

Get Started

Modern JavaScript

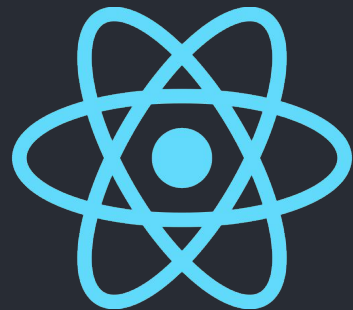


Table of Contents

- Variable
- Data Types
- Template Literals
- Conditional
- Looping
- Function
- Array
- Object

Variable

- Tempat untuk menyimpan nilai.
- Membuat variabel menggunakan keyword `let` dan `const`.
- Keyword `let` dapat diubah nilainya, sedangkan `const` tidak.
- Dahulukan penggunaan `const` dari pada `let`.
- `Const` bersifat immutable (tidak bisa re-assign, tapi bisa dimodifikasi).
- Name variables right.

Referensi: [JavaScript Info - Variables](#).



variable.js

```
/**  
 * Membuat variable dengan keyword const.  
 * Variable const tidak dapat diubah nilainya.  
 */  
const name = "Aufa Billah";  
const major = "Informatics";  
  
console.log(name, major);
```

Data Types

Nilai di JavaScript terdiri dari berbagai jenis (types).

Data Types (Primitives):

- Number: Nilai berupa angka (integer atau float).
- String: Nilai berupa kumpulan karakter.
- Boolean: Nilai yang terdiri dari `true` atau `false`.
- Null: Nilai yang tidak ada.
- Undefined: Nilai yang belum di-assign.

Primitives: hanya bisa menyimpan 1 nilai.

Gunakan operator `typeof` untuk mengecek tipe data.

Referensi: [JavaScript Info - Data Types](#).



data-types.js

```
const name = "Aufa Billah"; // string
const age = 23;              // number
const isMarried = false;    // boolean
const dateAt;               // undefined

/**
 * operator typeof untuk mengecek tipe data.
 */
console.log(typeof name, typeof age);
```

Template Literals

- Membuat string menggunakan kutip satu atau dua.
- Template Literals: membuat string menggunakan backtick: ``string``.

Supports:

- Multiline.
- interpolasi dan ekspresi: `${expression}`
- Tagged template (memanggil fungsi).

Referensi: [JavaScript Info - String](#).



template-literals.js

```
const name = "Aufa Billah";
const bod = 2001;

/**
 * Membuat string menggunakan template literals.
 * Dapat menggunakan multiline.
 * Dapat melakukan interpolasi.
 */
const greeting = `
Hello, my name ${name}.
Umur saya ${2022 - bod}
`;

console.log(greeting);
```


Conditional

Conditional

Menjalankan aksi tertentu berdasarkan kondisi tertentu.

Jenis:

- if: membuat satu kondisi.
- else if: membuat dua kondisi atau lebih.
- else: membuat kondisi terakhir.

Referensi: [JavaScript Info - Conditional](#).



conditional.js

```
const results = 85;

// if: membuat satu kondisi
if (results > 90) {
  console.log("Grade: A");
}
// else if: membuat 2 kondisi atau lebih
else if (results > 80) {
  console.log("Grade: B");
}
else if (results > 70) {
  console.log("Grade: C");
}
// else: membuat kondisi terakhir
else {
  console.log("Grade: D");
}
```

Ternary Operator

Menuliskan `if else` dengan cara yang lebih singkat.

Ternary operator menggunakan operator:

- `?` dijalankan ketika kondisi true.
- `:` dijalankan ketika kondisi false.

Referensi: [JavaScript Info - Ternary Operator](#).



ternary-operator.js

```
const age = 23;

if (age > 21) {
  console.log("Dewasa");
}
else {
  console.log("Belum Dewasa");
}
```



ternary-operator.js

```
const age = 23;

/**
 * Ternary operator digunakan untuk membuat if else lebih singkat.
 */
age > 21 ? console.log("Dewasa") : console.log("Belum Dewasa");
```



ternary-operator.js

```
const age = 23;

/**
 * Ternary operator dapat digunakan untuk menyimpan nilai ke variable.
 */
const status = age > 21 ? "Dewasa" : "Belum Dewasa";
console.log(status);
```

Looping



Looping

Melakukan operasi atau aksi yang berulang-ulang.

Basic Loops:

- While
- For
- Do

Referensi: [JavaScript Info - Loops](#).



for.js

```
/**
 * Looping menggunakan for.
 * Menampilkan angka 1 - 10.
 */
for (let i = 1; i < 11; i++) {
  console.log(`Perulangan ke: ${i}`);
}
```

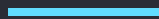


while.js

```
// membuat variable awal
let i = 1;

/**
 * Looping menggunakan while.
 * Membuat kondisi untuk batasan looping.
 */
while (i < 11) {
  console.log(`Perulangan ke: ${i}`);
  // melakukan increment
  i++;
}
```


Function



Function

- Sekumpulan kode yang menjalankan tugas tertentu.
- Reusable code.
- Function dapat memiliki parameter.
- Function dapat mengembalikan nilai (return).

Jenis:

- Function Declaration: Membuat fungsi menggunakan keyword `function`.
- Function Expression: Menyimpan fungsi ke variabel.
- Arrow Function: Sama seperti Function Expression, namun penulisan lebih singkat.

Referensi: [JavaScript Info - Function](#).



function.js

```
// menghitung umur  
const year = 2022;  
const bod = 2003;  
  
const age = year - bod;  
console.log(`Umur: ${age} tahun`);
```



function.js

```
// menghitung umur  
const year = 2022;  
const bod = 1999;  
  
const age = year - bod;  
console.log(`Umur: ${age} tahun`);
```



function.js

```
// menghitung umur  
const year = 2022;  
const bod = 2000;  
  
const age = year - bod;  
console.log(`Umur: ${age} tahun`);
```



function.js

```
// menghitung umur  
const year = 2022;  
const bod = 1990;  
  
const age = year - bod;  
console.log(`Umur: ${age} tahun`);
```



function-declaration.js

```
/**
 * Membuat fungsi menghitung umur.
 * Dibat menggunakan cara Function Declaration.
 *
 * @param {integer} bod (tanggal lahir)
 * @returns {integer} age (umur)
 */
function getAge(bod) {
  const year = 2022;
  const age = 2022 - bod;

  return age;
}

// Memanggil fungsi getAge
console.log(getAge(1997));
console.log(getAge(2000));
```



function-expression.js

```
/**
 * Membuat fungsi menghitung umur.
 * Dibat menggunakan cara Function
 * Expression.
 *
 * @param {integer} bod (tanggal lahir)
 * @returns {integer} age (umur)
 */
const getAge = function (bod) {
  const year = 2022;
  const age = year - bod;

  return age;
};

// Memanggil fungsi getAge
console.log(getAge(1997));
console.log(getAge(2000));
```



arrow-function.js

```
/**
 * Membuat fungsi menghitung umur.
 * Dibuat menggunakan cara Arrow Function.
 *
 * @param {integer} bod (tanggal lahir)
 * @returns {integer} age (umur)
 */
const getAge = (bod) => {
  const year = 2022;
  const age = year - bod;

  return age;
};

// Memanggil fungsi getAge
console.log(getAge(1997));
console.log(getAge(2000));
```



arrow-function.js

```
/**
 * Membuat fungsi menghitung umur.
 * Dibuat menggunakan cara Arrow Function.
 *
 * @param {integer} bod (tanggal lahir)
 * @returns {integer} age (umur)
 */
const getAge = (bod) => 2022 - bod;

// Memanggil fungsi getAge
console.log(getAge(1997));
console.log(getAge(2000));
```

Default Parameter

- Memberikan nilai default ke parameter.
- Mencegah terjadi error.
- Nilai default digunakan jika tidak ada parameter.

Referensi: [JavaScript Info - Default Value](#).



default-parameter.js

```
/**
 * Membuat fungsi menghitung umur.
 * Dibuat menggunakan cara Arrow Function.
 *
 * @param {integer} bod (tanggal lahir)
 * @returns {integer} age (umur)
 */
const getAge = (bod = 1999) => {
  const year = 2022;
  const age = year - bod;

  return age;
};

// Memanggil fungsi getAge
console.log(getAge());
console.log(getAge(2000));
```

Array

Array

- Salah satu jenis struktur data (data structure).
- Menyimpan banyak nilai dalam variabel.
- Disimpan dalam bentuk urutan (ordered): [0, 1, 2, 3, 4, 5].

Referensi: [JavaScript Info - Array](#).



array.js

```
// Problem without array  
const animal1 = "Cat";  
const animal2 = "Dog";  
const animal3 = "Fish";  
// . . . . .  
const animal100 = "Bird";
```



array.js

```
// Membuat variable array menggunakan const
const animals = ["Cat", "Dog", "Fish", "Bird"];

/**
 * Mengakses element atau nilai array.
 * Mengakses element berdasarkan index atau posisi.
 * Posisi (index) dimulai dari 0.
 */
console.log(animals[0], animals[1]);
```

Loops Array

Loops dapat digunakan untuk menampilkan seluruh data array.

Jenis:

- for/while/do-while: looping manual.
- for of: looping khusus untuk array.
- forEach: method khusus untuk array (HOF).

Referensi: [JavaScript Info - Array Methods](#).



looping-array.js

```
// Membuat variable array menggunakan const  
const animals = ["Cat", "Dog", "Fish", "Bird"];  
  
// Looping array menggunakan for  
for (let i = 0; i < animals.length; i++) {  
  console.log(`Hewan: ${animals[i]}`);  
}
```



for-of.js

```
// Membuat variable array menggunakan const  
const animals = ["Cat", "Dog", "Fish", "Bird"];  
  
// Looping array menggunakan for-of  
for (const animal of animals) {  
  console.log(`Hewan: ${animal}`);  
}
```

Object

Object

- Salah satu jenis struktur data (data structure).
- Menyimpan data yang lebih kompleks (banyak nilai).
- Nilai disimpan dalam bentuk `key:value`, bukan urutan (ordered).
- Object selalu digunakan di berbagai tempat di JavaScript.
- Object mirip seperti array asosiatif di bahasa pemrograman lain (PHP).

Referensi: [JavaScript Info - Object](#).



object.js

```
/**
 * Membuat object menggunakan {}.
 * Menyimpan nilai dengan format key : value
 */
const user = {
  name: "Aufa Billah",
  age: 22,
  major: "Informatics",
};

/**
 * Mengakses nilai object menggunakan key.
 * Cara akses bisa menggunakan dot atau bracket
 */
console.log(user.name, user["age"]);
```


Loops Object

- Loops dapat digunakan untuk menampilkan seluruh nilai object.
- Pada object tidak dapat menggunakan loops biasa: for/while/do-while.
- Loops khusus pada object menggunakan method: `for-in`.

Referensi: [JavaScript Info - for-in](#).



for-in.js

```
/**
 * Membuat object menggunakan {}.
 * Menyimpan nilai dengan format key : value
 */
const user = {
  name: "Aufa Billah",
  age: 22,
  major: "Informatics",
};

/**
 * Looping object menggunakan for-in.
 * Mengakses nilai menggunakan cara bracket.
 */
for (const key in user) {
  console.log(`${key}: ${user[key]}`);
}
```

QnA

Take Away

Task

- Task boilerplate and description: [Link](#).
- Practice Data Structure and Algorithm: Array and Object.
- Practice Modern JavaScript before learning React.
- Practice TDD (Test Driven Development).

Assignment:

- Push code to Repository Github (use branch for task management).
- Submit link repository to elena.
- Kamis: 03 Maret 2022, [Link Elena](#).
- Jumat: 05 Maret 2022, [Link Elena](#).



Main Function

```
const main = () => {  
  console.log("# Get All Users");  
  all();  
  
  console.log("# Add New User: Sabiq");  
  const newUser = {  
    name: "Sabiq",  
    age: 20,  
    major: "Informatics",  
  };  
  store(newUser);  
  
  console.log("# Edit User: Isfa");  
  const editedUser = {  
    name: "Isfhani Ghiyath",  
    age: 23,  
    major: "English",  
  };  
  update(1, editedUser);  
  
  console.log("# Delete User: Nurul");  
  destroy(2);  
};
```



TODO 1

```
/**  
 * TODO 1.  
 * Create array of object users (5 users).  
 * Object has property: name, age, major.  
 * Note: Use const instead var.  
 */  
var users;
```



TODO 2

```
/**  
 * TODO 2  
 * Create all function: Show all data users.  
 * Hint: use for/for-of.  
 * Note: Use arrow function and const.  
 */  
function all() {}
```



all - Get All Users

Get All Users

Nama: Aufa
Age: 22
Major: Informatics

Nama: Isfa
Age: 22
Major: Informatics

Nama: Nurul
Age: 2
Major: Information System



Main Function

```
const main = () => {  
  console.log("# Get All Users");  
  all();  
  
  console.log("# Add New User: Sabiq");  
  const newUser = {  
    name: "Sabiq",  
    age: 20,  
    major: "Informatics",  
  };  
  store(newUser);  
  
  console.log("# Edit User: Isfa");  
  const editedUser = {  
    name: "Isfhani Ghiyath",  
    age: 23,  
    major: "English",  
  };  
  update(1, editedUser);  
  
  console.log("# Delete User: Nurul");  
  destroy(2);  
};
```



TODO 3

```
/**  
 * TODO 3  
 * Create store function: Add data to users.  
 * Hint: use push method.  
 * Note: Use arrow function and const.  
 */  
function store(user) {}
```



create - Add New User

Add New User: Sabiq

Nama: Aufa
Age: 22
Major: Informatics

Nama: Isfa
Age: 22
Major: Informatics

Nama: Nurul
Age: 21
Major: Information System

Nama: Sabiq
Age: 20
Major: Informatics



Main Function

```
const main = () => {  
  console.log("# Get All Users");  
  all();  
  
  console.log("# Add New User: Sabiq");  
  const newUser = {  
    name: "Sabiq",  
    age: 20,  
    major: "Informatics",  
  };  
  store(newUser);  
  
  console.log("# Edit User: Isfa");  
  const editedUser = {  
    name: "Isfhani Ghiyath",  
    age: 23,  
    major: "English",  
  };  
  update(1, editedUser);  
  
  console.log("# Delete User: Nurul");  
  destroy(2);  
};
```



TODO 4

```
/**  
 * TODO 4.  
 * Create update function: Edit data user.  
 * Hint: just re-assign.  
 * Note: Use arrow function and const.  
 */  
function update(index, user) {}
```



update - Edit User

Edit User: Isfa

Nama: Aufa
Age: 22
Major: Informatics

Nama: Isfhani Ghiyath
Age: 23
Major: English

Nama: Nurul
Age: 21
Major: Information System

Nama: Sabiq
Age: 20
Major: Informatics



Main Function

```
const main = () => {  
  console.log("# Get All Users");  
  all();  
  
  console.log("# Add New User: Sabiq");  
  const newUser = {  
    name: "Sabiq",  
    age: 20,  
    major: "Informatics",  
  };  
  store(newUser);  
  
  console.log("# Edit User: Isfa");  
  const editedUser = {  
    name: "Isfhani Ghiyath",  
    age: 23,  
    major: "English",  
  };  
  update(1, editedUser);  
  
  console.log("# Delete User: Nurul");  
  destroy(2);  
};
```



TODO 5

```
/**  
 * TODO 5.  
 * Create destroy function: Delete data user.  
 * Hint: use splice method.  
 * Note: Use arrow function and const.  
 */  
function destroy(index) {}
```



destroy - Delete User

Delete User: Nurul

Nama: Aufa
Age: 22
Major: Informatics

Nama: Isfhani Ghiyath
Age: 23
Major: English

Nama: Sabiq
Age: 20
Major: Informatics

Attendance

—

Thanks

