1. OBJECT: Literal, Function Declaration, Constructor Function
2. OBJECT.CREATE() : Function Declaration -> let mahasiswa = Object.create(methodMahasiswa);
3. PROTOTYPE & CLASS -> Constructor Function ->function Mahasiswa(..).. -> Mahasiswa.prototype.main = function(..){}

CLASS: class Mahasiswa {constructor (nama,energi){this.nama; this.energi} makan(porsi){..}}

1. EXECUTING CONTEXT, HOISTING & SCOPE

CONTEXT, HOISTING ->variable di naikkan (hoisting) dengan nilai undefined, function di naikkan dengan nama fungsi dahulu…..

SCOPE -> cari dulu var terdekat, jika tidak ada cari di argument, jika tidak ada cari di var global…;

1. CLOSURE ->inner function yang mempunyai akese ke variable fungsi parentnya…inner f 🡪closure;

MENJALANKAN CLOSURE: 1. Fungsi closure ditampung di variable 2. Fungsi parent di kasi kurung +();

1. VAR. LET & CONST -> var (function scope), let & const (block scope) -> jangan pernah gunakan var lagi
2. ARROW FUNCTION -> Bentuk lain dari Function Expression yg lebih ringkas

Contoh: let tampilPesan=function (nama) {..} menjadi let tampilPesan=(nama)=>{..}

Implisit Return : const tampilNama = nama => `Hallo ${nama}`; const tampilNama = () =>’hello word’;

Arrow function ouputnya object???

1. THIS PADA ARROW FUNCTION

CONTRUCTION FUNCTION TIDAK BISA DI UBAH MENJADI ARROW FUNCTION

METHOD PADA CONSTRUCTOR FUNCTION BISA DI UBAH MENJADI ARROW FUNCTION

ARROW FUNCTION TIDAK MEMILIKI KONSEP THIS…

1. HIGHER ORDER FUNCTION: 1. Fungsi sebagai parameter 2. Fungsi sebagai return value (closure)
2. FILTER, MAP & REDUCE . METHOD CHAINING
3. LATIHAN FILTER, MAP & REDUCE
4. TEMPLATE LITERAL / TEMPLATE STRING :1. menggunakan tanda back tic 2. Multiline string 3. Embedded Expression 4. HTML Fragment 5. Expression Interpolation 6. Tagged Template
5. TEMPLATE LITERAL LATIHAN
6. TAGGED TEMPLATE LITERAL : 1. Escaping HTML Tags 2. Translation & Internationalization 3. Style comp.
7. DESTRUCTURING ASSIGNMENT

ARRAY => const coba = {‘satu’,’dua’,’tiga’]; const [a,b,c] = coba;

OBJECT => const mhs = {nama: ‘Sandika’, umur: 33}; const {umur,nama}=mhs;

ASSIGNMENT TANPA DEKLARASI OBJEK =>({nama,umur}={nama:’Rahmadi’, umur:46});

ASSIGNMENT KE VARIABEL BARU => ({nama:n,umur:u}={nama:’Rahmadi’, umur:46});

MEMBERIKAN DEFAULT VALUE => ({nama,umur,email = ‘ no email’}={nama:’Rahmadi’, umur:46});

SWAP ITEM => let a =10; let b = 15; [a,b]=[b,a];

REST PARAMETER => const [a,b,c]=[1,2,3,4,5] atau const [a, …values]=[1,2,3,4,5];

1. DESTRUCTURING FUNCTION

ARRAY => [jumlah, kali]=function jumlahkali (a,b){ return [a+b, a\*b]}

OBJECT=>{}=function kalkulasi (kali,jumlah){return {jumlah: a+b, kali: a\*b}}

ARGUMENTS =>

1. LOOPING FOR YANG BARU

FOR .. OF => const mhs=[‘sandika’,’dodi’,’erik’]; for ( m of mhs){console.log(m)}; 🡺mhs.entries()???

(utk iterable object 1. String 2. Array 3. Arguments/NodeList 4. TyPed Array 5. Map 6. Set 7. User define i )

FOR .. IN => const mhs={nama:‘sandika’, umur: 31}; for ( m inf mhs){console.log(m)};

1. SPREAD OPERATOR : memecah iterables menjadi single elements

GABUNGKAN DUA ARRAY =>const dosen=[‘rahmadi’,’eko’]; const mhs=[‘tari’,’diah’]; const gabung=[…dosen,…mhs];

MENGCOPY ARRAY => const dosen=[‘rahmadi’,’eko’; const dosen1=[…dosen];

MENGUBAH NODELIST MENJADI ARRAY => limhs -> […limhs]

MEMECAH STRING => const nama=’RAHMADI’; const huruf = […nama.textContent];

1. REST PARAMETER => SPEREAD pada argument fungsi…