**Perkenalan JavaScript And DOM**

Pada sub-modul ini kita akan banyak berkenalan dengan JavaScript. Berbeda dengan HTML dan CSS, sekarang kita berada di wilayah pemrograman yang benar-benar memanfaatkan *logic* dalam penulisan kodenya. Pengembangan website tidak hanya mengandalkan seni untuk membangun tampilan yang cantik, melainkan juga memerlukan bahasa pemrograman agar meningkatkan fungsionalitas dan membuat website lebih interaktif.

Kita akan memulainya dengan memahami apa sebenarnya itu JavaScript, mencoba menulis dan menggunakannya pada sebuah website. Selanjutnya kita akan mengenal variabel, fungsi operator, perulangan dan hal lainnya. Kita juga akan belajar bagaimana memanipulasi elemen menggunakan syntax JavaScript, menampilkan alert dan mengetahui fungsi - fungsi dari Web API yang ada seperti *document*, *window* dan sebagainya.

Pada akhir sub-modul ini kita akan menerapkan apa yang sudah kita pelajari pada project Web Kalkulator sehingga kalkulator dapat berfungsi sesuai yang kita harapkan.

**Apa itu JavaScript?**

JavaScript merupakan bahasa pemrograman *client-side* sehingga seluruh prosesnya berjalan pada sisi pengguna bukan server. JavaScript diperlukan pada pengembangan website ketika kita membutuhkan suatu interaksi dari pengguna. Sesungguhnya website hanya menampilkan konten yang statis jika hanya menggunakan HTML dan CSS.

Karena diolah pada sisi *client*, JavaScript sangat bergantung pada pengaturan dan kemampuan browser ketika melakukan sebuah proses (compiling atau rendering pada DOM). Bahkan pengguna dapat sepenuhnya tidak mengizinkan JavaScript berjalan pada browser dengan menonaktifkan dukungan JavaScript pada browser.

Meskipun memiliki nama **Java**Script bahasa pemrograman ini sama sekali **tidak ada hubungannya dengan bahasa pemrograman Java**. Sebenarnya pada tahun 1995 Netscape melahirkan bahasa pemrograman ini dengan nama “LiveScript”, namun pada saat itu bahasa pemrograman Java sedang populer. Untuk memanfaatkan kepopulerannya, namanya pun diubah menjadi “JavaScript” dan benar bahasa pemrograman ini pun akhirnya memiliki popularitas yang tidak kalah dari Java.

Setelah diadopsi di luar Netscape, JavaScript distandarisasi oleh *European Computer Manufacturer’s Association* (ECMA) itulah sebabnya terkadang ada yang menyebutnya dengan ECMAScript.

Terdapat beberapa versi dari JavaScript. Pada tahun 2000 - 2010, ECMAScript 3 merupakan versi yang banyak digunakan ketika JavaScript sedang mendominasi. Selama waktu tersebut, ECMAScript 4 sedang dalam proses pengembangan dengan harapan akan memberikan improvisasi yang cukup signifikan, namun ambisi tersebut tidak berjalan mulus sehingga pada tahun 2008 pengembangan ECMAScript 4 ditinggalkan.

Walaupun begitu, ini bukan akhir dari JavaScript. Pengembangan digantikan dengan ECMAScript 5 dengan mengurangi ambisinya dan hanya melakukan perbaikan pada hal yang tidak jadi kontroversi. Pembaharuan tersebut berhasil dan akhirnya ECMAScript 5 rilis pada tahun 2009.

Lalu pada tahun 2015 ECMAScript 6 rilis dengan membawa perubahan yang cukup besar termasuk ide - ide yang sudah direncanakan untuk versi 4. Mulai dari itu tiap tahun JavaScript melakukan update bersifat minor.

## Menuliskan Syntax Javascript pada HTML

Sama seperti styling, untuk menggunakan JavaScript pada website kita bisa menerapkannya melalui atribut HTML, embed script, atau menggunakan file external.

### Atribut HTML

Untuk menuliskan JavaScript menggunakan atribut, kita bisa menerapkannya pada atribut event seperti “onclick”, sehingga JavaScript akan dieksekusi ketika elemen tersebut ditekan oleh kursor. Contohnya sebagai berikut:

1. <button onclick="alert('Anda menekan elemen button!')">Click di sini!</button>

Ada banyak sekali atribut event yang dapat digunakan untuk menuliskan script di dalamnya. Kita bisa lihat apa saja atributnya pada tautan berikut: <https://www.w3schools.com/tags/ref_eventattributes.asp>.  
  
Tentunya atribut tersebut kita gunakan sesuai dengan kebutuhan kita. onclick merupakan salah satu atribut yang common atau banyak digunakan karena interaksi tersebut sering pengguna lakukan.

### Embedded Script

JavaScript juga dapat dituliskan dengan menanamnya (embedding) pada berkas HTML dengan menggunakan elemen <script>.

1. <script>
2. *// JavaScript dituliskan di sini.*
3. </script>

Elemen <script> dapat diletakan di dalam elemen <head> atau <body>. Akan tetapi jika kita menerapkan banyak script pada elemen <head> proses memuat halaman akan menjadi lambat, karena HTML akan membaca kode dari atas ke bawah.

### External Script

Metode lainnya yaitu dengan menggunakan berkas external yang berekstensi **.js**. Di dalam berkas tersebutlah seluruh JavaScript dituliskan. Keuntungan menggunakan metode ini adalah script dapat diterapkan pada berbagai berkas HTML tanpa menuliskan ulang scriptnya (keuntungan yang sama juga ketika kita menggunakan external css).

Untuk menghubungkan external script dengan berkas HTML, kita gunakan elemen <script> lalu tambahkan atribut src dengan nilai alamat berkas **.js** yang digunakan.

1. <script src="berkas-javascript.js"></script>

Sama seperti Embedded Script, kita bisa tuliskan tag <script> tersebut di dalam elemen <head>. Namun, direkomendasikan untuk disimpan di dalam elemen <body> sebelum tag penutup </body>.

Teknik inilah yang akan kita gunakan dalam penerapan JavaScript pada project kalkulator.

## Membuat Berkas JavaScript Pertama

Setelah mengetahui beberapa metode untuk menuliskan JavaScript pada HTML, mari kita terapkan salah satu metode yang sudah kita ketahui. Kita akan menerapkan metode external script dengan membuat berkas JavaScript yang nantinya akan digunakan untuk menuliskan script dari project kalkulator yang sudah kita buat.

Buka project kalkulator yang sudah kita buat. Di dalam folder **assets**, buat berkas baru dengan memberikan nama “**kalkulator.js**”. Ingat berkas ini harus diakhiri dengan ekstensi **.js**. Ekstensi tersebut menandakan bahwa berkas ini merupakan berkas JavaScript.

Setelah membuat berkas tersebut maka struktur dari proyek kalkulator akan tampak seperti berikut:

Buka berkas **kalkulator.js** dan tuliskan kode JavaScript berikut:

1. console.log("Selamat Anda berhasil menggunakan JavaScript pada Website");

Jangan khawatir jika Anda belum memahami maksud dari kode tersebut, nanti akan dijelaskan apa yang sudah kita tuliskan.  
  
Simpan perubahan pada berkas JavaScript tersebut kemudian buka **index.html** dan tambahkan elemen <script> dengan atribut src bernilai alamat berkas **kalkulator.js**. Tuliskan elemen <script> tepat sebelum closing tag </body>.

1. ...........
2. <script src="assets/kalkulator.js"></script>
3. </body>

Simpan perubahan pada **index.html** kemudian kita coba buka halaman tersebut dengan menggunakan browser. Maka hasilnya akan nampak seperti ini.

Tidak ada perubahan apa apa bukan? Yups, benar. Tetapi ketika kita mencoba membuka console pada DevTools kita akan mendapati pesan yang dituliskan menggunakan JavaScript. Silakan buka console pada DevTools dengan menekan ctrl+shift+i.

Script yang kita tuliskan sebelumnya berfungsi untuk menampilkan sebuah data baik itu teks (string) atau variabel, objek, fungsi dsb. Pada console website  console.log() biasanya digunakan sebagai sarana debugging sederhana untuk mengetahui nilai dari suatu variabel.

Mungkin ini akan membuat Anda bingung jika tidak memiliki dasar pemrograman. Tak apa, yang terpenting ketika pesan ini muncul pada console, itu menandakan kita sudah berhasil menerapkan JavaScript pada website kita.

**Dasar JavaScript**

Wajar jika kita menemukan banyak buku JavaScript dengan halaman super tebal karena memang banyak sekali bahasan seputar JavaScript ini. Tetapi pada materi ini kita rangkum menjadi beberapa pembahasan penting yang Anda perlu ketahui. Tujuannya agar Anda familier dengan dasar - dasar JavaScript. Dengan memahami dasar-dasar JavaScript, nantinya diharapkan kita dapat terus berlatih sehingga terbiasa dengan penggunaan sintaksnya.

Ketika mempelajari bahasa pemrograman apapun pertama kita harus memahami logika yang mendasari dari penulisan *script*, meskipun secara sintaksis setiap bahasa pemrograman berbeda namun secara umum logikanya akan sama.

Maka dari itu hindari belajar dengan cara menghafal seluruh struktur kode yang dituliskan, tapi belajarlah dengan memahami tujuan dari kode yang dituliskan. Dengan memahami kode-kode yang ada dan dipadukan logika maka kita akan mudah dalam menghadapi suatu masalah - masalah yang ada ketika mengembangkan suatu aplikasi/web.

Pada akhir materi ini, kita akan mengenal tentang variabel, array, operator komparasi, *if else statement* dan masih banyak lagi. Sudah siap belajar?

Mungkin selama pembelajaran dasar JavaScript kita perlu mencoba coba kode yang sedang dipelajari, kami menyediakan interaktif kode yang bisa Anda jalankan dan ubah secara langsung. Selain itu Anda bisa gunakan tools online seperti [glot.io](https://glot.io/) ketika menjalankan kodenya. Atau Anda juga bisa mencobanya langsung pada console browser melalui DevTools. Silakan baca artikel berikut sebelum melanjutkan pada materi selanjutnya:

* [Glot.io: Code Editor dan Compiler Online Terbaik](https://www.dicoding.com/blog/code-editor-dan-compiler-online-terbaik/)
* [Get Started With Running JavaScript In The Console](https://developers.google.com/web/tools/chrome-devtools/console/javascript)

### Statement

Sebuah script dibangun dari serangkaian statement. Statement merupakan sebuah perintah yang bertujuan untuk memberitahu apa yang harus dilakukan browser. Contohnya kode berikut merupakan statement yang menyatakan bahwa browser harus menampilkan pesan (alert) dengan kalimat “Terima kasih”.

1. alert("Terimakasih.");

Pada akhir kode terdapat tanda titik koma (;) yang digunakan untuk menandai akhir dari sebuah statement. Meskipun sebenarnya pada JavaScript kita dapat menghiraukan penulisan titik koma (;) pada akhir statement, tetapi dalam best practice nya kita biasakan akhiri statement dengan titik koma (;).

### Comment

Pada JavaScript juga terdapat fitur komentar. Seluruh teks yang dituliskan pada tanda komentar akan diabaikan (tidak dianggap ada) atau tidak akan dieksekusi. Dalam komentar ini kita bisa menuliskan teks untuk mengingatkan atau menjelaskan kode yang kita tuliskan. Hal ini bisa saja berguna jika kode yang kita tuliskan akan diubah oleh orang lain. Tapi ingat, jangan terlalu berupaya dalam menuliskan komentar yang sebenarnya tidak perlu dituliskan.

Terdapat dua metode ketika memberikan komentar. Untuk memberikan komentar pada satu baris saja, kita bisa gunakan tanda dua buah garis miring (//) pada awal barisnya.

1. *// ini merupakan komentar satu baris komentar*
2. *// alert("Terima kasih")*

Sedangkan untuk menggunakan komentar lebih dari satu baris (multiple lines comment) kita bisa gunakan tanda /\* sebagai pembuka komentar dan tanda \*/ sebagai penutup komentar. Teks apapun yang berada di antara tanda tersebut akan dijadikan komentar dan tidak akan dieksekusi.

1. */\* Ini merupakan komentar dengan lebih dari satu baris*
2. *Teks apapun yang berada disini akan dijadikan komentar.*
3. *Ketika menggunakan ini, jangan lupa untuk menutupnya.*
4. *\*/*

### Variabel

Ketika mendengar variabel apakah Anda teringat tentang pelajaran matematika? Yups, variabel di sini memiliki konsep yang sama. Variabel umumnya digunakan untuk menyimpan informasi atau nilai yang akan dikelola di dalam sebuah program.

Sebelum ECMAScript 2015 (ES6) untuk membuat variabel pada JavaScript kita gunakan keyword var**.**

1. var firstName = "Harry";

Tanda sama dengan (=) digunakan untuk menginisialisasikan nilai pada variabel, sehingga sekarang variabel **firstName**memiliki nilai teks **“Harry”**.

Kita bisa menggunakan apapun yang kita mau untuk menamai sebuah variabel, tetapi pastikan penamaannya masih masuk akal dengan konteksnya agar berikutnya kode mudah di-maintenance.

Sebaiknya hindari penamaan variabel dengan istilah umum seperti **“data”**, Gunakanlah penamaan variabel yang dapat mendeskripsikan nilai dari variabel itu sendiri. Berikut beberapa aturan dalam penamaan variabel yang perlu Anda ketahui:

* Harus dimulai dengan huruf atau underscore (\_).
* Dapat terdiri dari huruf, angka, dan underscore (\_) dengan berbagai kombinasi.
* Tidak dapat mengandung spasi (white space), jika penamaan variabel lebih dari dua kata maka tuliskan secara camelCase. Contoh firstName, lastName, catName, dll.
* Tidak dapat mengandung karakter spesial (! . , / \ + \* = dll.)

Nilai variabel yang diinisialisasi menggunakan var dapat diubah kembali nilainya, contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790?from=5789#run1-editor1)



1

var firstName = "Harry";

2

console.log(firstName);

3

​

4

firstName = "Ron";

5

console.log(firstName);

6

​

7

/\* output:

8

Harry

9

Ron

10

\*/

 Input  Reset

 Jalankan

Harry Ron

Sejak ECMAScript 2015 (ES6) selain var, menginisialisasikan variabel dapat menggunakan **let** dan **const**. ES6 melakukan improvisasi pada deklarasi variabel karena menggunakan var terdapat beberapa hal yang kontroversial, salah satunya hoisting.

Apa itu hoisting? sesuai artinya “Mengangkat” variabel yang dideklarasikan menggunakan var ini dapat diberikan nilai terlebih dahulu sebelum dideklarasikan, Contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790?from=5789#run2-editor1)



1

x = 100;

2

var x;

3

console.log(x);

4

​

5

/\* output: 100 \*/

 Input  Reset

 Jalankan

100

Ini dikarenakan proses hoisting, sebenarnya di belakang layar deklarasi variabel x diangkat ke atas sehingga kode yang tampak seperti ini:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790?from=5789#run3-editor1)



1

var x;

2

x = 100;

3

console.log(x);

4

​

5

/\* output: 100 \*/

 Input  Reset

 Jalankan

100

Hoisting menjadi kontroversial karena tidak sedikit developer yang dibuat bingung akan hal ini. Masalah ini sudah terselesaikan jika kita menggunakan let dalam mendeklarasikan variabel.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790?from=5789#run4-editor1)



1

y = 100;

2

let y;

3

console.log(y);

4

​

5

/\* ReferenceError: Cannot access 'y' before initialization \*/

 Input  Reset

 Jalankan

100

Kemudian **const** digunakan untuk mendeklarasikan sebuah variabel yang sifatnya immutable atau tidak perlu diubah nilainya. Jika kita menginisialisasi kembali nilai variabel yang menggunakan const, maka akan mendapati eror “TypeError: Assignment to constant variable.”

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790?from=5789#run5-editor1)



1

const z = 100;

2

console.log(z);

3

​

4

z = 200;

5

console.log(z)

6

​

7

/\* TypeError: Assignment to constant variable. \*/

 Input  Reset

 Jalankan

/home/glot/main.js:4 z = 200; ^ TypeError: Assignment to constant variable. at Object.<anonymous> (/home/glot/main.js:4:3) at Module.\_compile (node:internal/modules/cjs/loader:1103:14) at Object.Module.\_extensions..js (node:internal/modules/cjs/loader:1155:10) at Module.load (node:internal/modules/cjs/loader:981:32) at Function.Module.\_load (node:internal/modules/cjs/loader:822:12) at Function.executeUserEntryPoint [as runMain] (node:internal/modules/run\_main:77:12) at node:internal/main/run\_main\_module:17:47

Dengan begitu sebaiknya kita gunakan let atau const ketika mendeklarasikan variabel daripada menggunakan var.

### Variabel

Ketika mendengar variabel apakah Anda teringat tentang pelajaran matematika? Yups, variabel di sini memiliki konsep yang sama. Variabel umumnya digunakan untuk menyimpan informasi atau nilai yang akan dikelola di dalam sebuah program.

Sebelum ECMAScript 2015 (ES6) untuk membuat variabel pada JavaScript kita gunakan keyword var**.**

1. var firstName = "Harry";

Tanda sama dengan (=) digunakan untuk menginisialisasikan nilai pada variabel, sehingga sekarang variabel **firstName**memiliki nilai teks **“Harry”**.

Kita bisa menggunakan apapun yang kita mau untuk menamai sebuah variabel, tetapi pastikan penamaannya masih masuk akal dengan konteksnya agar berikutnya kode mudah di-maintenance.

Sebaiknya hindari penamaan variabel dengan istilah umum seperti **“data”**, Gunakanlah penamaan variabel yang dapat mendeskripsikan nilai dari variabel itu sendiri. Berikut beberapa aturan dalam penamaan variabel yang perlu Anda ketahui:

* Harus dimulai dengan huruf atau underscore (\_).
* Dapat terdiri dari huruf, angka, dan underscore (\_) dengan berbagai kombinasi.
* Tidak dapat mengandung spasi (white space), jika penamaan variabel lebih dari dua kata maka tuliskan secara camelCase. Contoh firstName, lastName, catName, dll.
* Tidak dapat mengandung karakter spesial (! . , / \ + \* = dll.)

Nilai variabel yang diinisialisasi menggunakan var dapat diubah kembali nilainya, contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790#run1-editor1)



1

var firstName = "Harry";

2

console.log(firstName);

3

​

4

firstName = "Ron";

5

console.log(firstName);

6

​

7

/\* output:

8

Harry

9

Ron

10

\*/

 Input  Reset

 Jalankan

Sejak ECMAScript 2015 (ES6) selain var, menginisialisasikan variabel dapat menggunakan **let** dan **const**. ES6 melakukan improvisasi pada deklarasi variabel karena menggunakan var terdapat beberapa hal yang kontroversial, salah satunya hoisting.

Apa itu hoisting? sesuai artinya “Mengangkat” variabel yang dideklarasikan menggunakan var ini dapat diberikan nilai terlebih dahulu sebelum dideklarasikan, Contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790#run2-editor1)



1

x = 100;

2

var x;

3

console.log(x);

4

​

5

/\* output: 100 \*/

 Input  Reset

 Jalankan

Ini dikarenakan proses hoisting, sebenarnya di belakang layar deklarasi variabel x diangkat ke atas sehingga kode yang tampak seperti ini:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790#run3-editor1)



1

var x;

2

x = 100;

3

console.log(x);

4

​

5

/\* output: 100 \*/

 Input  Reset

 Jalankan

Hoisting menjadi kontroversial karena tidak sedikit developer yang dibuat bingung akan hal ini. Masalah ini sudah terselesaikan jika kita menggunakan let dalam mendeklarasikan variabel.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790#run4-editor1)



1

y = 100;

2

let y;

3

console.log(y);

4

​

5

/\* ReferenceError: Cannot access 'y' before initialization \*/

 Input  Reset

 Jalankan

Kemudian **const** digunakan untuk mendeklarasikan sebuah variabel yang sifatnya immutable atau tidak perlu diubah nilainya. Jika kita menginisialisasi kembali nilai variabel yang menggunakan const, maka akan mendapati eror “TypeError: Assignment to constant variable.”

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5790#run5-editor1)



1

const z = 100;

2

console.log(z);

3

​

4

z = 200;

5

console.log(z)

6

​

7

/\* TypeError: Assignment to constant variable. \*/

 Input  Reset

 Jalankan

Dengan begitu sebaiknya kita gunakan let atau const ketika mendeklarasikan variabel daripada menggunakan var.

### ipe Data

Nilai yang kita tetapkan pada variabel pasti memiliki tipe data. Tipe data merupakan pengklasifikasian data berdasarkan jenis data tersebut. Pada JavaScript terdapat beberapa tipe data sebagai berikut:

#### Undefined

Tipe data ini terbentuk ketika sebuah variabel tidak memiliki nilai, dalam arti lain ketika kita mendeklarasikan variabel tanpa menginisialisasikan nilainya, maka variabel tersebut menjadi undefined. Contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run1-editor1)



1

let x;

2

console.log(typeof(x));

3

​

4

/\* output: undefined \*/

 Input  Reset

 Jalankan

Pada contoh kode di atas, kita mendeklarasikan variabel x, tetapi kita tidak menginisialisasikan dengan nilai apapun. Ketika kita memastikan tipe data dengan menggunakan fungsi **typeof()** maka menghasilkan output undefined.

Fungsi typeof() digunakan untuk memastikan tipe data pada variabel dengan mengembalikan tipe data tersebut dalam bentuk teks

#### Numbers

Nilai dari tipe data number adalah angka. JavaScript variabel bertipe data number dituliskan seperti ini:

1. let x = 10;

Jika angka tersebut merupakan sebuah bilangan desimal, maka kita bisa gunakan tanda titik pada pecahan bilangannya.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run2-editor1)



1

let x = 10;

2

console.log(typeof(x))

3

​

4

/\* output: number \*/

5

​

6

let y = 17.25;

7

console.log(typeof(y))

8

​

9

/\* output: number \*/

 Input  Reset

 Jalankan

Pada tipe data number kita juga dapat melakukan perhitungan aritmatika seperti penambahan, pengurangan, perkalian, dll. Berikut operator yang dapat kita gunakan dalam melakukan perhitungan aritmatika pada tipe data number:

| **Operator** | **Fungsi** | **Contoh** |
| --- | --- | --- |
| + | Penambahan | 10 + 10 = 20 |
| - | Pengurangan | 15 - 7 = 8 |
| / | Pembagian | 21 / 7 = 3 |
| \* | Perkalian | 9 \* 9 = 81 |
| % | Sisa hasil bagi | 5 % 2 = 1 |

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run3-editor1)



1

let a = 12;

2

let b = 9;

3

​

4

console.log(a + b)

5

console.log(a - b)

6

console.log(a \* b)

7

console.log(a / b)

8

console.log(a % b)

9

​

10

/\* output:

11

21

12

3

13

108

14

1.3333333333333333

15

3

16

\*/

 Input  Reset

 Jalankan

Pada operator aritmatika juga terdapat operator increment (++) dan decrement (--). Operator increment dan decrement digunakan untuk menambahkan atau mengurangi nilai 1 pada nilai variabel yang ada sekarang.

Operator ini dapat dituliskan sebelum atau sesudah variabel, tetapi hal tersebut bukan berarti sama. Berikut ketentuannya:

* Jika dituliskan setelah variabel (x++), statement akan menghasilkan nilai variabel sebelum ditingkatkan nilainya.
* Jika dituliskan sebelum variabel (++x), statement akan menghasilkan nilai variabel setelah ditingkatkan nilainya.

Lebih jelasnya berikut contoh kode dalam penerapan operator tersebut, perhatikan hasil yang di dapat.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run4-editor1)



1

/\* Increment dan Decrement \*/

2

​

3

let postfix = 5;

4

console.log(postfix++);

5

/\* output: 5 \*/

6

​

7

let prefix = 5;

8

console.log(++prefix);

9

/\* output: 6 \*/

 Input  Reset

 Jalankan

#### Strings

Tipe data selanjutnya adalah strings, String ini dasarnya adalah sebuah teks. Pada JavaScript untuk menetapkan nilai string pada variabel, gunakan tanda single (‘) atau double quote (“) di antara teksnya. Contohnya:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run5-editor1)



1

let greet = "Hello";

2

console.log(typeof(greet))

3

​

4

/\* output: string \*/

 Input  Reset

 Jalankan

Apa pun yang berada di antara tanda double quote atau single quote merupakan sebuah string, baik itu angka, huruf maupun karakter spesial.

Pada string juga kita dapat menggunakan operator plus (+). Operator tersebut pada string berfungsi untuk menggabungkan dua teks yang terpisah menjadi satu buah teks. Contohnya seperti ini:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run6-editor1)



1

let greet = "Hello";

2

let moreGreet = greet + greet;

3

console.log(moreGreet);

4

​

5

/\* output: HelloHello \*/

 Input  Reset

 Jalankan

#### Boolean

Boolean hanya dapat memiliki dua nilai, yakni true atau false. Tipe data ini menjadi kunci utama dalam penentuan logika, kita akan memahaminya nanti ketika pembahasan if/else statement. Untuk menetapkan nilai boolean pada variabel kita bisa menggunakan keyword true atau false.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run7-editor1)



1

let x = true;

2

let y = false;

3

​

4

console.log(typeof(x))

5

console.log(typeof(y))

6

​

7

/\* output:

8

boolean

9

boolean

10

\*/

 Input  Reset

 Jalankan

Atau kita bisa gunakan operator komparasi seperti lebih dari (>) atau kurang dari (<). Contohnya:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run8-editor1)



1

const a = 10;

2

const b = 12;

3

​

4

let isGreater = a > b;

5

let isLess = a < b;

6

​

7

console.log(isGreater);

8

console.log(isLess);

9

​

10

/\* output:

11

false

12

true

13

\*/

 Input  Reset

 Jalankan

#### Null

Yang terakhir adalah null. Serupa dengan undefined, namun null perlu diinisialisasikan pada variabel. null biasa digunakan sebagai nilai sementara pada variabel, tapi sebenarnya nilai tersebut “tidak ada”.

Terkadang kita perlu membuat sebuah variabel, namun kita belum memerlukan nilai apa apa dan tidak ingin terikat oleh tipe data apapun. Nah, daripada kita tidak menetapkan nilai apapun (variabel akan undefined) sebaiknya kita beri nilai null pada variabel tersebut, dan ubah nanti ketika kita membutuhkannya.

Untuk menetapkan null pada variabel, kita dapat gunakan keyword null ketika variabel tersebut diinisialisasikan.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5793?from=5790#run9-editor1)



1

let someLaterData = null;

2

console.log(someLaterData);

3

​

4

/\* output:

5

null

6

\*/

 Input  Reset

 Jalankan

Meskipun terdapat beberapa tipe data dalam JavaScript, tetapi variabel pada JavaScript memiliki sifat tipe data yang dinamis. Artinya, kita dapat memberikan tipe data yang berubah-ubah pada satu variabel yang sama. Contohnya:

1. let x; *// x merupakan undefined*
2. x = 1 *// sekarang x merupakan number*
3. x = true *// sekarang x merupakan boolean*
4. x = "Harry" *// sekarang x merupakan string*

### Arrays dan Objek

Kita sudah mengenal tipe data dasar yang ada pada JavaScript. Kali ini kita akan membahas mengenai array dan objek pada JavaScript. Keduanya dapat menampung lebih dari satu tipe data dasar yang digunakan untuk mengelola sebuah data.

#### Arrays

Array merupakan tipe data yang dapat mengelompokkan lebih dari satu nilai dari tipe data lain dengan menempatkannya pada satu variabel. Contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5798?from=5793#run1-editor1)



1

let myArray = ["Coklat", 42.5, 22, true, "Programming"];

2

console.log(myArray);

3

​

4

/\* output:

5

[ 'Coklat', 42.5, 22, true, 'Programming' ]

6

\*/

 Input  Reset

 Jalankan

[ 'Coklat', 42.5, 22, true, 'Programming' ]

Nilai - nilai yang berada pada array disusun dan diakses secara indexing. Untuk mengakses nilai di dalam array kita gunakan tanda kurung siku [] yang di dalamnya berupa angka yang merupakan posisi nilai yang ingin diakses.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5798?from=5793#run2-editor1)



1

let myArray = ["Coklat", 42.5, 22, true, "Programming"];

2

console.log(myArray[1]);

3

​

4

/\* output:

5

42.5

6

\*/

 Input  Reset

 Jalankan

42.5

Yang perlu kita ketahui adalah nilai index dimulai dari angka 0. Terlihat pada contoh kode di atas, kita mengakses index ke-1 pada myArray. Nilai yang muncul adalah nilai ke-2 dari array tersebut, yakni 42.5. Jika kita mengakses nilai array lebih dari index-nya maka hasilnya akan undefined. Index terakhir array selalu jumlah nilai array - 1.

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5798?from=5793#run3-editor1)



1

let myArray = ["Coklat", 42.5, 22, true, "Programming"];

2

console.log(myArray[0]);

3

console.log(myArray[1]);

4

console.log(myArray[2]);

5

console.log(myArray[3]);

6

console.log(myArray[4]);

7

console.log(myArray[5]);

8

console.log("Panjang nilai myArray adalah " + myArray.length + ".");

9

​

10

/\* output:

11

Coklat

12

42.5

13

22

14

true

15

Programming

16

undefined

17

Panjang nilai myArray adalah 5.

18

\*/

 Input  Reset

 Jalankan

Coklat 42.5 22 true Programming undefined Panjang nilai myArray adalah 5.

#### Objek

Objek serupa dengan array yang dapat menampung banyak nilai dengan tipe data yang beragam. Untuk mengelola data menggunakan objek, bedanya objek diakses tidak melalui indexing,  melainkan menggunakan pendekatan key-value. Untuk mengakses nilainya kita gunakan key. Key juga biasa disebut dengan properti.

Untuk menetapkan objek pada variabel gunakan tanda kurung kurawal { } dalam menginisialisasinya. Kemudian di dalamnya kita tetapkan **key: value**.

1. let object = {key1: "value1", key2: "value2", key3: "value3"}

Dalam menentukan nama key, gunakanlah nama yang dapat mendeskripsikan dari value-nya. Pada value, kita dapat mengisikan nilai dengan tipe data apapun, termasuk array. Contoh:

1. let user = {firstName: "Harry", lastName: "Potter",  age: 20, isMuggle: false, stuff: ["Wand", "Flying Car", "Owl"]};

Dalam menuliskan objek, baris baru tidaklah penting dan tidak akan berpengaruh apa pun. Sehingga lebih baik setiap kita menetapkan key-value buatlah baris baru untuk memisahkan antar nilainya, hal ini akan memudahkan kita dalam memahami struktur data yang berada pada objek.

1. let user = {
2. firstName: "Harry",
3. lastName: "Potter",
4. age: 20,
5. isMuggle: false,
6. stuff: ["Magic Wind", "Flying Car", "Owl"]
7. };

Kemudian untuk mengakses nilai dari properti objek kita dapat gunakan tanda titik diikuti dengan nama properti-nya. Contoh:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5798?from=5793#run4-editor1)



1

let user = {

2

firstName: "Harry",

3

lastName: "Potter",

4

age: 20,

5

isMuggle: false,

6

stuff: ["Magic Wind", "Flying Car", "Owl"]

7

};

8

​

9

console.log("Hallo, nama saya " + user.firstName + " " + user.lastName);

10

console.log("Umur saya " + user.age + " tahun");

11

​

12

/\* output

13

Hallo, nama saya Harry Potter

14

Umur saya 20 tahun

15

\*/

 Input  Reset

 Jalankan

Bahkan dalam properti objek, kita dapat menyimpan nilai objek lainnya. Contohnya properti firstName dan lastName dapat dikelompokan kembali dalam sebuah objek baru sebagai berikut:

1. let user = {
2. name: {
3. first: "Harry",
4. last: "Potter"
5. },
6. age: 20,
7. isMuggle: false,
8. stuff: ["Magic Wind", "Flying Car", "Owl"]
9. }

Untuk mendapatkan nilainya kita perlu mengakses properti dari objek user kemudian name. Sehingga penulisannya menjadi seperti berikut:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5798?from=5793#run5-editor1)



1

let user = {

2

name: {

3

first: "Harry",

4

last: "Potter",

5

},

6

age: 20,

7

isMuggle: false,

8

stuff: ["Magic Wind", "Flying Car", "Owl"]

9

};

10

​

11

console.log("Hallo, nama saya " + user.name.first + " " + user.name.last);

12

console.log("Umur saya " + user.age + " tahun");

13

​

14

/\* output

15

Hallo, nama saya Harry Potter

16

Umur saya 20 tahun

17

\*/

 Input  Reset

 Jalankan

Mungkin seperti itulah gambaran mengenai objek di JavaScript, penting untuk Anda garis bawahi bahwa dalam mengelola banyak tipe data pada JavaScript, Anda dapat menggunakan array maupun objek. Pembahasan di atas cukup sebagai landasan pengetahuan mengenai dua hal itu ya.  
  
Banyak hal sebenarnya yang dapat diceritakan tentang dua hal ini, terutama untuk objek. Jika Anda ingin tahu lebih dalam, Anda bisa baca dokumentasinya pada tautan yang disediakan oleh MDN:

* Array : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array>
* Objek : <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object>

### Assignment Operator

Dari contoh kode yang kita gunakan sebelumnya sebenarnya kita sudah menggunakan assignment operator. Operator ini digunakan untuk memberikan nilai pada variabel.

Pada dasarnya operator ini adalah tanda sama dengan (=), di mana tanda ini digunakan untuk menginisialisasi nilai pada variabel. Variabel yang akan diberikan nilai ditempatkan pada sebelah kiri dan nilainya ditempatkan pada sebelah kanan (nilai dapat berupa variabel lain atau nilai primitif, array, atau objek). Di antara keduanya terdapat operator assignment.

1. x = y;

Contoh pada statement tersebut kita menginisialisasikan nilai y pada variabel x, sehingga nilai x sekarang memiliki nilai yang sama dengan y.

Ada beberapa assignment operator tambahan lain dalam menginisialisasikan nilai pada variabel, atau bisa kita sebut sebagai shortcut dalam menentukan nilai. Contohnya:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/6114?from=5798#run1-editor1)



1

let x = 10;

2

let y = 5

3

​

4

x += y;

5

​

6

console.log(x);

7

​

8

/\* output

9

15

10

\*/

 Input  Reset

 Jalankan

15

Pada contoh kode di atas bisa terdapat statement x += y; apa itu artinya? Assignment operator tersebut digunakan sebagai shortcut dari x = x + y. Cara ini juga dapat digunakan pada operator aritmatika lain seperti, perkalian, pengurangan, pembagian, dan lainnya.

1. let x = 10;
2. let y = 5
4. x += y; // artinya -> x = x + y;
5. x -= y; // artinya -> x = x - y;
6. x \*= y; // artinya -> x = x \* y;
7. x /= y; // artinya -> x = x / y;
8. x %= y; // artinya -> x = x % y;

### Operator Komparasi

Sekarang kita sudah mengetahui bagaimana cara menyimpan nilai pada sebuah variabel, array, ataupun objek. Nah, selanjutnya kita akan belajar mengenai operator komparasi sebagai logika dasar dalam membandingkan nilai pada JavaScript.

Terdapat serangkaian karakter khusus yang disebut dengan operator pembanding/komparasi yang dapat mengevaluasi dan membandingkan dua nilai. Berikut daftar operator dan fungsinya:

| **Operator** | **Fungsi** |
| --- | --- |
| == | Membandingkan kedua nilai **apakah sama.** (Tidak Identik) |
| != | Membandingkan kedua nilai **apakah tidak sama**. (Tidak Identik) |
| === | Membandingkan kedua nilai **apakah identik**. |
| !== | Membandingkan kedua nilai **apakah tidak identik.** |
| > | Membandingkan dua nilai **apakah nilai pertama lebih besar dari nilai kedua**. |
| >= | Membandingkan dua nilai **apakah nilai pertama lebih besar atau sama dengan dari nilai kedua**. |
| < | Membandingkan dua nilai apakah nilai pertama **lebih kecil dari nilai kedua.** |
| <= | Membandingkan dua nilai **apakah nilai pertama lebih kecil dari atau sama dengan nilai kedua.** |

Ketika kita melakukan perbandingan antara dua nilai, JavaScript akan mengevaluasi kedua nilai tersebut dan akan mengembalikan boolean dengan nilai hasil perbandingan tersebut, baik false, atau true. Berikut contohnya:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5799?from=6114#run1-editor1)



1

let a = 10;

2

let b = 12;

3

​

4

console.log(a < b);

5

console.log(a > b);

6

​

7

/\* output

8

true

9

false

10

\*/

 Input  Reset

 Jalankan

true false

#### Perbedaan antara “sama” dan “Identik”

Dalam operator komparasi di JavaScript, hal yang menjadi sedikit “tricky” adalah membedakan antara “sama” (==) dan “identik” (===).

Kita sudah mengetahui bahwa setiap nilai pasti memiliki tipe data baik itu number, string atau boolean. Contohnya sebuah string “10” dan number 10 merupakan hal yang serupa, tetapi keduanya tidak benar-benar sama.

Hal inilah yang membedakan antara sama dan identik pada JavaScript. Jika kita ingin membandingkan hanya dari kesamaan nilainya kita bisa gunakan == tapi jika kita ingin membandingkan dengan memperhatikan tipe datanya kita gunakan ===.

Contohnya sebagai berikut:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5799?from=6114#run2-editor1)



1

const aString = '10';

2

const aNumber = 10

3

​

4

console.log(aString == aNumber) // true, karena nilainya sama-sama 10

5

console.log(aString === aNumber) // false, karena walaupun nilainya sama, tetapi tipe datanya berbeda

6

​

7

/\* output

8

true

9

false

10

\*/

 Input  Reset

 Jalankan

true false

#### Logical Operators

Terdapat beberapa operator lain yang dapat kita gunakan untuk menetapkan logika yang lebih kompleks, yakni dengan logical operators. Dengan logical operator kita dapat menggunakan kombinasi dari dua nilai boolean atau bahkan lebih dalam menetapkan logika.

Pada JavaScript terdapat tiga buah karakter khusus yang berfungsi sebagai logical operator, berikut macam-macam logical operator dan fungsinya:

| **Operator** | **Deskripsi** |
| --- | --- |
| && | Operator dan (and), logika akan menghasilkan **true** **apabila** **semua kondisi terpenuhi** (bernilai true). |
| || | Operator atau (or), logika akan menghasilkan **true** **apabila ada** **salah satu kondisi terpenuhi** (bernilai true). |
| ! | Operator tidak (not), digunakan untuk membalikan suatu kondisi. |

Berikut contoh penerapannya pada JavaScript:

* [**main.js**](https://www.dicoding.com/academies/123/tutorials/5799?from=6114#run3-editor1)



1

let a = 10;

2

let b = 12;

3

​

4

/\* AND operator \*/

5

console.log(a < 15 && b > 10); // (true && true) -> true

6

console.log(a > 15 && b > 10); // (false && true) -> false

7

​

8

/\* OR operator \*/

9

console.log(a < 15 || b > 10); // (true || true) -> true

10

console.log(a > 15 || b > 10); // (false || true) -> true

11

​

12

/\* NOT operator \*/

13

console.log(!(a < 15)); // !(true) -> false

14

console.log(!(a < 15 && b > 10)); // !(true && true) -> !(true) -> false

15

​

16

/\* output

17

true

18

false

19

true

20

true

21

false

22

false

23

\*/

 Input  Reset

 Jalankan

true false true true false false

Mungkin sebagian dari kita bertanya, sebenarnya apa kegunaan dari nilai boolean selain hanya menampilkan nilai true dan false saja? Pada pembahasan tipe data sudah pernah disebutkan bahwa boolean merupakan salah satu kunci dari logika pemrograman, karena boolean dapat mengontrol aliran pada program.

Lantas bagaimana cara boolean mengontrol sebuah aliran program? Pada bab selanjutnya, kita akan membahas mengenai if/else statement yang dapat mengontrol flow pada program, tentunya pada penggunaan statement ini boolean sangat berperan.