



Array and Function

Array

Array merupakan koleksi data dari satu atau banyak *element*.

Data pada *array* dapat diakses melalui nomor indeks dari *array* tersebut

A	B	C	D	E
0	1	2	3	4

```
let arr = [];  
let arr1 = new Array();  
  
arr = ['A', 'B', 'C', 'D', 'E'];  
arr1 = [1, 2, 3, 4, 5];  
  
let students = [  
  {  
    name: 'Bagus',  
    email: `bagus@gmail.com`,  
  },  
  {  
    name: 'Bagus',  
    email: `bagus@gmail.com`,  
  },  
  {  
    name: 'Bagus',  
    email: `bagus@gmail.com`,  
  },  
  {  
    name: 'Bagus',  
    email: `bagus@gmail.com`,  
  },  
];
```



Array built-in methods

Array memiliki beberapa built in methods seperti *toString*, *join*, *pop*, dan *push*.

[Reference link](#)



Function

Function merupakan sebuah *block code*, yang dapat kita jalankan dengan cara kita panggil(atau *execute*) *function* tersebut.

Sebuah *function* juga dapat dipanggil di dalam *function* lainnya.

Kita juga dapat memasukkan *value* ke dalam *function*, dan *function* tersebut dapat mengembalikan *value* tersebut, baik sudah, maupun belum diproses.



Mendefinisikan function

Function dapat didefinisikan dengan dua cara:

- *Function declaration*
- *Function expression*



Function Declaration

Function declaration terdiri atas keyword *function*, yang lalu diikuti dengan:

1. Nama *function*.
2. Daftar *parameter* yang akan diterima *function*. Kalau *parameter* lebih dari satu, dipisahkan dengan koma.
3. Block code.

```
function square(number) {  
    return number * number;  
}
```



Function Expression

Function expression merupakan cara membuat sebuah *function*, yang dapat dibuat secara *anonymous*.

Yang membedakan adalah, kita memasukkan *function* ke dalam *variabel*.

```
const square = function (number) {  
  return number * number;  
};
```



Memanggil function

Untuk dapat menjalankan *code* yang ada didalam *function*, kita harus memanggilnya(atau *execute*).

Memanggil function itu *simple*, dengan menambahkan `()` setelah nama function.

```
console.log(square(10)); 100
```




Function scope

Sederhananya, *variable* di dalam *function*, tidak bisa diakses dari luar *function*.

```
const greeting = () => {  
  const hello = 'Hello';  
  
  return hello;  
};  
  
console.log(hello);  hello is not defined
```



Parameter & Argument

Argument merupakan *value* yang dimasukkan ke dalam sebuah *function*.

Sedangkan *parameter*, adalah nama *variable* yang digunakan untuk menampung *argument* yang dimasukkan ke dalam *function*.

```
const greeting = (name) => {  
  const hello = 'Hello';  
  
  return `${hello} ${name}`;  
};  
  
console.log(greeting('John Doe')); 'Hello John Doe'
```



Default Parameter

Secara default, nilai *parameter* (kalau ada) yang ada di dalam sebuah *function* adalah *undefined*.

Jadi, kita dapat memberikan nilai *default* secara manual terhadap *parameter* tersebut.

Di beberapa kasus, default *parameter* dapat mencegah terjadinya error.

```
const greeting = (name = 'John Doe') => {  
  const hello = 'Hello';  
  
  return `${hello} ${name}`;  
};  
  
console.log(greeting()); // 'Hello John Doe'
```



Rest Parameters

Merupakan sebuah *syntax* yang dapat membuat function menerima *argument* yang banyak, namun tidak perlu membuat banyak *parameter*.

Hanya *parameter* terakhir yang dapat diberi peran sebagai *rest parameter*.

```
const greeting = (name = 'John Doe', ...restName) => {  
  const hello = 'Hello';  
  
  return 'Hello' + ' [restName]: ' + restName + ' [name]: ' + name;  
};  
  
console.log(greeting('Bagus', 'Jegeg', 'Wayan', 'Nyoman')); 'Hello [restName]: Jegeg,Wayan,Nyoman [name]: Bagus'
```



Nested Function

Kita dapat membuat satu ataupun lebih *function* didalam *function*.

Inner function dapat mengakses *variable* maupun *parameter* yang ada diluar dirinya sendiri..

```
const greeting = (name = 'John Doe') => {  
  const hello = 'Hello';  
  const defineGender = (name) => {  
    if (name === 'Bagus') {  
      return 'Male';  
    }  
    return 'Unknown';  
  };  
  return `${name} | Gender : ${defineGender()}`;  
};  
  
console.log(greeting()); 'John Doe | Gender : Unknown'
```



Recursive Function

Merupakan *function* yang memanggil dirinya sendiri, sampai ia tidak memanggil dirinya sendiri.

```
function countdown(fromNumber) {  
  console.log(fromNumber); 10, 9, 8, 7, 6, 5, 4, 3, 2, 1  
  
  let nextNumber = fromNumber - 1;  
  
  if (nextNumber > 0) {  
    countdown(nextNumber);  
  }  
}  
  
countdown(10);
```



Arrow Function

Merupakan sebuah bentuk lebih sederhana dari sebuah *function*.

```
const sayHi = () => {  
  return 'Hi Friend';  
};
```



Predefined Function

Predefined Function mirip dengan *built-in methods*. Bedanya, kita tidak harus membuat sebuah *object* ataupun *variable*, untuk memanggil *built-in methods* tersebut. Bisa dibilang, *predifined function* adalah *build-in methods global*.

Contoh:

- `isNaN()` → Mengecek apakah sebuah *value* NaN atau tidak.
- `parseInt()` → Mengembalikan nilai integer dari sebuah *value*.



Exercise

- Buat function yang dapat membuat triangle seperti ini:


01

02 03

04 05 06

07 08 09 10

- Parameters: height → Merupakan tinggi dari segitiga yang akan kita buat.

- 
- Buat function yang akan `console.log(n)` dari 0 sampai ke N.
 - Jika angka sama dengan perkalian 3 `console.log "TIGA"`
 - Jika angka sama dengan perkalian 5 `console.log "LIMA"`
 - Jika angka sama dengan perkalian 3 & 5 `console.log "TIGA LIMA"`
 - Parameter : n



- Buat function untuk hitung Body Mass Index (BMI)
- Formula : $BMI = \text{berat (kg)} / (\text{tinggi (meter)})^2$
- Return
 - < 18.5 return “berat kurang”
 - $18.5 - 24.9$ return “ideal”
 - $25.0 - 29.9$ return “kelebihan berat badan”
 - $30.0 - 34.9$ return “sangat kelebihan berat badan”
 - > 34.9 return “obesitas”
- Parameter : weight & height



- Tulis fungsi untuk menghapus semua angka ganjil dalam array dan mengembalikan array baru yang berisi bilangan genap saja.
- Parameter : arrayOfNumber



- Buat function untuk mengubah string menjadi array of string.
- Contoh : “Hello World” → [“Hello”, “Word”]
- Parameter : string