Javascript my notes

Javascript is case sensitive and loosely typed language variable var keyword,initialize before access.

Object literal

Var p= {

Fname:”h”

Age:25

Getfullname:function() {}

}

Ternary operator: ?: assign value on condition

Primitive datatype: type of used string, number,boolean, undefined

Structural: object,date, array

Array methods

Concat: combine array with array values

Every: return true or false (if one condition unsatisfy)

Filter: return new array that satisfy condition specified in callback func

Foreach: execute callback for each element of array

Indexof: returns index of first occurance

Join: return string of array elements

Lastindexof: last occurance

Map: create new array ( using filter and other array method)

* Reduce: pass 2 elements simultaneously in callback and return single value
* Reduce.right() pass 2 elements simultaneously from right to left

Reverse:reverse

Shift: first element remove and returns

Slice: returns new array with specified start to end

Some:return true if one element satisfy condition

Splice: add/remove element from array

Splice(2,0,’’)->add

Splice(2,2) -> remove

Tostring: string represent

Unshift: add 1 or more element at front

Null and undefined are primitive values in javascript

Null:absence

Undefined:lack of value

Settimeout:call func after delay finish and after queued event handlers

Setimmediate: it is faster,checks queue of event handlers. It queues them immediately after the last i/o handler

Higher order func: func which takes another func as argument.

….done

Es6 featusres

https://www.w3schools.com/js/js\_es6.asp

* [The let keyword](https://www.w3schools.com/js/js_es6.asp#mark_let)
* [The const keyword](https://www.w3schools.com/js/js_es6.asp#mark_const)
* [Arrow Functions](https://www.w3schools.com/js/js_es6.asp#mark_arrow)
* [The ... Operator](https://www.w3schools.com/js/js_es6.asp#mark_spread)
* [For/of](https://www.w3schools.com/js/js_es6.asp#mark_forof)
* [Map Objects](https://www.w3schools.com/js/js_es6.asp#mark_map)
* [Set Objects](https://www.w3schools.com/js/js_es6.asp#mark_set)
* [Classes](https://www.w3schools.com/js/js_es6.asp#mark_class)
* [Promises](https://www.w3schools.com/js/js_es6.asp#mark_promise)
* [Symbol](https://www.w3schools.com/js/js_es6.asp#mark_symbol)
* [Default Parameters](https://www.w3schools.com/js/js_es6.asp#mark_param)
* [Function Rest Parameter](https://www.w3schools.com/js/js_es6.asp#mark_rest)
* [String.includes()](https://www.w3schools.com/js/js_es6.asp#mark_includes)
* [String.startsWith()](https://www.w3schools.com/js/js_es6.asp#mark_startswith)
* [String.endsWith()](https://www.w3schools.com/js/js_es6.asp#mark_endswith)
* [Array.from()](https://www.w3schools.com/js/js_es6.asp#mark_array_from)
* [Array keys()](https://www.w3schools.com/js/js_es6.asp#mark_array_keys)
* [Array find()](https://www.w3schools.com/js/js_es6.asp#mark_array_find)
* [Array findIndex()](https://www.w3schools.com/js/js_es6.asp#mark_array_findIndex)
* [New Math Methods](https://www.w3schools.com/js/js_es6.asp#mark_math_methods)
* [New Number Properties](https://www.w3schools.com/js/js_es6.asp#mark_number_properties)
* [New Number Methods](https://www.w3schools.com/js/js_es6.asp#mark_number_methods)
* [New Global Methods](https://www.w3schools.com/js/js_es6.asp#mark_global_methods)
* [Object entries](https://www.w3schools.com/js/js_es6.asp#mark_entries)
* [JavaScript Modules](https://www.w3schools.com/js/js_es6.asp#mark_modules)

he let keyword allows you to declare a variable with block scope.

var x = 10;  
// Here x is 10  
{  
  let x = 2;  
  // Here x is 2  
}  
// Here x is 10

The const keyword allows you to declare a constant (a JavaScript variable with a constant value).

Constants are similar to let variables, except that the value cannot be changed.

var x = 10;  
// Here x is 10  
{  
  const x = 2;  
  // Here x is 2  
}  
// Here x is 10

Arrow Functions

Arrow functions allows a short syntax for writing function expressions.

You don't need the function keyword, the return keyword, and the **curly brackets**.

// ES5  
var x = function(x, y) {  
   return x \* y;  
}  
  
// ES6

const x = (x, y) => x \* y;

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_es6_arrow)

Arrow functions do not have their own this. They are not well suited for defining **object methods**.

Arrow functions are not hoisted. They must be defined **before** they are used.

Using const is safer than using var, because a function expression is always a constant value.

You can only omit the return keyword and the curly brackets if the function is a single statement. Because of this, it might be a good habit to always keep them:

### Example

const x = (x, y) => { return x \* y };

## The Spread (...) Operator

The ... operator expands an iterable (like an array) into more elements:

### Example

const q1 = ["Jan", "Feb", "Mar"];  
const q2 = ["Apr", "May", "Jun"];  
const q3 = ["Jul", "Aug", "Sep"];  
const q4 = ["Oct", "Nov", "May"];  
  
const year = [...q1, ...q2, ...q3, ...q4];

The ... operator can be used to expand an iterable into more arguments for function calls:

### Example

const numbers = [23,55,21,87,56];  
let maxValue = Math.max(...numbers);

## The For/Of Loop

The JavaScript for/of statement loops through the values of an iterable objects.

for/of lets you loop over data structures that are iterable such as Arrays, Strings, Maps, NodeLists, and more.

The for/of loop has the following syntax:

for (*variable* of *iterable*) {  
  // *code block to be executed*  
}

variable - For every iteration the value of the next property is assigned to the variable. Variable can be declared with const, let, or var.

iterable - An object that has iterable properties.

### Looping over an Array

### Example

const cars = ["BMW", "Volvo", "Mini"];  
let text = "";  
  
for (let x of cars) {  
  text += x + " ";  
}

### Looping over a String

### Example

let language = "JavaScript";  
let text = "";  
  
for (let x of language) {  
    text += x + " ";  
}

## JavaScript Maps

Being able to use an Object as a key is an important Map feature.

### Example

const fruits = new Map([  
["apples", 500],  
["bananas", 300],  
["oranges", 200]  
]);

## JavaScript Sets

### Example

// Create a Set  
const letters = new Set();  
  
// Add some values to the Set  
letters.add("a");  
letters.add("b");  
letters.add("c");

## JavaScript Classes

JavaScript Classes are templates for JavaScript Objects.

Use the keyword class to create a class.

Always add a method named constructor():

### Syntax

class ClassName {  
  constructor() { ... }  
}

class Car {  
  constructor(name, year) {  
    this.name = name;  
    this.year = year;  
  }  
}

The example above creates a class named "Car".

The class has two initial properties: "name" and "year".

A JavaScript class is **not** an object.

It is a **template** for JavaScript objects.

## Using a Class

When you have a class, you can use the class to create objects:

### Example

const myCar1 = new Car("Ford", 2014);  
const myCar2 = new Car("Audi", 2019);

## JavaScript Promises

A Promise is a JavaScript object that links "Producing Code" and "Consuming Code".

"Producing Code" can take some time and "Consuming Code" must wait for the result.

### Promise Syntax

const myPromise = new Promise(function(myResolve, myReject) {  
// "Producing Code" (May take some time)  
  
  myResolve(); // when successful  
  myReject();  // when error  
});  
  
// "Consuming Code" (Must wait for a fulfilled Promise).  
myPromise.then(  
  function(value) { /\* code if successful \*/ },  
  function(error) { /\* code if some error \*/ }  
);

### Example Using a Promise

const myPromise = new Promise(function(myResolve, myReject) {  
  setTimeout(function() { myResolve("I love You !!"); }, 3000);  
});  
  
myPromise.then(function(value) {  
  document.getElementById("demo").innerHTML = value;  
});

## The Symbol Type

A JavaScript Symbol is a primitive data type just like Number, String, or Boolean.

It represents a unique "hidden" identifier that no other code can accidentally access.

For instance, if different coders want to add a person.id property to a person object belonging to a third-party code, they could mix each others values.

Using Symbol() to create a unique identifiers, solves this problem:

### Example

const person = {  
  firstName: "John",  
  lastName: "Doe",  
  age: 50,  
  eyeColor: "blue"  
};  
  
let id = Symbol('id');  
person[id] = 140353;  
// Now person[id] = 140353  
// but person.id is still undefined

Symbols are always unique.

If you create two symbols with the same description they will have different values:

Symbol("id") == Symbol("id"); // false

## Default Parameter Values

ES6 allows function parameters to have default values.

### Example

function myFunction(x, y = 10) {  
  // y is 10 if not passed or undefined  
  return x + y;  
}  
myFunction(5); // will return 15

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_es6_default)

## Function Rest Parameter

The rest parameter (...) allows a function to treat an indefinite number of arguments as an array:

### Example

function sum(...args) {  
  let sum = 0;  
  for (let arg of args) sum += arg;  
  return sum;  
}  
  
let x = sum(4, 9, 16, 25, 29, 100, 66, 77);

## String.includes()

The includes() method returns true if a string contains a specified value, otherwise false:

### Example

let text = "Hello world, welcome to the universe.";  
text.includes("world")    // Returns true

## String.startsWith()

The startsWith() method returns true if a string begins with a specified value, otherwise false:

### Example

let text = "Hello world, welcome to the universe.";  
  
text.startsWith("Hello")   // Returns true

## String.endsWith()

The endsWith() method returns true if a string ends with a specified value, otherwise false:

### Example

var text = "John Doe";  
text.endsWith("Doe")    // Returns true

## Array.from()

The Array.from() method returns an Array object from any object with a length property or any iterable object.

### Example

Create an Array from a String:

Array.from("ABCDEFG")   // Returns [A,B,C,D,E,F,G]

## Array keys()

The keys() method returns an Array Iterator object with the keys of an array.

### Example

Create an Array Iterator object, containing the keys of the array:

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
const keys = fruits.keys();  
  
let text = "";  
for (let x of keys) {  
  text += x + "<br>";  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_keys)

## Array find()

The find() method returns the value of the first array element that passes a test function.

This example finds (returns the value of ) the first element that is larger than 18:

### Example

const numbers = [4, 9, 16, 25, 29];  
let first = numbers.find(myFunction);  
  
function myFunction(value, index, array) {  
  return value > 18;  
}

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_find)

Note that the function takes 3 arguments:

* The item value
* The item index
* The array itself

## Array findIndex()

The findIndex() method returns the index of the first array element that passes a test function.

This example finds the index of the first element that is larger than 18:

### Example

const numbers = [4, 9, 16, 25, 29];  
let first = numbers.findIndex(myFunction);  
  
function myFunction(value, index, array) {  
  return value > 18;  
}

Note that the function takes 3 arguments:

* The item value
* The item index
* The array itself

## New Math Methods

ES6 added the following methods to the Math object:

* Math.trunc()
* Math.sign()
* Math.cbrt()
* Math.log2()
* Math.log10()

## The Math.trunc() Method

Math.trunc(x) returns the integer part of x:

### Example

Math.trunc(4.9);    // returns 4  
Math.trunc(4.7);    // returns 4  
Math.trunc(4.4);    // returns 4  
Math.trunc(4.2);    // returns 4  
Math.trunc(-4.2);    // returns -4

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_math_trunc)

## New Global Methods

ES6 added 2 new global number methods:

* isFinite()
* isNaN()

## The isFinite() Method

The global isFinite() method returns false if the argument is Infinity or NaN.

Otherwise it returns true:

### Example

isFinite(10/0);       // returns false  
isFinite(10/1);       // returns true

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_es6_isfinite)

## The isNaN() Method

The global isNaN() method returns true if the argument is NaN. Otherwise it returns false:

### Example

isNaN("Hello");       // returns true

## Object entries()

### Example

Create an Array Iterator, and then iterate over the key/value pairs:

const fruits = ["Banana", "Orange", "Apple", "Mango"];  
const f = fruits.entries();  
  
for (let x of f) {  
  document.getElementById("demo").innerHTML += x;  
}

The entries() method returns an Array Iterator object with key/value pairs:

[0, "Banana"]  
[1, "Orange"]  
[2, "Apple"]  
[3, "Mango"]

The entries() method does not change the original array.

## Modules

Modules are imported in two different ways:

### Import from named exports

Import named exports from the file person.js:

import { name, age } from "./person.js";

## JavaScript Array with() Method

ES2023 added the Array with() method as a safe way to update elements in an array without altering the original array.

### Example

const months = ["Januar", "Februar", "Mar", "April"];  
const new = months.with(2, "March");

[Try it Yourself »](https://www.w3schools.com/js/tryit.asp?filename=tryjs_array_with)