate */
  offset-rotate: 0deg;                        /* Fixed rotation */
  offset-rotate: auto 90deg;                  /* Auto + offset */

  /* Shorthand */
  offset: path('M0,0 L100,100') 50% auto;
}

/* Animation properties */
.element {
  /* Will-change - hint for browser optimization */
  will-change: transform, opacity;

  /* Pointer events - control interaction during animation */
  pointer-events: none;                       /* Disable interaction */
  pointer-events: auto;                       /* Enable interaction */
}

/* Media query for reduced motion preference */
@media (prefers-reduced-motion: reduce) {
```

```
  .element {
    animation: none;
    transition: none;
  }
}
```

Advanced CSS

```
/* Variables (Custom Properties) */
:root {
  --main-color: #3498db;
  --secondary-color: #2ecc71;
  --spacing-unit: 8px;
  --border-radius: 4px;
}

.element {
  color: var(--main-color);
  margin: var(--spacing-unit);
  border-radius: var(--border-radius);

  /* With fallback */
  color: var(--text-color, black);

  /* Local variable */
  --local-padding: 16px;
  padding: var(--local-padding);

  /* Variable with calc() */
  margin: calc(var(--spacing-unit) * 2);

  /* Reassigning variables */
  --spacing-unit: 16px;
}

/* Calculations */
.element {
  width: calc(100% - 20px);
  height: calc(100vh - 80px);
  font-size: calc(1rem + 2vw);
```

```
  padding: calc(var(--spacing-unit) * 2);
  margin: min(20px, 5%);
  border-radius: max(4px, 0.5rem);
  gap: clamp(10px, 5%, 50px);
}

/* Gradients */
.element {
  /* Linear gradient */
  background: linear-gradient(to right, red, blue);
  background: linear-gradient(45deg, red, blue);
  background: linear-gradient(to bottom right, red, blue);
  background: linear-gradient(135deg, red, blue);

  /* With stops */
  background: linear-gradient(to right, red 0%, blue 100%);
  background: linear-gradient(to right, red 0%, yellow 50%, blue 100%);
  background: linear-gradient(to right, red 0%, red 20%, blue 20%, blue 100%);

  /* Repeating */
  background: repeating-linear-gradient(45deg, red, red 10px, blue 10px, blue 20px)

  /* Radial gradient */
  background: radial-gradient(circle, red, blue);
  background: radial-gradient(ellipse, red, blue);

  /* Shape and position */
  background: radial-gradient(circle at center, red, blue);
  background: radial-gradient(circle at top left, red, blue);
  background: radial-gradient(ellipse at 50% 50%, red, blue);

  /* Size */
  background: radial-gradient(circle closest-side, red, blue);
  background: radial-gradient(circle closest-corner, red, blue);
  background: radial-gradient(circle farthest-side, red, blue);
  background: radial-gradient(circle farthest-corner, red, blue);

  /* Repeating */
  background: repeating-radial-gradient(circle, red, red 10px, blue 10px, blue 20px

  /* Conic gradient */
  background: conic-gradient(red, blue, green, red);
  background: conic-gradient(from 45deg, red, blue, green, red);
```

```
  background: conic-gradient(at 50% 50%, red, blue, green, red);
  background: conic-gradient(from 0deg at center, red, blue, green, red);

  /* With stops */
  background: conic-gradient(red 0deg, blue 180deg, green 270deg, red 360deg);

  /* Repeating */
  background: repeating-conic-gradient(red 0deg, blue 45deg, red 90deg);
}

/* Backgrounds */
.element {
  /* Multiple backgrounds - bottom layer is last */
  background:
    linear-gradient(to right, rgba(255,0,0,0.5), rgba(0,0,255,0.5)),
    url('image.jpg') center / cover;

  /* Size options */
  background-size: cover;            /* Cover entire area */
  background-size: contain;          /* Fit within area */
  background-size: 100% 100%;        /* Stretch to fill */
  background-size: 100px 100px;      /* Fixed size */
  background-size: 50% auto;         /* Percentage width, auto height */

  /* Position options */
  background-position: center;       /* Center */
  background-position: top left;     /* Top left */
  background-position: 50% 50%;      /* Center (percentages) */
  background-position: 20px 30px;    /* Pixels from top left */

  /* Repeat options */
  background-repeat: repeat;         /* Default, repeat both directions */
  background-repeat: no-repeat;      /* No repetition */
  background-repeat: repeat-x;       /* Repeat horizontally only */
  background-repeat: repeat-y;       /* Repeat vertically only */
  background-repeat: space;          /* Repeat with space between */
  background-repeat: round;          /* Repeat and scale to fit */

  /* Attachment options */
  background-attachment: scroll;     /* Scroll with content */
  background-attachment: fixed;      /* Fixed to viewport */
  background-attachment: local;      /* Scroll with element's content */
```

```
  /* Origin options */
  background-origin: padding-box;    /* Default, relative to padding box */
  background-origin: border-box;     /* Relative to border box */
  background-origin: content-box;    /* Relative to content box */

  /* Clip options */
  background-clip: border-box;       /* Default, clipped to border */
  background-clip: padding-box;      /* Clipped to padding */
  background-clip: content-box;      /* Clipped to content */
  background-clip: text;             /* Clipped to text */
  -webkit-background-clip: text;     /* For Safari */

  /* Text with gradient background */
  background: linear-gradient(45deg, #ff0000, #0000ff);
  background-clip: text;
  -webkit-background-clip: text;
  color: transparent;
}

/* Filters */
.element {
  filter: blur(5px);
  filter: brightness(150%);
  filter: contrast(200%);
  filter: grayscale(100%);
  filter: hue-rotate(90deg);
  filter: invert(100%);
  filter: opacity(50%);
  filter: saturate(200%);
  filter: sepia(100%);
  filter: drop-shadow(5px 5px 5px rgba(0,0,0,0.5));

  /* Multiple filters */
  filter: contrast(175%) brightness(103%) blur(1px);

  /* Backdrop filter (applies to background) */
  backdrop-filter: blur(10px);
  -webkit-backdrop-filter: blur(10px);
}

/* Masks */
.element {
  /* Image mask */
```

```
  mask-image: url('mask.png');
  -webkit-mask-image: url('mask.png');

  /* Gradient mask */
  mask-image: linear-gradient(to right, black, transparent);
  -webkit-mask-image: linear-gradient(to right, black, transparent);

  /* SVG mask */
  mask-image: url('mask.svg');
  -webkit-mask-image: url('mask.svg');

  /* Positioning */
  mask-position: center;
  -webkit-mask-position: center;

  /* Size */
  mask-size: cover;
  -webkit-mask-size: cover;

  /* Repeat */
  mask-repeat: no-repeat;
  -webkit-mask-repeat: no-repeat;

  /* Origin */
  mask-origin: padding-box;
  -webkit-mask-origin: padding-box;

  /* Clip */
  mask-clip: padding-box;
  -webkit-mask-clip: padding-box;

  /* Composite operation */
  mask-composite: add;
  -webkit-mask-composite: source-over;

  /* Multiple masks */
  mask-image: url('mask1.png'), linear-gradient(to right, black, transparent);
  -webkit-mask-image: url('mask1.png'), linear-gradient(to right, black, transparer
}

/* Shape related */
.element {
  /* Clip path */
```

```
  clip-path: circle(50%);
  clip-path: ellipse(50% 40% at 50% 50%);
  clip-path: polygon(50% 0%, 100% 38%, 82% 100%, 18% 100%, 0% 38%);
  clip-path: path('M 0 200 L 0,75 A 5,5 0,0,1 150,75 L 200 200 z');
  clip-path: url(#clip-path-id);

  /* Shape outside - wrap text around shape */
  shape-outside: circle(50%);
  shape-outside: ellipse(50% 40% at 50% 50%);
  shape-outside: polygon(50% 0%, 100% 38%, 82% 100%, 18% 100%, 0% 38%);
  shape-outside: url(shape.png);

  /* Shape margin - space between shape and content */
  shape-margin: 20px;
}

/* Blend modes */
.element {
  /* Background blend mode */
  background-blend-mode: normal;
  background-blend-mode: multiply;
  background-blend-mode: screen;
  background-blend-mode: overlay;
  background-blend-mode: darken;
  background-blend-mode: lighten;
  background-blend-mode: color-dodge;
  background-blend-mode: color-burn;
  background-blend-mode: hard-light;
  background-blend-mode: soft-light;
  background-blend-mode: difference;
  background-blend-mode: exclusion;
  background-blend-mode: hue;
  background-blend-mode: saturation;
  background-blend-mode: color;
  background-blend-mode: luminosity;

  /* Mix blend mode - how element blends with elements behind it */
  mix-blend-mode: normal;
  mix-blend-mode: multiply;
  mix-blend-mode: screen;
  mix-blend-mode: overlay;
  mix-blend-mode: darken;
  mix-blend-mode: lighten;
```

```
  mix-blend-mode: color-dodge;
  mix-blend-mode: color-burn;
  mix-blend-mode: hard-light;
  mix-blend-mode: soft-light;
  mix-blend-mode: difference;
  mix-blend-mode: exclusion;
  mix-blend-mode: hue;
  mix-blend-mode: saturation;
  mix-blend-mode: color;
  mix-blend-mode: luminosity;

  /* Isolation - create new stacking context */
  isolation: auto;
  isolation: isolate;
}

/* Scrolling */
.element {
  /* Scroll behavior */
  scroll-behavior: auto;
  scroll-behavior: smooth;

  /* Scroll snap */
  scroll-snap-type: x mandatory;
  scroll-snap-type: y proximity;
  scroll-snap-type: both mandatory;

  /* Padding for overscroll */
  overscroll-behavior: auto;
  overscroll-behavior: contain;
  overscroll-behavior: none;

  /* Individual overscroll behaviors */
  overscroll-behavior-x: contain;
  overscroll-behavior-y: none;

  /* Scroll margin - offset for snap points */
  scroll-margin: 10px;
  scroll-margin-top: 20px;

  /* Scroll padding - inset of snap area */
  scroll-padding: 10px;
  scroll-padding-bottom: 20px;
```

```
  /* Scroll snap align - alignment within snap container */
  scroll-snap-align: start;
  scroll-snap-align: center;
  scroll-snap-align: end;

  /* Scroll snap stop - whether to force stopping at snap points */
  scroll-snap-stop: normal;
  scroll-snap-stop: always;
}

/* Feature queries */
@supports (display: grid) {
  .container {
    display: grid;
  }
}

@supports not (display: grid) {
  .container {
    display: flex;
  }
}

@supports (display: grid) and (not (position: sticky)) {
  /* Code for browsers with grid but without sticky */
}

@supports selector(:has(.child)) {
  /* Code for browsers that support :has() */
}

/* Container queries */
@container (min-width: 400px) {
  .element {
    /* Styles when parent container is at least 400px wide */
  }
}

@container sidebar (max-width: 300px) {
  .element {
    /* Styles when container with class/name "sidebar" is at most 300px wide */
  }
```

```
}

/* Custom container query units */
.element {
  width: 50cqw;  /* 50% of container query width */
  height: 50cqh; /* 50% of container query height */
  font-size: 5cqi; /* 5% of the container query inline size */
}

/* Container for container queries */
.container {
  container-type: inline-size; /* Enable inline axis container queries */
  container-type: size;        /* Enable container queries on both axes */
  container-name: sidebar;     /* Named container for targeting */
}

/* Color functions */
.element {
  /* RGB and RGBA */
  color: rgb(255, 0, 0);
  color: rgb(100%, 0%, 0%);
  color: rgba(255, 0, 0, 0.5);

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

  /* Modern RGB syntax */
  color: rgb(255 0 0);
  color: rgb(255 0 0 / 50%);

  /* Modern HSL syntax */
  color: hsl(0 100% 50%);
  color: hsl(0 100% 50% / 50%);

  /* Hex */
  color: #FF0000;
  color: #F00;
  color: #FF0000AA; /* With alpha */

  /* Color mix */
  color: color-mix(in srgb, red, blue);
  color: color-mix(in srgb, red 30%, blue 70%);
```

```
  color: color-mix(in hsl, red, blue);

  /* Color contrast */
  color: color-contrast(rgb(200, 0, 0) vs black, white, gray);

  /* Color adjustments */
  color: adjust-color(hsl(0, 100%, 50%), lightness -10%);
  color: color-adjust(rgb(255, 0, 0), lightness -10%);
}

/* Logical properties */
.element {
  /* Margin */
  margin-inline: 10px;          /* Left and right margin in LTR */
  margin-inline-start: 10px;    /* Left margin in LTR */
  margin-inline-end: 20px;      /* Right margin in LTR */
  margin-block: 10px;           /* Top and bottom margin */
  margin-block-start: 10px;     /* Top margin */
  margin-block-end: 20px;       /* Bottom margin */

  /* Padding */
  padding-inline: 10px;
  padding-block: 20px;

  /* Width and height */
  inline-size: 200px;           /* Width in LTR */
  block-size: 100px;            /* Height */
  min-inline-size: 100px;       /* Min width in LTR */
  max-block-size: 300px;        /* Max height */

  /* Border */
  border-inline: 1px solid black;
  border-block: 2px dashed red;
  border-start-start-radius: 10px; /* Top-left in LTR */
  border-end-end-radius: 10px;     /* Bottom-right in LTR */

  /* Text align */
  text-align: start;            /* Left in LTR, right in RTL */
  text-align: end;              /* Right in LTR, left in RTL */
}

/* Viewport units */
.element {
```

```
  width: 50vw;                  /* 50% of viewport width */
  height: 50vh;                 /* 50% of viewport height */
  font-size: 5vmin;             /* 5% of viewport smaller dimension */
  padding: 3vmax;               /* 3% of viewport larger dimension */

  /* Small viewport units (ignore address bar) */
  height: 100svh;               /* 100% of small viewport height */
  width: 50svw;                 /* 50% of small viewport width */
  margin: 5svmin;               /* 5% of small viewport smaller dimension */

  /* Large viewport units (largest possible area) */
  height: 100lvh;               /* 100% of large viewport height */
  width: 50lvw;                 /* 50% of large viewport width */

  /* Dynamic viewport units (changes as UI elements appear/disappear) */
  height: 100dvh;               /* 100% of dynamic viewport height */
  width: 50dvw;                 /* 50% of dynamic viewport width */
}

/* Font features */
.element {
  /* Font feature settings */
  font-feature-settings: "liga" on;    /* Standard ligatures */
  font-feature-settings: "dlig" on;    /* Discretionary ligatures */
  font-feature-settings: "smcp" on;    /* Small caps */
  font-feature-settings: "zero" on;    /* Slashed zero */
  font-feature-settings: "tnum" on;    /* Tabular numbers */
  font-feature-settings: "frac" on;    /* Fractions */
  font-feature-settings: "ss01" on;    /* Stylistic set 1 */

  /* Shorthand for multiple features */
  font-feature-settings: "liga" on, "tnum" on;

  /* Font variation settings (variable fonts) */
  font-variation-settings: "wght" 700, "wdth" 80;

  /* Common properties for OpenType features */
  font-variant-ligatures: common-ligatures;
  font-variant-numeric: oldstyle-nums tabular-nums;
  font-variant-caps: small-caps;
  font-variant-position: sub;
  font-variant-east-asian: ruby;
  font-variant-alternates: historical-forms;
```

```
  /* All font variants shorthand */
  font-variant: small-caps tabular-nums;
}
```

Responsive Design

```
/* Media queries */
/* Width breakpoints */
@media (min-width: 600px) {
  /* Styles for viewport width >= 600px */
}

@media (max-width: 600px) {
  /* Styles for viewport width <= 600px */
}

@media (min-width: 600px) and (max-width: 900px) {
  /* Styles for viewport width between 600px and 900px */
}

/* Height breakpoints */
@media (min-height: 600px) {
  /* Styles for viewport height >= 600px */
}

/* Orientation */
@media (orientation: portrait) {
  /* Portrait mode */
}

@media (orientation: landscape) {
  /* Landscape mode */
}

/* Aspect ratio */
@media (aspect-ratio > 16/9) {
  /* Wider than 16:9 */
}

@media (min-aspect-ratio: 1/1) {
```

```
  /* Square or landscape */
}

/* Display type */
@media screen {
  /* Screen devices */
}

@media print {
  /* Print preview and printing */
}

@media speech {
  /* Screen readers */
}

/* Pixel density / Retina displays */
@media (-webkit-min-device-pixel-ratio: 2),
       (min-resolution: 192dpi) {
  /* High DPI screens */
}

/* Dynamic range */
@media (dynamic-range: high) {
  /* HDR displays */
}

/* Color scheme */
@media (prefers-color-scheme: dark) {
  /* Dark mode */
}

@media (prefers-color-scheme: light) {
  /* Light mode */
}

/* Reduced motion */
@media (prefers-reduced-motion: reduce) {
  /* Remove animations */
}

/* Contrast */
@media (prefers-contrast: high) {
```

```
  /* High contrast mode */
}

/* Hover capabilities */
@media (hover: hover) {
  /* Device supports hover */
}

@media (hover: none) {
  /* Device doesn't support hover (touch devices) */
}

/* Pointer precision */
@media (pointer: fine) {
  /* Precise pointer (mouse) */
}

@media (pointer: coarse) {
  /* Imprecise pointer (touch) */
}

/* Combining queries */
@media screen and (min-width: 600px) and (prefers-color-scheme: dark) {
  /* Dark mode on screens at least 600px wide */
}

/* Logical operators */
@media screen and (min-width: 600px) and (max-width: 900px) {
  /* AND - both conditions must be true */
}

@media screen, print {
  /* OR - either condition */
}

@media not print {
  /* NOT - everything except print */
}

/* Complex logical expressions */
@media (min-width: 600px) and ((max-width: 900px) or (prefers-color-scheme: dark))
  /* 600px+ AND either (max-width: 900px OR dark mode) */
}
```

```css
/* Container queries */
.container {
  container-type: inline-size;
}

@container (min-width: 400px) {
  .element {
    /* Styles when container is >= 400px */
  }
}

/* Container query units */
.element {
  font-size: calc(1.5rem + 1cqw);  /* Responsive to container width */
}
```

RegEx

```
// Basic patterns
/hello/          // Matches the literal string "hello"
/hello/i         // Case-insensitive match
/hello/g         // Global match (all occurrences)
/hello/m         // Multi-line match
/hello/u         // Unicode match
/hello/y         // Sticky match
/hello/s         // Dot matches newlines (dotAll)
/hello/gi        // Combining flags (global, case-insensitive)

// Character classes
/[abc]/          // Matches 'a', 'b', or 'c'
/[^abc]/         // Matches any character except 'a', 'b', or 'c'
/[a-z]/          // Matches any lowercase letter
/[A-Z]/          // Matches any uppercase letter
/[0-9]/          // Matches any digit
/[a-zA-Z0-9]/    // Matches any alphanumeric character

// Shorthand character classes
/\d/             // Digit [0-9]
/\D/             // Non-digit [^0-9]
```

```
/\w/             // Word character [a-zA-Z0-9_]
/\W/             // Non-word character [^a-zA-Z0-9_]
/\s/             // Whitespace character [ \t\r\n\f\v]
/\S/             // Non-whitespace character [^ \t\r\n\f\v]
/./              // Any character except newline
/\n/             // Newline character
/\t/             // Tab character
/\r/             // Carriage return
/\f/             // Form feed
/\v/             // Vertical tab
/\0/             // Null character
/\\/             // Backslash

// Unicode character classes
/\p{L}/u         // Any letter from any language
/\p{Ll}/u        // Lowercase letter
/\p{Lu}/u        // Uppercase letter
/\p{N}/u         // Number
/\p{Sc}/u        // Currency symbol
/\p{P}/u         // Punctuation
/\p{Emoji}/u     // Emoji

// Anchors and boundaries
/^hello/         // Matches 'hello' at the start of a string/line
/hello$/         // Matches 'hello' at the end of a string/line
/\bhello\b/      // Word boundary - matches 'hello' as a whole word
/\Bhello\B/      // Non-word boundary - matches 'hello' only if surrounded by word

// Quantifiers
/a*/             // 0 or more 'a's
/a+/             // 1 or more 'a's
/a?/             // 0 or 1 'a'
/a{3}/           // Exactly 3 'a's
/a{3,}/          // 3 or more 'a's
/a{1,3}/         // Between 1 and 3 'a's

// Greedy vs. lazy quantifiers
/a.*b/           // Greedy - matches from 'a' to the last 'b'
/a.*?b/          // Lazy - matches from 'a' to the first 'b'

// Groups and capturing
/(hello)/        // Capturing group
/(?:hello)/      // Non-capturing group
```

```
/hello|world/    // Alternation - matches 'hello' or 'world'

// Named groups
/(?<name>hello)/ // Named capturing group
/\k<name>/       // Back-reference to named group

// Back-references
/(hello)\1/      // Matches 'hellohello' (\1 refers to first group)
/(hi) (there) \1 \2/ // Matches 'hi there hi there'

// Positive lookahead
/hello(?=world)/ // Matches 'hello' only if followed by 'world'

// Negative lookahead
/hello(?!world)/ // Matches 'hello' only if not followed by 'world'

// Positive lookbehind
/(?<=hi )hello/  // Matches 'hello' only if preceded by 'hi '

// Negative lookbehind
/(?<!hi )hello/  // Matches 'hello' only if not preceded by 'hi '

// Common regex patterns
/^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$/ // Email
/^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,}$/           // Password (at least 8 chars, 1
/^(0[1-9]|1[0-2])\/(0[1-9]|[12][0-9]|3[01])\/\d{4}$/ // MM/DD/YYYY date format
/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9
/^[a-f0-9]{8}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{4}-[a-f0-9]{12}$/ // UUID

// Using RegExp in JavaScript
const regex = /pattern/flags;
const regex = new RegExp('pattern', 'flags');

// Test if pattern exists
regex.test('string');         // Returns true or false

// Match pattern and return info
'string'.match(regex);        // Returns array of matches or null
'string'.matchAll(regex);     // Returns iterator of all matches (with groups)

// Replace
'string'.replace(regex, 'replacement');     // Replace first match
'string'.replaceAll(regex, 'replacement');  // Replace all matches
```

```
// Split
'string'.split(regex);        // Split string by matches

// Search
'string'.search(regex);       // Returns index of first match or -1

// Exec method for iteration
let match;
while ((match = regex.exec('string')) !== null) {
  // Process match
}

// Using capture groups in replace
'John Smith'.replace(/(\w+) (\w+)/, '$2, $1');  // "Smith, John"

// Using function in replace
'John Smith'.replace(/(\w+) (\w+)/, (match, first, last) => {
  return `${last}, ${first}`;
});  // "Smith, John"
```

HTML DOM

Common Elements

```
<!-- Document Structure -->
<!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>
  <link rel="stylesheet" href="styles.css">
  <script src="script.js" defer></script>
</head>
<body>
  <header>
    <h1>Main Heading</h1>
    <nav>
```

```
      <ul>
        <li><a href="#section1">Section 1</a></li>
        <li><a href="#section2">Section 2</a></li>
      </ul>
    </nav>
  </header>

  <main>
    <section id="section1">
      <h2>Section 1 Heading</h2>
      <p>This is a paragraph with <strong>bold text</strong> and <em>italic text</e
    </section>

    <section id="section2">
      <h2>Section 2 Heading</h2>
      <p>Another paragraph with a <a href="https://example.com">link</a>.</p>
    </section>
  </main>

  <aside>
    <h3>Related Info</h3>
    <p>Sidebar content goes here.</p>
  </aside>

  <footer>
    <p>© 2025 Example Company</p>
  </footer>
</body>
</html>

<!-- Text Formatting -->
<p>Paragraph</p>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
<a href="url">Link</a>
<strong>Bold text</strong>
<b>Bold text (less semantic)</b>
<em>Italic text</em>
<i>Italic text (less semantic)</i>
```

```
<u>Underlined text</u>
<mark>Highlighted text</mark>
<del>Deleted text</del>
<ins>Inserted text</ins>
<sub>Subscript</sub>
<sup>Superscript</sup>
<small>Smaller text</small>
<code>Inline code</code>
<kbd>Keyboard input</kbd>
<samp>Sample output</samp>
<var>Variable</var>
<pre>Preformatted text</pre>
<blockquote>Block quotation</blockquote>
<q>Inline quotation</q>
<abbr title="Abbreviation">Abbr</abbr>
<address>Contact information</address>
<cite>Citation</cite>
<bdo dir="rtl">Right to left text</bdo>
<br> <!-- Line break -->
<hr> <!-- Horizontal rule -->
<wbr> <!-- Word break opportunity -->

<!-- Lists -->
<ul>
  <li>Unordered list item</li>
  <li>Another item</li>
</ul>

<ol>
  <li>Ordered list item</li>
  <li>Another item</li>
</ol>

<ol start="5" reversed type="A">
  <li>Custom ordered list</li>
  <li>Another item</li>
</ol>

<dl>
  <dt>Definition term</dt>
  <dd>Definition description</dd>
</dl>
```

```html
<!-- Links -->
<a href="https://example.com">External link</a>
<a href="/page">Internal link</a>
<a href="#section">Anchor link</a>
<a href="mailto:user@example.com">Email link</a>
<a href="tel:+1234567890">Phone link</a>
<a href="https://example.com" target="_blank">Open in new tab</a>
<a href="https://example.com" download>Download link</a>
<a href="https://example.com" rel="nofollow">No-follow link</a>
<a href="https://example.com" rel="noopener noreferrer">Safe external link</a>

<!-- Images -->
<img src="image.jpg" alt="Description">
<img src="image.jpg" alt="Description" width="300" height="200">
<img src="image.jpg" alt="Description" loading="lazy">
<picture>
  <source srcset="large.jpg" media="(min-width: 800px)">
  <source srcset="medium.jpg" media="(min-width: 600px)">
  <img src="small.jpg" alt="Description">
</picture>
<figure>
  <img src="image.jpg" alt="Description">
  <figcaption>Image caption</figcaption>
</figure>

<!-- Tables -->
<table>
  <caption>Table caption</caption>
  <colgroup>
    <col span="1" style="background-color: #eee;">
    <col span="2">
  </colgroup>
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Cell 1,1</td>
      <td>Cell 1,2</td>
    </tr>
```

```
    <tr>
      <td>Cell 2,1</td>
      <td>Cell 2,2</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="2">Footer spans 2 columns</td>
    </tr>
  </tfoot>
</table>

<!-- Forms -->
<form action="/submit" method="post" enctype="multipart/form-data">
  <fieldset>
    <legend>Personal Information</legend>

    <label for="name">Name:</label>
    <input type="text" id="name" name="name" placeholder="Enter name" required>

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

    <label for="password">Password:</label>
    <input type="password" id="password" name="password" minlength="8">

    <label for="phone">Phone:</label>
    <input type="tel" id="phone" name="phone" pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}">

    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob" min="1900-01-01" max="2023-12-31">

    <label for="color">Favorite Color:</label>
    <input type="color" id="color" name="color">

    <label for="quantity">Quantity:</label>
    <input type="number" id="quantity" name="quantity" min="1" max="10" step="1">

    <label for="range">Rating:</label>
    <input type="range" id="range" name="range" min="1" max="10" step="1">

    <label for="file">Upload File:</label>
    <input type="file" id="file" name="file" accept=".jpg,.png,.pdf" multiple>
```

```
<input type="checkbox" id="agree" name="agree" required>
<label for="agree">I agree to terms</label>

<fieldset>
  <legend>Gender</legend>
  <input type="radio" id="male" name="gender" value="male">
  <label for="male">Male</label>

  <input type="radio" id="female" name="gender" value="female">
  <label for="female">Female</label>

  <input type="radio" id="other" name="gender" value="other">
  <label for="other">Other</label>
</fieldset>

<label for="country">Country:</label>
<select id="country" name="country">
  <option value="" disabled selected>Select a country</option>
  <option value="us">United States</option>
  <option value="ca">Canada</option>
  <option value="mx">Mexico</option>
  <optgroup label="Europe">
    <option value="uk">United Kingdom</option>
    <option value="fr">France</option>
  </optgroup>
</select>

<label for="message">Message:</label>
<textarea id="message" name="message" rows="4" cols="40" maxlength="500" placeh

<label for="browser" list="browsers">Choose a browser:</label>
<input type="text" id="browser" name="browser" list="browsers">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Safari">
</datalist>

<input type="hidden" name="form_id" value="123">

<button type="submit">Submit</button>
<button type="reset">Reset</button>
```

```
    <button type="button">Regular Button</button>
    <input type="submit" value="Submit (Input)">
    <input type="reset" value="Reset (Input)">
    <input type="button" value="Button (Input)">
  </fieldset>
</form>

<!-- Output elements -->
<output name="result" for="a b">0</output>
<progress value="70" max="100">70%</progress>
<meter value="0.7" min="0" max="1" low="0.3" high="0.7" optimum="0.5">70%</meter>
<details>
  <summary>Click to show/hide</summary>
  <p>Hidden content revealed when clicked.</p>
</details>

<!-- Media -->
<audio controls autoplay muted loop>
  <source src="audio.mp3" type="audio/mp3">
  <source src="audio.ogg" type="audio/ogg">
  <p>Your browser doesn't support audio.</p>
</audio>

<video controls width="400" height="300" poster="poster.jpg" autoplay muted loop>
  <source src="video.mp4" type="video/mp4">
  <source src="video.webm" type="video/webm">
  <track src="subtitles.vtt" kind="subtitles" srclang="en" label="English">
  <p>Your browser doesn't support video.</p>
</video>

<iframe src="https://example.com" width="600" height="400" frameborder="0" allowful
<iframe src="https://www.youtube.com/embed/VIDEO_ID" width="560" height="315" frame

<object data="file.pdf" type="application/pdf" width="600" height="400">
  <p>Your browser doesn't support embedded PDFs. <a href="file.pdf">Download the PD
</object>

<embed src="file.svg" type="image/svg+xml" width="300" height="200">

<!-- Canvas and SVG -->
<canvas id="myCanvas" width="200" height="100"></canvas>

<svg width="200" height="100">
```

```
  <rect width="200" height="100" fill="blue" />
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
  <text x="100" y="50" font-family="Arial" font-size="16" text-anchor="middle" fill
</svg>

### Less Common & Niche Elements

<!-- Semantic Structure -->
<article>Self-contained content (blog post, article, comment)</article>
<section>Thematic grouping of content</section>
<nav>Navigation links</nav>
<aside>Content tangentially related to the main content</aside>
<header>Introductory content or navigational aids</header>
<footer>Footer for nearest sectioning content or root</footer>
<main>Main content of the document</main>
<hgroup>Heading group for a multi-level heading</hgroup>

<!-- Ruby Annotations (primarily for East Asian typography) -->
<ruby>
  漢 <rt>kan</rt> 字 <rt>ji</rt>
  <rp>(</rp><rt>kan</rt><rp>)</rp>
</ruby>

<!-- Definition References -->
<dfn>Term being defined</dfn>
<abbr title="World Health Organization">WHO</abbr>

<!-- Time and Date -->
<time datetime="2023-12-25">December 25, 2023</time>
<time datetime="2023-12-25T20:00:00Z">8 PM UTC on December 25</time>

<!-- Bidirectional Text -->
<bdo dir="rtl">Right-to-left text</bdo>
<bdi>Text isolated from its surroundings for bidirectional formatting</bdi>

<!-- Math Markup -->
<math>
  <mrow>
    <mi>x</mi>
    <mo>=</mo>
    <mfrac>
      <mrow>
        <mo>-</mo>
```

```
        <mi>b</mi>
        <mo>±</mo>
        <msqrt>
          <msup>
            <mi>b</mi>
            <mn>2</mn>
          </msup>
          <mo>-</mo>
          <mn>4</mn>
          <mi>a</mi>
          <mi>c</mi>
        </msqrt>
      </mrow>
      <mrow>
        <mn>2</mn>
        <mi>a</mi>
      </mrow>
    </mfrac>
  </mrow>
</math>

<!-- Dialog -->
<dialog open>
  <h2>Dialog Title</h2>
  <p>This is a dialog box.</p>
  <button>Close</button>
</dialog>

<!-- Menu Elements -->
<menu type="toolbar">
  <li><button>File</button></li>
  <li><button>Edit</button></li>
</menu>

<!-- Content Metadata -->
<meta name="description" content="Page description">
<meta name="keywords" content="HTML, CSS, JavaScript">
<meta name="author" content="John Doe">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta http-equiv="refresh" content="30">
<meta charset="UTF-8">
<meta property="og:title" content="Page Title">
<meta property="og:description" content="Page description">
```

```
<meta property="og:image" content="image.jpg">
<meta name="twitter:card" content="summary_large_image">

<!-- Obscure Form Elements -->
<keygen name="security">
<output name="result" for="a b">0</output>
<meter value="0.75" min="0" max="1">75%</meter>

<!-- Template Element (hidden content for JS use) -->
<template id="myTemplate">
  <div>Content that won't be rendered until JS uses it</div>
</template>

<!-- Web Components -->
<slot name="header">Default content if no slot provided</slot>
<shadow></shadow>

<!-- Content Editable -->
<div contenteditable="true">Edit this content</div>

<!-- Interactive Elements -->
<details>
  <summary>Click to expand</summary>
  <p>Hidden details text</p>
</details>

<dialog open>
  <p>Dialog content</p>
  <form method="dialog">
    <button>Close</button>
  </form>
</dialog>

<!-- Multimedia elements -->
<picture>
  <source srcset="large.jpg" media="(min-width: 800px)">
  <source srcset="medium.jpg" media="(min-width: 600px)">
  <img src="small.jpg" alt="Responsive image">
</picture>

<video controls crossorigin playsinline poster="poster.jpg">
  <source src="video.webm" type="video/webm">
  <source src="video.mp4" type="video/mp4">
```

```
  <track kind="subtitles" src="captions.vtt" srclang="en" label="English">
  <track kind="descriptions" src="descriptions.vtt" srclang="en" label="Descriptior
  <track kind="chapters" src="chapters.vtt" srclang="en" label="Chapters">
</video>

<!-- Data and Code -->
<data value="42">Forty-two</data>
<pre><code class="language-javascript">
function example() {
  console.log("Hello world");
}
</code></pre>

<!-- Semantic Text -->
<ins datetime="2023-03-15T15:30:00Z" cite="https://example.com/changes">Added text<
<del datetime="2023-03-15T15:30:00Z" cite="https://example.com/changes">Removed tex
<s>Text that is no longer relevant</s>

<!-- Niche Attributes -->
<p translate="no">Brand name</p>
<div hidden>Hidden content</div>
<div inert>Cannot be interacted with</div>
<img loading="lazy" decoding="async" src="image.jpg" alt="Lazy loaded image">
<a ping="https://example.com/tracker">Tracked link</a>
<div tabindex="-1">Not in tab order but can be focused programmatically</div>
<button autofocus>Auto-focused button</button>
<video disablepictureinpicture controlslist="nodownload noremoteplayback">Restricte
<form autocomplete="off" novalidate>Form without validation or autocomplete</form>

## Web Requests

### Fetch API

```javascript
// Basic GET request
fetch('https://api.example.com/data')
  .then(response => {
    // Check if response is OK (status 200-299)
    if (!response.ok) {
      throw new Error(`HTTP error! Status: ${response.status}`);
    }
    return response.json(); // Parse JSON response
  })
```
```

```
  .then(data => {
    console.log('Data:', data);
  })
  .catch(error => {
    console.error('Fetch error:', error);
  });

// Using async/await
async function fetchData() {
  try {
    const response = await fetch('https://api.example.com/data');

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

    const data = await response.json();
    console.log('Data:', data);
  } catch (error) {
    console.error('Fetch error:', error);
  }
}

// POST request with JSON data
fetch('https://api.example.com/post', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer TOKEN_HERE'
  },
  body: JSON.stringify({
    name: 'John Doe',
    email: 'john@example.com'
  })
})
.then(response => response.json())
.then(data => console.log('Success:', data))
.catch(error => console.error('Error:', error));

// PUT request
fetch('https://api.example.com/update/1', {
  method: 'PUT',
  headers: {
```

```
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({
    name: 'Updated Name'
  })
})
.then(response => response.json())
.then(data => console.log('Success:', data));

// DELETE request
fetch('https://api.example.com/delete/1', {
  method: 'DELETE'
})
.then(response => response.json())
.then(data => console.log('Success:', data));

// Form data
const formData = new FormData();
formData.append('name', 'John Doe');
formData.append('file', fileInput.files[0]);

fetch('https://api.example.com/upload', {
  method: 'POST',
  body: formData
})
.then(response => response.json())
.then(data => console.log('Success:', data));

// URL parameters
const params = new URLSearchParams();
params.append('search', 'query');
params.append('limit', '10');

fetch(`https://api.example.com/search?${params.toString()}`)
  .then(response => response.json())
  .then(data => console.log('Search results:', data));

// Request with timeout
const controller = new AbortController();
const timeoutId = setTimeout(() => controller.abort(), 5000); // 5 seconds

fetch('https://api.example.com/data', {
  signal: controller.signal
```

```
})
.then(response => response.json())
.then(data => {
  clearTimeout(timeoutId);
  console.log('Data:', data);
})
.catch(error => {
  if (error.name === 'AbortError') {
    console.log('Request timed out');
  } else {
    console.error('Fetch error:', error);
  }
});

// Response types
fetch('https://api.example.com/data')
  .then(response => {
    // Different response types
    // response.json() - Parse as JSON
    // response.text() - Parse as text
    // response.blob() - Parse as Blob (binary)
    // response.formData() - Parse as FormData
    // response.arrayBuffer() - Parse as ArrayBuffer

    // Response properties
    console.log('Status:', response.status);
    console.log('Status text:', response.statusText);
    console.log('HTTP version:', response.version);
    console.log('Success?', response.ok);
    console.log('Type:', response.type);
    console.log('URL:', response.url);

    // Headers
    console.log('Has header:', response.headers.has('Content-Type'));
    console.log('Header value:', response.headers.get('Content-Type'));

    // Iterate all headers
    for (const [key, value] of response.headers.entries()) {
      console.log(`${key}: ${value}`);
    }

    return response.json();
  });
```

```
// Fetch with credentials (cookies)
fetch('https://api.example.com/data', {
  credentials: 'include' // include, same-origin, or omit
});

// Mode options
fetch('https://api.example.com/data', {
  mode: 'cors' // cors, no-cors, same-origin, or navigate
});

// Cache options
fetch('https://api.example.com/data', {
  cache: 'no-cache' // default, no-store, reload, no-cache, force-cache, or only-if
});

// Redirect options
fetch('https://api.example.com/data', {
  redirect: 'follow' // follow, error, or manual
});

// Referrer policies
fetch('https://api.example.com/data', {
  referrerPolicy: 'no-referrer-when-downgrade'
  // no-referrer, no-referrer-when-downgrade, origin,
  // origin-when-cross-origin, same-origin, strict-origin,
  // strict-origin-when-cross-origin, or unsafe-url
});
```

XMLHttpRequest (Ajax)

```
// Basic GET request
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);

xhr.onload = function() {
  if (xhr.status >= 200 && xhr.status < 300) {
    const data = JSON.parse(xhr.responseText);
    console.log('Success:', data);
  } else {
```

```
    console.error('Error:', xhr.status, xhr.statusText);
  }
};

xhr.onerror = function() {
  console.error('Request failed');
};

xhr.send();

// POST request
const xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.example.com/post', true);
xhr.setRequestHeader('Content-Type', 'application/json');

xhr.onload = function() {
  if (xhr.status >= 200 && xhr.status < 300) {
    const data = JSON.parse(xhr.responseText);
    console.log('Success:', data);
  } else {
    console.error('Error:', xhr.status, xhr.statusText);
  }
};

xhr.send(JSON.stringify({
  name: 'John Doe',
  email: 'john@example.com'
}));

// Monitoring progress
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/large-file', true);

xhr.onprogress = function(event) {
  if (event.lengthComputable) {
    const percentComplete = (event.loaded / event.total) * 100;
    console.log(`Progress: ${percentComplete.toFixed(2)}%`);
  }
};

xhr.onload = function() {
  console.log('Request completed');
};
```

```
xhr.send();

// Timeout
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);
xhr.timeout = 5000; // 5 seconds

xhr.ontimeout = function() {
  console.error('Request timed out');
};

xhr.send();

// Form submission
const form = document.getElementById('myForm');
const formData = new FormData(form);

const xhr = new XMLHttpRequest();
xhr.open('POST', 'https://api.example.com/submit', true);

xhr.onload = function() {
  if (xhr.status >= 200 && xhr.status < 300) {
    console.log('Form submitted successfully');
  } else {
    console.error('Form submission failed');
  }
};

xhr.send(formData);

// All event handlers
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);

xhr.onreadystatechange = function() {
  // readyState values:
  // 0: UNSENT - Client created, open() not called
  // 1: OPENED - open() called
  // 2: HEADERS_RECEIVED - send() called, headers received
  // 3: LOADING - Downloading, responseText has partial data
  // 4: DONE - Operation complete
```

```
  console.log(`Ready state: ${xhr.readyState}`);

  if (xhr.readyState === 4) {
    if (xhr.status >= 200 && xhr.status < 300) {
      console.log('Complete!');
    }
  }
};

xhr.onloadstart = function() {
  console.log('Request started');
};

xhr.onprogress = function(event) {
  console.log(`Received ${event.loaded} of ${event.total || 'unknown'} bytes`);
};

xhr.onabort = function() {
  console.log('Request aborted');
};

xhr.onerror = function() {
  console.log('Request failed');
};

xhr.onload = function() {
  console.log('Request succeeded');
};

xhr.ontimeout = function() {
  console.log('Request timed out');
};

xhr.onloadend = function() {
  console.log('Request completed (success or failure)');
};

xhr.send();

// Aborting a request
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);
xhr.send();
```

```
// Later...
xhr.abort();

// Binary data
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://example.com/image.jpg', true);
xhr.responseType = 'blob';

xhr.onload = function() {
  if (xhr.status === 200) {
    const blob = xhr.response;
    const url = URL.createObjectURL(blob);
    const img = document.createElement('img');
    img.src = url;
    document.body.appendChild(img);
  }
};

xhr.send();

// withCredentials (CORS with cookies)
const xhr = new XMLHttpRequest();
xhr.open('GET', 'https://api.example.com/data', true);
xhr.withCredentials = true;
xhr.send();
```

Axios Library

```
// GET request
axios.get('https://api.example.com/data')
  .then(response => {
    console.log('Data:', response.data);
  })
  .catch(error => {
    console.error('Error:', error);
  });

// POST request
axios.post('https://api.example.com/post', {
  name: 'John Doe',
```

```
  email: 'john@example.com'
})
  .then(response => {
    console.log('Response:', response.data);
  })
  .catch(error => {
    console.error('Error:', error);
  });

// Async/await
async function fetchData() {
  try {
    const response = await axios.get('https://api.example.com/data');
    console.log('Data:', response.data);
  } catch (error) {
    console.error('Error:', error);
  }
}

// Request with configuration
axios({
  method: 'post',
  url: 'https://api.example.com/post',
  data: {
    name: 'John Doe'
  },
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer TOKEN'
  },
  timeout: 5000,
  withCredentials: true
})
  .then(response => console.log(response.data));

// Creating an instance with defaults
const api = axios.create({
  baseURL: 'https://api.example.com',
  timeout: 5000,
  headers: {
    'Authorization': 'Bearer TOKEN',
    'Content-Type': 'application/json'
  }
```

```
});

api.get('/users')
  .then(response => console.log(response.data));

// Form data
const formData = new FormData();
formData.append('name', 'John Doe');
formData.append('file', fileInput.files[0]);

axios.post('https://api.example.com/upload', formData, {
  headers: {
    'Content-Type': 'multipart/form-data'
  }
})
  .then(response => console.log(response.data));

// Request/response interceptors
axios.interceptors.request.use(
  config => {
    // Modify config before request is sent
    config.headers.Authorization = `Bearer ${getToken()}`;
    return config;
  },
  error => {
    // Do something with request error
    return Promise.reject(error);
  }
);

axios.interceptors.response.use(
  response => {
    // Any status code in range 2xx
    return response;
  },
  error => {
    // Any status codes outside range 2xx
    if (error.response && error.response.status === 401) {
      // Redirect to login
    }
    return Promise.reject(error);
  }
);
```

```
// Cancellation
const source = axios.CancelToken.source();

axios.get('https://api.example.com/data', {
  cancelToken: source.token
})
  .catch(error => {
    if (axios.isCancel(error)) {
      console.log('Request canceled:', error.message);
    } else {
      console.error('Error:', error);
    }
  });

// Cancel the request
source.cancel('Operation canceled by the user.');

// Multiple concurrent requests
axios.all([
  axios.get('https://api.example.com/users'),
  axios.get('https://api.example.com/posts')
])
  .then(axios.spread((usersResponse, postsResponse) => {
    console.log('Users:', usersResponse.data);
    console.log('Posts:', postsResponse.data);
  }));
```

HTTP Status Codes

```
// 1xx: Informational
100 Continue                      // Server received headers, client should send
101 Switching Protocols           // Server is switching protocols
102 Processing                    // Server received and is processing, no respor
103 Early Hints                   // With Link headers, helping browser preload r

// 2xx: Success
200 OK                            // Standard success response
201 Created                       // Resource created successfully
202 Accepted                      // Request accepted, processing ongoing
203 Non-Authoritative Information // From third-party or transformed by proxy
```

```
204 No Content                      // Success but no content returned
205 Reset Content                   // Reset document view
206 Partial Content                // Part of resource returned (Range header)
207 Multi-Status                    // Multiple status codes for multiple operatior
208 Already Reported                // Already included in a previous response
226 IM Used                         // GET request responded with instance manipula

// 3xx: Redirection
300 Multiple Choices                // Multiple options for resource
301 Moved Permanently               // Resource permanently moved to new URL
302 Found                           // Resource temporarily at different URL
303 See Other                       // Response found at different URL via GET
304 Not Modified                    // Resource not modified since last request
307 Temporary Redirect              // Temporary redirect keeping same method
308 Permanent Redirect              // Permanent redirect keeping same method

// 4xx: Client Error
400 Bad Request                     // Server cannot understand request
401 Unauthorized                    // Authentication required
402 Payment Required                // Reserved for future use
403 Forbidden                       // Server understood but refuses to authorize
404 Not Found                       // Resource not found
405 Method Not Allowed              // Method not allowed for resource
406 Not Acceptable                  // Can't respond with Accept headers requiremer
407 Proxy Authentication Required   // Authentication with proxy needed
408 Request Timeout                 // Server timed out waiting for request
409 Conflict                        // Request conflict with server state
410 Gone                            // Resource permanently gone
411 Length Required                 // Content-Length header required
412 Precondition Failed             // Server doesn't meet precondition headers
413 Payload Too Large               // Request entity too large
414 URI Too Long                    // URI too long for server to process
415 Unsupported Media Type          // Media format not supported
416 Range Not Satisfiable           // Range header cannot be fulfilled
417 Expectation Failed              // Expect header can't be met
418 I'm a Teapot                    // Joke response from RFC 2324
421 Misdirected Request             // Server can't produce response
422 Unprocessable Entity            // Semantic errors in request
423 Locked                          // Resource is locked
424 Failed Dependency               // Failed due to failure of previous request
425 Too Early                       // Server unwilling to risk processing potentia
426 Upgrade Required                // Client should switch to different protocol
428 Precondition Required           // Origin server requires conditional request
```

```
429 Too Many Requests               // User sent too many requests (rate limiting)
431 Request Header Fields Too Large // Header fields too large
451 Unavailable For Legal Reasons   // Legal reasons (censorship)

// 5xx: Server Error
500 Internal Server Error           // Generic server error
501 Not Implemented                 // Server doesn't support request functionality
502 Bad Gateway                     // Invalid response from upstream server
503 Service Unavailable             // Server temporarily unavailable
504 Gateway Timeout                 // Timeout from upstream server
505 HTTP Version Not Supported      // HTTP version not supported
506 Variant Also Negotiates         // Circular reference in content negotiation
507 Insufficient Storage            // Server can't store to complete request
508 Loop Detected                   // Infinite loop detected
510 Not Extended                    // Further extensions needed
511 Network Authentication Required // Client needs to authenticate for network
```

HTTP Methods

```
GET         // Retrieve data, should be idempotent, can be cached
HEAD        // Like GET but response has no body, just headers
POST        // Submit data to be processed, creates/updates resource
PUT         // Replace target resource with request payload, idempotent
DELETE      // Delete specified resource, idempotent
CONNECT     // Establish tunnel to server identified by target resource
OPTIONS     // Describe communication options for target resource
TRACE       // Loop-back test along path to target resource
PATCH       // Apply partial modifications to resource, may not be idempotent
```

HTTP Headers

```
// General Headers
Cache-Control        // Directives for caching mechanisms
Connection           // Control options for current connection
Content-Length       // Size of the body in bytes
Content-Type         // Media type of the resource
Date                 // Date and time message was sent
Keep-Alive           // Parameters to keep connection alive
Pragma               // Implementation-specific directives
```

```
Transfer-Encoding    // Form of encoding used to transfer entity
Upgrade              // Preferred communication protocols
Via                  // Intermediate protocols
Warning              // Warning information

// Request Headers
Accept               // Media types client can process
Accept-Charset       // Character sets client can process
Accept-Encoding      // Content encodings client can process
Accept-Language      // Natural languages client prefers
Authorization        // Credentials for HTTP authentication
Cookie               // Stored HTTP cookies
Expect               // Indicates server behaviors client requires
Forwarded            // Disclose original info of client connecting to proxy
From                 // Email address of user making request
Host                 // Target host's domain name and port
If-Match             // Perform conditionally if ETag matches
If-Modified-Since    // Perform conditionally if modified since time
If-None-Match        // Perform conditionally if ETag doesn't match
If-Range             // Conditionally request missing parts
If-Unmodified-Since  // Perform conditionally if unmodified since time
Origin               // Initiating request during CORS
Proxy-Authorization  // Authentication credentials for proxy
Range                // Request only a part of entity
Referer              // Previous web page that linked to this request
User-Agent           // Browser/client information

// Response Headers
Access-Control-Allow-Origin   // CORS - allowed origins
Access-Control-Allow-Methods  // CORS - allowed methods
Access-Control-Allow-Headers  // CORS - allowed headers
Access-Control-Max-Age        // CORS - max cache time
Age                  // Time in seconds object was in proxy cache
Allow                // Valid methods for resource
Content-Disposition  // Suggests filename for downloads
Content-Encoding     // Encoding transformations applied
Content-Language     // Natural language(s) intended for audience
Content-Location     // Alternate location for returned data
Content-Range        // Position of partial entity in full entity
ETag                 // Version identifier for current entity
Expires              // Date/time after which response is stale
Last-Modified        // Date/time resource was last modified
Location             // URL to redirect to
```

```
Proxy-Authenticate   // Authentication for proxy
Retry-After          // Time to wait before retry
Server               // Software used by origin server
Set-Cookie           // Send cookie to client
Vary                 // How to match future request headers
WWW-Authenticate     // Authentication for resource

// Security Headers
Content-Security-Policy    // Control resources browser can load
Strict-Transport-Security  // Force HTTPS use
X-Content-Type-Options     // Prevent MIME type sniffing
X-Frame-Options            // Control if/how page can be in frames
X-XSS-Protection           // Filter cross-site scripting attacks
```

WebSockets

```
// Creating a WebSocket connection
const socket = new WebSocket('wss://example.com/socket');

// Connection opened
socket.addEventListener('open', (event) => {
  console.log('Connection established');

  // Send a message to the server
  socket.send('Hello Server!');

  // Send JSON data
  socket.send(JSON.stringify({
    type: 'message',
    content: 'Hello',
    timestamp: Date.now()
  }));

  // Send binary data
  const buffer = new ArrayBuffer(8);
  const view = new DataView(buffer);
  view.setFloat64(0, Math.PI);
  socket.send(buffer);
});

// Listen for messages
```

```
socket.addEventListener('message', (event) => {
  console.log('Message from server:', event.data);

  // If JSON data
  try {
    const jsonData = JSON.parse(event.data);
    console.log('Parsed JSON:', jsonData);
  } catch (e) {
    console.log('Not JSON data');
  }

  // If binary data
  if (event.data instanceof Blob) {
    const reader = new FileReader();
    reader.onload = () => {
      const arrayBuffer = reader.result;
      console.log('Binary data:', new Uint8Array(arrayBuffer));
    };
    reader.readAsArrayBuffer(event.data);
  }
});

// Listen for errors
socket.addEventListener('error', (event) => {
  console.error('WebSocket error:', event);
});

// Listen for connection close
socket.addEventListener('close', (event) => {
  // event.code - close code
  // event.reason - close reason
  // event.wasClean - if connection closed cleanly

  console.log(`Connection closed. Code: ${event.code}, Reason: ${event.reason}`);

  if (event.wasClean) {
    console.log('Connection closed cleanly');
  } else {
    console.log('Connection died');
  }
});

// Closing the connection
```

```
socket.close();

// Close with code and reason
socket.close(1000, 'Closing normally');

// WebSocket properties
console.log('URL:', socket.url);
console.log('Protocol:', socket.protocol);
console.log('State:', socket.readyState);
// readyState values:
// WebSocket.CONNECTING (0): Connection not yet established
// WebSocket.OPEN (1): Connection established
// WebSocket.CLOSING (2): Connection closing
// WebSocket.CLOSED (3): Connection closed

console.log('Buffered amount:', socket.bufferedAmount);
// Amount of data queued but not yet sent

// Ping/Pong (heartbeat) handled automatically by browser
```

Server-Sent Events (SSE)

```
// Creating an EventSource connection
const eventSource = new EventSource('https://example.com/events');

// Listen for all messages
eventSource.addEventListener('message', (event) => {
  console.log('Received message:', event.data);
  console.log('Origin:', event.origin);
  console.log('Last event ID:', event.lastEventId);
});

// Listen for a specific event type
eventSource.addEventListener('userconnected', (event) => {
  const user = JSON.parse(event.data);
  console.log(`User connected: ${user.name}`);
});

// Listen for open event
eventSource.addEventListener('open', (event) => {
  console.log('Connection opened');
```

```
});

// Listen for error event
eventSource.addEventListener('error', (event) => {
  if (eventSource.readyState === EventSource.CLOSED) {
    console.log('Connection closed');
  } else {
    console.error('Error occurred:', event);
  }
});

// Properties
console.log('URL:', eventSource.url);
console.log('State:', eventSource.readyState);
// readyState values:
// EventSource.CONNECTING (0): Connection not yet established
// EventSource.OPEN (1): Connection established
// EventSource.CLOSED (2): Connection closed

console.log('WithCredentials:', eventSource.withCredentials);

// Close the connection
eventSource.close();

// EventSource with credentials
const eventSourceWithCredentials = new EventSource('https://example.com/events', {
  withCredentials: true
});
```

Service Workers

```
// Registering a service worker
if ('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js', { scope: '/' })
    .then(registration => {
      console.log('Service Worker registered with scope:', registration.scope);
    })
    .catch(error => {
      console.error('Service Worker registration failed:', error);
    });
}
```

```
// Check if service worker is active
navigator.serviceWorker.ready
  .then(registration => {
    console.log('Service Worker is active');
  });

// Communicating with service worker
navigator.serviceWorker.ready
  .then(registration => {
    registration.active.postMessage({
      type: 'CACHE_NEW_ROUTE',
      payload: '/dashboard'
    });
  });

// Listening for messages from service worker
navigator.serviceWorker.addEventListener('message', event => {
  console.log('Message from Service Worker:', event.data);
});

// Unregister service worker
navigator.serviceWorker.getRegistration()
  .then(registration => {
    if (registration) {
      registration.unregister()
        .then(success => {
          console.log('Service Worker unregistered:', success);
        });
    }
  });

// Service Worker script (sw.js)
// Installation - cache resources
self.addEventListener('install', event => {
  event.waitUntil(
    caches.open('v1').then(cache => {
      return cache.addAll([
        '/',
        '/index.html',
        '/styles.css',
        '/script.js',
        '/image.png'
```

```
      ]);
    })
  );

  // Skip waiting - activate immediately
  self.skipWaiting();
});

// Activation - clean up old caches
self.addEventListener('activate', event => {
  event.waitUntil(
    caches.keys().then(cacheNames => {
      return Promise.all(
        cacheNames.filter(cacheName => {
          return cacheName !== 'v1';
        }).map(cacheName => {
          return caches.delete(cacheName);
        })
      );
    })
  );

  // Claim clients - take control of uncontrolled clients
  self.clients.claim();
});

// Intercept network requests
self.addEventListener('fetch', event => {
  event.respondWith(
    caches.match(event.request)
      .then(response => {
        // Cache hit - return response
        if (response) {
          return response;
        }

        // Clone the request
        const fetchRequest = event.request.clone();

        return fetch(fetchRequest).then(response => {
          // Check if valid response
          if (!response || response.status !== 200 || response.type !== 'basic') {
            return response;
```

```
          }

          // Clone the response
          const responseToCache = response.clone();

          caches.open('v1').then(cache => {
            cache.put(event.request, responseToCache);
          });

          return response;
        });
      })
  );
});

// Listening for messages from main thread
self.addEventListener('message', event => {
  console.log('Message received:', event.data);

  // Send response back
  event.source.postMessage({
    type: 'RESPONSE',
    payload: 'Message received'
  });
});

// Push notifications
self.addEventListener('push', event => {
  const data = event.data.json();

  const options = {
    body: data.body,
    icon: '/icon.png',
    badge: '/badge.png',
    vibrate: [100, 50, 100],
    data: {
      url: data.url
    }
  };

  event.waitUntil(
    self.registration.showNotification(data.title, options)
  );
```

```
});

// Notification click
self.addEventListener('notificationclick', event => {
  event.notification.close();

  event.waitUntil(
    clients.openWindow(event.notification.data.url)
  );
});

// Background sync
self.addEventListener('sync', event => {
  if (event.tag === 'sync-messages') {
    event.waitUntil(syncMessages());
  }
});

async function syncMessages() {
  try {
    const messagesQueue = await getMessagesFromIndexedDB();
    for (const message of messagesQueue) {
      await sendMessage(message);
      await removeMessageFromIndexedDB(message.id);
    }
  } catch (error) {
    console.error('Sync failed:', error);
  }
}
```