NASHTA
GLOBAL UTAMA

Introduction
JavaScript

JavaScript

Bahasa pemrograman yang digunakan dalam pengembangan website agar lebih dinamis dan interaktif. JavaScript dapat meningkatkan fungsionalitas pada halaman web. Bahkan dengan JavaScript ini kamu bisa membuat aplikasi, tools, atau bahkan game pada web.

Bicara teknis, JavaScript atau kita singkat menjadi JS merupakan bahasa pemrograman jenis interpreter, sehingga kamu tidak memerlukan compiler untuk menjalankannya. JavaScript memiliki fitur-fitur seperti berorientasi objek, client-side, high-level programming.

Variable

JavaScript memiliki beberapa cara deklarasi variabel:

Format: keyword namaVariabel

	Redeclare	Hoisting	Block Scope	Create global props
var	√	√	X	√
let	X	X	√	X
const	X	X	√	X

keyword	var
global scope	YES
function scop	YES
block scope	NO
can be reassigned	YES

Data Types

1. String
2. Number
3. Boolean
4. Object
5. null
6. Array
7. Undefined

syntax: typeof operan atau typeof(operand)

Condition and Looping

Condition

- if-else
- switch-case
- Ternary Operators / Short-Circuit Logic

Looping

- for loop
- while

Comparison Operators

Operator logika di JavaScript, yaitu:

- Equal value, ==
- Equal value and type, ===
- Not equal, !=
- Not equal value and type, !==
- Greater than, >
- Less than, <
- Greater than or equal, >=
- Less than or equal, <=

Template Literals

Template Literals adalah literal string yang memungkinkan untuk penempelan ekspresi.

Kegunaan:

- Multi-line
- Expression
- and other

Sintaks:

`string text ${expression} string text`

concatenated strings (ES5)

```
const a = 10;
const b = 15;

console.log('value a = '+ a +'\n'+
'value b = '+ b + '\n'+
'the sum of a with b is '+ (a+b))
/*
output :
value a = 10
value b = 15
the sum of a with b is 25
*/
```

array_2 = [80, 70, 60, 50, 40, 30]

Template Literal (ES6)

```
const a = 10;
const b = 15;

console.log(`nilai a = ${a}
nilai b = ${b}
the sum of a with b is ${a + b}`)

/*
output :
value a = 10
value b = 15
the sum of a with b is 25
*/
```

Spread Operator

Penggunaan spread operator memakai simbol tiga dot atau titik (...).

- Memasukkan array ke dalam array lain
- Menggabungkan 2 array
- Mengcopy/clone objek
- Menggabungkan objek

```
// memasukkan array ke array lain
const hobby = ['swimming', 'gaming'];
const newHobby = [...hobby, 'karaoke', 'Hiking'];

// newHobby = [ 'swimming', 'gaming', 'karaoke', 'Hiking' ]
```

Destructuring

Ekspresi javascript yang memungkinkan untuk membagi atau memecah nilai dari sebuah array atau objek ke dalam variabel yang berbeda

- Destructuring Object

```
const student = {
    firstName: 'Grad',
    lastName: 'Chinda',
    country: 'Nigeria'
}
// before using destructuring
const firstName = student.firstName
const lastName = student.lastName
const country = student.country
```

```
const student = {
    firstName: 'Grad',
    lastName: 'Chinda',
    country: 'Nigeria'
}
// destructuring assignment
const {firstName, lastName, country} = student
```

- Destructuring Array

// Sebelum menggunakan Destructuring yaitu dengan mengambil nilai di dalam array berdasarkan index nya

```
const rgb = [255, 140, 0];
// retrieve data based on index
const red = rgb[0];
const green = rgb[1];
const blue = rgb[2];
```

// Setelah menggunakan destructuring

```
const rgb = [255, 140, 0];

const [red,green,blue] = rgb;
```

Function

Function adalah blok kode untuk melakukan tugas tertentu. Dimulai dengan kata kunci function dan dicakup oleh tanda kurung. Fungsi akan berhenti jika berakhir atau ada sintaks return

Sintaks:

ES5
Declaration
```
function doStuff() {};
```

Expression
```
const doStuff = function() {}
```

ES6
Arrow Function
```
const doStuff = () => {}
```

NASHTA
GLOBAL UTAMA

Contoh :

```
const addition = (num1, num2) => {
  const total = num1 + num2

// untuk mengakhiri atau mengembalikan nilai pada fungsi bisa menggukana return
  return total;
}

console.log(addition(10,15))
// output: 25
```

Method (built-in)

Method adalah fungsi di dalam objek. Ketika kita membuat objek, kita dapat membuat fungsi di dalamnya. Untuk mengakses bagian lain dari objek menggunakan kata kunci this.

Berikut adalah beberapa method untuk array di JavaScript.

- sort()
- map()
- filter()
- push()

Callback Function

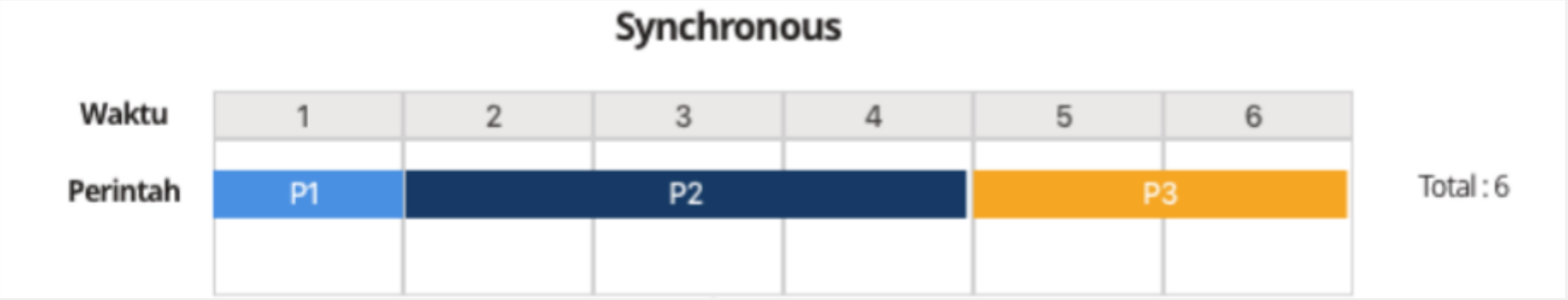
Callback Function adalah function biasa, yang dikirimkan sebagai parameter ke function lain, kemudian dieksekusi di function tersebut.

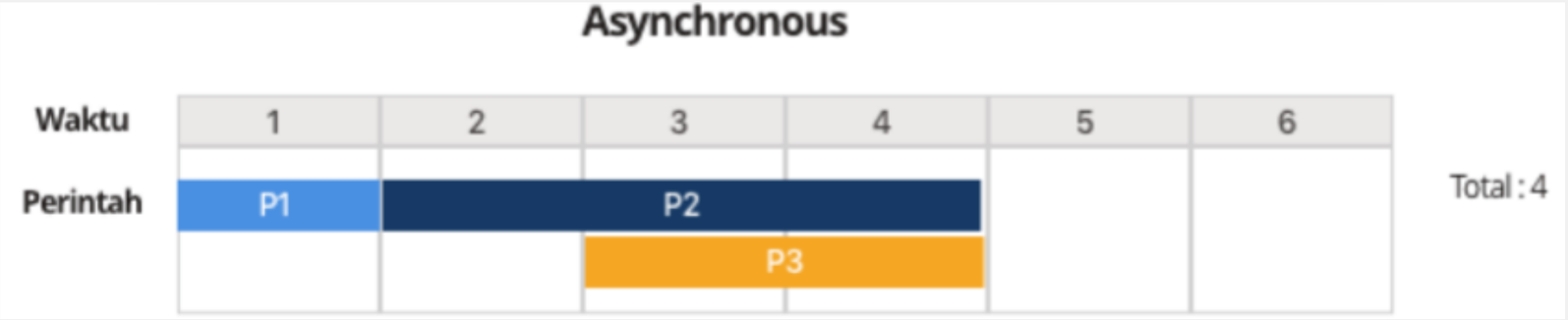
```
function sayHello(name, callback){
  let greeting = `Hello ${name}`
  callback(greeting)
}

function showGreeting(quote){
  console.log(quote)
}

sayHello('John', showGreeting)
```

Asynchronous

Asynchronous VS Synchronous

- Synchronus = perintah dieksekusi satu persatu sesuai urutan kode yang anda tuliskan
- Asynchronous = hasil eksekusi atau output tidak selalu berdasarkan urutan kode, tetapi berdasarkan waktu proses





Callback Asynchronous

Callback function atau callback (biasa disingkat dengan cb) adalah salah satu metode yang paling umum yang digunakan untuk menghandle return value dari operasi asynchoronous.

Callback sendiri adalah sebuah regular function (yang biasanya anonymous) dan ditaruh di argumen paling belakang dari sebuah asynchronous function. Layaknya function biasa, callback juga dapat menerima parameter dan mengembalikan value.

```
const getmonth = (callback) => {
    setTimeout (() => {
        let error = true;
        let month = ['Januari', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'Desember']
        if(!error){
            callback(null, month)
        } else {
            callback(new Error('Sorry Data Not Found'), [])
        }
    }, 4000)
}

showMonth = (n, m) => {
    if(n === null){
        m.map(x => console.log(x))
    } else {
        console.log(n)
        if(m === undefined){
            console.log(`[]`)
        }
    }
}
getmonth(showMonth)
```

Promise

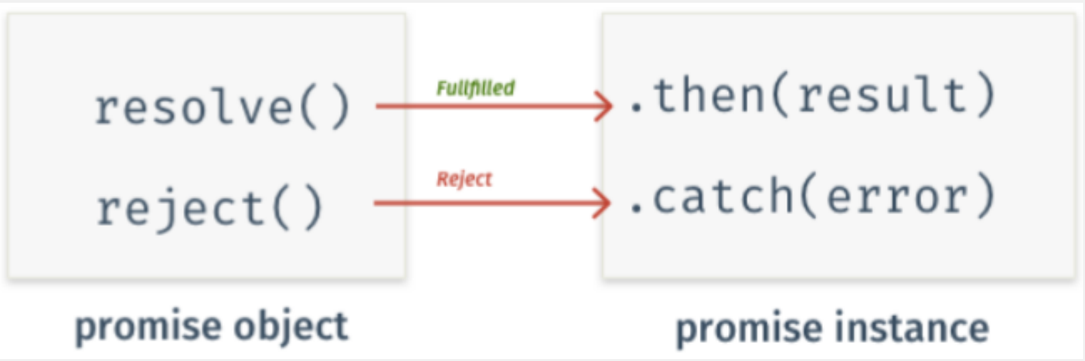
Promise merupakan perwakilan dari sebuah nilai yang belum tentu diketahui nilainya saat promise dibuat. Promise memungkinkan pengguna untuk menghubungkan fungsi handler dengan keberhasilan atau kegagalan aksi asynchronous.

```
let janjian = new Promise((resolve, reject)=>{
    let success = false
    if(success){
        resolve('berhasil')
    }else{
        reject(new Error('janji dibatalkan'))
    }
})
```

Promise (Then Catch)

Ketika menggunakan promise bisa menggunakan then dan catch

```
janjian
.then((result)=>{
    console.log(result)
})
.catch((error)=>{
    console.log(error)
})
```



Async/Await

Async/await adalah fitur yang hadir sejak ES2017. Fitur ini mempermudah kita dalam menangani proses asynchronous.

```
1   async function helloWorld(){
2       let result = await doAsync()
3       console.log(result)
4   }
```

Keterangan :
async → mengubah function menjadi synchronous
await → menunda eksekusi hingga proses asynchronous selesai,

Try Catch

Untuk mengatasi Error (error handling) pada async/await bisa menggunakan try catch

- Try

  biasanya kita sisipkan code javascript yang mungkin terjadi error
- Catch

  blok inilah yang akan menangkap error yang terjadi pada blok Try apabila pada blok Try si error muncul.