📘 JavaScript & DOM Complete Cheat Sheet

A comprehensive reference guide for JavaScript DOM manipulation, events, and common patterns

📑 Table of Contents

🧩 DOM Basics

What is the DOM?

DOM (Document Object Model) is the programming interface for HTML documents. It represents the page as a tree of objects.

```
<input type="text" name="address">
```

➡️ In JavaScript:

```
input.type   // "text"
input.name   // "address"
```

Document Properties

Property	Description	Example
`document.body`	`<body>` element	`document.body.style.background = 'blue'`
`document.head`	`<head>` element	`document.head.querySelector('title')`
`document.title`	Page title	`document.title = 'New Title'`
`document.forms`	All `<form>` elements	`document.forms[0]`
`document.images`	All `<img>` elements	`document.images.length`
`document.links`	All `<a>` with href	`document.links[0].href`

🔍 DOM Selection Methods

`getElementById()`

Selects **one element** by ID (fastest method).

```
const header = document.getElementById('mainHeader');
header.textContent = 'Welcome!';
```

`getElementsByClassName()`

Returns a **live HTMLCollection** of elements with the class.

```
const menuItems = document.getElementsByClassName('menu-item');
// Convert to array
Array.from(menuItems).forEach(item => {
    item.style.color = 'blue';
});
```

`getElementsByTagName()`

Returns all elements with the specified tag name.

```
const allParagraphs = document.getElementsByTagName('p');
for (let p of allParagraphs) {
    p.style.lineHeight = '1.6';
}
```

`querySelector()`

Returns the **first element** matching a CSS selector.

```
const firstButton = document.querySelector('.btn-primary');
const activeItem = document.querySelector('.menu-item.active');
const dataAttr = document.querySelector('[data-id="123"]');
```

`querySelectorAll()`

Returns a **static NodeList** of all matching elements.

```
const allButtons = document.querySelectorAll('button');
allButtons.forEach(btn => {
    btn.addEventListener('click', () => console.log('Clicked!'));
});

// Complex selectors
const items = document.querySelectorAll('.menu-item:nth-child(odd)');
```

💡 **Tip:** Use `querySelectorAll()` for complex CSS selectors!

✏️ DOM Manipulation

Creating Elements

```
// Create element
const newDiv = document.createElement('div');
newDiv.className = 'card';
newDiv.id = 'card-1';
newDiv.textContent = 'Hello World';

// Create with template literal
newDiv.innerHTML = `
    <h3>${title}</h3>
    <p>${content}</p>
`;
```

Adding Elements

```
// Append to end
parent.appendChild(newDiv);

// Insert at beginning
parent.insertBefore(newDiv, parent.firstChild);

// Insert adjacent (modern)
element.insertAdjacentHTML('beforebegin', '<div>Before</div>');
element.insertAdjacentHTML('afterend', '<div>After</div>');
```

Removing Elements

```
// Modern way
element.remove();

// Classic way
parent.removeChild(element);

// Remove all children
container.innerHTML = '';
```

Cloning Elements

```
// Shallow clone (no children)
const clone = element.cloneNode(false);

// Deep clone (with all children)
const deepClone = element.cloneNode(true);
document.body.appendChild(deepClone);
```

Modifying Content

```
// Text only (escapes HTML)
element.textContent = 'Plain text';

// HTML content
element.innerHTML = '<b>Bold</b> text';

// Get/set value of form inputs
input.value = 'New value';
```

Working with Attributes

```
// Set attribute
img.setAttribute('src', 'image.jpg');
img.setAttribute('alt', 'Description');

// Get attribute
const href = link.getAttribute('href');

// Remove attribute
link.removeAttribute('target');

// Data attributes
element.dataset.userId = '123';
const id = element.dataset.userId; // '123'
```

Working with Classes

```
// Add classes
element.classList.add('active', 'highlighted');

// Remove classes
element.classList.remove('active');

// Toggle class
element.classList.toggle('open');

// Check if has class
if (element.classList.contains('active')) {
    console.log('Element is active');
}

// Replace class
element.classList.replace('old-class', 'new-class');
```

Modifying Styles

```
// Individual styles
element.style.backgroundColor = 'lightcoral';
element.style.fontSize = '20px';
element.style.display = 'none';

// Multiple styles with cssText
element.style.cssText = 'background: blue; color: white; padding: 10px;';

// Computed styles (read-only)
const computed = window.getComputedStyle(element);
console.log(computed.backgroundColor);
```

🌳 DOM Traversal

Parent Navigation

```
// Get parent element
const parent = element.parentElement;
const parentNode = element.parentNode; // Can be non-element

// Climb up to find ancestor
const ancestor = element.closest('.container');
```

Children Navigation

```
// All children (elements only)
const children = element.children;

// All nodes (including text nodes)
const childNodes = element.childNodes;

// First and last child
const first = element.firstElementChild;
const last = element.lastElementChild;
```

Sibling Navigation

```
// Next sibling element
const next = element.nextElementSibling;

// Previous sibling element
const prev = element.previousElementSibling;

// Iterate through siblings
let sibling = element.nextElementSibling;
while (sibling) {
    console.log(sibling);
    sibling = sibling.nextElementSibling;
}
```

Node Properties

Property	Description
`nodeName`	Tag name (uppercase)
`nodeType`	1 = element, 3 = text
`parentNode`	Parent node
`childNodes`	All child nodes
`firstChild` / `lastChild`	First/last node
`nextSibling` / `previousSibling`	Adjacent nodes

⚡ Event Handling

Adding Event Listeners

```javascript
// Basic event listener
button.addEventListener('click', function(e) {
    console.log('Clicked!', e);
});

// Arrow function
button.addEventListener('click', (e) => {
    console.log('Clicked!');
});

// With options
element.addEventListener('click', handler, {
    once: true,      // Remove after first call
    capture: false,  // Use bubble phase
    passive: true    // Never call preventDefault
});
```

Removing Event Listeners

```javascript
function handleClick(e) {
    console.log('Clicked!');
}

// Add listener
button.addEventListener('click', handleClick);

// Remove listener (must use same function reference)
button.removeEventListener('click', handleClick);
```

Event Object Properties

```
element.addEventListener('click', (e) => {
    e.target         // Element that triggered event
    e.currentTarget  // Element with the listener
    e.type           // Event type ('click')
    e.timeStamp      // Time when event occurred

    // Mouse events
    e.clientX, e.clientY  // Relative to viewport
    e.pageX, e.pageY      // Relative to document
    e.offsetX, e.offsetY  // Relative to element

    // Keyboard events
    e.key            // Key pressed ('a', 'Enter')
    e.keyCode        // Numeric key code (deprecated)
    e.ctrlKey        // Ctrl key pressed
    e.shiftKey       // Shift key pressed
    e.altKey         // Alt key pressed
});
```

Preventing Default Behavior

```
// Prevent form submission
form.addEventListener('submit', (e) => {
    e.preventDefault();
    // Handle form data
});

// Prevent link navigation
link.addEventListener('click', (e) => {
    e.preventDefault();
    // Custom behavior
});

// Prevent right-click menu
element.addEventListener('contextmenu', (e) => {
    e.preventDefault();
});
```

🎯 Event Types

Mouse Events

```
// Click events
element.addEventListener('click', handler);
element.addEventListener('dblclick', handler);
element.addEventListener('contextmenu', handler); // Right-click

// Mouse movement
element.addEventListener('mouseenter', handler);
element.addEventListener('mouseleave', handler);
element.addEventListener('mouseover', handler);
element.addEventListener('mouseout', handler);
element.addEventListener('mousemove', handler);

// Mouse buttons
element.addEventListener('mousedown', handler);
element.addEventListener('mouseup', handler);
```

Keyboard Events

```
// Keyboard events
input.addEventListener('keydown', (e) => {
    console.log('Key:', e.key);
    console.log('Code:', e.code);

    // Check modifiers
    if (e.ctrlKey && e.key === 's') {
        e.preventDefault();
        // Save functionality
    }
});

input.addEventListener('keyup', handler);
input.addEventListener('keypress', handler); // Deprecated
```

Form Events

```
// Input changes
input.addEventListener('input', (e) => {
    console.log('Value:', e.target.value);
});

// Value committed (blur or enter)
input.addEventListener('change', (e) => {
    console.log('Changed to:', e.target.value);
});

// Form submission
form.addEventListener('submit', (e) => {
    e.preventDefault();
    const formData = new FormData(form);
});

// Focus events
input.addEventListener('focus', handler);
input.addEventListener('blur', handler);
```

Drag and Drop Events

```
// On draggable element
element.addEventListener('dragstart', (e) => {
    e.dataTransfer.setData('text/plain', element.id);
});

element.addEventListener('dragend', handler);

// On drop zone
dropZone.addEventListener('dragover', (e) => {
    e.preventDefault(); // Required!
});

dropZone.addEventListener('drop', (e) => {
    e.preventDefault();
    const data = e.dataTransfer.getData('text/plain');
});
```

Window Events

```javascript
// Page load
window.addEventListener('load', () => {
    console.log('Page fully loaded');
});

// DOM ready (better than load)
document.addEventListener('DOMContentLoaded', () => {
    console.log('DOM ready');
});

// Window resize
window.addEventListener('resize', () => {
    console.log(`${window.innerWidth} x ${window.innerHeight}`);
});

// Scroll event
window.addEventListener('scroll', () => {
    console.log('Scroll Y:', window.scrollY);
});

// Before unload
window.addEventListener('beforeunload', (e) => {
    e.preventDefault();
    e.returnValue = ''; // Show confirmation dialog
});
```

🎭 Event Propagation

Event Flow Phases

1. **Capture Phase**: Event travels from `window` down to target
2. **Target Phase**: Event reaches the target element
3. **Bubble Phase**: Event bubbles back up to `window`

```
// Bubble phase (default)
element.addEventListener('click', handler, false);

// Capture phase
element.addEventListener('click', handler, true);
```

Stopping Propagation

```
element.addEventListener('click', (e) => {
    // Stop bubbling to parents
    e.stopPropagation();

    // Stop other listeners on same element
    e.stopImmediatePropagation();
});
```

Event Delegation

Attach one listener to parent instead of many to children.

```
// Bad: Multiple listeners
document.querySelectorAll('.item').forEach(item => {
    item.addEventListener('click', handler);
});

// Good: Single delegated listener
document.getElementById('list').addEventListener('click', (e) => {
    if (e.target.classList.contains('item')) {
        // Handle item click
        console.log('Item clicked:', e.target);
    }

    if (e.target.classList.contains('delete-btn')) {
        // Handle delete
        e.target.closest('.item').remove();
    }
});
```

Benefits:

- Better performance
- Works with dynamically added elements
- Less memory usage

Custom Events

```
// Create custom event
const event = new CustomEvent('userLogin', {
    detail: { username: 'john', role: 'admin' },
    bubbles: true,
    cancelable: true
});

// Dispatch event
document.dispatchEvent(event);

// Listen for custom event
document.addEventListener('userLogin', (e) => {
    console.log('User:', e.detail.username);
    console.log('Role:', e.detail.role);
});
```

📊 Array Methods

`forEach()` - Iterate

```javascript
const colors = ['red', 'green', 'blue'];
colors.forEach((color, index) => {
    console.log(`${index}: ${color}`);
});
```

`map()` - Transform

```javascript
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num => num * 2);
// [2, 4, 6, 8]

const grades = students.map(s => s.grade);
```

`filter()` - Select

```javascript
const numbers = [1, 2, 3, 4, 5, 6];
const even = numbers.filter(num => num % 2 === 0);
// [2, 4, 6]

const active = todos.filter(todo => !todo.completed);
```

`find()` - Find First

```javascript
const users = [
    { id: 1, name: 'John' },
    { id: 2, name: 'Jane' }
];
const user = users.find(u => u.id === 2);
// { id: 2, name: 'Jane' }
```

`findIndex()` - Find Index

```
const index = students.findIndex(s => s.id === 5);
if (index !== -1) {
    students.splice(index, 1); // Remove student
}
```

`reduce()` - Accumulate

```
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((total, num) => total + num, 0);
// 10

const average = grades.reduce((sum, g) => sum + g, 0) / grades.length;
```

`sort()` - Sort

```
// Numbers ascending
numbers.sort((a, b) => a - b);

// Numbers descending
numbers.sort((a, b) => b - a);

// Strings alphabetically
names.sort((a, b) => a.localeCompare(b));

// Objects by property
students.sort((a, b) => a.name.localeCompare(b.name));
students.sort((a, b) => b.grade - a.grade); // By grade desc
```

Other Useful Methods

```
// Add to end
array.push(item);

// Remove from end
const last = array.pop();

// Add to beginning
array.unshift(item);

// Remove from beginning
const first = array.shift();

// Join to string
const csv = array.join(', ');

// Check if includes
if (array.includes(value)) { }

// Convert to array
const arr = Array.from(nodeList);
```

📝 String Methods

```javascript
const text = '  Hello World  ';

// Remove whitespace
text.trim()  // 'Hello World'

// Case conversion
text.toLowerCase()  // '  hello world  '
text.toUpperCase()  // '  HELLO WORLD  '

// Check contains
text.includes('World')  // true

// Split to array
'apple,banana,orange'.split(',')  // ['apple', 'banana', 'orange']

// Replace
text.replace('World', 'JavaScript')
text.replaceAll('o', '0')

// Substring
text.substring(0, 5)  // '  Hel'
text.slice(2, 7)      // 'Hello'

// Repeat
'- '.repeat(3)  // '- - - '

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

🔢 Math Functions

```
// Rounding
Math.round(4.5)   // 5
Math.floor(4.9)   // 4
Math.ceil(4.1)    // 5

// Min and max
Math.max(1, 5, 3, 9, 2)  // 9
Math.min(1, 5, 3, 9, 2)  // 1

const highest = Math.max(...grades);

// Random
Math.random()  // 0 to 1 (exclusive)

// Random integer between min and max (inclusive)
function randomInt(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

// Other
Math.abs(-5)      // 5
Math.sqrt(16)     // 4
Math.pow(2, 3)    // 8
```

📅 Date & Time

```
// Current date/time
const now = new Date();

// Specific date
const date = new Date('2024-01-15');
const date2 = new Date(2024, 0, 15); // Month is 0-indexed!

// Formatting
now.toLocaleDateString()        // '1/15/2024'
now.toLocaleTimeString()        // '2:30:45 PM'
now.toISOString()               // '2024-01-15T14:30:45.000Z'

// Timestamp
Date.now()               // Milliseconds since 1970
now.getTime()            // Same

// Components
now.getFullYear()        // 2024
now.getMonth()           // 0-11
now.getDate()            // 1-31
now.getDay()             // 0-6 (Sunday = 0)
now.getHours()           // 0-23
now.getMinutes()         // 0-59
now.getSeconds()         // 0-59
```

💾 LocalStorage API

```
// Store data
localStorage.setItem('username', 'John');

// Store objects (must stringify)
const user = { name: 'John', age: 30 };
localStorage.setItem('user', JSON.stringify(user));

// Retrieve data
const username = localStorage.getItem('username');

// Retrieve and parse object
const storedUser = JSON.parse(localStorage.getItem('user'));

// Remove item
localStorage.removeItem('username');

// Clear all
localStorage.clear();

// Check if exists
if (localStorage.getItem('user') !== null) {
    // Item exists
}
```

🔍 Type Checking & Conversion

Type Checking

```
typeof 42                // 'number'
typeof 'hello'           // 'string'
typeof true              // 'boolean'
typeof undefined         // 'undefined'
typeof {}                // 'object'
typeof []                // 'object'
typeof null              // 'object' (quirk!)

// Check for NaN
isNaN('hello')           // true
isNaN(123)               // false
```

Type Conversion

```
// To Number
parseInt('42')           // 42
parseInt('FF', 16)       // 255 (hexadecimal)
parseFloat('3.14')       // 3.14
Number('42')             // 42
+'42'                    // 42 (unary plus)

// To String
String(42)               // '42'
(42).toString()          // '42'
42 + ''                  // '42'

// To Boolean
Boolean(0)               // false
Boolean(1)               // true
!!value                  // Boolean conversion
```

🛠️ Utility Functions

Timers

```
// Execute once after delay
const timeoutId = setTimeout(() => {
    console.log('Executed after 2 seconds');
}, 2000);

// Cancel timeout
clearTimeout(timeoutId);

// Execute repeatedly
const intervalId = setInterval(() => {
    console.log('Every second');
}, 1000);

// Stop interval
clearInterval(intervalId);
```

Console Methods

```
console.log('Normal message');
console.error('Error message');
console.warn('Warning message');
console.info('Info message');
console.table(arrayOfObjects);
console.group('Group');
console.groupEnd();
```

Dialogs

```
// Alert
alert('Welcome!');

// Confirm (returns boolean)
if (confirm('Are you sure?')) {
    // User clicked OK
}

// Prompt (returns string or null)
const name = prompt('Enter your name:', 'Default');
if (name !== null) {
    console.log('Hello, ' + name);
}
```

JSON Operations

```
// Object to JSON string
const user = { name: 'John', age: 30 };
const json = JSON.stringify(user);
// '{"name":"John","age":30}'

// Pretty print
const formatted = JSON.stringify(user, null, 2);

// JSON string to object
const obj = JSON.parse(json);
```

💡 Common Patterns & Best Practices

Form Validation

```
form.addEventListener('submit', (e) => {
    e.preventDefault();

    const formData = new FormData(form);
    const data = Object.fromEntries(formData);

    // Validate
    if (!data.email.includes('@')) {
        alert('Invalid email');
        return;
    }

    // Submit data
    submitForm(data);
});
```

Debouncing (Limit function calls)

```
function debounce(func, delay) {
    let timeoutId;
    return function(...args) {
        clearTimeout(timeoutId);
        timeoutId = setTimeout(() => func.apply(this, args), delay);
    };
}

// Usage
searchInput.addEventListener('input', debounce((e) => {
    searchAPI(e.target.value);
}, 300));
```

Loading Dynamic Content

```javascript
async function loadData() {
    try {
        showLoading();
        const response = await fetch('/api/data');
        const data = await response.json();
        displayData(data);
    } catch (error) {
        showError(error.message);
    } finally {
        hideLoading();
    }
}
```

📚 Quick Reference Card

Task	Method	Example
Select by ID	`getElementById()`	`document.getElementById('header')`
Select by class	`getElementsByClassName()`	`document.getElementsByClassName('btn')`
Select (CSS)	`querySelector()`	`document.querySelector('.btn.active')`
Select all (CSS)	`querySelectorAll()`	`document.querySelectorAll('li')`
Create element	`createElement()`	`document.createElement('div')`
Add child	`appendChild()`	`parent.appendChild(child)`
Remove element	`remove()`	`element.remove()`
Set attribute	`setAttribute()`	`img.setAttribute('src', 'url')`
Add class	`classList.add()`	`el.classList.add('active')`
Add listener	`addEventListener()`	`btn.addEventListener('click', fn)`
Stop bubbling	`stopPropagation()`	`e.stopPropagation()`
Prevent default	`preventDefault()`	`e.preventDefault()`