|  |  |  |
| --- | --- | --- |
| **What** | **Why- How** | **Code Example** |
| Data Types |  |  |
| Var/let/const |  |  |
| Array- | An array is a special variable, which can hold more than one value: | let array = [ 1, "one", 2, "two", 3, "three"]; |
| Splice Method | The splice() method adds new items to an array. | array.splice(3, 4, "TWO", "THREE"); |
| Slice Method | The slice() method slices out a piece o an array. | let newarray = array.slice(3); |
| Push Array | It adds given value in the end | array.push(4, "four") |
| Shift Array | it removed the first element from an array | array.shift() |
| Unshift Array | adds new element in an array at first place | array.unshift(5, "five"); |
| Array length | to get array length | (array.length); |
| Array pop |  | (array.pop()); |
| New Array |  | (newarray); |
| Empty Array |  | let emptyarray = [];  emptyarray[0] = 6, "six";  emptyarray[1] = 7, "seven"; |
| Concat method | merge 2 array and create  a new array | let concatArray = array.concat(emptyarray); |
| Array Sorting | sorts array alphabetically/numerically | sortarray.sort(); |
| Array Reverse | The reverse() method reverses the elements in an array | sortarray.reverse(); |
| This Keyword | This keyword refers to an object. | var employee =  {      Name: "Hardik",      DOJ: 10-12-2022,      employeedetails: function() {          return this.Name + this.DOJ;     }  };  console.log(employee.employeedetails()); |
| Strict Mode | In a function, in strict mode, this is undefined. Strict mode makes it easier to write "secure" JavaScript. Strict mode changes previously accepted "bad syntax" into real errors.  As an example, in normal JavaScript, mistyping a variable name creates a new global variable. In strict mode, this will throw an error, making it impossible to accidentally create a global variable.  In normal JavaScript, a developer will not receive any error feedback assigning values to non-writable properties.  In strict mode, any assignment to a non-writable property, a getter-only property, a non-existing property, a non-existing variable, or a non-existing object, will throw an error. | "use strict"; myFunction();  function myFunction() {   y = 3.14;   // This will also cause an error because y is not declared } |
| AJAX | AJAX is a developer's dream, because you can:  Update a web page without reloading the page  Request data from a server - after the page has loaded  Receive data from a server - after the page has loaded  Send data to a server - in the background |  |
| Jquery $ sign | Jquery $ sign is just a valid JavaScript identifier which is used as an alias for jQuery. |  |
| Https request in ajax | The XMLHttpRequest object is used to exchange data with a server.To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object: |  |
| Status in AJAX HTML | HTTP response status codes indicate whether a specific HTTP request has been successfully completed. |  |
| Functions |  |  |
| Understanding of Asycnhronous behaviour |  |  |
| For loop |  |  |
| Objects |  |  |
| if else |  |  |
| math.random |  |  |
| Regular expressions | Regular expressions are patterns used to match character combinations in strings. In JavaScript, regular expressions are also objects. These patterns are used with the | These patterns are used with the exec() and test() methods of RegExp, and with the match(), matchAll(), replace(), replaceAll(), search(), and split() methods of String. |
| pure and impure functions |  |  |
| constructor  Function | In a constructor function this does not have a value. It is a substitute for the new object. The value of this will become the new object when a new object is created. |  |
| Eventloop | JavaScript has a runtime model based on an event loop, which is responsible for executing the code, collecting and processing events, and executing queued sub-tasks. This model is quite different from models in other languages like C and Java. |  |
| Arrow function | It gets shorter! If the function has only one statement, and the statement returns a value, you can remove the brackets and the return keyword: | let myFunction = (a, b) => a \* b; |
| window.onload | This function execute a JavaScript immediately after a page has been loaded: |  |
| Fetching API |  | let p = fetch("https: api.chucknorris.io/jokes/random%22)  p.then((response) => {    return response.json()  }).then((value2) => {    console.log(value2)    const jokeData = value2.value;    console.log(jokeData);  }) |
| async and await | async and await make promises easier to write" async makes a function return a Promise await makes a function wait for a Promise | const getData = async() => {    var y = await "Hello World";    console.log(y);  }  console.log(1);  getData();  console.log(2); |
| Promises | - Promises are used to handle asynchronous operations in JavaScript. They are easy to manage when dealing with multiple asynchronous operations where callbacks can create callback hell leading to unmanageable code.  A Promise has four states:  fulfilled: Action related to the promise succeeded  rejected: Action related to the promise failed  pending: Promise is still pending i.e. not fulfilled or rejected yet  settled: Promise has fulfilled or rejected  Benefits of Promises  Improves Code Readability  Better handling of asynchronous operations  Better flow of control definition in asynchronous logic  Better Error Handling | var promise = new Promise(function(resolve, reject){        do something  }); |
| iterators and generators |  |  |
| differnce between map and for each |  |  |
|  |  |  |
|  |  |  |