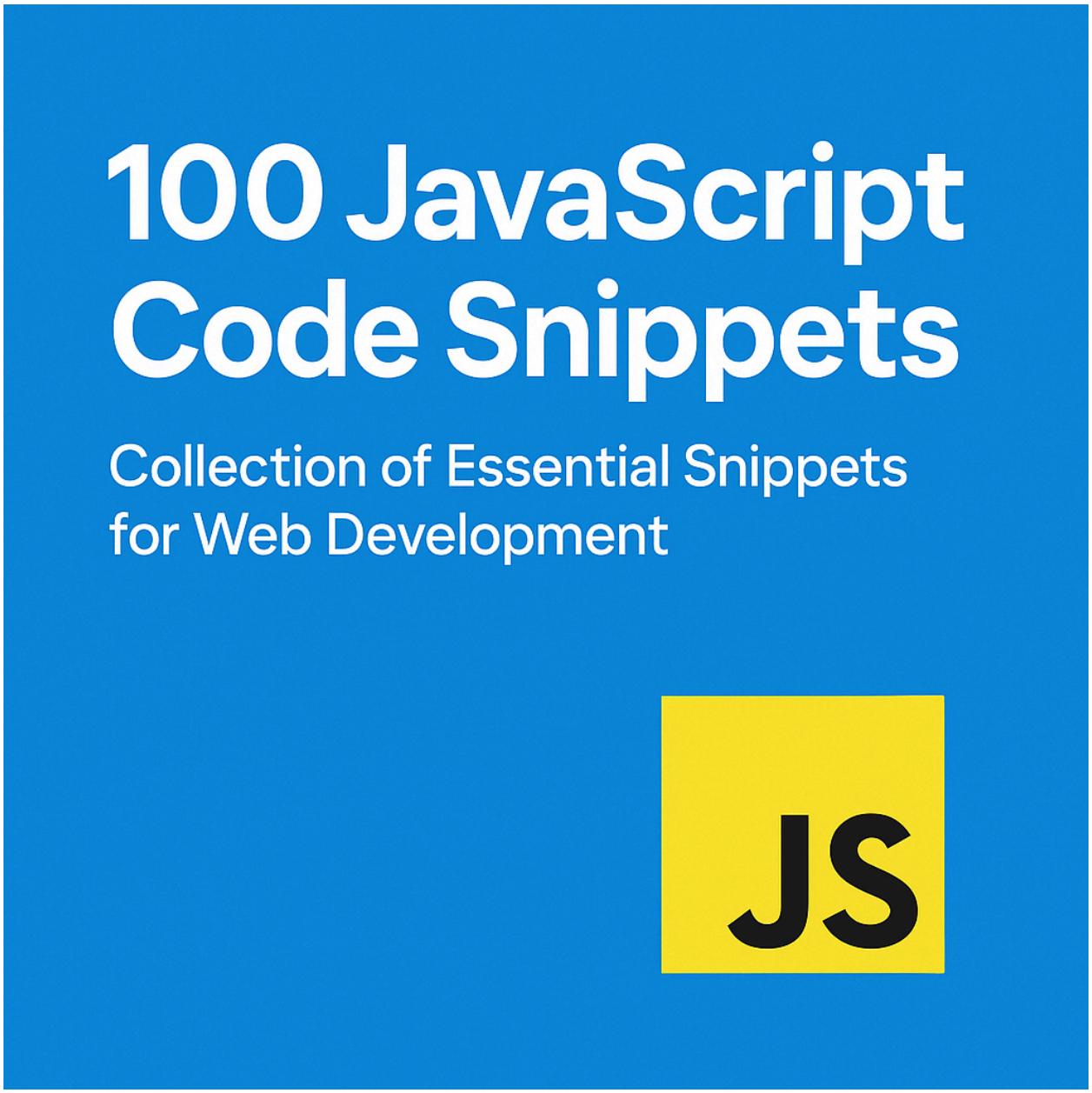
JavaScript Code Snippet Reference (100 Essential Snippets)



---

1. DOM Selection & Manipulation

---

1. Select an Element by ID

```
const element = document.getElementById('myId');
```

Explanation: Selects a single DOM element with the ID `myId`.

---

2. Select Elements by Class Name

```
const elements = document.getElementsByClassName('myClass');
```

Explanation: Returns a live HTMLCollection of all elements with the class `myClass`.

---

3. Select Elements by Tag Name

```
const elements = document.getElementsByTagName('div');
```

Explanation: Returns all `<div>` elements as a live HTMLCollection.

---

4. Select First Matching Element (Query Selector)

```
const element = document.querySelector('.myClass');
```

Explanation: Returns the first element matching the CSS selector.

---

5. Select All Matching Elements (Query Selector All)

```
const elements = document.querySelectorAll('.myClass');
```

Explanation: Returns a static NodeList of all elements matching the selector.

---

6. Change Element Text

```
document.getElementById('demo').textContent = 'Hello World!';
```

Explanation: Sets the text inside an element with ID demo to "Hello World!".

---

7. Change Element HTML

```
document.getElementById('demo').innerHTML =
'<strong>Hello</strong>';
```

Explanation: Replaces the inner HTML content of the element.

---

8. Change Element Style

```
document.getElementById('demo').style.color = 'red';
```

Explanation: Changes the text color of the element to red.

---

9. Add a Class to an Element

```
document.getElementById('demo').classList.add('active');
```

Explanation: Adds the `active` class to the element.

---

10. Remove a Class from an Element

```
document.getElementById('demo').classList.remove('active');
```

Explanation: Removes the `active` class from the element.

---

11. Toggle a Class on an Element

```
document.getElementById('demo').classList.toggle('active');
```

Explanation: Adds the class if it's missing, removes it if it's present.

---

12. Check If Element Has a Class

```
document.getElementById('demo').classList.contains('active');
```

Explanation: Returns `true` if the element has the `active` class.

---

13. Set an Attribute

```
document.getElementById('demo').setAttribute('title', 'Tooltip
text');
```

Explanation: Sets the `title` attribute (hover tooltip) on the element.

---

14. Get an Attribute

```
const attr =
document.getElementById('demo').getAttribute('title');
```

Explanation: Gets the current value of the `title` attribute.

---

15. Remove an Attribute

```
document.getElementById('demo').removeAttribute('title');
```

Explanation: Removes the `title` attribute from the element.

---

16. Create and Append an Element

```
const newDiv = document.createElement('div');
newDiv.textContent = 'New!';
document.body.appendChild(newDiv);
```

Explanation: Creates a `<div>`, adds text, and appends it to the document body.

---

17. Remove an Element

```
const element = document.getElementById('demo');
element.parentNode.removeChild(element);
```

Explanation: Removes the element with ID `demo` from the DOM.

---

18. Clone an Element

```
const clone = document.getElementById('demo').cloneNode(true);
document.body.appendChild(clone);
```

Explanation: Makes a deep copy of the element (including children).

---

19. Insert HTML Adjacent to an Element

```
document.getElementById('demo').insertAdjacentHTML('afterend',
'<p>After!</p>');
```

Explanation: Inserts HTML directly after the element in the DOM.

---

20. Get/Set Input Value

```
// Get

const value = document.getElementById('myInput').value;

// Set

document.getElementById('myInput').value = 'New value';
```

Explanation: Get or set the value of an input field.

---

2. Event Handling

---

21. Add a Click Event Listener

```
document.getElementById('btn').addEventListener('click',
function() {

  alert('Button clicked!');

});
```

Explanation: Runs code when a button is clicked.

---

22. Remove an Event Listener

```
function handleClick() { alert('Clicked!'); }

const btn = document.getElementById('btn');

btn.addEventListener('click', handleClick);

// Remove

btn.removeEventListener('click', handleClick);
```

Explanation: Attaches and then removes a click event handler.

---

23. Event Delegation

```
document.getElementById('list').addEventListener('click',
function(e) {

  if (e.target.tagName === 'LI') {

    e.target.classList.toggle('done');

  }

});
```

Explanation: Handles events for dynamically added child elements.

---

24. Prevent Default Behavior

```
document.getElementById('myForm').addEventListener('submit',
function(e) {
  e.preventDefault();
});
```

Explanation: Prevents the default form submission behavior.

---

25. Stop Event Propagation

```
element.addEventListener('click', function(e) {
  e.stopPropagation();
});
```

Explanation: Stops the event from bubbling up the DOM tree.

---

26. Mouse Enter/Leave Events

```
element.addEventListener('mouseenter', () => { /* ... */ });
element.addEventListener('mouseleave', () => { /* ... */ });
```

Explanation: Runs code when the mouse enters/leaves an element.

---

27. Keyboard Events

```
document.addEventListener('keydown', function(e) {
  if (e.key === 'Enter') {
    console.log('Enter key pressed');
  }
});
```

Explanation: Runs code when the Enter key is pressed.

---

28. Window Resize Event

```
window.addEventListener('resize', () => {
  console.log('Window resized!');
});
```

Explanation: Runs code whenever the browser window is resized.

---

29. Detect When DOM is Ready

```
document.addEventListener('DOMContentLoaded', () => {
  // DOM fully loaded
```

```
});
```

Explanation: Ensures code runs after the DOM is ready, before images/stylesheets may be loaded.

---

30. Double Click Event

```
element.addEventListener('dblclick', function() {
  alert('Element double-clicked!');
});
```

Explanation: Runs code on a double click.

---

3. Arrays & Objects

---

31. Loop Over an Array

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

Explanation: Iterates over each value in an array.

---

32. Map an Array

```
const squared = [1, 2, 3].map(num => num * num);
```

Explanation: Returns a new array with each value squared.

---

33. Filter an Array

```
const evens = [1, 2, 3, 4].filter(num => num % 2 === 0);
```

Explanation: Returns a new array with only even numbers.

---

34. Find an Element in an Array

```
const found = [5, 10, 15].find(num => num > 8);
```

Explanation: Returns the first element greater than 8.

---

35. Check if Array Includes Value

```
[1, 2, 3].includes(2); // true
```

Explanation: Returns `true` if the array contains the value 2.

---

36. Find Index in Array

```
const idx = ['a', 'b', 'c'].indexOf('b');
```

Explanation: Returns the index of `'b'` in the array.

---

37. Sort an Array

```
const nums = [3, 1, 2].sort((a, b) => a - b);
```

Explanation: Sorts numbers in ascending order.

---

38. Remove Duplicates from Array

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

Explanation: Creates a new array with unique values.

---

39. Merge Arrays

```
const merged = [1, 2].concat([3, 4]);
```

Explanation: Combines two arrays into one.

---

40. Spread Operator (Copy Array)

```
const copy = [...[1, 2, 3]];
```

Explanation: Copies all values from the array.

---

41. Loop Through Object Properties

```
const obj = {a:1, b:2};
for (let key in obj) {
  console.log(key, obj[key]);
}
```

Explanation: Iterates through all keys and values in an object.

---

42. Object.keys/Object.values/Object.entries

```
const obj = {a:1, b:2};
Object.keys(obj);    // ['a', 'b']
Object.values(obj);  // [1, 2]
```

```
Object.entries(obj); // [['a', 1], ['b', 2]]
```

Explanation: Useful methods to work with objects.

---

43. Clone an Object (Shallow)

```
const original = {a:1, b:2};
const clone = {...original};
```

Explanation: Copies all enumerable properties into a new object.

---

44. Merge Two Objects

```
const obj1 = {a:1};
const obj2 = {b:2};
const merged = {...obj1, ...obj2};
```

Explanation: Combines properties from both objects.

---

45. Destructure an Object

```
const {a, b} = {a:1, b:2};
```

Explanation: Extracts values from objects into variables.

---

4. Strings

---

46. String Interpolation (Template Literals)

```
const name = 'Alice';
const greeting = `Hello, ${name}!`;
```

Explanation: Embeds variables directly into strings.

---

47. Uppercase/Lowercase a String

```
'hello'.toUpperCase(); // "HELLO"
'HELLO'.toLowerCase(); // "hello"
```

Explanation: Converts a string to all uppercase or lowercase.

---

48. Trim Whitespace

```
'  hello  '.trim();
```

Explanation: Removes whitespace from both ends of the string.

---

49. Replace Substring

```
'foo bar'.replace('foo', 'baz'); // "baz bar"
```

Explanation: Replaces the first occurrence of `foo` with `baz`.

---

50. Check If String Includes Substring

```
'apples and oranges'.includes('apples'); // true
```

Explanation: Checks if the substring exists in the string.

---

51. Split a String Into an Array

```
'apple,banana,pear'.split(',');
```

Explanation: Splits the string into an array by the comma delimiter.

---

52. Join Array into String

```
['a', 'b', 'c'].join('-');
```

Explanation: Combines array elements into a single string, separated by -.

---

53. Get Substring

```
'apple'.substring(1, 3); // "pp"
```

Explanation: Returns the substring from index 1 to 3 (not including 3).

---

54. String to Number

```
const num = Number('123'); // 123
```

Explanation: Converts a string to a number.

---

55. Number to String

```
const str = (123).toString(); // "123"
```

Explanation: Converts a number to a string.

---

5. Utilities

---

56. Generate a Random Number (0 to 1)

```
const rnd = Math.random();
```

Explanation: Generates a random number between 0 (inclusive) and 1 (exclusive).

---

57. Random Integer in Range

```
function getRandomInt(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

Explanation: Returns a random integer between min and max (inclusive).

---

58. Format Number as Currency

```
const price = 9.99;
price.toLocaleString('en-US', {style: 'currency', currency:
'USD'});
```

Explanation: Formats a number as a currency string.

---

59. Debounce Function

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

Explanation: Ensures the function runs only after the specified delay since the last call.

---

60. Throttle Function

```
function throttle(fn, limit) {
  let lastCall = 0;
  return function(...args) {
    const now = Date.now();
    if (now - lastCall >= limit) {
      lastCall = now;
      fn.apply(this, args);
    }
  };
}
```

Explanation: Ensures the function runs at most once every specified interval.

---

61. Copy to Clipboard

```
function copyText(text) {
  navigator.clipboard.writeText(text);
}
```

Explanation: Copies a string to the user's clipboard.

---

62. Delay/Wait (Promise-based)

```
function wait(ms) {
  return new Promise(resolve => setTimeout(resolve, ms));
}
```

Explanation: Returns a promise that resolves after `ms` milliseconds.

---

63. Check if Variable is Array

```
Array.isArray([1,2,3]); // true
```

Explanation: Checks if the variable is an array.

---

64. Deep Clone an Object (JSON)

```
const deepClone = JSON.parse(JSON.stringify(obj));
```

Explanation: Deep clones objects (not suitable for functions, dates, etc.).

---

65. Get Current Date and Time

```
const now = new Date();
```

Explanation: Creates a new Date object for the current date and time.

---

66. Format Date as String

```
const now = new Date();
const dateStr = now.toLocaleDateString();
```

Explanation: Formats the current date as a locale-specific string.

---

67. Get Query Parameter from URL

```
const params = new URLSearchParams(window.location.search);
const value = params.get('key');
```

Explanation: Gets the value of a query parameter from the URL.

---

68. Scroll to Top of Page

```
window.scrollTo({ top: 0, behavior: 'smooth' });
```

Explanation: Smoothly scrolls the page to the top.

---

69. Shuffle Array

```
function shuffle(array) {
  return array.sort(() => Math.random() - 0.5);
}
```

Explanation: Randomly reorders the elements of an array.

---

70. Capitalize First Letter

```
function capitalize(str) {
  return str.charAt(0).toUpperCase() + str.slice(1);
```

```
}
```

Explanation: Returns the string with the first letter in uppercase.

---

71. Pad String

```
'5'.padStart(3, '0'); // "005"
```

Explanation: Pads the string to the target length with zeros.

---

72. Escape HTML

```
function escapeHTML(str) {
  return str.replace(/[&<>"']/g, m => ({
    '&':'&amp;', '<':'&lt;', '>':'&gt;', '"':'&quot;',
"'":'&#39;'
  })[m]);
}
```

Explanation: Escapes HTML special characters in a string.

---

6. Local Storage & Cookies

---

73. Set Item in Local Storage

```
localStorage.setItem('username', 'Alice');
```

Explanation: Stores the key-value pair in browser local storage.

---

74. Get Item from Local Storage

```
const user = localStorage.getItem('username');
```

Explanation: Retrieves the value associated with the key.

---

75. Remove Item from Local Storage

```
localStorage.removeItem('username');
```

Explanation: Deletes the key-value pair from local storage.

---

76. Clear Local Storage

```
localStorage.clear();
```

Explanation: Removes all data from local storage.

---

77. Set a Cookie

```
document.cookie = "user=Alice; expires=Fri, 31 Dec 9999 23:59:59
GMT";
```

Explanation: Sets a cookie for the user.

---

78. Get a Cookie by Name

```
function getCookie(name) {
  const match = document.cookie.match(new RegExp('(^| )' + name
+ '=([^;]+)'));
  return match ? match[2] : null;
}
```

Explanation: Retrieves a cookie value by its name.

---

7. Forms & Validation

---

79. Check If Form is Valid

```
const form = document.getElementById('myForm');
```

```
if (form.checkValidity()) {
  // Valid
}
```

Explanation: Checks if all form fields meet their constraints.

---

80. Serialize Form Data

```
function serializeForm(form) {
  return new FormData(form);
}
```

Explanation: Returns a FormData object containing form data for submission.

---

81. Custom Form Validation Message

```
const input = document.getElementById('myInput');
input.setCustomValidity('Custom error message');
```

Explanation: Sets a custom error message for a form field.

---

82. Prevent Form Submission on Enter

```
form.addEventListener('keydown', function(e) {
  if (e.key === 'Enter') e.preventDefault();
});
```

Explanation: Stops form from submitting when pressing Enter.

---

8. Animation & Effects

---

83. Simple Fade In

```
function fadeIn(el) {
  el.style.opacity = 0;
  el.style.display = 'block';
  let last = +new Date();
  const tick = function() {
    el.style.opacity = +el.style.opacity + (new Date() - last) /
400;
    last = +new Date();
    if (+el.style.opacity < 1) requestAnimationFrame(tick);
  };
  tick();
}
```

Explanation: Gradually fades in an element.

---

84. Simple Fade Out

```
function fadeOut(el) {
  el.style.opacity = 1;
  let last = +new Date();
  const tick = function() {
    el.style.opacity = +el.style.opacity - (new Date() - last) /
400;
    last = +new Date();
    if (+el.style.opacity > 0) requestAnimationFrame(tick);
    else el.style.display = 'none';
  };
  tick();
}
```

Explanation: Gradually fades out an element.

---

85. Smooth Scroll to Element

```
document.getElementById('myLink').addEventListener('click',
function(e) {
```

```
  e.preventDefault();

  document.getElementById('target').scrollIntoView({ behavior:
'smooth' });

});
```

Explanation: Smoothly scrolls the page to the target element.

---

86. Toggle Visibility

```
function toggleVisibility(id) {

  const el = document.getElementById(id);

  el.style.display = (el.style.display === 'none' ? 'block' :
'none');

}
```

Explanation: Toggles an element's visibility.

---

87. Animate Height Toggle

```
function toggleHeight(el) {

  el.style.transition = 'height 0.3s';

  el.style.height = el.style.height === '0px' ? '100px' : '0px';

}
```

Explanation: Animates an element's height.

---

9. Networking & APIs

---

88. Fetch Data from API

```
fetch('https://api.example.com/data')
  .then(res => res.json())
  .then(data => console.log(data));
```

Explanation: Fetches data from a URL and logs the result.

---

89. Post Data to API

```
fetch('https://api.example.com/data', {
  method: 'POST',
  headers: {'Content-Type': 'application/json'},
  body: JSON.stringify({name: 'Alice'})
})
  .then(res => res.json())
  .then(data => console.log(data));
```

Explanation: Sends JSON data to a URL using POST.

---

90. Handle Fetch Errors

```
fetch('https://api.example.com/data')
  .then(res => {
    if (!res.ok) throw new Error('Network error');
    return res.json();
  })
  .catch(err => console.error(err));
```

Explanation: Handles failed fetch requests gracefully.

---

91. Load JSON Dynamically

```
async function getData(url) {
  const res = await fetch(url);
  return await res.json();
}
```

Explanation: Asynchronously loads JSON data.

---

92. Parse JSON String

```
const obj = JSON.parse('{"name":"Alice"}');
```

Explanation: Converts a JSON string to an object.

---

93. Stringify Object to JSON

```
const str = JSON.stringify({name: 'Alice'});
```

Explanation: Converts an object to a JSON string.

---

10. Browser & Window

---

94. Detect Browser Language

```
const lang = navigator.language || navigator.userLanguage;
```

Explanation: Gets the browser's preferred language setting.

---

95. Detect if Mobile Device

```
function isMobile() {
```

```
  return /Mobi|Android/i.test(navigator.userAgent);
}
```

Explanation: Checks if the user is on a mobile device.

---

96. Redirect to Another URL

```
window.location.href = 'https://example.com';
```

Explanation: Redirects the browser to a new URL.

---

97. Reload the Page

```
window.location.reload();
```

Explanation: Reloads the current web page.

---

98. Open a New Window/Tab

```
window.open('https://example.com', '_blank');
```

Explanation: Opens a new browser tab or window.

---

99. Get Current URL

```
const url = window.location.href;
```

Explanation: Retrieves the current page's URL.

---

100. Get Page Scroll Position

```
const x = window.scrollX;
const y = window.scrollY;
```

Explanation: Gets the horizontal and vertical scroll position of the page.

Summary

This collection is a **ready-to-use, categorized resource** for any frontend JavaScript developer working in vanilla JS.