

JavaScript



Apa yang Kita Pelajari?

Memulai JavaScript	Variabel	Output	Tipe Data	Operator
Fungsi	Event	String	Number	If dan Else
Switch ... Case	Array	Looping	Object	

Memulai JavaScript

Menggunakan Browser

- JavaScript adalah bahasa pemrograman yang berjalan di browser web.
- Kita dapat memulai dengan membuat file HTML (index.html) sederhana dan menyisipkan kode JavaScript di dalamnya.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh JavaScript</title>
  </head>
  <body>
    <h1>Halo, dunia!</h1>
    <script>
      // Kode JavaScript akan diletakkan di sini
      console.log("Halo dari JavaScript!");
    </script>
  </body>
</html>
```

External Script

- Kita juga dapat menggunakan file JavaScript eksternal yang dihubungkan dengan file HTML menggunakan elemen <script>.
- Cara ini membantu dalam pemisahan kode JavaScript dari markup HTML kita.

```
// script.js
console.log("Halo dari file JavaScript eksternal!");

// index.html
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh JavaScript</title>
  </head>
  <body>
    <h1>Halo, dunia!</h1>
    <script src="script.js"></script>
  </body>
</html>
```

Inline Script

- Menempatkan kode JavaScript secara inline artinya menempatkan kode JavaScript di dalam elemen HTML langsung.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh JavaScript Inline</title>
  </head>
  <body>
    <h1>Halo, dunia!</h1>
    <script>
      // Kode JavaScript Anda
      // akan diletakkan di sini
      var nama = "John";
      console.log("Halo, " + nama + "!");
    </script>
  </body>
</html>
```

Script di Head

- Kita juga dapat menempatkan elemen <script> di dalam elemen <head> HTML.
- Kode JavaScript di dalamnya akan dieksekusi sebelum konten utama halaman dimuat.
- Ini berarti kode JavaScript akan dieksekusi sebelum elemen-elemen di dalam elemen <body> dimuat.

Kapan Menempatkan Script di Head?

- Jika kita perlu mendefinisikan variabel atau fungsi yang akan digunakan secara global di seluruh halaman, Anda bisa menempatkannya di dalam elemen <head>.
- Jika kita perlu mengeksekusi kode JavaScript setelah halaman selesai dimuat, kita dapat menempatkannya di dalam elemen <head> dan menggunakan event seperti **onload** atau **DOMContentLoaded** untuk memastikan kode dieksekusi pada waktu yang tepat. Ini berguna jika Anda ingin memastikan seluruh struktur HTML telah dimuat sebelum mengeksekusi kode JavaScript tertentu.

Kapan Menempatkan Script di Body?

- Jika kode JavaScript kita berinteraksi langsung dengan elemen-elemen HTML dalam halaman, seperti mengubah konten, mengatur atribut, atau menangani interaksi pengguna, sebaiknya tempatkan kode tersebut di luar elemen <head> (biasanya di dalam elemen <body>).
- Menempatkan semua kode JavaScript di dalam elemen <head> **dapat menyebabkan waktu muat halaman yang lebih lama karena** JavaScript dieksekusi sebelum elemen-elemen di dalam elemen <body> dimuat.

Variabel

Variabel

- Variabel adalah tempat di dalam memori yang digunakan untuk menyimpan dan mereferensikan data dalam sebuah program komputer.
- Ketika kita membuat variabel, kita memberikan nama pada lokasi memori tertentu untuk menampung nilai tertentu, seperti angka, teks, atau tipe data lainnya.

Membuat Variabel

- Menggunakan **var** (versi lama, sebaiknya hindari penggunaan ini).
- Menggunakan **let** (dapat diubah nilainya).
- Menggunakan **const** (konstan, nilainya tidak bisa diubah setelah diberikan nilai awal).

```
var nama = "John";  
var umur = 30;
```

```
let nama = "John";  
let umur = 30;
```

```
const pi = 3.14;  
const nama = "John";
```

Aturan Penamaan Variabel

- Nama variabel harus dimulai dengan huruf, garis bawah (_), atau tanda dolar (\$).
- Nama variabel dapat mengandung huruf, angka, garis bawah (_), atau tanda dolar (\$).
- Nama variabel bersifat case-sensitive, artinya namaVariabel berbeda dengan namavariabel.

```
let nama = "John";
let umur = 30;

console.log("Nama: " + nama); // Output: Nama: John
console.log("Umur: " + umur); // Output: Umur: 30

// Mengubah nilai variabel
nama = "Jane";
console.log("Nama baru: " + nama); // Output: Nama baru: Jane

// Contoh variabel const
const pi = 3.14;
console.log("Nilai pi: " + pi); // Output: Nilai pi: 3.14

// Akan menyebabkan error karena
// variabel const tidak dapat diubah nilainya.
pi = 3.14159; // Error: "Assignment to constant variable."
```

Keyword let

- Variabel yang dideklarasikan dengan let bersifat block-scoped, artinya variabel tersebut hanya dapat diakses di dalam blok kode tempat variabel tersebut dideklarasikan (misalnya, di dalam fungsi atau pernyataan if/else).
- Variabel let dapat diinisialisasi ulang (nilai di dalamnya dapat diubah).
- Variabel let lebih disarankan daripada var karena memberikan perilaku cakupan yang lebih aman dan terprediksi.

```
function contohLet() {  
    let x = 5;  
    if (true) {  
        // Variabel x ini hanya berlaku di dalam blok if  
        let x = 10;  
        console.log(x); // Output: 10  
    }  
  
    // Output: 5 (Variabel x di dalam fungsi)  
    console.log(x);  
}  
  
contohLet();
```

Keyword const

- Variabel yang dideklarasikan dengan const juga bersifat block-scoped dan tidak dapat diinisialisasi ulang setelah diberi nilai awal. Nilai variabel const tidak dapat diubah setelahnya.
- Jika variabel const memiliki tipe data primitif (seperti angka, teks, atau boolean), nilainya tidak dapat diubah. Namun, jika variabel const menampung objek atau array, kita masih dapat mengubah properti atau elemennya, tetapi kita tidak dapat memberikan nilai baru secara langsung ke variabel itu sendiri.

```
function contohConst() {  
    const pi = 3.14;  
    // pi = 3.14159;  
    // Akan menyebabkan error,  
    // karena const tidak dapat diubah nilainya  
  
    const data = {  
        nama: "John",  
        usia: 30  
    };  
    data.nama = "Jane"; // Ini diperbolehkan karena  
                        // properti dalam objek bisa  
                        // diubah  
  
    // Output: { nama: 'Jane', usia: 30 }  
    console.log(data);  
}  
  
contohConst();
```

Keyword var

- Variabel yang dideklarasikan dengan var memiliki function scope atau global scope.
Variabel var dikenal oleh seluruh fungsi di mana variabel tersebut dideklarasikan, atau jika dideklarasikan di luar semua fungsi, maka akan menjadi variabel global yang dapat diakses dari mana saja di dalam program.
- Variabel var dapat diinisialisasi ulang dan nilainya dapat diubah di dalam blok kode apa pun.

```
function contohVar() {  
    var a = 5;  
    if (true) {  
        var a = 10; // Variabel a ini berlaku di seluruh fungsi  
        console.log(a); // Output: 10  
    }  
    console.log(a); // Output: 10 (Nilai a telah diubah di dalam blok if)  
}  
  
contohVar();  
  
var nama = "John";  
console.log(nama); // Output: John
```

Tips Menggunakan Variabel

- Gunakan **const** jika kita tahu nilainya tidak akan berubah.
- Gunakan **let** jika kita membutuhkan variabel yang nilainya dapat diubah.
- **Hindari penggunaan `var`** kecuali jika kita bekerja dengan kode lama yang membutuhkannya, sebaiknya gunakan `let` untuk variabel yang perlu cakupan (scope) fungsi atau blok.

Output

Menulis Output ke Console Browser

- Untuk menulis output ke console browser, kita dapat menggunakan fungsi **console.log()**.
- Fungsi ini memungkinkan kita menampilkan pesan, variabel atau data lainnya di console browser saat kita sedang mengembangkan aplikasi web.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh Output ke Konsol Browser</title>
  </head>
  <body>
    <h1>Halo, dunia!</h1>
    <script>
      // Menulis output ke konsol browser
      console.log("Ini adalah pesan untuk konsol browser.");
      // Contoh output variabel
      let nama = "John";
      let usia = 30;
      console.log("Nama: " + nama + ", Usia: " + usia);
      // Contoh output data lainnya
      let arr = [1, 2, 3, 4, 5];
      console.log("Array: ", arr);
      let obj = {
        nama: "Jane",
        pekerjaan: "Developer"
      };
      console.log("Objek: ", obj);
    </script>
  </body>
</html>
```

Menampilkan Output di Alert

- Untuk menampilkan output dalam bentuk pop-up alert di browser, kita dapat menggunakan fungsi **alert()**.
- Fungsi ini akan menampilkan pesan atau nilai variabel dalam bentuk jendela pop-up yang harus dikonfirmasi oleh pengguna sebelum melanjutkan berinteraksi dengan halaman web.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Contoh Menampilkan Output dalam Alert</title>
  </head>
  <body>
    <h1>Halo, dunia!</h1>
    <script>
      // Menampilkan output dalam pop-up alert
      alert("Ini adalah pesan dalam pop-up alert.");
      // Contoh output variabel
      let nama = "John";
      let usia = 30;
      alert("Nama: " + nama + ", Usia: " + usia);
      // Contoh output data lainnya
      let arr = [1, 2, 3, 4, 5];
      alert("Array: " + arr);
      let obj = {
        nama: "Jane",
        pekerjaan: "Developer"
      };
      alert("Objek: " + JSON.stringify(obj));
    </script>
  </body>
</html>
```

Menulis Output ke Jendela Browser

- Anda dapat menulis output ke jendela browser menggunakan metode `document.write()`.
- Metode ini memungkinkan Anda menambahkan teks atau elemen HTML langsung ke dalam dokumen saat halaman web sedang dimuat.
- Namun, perlu diingat bahwa `document.write()` memiliki beberapa kelemahan, terutama jika dipanggil setelah halaman selesai dimuat, karena ia akan mengganti seluruh konten halaman dengan teks yang baru ditambahkan.

```
// Printing a simple text message
document.write("Hello World!"); // Prints: Hello World!

// Printing a variable value
var x = 10;
var y = 20;
var sum = x + y;
document.write(sum); // Prints: 30
```

Memasukkan Output Di Dalam Elemen HTML

- Kita dapat menggunakan properti innerHTML dari elemen HTML untuk memasukkan output atau konten HTML ke dalam elemen tersebut.
- Properti innerHTML memungkinkan kita menetapkan teks, tag HTML, atau kombinasi keduanya ke dalam elemen dengan cara yang lebih fleksibel.

```
<p id="greet"></p>
<p id="result"></p>

<script>
// Writing text string inside an element
document.getElementById("greet").innerHTML = "Hello World!";

// Writing a variable value inside an element
var x = 10;
var y = 20;
var sum = x + y;
document.getElementById("result").innerHTML = sum;
</script>
```

Tipe Data

Tipe Data di JavaScript

- Di JavaScript, ada beberapa tipe data yang digunakan untuk merepresentasikan berbagai jenis nilai.
- Tipe data ini membantu JavaScript memahami dan memanipulasi nilai dengan cara yang sesuai.

Tipe Data String

- Tipe data string digunakan untuk merepresentasikan teks atau karakter.
- String diapit oleh tanda kutip (petik satu atau petik dua).

```
let a = 'Hi there!'; // using single quotes
let b = "Hi there!"; // using double quotes
```

Tipe Data Number

- Tipe data number digunakan untuk merepresentasikan angka, baik bilangan bulat, decimal maupun eksponensial.

```
var a = 25;          // integer
var b = 80.5;        // floating-point number
var c = 4.25e+6;    // exponential notation, same as 4.25e6 or
4250000
var d = 4.25e-6;    // exponential notation, same as 0.00000425
```

Infinity

- Infinity adalah nilai yang merepresentasikan bilangan tak terhingga.
- Nilai ini digunakan untuk mewakili hasil dari operasi matematika yang melibatkan bilangan yang sangat besar atau tak terhingga.

```
let angkaBesar = 1e309; // Ini merupakan angka yang sangat besar
console.log(angkaBesar); // Output: Infinity

let hasilBagi = 10 / 0; // Hasil bagi angka dengan 0
console.log(hasilBagi); // Output: Infinity
```

NaN

- NaN adalah singkatan dari "Not a Number".
- Ini merupakan nilai khusus yang digunakan untuk mewakili hasil operasi matematika yang tidak dapat diidentifikasi atau tidak dapat diwakili sebagai angka.

```
let hasilOperasi = 10 / "abc"; // Operasi pembagian angka & teks  
console.log(hasilOperasi); // Output: NaN  
  
let angka = parseInt("abc"); // Mengkonversi teks menjadi angka  
console.log(angka); // Output: NaN
```

NaN

- Ketika NaN digunakan dalam operasi matematika lainnya, hasilnya tetap NaN.
- NaN adalah nilai khusus yang berbeda dari angka lainnya, termasuk dari dirinya sendiri.
- Ini berarti, jika kita membandingkan dua nilai NaN, hasilnya akan selalu false.

```
console.log(NaN === NaN); // Output: false
```

Tipe Data Boolean

- Tipe data boolean dalam JavaScript hanya memiliki dua nilai: true dan false.
- Tipe data ini digunakan untuk merepresentasikan nilai kebenaran atau logika dalam sebuah pernyataan.

```
let benar = true;
let salah = false;

console.log(benar); // Output: true
console.log(salah); // Output: false
```

Tipe Data Boolean

- Tipe data boolean juga dapat dihasilkan dari operasi perbandingan atau evaluasi ekspresi.

```
let x = 5;
let y = 10;

let hasil1 = x > y; // Evaluasi ekspresi, false
let hasil2 = x <= y; // Evaluasi ekspresi, true

console.log(hasil1); // Output: false
console.log(hasil2); // Output: true
```

Tipe Data Undefined

- Saat sebuah variabel dideklarasikan, tetapi tidak diinisialisasi dengan nilai tertentu, nilai dari variabel tersebut secara otomatis menjadi **undefined**.

```
let umur; // Variabel dideklarasikan tapi belum diinisialisasi
console.log(umur); // Output: undefined
console.log(typeof umur); // Output: "undefined" (tipe data string)
```

Tipe Data Undefined

- Tipe data **undefined** juga muncul ketika sebuah fungsi tidak mengembalikan nilai secara eksplisit.

```
function fungsiTanpaReturn() {  
    // Fungsi ini tidak mengembalikan nilai secara eksplisit  
}  
  
let hasilFungsi = fungsiTanpaReturn();  
console.log(hasilFungsi); // Output: undefined
```

Tipe Data Null

- Tipe data **null** dalam JavaScript digunakan untuk menyatakan bahwa sebuah variabel tidak memiliki nilai atau tidak memiliki nilai yang valid.
- Ketika sebuah variabel diberi nilai **null**, ini berarti variabel tersebut disengaja diberi nilai kosong atau tidak ada nilai yang relevan untuk diberikan.

```
var a = null;  
alert(a); // Output: null
```

```
var b = "Hello World!"  
alert(b); // Output: Hello World!
```

```
b = null;  
alert(b) // Output: null
```

Tipe Data Object

- Tipe data Object dalam JavaScript adalah tipe data yang kompleks, digunakan untuk menyimpan dan mengelompokkan nilai-nilai dalam bentuk pasangan key-value (kunci-nilai).

```
let emptyObject = {};
let person = {"name": "Clark",
"surname": "Kent", "age": "36"};

// For better reading
let car = {
  "modal": "BMW X3",
  "color": "white",
  "doors": 5
}
```

Tipe Data Array

- Tipe data Array dalam JavaScript digunakan untuk menyimpan koleksi nilai dalam satu variabel.
- Array adalah tipe data yang sederhana namun sangat kuat, karena dapat menyimpan nilai-nilai dari berbagai tipe data (seperti angka, teks, objek, dan sebagainya) dalam satu struktur data yang terindeks.

```
var colors = ["Red", "Yellow", "Green",
"Orange"];
var cities = ["London", "Paris", "New York"];

alert(colors[0]); // Output: Red
alert(cities[2]); // Output: New York
```

Tipe Data Function

- Tipe data Function dalam JavaScript adalah tipe data yang digunakan untuk menyimpan fungsi sebagai nilai.

```
var greeting = function(){
    return "Hello World!";
}

// Check the type of greeting variable
alert(typeof greeting) // Output: function
alert(greeting());      // Output: Hello World!
```

Tipe Data Function

- Fungsi dalam JavaScript adalah nilai yang sangat fleksibel dan dapat digunakan di mana saja, termasuk sebagai argumen dalam fungsi lain.

```
function createGreeting(name){  
    return "Hello, " + name;  
}  
  
function displayGreeting(greetingFunction, userName){  
    return greetingFunction(userName);  
}  
  
var result = displayGreeting(createGreeting, "Peter");  
alert(result); // Output: Hello, Peter
```

Operator Typeof

- Operator typeof dalam JavaScript digunakan untuk mengambil tipe data dari sebuah nilai atau variabel.
- Operator ini mengembalikan sebuah string yang berisi nama tipe data dari nilai yang diberikan.

```
let angka = 10;
let teks = "Halo";
let array = [1, 2, 3];
let objek = {nama: "John", usia: 30};
let fungsi = function() {
    console.log("Ini adalah fungsi");
};

console.log(typeof angka); // Output: "number"
console.log(typeof teks); // Output: "string"
console.log(typeof array); // Output: "object"
console.log(typeof objek); // Output: "object"
console.log(typeof fungsi); // Output: "function"
console.log(typeof undefinedVariable); // Output: "undefined"
console.log(typeof nullValue); // Output: "object"
```

Operator

Operator

- Operator dalam pemrograman adalah simbol atau tanda khusus yang digunakan untuk melakukan operasi pada satu atau beberapa nilai (operand).
- Operator dapat digunakan untuk melakukan berbagai jenis operasi matematika, perbandingan, logika dan manipulasi data.

Operator Aritmatika

- Dalam JavaScript, terdapat beberapa operator aritmatika yang digunakan untuk melakukan operasi matematika pada operand numerik.

```
var x = 10;  
var y = 4;  
alert(x + y); // Outputs: 14  
alert(x - y); // Outputs: 6  
alert(x * y); // Outputs: 40  
alert(x / y); // Outputs: 2.5  
alert(x % y); // Outputs: 2
```

Operator Penugasan

- Operator penugasan dalam JavaScript digunakan untuk mengisi atau menetapkan nilai ke sebuah variabel.

```
var x;      // Declaring Variable  
  
x = 10;  
alert(x); // Outputs: 10  
  
x = 20;  
x += 30;  
alert(x); // Outputs: 50  
  
x = 50;  
x -= 20;  
alert(x); // Outputs: 30  
  
x = 5;  
x *= 25;  
alert(x); // Outputs: 125  
  
x = 50;  
x /= 10;  
alert(x); // Outputs: 5  
  
x = 100;  
x %= 15;  
alert(x); // Outputs: 10
```

Operator String

- Dalam JavaScript, ada beberapa operator yang digunakan untuk melakukan operasi pada tipe data string.
- Operator-operator ini memungkinkan kita untuk menggabungkan (concatenate) string.

```
var str1 = "Hello";
var str2 = " World!";

alert(str1 + str2); // Outputs: Hello World!

str1 += str2;
alert(str1); // Outputs: Hello World!
```

Operator Increment dan Decrement

- Operator Increment dan Decrement digunakan untuk meningkatkan (increment) atau mengurangi (decrement) nilai variabel angka dengan 1.

```
var x; // Declaring Variable  
  
x = 10;  
alert(++x); // Outputs: 11  
alert(x); // Outputs: 11  
  
x = 10;  
alert(x++); // Outputs: 10  
alert(x); // Outputs: 11  
  
x = 10;  
alert(--x); // Outputs: 9  
alert(x); // Outputs: 9  
  
x = 10;  
alert(x--); // Outputs: 10  
alert(x); // Outputs: 9
```

Operator Logika

- Operator logika dalam JavaScript digunakan untuk menggabungkan atau memanipulasi nilai boolean (true atau false).

```
var year = 2018;

// Leap years are divisible by 400 or by 4 but not 100
if((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0))){
    alert(year + " is a leap year.");
} else{
    alert(year + " is not a leap year.");
}
```

Operator	Keterangan	Contoh Penggunaan	Hasil
&&	AND Logika	'true && false'	false
	OR Logika	'true false'	true
!	NOT Logika	'!true'	false

Operator Perbandingan

- Operator perbandingan digunakan untuk membandingkan dua nilai dan menghasilkan nilai boolean (true atau false) berdasarkan hasil perbandingan.

```
var x = 25;  
var y = 35;  
var z = "25";  
  
alert(x == z); // Outputs: true  
alert(x === z); // Outputs: false  
alert(x != y); // Outputs: true  
alert(x !== z); // Outputs: true  
alert(x < y); // Outputs: true  
alert(x > y); // Outputs: false  
alert(x <= y); // Outputs: true  
alert(x >= y); // Outputs: false
```

Fungsi

Fungsi

- Dalam JavaScript, fungsi adalah blok kode yang dapat didefinisikan sekali dan digunakan berulang kali untuk melakukan tugas tertentu atau menghitung nilai.
- Fungsi merupakan salah satu konsep dasar dalam pemrograman, dan ia memungkinkan kita untuk mengorganisasi kode menjadi bagian-bagian yang terpisah dan dapat digunakan kembali.

Mendefinisikan Fungsi

- Gunakan kata kunci function diikuti dengan nama fungsi dan parameter (jika ada) di dalam tanda kurung.
- Selanjutnya, definisikan tubuh fungsi yang berisi kode yang ingin kita eksekusi ketika fungsi dipanggil.

```
function sapa() {  
    console.log("Halo, selamat datang!");  
}
```

Memanggil Fungsi

- Untuk memanggil fungsi,
cukup tuliskan nama fungsi
diikuti dengan tanda kurung ()..

```
function sapa() {  
    console.log("Halo, selamat datang!");  
}  
  
sapa(); // Output: "Halo, selamat datang!"
```

Menambahkan Parameter

- Kita dapat menambahkan parameter ke dalam definisi fungsi untuk menerima input atau argumen yang akan digunakan dalam tubuh fungsi.

```
// Defining function
function displaySum(num1, num2) {
  var total = num1 + num2;
  alert(total);
}

// Calling function
displaySum(6, 20); // Outputs: 26
displaySum(-5, 17); // Outputs: 12
```

Menambahkan Parameter

- Kita harus memastikan bahwa saat memanggil fungsi, kita memberikan argumen yang sesuai dengan jumlah dan urutan parameter yang didefinisikan.
- Dalam kasus ini, parameter **lastName** tidak menerima nilai apa pun saat fungsi dipanggil, sehingga nilainya menjadi tidak terdefinisi (**undefined**).

```
// Defining function
function showFullname(firstName, lastName) {
  alert(firstName + " " + lastName);
}

// Calling function
showFullname("Clark", "Kent"); // Outputs: Clark Kent
showFullname("John"); // Outputs: John undefined
```

Nilai Default untuk Parameter

- Kita dapat memberikan nilai default untuk parameter fungsi menggunakan sintaksis default parameter yang diperkenalkan dalam ECMAScript 6 (ES6).

```
function sayHello(name = 'Guest') {  
  alert('Hello, ' + name);  
}  
  
sayHello(); // Outputs: Hello, Guest  
sayHello('John'); // Outputs: Hello, John
```

Mengembalikan Nilai

- Dalam JavaScript, fungsi dapat mengembalikan nilai (return value) menggunakan statement **return**.

```
// Defining function
function getSum(num1, num2) {
    var total = num1 + num2;
    return total;
}

// Displaying returned value
alert(getSum(6, 20)); // Outputs: 26
alert(getSum(-5, 17)); // Outputs: 12
```

Mengembalikan Nilai

- Di JavaScript, fungsi secara alami hanya dapat mengembalikan satu nilai.
- Namun, kita dapat mengembalikan banyak nilai dalam bentuk array atau objek.

```
// Defining function
function divideNumbers(dividend, divisor){
    var quotient = dividend / divisor;
    var arr = [dividend, divisor, quotient];
    return arr;
}

// Store returned value in a variable
var all = divideNumbers(10, 2);

// Displaying individual values
alert(all[0]); // Outputs: 10
alert(all[1]); // Outputs: 2
alert(all[2]); // Outputs: 5
```

Ekspresi Fungsi

- Di dalam ekspresi fungsi, fungsi didefinisikan sebagai bagian dari sebuah ekspresi dan disimpan dalam variabel.

```
var getSum = function(num1, num2) {  
    var total = num1 + num2;  
    return total;  
};  
  
alert(getSum(5, 10)); // Outputs: 15  
  
var sum = getSum(7, 25);  
alert(sum); // Outputs: 32
```

Ekspresi Fungsi

- Sintaks **deklarasi fungsi** dan **ekspresi fungsi** terlihat sangat mirip, tetapi keduanya berbeda dalam cara mengevaluasinya.

```
declaration(); // Outputs: Hi, I'm a function declaration!
function declaration() {
    alert("Hi, I'm a function declaration!");
}

expression(); // Uncaught TypeError: undefined is not a function
var expression = function() {
    alert("Hi, I'm a function expression!");
};
```

Ruang Lingkup Variabel di Fungsi

- Ruang lingkup variabel di fungsi, juga dikenal sebagai "scope", mengacu pada bagaimana akses dan keterlihatan variabel yang dideklarasikan di dalam fungsi.

```
// Ruang Lingkup Global
// -----
let globalVar = "Ini variabel global";

function fungsiSatu() {
  console.log(globalVar); // Dapat mengakses variabel global
}

fungsiSatu(); // Output: "Ini variabel global"
console.log(globalVar); // Output: "Ini variabel global"

// Ruang Lingkup Lokal
// -----
function fungsiDua() {
  let lokalVar = "Ini variabel lokal";
  console.log(lokalVar); // Dapat mengakses variabel lokal
}

fungsiDua(); // Output: "Ini variabel lokal"
console.log(lokalVar); // Error: lokalVar is not defined
```

Event

Event

- JavaScript Event adalah tindakan atau peristiwa yang terjadi di dalam halaman web saat pengguna berinteraksi dengan elemen-elemen di dalamnya.
- Interaksi tersebut bisa berupa mengklik sebuah tombol, menggerakkan mouse, mengetik pada keyboard atau bahkan mengubah ukuran jendela browser.
- Saat peristiwa ini terjadi, JavaScript dapat "mendengarkan" dan meresponsnya dengan menjalankan kode tertentu.

Event Click

- Event ini terjadi ketika pengguna mengklik (klik kiri) elemen tertentu di halaman web.
- Ketika event "click" terjadi, JavaScript dapat meresponsnya dengan menjalankan kode tertentu yang telah ditentukan.

```
<button id="tombolKlik">Klik Saya</button>

<script>
    // Mendapatkan referensi elemen tombol dengan ID "tombolKlik"
    const tombolKlik = document.getElementById('tombolKlik');

    // Menambahkan event listener untuk event "click"
    tombolKlik.addEventListener('click', function() {
        // Kode yang akan dijalankan saat tombol diklik
        console.log('Tombol telah diklik!');
    });
</script>
```

Event Contextmenu

- Event "contextmenu" adalah salah satu event dalam JavaScript yang terjadi ketika pengguna mengklik tombol kanan mouse pada elemen tertentu di halaman web.

```
<div id="elemenKlikKanan">Klik Kanan Saya</div>

<script>
    // Mendapatkan referensi elemen dengan ID "elemenKlikKanan"
    const elemenKlikKanan = document.getElementById('elemenKlikKanan');

    // Menambahkan event listener untuk event "contextmenu"
    elemenKlikKanan.addEventListener('contextmenu', function(event) {
        // Mencegah menu konteks bawaan muncul
        event.preventDefault();

        // Menampilkan pesan ketika tombol kanan mouse diklik
        console.log('Tombol kanan mouse telah diklik pada elemen!');
    });

</script>
```

Event Mouseover

- Event "mouseover" adalah salah satu event dalam JavaScript yang terjadi ketika kurSOR mouse bergerak masuk ke dalam batas area suatu elemen di halaman web.

```
<div id="elemenMouseover">Arahkan mouse ke sini</div>

<script>
    // Mendapatkan referensi elemen dengan ID "elemenMouseover"
    const elemenMouseover = document.getElementById('elemenMouseover');

    // Menambahkan event listener untuk event "mouseover"
    elemenMouseover.addEventListener('mouseover', function() {
        // Kode yang akan dijalankan saat kurSOR
        // mouse masuk ke dalam elemen
        console.log('Kursor mouse masuk ke dalam elemen!');
    });
</script>
```

Event Keydown

- Event ini terjadi ketika pengguna menekan tombol pada keyboard.
- Event "keydown" mendeteksi semua tombol keyboard, termasuk tombol kontrol (seperti tombol panah, tombol fungsi dan lain-lain), tombol aksi (seperti Enter, Esc dan lain-lain) dan karakter biasa (huruf, angka dan simbol).

```
<input type="text" id="inputTeks" placeholder="Ketik di sini">

<script>
  // Mendapatkan referensi elemen input teks dengan ID "inputTeks"
  const inputTeks = document.getElementById('inputTeks');

  // Menambahkan event listener untuk event "keydown"
  inputTeks.addEventListener('keydown', function(event) {
    // Mendapatkan nilai tombol yang ditekan (berdasarkan kode tombol)
    const tombolDitekan = event.key;

    // Kode yang akan dijalankan saat tombol pada keyboard ditekan
    console.log('Tombol yang ditekan: ' + tombolDitekan);
  });

</script>
```

Event Keyup

- Event keyup terjadi ketika pengguna melepaskan tombol pada keyboard.

```
<input type="text" id="inputTeks" placeholder="Ketik di sini">

<script>
    // Mendapatkan referensi elemen input teks dengan ID "inputTeks"
    const inputTeks = document.getElementById('inputTeks');

    // Menambahkan event listener untuk event "keyup"
    inputTeks.addEventListener('keyup', function(event) {
        // Mendapatkan nilai teks yang dimasukkan setelah tombol dilepaskan
        const nilaiTeks = event.target.value;

        // Kode yang akan dijalankan saat tombol pada keyboard dilepaskan
        console.log('Teks yang dimasukkan: ' + nilaiTeks);
    });
</script>
```

Event Keypress

- Event ini terjadi ketika pengguna menekan tombol pada keyboard ketika suatu elemen tertentu sedang dalam fokus (dalam keadaan aktif).
- Event "keypress" hanya mendeteksi karakter yang dapat ditampilkan (huruf, angka dan simbol) dan tidak akan mendeteksi tombol kontrol atau tombol aksi.

```
<input type="text" id="inputTeks" placeholder="Ketik di sini">

<script>
    // Mendapatkan referensi elemen input teks dengan ID "inputTeks"
    const inputTeks = document.getElementById('inputTeks');

    // Menambahkan event listener untuk event "keypress"
    inputTeks.addEventListener('keypress', function(event) {
        // Mendapatkan karakter yang ditekan
        const karakter = event.key;

        // Kode yang akan dijalankan saat karakter ditekan
        console.log('Karakter ditekan: ' + karakter);
    });
</script>
```

Event Focus

- Event "focus" terjadi ketika suatu elemen di halaman web mendapatkan fokus.
- Fokus berarti elemen tersebut menjadi "aktif" atau "terpilih".

```
<input type="text" id="nama" placeholder="Masukkan nama Anda">

<script>
    // Mendapatkan referensi elemen input teks dengan ID "nama"
    const inputNama = document.getElementById('nama');

    // Menambahkan event listener untuk event "focus"
    inputNama.addEventListener('focus', function() {
        // Kode yang akan dijalankan saat elemen mendapatkan fokus
        console.log('Elemen input mendapatkan fokus!');
    });
</script>
```

Event Blur

- Event "blur" adalah salah satu event dalam JavaScript yang terjadi ketika suatu elemen kehilangan fokus (tidak lagi aktif).

```
<input type="text" id="nama" placeholder="Masukkan nama Anda">

<script>
    // Mendapatkan referensi elemen input teks dengan ID "nama"
    const inputNama = document.getElementById('nama');

    // Menambahkan event listener untuk event "blur"
    inputNama.addEventListener('blur', function() {
        // Kode yang akan dijalankan saat elemen kehilangan fokus
        console.log('Elemen input kehilangan fokus!');
    });
</script>
```

Event Change

- Event "change" terjadi ketika nilai atau isi dari suatu elemen input berubah dan kehilangan fokus.

```
<input type="text" id="nama" placeholder="Masukkan nama Anda">

<script>
    // Mendapatkan referensi elemen input teks dengan ID "nama"
    const inputNama = document.getElementById('nama');

    // Menambahkan event listener untuk event "change"
    inputNama.addEventListener('change', function() {
        // Kode yang akan dijalankan saat nilai elemen berubah
        // dan kehilangan fokus
        console.log('Nilai input telah berubah: ' + inputNama.value);
    });
</script>
```

Event Change (Checkbox)

- saat pengguna mencentang atau melepaskan centang pada checkbox dengan ID "notif" dan kemudian mengalihkan fokus, maka akan muncul pesan sesuai dengan status checkbox, apakah notifikasi diaktifkan atau dinonaktifkan.

```
<input type="checkbox" id="notif" name="notif">
<label for="notif">Aktifkan Notifikasi</label>

<script>
    // Mendapatkan referensi elemen checkbox dengan ID "notif"
    const checkboxNotif = document.getElementById('notif');

    // Menambahkan event listener untuk event "change"
    checkboxNotif.addEventListener('change', function() {
        // Kode yang akan dijalankan saat
        // nilai checkbox berubah dan kehilangan fokus
        if (checkboxNotif.checked) {
            console.log('Notifikasi diaktifkan!');
        } else {
            console.log('Notifikasi dinonaktifkan!');
        }
    });
</script>
```

Event Submit

- Event "submit" terjadi ketika pengguna mengirimkan sebuah form melalui tombol "Submit".
- Event "submit" biasanya digunakan pada elemen <form> untuk menangani aksi yang terjadi saat pengguna mengirimkan data form ke server.

```
<form id="formRegistrasi">
  <label for="nama">Nama:</label>
  <input type="text" id="nama" name="nama" required>
  <br>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>
  <br>
  <button type="submit">Daftar</button>
</form>

<script>
  // Mendapatkan referensi elemen formulir dengan ID "formRegistrasi"
  const formRegistrasi = document.getElementById('formRegistrasi');

  // Menambahkan event listener untuk event "submit"
  formRegistrasi.addEventListener('submit', function(event) {
    // Mencegah perilaku bawaan pengiriman formulir (refresh halaman)
    event.preventDefault();

    // Mendapatkan data dari formulir
    const formData = new FormData(formRegistrasi);
    const data = {};
    formData.forEach((value, key) => {
      data[key] = value;
    });

    // Kirim data ke server atau lakukan aksi lain sesuai kebutuhan
    console.log('Data yang dikirimkan:', data);
  });
</script>
```

Event Load

- Event "load" adalah salah satu event dalam JavaScript yang terjadi ketika suatu objek atau elemen di halaman web telah selesai dimuat atau di-load sepenuhnya.

Event Load

```
<!DOCTYPE html>
<html>
<head>
    <title>Contoh Event Load</title>
</head>
<body>
    <h1>Selamat datang di halaman web!</h1>
    <script src="script.js"></script>
</body>
</html>
```

```
// Mendapatkan referensi elemen body
const body = document.body;

// Menambahkan event listener untuk event "load"
body.addEventListener('load', function() {
    // Kode yang akan dijalankan saat halaman selesai dimuat
    console.log('Halaman web telah selesai dimuat.');
});
```

Event Resize

- Event "resize" terjadi ketika ukuran jendela browser atau elemen diubah oleh pengguna.

```
<p id="result"></p>

<script>
    function displayWindowSize() {
        var w = window.outerWidth;
        var h = window.outerHeight;
        var txt = "Window size: width=" + w + ", height=" + h;
        document.getElementById("result").innerHTML = txt;
    }
    window.onresize = displayWindowSize;
</script>
```

String

String

- Dalam JavaScript, sebuah string adalah urutan dari nol atau lebih karakter, seperti huruf, angka, simbol atau spasi yang diapit oleh tanda kutip (tanda petik tunggal atau tanda petik ganda).

```
const nama = "John"; // String dengan tanda kutip ganda
const alamat = 'Jl. Raya No. 123'; // String dengan tanda kutip tunggal
const pesan = "Halo, apa kabar?"; // String dengan tanda kutip ganda
const kataKunci = 'JavaScript'; // String dengan tanda kutip tunggal
```

Escape Sequence

- Escape sequence adalah serangkaian karakter khusus yang digunakan dalam string untuk merepresentasikan karakter-karakter tertentu yang sulit atau tidak mungkin digunakan secara langsung dalam teks.
- Escape sequence biasanya dimulai dengan karakter backslash (\) diikuti oleh karakter khusus atau kode.

```
/**  
\' mewakili tanda petik tunggal (single quote)  
di dalam string dengan tanda kutip tunggal.  
*/  
const teks1 = 'It\'s a beautiful day.'; // Output: It's a beautiful day.  
  
/**  
\" mewakili tanda petik ganda (double quote) di dalam  
string dengan tanda kutip ganda.  
*/  
const teks2 = "He said, \"Hello!\\""; // Output: He said, "Hello!"  
  
/**  
\\ mewakili karakter backslash itu sendiri.  
*/  
const teks3 = "C:\\Windows\\\\System32"; // Output: C:\Windows\System32  
  
/**  
\n mewakili karakter baris baru (newline)  
untuk membuat teks berpindah ke baris berikutnya.  
*/  
  
const teks4 = "Baris pertama.\nBaris kedua.";  
// Output:  
// Baris pertama.  
// Baris kedua.  
  
/**  
\t mewakili karakter tab untuk memberi jarak antar teks.  
*/  
const teks5 = "Nama:\tJohn\tUsia:\t30";  
// Output: Nama: John Usia: 30
```

Mendapatkan Jumlah Karakter

- Untuk mendapatkan jumlah karakter dalam sebuah string, kita dapat menggunakan properti `length` yang ada pada objek string di JavaScript.
- Properti **length** akan mengembalikan angka yang menunjukkan jumlah karakter dalam string tersebut.

```
const teks = "Halo, dunia!";
const jumlahKarakter = teks.length;
console.log(jumlahKarakter); // Output: 12
```

Method indexOf()

- Method **indexOf()** digunakan untuk mencari indeks pertama dari kemunculan sebuah substring dalam string.
- Jika substring tidak ditemukan, method ini akan mengembalikan nilai.

```
const kalimat = "Hari ini adalah hari yang cerah.";
const kataCari = "hari";

const posisi = kalimat.indexOf(kataCari);
console.log(posisi); // Output: 0
```

Method lastIndexOf()

- Method **lastIndexOf()** digunakan untuk mencari indeks dari kemunculan terakhir suatu substring dalam string utama.

```
const kalimat = "Saya suka JavaScript, JavaScript itu menyenangkan.";
const kataCari = "JavaScript";

const posisiTerakhir = kalimat.lastIndexOf(kataCari);
console.log(posisiTerakhir); // Output: 26
```

Method search()

- Method **search()** digunakan untuk mencari indeks dari pertama kali kemunculan suatu pola (pattern) dalam string.
- Method **search()** dapat menerima ekspresi reguler sebagai pola pencarian.

```
var str = "Color red looks brighter than color blue.";  
  
// Case sensitive search  
var pos1 = str.search("color");  
alert(pos1); // Outputs: 30  
  
// Case insensitive search using regexp  
var pos2 = str.search(/color/i);  
alert(pos2); // Outputs: 0
```

Method slice()

- Method slice() adalah salah satu method pada objek string di JavaScript yang digunakan untuk mengambil potongan (substring) dari string utama berdasarkan indeks awal dan akhir tertentu.
- Method ini tidak mengubah string asli, tetapi mengembalikan potongan baru sebagai string baru.

```
const teks = "Halo, dunia!";

// Mengambil potongan dari indeks 5 hingga akhir string
const potongan1 = teks.slice(5);
console.log(potongan1); // Output: "dunia!"

// Mengambil potongan dari indeks 0 hingga 4
// (indeks 4 tidak ikut termasuk)
const potongan2 = teks.slice(0, 4);
console.log(potongan2); // Output: "Halo"

// Mengambil potongan dari indeks -5
// (dimulai dari akhir string) hingga akhir string
const potongan3 = teks.slice(-5);
console.log(potongan3); // Output: "dunia!"
```

Method replace()

- Method replace() digunakan untuk mengganti atau mengubah teks dalam string dengan teks baru atau ekspresi reguler.

```
const teks = "Halo, dunia!";
const teksBaru = teks.replace("dunia", "teman");

console.log(teksBaru); // Output: "Halo, teman!"
```

Method replace()

- Method replace() hanya mengganti kemunculan pertama dari substring yang dicari.
- Jika kita ingin mengganti semua kemunculan, kita dapat menggunakan ekspresi reguler dengan menggunakan opsi global (g).

```
const teks = "JavaScript, JavaScript, JavaScript!";
const pola = /JavaScript/g;
const teksBaru = teks.replace(pola, "JS");

console.log(teksBaru); // Output: "JS, JS, JS!"
```

Method replace()

- Kita juga dapat menggunakan fungsi sebagai argumen kedua untuk melakukan penggantian dengan lebih kompleks.

```
const teks = "Halo, dunia!";
const teksBaru = teks.replace("dunia", function(match)
{
    return match.toUpperCase();
});

console.log(teksBaru); // Output: "Halo, DUNIA!"
```

Method toLowerCase()

- Method toLowerCase()
digunakan untuk mengubah
semua karakter dalam string
menjadi huruf kecil.

```
const teks = "Halo, DUNIA!";
const teksKecil = teks.toLowerCase();

console.log(teksKecil); // Output: "halo, dunia!"
```

Method toUpperCase()

- Method toUpperCase()
digunakan untuk mengubah
semua karakter dalam string
menjadi huruf kapital (besar).

```
const teks = "Halo, dunia!";
const teksKapital = teks.toUpperCase();

console.log(teksKapital); // Output: "HALO, DUNIA!"
```

Concate String

- Kita dapat menggunakan operator + untuk menggabungkan dua atau lebih string menjadi satu.

```
const kata1 = "Halo, ";
const kata2 = "dunia!";
const gabungan = kata1 + kata2;

console.log(gabungan); // Output: "Halo, dunia!"
```

Concate String

- Kita dapat menggunakan method **concat()** untuk menggabungkan dua atau lebih string dengan cara yang sama seperti operator +.

```
const kata1 = "Halo, ";
const kata2 = "apa ";
const kata3 = "kabar?";
const gabungan = kata1.concat(kata2, kata3);

console.log(gabungan); // Output: "Halo, apa kabar?"
```

Method charAt()

- Method charAt() adalah salah satu metode pada objek string di JavaScript yang digunakan untuk mengambil karakter pada indeks tertentu dalam string.
- Method ini mengembalikan karakter dalam bentuk string, tepat pada indeks yang ditentukan.

```
const teks = "Hello, World!";
const karakterPertama = teks.charAt(0);
const karakterKelima = teks.charAt(4);

console.log(karakterPertama); // Output: "H"
console.log(karakterKelima); // Output: "o"
```

Method split()

- Metode split() adalah metode bawaan JavaScript yang digunakan untuk membagi sebuah string menjadi array substring berdasarkan pemisah yang ditentukan.

```
var fruitsStr = "Apple, Banana, Mango, Orange, Papaya";
var fruitsArr = fruitsStr.split(", ");

document.write(fruitsArr[0]); // Prints: Apple
document.write(fruitsArr[2]); // Prints: Mango
document.write(fruitsArr[fruitsArr.length - 1]); // Prints: Papaya

// Loop through all the elements of the fruits array
for (var i in fruitsArr) {
  document.write("<p>" + fruitsArr[i] + "</p>");
}
```

Method split()

- Untuk memisahkan string menjadi array karakter, tentukan string kosong ("") sebagai pemisah.

```
var str = "INTERSTELLAR";
var strArr = str.split("");
document.write(strArr[0]); // Prints: I
document.write(strArr[1]); // Prints: N
document.write(strArr[strArr.length - 1]); // Prints: R

// Loop through all the elements of the characters array
// and print them
for (var i in strArr) {
  document.write("<br>" + strArr[i]);
}
```

Number

Number

- JavaScript mendukung bilangan integer dan floating-point yang dapat direpresentasikan dalam notasi desimal, heksadesimal atau oktal.

```
var x = 2; // integer number
var y = 3.14; // floating-point number
var z = 0xff; // hexadecimal number
```

Number

- Angka besar juga bisa direpresentasikan dalam notasi eksponensial, misalnya 6.02e+23 (sama dengan 6.02×10^{23}).

```
var x = 1.57e4; // same as 15700
var y = 4.25e+6; // same as 4.25e6 or 4250000
var z = 4.25e-6; // same as 0.00000425
```

Number

- Angka juga dapat direpresentasikan dalam notasi heksadesimal (basis 16).

```
var x = 0xff; // same as 255
var y = 0xb4; // same as 180
var z = 0x00; // same as 0
```

Operasi pada Number

- Melakukan operasi matematika pada angka dan string dapat menghasilkan hasil yang menarik.

```
var x = 10;
var y = 20;
var z = "30";

// Adding a number with a number,
// the result will be sum of numbers
console.log(x + y); // 30

// Adding a string with a string,
// the result will be string concatenation
console.log(z + z); // '3030'

// Adding a number with a string,
// the result will be string concatenation
console.log(x + z); // '1030'

// Adding a string with a number,
// the result will be string concatenation
console.log(z + x); // '3010'

// Adding strings and numbers,
// the result will be string concatenation
console.log("The result is: " + x + y); // 'The result is: 1020'

// Adding numbers and strings,
// calculation performed from left to right
console.log(x + y + z); // 'The result is: 3030'
```

Operasi pada Number

- Jika kita melakukan operasi matematika seperti perkalian, pembagian atau pengurangan hasilnya akan berbeda.

```
var x = 10;
var y = 20;
var z = "30";

// Subtracting a number from a number
console.log(y - x); // 10

// Subtracting a number from a numeric string
console.log(z - x); // 20

// Multiplying a number with a numeric string
console.log(x * z); // 300

// Dividing a number with a numeric string
console.log(z / x); // 3
```

Operasi pada Number

- Jika kita mencoba mengalikan atau membagi angka dengan string yang bukan angka, hasilnya akan mengembalikan NaN (Not a Number).

```
var x = 10;
var y = "foo";
var z = NaN;

// Subtracting a number from a non-numeric string
console.log(y - x); // NaN

// Multiplying a number with a non-numeric string
console.log(x * y); // NaN

// Dividing a number with a non-numeric string
console.log(x / y); // NaN

// Adding NaN to a number
console.log(x + z); // NaN

// Adding NaN to a string
console.log(y + z); // fooNaN
```

Method parseInt()

- Method parseInt() digunakan untuk mengonversi string menjadi bilangan bulat (integer).

```
console.log(parseInt("3.14")); // 3
console.log(parseInt("50px")); // 50
console.log(parseInt("12pt")); // 12
console.log(parseInt("0xFF", 16)); // 255
console.log(parseInt("20 years")); // 20
console.log(parseInt("Year 2048")); // NaN
console.log(parseInt("10 12 2018")); // 10
```

Method parseFloat()

- Method parseFloat() digunakan untuk mengonversi string menjadi bilangan pecahan (floating-point number).

```
console.log(parseFloat("3.14")); // 3.14
console.log(parseFloat("50px")); // 50
console.log(parseFloat("1.6em")); // 1.6
console.log(parseFloat("124.5 lbs")); // 124.5
console.log(parseFloat("weight 124.5 lbs")); // NaN
console.log(parseFloat("6.5 acres")); // 6.5
```

Method `toString()`

- Method `toString()` digunakan untuk mengonversi tipe data non-string menjadi string.
- Method ini ada pada hampir semua tipe data primitif dalam JavaScript, seperti `number`, `boolean`, `array`, `object` dan lain-lain.

```
var x = 10;
var y = x.toString();
console.log(y); // '10'
console.log(typeof y); // string
console.log(typeof x); // number

console.log((12).toString()); // '12'
console.log((15.6).toString()); // '15.6'
console.log((6).toString(2)); // '110'
console.log((255).toString(16)); // 'ff'
```

Method toFixed()

- Method toFixed() digunakan untuk mengubah angka menjadi string dengan jumlah desimal yang tetap (fixed) setelah koma.

```
var x = 72.635;
console.log(x.toFixed());
// '73' (note rounding, no fractional part)

console.log(x.toFixed(2)); // '72.64' (note rounding)
console.log(x.toFixed(1)); // '72.6'

var y = 6.25e+5;
console.log(y.toFixed(2)); // '625000.00'

var z = 1.58e-4;
console.log(z.toFixed(2));
// '0.00' (since 1.58e-4 is equal to 0.000158)
```

Method toPrecision()

- Method toPrecision()
digunakan untuk mengubah angka menjadi string dengan panjang presisi tertentu.

```
var x = 6.235;
console.log(x.toPrecision()); // '6.235'
console.log(x.toPrecision(3)); // '6.24' (note rounding)
console.log(x.toPrecision(2)); // '6.2'
console.log(x.toPrecision(1)); // '6'

var y = 47.63;
console.log(y.toPrecision(2));
// '48' (note rounding, no fractional part)

var z = 1234.5;
console.log(z.toPrecision(2)); // '1.2e+3'
```

If dan Else

If

- if adalah struktur kendali (control flow) dalam JavaScript yang digunakan untuk mengimplementasikan logika kondisional.
- Dengan menggunakan statement if, kita bisa mengevaluasi suatu kondisi dan menjalankan blok kode tertentu jika kondisi tersebut bernilai benar (true).

```
var now = new Date();

// Sunday - Saturday : 0 - 6
var dayOfWeek = now.getDay();

if (dayOfWeek == 5) {
    alert("Have a nice weekend!");
}
```

If Else

- Statement if...else memungkinkan kita untuk mengatur aliran eksekusi program dengan cara yang berbeda tergantung pada apakah kondisi yang diberikan benar (true) atau salah (false).

```
var now = new Date();

// Sunday - Saturday : 0 - 6
var dayOfWeek = now.getDay();

if (dayOfWeek == 5) {
    alert("Have a nice weekend!");
} else {
    alert("Have a nice day!");
}
```

Else If

- else if adalah tambahan dari struktur kendali kondisional if...else yang memungkinkan kita untuk mengevaluasi lebih dari dua kondisi yang berbeda.
- Jika kita memiliki beberapa kondisi yang perlu diperiksa, else if memungkinkan kita untuk mengevaluasi kondisi-kondisi tersebut secara berurutan.

```
var now = new Date();

// Sunday - Saturday : 0 - 6
var dayOfWeek = now.getDay();

if (dayOfWeek == 5) {
    alert("Have a nice weekend!");
} else if (dayOfWeek == 0) {
    alert("Have a nice Sunday!");
} else {
    alert("Have a nice day!");
}
```

Operator Ternary

- Operator ini digunakan untuk menggantikan pernyataan if...else yang sederhana dalam bentuk yang lebih ringkas.

```
var age = 21;  
var userType = age < 18 ? 'Child' : 'Adult';  
alert(userType); // Displays Adult
```

Switch ... Case

Switch... Case

- switch...case adalah struktur kendali yang digunakan dalam untuk memilih tindakan yang akan diambil berdasarkan nilai dari sebuah ekspresi.
- Jika nilai ekspresi cocok dengan salah satu dari nilai yang telah ditentukan, blok kode yang sesuai akan dieksekusi.

```
const hari = "Senin";

switch (hari) {
    case "Senin":
        console.log("Hari Senin");
        break;
    case "Selasa":
        console.log("Hari Selasa");
        break;
    case "Rabu":
        console.log("Hari Rabu");
        break;
    default:
        console.log("Hari lainnya");
}
```

Switch... Case

- Dalam struktur switch...case, beberapa case dapat berbagi tindakan yang sama atau blok kode yang sama.
- Hal ini memungkinkan untuk mengelompokkan beberapa case yang memiliki tindakan yang identik atau serupa.

```
const day = "Monday";

switch (day) {
  case "Monday":
  case "Tuesday":
  case "Wednesday":
  case "Thursday":
  case "Friday":
    console.log("Hari kerja");
    break;
  case "Saturday":
  case "Sunday":
    console.log("Hari libur");
    break;
  default:
    console.log("Hari tidak valid");
}
```

Array

Array

- Di JavaScript, array adalah tipe data yang digunakan untuk menyimpan kumpulan nilai atau elemen secara terurut dalam satu variabel.
- Array dapat berisi berbagai jenis nilai, seperti angka, string, objek, bahkan array lainnya.
- Array digunakan untuk mengelompokkan dan mengakses nilai-nilai di dalamnya melalui indeks.

Membuat Array

- Untuk membuat array dalam JavaScript, kita dapat menggunakan tanda kurung siku [] dan memasukkan elemen-elemen yang ingin disimpan di dalamnya, dipisahkan dengan koma.

```
// Array kosong
const kosong = [];

// Array dengan beberapa elemen
const angka = [1, 2, 3, 4, 5];
const nama = ["John", "Jane", "Bob"];
const campuran = [
  1, "dua", true,
  { nama: "John", usia: 30 }
];
```

Mengakses Elemen Array

- Setiap elemen dalam array memiliki indeks berbasis nol (mulai dari 0) yang dapat digunakan untuk mengakses nilai tersebut.
- Misalnya, untuk mengakses nilai pada indeks pertama dari array angka, kita bisa menggunakan **angka[0]**, yang akan mengembalikan nilai 1.

```
const angka = [1, 2, 3, 4, 5];  
  
console.log(angka[0]); // Output: 1  
console.log(angka[2]); // Output: 3  
console.log(angka[4]); // Output: 5
```

Method & Property Array

- Array memiliki properti dan method bawaan yang memungkinkan kita untuk memanipulasi dan mengolah data di dalamnya.

```
const angka = [1, 2, 3, 4, 5];

angka.push(6); // Menambahkan elemen 6 ke akhir array
console.log(angka); // Output: [1, 2, 3, 4, 5, 6]

angka.pop(); // Menghapus elemen terakhir dari array
console.log(angka); // Output: [1, 2, 3, 4, 5]

angka.splice(2, 1); // Menghapus 1 elemen mulai dari indeks 2
console.log(angka); // Output: [1, 2, 4, 5]
```

Method & Property Array

- Properti **length** adalah properti bawaan pada objek array dalam JavaScript yang digunakan untuk mengukur jumlah elemen atau panjang dari array tersebut.
- Slicing adalah proses mengambil sebagian elemen dari sebuah array untuk membentuk array baru menggunakan method **slice()** untuk melakukan slicing pada array.

```
const angka = [1, 2, 3, 4, 5];
console.log(angka.length); // Output: 5

const slicedArray = angka.slice(1, 4);
console.log(slicedArray); // Output: [2, 3, 4]
```

Method & Property Array

- Method **concat()** digunakan untuk menggabungkan atau menggabungkan dua atau lebih array menjadi satu array baru.

```
const array1 = [1, 2, 3];
const array2 = [4, 5, 6];
const array3 = [7, 8, 9];

const newArray = array1.concat(array2, array3);

console.log(newArray);
// Output: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Method & Property Array

- Method **indexOf()** digunakan untuk mencari indeks dari suatu elemen tertentu di dalam array.
- Jika elemen ditemukan dalam array, maka method ini akan mengembalikan indeks pertama kali ditemukan.
- Jika elemen tidak ditemukan, method ini akan mengembalikan nilai -1.

```
const angka = [1, 2, 3, 4, 5];
const cariAngka = 3;
const indeks = angka.indexOf(cariAngka);

if (indeks !== -1) {
  console.log(`Angka ${cariAngka} ditemukan pada indeks ${indeks}`);
} else {
  console.log(`Angka ${cariAngka} tidak ditemukan dalam array.`);
}
```

Method & Property Array

- Method **includes()** digunakan untuk memeriksa apakah sebuah elemen tertentu ada dalam array atau tidak.
- Method ini mengembalikan nilai boolean (true jika ditemukan, false jika tidak ditemukan).

```
const angka = [1, 2, 3, 4, 5];
const cariAngka = 3;
const adaAngka = angka.includes(cariAngka);

if (adaAngka) {
  console.log(`Angka ${cariAngka} ditemukan dalam array.`);
} else {
  console.log(`Angka ${cariAngka} tidak ditemukan dalam array.`);
}
```

Method & Property Array

- Metode **find()** digunakan untuk mencari elemen pertama dalam array yang memenuhi kondisi tertentu yang diberikan dalam bentuk fungsi callback.
- Method ini akan mengembalikan nilai elemen yang pertama kali ditemukan atau **undefined** jika tidak ada elemen yang memenuhi kondisi.

```
const data = [
  { id: 1, nama: "John" },
  { id: 2, nama: "Jane" },
  { id: 3, nama: "Bob" },
];

const cariData = data.find((item) => item.id === 2);

if (cariData) {
  console.log(`Data dengan ID 2 ditemukan: ${cariData.nama}`);
} else {
  console.log("Data tidak ditemukan.");
}
```

Method & Property Array

- Method **findIndex()** digunakan untuk mencari indeks dari elemen pertama dalam array yang memenuhi kondisi tertentu yang diberikan dalam bentuk fungsi callback.
- Method ini akan mengembalikan indeks elemen yang pertama kali ditemukan atau **-1** jika tidak ada elemen yang memenuhi kondisi.

```
var arr = [1, 0, 3, 1, false, 5, 1, 4, 7];  
  
var result = arr.findIndex(function(element) {  
    return element > 6;  
});  
  
console.log(result); // Prints: 8
```

Method & Property Array

- Method **filter()** digunakan untuk membuat array baru yang berisi elemen-elemen dari array asli yang memenuhi kondisi tertentu yang diberikan dalam bentuk fungsi callback.
- Method filter() mengembalikan array baru berisi elemen-elemen yang lolos seleksi berdasarkan kondisi.

```
const angka = [10, 20, 30, 40, 50];

const angkaLebihDari30 = angka.filter(
  (elemen) => elemen > 30
);

console.log(angkaLebihDari30);
// Output: [40, 50]
```

Method & Property Array

- Method **sort()** digunakan untuk mengurutkan elemen-elemen dalam array secara alfanumerik, yang berarti angka diurutkan berdasarkan urutan nilai angka dan string diurutkan berdasarkan urutan abjad.
- Method **reverse()** digunakan untuk membalikkan urutan elemen-elemen dalam array.

```
const angka = [3, 1, 4, 2, 5];
angka.sort();
console.log(angka);
// Output: [1, 2, 3, 4, 5]

const buah = ["apel", "mangga", "pisang", "jeruk"];
buah.reverse();
console.log(buah);
// Output: ["jeruk", "pisang", "mangga", "apel"]
```

Looping

Looping

- Looping dalam JavaScript adalah konsep yang digunakan untuk melakukan perulangan atau iterasi atas satu atau lebih pernyataan atau blok kode selama kondisi tertentu terpenuhi.
- Dengan menggunakan looping, kita dapat mengulang eksekusi serangkaian instruksi secara berulang berdasarkan kondisi yang telah ditentukan.
- Hal ini memungkinkan kita untuk melakukan tugas yang sama berulang kali tanpa harus menulis instruksi secara berulang manual.

While Loop

- while loop digunakan untuk melakukan perulangan selama kondisi tertentu bernilai benar (true).

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

Do While Loop

- Loop do...while adalah varian dari while loop, di mana blok kode akan dieksekusi setidaknya satu kali sebelum kondisi diuji.

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

For Loop

- Perulangan for digunakan untuk mengulang blok kode selama kondisi tertentu terpenuhi.
- Biasanya digunakan untuk mengeksekusi blok kode untuk beberapa kali.

```
// An array with some elements
var fruits = ["Apple", "Banana",
              "Mango", "Orange",
              "Papaya"];

// Loop through all the elements in the array
for (var i = 0; i < fruits.length; i++) {
    console.log(fruits[i]);
}
```

For Of Loop

- Loop for...of digunakan untuk melakukan iterasi langsung terhadap nilai-nilai elemen dalam objek yang dapat diiterasi, seperti array, string, dan objek iterable lainnya.

```
const fruits = ["apple", "banana", "orange"];
for (const fruit of fruits) {
  console.log(fruit);
}
```

For In Loop

- Loop for...in digunakan untuk melakukan iterasi melalui properti-properti enumerable dari sebuah objek.

```
const person = {  
    name: "John",  
    age: 30,  
    occupation: "developer",  
};  
  
for (const key in person) {  
    console.log(key + ": " + person[key]);  
}
```

Object

Object

- Di JavaScript, objek (object) adalah tipe data yang berfungsi sebagai kumpulan pasangan kunci-nilai (key-value pairs).
- Setiap nilai dalam objek disimpan bersama dengan sebuah kunci yang berfungsi sebagai identifikasi atau nama dari nilai tersebut.
- Objek memungkinkan kita untuk menyimpan dan mengelompokkan beberapa nilai atau informasi yang terkait dalam satu struktur data.

```
const namaObjek = {  
    kunci1: nilai1,  
    kunci2: nilai2,  
    // dan seterusnya...  
};
```

Membuat dan Mengakses Object

- Objek bisa dibuat dengan tanda kurung kurawal {} dengan daftar properti yang sifatnya opsional.

```
const person = {  
    name: "John",  
    age: 30,  
    occupation: "Developer",  
};
```

Membuat dan Mengakses Object

- Setelah membuat objek, kita dapat mengakses nilai-nilai yang ada di dalamnya dengan menggunakan notasi titik atau notasi kurung siku.

```
const person = {  
    name: "John",  
    age: 30,  
    occupation: "Developer",  
};  
  
console.log(person.name); // Output: "John"  
console.log(person["age"]); // Output: 30
```

Menambahkan dan Mengubah Object

- Kita juga dapat menambahkan dan mengubah nilai dalam objek setelah objek dibuat.

```
const person = {  
    name: "John",  
    age: 30,  
    occupation: "Developer",  
};  
  
// Menambahkan properti "gender"  
// dengan nilai "Male" ke objek "person"  
person.gender = "Male";  
  
// Mengubah nilai properti "age"  
// dari 30 menjadi 31  
person.age = 31;
```

Method pada Object

- Kita juga dapat menambahkan method ke dalam objek setelah objek dibuat.
- Method adalah fungsi yang bertindak sebagai properti dari objek.
- Keyword this digunakan untuk merujuk ke objek saat ini (yaitu, objek person) sehingga kita dapat mengakses nilai-nilai properti dalam objeknya.

```
const person = {
    name: "John",
    age: 30,
    occupation: "Developer",
    sayHello: function() {
        console.log(`Hello, my name is ${this.name}.`);
    },
    introduce: function() {
        console.log(`I am ${this.name},
                    ${this.age} years old, working as a
                    ${this.occupation}.`);
    }
};

person.sayHello();
// Output: "Hello, my name is John."

person.introduce();
// Output: "I am John, 30 years old,
// working as a Developer."
```

Menghapus Properti Object

- Kita dapat menggunakan operator delete untuk menghapus properti tertentu dari objek.

```
const person = {
    name: "John",
    age: 30,
    occupation: "Developer",
};

// Menghapus properti "age" dari objek "person"
delete person.age;

// Output: { name: "John", occupation: "Developer" }
console.log(person);
```