JAVASCRIPT NOTES

What is Javascript?

Javascript is High level, object oriented, multi paradigm programming language.

* A programming language = is a tool that allows us to write code that will instruct a computer to do something.
* Prototype based object oriented.
* Interpreted/just in time compiled – compiling happens in JS engine.
* Dynamic typed – no data type of variables are defined, types are known at runtime. Data type can be changed automatically as well.
* Single threaded – only do one thing at a time.(thread is where instruction is executed)
* Garbage collected programming language with– removed old unused object which are not in use hence do not clog up space .i.e clean our memory.
* first class functions – functions are treated as regular variables, we can pass functions into another function and return them from functions.
* Non-blocking event loop concurrency model(handles multiple task at same time) – using event loop, long running task are executed in background and put them on thread once they are finished.
* High level = we don’t have to think a lot of stuff that we don’t have to think about example managing computer memory while it runs the code.
* Object oriented = based on objects for storing data.
* Multi paradigm = we can use all style of programming such as imperative and declarative. (paradigm = an approach and mindset of structuring code, which will direct your coding style and technique)

What do we use Javascript for? What is the role of JavaScript in Web development ?

Allows Developers to add dynamic and interactive effects to any web page.

Use to manipulate contents or CSS.

Load data from remote servers and build entire application in browser which then called web application.

Example of Dynamic effect?

Spinner, display data on hovering, showing box after clicking, etc.

Values?

A piece of data, it is the most fundamental piece of information in programming language.

Note - Value can be primitive or Object.

Variables?

Values are stored in a variable which can be used over and over again.

We can change the value of variable.

A variable in JavaScript is a named container for storing values, like numbers, strings, objects, or arrays, that can be reassigned, and used throughout the code.

Let firstName = “Shilpa” //declaring a variable

firstName is variable and Value is Shilpa.

Data Types?

Value has data types NOT variables!!

Data types in JavaScript define the data type that a variable can store.

There are 7 PRIMITIVE types of data types –

1. Number – Floating point number, i.e. They have decimal and integers. All numbers are of type Numbers.
2. String – Sequence of characters, used for text. Put them in “”, ‘’.
3. Boolean – Logical type that can only be true or false. Used for taking actions.
4. Undefined – Value taken by variable which is not yet defined. Value doesn’t not exist in compiler. It is a global object.
5. Null – Contains no value. Intentional absence of value.
6. Symbol(ES2015) – Value that is unique and cannot be changed.
7. BigInt(ES2020) – Larger integers that Number type cannot hold.

NOTE : Numbers are stored internally as 64 bits(0,1) but only 53 are used to store digits themselves rest are used to store decimal position and sign. Hence there is restriction/limit to store number hence BIGINT is used to store large numbers.

Javascript has dynamic typing – We do not have to define data type of value stored in variables, it is automatically.

Let, Const, Var

Used to declare variables

|  |  |  |
| --- | --- | --- |
| Let | Var | Const |
| ES 6 | Old way declaring variable. | ES 6 |
| Declare variable that can change later. i.e. during the execution of our program./  Mutate the variable. | Declare variable that can change later. i.e. during the execution of our program./  Mutate the variable. | Declare variable that cannot change later. i.e. during the execution of our program./  Immutate the variable. |
| Can declare empty let variable. |  | Cannot declare empty const variable. |
|  |  |  |

Basic Operators

operator – allow us to transform value or combine multiple value.

Mathematical, comparison, logical, assignment operator.

Mathematical operator - +, -, %, \*

Comparison – Boolean value as a result, > , < , >= , <=

Assignment - = , ++, --, +=, -=, \*= etc.

Ques . How does JS know to do maths first or comparison first?

Ans. Operator Precedence – order in which way operators are executed. There is an operator Precedence table defined in Js. We can find that on MDN web docs.

Strings and Template Literals

Template literals – can assemble multiple pieces into one final string. Can use to create muti line string.

Old way of multiline was \n\

` - this is a template literal.

IF/ else Statement

If /else – It is a controlled structured. The if...else statement executes a statement if a specified condition is truthy. If the condition is falsy, another statement in the optional else clause will be executed.

Nan – Js gives Not a number (Nan) when number fails to produce a number. Nan is an invalid number

Type Conversion and Type Coercion

|  |  |
| --- | --- |
| **Type Conversion** | **Type Coercion** |
| When we manually convert from one type to another | When Js automatically convert types behind the scenes for us. |
| Happen Explicitly, we convert from one type to another. | Happens Implicitly, hidden from us. |
|  | When operator is dealing with values that have 2 types, then Js will convert one value into another value, so that operation can be executed. |
|  | Conversion to Boolean is always Coercion. |
|  | This happens in 2 scenarios – When using logical operator and in condition of if / else statement. |

Truthy and Falsy values

Truthy – Values which are not exactly true but will become true when we try to convert them in Boolean.

Ex - Any number that is not 0, string, and not an empty string will be converted to truth when we attempt to convert them into a Boolean.

Note – values that are not falsy are truthy!!

Falsy - Values which are not exactly false but will become false when we try to convert them in Boolean.

There 5 falsy values –

* 0
* Empty string
* Undefined
* Null
* Nan

Equality operator : == vs ===

|  |  |
| --- | --- |
| **==** | **===** |
| Return boolean value | Return boolean value |
|  | Both LHS and RHs should be same |
| Loose equality operator | Strict equality operator |
| Does type Coercion | Does not type Coercion |
| != - not equal | !== - not equal |

Note – always use === for clean code

Boolean logic

Branch of CS which use true and false values to solve complex logical problems.

|  |  |  |
| --- | --- | --- |
| **AND** | **OR** | **NOT** |
| Only if both the values are true then operation is true. | If only one value is true the whole operation is true, but if both are false whole operation is false. | Inverts the value |
|  |  | Has higher precedence over And & OR |
| && | || | ! |

Switch Statement

The switch statement evaluates an expression. The value of the expression is then compared with the values of each case in the structure.

­Switch uses strict. ‘===’ // designed for equality and not for comparison.

Without Break code will simply continue down the structure.

It is a more organized and concise alternative to using multiple if-else statements

Statement and Expression

Expression – produces a value.

Operator is an expression

In template literal ${} we can have Expression not Statements.

Statement – a bigger piece of code that is executed and which does not produce a value on itself and we can compare this to normal spoken language.

Line that ends in “;” is a statement

If/else is a statement.

Note – Declaration doesn’t produce a value

Conditional Operator/Ternary operator (?)

Allows us to write something similar to if/else statement but all in one line.

Use to take quick decision.

It has 3 parts – condition, if, else.

Activating strict mode

Special mode we can activate in Js to write safer code, i.e. we can avoid accidental errors using this.

We can see visible errors.

‘use strict’;

This above has to be very first statement.

Functions

A piece of code that we can reuse in our code.

It can receive data, return data that we can use for something else in the program.

Function Declaration vs function Expression

|  |  |
| --- | --- |
| Function Declaration | function Expression |
| Function name defined with function keyword | Function name stored in a variable. |
| function shilpa(){} | Const shilpa = function(){} |
|  | Function without a name = anonymous function. |
| We can call Function Declaration before it is defined in the code. | We cannot call Function Declaration before it is defined in the code. |

Arrow Function

Introduced in ES6

Special form of function expression that is shorter and use for faster write.

Return happens implicitly( no need to write return word) in single in code.

Arrow function do not get “This” keyword.

Array

Is a data structure that enables [storing a collection of multiple items under a single variable name](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/Arrays), and has members for [performing common array operations](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array#examples).

Order of value matter when we retrieve the data/ access that elements.

Starts with 0,1,2,3….

Array are mutable even though they are declared with const but we cannot replace the entire array.

Array Method

push = add element at the end of an array. Push is a function and return a value. It returns length.

unshift = add element at beginning of an array, also returns length.

pop = removes last element from an array.

shift = removes first element from array.

Note – Below both are Case Sensitive and checks for Strict equality

indexOf = At which position element is in the array.

Includes = will return true if element is in the array and false if element is not in the array.

Objects

A data structure which has key value pair. Group together different data that belong together.

Order of value doesn’t matter when we retrieve.

Keys are called properties as well.

“{“are called object literals.

When to use Dot notation and Bracket notation ?

Bracket – when we need to first compute property name.

Dot – otherwise use Dot.

Object Methods

Function attached to an object is called Method like calAge().

|  |  |
| --- | --- |
| Objects | Arrays |
| Order of value doesn’t matter | Order of value matters that is how we access |
| used for unstructured data | Use for more ordered data |
|  |  |
|  |  |

Iteration for loop

Allow us to automate repetitive tasks, i.e. tasks we have to perform over and over again.

Looping array, breaking and continuing

Array is 0 based hance for loop starts with i =

Continue – to exit current iteration of loop and continue the other one.

Break – to completely terminate the whole loop.

While loop

Loop will run while the condition is true.

While loop does not need a counter( let i = 1 or let i = 0) it just needs a condition to stay true for it to keep running.

What is DOM and Dom Manipulation?

Document Object Model – structured representation of HTML Documents. Allows JS to access HTML elements and styles to manipulate them.(change text, attribute, styles), Basically a connection point between HTML and JS code.

Created by browser as soon as HTML page loads

The DOM tree Structure

Starts with Document – entry point to DOM.(document.querySelector)

Whatever is in HTML has to be in DOM.

DOM is part of webAPI(library the browser implements), webAPI interacts with JS.

QuerySelector – when we use this which has multiple selector it only chooses first one.

QuerySelectorAll – it chooses all selector.

KeyBoard events

Keypress event are global events as they do not happen at one specific event, they happen at whole document.

3 type of event on keyboard events –

1. Keydown –as soon as we just press a key.
2. Keypress – continuously as we press the key
3. Keyup – lift our finger off the keyboard

When to use QuerySelector vs getelementById

If you want your code to be the most efficient and effective code, then when you are trying to retrieve an id then you should use getElementById. If you are trying to retrieve a class or tag then you should use querySelector.

State Variable

state behaves more like a snapshot. Setting it does not change the state variable you already have, but instead triggers a re-render.

Js engine and runtime

Js engine (example - google Ve8 engine) - is program that runs Js code.

JS engine componenets –

Call stack and heap

1. Call stack –

Where code is executed using execution context.

1. Heap – unstructured memory pool which stores all the objects that our memory needs.

Difference between compilation and Interpretation

|  |  |
| --- | --- |
| Compilation | Interpretation |
| Entire code is converted into machine code at once and written to binary file that can be executed by a computer. | Runs through the source code(machine code) and executes it line by line. |
| 2 steps : compiled first then executed the whole compiled code after a while. | 1 step : Execution line by line. |
|  | Js is Interpretation language. |
|  |  |
|  |  |
|  |  |
|  |  |

Js now uses JIT(just in time) : Entire code is converted into machine code at once and the executed immediately.

JIT Compilation of JS

When code enters the engine:

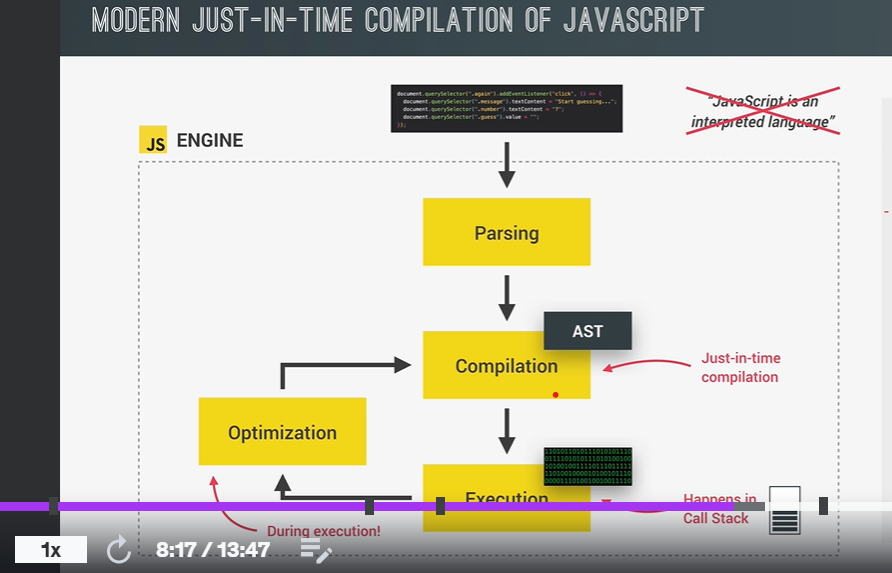
1. Parse the code : read the code, read/parse the code into data structure AST(abstract syntax tree). The code is parse into meaningful language like const and function keyword and then saving these into tree in structured way and checks if there are any syntax error.

This tree is used later for generating machine code.

1. Compilation – Takes AST and compiles it into machine code.
2. Execution – Execution happens right away.(execution happens in Js engine in stack)

Note – when execution happens an unoptimzed version of machine code is made in the beginning so it can start executing as early as possible. Then in background this code is optimized and compiled in already running program execution. Unoptimzed is swapped with optimized and this keeps happening over and over.

This happens inside special thread in Js engine(V8 engine)



JS runtime in browser

Js runtime is big container includes things we need to run JS code.

To run Js we need webAPI - functionalities provided to the engine which are not part of JS language itself, Js gets access to these API through global window object.

JS runtime includes CALLBACK QUEUE – it is Datasctructure that contains all the callback function that are ready to be executed like we attach event handling functions to DOM elements like a button to react to certain events(click). These event handling functions are called call back functions(click, timer).

When event happens like click the call back function will be called.

Callback function which will put in callback queue, when call stack is empty the callback function is put in call stack for execution this is called Event loop.

How is JS code executed ??

Execution happens in Callstack.

When compilation happens, global execution context is created for top level code(code not inside any function).

Execution context is a piece of environment where Js is executed.

There is always only 1 global execution context – default context created for code that is not inside any function, i.e where top level code executes.

When top level code is executed, functions/metods start to executes.

For every function/method call every new execution context is created and waits for callback using event loop.

Inside execution context –

1. Variable environment –

Let, var, const,

Functions

Argument objects – this has all the arguments that were passed into the function of the particular execution context.

1. Scope chain- consists reference to variables that are located outside of current function.
2. This keyword –

NOTE – arrow functions do not get This keyword and Argument objects

CALL stack –

Execution context get stack on each other to keep track of which execution is taking place.

The top of stack execution is currently running. When its done it will be removed from stack then it will go to previous one.

Scoping and scope chain

Scoping – How program variables are organized and accessed.

Lexical scoping: scoping controlled by placement of functions and blocks in the code.

Scope: space or environment in which certain variable is declared.

1. Global scope.
2. Function scope
3. Block scope.

Scope of a variable: region of code where a certain variable can be accessed.

|  |  |  |
| --- | --- | --- |
| Global | Function | Block(ES6) |
| Outside of any function or block |  |  |
| Variable declared here can be accessed everywhere | Variable accessible only inside the function | Variable accessible only inside the block. |
| Called global scope | Called Local scope | Applies only to Let, const variables as these are ES6 variable types. |
|  |  | Functions are also blocked scope.(only in strict mode) |
|  |  |  |

If one scope needs to use a certain variable but cant find it in current scope, it will look up in the scope chain and see if it can find it in parent scope. – This is called variable look up.

Hoisting

Makes some types of variables accessible/useable in code before they are declared.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hoisted | Initial Value in variable environment | Scope |
| Function declaration | yes | Actual function | block |
| Var variables | yes | undefined | Function |
| Let , const | NO | TDZ(temporal dead zone), uninitialized | Block |
| Function expression and arrow |  | Depends if using var or let /const.  (var = hoisted to undefined)(let/const = unusable , not hoisted) |  |

Temporal Dead Zone, let , const

the block where a variable is inaccessible until the moment the computer initializes it with a value

Why TDZ ?

Makes it easier to avoid and catch errors: accessing variables is bad practise and should be avoided.

Const – coz of TDZ const work the way there are suppose to work. Const can never be reassigned,

We cannot declare it as undefined and then assign a value to it.

Why Hoisting ?

Use function before declaration.

This Keyword ?

Special variable that is created for every execution context.

Will take the value of “Owner” of function .i.e. points to the owner of the function.

This is not static, it keeps changing and value is assigned when function is actually called.

This keyword will always point whoever is calling the function/method/value.

4 ways of calling a function-

1. Method - when we call a method, the This keyword inside the method will point to the object on which the method is called.

i.e. points to the object which is calling the method.

1. Simple function call – this is undefined only in strict mode.
2. Arrow functions – Do not get THIS keyword. It will get THIS keyword of surrounding function/parent function.(can be window function)
3. Event listener – THIS will point to the DOM element of handler function.

This will not point to the function itself and also not the variable environment.

Method Borrowing??

also known as function borrowing, is, as its name implies, a way for an object to use the methods of another object without redefining that same method.

Primitive vs Reference types objects

|  |  |
| --- | --- |
| Primitive | Reference |
| Stored in call stack in Js engine | Stored in Heap memory in JS engine |
| Identifier points to address and not value | Points to address in call stack which reference to address in heap. |
| Const in primitive is immutable | Const in primitive is mutable |

Object.assign()?

Merge two object and return a new one.

It only creates a shallow copy, i.e. object inside the object will still have the same value, not returning a new one and will still point to same place in memory.

Destructing arrays

It is a way of unpacking values from array/object into separate variables.

Break complex data structure into simple data structure.

For array we use destructing to retrieve elements from the array and store them into variables in a very easy way.

Original array is intact, it is not destroyed.

Spread operator

Expand an array to all its elements.

Use case of spread operator:

* Pass new value to functions.
* To build new array.

When we need the elements of array individually, we use spread operator.

It doesn’t create new variables.

Iterables

Arrays, Strings , Maps, Sets but NOT OBJECTS!!

Rest Patterns

Collect multiple elements and condense them into array.

Short circuiting(&& and ||)

OR :

* if 1st value is truthy, it will return truthy, it will not look at 2nd value.

AND:

* if 1st value is falsy, it will return falsy, it will not look at 2nd value.

Nullish Operator(??)

Works with the nulllish value instead of falsy values.

0 and “ “ are truthy values not falsy for Nullish.

Logical assignment operator(

OR assignment – ( ||= )

Nullish (??=)

AND (&&=)

Looping arrays(The for of loop)

.entries() – tells us the index and array element itself

Optional Chaining(?.)

If property does not exist undefined is returned immediately.

Object.keys/values/entries

Entries - return the index number and element itself.

SETS

Data structures where there is collection of unique values, it cannot have any duplicates.

Can hold mixed data types and order of element does not matter.

Can be iterable.(we can loop over them)

Set has no index hence we cannot get any value out of set.

NOTE – Basically used to remove duplicate values from arrays.

* SetName.size
* SetName.has(“something”) – if it has “something in set or not it will return true or false. (this is a method)
* SetName.add(“Shilpa”)
* SetName.delete(“Shillpa”)
* SetName.clear() – clear the whole set.

MAP

DS that we use to map values to keys.

Keys can have any type.

MAP methods –

* .has
* .delete
* .size – how many items in map
* …mapName.entries
* …mapName..keys
* …mapName..values

Strings

Strings are immutable

String methods(ALL case sensitive)

* indexOf – position in which letter is. (case sensitive)
* lastIndexOf -
* slice -position where extraction start (if 4, then it will start from 4th position, 4th position will be counted) spaces are counted

name.slice(3,7) – 3rd position is counted and 7th is not counted

Length of string will be end – beginning .i.e. 7 -3 = 4

* toLowerCase()
* toUpperCase()
* trim() – remove space
* .replace(‘ what u want to replace‘,’ what u want to replace with‘)
* .replaceAll
* .includes(“what you are searching for”) – gives true or false whether it has the word/letter in string or not.
* .startsWith(“what you are searching for”) - gives true or false whether it has the word/letter starts with that or not
* .endsWith(“what you are searching for”)
* .split(“Whatever you want to split it with”) – it will split Whatever you want to split it with and store it in array.
* .join(“Whatever you want to join it with”) - – it will join Whatever you want to join it with and if it was an array, it will convert it into normal String.
* .padStart(number you want the padding for, “whatever you want to pad with)
* .repeat(number of times you want to repeat it)

Q : Why do strings have methods when it is primitive type ?

A: Coz of BOXING

BOXING

Takes string and put it in a BOX and that box is object and does operation on it and when operation is done , it is converted back into string .i.e. Primitive Type.

FUNCTIONS

Default Parameters

Parameters we do not have to pass manually in case we do not want to change the default.

Passing argument as VALUE vs REFERENECE

FIRST CLASS FUNCTIONS

Functions are treated as values.

Pass function as values in other functions

Return function from other functions

Call methods on functions

Its just a feature that programming language has.

Higher Order functions

Function that receives other function as arguments and return a new function or both.

Ex – addEventListener, forEach

CallBACK FUNCTION

A callback is a function passed as an argument to another function

This technique allows a function to call another function

call() and apply()

call points to this keyword and has name of function as first argument and then string as second argument and so on.

Apply points to this keyword and has name of function as first argument and then array as second argument and so on.

Bind()

IIFE

Immediately invoked function expression

An **IIFE** (Immediately Invoked Function Expression) is a [JavaScript](https://developer.mozilla.org/en-US/docs/Glossary/JavaScript) [function](https://developer.mozilla.org/en-US/docs/Glossary/Function) that runs as soon as it is defined, will not run again and is not saved anywhere in the code.

USES :

Data privacy

When we need to execute a function just once.

CLOSURES

a closure is a function that has access to its parent function's scope, even after the parent function has finished executing.

It has priority over scope chain.

Closure is like a backpack that a function carries around where ever it goes. This backpack has all the variables that were present in the environment where the function was created.

Simple array methods

* .slice(start slicing from(included),not included) – slice array and returns new array.
* .splice(start slicing from(included),number of element you want to delete) - slice original array.
* .reverse()- mutate original array.
* .concat() – Does not mutate original array
* .join()- result is a string.
* .at() – tells you the index of the array element, also works on strings

forEach()

Requires a callback function.

It will iterate over the array and with each iteration it will execute callback function.

It passes current element, current index and entire array in callback function.

You cannot breakout in forEach loop.

Continue and break statement does not work in this.

forEach() on Maps

Requires a callback function.

It will iterate over the array and with each iteration it will execute callback function.

It passes current value, current key and entire map in callback function.

forEach() on Sets

Requires a callback function.

It will iterate over the array and with each iteration it will execute callback function.

In sets value and key are same, i.e. they have not key.

insertAdjacentHTML

it is a method through which we can attached HTML code to webpage.

It takes 2 argument :

1.

2. the string containing html we want to insert.

Data transformation: MAPS, FILTERS, REDUCE

|  |  |  |
| --- | --- | --- |
| MAP | FILTER | REDUCE |
| To loop over array, similar as forEach | Use to filter for elements in the original array which satisfy certain conditions | Use to boil down all the elements of original array into one single element. |
| Creates new array based on original array.  In each iteration, it applies an call back function that we specify in code to each element in array. | Example – look for element in array which is greater than 2. | Example – Add all the elements of array. |
| Example - in array multiply each element by 2 | Returns new array. | No new array, only reduced value. |
| It maps the value of original array to new array. | Also use to loop over array. |  |
| It has access to element, index and whole array |  |  |

Map method

Filter Method

Filter for elements that satisfy a certain condition.

Reduce method

Boil down all the method of array into one single value

First parameter is accumulate– this accumulate all the value that we want to return i.e. value we keep adding to.

We also have to set the starting value from which it will start accumulating/adding.

Find method

Loops over an array to retrieve one element of array based on condition.

Returns the first element of array that satisfies the condition.

|  |  |
| --- | --- |
| Filter | Find |
| Returns all elements that matches the condition | Returns the first element of array that satisfies the condition. |
| Returns new array | Only returns element , NOT the array |
|  |  |
|  |  |

FindIndex()

Return the index of found element and not the element itself.

Some

Test if condition is true or not

Every

Only return true if all elements satisfy the condition

Flat

Sub array/nested array can be put in one array.

No call back function.

Deep Sub array/ deep nested array can be put in one array.

Need to give parameter during calling

Flat MAP

Combines map and flat method together

But this is only one level deep array , for deeper array u need flat method.

Sort array

arr.sort() – mutates original array.

This sort based on strings

So for numbers we use compare see ex In VS code.

Parameters are 0,a, b – these are consecutive numbers, doesn’t matter which ones (a could be bigger than b or vice versa)

Fill array

Const x = new Array(7) -> this will create empty array with 7 length

* fill – mutate the original array

arr.fill(number u want to fill , start index, end index[does not include end index])

arr.fill(23,4,6)

* Array.from – use to create array from array like structure like string, maps or sets .

Which array method to use when?

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| To mutate array | | A new array | |  | Want An array index |  | Retrieve entire array element |  | Want to know if array include certain elements |  | Want new string |  | Transfer array into new value |  | Loop over array |
| Add to original array | .push(end) | Compute from original | .map(loop) | Based on value | .indexOf |  | | Based on value | .includes | Based on separator string | .join | Based on accumulator | .reduce | Based on callback | .forEach |
| .unshift(start) | Filter using condition | .filter | Based on test condition | .findIndex | Based on test consition | .find | Based on test condition | .some |  |  |  |  |  |  |
| Remove from original | .pop(end) | Portion of original | .slice |  |  |  |  | .every |  |  |  |  |  |  |
| .shift(start) | Adding original to other | .concat |  |  |  |  |  |  |  |  |  |  |  |  |
| .splice(any) | Flattening the original | .flat |  |  |  |  |  |  |  |  |  |  |  |  |
| Others | .reverse | .flatMap |  |  |  |  |  |  |  |  |  |  |  |  |
| .sort |  |  |  |  |  |  |  |  |  |  |  |  |  |
| .fill |  |  |  |  |  |  |  |  |  |  |  |  |  |

Converting and checking numbers

All numbers are represented internally as floating point numbers.

* ParseInt = get rid of unnecessary symbols/strings, should strt with number only

Accepts second argument REDUX (base of numeral system that we are using)

* ParseFloat
* isNaN – check if value is number or not.
* isFinite – check if value is a real number
* isInteger

Math and Round

* Math.sqrt
* Math.max – max value is return and does type coercion
* Math.min
* Math.PI
* Math.random – gives value between 0 and 1
* Math.trunc – removes decimal part
* Math.round – round to the nearest integer
* Math.ceil – round up to the integer
* Math.floor - round down to the integer
* toFixed – returns strings, round decimal to whatever number is given after toFixed

Remainder operator(%)

Remainder of division

Number is even if remainder is 0

Numeric separator

Numeric separators are undeScores that we can place anywhere in our number so that we can esily parse large number.

2300\_000

Create Dates