**Dasar-Dasar Javascript**

* **Variabel dan Konstanta**

Variabel pada pemrograman digunakan untuk **menyimpan sebuah nilai atau data**. Nilai yang disimpan pada sebuah variabel mempunyai karakteristik yang berbeda-beda seperti numerik/angka, string (kumpulan karakter), kondisi (true/false) dan lain-lain. Pada bahasa javascript, definisi sebuah variabel tidak memerlukan tipe data yang spesifik, namun dengan satu keyword **let**, sebuah variabel dapat menyimpan nilai secara umum tanpa mengkategorikan karakteristik dari nilai tersebut. Berikut contoh pembuatan variabel pada bahasa javascript.



Jadi untuk membuat sebuah variabel, kita hanya perlu mendefinisikannya dengan menggunakan keyword **let**, tanpa harus menyertakan tipe data dari variabel yang kita buat.

Konstanta pada pemrograman digunakan untuk **menyimpan sebuah nilai atau data yang bersifat statis/tetap**. Persamaan antara variabel dan konstanta adalah sama-sama untuk menyimpan sebuah nilai, tetapi nilai dari **variabel dapat diubah secara dinamis** saat program dijalankan, sedangkan **konstanta mempunyai nilai yang tetap dan tidak dapat diubah** saat program dijalankan. Untuk mendefinisikan kontanta pada bahasa javascript menggunakan keyword **const**. Berikut contoh pembuatan konstanta pada bahasa javascript.





* **Objek**

Objek merupakan struktur data yang menyimpan nilai (property) dan/atau fungsi(method). Misal kita akan menyimpan data siswa berikut.

|  |  |  |
| --- | --- | --- |
| NIS | Nama | Jurusan |
| 101 | Suneo | RPL |
| 102 | Jeko | TKJ |

Jika kita ingin menyimpan data tersebut, biasanya kita akan membuat beberapa variabel untuk menyimpannya seperti berikut ini.



Namun cara di atas tidak efektif karena membutuhkan variabel yang banyak. Oleh karena itu data di atas bisa kita simpan dengan konsep objek seperti berikut ini.



Pada sebuah objek, terdapat penulisan key dan value. **Key** merupakan property atau sub data dari objek, **value** merupakan nilai dari setiap key. Pendefinisian key dan value pada objek diletakkan di dalam tanda **brace** atau kurung kurawal **{}**.



Untuk mengubah atau mendapatkan nilai dari sebuah objek, kita dapat menuliskannya dengan format ***nama\_objek.key***. Berikut contoh penerapannya.



Output



* **Array**

Array merupakan kumpulan dari beberapa variabel atau objek. Setiap elemen yang disimpan pada array mempunyai posisi masing-masing yang disebut dengan **indeks** **array**. Indeks array dimulai dari posisi 0. Pendefinisian array dapat menggunakan keyword **let** dan elemen array berada di dalam tanda **kurung siku** atau **[]**. Setiap elemen pada array, dipisahkan dengan **tanda koma (,)**. Berikut contoh pendefinisian array.



* **Mendapatkan jumlah data yang tersimpan pada array**

Untuk mengetahui jumlah elemen atau data yang tersimpan pada array menggunakan format ***nama\_array.length***. Berikut contoh penerapannya.Output



* **Menambahkan / memasukkan data pada array**

Untuk menambahkan data bar uke dalam array yang telah dibuat menggunakan format ***nama\_array.push(new data)***. Berikut contoh penerapannya.

Code:



Output



Contoh penerapan penambahan data pada array objek.

Code:



Output:



* **Menghapus data pada array**

Untuk menghapus data pada array menggunakan format ***nama\_array.splice(index, jumlah data)***. Berikut contoh penerapan penghapusan data pada array.

Data array nama saat didefinisikan



Kode penghapusan data



Dari kode di atas, maka data array nama akan dihapus dari indeks kedua sebanyak 1 data. Sehingga data pada index kedua yaitu “Putri” akan dihapus, dan array nama akan menjadi seperti berikut ini.



* **Menampilkan data pada array object**

Array Object merupakan kumpulan dari beberapa objek, dan setiap objek memiliki beberapa atribut didalamnya. Salah satu cara untuk menampilkan data yang tersimpan pada array object adalah dengan menggunakan fungsi **map()** dari array. Berikut ini contoh penerapan fungsi map() pada array.



Fungsi **map()** bisa disebut proses **scanning satu per satu data** yang terdapat pada array. Pada baris 9, terdapat variabel “sis” dan “index”. Variabel “sis” merupakan variabel yang menunjukkan objek yang discan pada saat itu (current value). Variabel “index” merupakan variabel yang menunjukkan posisi indeks dari objek yang discan pada saat itu (current index). Penamaan variabel tersebut bisa diganti sesuai dengan kehendak programmer.

Output



* **Operator**

Operator merupakan simbol atau karakter khusus yang digunakan di dalam program untuk melakukan suatu operasi, misal operasi penjumlahan dua nilai, membandingkan kesamaan dua nilai, memberi nilai ke variabel, dan sebagainya.

Jika yang mengoperasikan adalah operator, maka yang dioperasikan disebut sebagai operand. Dalam suatu operasi, suatu nilai atau variabel dapat berperan sebagai operand.

Jenis operator pada Javascript

* **Operator Aritmatika (Arithmetic)**

Operator aritmatika digunakan untuk melakukan operasi aritmatika/perhitungan matematis. Berikut ini adalah macam operator aritmatika pada Javascript.

|  |  |
| --- | --- |
| **Operator** | **Fungsi** |
| + | Operasi penjumlahan / penggabungan (concat) |
| - | Operasi pengurangan |
| \* | Operasi perkalian |
| / | Operasi pembagian |
| % | Operasi sisa hasil bagi |
| \*\* | Operasi perpangkatan |
| ++ | Operasi increment (penambahan nilai 1) |
| -- | Operasi decrement (pengurangan nilai 1) |

Berikut ini contoh penerapan dari operator aritmatika.

Code



Output



Dari output di atas, pada saat a + b menghasilkan nilai 7, dan pada saat a + c menghasilkan nilai 210 (bukan 12). Dapat disimpulkan bahwa pada saat operator (+) digunakan akan melakukan **operasi penjumlahan** jika data yang dioperasikan **sama-sama berjenis numerik**, dan akan melakukan **operasi penggabungan (*concat*)** jika **salah satu data yang dioperasikan berjenis string**.

* **Operator Penugasan (Assignment)**

Operator penugasan digunakan untuk memberi nilai ke variabel yang sudah didefinisikan. Berikut ini adalah macam-macam operator penugasan.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Contoh** | **Penjelasan** |
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| \*= | x \*= y | x = x \* y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |
| \*\*= | x \*\*= y | x = x \*\* y |

* **Operator Perbandingan (Comparation)**

Operator perbandingan digunakan untuk membandingkan nilai dari dua operand. Operasi dengan operator pembanding selalu menghasilkan nilai boolean (TRUE atau FALSE). Tabel berikut menunjukkan macam-macam operator perbandingan.

|  |  |
| --- | --- |
| **Operator** | **Deskripsi** |
| == | Membandingkan kesamaan nilai antar operand |
| === | Membandingkan kesamaan nilai dan tipe antar operand |
| != | Membandingkan ketidaksamaan nilai antar operand |
| !== | Membandingkan ketidaksamaan nilai dan tipe antar operand |
| > | Membandingkan nilai yang lebih besar |
| < | Membandingkan nilai yang lebih kecil |
| >= | Membandingkan nilai yang lebih besar sama dengan |
| <= | Membandingkan nilai yang lebih kecil sama dengan |

Berikut ini contoh penggunaan operator perbandingan

Code



Output



Berdasarkan output di atas, saat **a == c** bernilai **true** karena a dan c sama-sama bernilai 5 meskipun berbeda tipe (a bertipe numeric dan c bertipe string). Namun saat **a===c** bernilai **false** karena operator === tidak hanya membandingkan kesamaan nilai, namun juga membandingkan kesamaan tipe dari data yang dibandingkan.

* **Operator Logika (Logic)**

Operator logika mengoperasikan dua nilai operand yang bertipe data boolean (TRUE atau FALSE) dan menghasilkan nilai boolean pula.

Perhatikan tabel berikut.

|  |  |
| --- | --- |
| **Operator** | **Deskripsi** |
| && | Disebut operator AND.  Jika kedua nilai operand adalah TRUE maka hasilnya TRUE. Jika salah satu atau keduanya bernilai false, maka hasilnya false.  Contoh:   * Jika a = TRUE dan b = TRUE a && b menghasilkan TRUE * Jika a = TRUE dan b = FALSE a && b menghasilkan FALSE * Jika a = FALSE dan b = FALSE a && b menghasilkan FALSE |
| || | Disebut operator OR.  Jika salah satu atau kedua nilai operand adalah TRUE maka hasilnya TRUE. Jika keduanya bernilai false, maka hasilnya FALSE.  Contoh:   * Jika a = TRUE dan b = TRUE a || b menghasilkan TRUE * Jika a = TRUE dan b = FALSE a || b menghasilkan TRUE * Jika a = FALSE dan b = FALSE a && b menghasilkan FALSE |
| ! | Disebut operator NOT  Membalikkan nilai logika suatu operand.  Contoh:   * Jika a = TRUE !a menghasilkan FALSE * Jika a = FALSE !a menghasilkan TRUE |

* **Percabangan**

Konsep percabangan pada pemrograman digunakan untuk menentukan aksi atau perintah yang dilakukan berdasarkan kondisi tertentu. Terdapat beberapa bentuk percabangan dalam Javascript yaitu sebagai berikut.

**Bentuk If**

If merupakan sebuah blok yang berisikan kode Javascript yang akan dieksekusi jika kondisinya bernilai true. Perhatikan contoh berikut ini.



Statement “2020 adalah tahun kabisat” akan ditampilkan jika sisa hasil bagi 4 dari 2020 bernilai 0.

**Bentuk If.. Else..**

**Else** merupakan sebuah blok kode Javacript yang akan dieksekusi jika kondisi pada **If** bernilai false. Perhatikan contoh berikut ini.



Statement “2020 adalah tahun kabisat” akan ditampilkan jika sisa hasil bagi 4 dari 2020 bernilai 0 dan jika sisa hasil bagi 4 dari 2020 tidak sama dengan 0, maka statement “2020 bukan tahun kabisat” akan ditampilkan.

**Bentuk If.. Else If… Else…**

Else If merupakan limpahan pengencekan kondisi jika pada If sebelumnya tidak terpenuhi. Jika kondisi else if ini tidak terpenuhi akan dilimpahkan ke kondisi else if yang lain. Jika tidak ada kondisi yang terpenuhi maka akan mengeksekusi blok else. Perhatikan contoh berikut ini.



* **Perulangan (Looping)**

Konsep perulangan digunakan untuk mengeksekusi perintah berulang-ulang selama kondisi perulangan bernilai true. Terdapat beberapa bentuk perulangan dalam Javascript yaitu sebagai berikut.

**For Loop**

Format For Loop



**Statement 1** dieksekusi sebelum proses perulangan berlangsung, biasanya berisi nilai awal dari sebuah iterasi atau perulangan.

**Statement 2** mendefinisikan kondisi agar perulangan terus berlangsung

**Statement 3** diekskusi setiap kali perulangan berlangsung, biasanya berisi perubahan nilai iterasi saat terjadi perulangan.

Perhatikan contoh penerapan for loop berikut ini.



Output



**For/In Loop**

For/In Loop digunakan untuk menampilkan property / atribut yang dimiliki oleh sebuah objek. Perhatikan contoh penerapan For/In Loop berikut ini.



Output



**For/Of Loop**

For/Of Loop digunakan untuk menampilkan seluruh nilai yang tersimpan pada array. Perhatikan penerapan For/Of Loop berikut ini.



Output



**While Loop**

While Loop akan mengulang perintah yang ada di dalam bloknya selama kondisinya bernilai benar. Berikut format penulisan While Loop.



Contoh penerapan While Loop



Output



**Do.. While Loop**

Do.. While Loop merupakan jenis perulangan While. Perulangan ini akan mengeksekusi perintah yang ada di dalam bloknya sekali sebelum terjadi pengecekan kondisi, kemudian akan melakukan perulangan jika kondisinya bernilai benar. Berikut ini format penulisan Do..While Loop.



Contoh penerapan Do.. While Loop



Output



* **Fungsi (Function)**

Fungsi merupakan sebuah blok kode yang dibuat untuk bagian-bagian tertentu. Fungsi ini akan dieksekusi jika dipanggil. Format penulisan fungsi pada Javascript versi ES6 adalah sebagai berikut.



Parameter merupakan sebuah data yang akan diolah di dalam fungsi, data tersebut diisikan Ketika fungsi itu dipanggil. Berikut contoh penerapan fungsi menggunakan Javascript.



Output



* **Class**

Class merupakan template atau rancangan dari sebuah objek. Class mendefinisikan property atau atribut dan method dari sebuah objek. Misal kita akan membuat sebuah class **PersegiPanjang** yang memiliki atribut panjang dan lebar, dan memiliki method menghitung luas dan keliling. Perhatikan contoh berikut.

Code



Output



Penjelasan:

**Constructor()** merupakan sebuah fungsi yang pertama kali dipanggil saat proses inisiasi objek (*instance object*) yang ditandai dengan adanya penggunaan keyword ***new***. **Atribut atau property** dari objek **dituliskan/didefinisikan/dideklarasikan** di dalam method **constructor()** dan menggunakan keyword ***this***. Pada baris 20, didefiniskan objek dengan nama **tanah** yang merupakan implementasi dari class PersegiPanjang dengan nilai atribut panjang diset 10, dan atribut lebar diset 50.

**Inheritance (Pewarisan)**

Inheritance atau pewarisan merupakan salah satu konsep OOP yang digunakan untuk mewariskan property atau method dari super class (kelas induk) kepada sub class (kelas anak). Perhatikan contoh implementasi inheritance berikut ini.



Berdasarkan script di atas, class PersegiPanjang sebagai super class mempunyai atribut panjang dan lebar, dan method luas() dan keliling(). Sedangkan class Balok mewarisi class PersegiPanjang dengan sehingga class Balok dapat menggunakan atribut dan method milik class PersegiPanjang. Pada baris 18, nilai panjang dan lebar diset dengan memanggil method **super()** yang berarti memanggil method **constructor()** milik **super class**. Penggunaan atribut atau method milik super class tetap menggunakan keyword ***this***.

Berikut ini implementasi dari class Balok.



Output



* **JSON**

JSON merupakan singkatan dari **J**ava**S**cript **O**bject **N**otation. JSON adalah format untuk penyimpanan dan pengiriman data. JSON sering digunakan saat data dikirimkan dari server ke halaman web. Format JSON hampir sama dengan format object pada Javascript (pada bahasan sebelumnya). Berikut format penulisan JSON:

* Data terdiri dari key dan value. **Key** merupakan label dari data dan **value** adalah nilai dari data tersebut. Penulisan key dan value pada JSON harus berada di dalam tanda *double quotes* (petik dua).
* Setiap data dipisahkan dengan koma.
* Curly Braces atau {…} menunjukkan data berformat object.
* Square Braces atau […] menunjukkan data berformat array.

Berikut ini contoh dari JSON.

Code:

|  |
| --- |
| let barang = [    { nama: "Beras", harga: 10000, jumlah: 5 },      { nama: "Gula", harga: 14000, jumlah: 5},      { nama: "Telur", harga: 20000, jumlah: 2},      { nama: "Minyak Goreng", harga: 9000, jumlah: 10}  ]  console.log(JSON.stringify(barang)); |

Output:



**JSON** itu sendiri berformat **text**, sehingga saat data yang akan dijadikan format JSON harus menggunakan sebuah fungsi **JSON.stringify(*data*)**. Sebaliknya, jika kita akan akan mengolah data JSON, kita harus mengkonversi terlebih dahulu ke dalam bentuk object atau array menggunkan fungsi **JSON.parse(*data*)**.