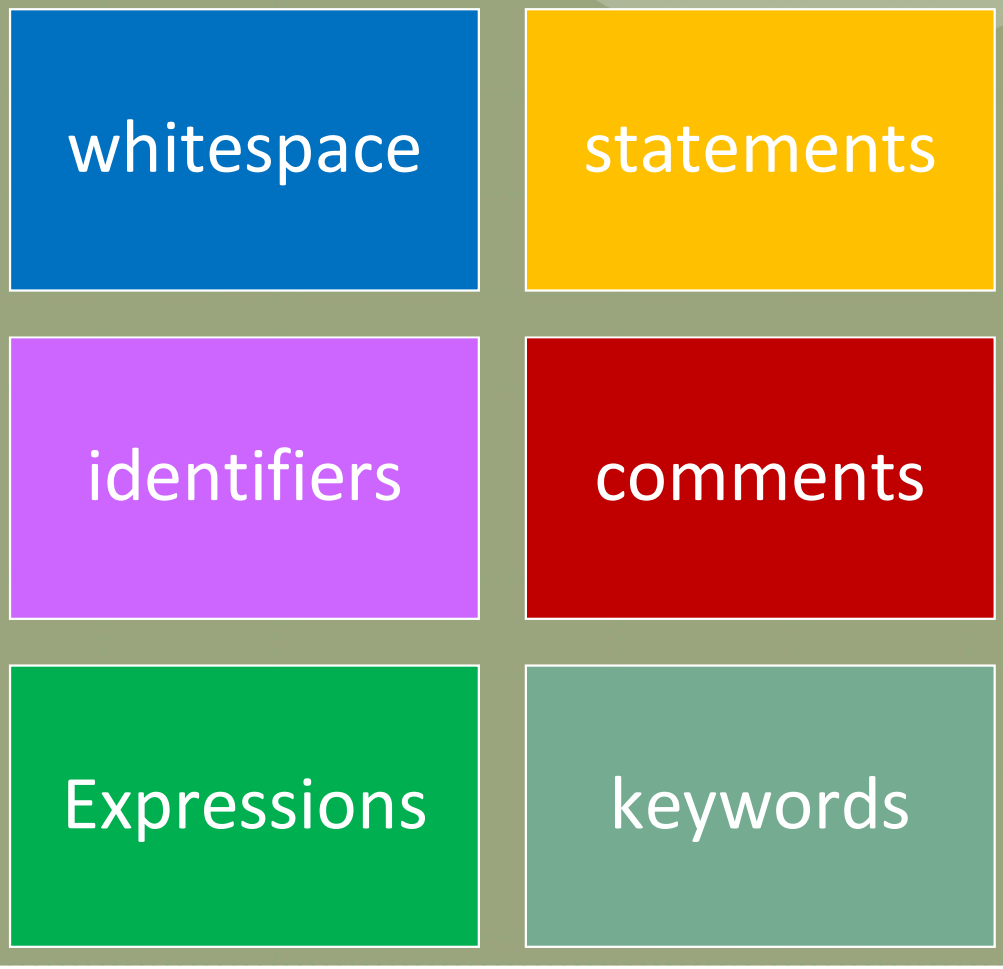
Javascript

Imelda Doharta Aritonang

Javascript Syntax

- Sintaks javascript mencakup



Whitespace

Tanpa Whitespace

```
let formatted = true; if (formatted) {console.log('The code is easy to read');}
```

Dengan Whitespace

```
let formatted = true;

if (formatted) {
  console.log('The code is easy to read');
}
```

Statements

- Kumpulan instruksi yang akan dikerjakan mesin javascript, walaupun tidak wajib tapi sebaiknya diakhiri dengan semicolon (;)

```
let message = "Welcome to JavaScript";
console.log(message);
```

Statements didalam kurung kurawal {} disebut block

```
if (window.localStorage) {
  console.log('The local storage is supported');
}
```

Identifier

- Adalah nama yang diberikan pada sebuah variable, fungsi, class, dll.
- Sebuah identifier diawali huruf (a-z atau A-Z),underscore(_), atau $, kemudian diikuti a-z, A-Z, underscore(_), angka, $. Contoh: namaAwal, Nama_Awal, _nama1,
- Camel Case (firstName), Pascal Case (FirstName), Snake Case (snake_case)
- Identifier case sensitive, **message** tidak sama dengan **Message**

Comments

```
// this is a single-line comment
```

```
/* This is a block comment
that can span multiple lines */
```

Ekspresi

```
2 + 1
```

Keywords

break	case	catch
continue	debugger	default
else	export	extends
function	if	import
new	return	super
throw	try	null
void	while	with
class	delete	finally
in	switch	typeof
yield	const	do
for	instanceof	this
var		

enum	implements	let
protected	private	public
await	interface	package
implements	public	

Data Type

- Javascript Bahasa yang bertipe dinamis, tergantung nilai yang diberikan.

```
let counter = 120; // counter is a number
counter = false;   // counter is now a boolean
counter = "foo";   // counter is now a string
```

```
let counter = 120;
console.log(typeof(counter)); // "number"

counter = false;
console.log(typeof(counter)); // "boolean"

counter = "Hi";
console.log(typeof(counter)); // "string"
```

Undefined Type

- Jika variable dideklarasikan tapi tidak diberi nilai

```
let counter;
console.log(counter);        // undefined
console.log(typeof counter); // undefined
```

- Jika memanggil sebuah variable yang belum dideklarasikan

```
console.log(typeof undeclaredVar); // undefined
```

Null Type

```
let obj = null;
console.log(typeof obj); // object
```

Javascript menetapkan bahwa null sama dengan undefined

```
console.log(null == undefined); // true
```

Number Type

- Hanya integer dan floating-point (bilangan desimal)

```
let num = 100;
```

```
let price= 12.5;
let discount = 0.05;
```

```
const budget = 1_000_000_000;
```

- Jika dibelakang koma adalah 0 maka dianggap integer untuk menghemat penggunaan memory, karena floating point menggunakan memory 2 kali lipat dari integer

```
let price = 200.00; // interpreted as an integer 200
```

- NaN singkatan dari Not a Number

```
console.log('a'/2); // NaN;
```

```
console.log(NaN/2); // NaN
console.log(NaN == NaN); // false
```

Number Type (Sambungan)

Methods	Description
isFinite()	It determines whether the given value is a finite number.
isInteger()	It determines whether the given value is an integer.
parseFloat()	It converts the given string into a floating point number.
parseInt()	It converts the given string into an integer number.
toExponential()	It returns the string that represents exponential notation of the given number.
toFixed()	It returns the string that represents a number with exact digits after a decimal point.
toPrecision()	It returns the string representing a number of specified precision.
toString()	It returns the given number in the form of string.

String Type

```
let status = false;
let str = status.toString(); // "false"
let back = Boolean(str); // true
```

```
substr(startIndex,[length]);

substring(startIndex,endIndex)
```

- Bisa karakter kosong atau beberapa karakter, diawali dan diakhiri single quote atau double quote

```
let greeting = 'Hi';
let message  = "Bye";
```

```
let message = 'I\'m also a valid string'; // use \ to escape the single quote
```

```
let name = 'John';
let message = `Hi, I'm ${name}.`;

console.log(message);
```

```
let str = 'JavaScript';
str = str + ' String';
```

```
let s = 'JavaScript';
s[0] = 'j';
console.log(s)
```

```
let className = 'btn';
className += ' btn-primary'
className += ' none';

console.log(className);
```

```
let str = "Good Morning!";
console.log(str.length);  // 13
```

```
let str = "Hello";
console.log(str[str.length -1]); // "o"
```

String Type (Sambungan)

```
let str = "This is a test of search()";
let pos = str.search(/is/);
console.log(pos); // 2
```

```
substr(startIndex,[length]);
```

```
substring(startIndex,endIndex)
```

```
let expression = '1 + 2 = 3';
let matches = expression.match(/\d+/);
console.log(matches[0]); // "1"
```

```
string.indexOf(substring,[fromIndex]);
```

```
let expression = '1 + 2 = 3';
let matches = expression.match(/\d+/g);

for (const match of matches) {
  console.log(match);
}
```

```
let rawString = ' Hi  ';
let strippedString = rawString.trim();
console.log(strippedString); // "Hi"
```

```
let greeting = 'Hello';
console.log(greeting.toLowerCase()); // 'hello'
console.log(greeting.toUpperCase()); // 'HELLO';
```

```
let str = "the baby kicks the ball";

// replace "the" with "a"
let newStr = str.replace(/the/g, "a");
```

Boolean Type

- Type Boolean hanya True atau False
- Konversi ke Boolean

```
let inProgress = true;
let completed = false;

console.log(typeof completed); // boolean
```

```
let error = 'An error occurred';

if (error) {
  console.log(error);
}
```

```
console.log(Boolean('Hi'));// true
console.log(Boolean(''));  // false

console.log(Boolean(20));  // true
console.log(Boolean(Infinity));  // true
console.log(Boolean(0));  // false

console.log(Boolean({foo: 100}));  // true on non-empty object
console.log(Boolean(null));// false
```

Type	true	false
string	non-empty string	empty string
number	non-zero number and Infinity	0, NaN
object	non-null object	null
undefined		undefined

BigInt Type

- Menyelesaikan keterbatasan Integer
- Menyimpan lebih besar dari $2^{53} - 1$

```
let pageView = 9007199254740991n;
console.log(typeof(pageView)); // 'bigint'
```

Object Type

```
let emptyObject = {};
```

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

- Kumpulan dari propertis, dimana propertis adalah pasangan key dan nilainya
- Cara akses data pada Object Type:
- Menggunakan dot notation (.)
- Menggunakan kurung siku (membaca index)

```
let contact = {
    firstName: 'John',
    lastName: 'Doe',
    email: 'john.doe@example.com',
    phone: '(408)-555-9999',
    address: {
        building: '4000',
        street: 'North 1st street',
        city: 'San Jose',
        state: 'CA',
        country: 'USA'
    }
}
```

```
console.log(contact.firstName);
console.log(contact.lastName);
```

```
console.log(contact['phone']); // '(408)-555-9999'
console.log(contact['email']); // 'john.doe@example.com'
```

```
console.log(contact.age); // undefined
```

```
let address = {
    'building no': 3960,
    street: 'North 1st street',
    state: 'CA',
    country: 'USA'
};
```

```
address['building no'];
```

```
delete person.age;
```

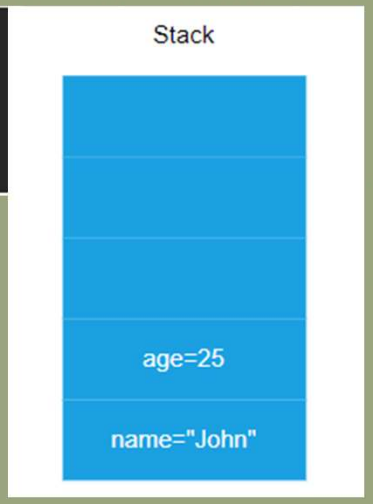
```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

person.firstName = 'Jane';

console.log(person);
```

```
let employee = {
    firstName: 'Peter',
    lastName: 'Doe',
    employeeId: 1
};

console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Stack and Heap Memory

- Ketika mendeklarasikan variabel, javascript mengalokasikan variabel tersebut pada stack dan heap memory
- Stack memory menyimpan variable yang ukurannya sudah fix yaitu variabel selain tipe object

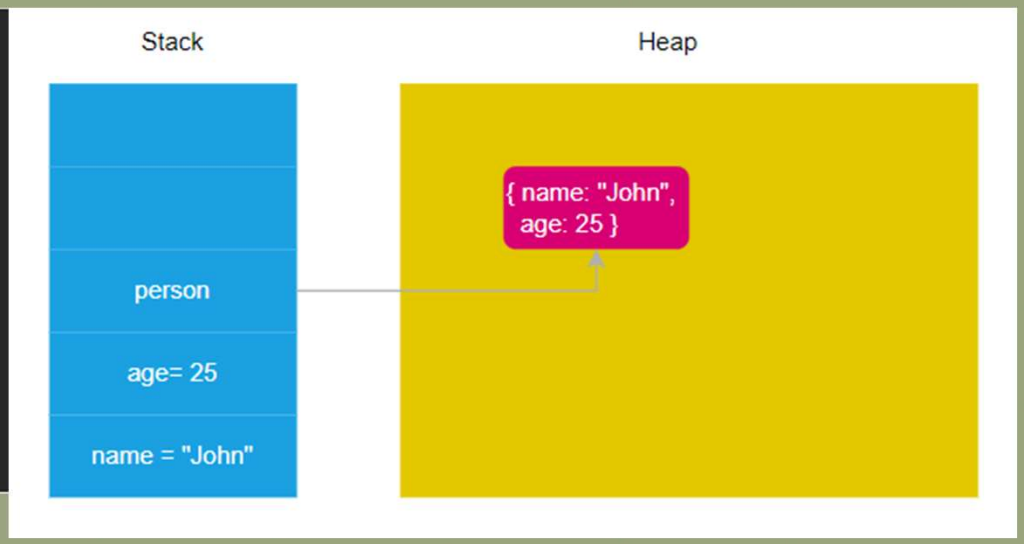
```
let name = 'John';
let age = 25;
```



Stack and Heap Memory (Sambungan)

- Dan menyimpan nama variable tipe objet, namun propertisnya disimpan pada heap memory

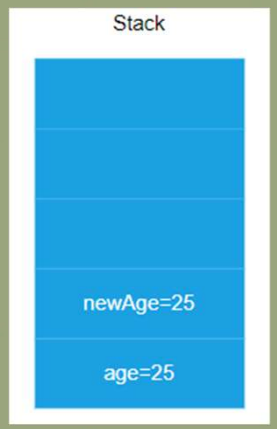
```
let name = 'John';
let age = 25;

let person = {
  name: 'John',
  age: 25,
};
```



Stack and Heap Memory (Sambungan)

```
let person = {
  name: 'John',
  age: 25,
};

// add the ssn property
person.ssn = '123-45-6789';

// change the name
person.name = 'John Doe';

// delete the age property
delete person.age;


console.log(person);
```

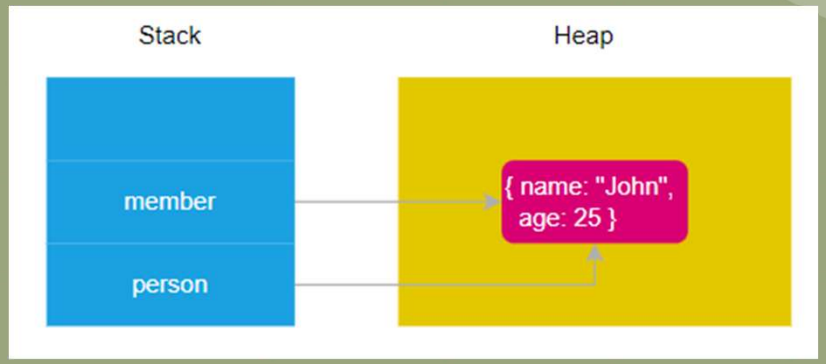
```
let age = 25;
let newAge = age;
```

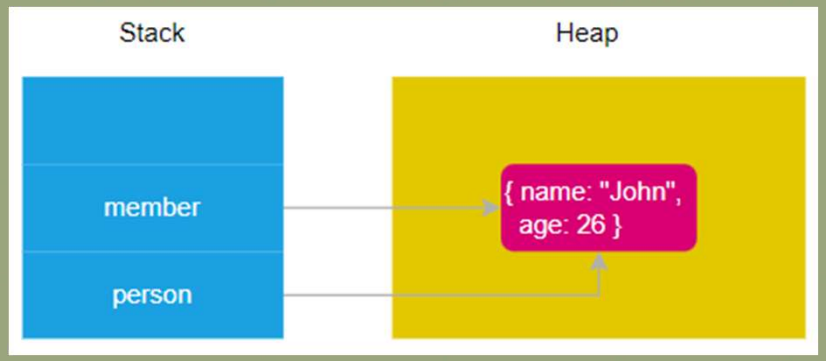



```
let age = 25;
let newAge = age;

newAge = newAge + 1;
```

Stack and Heap Memory (Sambungan)

```
let person = {
  name: 'John',
  age: 25,
};

let member = person;

member.age = 26;

console.log(person);
console.log(member);
```





Scope Variabel

- Scope Variabel menetapkan visibility dan accessibility pada variable
- Javascript memiliki 3 scope: Global, Local, dan Block
- Ketika deklarasi variabel menggunakan var, maka scopenya global atau local.
- Deklarasi diluar fungsi maka scopenya global
- Deklarasi didalam fungsi maka scopenya local
- Perkembangan berikutnya deklarasi variabel menggunakan let dan const, let sama dengan var kecuali scopenya. Let sama dengan const, kecuali nilai pada const tidak dapat diubah karena suatu konstanta kecuali tipe data object
- Scope let adalah dalam block. Dalam block ditandai dengan kurung kurawal

Scope Variabel

- Pada ES5, Ketika deklarasi sebuah variable menggunakan var, maka scopenya global dan local. Jika deklarasi dilakukan diluar sebuah fungsi maka scopenya adalah global, ketika deklarasi dilakukan di dalam sebuah fungsi maka scopenya adalah local.

Scope Variabel (Sambungan)

```
var message = 'Hi';

function say() {
    var message = 'Hello';
    console.log(message);
}

say();
console.log(message);
```

```
var message = 'Hi';

function say() {
    console.log(message);
}

say();
```

```
var y = 20;

function bar() {
    var y = 200;

    function baz() {
        console.log(y);
    }

    baz();
}

bar();
```

```
if(condition) {
    // inside a block
}
```

```
let x = 10;
if (x == 10) {
    let x = 20;
    console.log(x); // 20:  reference x inside the block
}
console.log(x); // 10: reference at the begining of the script
```

Scope Variabel (Sambungan)

```
var greeter = "hey hi";
var times = 4;

if (times > 3) {
    var greeter = "say Hello instead";
}

console.log(greeter) // "say Hello instead"
```

```
var tester = "hey hi";

function newFunction() {
    var hello = "hello";
}
console.log(hello); // error: hello is not defined
```

```
let greeting = "say Hi";
if (true) {
    let greeting = "say Hello instead";
    console.log(greeting); // "say Hello instead"
}
console.log(greeting); // "say Hi"
```

```
let greeting = "say Hi";
let times = 4;

if (times > 3) {
    let hello = "say Hello instead";
    console.log(hello);// "say Hello instead"
}
console.log(hello) // hello is not defined
```

Scope Variabel (Sambungan)

```
const nama = {
    namaDepan: 'Yasya',
    namaBelakang: 'El Hakim'
};

nama = {
    namaDepan: 'Agung',
    namaBelakang: 'Cahyadi'
}; // Akan langsung menghasilkan Error: invalid assignment to const 'nama'
```

```
const nama = {
    namaDepan: 'Yasya',
    namaBelakang: 'El Hakim'
};

nama.namaDepan = 'Agung'; // Akan menghasilkan 'Agung'
nama.namaBelakang = 'Cahyadi'; // Akan menghasilkan 'Cahyadi'

console.log(nama.namaDepan + nama.namaBelakang); // Akan menghasilkan 'AgungCahyadi'
```

```
const company = Object.freeze({
    name: 'ABC corp',
    address: {
        street: 'North 1st street',
        city: 'San Jose',
        state: 'CA',
        zipcode: 95134
    }
});
```

Operators

- Arithmetic Operators
- Khusus untuk penambahan, apabila salah satu operan yang dijumlahkan tidak bertipe integer maka hasilnya adalah penggabungan operan yang dijumlahkan

Operator	Sign
Addition	+
Subtraction	-
Multiplication	*
Division	/

Operators (Sambungan)

- Remainder Operator, operator yang digunakan adalah %
- Hasil remainder operator menghasilkan sisa hasil bagi
- Pada javascript remainder (%) berbeda dengan modulus (mod)

```
const mod = (dividend, divisor) => ((dividend % divisor) + divisor) % divisor;

// dividen and divisor have the same sign
console.log('remainder:', 5 % 3); // 2
console.log('modulo:', mod(5, 3)); // 2

// dividen and divisor have the different signs
console.log('remainder:', -5 % 3); // -2
console.log('modulo:', mod(-5, 3)); // 1
```

Operators (Sambungan)

- Assignment Operator
- Operator ini untuk memberi nilai pada sebuah variabel

```
let a = 10, b = 20, c = 30;
a = b = c; // all variables are 30
```

```
let a = 10, b = 20, c = 30;

b = c; // b is 30
a = b; // a is also 30
```

Operator	Meaning	Description
a = b	a = b	Assigns the value of b to a.
a += b	a = a + b	Assigns the result of a plus b to a.
a -= b	a = a - b	Assigns the result of a minus b to a.
a *= b	a = a * b	Assigns the result of a times b to a.
a /= b	a = a / b	Assigns the result of a divided by b to a.
a %= b	a = a % b	Assigns the result of a modulo b to a.
a &=b	a = a & b	Assigns the result of a AND b to a.
a |=b	a = a | b	Assigns the result of a OR b to a.
a ^=b	a = a ^ b	Assigns the result of a XOR b to a.
a <<= b	a = a << b	Assigns the result of a shifted left by b to a.
a >>= b	a = a >> b	Assigns the result of a shifted right (sign preserved) by b to a.
a >>>= b	a = a >>> b	Assigns the result of a shifted right by b to a.

Operators (Sambungan)

Unary Operators	Name
+x	Unary Plus
-x	Unary Minus
++x	Increment Operator (Prefix)
–x	Decrement Operator (Prefix)
x++	Increment Operator (Postfix)
x–	Decrement Operator (Postfix)

```
let x = 10;
let y = +x;
```

```
let s = '10';
console.log(+s);
```

```
let f = false,
    t = true;

console.log(+f);
console.log(+t);
```

```
let x = 10;
let y = -x;
```

```
let age = 25;
++age;

console.log(age);
```

```
let age = 25;
age = age + 1;
console.log(age);
```

```
let weight = 90;
weight = ++weight + 5;
```

```
let weight = 90;
let newWeight = weight++ + 5;
```

+ atau – hanya menconvert ke number, dan unary – akan mengubah menjadi negative setelah convert ke number

Operators (Sambungan)

- Comparison Operators
- Membandingkan string mengacu pada Unicode https://en.wikipedia.org/wiki/List_of_Unicode_characters
- strict equal dan not strict equal

Operator	Meaning
<	less than
>	greater than
<=	less than or equal to
>=	greater than or equal to
==	equal to
!=	not equal to

Operator	Meaning
===	strict equal
!==	not strict equal

```
let name1 = 'alice',
    name2 = 'bob';

let result = name1 < name2;
```

Logical Operators

- Javascript menyediakan 3 Logical Operator, yaitu:
- ! (Logical NOT)
- && (Logical AND)
- || (Logical OR)

```
let eligible = false,
    required = true;

console.log(!eligible);
console.log(!required);
```

```
console.log(!undefined); // true
console.log(!null); // true
console.log(!20); //false
console.log(!0); //true
console.log(!NaN); //true
console.log(!{}); // false
console.log(!''); //true
console.log(!'OK'); //false
console.log(!false); //true
console.log(!true); //false
```

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

Covention Name

- Penamaan variable Javascript adalah case-sensitive, huruf besar dan huruf kecil dibedakan
- Rekomendasi penamaan adalah camel case
- Selain camel case ada snake case (pakai underscore) atau pascal case (mirip camel case)
- Penamaan variabel menggambarkan isinya (informatif)
- Penamaan nama fungsi sama seperti aturan penaman variabel

```
var DogName = 'Scooby-Doo';
var dogName = 'Droopy';
var DOGNAME = 'Odie';
```

```
// bad
var dogname = 'Droopy';
// bad
var dog_name = 'Droopy';
// bad
var DOGNAME = ‘Droopy’;
// bad
var DOG_NAME = 'Droopy';
// good
var dogName = 'Droopy';
```

```
// bad
function name(dogName, ownerName) {
  return '${dogName} ${ownerName}';
}

// good
function getName(dogName, ownerName) {
  return '${dogName} ${ownerName}';
}
```

Covention Name (Sambungan)

- Rekomendasi penamaan variable type constant Upper Snake Case, yang menandakan nilainya tidak bisa diubah
- Penamaan class mirip dengan aturan penamaan variabel dan fungsi, namun rekomendasi menggunakan pascal case
- Penamaan method sama dengan penamaan fungsi
- Rekomendasi untuk yang case sensitive, penamaan file huruf kecil semua

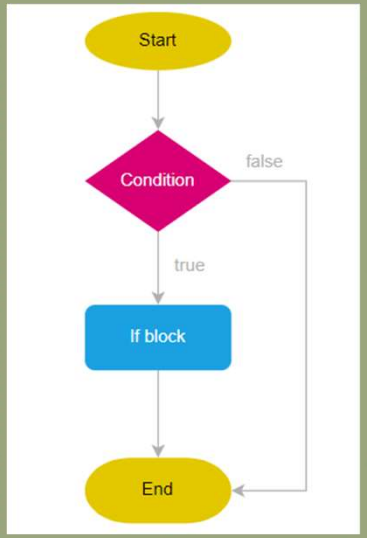
```
class DogCartoon {
  constructor(dogName, ownerName) {
    this.dogName = dogName;
    this.ownerName = ownerName;
  }
}

var cartoon = new DogCartoon('Scooby-Doo', 'Shaggy');
```

Coversion

- Type Conversion adalah proses mengubah type data menjadi type data lainnya.
- Bisa 2 cara yaitu, secara implicit dan secara explicit
- Secara implicit artinya secara otomatis setelah ekseskusi kode program, sering disebut Coersion
- Secara explicit artinya dilakukan oleh developer, sering disebut type casting

Control Flow Statements

- Penggunaan IF untuk mengerjakan statements yang berada dalam block ketika kondisi True
- Kondisi dapat berupa nilai atau ekspresi
- Rekomendasi tetap pakai kurung kurawal, untuk kemudahan maintain kode program




```
if( condition )
    statement;
```

```
if (condition) {
  // statements to execute
}
```

```
let age = 18;
if (age >= 18) {
  console.log('You can sign up');
}
```

Control Flow Statements (Sambungan)

- Nested If adalah ketika ada IF didalam IF
- Sebisa mengkin menhindari penggunaan Nested if

```
let age = 16;
let state = 'CA';

if (state == 'CA') {
  if (age >= 16) {
    console.log('You can drive.');
  }
}
```

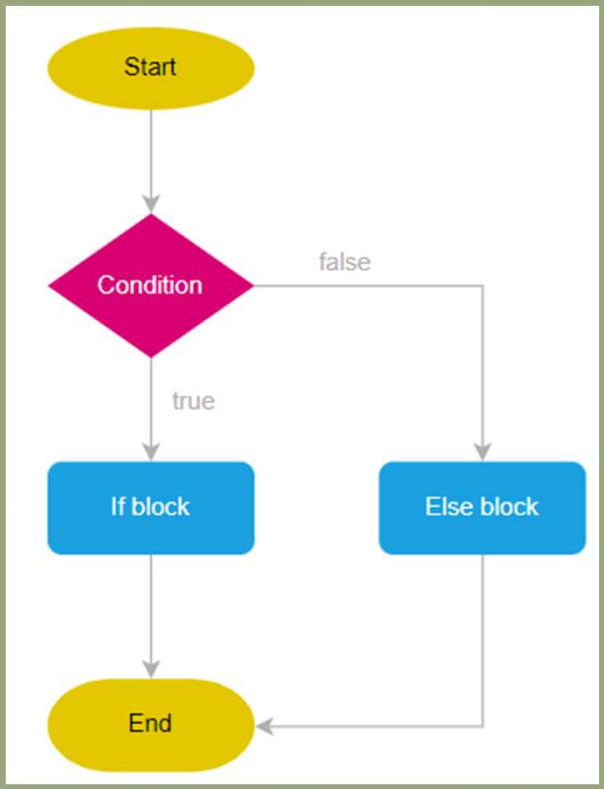
```
let age = 16;
let state = 'CA';

if (state == 'CA' && age == 16) {
  console.log('You can drive.');
}
```

Control Flow Statements (Sambungan)

- Penggunaan IF jika kondisi True ingin mengerjakan kode program dalam block, jika kondisi False tidak melakukan apa-apa
- Penggunaan IF..ELSE jika kondisi False ingin mengerjakan kode program dalam block

```
if( condition ) {
  // ...
} else {
  // ...
}
```

```
let age = 16;
let country = 'USA';

if (age >= 16 && country === 'USA') {
  console.log('You can get a driving license.');
} else {
  console.log('You are not eligible to get a driving license.');
}
```

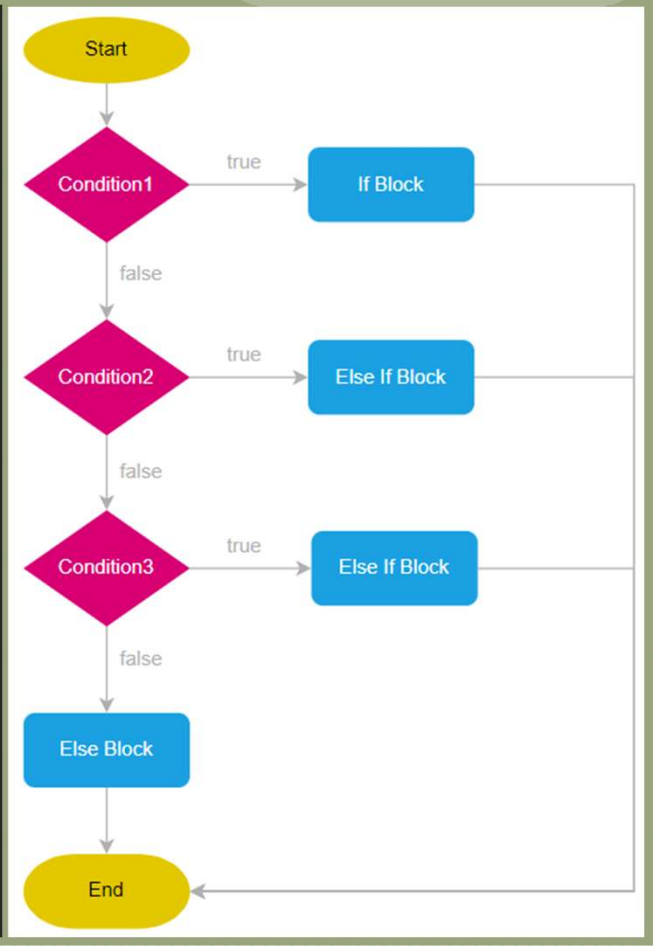


Control Flow Statements (Sambungan)

- Penggunaan IF..ELSE..IF jika mengecek banyak kondisi

```
if (condition1) {
  // ...
} else if (condition2) {
  // ...
} else if (condition3) {
  //...
} else {
  //...
}
```

```
let weight = 70; // kg
let height = 1.72; // meter

// calculate the body mass index (BMI)
let bmi = weight / (height * height);

let weightStatus;

if (bmi < 18.5) {
  weightStatus = 'Underweight';
} else if (bmi >= 18.5 && bmi <= 24.9) {
  weightStatus = 'Healthy Weight';
} else if (bmi >= 25 && bmi <= 29.9) {
  weightStatus = 'Overweight';
} else {
  weightStatus = 'Obesity';
}

console.log(weightStatus);
```



Control Flow Statements (Sambungan)

- Ternary Operator adalah alternatif penulisan kondisi IF..ELSE

```
let age = 18;
let message;

if (age >= 16) {
  message = 'You can drive.';
} else {
  message = 'You cannot drive.';
}

console.log(message);
```

```
condition ? expressionIfTrue : expressionIfFalse;
```

```
let age = 18;
let message;

age >= 16 ? (message = 'You can drive.') : (message = 'You cannot drive.');

console.log(message);
```

```
let variableName = condition ? expressionIfTrue : expressionIfFalse;
```

```
let locked = 1;
let canChange = locked != 1 ? true : false;
```

```
let age = 18;
let message;

message = age >= 16 ? 'You can drive.' : 'You cannot drive.';

console.log(message);
```

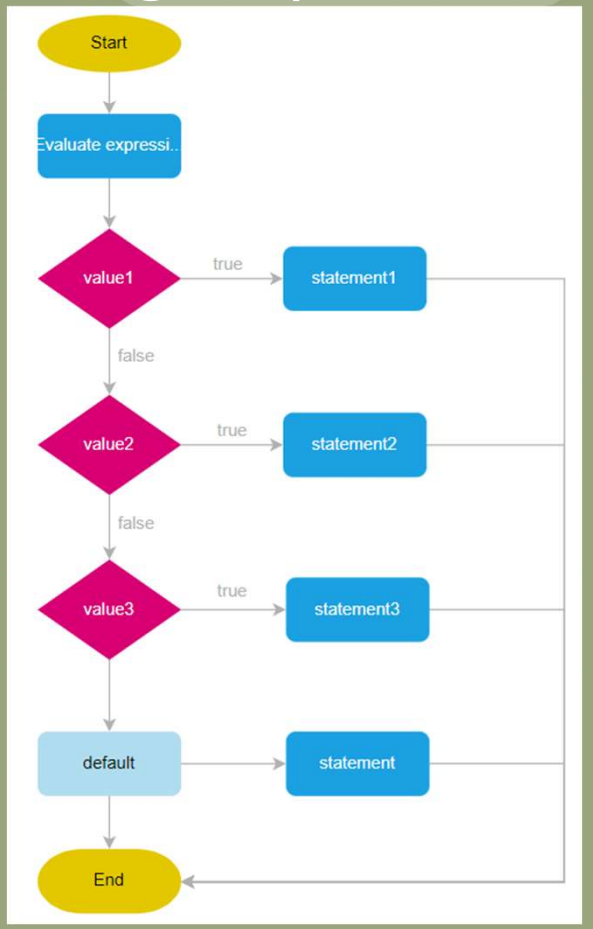
```
let speed = 90;
let message = speed >= 120 ? 'Too Fast' : speed >= 80 ? 'Fast' : 'OK';

console.log(message);
```

Control Flow Statements (Sambungan)

- Penggunaan SWITCH CASE untuk mengerjakan kode program dalam block
- Switch mengevaluasi ekspresi, hasilnya dibandingkan dengan nilai pada case
- Alternatif penggunaan IF..ELSE..IF

```
if (expression === value1) {
  statement1;
} else if (expression === value2) {
  statement2;
} else if (expression === value3) {
  statement3;
} else {
  statement;
}
```

```
switch (expression) {
    case value1:
        statement1;
        break;
    case value2:
        statement2;
        break;
    case value3:
        statement3;
        break;
    default:
        statement;
}
```



Control Flow Statements (Sambungan)

```
let day = 3;
let dayName;

switch (day) {
  case 1:
    dayName = 'Sunday';
    break;
  case 2:
    dayName = 'Monday';
    break;
  case 3:
    dayName = 'Tuesday';
    break;
  case 4:
    dayName = 'Wednesday';
    break;
  case 5:
    dayName = 'Thursday';
    break;
  case 6:
    dayName = 'Friday';
    break;
  case 7:
    dayName = 'Saturday';
    break;
  default:
    dayName = 'Invalid day';
}

console.log(dayName); // Tuesday
```

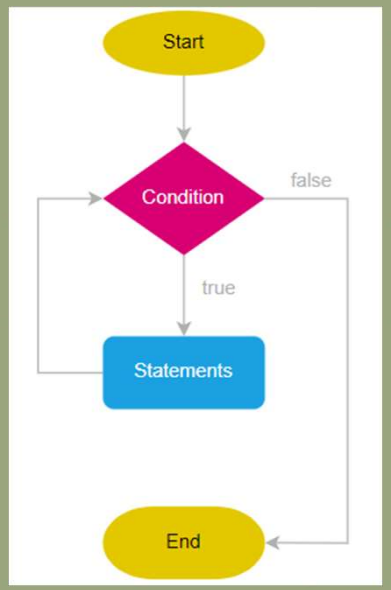
```
let year = 2016;
let month = 2;
let dayCount;

switch (month) {
  case 1:
  case 3:
  case 5:
  case 7:
  case 8:
  case 10:
  case 12:
    dayCount = 31;
    break;
  case 4:
  case 6:
  case 9:
  case 11:
    dayCount = 30;
    break;
  case 2:
    // Leap year
    if ((year % 4 == 0 && !(year % 100 == 0)) || year % 400 == 0) {
      dayCount = 29;
    } else {
      dayCount = 28;
    }
    break;
  default:
    dayCount = -1; // invalid month
}

console.log(dayCount); // 29
```

Control Flow Statements (Sambungan)

- Jika block berada di dalam WHILE

```
while (expression) {
    // statement
}
```

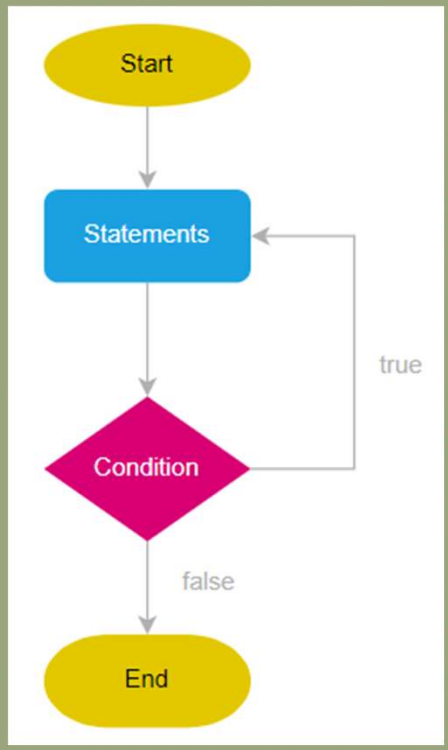
```
let count = 1;
while (count < 10) {
    console.log(count);
    count +=2;
}
```



- Jika block berada di dalam Do..WHILE

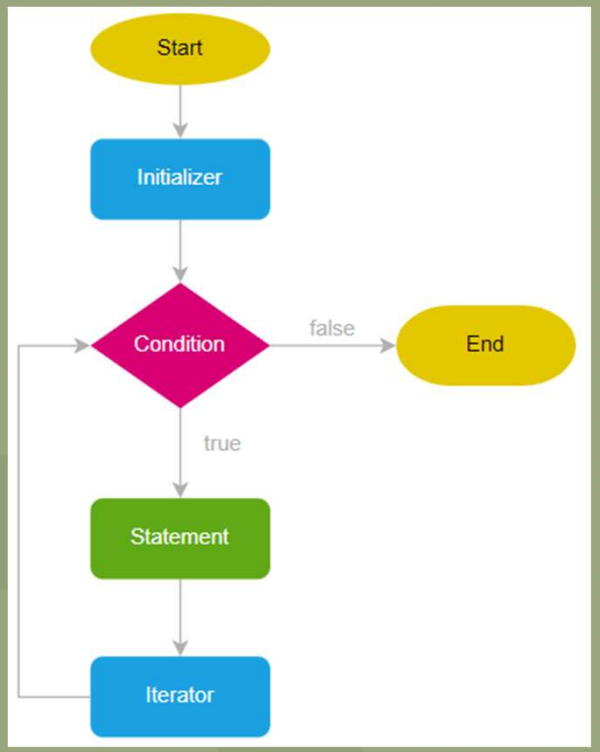
```
do {
  statement;
} while(expression)
```

```
let count = 0;
do {
  console.log(count);
  count++;
} while (count < 5)
```



Control Flow Statements (Sambungan)

- Jika block berada di dalam For




```
for (initializer; condition; iterator) {
    // statements
}
```

```
for (let i = 1; i < 5; i++) {
  console.log(i);
}
```

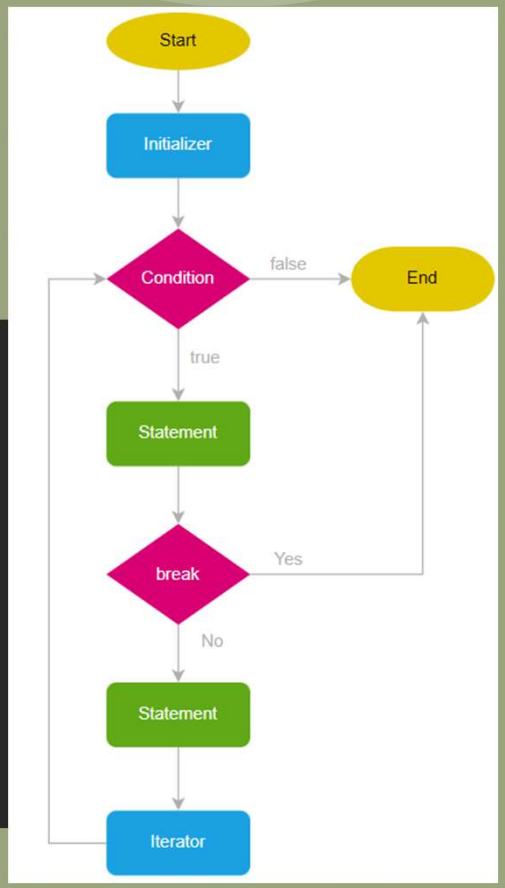
```
let sum = 0;
for (let i = 0; i <= 9; i++, sum += i);
console.log(sum);
```

Control Flow Statements (Sambungan)

- BREAK digunakan jika ingin keluar dari proses loop yang sedang berjalan
- Terminate loop dimana break berada.

```
for (let i = 0; i < 5; i++) {
  console.log(i);
  if (i == 2) {
    break;
  }
}
```

```
for (let i = 1; i <= 3; i++) {
  for (let j = 1; j <= 3; j++) {
    if (i + j == 4) {
      break;
    }
    console.log(i, j);
  }
}
```

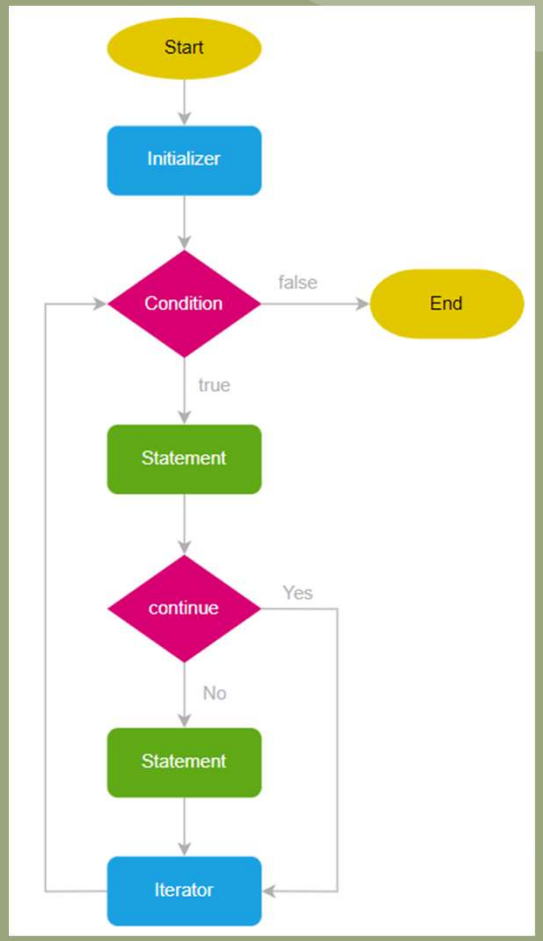


Control Flow Statements (Sambungan)

- Continue digunakan jika ingin melewatkan proses iterasi loop yang sedang berjalan dan meneruskan ke iterasi selanjutnya

```
// inside a loop
if(condition){
  continue;
}
```

```
for (let i = 0; i < 10; i++) {
  if (i % 2 === 0) {
    continue;
  }
  console.log(i);
}
```



Exception

- Try..catch

```
let result = add(10, 20);
console.log(result);

console.log('Bye');
```

```
Uncaught TypeError: add is not a function
```

```
try {
  // code may cause error
} catch(error){
  // code to handle error
}
```

- Pada dasarnya error memiliki paling sedikit 2 propertis yaitu name dan message

```
try {
  let result = add(10, 20);
  console.log(result);
} catch (e) {
  console.log({ name: e.name, message: e.message });
}
console.log('Bye');
```

Exception (Sambungan)

- Try..catch..finally

```
let result = 0;
try {
  result = add(10, 20);
} catch (e) {
  console.log(e.message);
} finally {
  console.log({ result });
}
```

```
try {
  // code may cause exceptions
} catch (error) {
  // code to handle exceptions
} finally {
  // code to execute whether exceptions occur or not
}
```

```
const add = (x, y) => x + y;

let result = 0;

try {
  result = add(10, 20);
} catch (e) {
  console.log(e.message);
} finally {
  console.log({ result });
}
```

Exception (Sambungan)

- throw

```
function add(x, y) {
  if (typeof x !== 'number') {
    throw new Error('The first argument must be a number');
  }
  if (typeof y !== 'number') {
    throw new Error('The second argument must be a number');
  }

  return x + y;
}

try {
  const result = add('a', 10);
  console.log(result);
} catch (e) {
  console.log(e.name, ':', e.message);
}
```

```
function add(x, y) {
  if (typeof x !== 'number') {
    throw 'The first argument must be a number';
  }
  if (typeof y !== 'number') {
    throw 'The second argument must be a number';
  }

  return x + y;
}

try {
  const result = add('a', 10);
  console.log(result);
} catch (e) {
  console.log(e);
}
```

Iteration

- Menurut KBBI iterasi adalah perulangan

```
let ranks = ['A', 'B', 'C'];

for (let i = 0; i < ranks.length; i++) {
    console.log(ranks[i]);
}
```

```
for(let rank of ranks) {
    console.log(rank);
}
```

```
let colors = ['Red', 'Green', 'Blue'];

for (const [index, color] of colors.entries()) {
    console.log(`${color} is at index ${index}`);
}
```

```
const user = {
    name: 'John Doe',
    email: 'john.doe@example.com',
    age: 25,
    dob: '08/02/1989',
    active: true
};

// iterate over the user object
for (const key in user) {
    console.log(`${key}: ${user[key]}`);
}

// name: John Doe
// email: john.doe@example.com
// age: 25
// dob: 08/02/1989
```

Fungsi

- Reuse

```
function functionName(parameters) {
    // function body
    // ...
}
```

```
function say(message) {
    console.log(message);
}

say('Hello');
```

```
function say(message) {
    console.log(message);
}

let result = say('Hello');
console.log('Result:', result);
```

```
function add(a, b) {
    return a + b;
}

let sum = add(10, 20);
console.log('Sum:', sum);
```

```
function add() {
    let sum = 0;
    for (let i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }
    return sum;
}
```

```
let person = {
  name: 'John',
  age: 25,
};

function increaseAge(obj) {
  obj.age += 1;
}

increaseAge(person);

console.log(person);
```

```
console.log(add(1, 2)); // 3
console.log(add(1, 2, 3, 4, 5)); // 15
```

Fungsi (Sambungan)

Built-in Fungsi
1. Fungsi Array
2. Fungsi Matematika
3. Fungsi String
4. Fungsi Tanggal dan Waktu

- Fungsi sebagai parameter, disebut juga callback function

- Anonymous Function, fungsi tanpa nama,fungsi sebagai ekspresi dan dieksekusi langsung ketika membuatnya
- Cara panggil dengan diassign ke sebuah variabel

```
function add(a, b) {
    return a + b;
}

let sum = add;

function average(a, b, fn) {
    return fn(a, b) / 2;
}

let result = average(10, 20, sum);

console.log(result);
```

```
(function () {
    //...
});
```

```
let show = function() {
    console.log('Anonymous function');
};

show();
```

Fungsi (Sambungan)

- Anonymous Fungsi sebagai arguments

```
setTimeout(function() {
    console.log('Execute later after 1 second')
}, 1000);
```

- Cara panggil secara langsung

```
(function () {
    console.log('Immediately invoked function execution');
})();
```

Kirim parameter person

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

(function () {
    console.log(person.firstName} + ' ' + person.lastName);
})(person);
```

Fungsi (Sambungan)

- Arrow Function

```
let show = function () {
    console.log('Anonymous function');
};
```

- Rekursif yaitu fungsi yang memanggil diri sendiri

```
function countDown(fromNumber) {
    console.log(fromNumber);
    countDown(fromNumber-1);
}

countDown(3);
```

```
function countDown(fromNumber) {
    console.log(fromNumber);

    let nextNumber = fromNumber - 1;

    if (nextNumber > 0) {
        countDown(nextNumber);
    }
}
countDown(3);
```

Sebagai argument pada method map

```
let names = ['John', 'Mac', 'Peter'];
let lengths = names.map(name => name.length);

console.log(lengths);
```

```
let show = () => console.log('Anonymous function');
```

```
let add = function (a, b) {
    return a + b;
};
```

```
let add = (a, b) => a + b;
```

Fungsi (Sambungan)

- Default parameter pada fungsi

```
function say(message='Hi') {
    console.log(message);
}

say(); // 'Hi'
say(undefined); // 'Hi'
say('Hello'); // 'Hello'
```

```
function say(message) {
    message = typeof message !== 'undefined' ? message : 'Hi';
    console.log(message);
}
say(); // 'Hi'
```

Fungsi (Sambungan)

- Fungsi dengan keluaran multiple value, dalam array atau object

```
function getNames() {
    // get names from the database or API
    let firstName = 'John',
        lastName = 'Doe';

    // return as an array
    return [firstName, lastName];
}
```

```
let names = getNames();
```

```
const [firstName, lastName] = getNames();
```

```
const firstName = names[0],
      lastName  = names[1];
```

```
function getNames() {
  // get names from the database or API
  let firstName = 'John',
      lastName = 'Doe';

  // return values
  return {
    'firstName': firstName,
    'lastName': lastName
  };
}
```

Data Structure

- Struktur data merupakan sekumpulan data yang saling berkaitan dan dapat dioperasikan.
- Struktur data bertujuan untuk memberi kemudahan mengakses dan memodifikasi sekumpulan data
- Array, selalu dimulai dari index 0. Cara akses menggunakan indexnya

```
const arr = ['a', 'b', 'c', 'd']
console.log(arr[2]) // c
```

```
const arr = ['store', 1, 'whatever', 2, 'you want', 3]
```

```
const arr = [
    [1,2,3],
    [4,5,6],
    [7,8,9],
]
```

Data Structure (Sambungan)

- Object, yaitu kumpulan pasangan key-value yang biasa disebut properti
- Properti sebuah object berupa pasangan key-value maupun fungsi. Fungsi dalam object disebut metod

```
const obj = {
    prop1: "Hello!",
    prop3: function() {console.log("I'm a property dude!")
}}
```

```
obj.prop4 = 125
obj["prop5"] = "The new prop on the block"
obj.prop6 = () => console.log("yet another example")
```

```
console.log(obj.prop1) // "Hello!"
console.log(obj["prop1"]) // "Hello!"
obj.prop3() // "I'm a property dude!"
```

Object Type

- Properti object dibuat melalui inisialisasi atau bisa melalui assign value

```
var person = {
    firstName: 'John',
    lastName: 'Doe',
    ssn: '299-24-2351'
};

for(var prop in person) {
    console.log(prop + ':' + person[prop]);
}
```

```
var person={
    firstname:"Budi",
    lastname:"Hartono"
}

let member=Object.create(person);
let memberPerson=Object.assign(person);
```

```
var decoration = {
    color: 'red'
};

var circle = Object.create(decoration);
circle.radius = 10;


for(const prop in circle) {
    console.log(prop);
}
```

```
const person = {
    firstName: 'John',
    lastName: 'Doe',
    age: 25
};

const profile = Object.values(person);

console.log(profile);
```

```
[ 'John', 'Doe', 25 ]
```

Async Await

```
const promise = new Promise((resolve, reject) => {
  // contain an operation
  // ...

  // return the state
  if (success) {
    resolve(value);
  } else {
    reject(error);
  }
});
```

Pengolahan data object

```
function getUsers() {
  return [
    { username: 'john', email: 'john@test.com' },
    { username: 'jane', email: 'jane@test.com' },
  ];
}

function findUser(username) {
  const users = getUsers();
  const user = users.find((user) => user.username === username);
  return user;
}

console.log(findUser('john'));
```

```
{ username: 'john', email: 'john@test.com' }
```