JavaScript Cheat Sheet in Simple Terms

Basics and Fundamentals

JS HOME: Introduces JavaScript, a language that adds interactivity to websites.

Example :

```
<script> alert('Hello, world!'); </script>
```

JS Introduction: Explains JavaScript's role in enhancing user interaction on web pages.

Inline Example :

```
<button onclick="alert('Clicked!')">Click Me!</button>
```

Internal Example :

```
<script>function sayHello() { alert('Hello!');}</script>
```

External Example :

```
<script src='myscripts.js'></script>
```

JS Output : Shows how to display data using JavaScript.

Example :

```
Console output: console.log('Hello World');
```

```
Writing to HTML: document.write('Hello World');
```

```
Alerts: alert('Hello World');
```

JS Statements : Describes how to write instructions for the browser.

Example :

```
var x = 5; var y = 6; var z = x + y;
```

JS Comments : Explains how to add explanatory notes in JavaScript code.

Example :

```
Single-line: // This is a comment
```

```
Multi-line: /* This is a multi-line comment */
```

JS Variables : Introduction to storing data values.

Example :

```
var name = 'John'; // Function scope
```

```
let age = 30; // Block scope
```

```
const birthday = 1990; // Immutable and block scoped
```

JS Let & JS Const : Discusses block-scoped variables, with const providing immutability.

Example :

```
let score = 150;
```

```
const pi = 3.14;
```

JS Operators : Covers operators for arithmetic, assignment, and comparison.

Example :

```
Arithmetic: x + y, x - y
```

```
Assignment: x = y, x += y
```

```
Comparison: x == y, x > y
```

Advanced Programming

JS Functions & JS Objects: Defines functions for reusable code and objects for organizing data.

Example :

```
Function: function myFunc() { return 'Hello'; }
```

```
Object: var car = {make: 'Ford', model: 'Mustang', year: 1969};
```

JS Object Properties & Methods: Manipulating properties and methods of objects.

Example :

```
Properties: console.log(car.make);
```

```
Methods: car.start = function() { console.log('Car starts'); };
```

JS Object Display & Constructors: Showing properties of objects and using constructors to create instances.

Example :

```
Display: for (var key in car) { console.log(key + ': ' + car[key]); }
```

```
Constructor: function Car(make, model, year) { this.make = make; this.model = model; this.year = year; } var myCar = new Car('Ford', 'Mustang', 1969);
```

Handling Data and Events

JS Events & JS Strings: Interacting with DOM events and handling string operations.

Example :

```
Events: element.onclick = function() { alert('Clicked'); };
```

```
Strings: var txt = 'Hello'; console.log(txt.length);
```

JS String Methods & Search: Methods to process strings and search text within them.

Example :

```
Methods: var x = 'Hello, world!'; console.log(x.indexOf('world'));
```

```
Search: var s = 'Hello, world!'; console.log(s.includes('world'));
```

JS String Templates: Using template literals for easy embedded expressions.

Example :

```
var name = 'John'; var greeting = `Hello, ${name}!`;
```

Numbers and Arrays

JS Numbers & JS Bigint: Handling both standard and very large numbers.

Example :

```
Numbers: var x = 123;
```

```
Bigint: const bigint = 1234567890123456789012345678901234567890n;
```

JS Arrays & JS Array Methods: Techniques for managing lists of data.

Example :

```
Arrays: var fruits = ['Apple', 'Banana', 'Cherry'];
```

```
Array Methods: fruits.forEach(function(item, index) { console.log(item, index); });
```

Date, Time, and Logic

JS Dates & Date Formats: Managing and formatting date values.

Example :

```
Dates: var today = new Date();
```

```
Formats: console.log(today.toISOString());
```

JS Math & JS Random: Performing mathematical calculations and generating random numbers.

Example :

```
Math: console.log(Math.sqrt(16));
```

```
Random: console.log(Math.random());
```

JS If Else & JS Switch: Using conditional logic to control program flow.

Example :

```
If Else: if (x > 0) { console.log('Positive'); } else {console.log('Negative'); }
```

```
Switch: switch(x) { case 0: console.log('Zero'); break; default: console.log('Non-zero'); }
```

Looping and Data Structures

JS Loop For, In, Of & While: Different ways to repeat actions in JavaScript.

Example :

```
For: for (var i = 0; i < 10; i++) { console.log(i); }
```

```
For in: for (var key in object) { console.log(key); }
```

```
For Of: for (var item of array) { console.log(item); }
```

```
While: while (x < 10) { x++; }
```

JS Break & JS Iterables: Managing loop interruptions and iterating over data structures.

Example :

```
Break: for (var i = 0; i < 10; i++){ if i === 5){break; } console.log(i); }
```

```
Iterables: let iterable = [10, 20, 30]; for (let value of iterable) { console.log(value); }
```

JS Sets & JS Maps: Utilizing sets for unique item collections and maps for key-value pairs.

Example :

```
Sets: var mySet = new Set([1, 2, 3]); mySet.add(4); console.log(mySet.has(3));
```

```
Maps: var myMap = new Map(); myMap.set('key', 'value'); console.log(myMap.get('key')); myMap.delete('key');
```

JS Sets & JS Maps: Utilizing sets for unique item collections and maps for key-value pairs.

Example :

```
Sets: var mySet = new Set([1, 2, 3]); mySet.add(4); console.log(mySet.has(3));
```

```
Maps: var myMap = new Map(); myMap.set('key', 'value'); console.log(myMap.get('key')); myMap.delete('key');
```

Advanced JavaScript Features

JS Typeof & JS Destructuring: Identifying data types and extracting multiple properties or values.

Example :

```
Typeof: console.log(typeof 'Hello');
```

```
Destructuring: var[a, b] = [1, 2]; var{x, y} = {x: 10, y: 20};
```

JS RegExp & JS Errors: Using regular expressions for pattern matching and handling errors.

Example :

```
RegExp: var pattern = /ab+c/; var result = pattern.exec('abc');
```

```
Errors: try { null.f(); }catch (e) { console.log(e); }
```

JS Scope & JS Hoisting: Understanding the accessibility of variables and how declarations are moved to the top.

Example :

```
Scope: function test() { var local = 'scoped'; } console.log(local); // Error
```

```
Hoisting: console.log(value); var value = 'Hoisted';
```

JS Strict Mode & JS this Keyword: Enforcing stricter parsing and execution of JavaScript, and understanding context.

Example :

```
Strict Mode: 'use strict'; x = 3.14; // Throws error
```

```
This: var obj = {a: 10, b: function() { console.log(this.a); }}; obj.b();
```

JS Arrow Function & JS Classes: Modern syntax for functions and using classes for object-oriented programming.

Example :

```
Arrow Function: var add = (x, y) => x + y;
```

```
Classes: class Rectangle { constructor(height, width){ this.height = height; this.width = width; }}
```

Web Development and APIs

JS Modules & JS JSON: Organizing code and handling data in JSON format.

Example :

```
Modules: export function hello() { console.log('Hello!'); } import { hello } from './module.js';
```

```
JSON: var jsonStr = '{"name":"John","age":30}'; var obj = JSON.parse(jsonStr); console.log(obj.name);
```

JS Debugging & JS Style Guide: Techniques for troubleshooting code and maintaining a clean coding style.

Debugging Example:

```
Use console.log, breakpoints, and network tabs in developer tools.
```

JS Best Practices & JS Mistakes: Recommendations for effective coding and avoiding common pitfalls.

Example :

```
Best Practices: Use let and const for variable declarations, always handle errors in promises with .catch().
```

```
Mistakes: Avoid common mistakes like forgetting a semicolon, using == instead of ===.
```

JavaScript Versions and History

JS Versions & JS History: Overview of JavaScript's development since its inception.

Historical Note :

```
JavaScript was created by Brendan Eich in 1995 and has evolved significantly, with major updates outlined in ES5 (2009), ES6 (2015), and subsequent yearly updates through ES2024.
```

JS HTML DOM & JS Browser BOM: Interactions with the HTML Document Object Model and the Browser Object Model.

DOM Methods:

```
document.getElementById(), element.innerHTML = 'new content';
```

BOM Examples:

```
Window: window.location = 'http://www.example.com';
```

```
Screen: console.log(window.screen.width);
```

JS Navigator & JS Popup Alert: Using browser information and managing pop-up alerts.

Navigator:

```
console.log(navigator.userAgent);
```

Popup Alert:

```
alert('This is an alert box!');
```

JS Timing & JS Cookies: Setting timers and managing cookies in the browser.

Timing Example:

```
setTimeout(function() { alert('Hello'); }, 3000);
```

Cookies:

```
document.cookie = "username=John Doe";
```

JavaScript Web APIs and AJAX

JS Web APIs & JS AJAX: Engaging with web APIs for enhanced functionality and performing asynchronous server requests.

Fetch API Example :

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

AJAX Example :

```
var xhr = new XMLHttpRequest(); xhr.open('GET', 'file.txt', true); xhr.send();
```

JavaScript Graphics and Visualization

JS Graphics & JS Canvas: Creating and managing graphics on the web.

Canvas Example :

```
var canvas = document.getElementById('myCanvas'); var ctx = canvas.getContext('2d'); ctx.fillStyle = 'red'; ctx.fillRect(10, 10, 50, 50);
```

JS Chart.js & JS Google Chart: Using JavaScript libraries to create charts and visualizations.

Chart.js Example:

```
var myChart = new Chart(ctx, {type: 'line', data: data, options: options});
```

Google Chart Example:

```
google.charts.load('current', {'packages':['corechart']}); google.charts.setOnLoadCallback(drawChart);
```

Advanced Topics and Usage

JS Examples & JS Quiz: Practical coding examples and quizzes to enhance learning.

Example Usage :

```
Examples provided in documentation or tutorials.
```

Quiz Example :

```
Interactive coding challenges or multiple-choice questions.
```

JS Interview Prep & JS Bootcamp: Preparing for technical interviews and comprehensive learning programs.

Interview Prep :

```
Typical questions about JavaScript concepts like closures, hoisting, and event handling.
```

Bootcamp:

```
Programs designed to teach JavaScript from the ground up in a structured environment.
```

JavaScript References and Learning Resources

JS References: Comprehensive documentation on JavaScript's capabilities.

Details :

```
In-depth descriptions of JavaScript's built-in objects, methods, and properties.
```