**Belajar Javascript Dasar**

Sumber : Programming Zaman Now (Eko Kurniawan Khannedy)

**Membuat Kode Javascript (JS)**

* Langsung di file HTML
* Menggunakan file.js
* Include file.js di file HTML

**Tipe Data Number**

JS Hanya mendukung satu tipe data number (bil. Bulat, bil. Desimal, bil. Pecahan, dll)

**Number Notation**

JS mendukung beberapa number notation, defaultnya berbasis 10 (desimal). Selain itu, Hexadecimal (0xFF), Binary (0b101010), dan Octal (0o10).

**Escape Sequence (karakter khusus)**

|  |  |
| --- | --- |
| **Code** | **Fungsi** |
| \n | ENTER |
| \t | TAB |
| \’ | ‘ |
| \” | “ |
| \\ | \ |

**Variable ‘let’ dan ‘const’**

*‘****let’***, (nilai/valuenya bisa diubah2) dan ***‘const’***, (nilai/valuenya tidak bisa dirubah)

**Operator Perbandingan** *(output boolean)*

|  |  |
| --- | --- |
| **Operator** | **Keterangan** |
| > | Lebih dari |
| < | Kurang dari |
| >= | Lebih dari sama dengan |
| <= | Kurang dari sama dengan |
| == | Sama dengan |
| === | Sama dengan dan sama tipe data |
| != | Tidak sama dengan |
| !== | Tidak sama dengan atau tidak sama tipe data |

**Operator Logika** *(output boolean)*

|  |  |
| --- | --- |
| **Operator** | **Keterangan** |
| && | AND |
| || | OR |
| ! | NOT |

**Console Method**

|  |  |
| --- | --- |
| **Console Method** | **Keterangan** |
| console.**info**(...) | Memberitahu informasi |
| console.**warn**(...) | memberi info peringatan |
| console.**error**(...) | memberi info error |
| console.**table**(...) | memberi info dalam bentuk tabel |

**String Tamplate**

Bisa mensubstitusi data dari luar string ke dalam string, menggunakan *backtick* (` `), jadi tidak perlu (+) untuk menambah string.   
Dapat mengambil nilai dari sebuah variable, menggunakan ${namaVariable}.  
Dapat menambahkan *expression,* menggunakan ${expression}.  
Dapat digunakan untuk menulis *multiline string*.

**Konversi Tipe Data String dan Number**

|  |  |
| --- | --- |
| **Function** | **Keterangan** |
| **parseInt**(*string*) | mengkonversi dari string ke number (bil. Bulat) |
| **parseFloat**(*string*) | mengkonversi dari string ke number (bil. Pecahan) |
| **Number**(*string*) | mengkonversi dari string ke number (bil. Bulat atau bil. pecahan) |
| **numberVariable.toString**() | mengkonversi dari number ke string |

**NaN (Not a Number),** jika NaN dioperasikan dengan data number lainnya, maka hasilnya akan NaN juga dan tidak akan terjadi error. Untuk fungsi *Number()*, tidak akan mentolelir kesalahan pada data, sehingga bila tipe data bukan number, maka hasilnya akan NaN.

**isNaN(*number*) , Function.** Digunakan untuk mengecek apakah suatu number NaN atau bukan, dan hasilnya berupa tipe data boolean (*true/false*).

**Tipe Data Array**

Tipe data yang berisi kumpulan data (tidak ada batasan harus tipe data apa, number, string, campuran, array didalam array, dll), dan bersifat dinamis (dapat ditambah/dikurangi datanya).   
Array dihitung dengan indexing dimulai dari nilai index ke 0,1,2,..dst. Simbol array adalah kurung siku ‘[]’.  
Setiap menambahkan data/nilai pada suatu array otomatis data/nilai tsb ditempatkan pada urutan index terakhir dari array tsb.  
Menambah array dilakukan dengan kode, **namaVariable.push(*nilai array yg ingin ditambahkan*)**

**Operasi di Array**

|  |  |
| --- | --- |
| **Operasi** | **Keterangan** |
| array.push(value) | menambah data ke array |
| array.length | untuk mengetahui/mendapatkan panjang array |
| array[index-ke] | mendapatkan data dari posisi index tertentu |
| array[index-ke] = value | mengubah data pada posisi index tertentu |
| delete array[index] | menghapus data pada suatu index, namun index tidak bergeser |

**Tipe Data Object**Di Javascript associative array tidak didukung. Jika kita ingin memasukan data bukan number di index array, maka JS akan mengubah array tsb menjadi object, dan beberapa operasi di array mungkin bisa merubahnya.  
Tipe data object mirip dengan array, yg membedakan index tipe data object bisa menggunakan string dan biasa disebut attributes atau properties, bukan index.   
Simbol object adalah dengan kurung kurawal, **{object}.** Untuk operasi mirip seperti operasi di array.

**If Expression**  
digunakan untuk fungsi percabangan, akan dieksekusi bila kondisi ***true***.

**Else Expression**  
bila kondisi pada if bernilai ***false***, maka else expression dapat dijalankan.

**Else If Expression**  
kadang dalam if, kita butuh beberapa kondisi sehingga Else if digunakan untuk menentukan kondisi lain bila kondisi pada if tidak terpenuhi.

**Alert, Prompt dan Confirm**Alert digunakan untuk peringatan berupa pop up text.  
Prompt digunakan untuk meminta input string dari user berupa pop up input text.  
Confirm digunakan untuk meminta input boolean dari user berupa pop up input pilihan.

**Undifined**Undifined adalah kata kunci di JS *(tidak bisa digunakan sebagai nama variable)*, dan sebuah tipe data.  
Bila suatu variable belum ditambahkan value, maka variable tsb bertipe data undifined.  
Undifined berbeda dengan null.

**Null,** merupakan representasai data kosong. Null sudah didefinisikan datanya hanya saja nilainya null, sedangkan undifined belum didefinisikan datanya.

**Switch Expression**  
Switch adalah statement percabangan yang sama dengan if, tetapi lebih sederhana (hanya untuk perbandingan ‘**==**’).

**Operator typeof**  
typeof adalah operator yg digunakan untuk melihat tipe data sebuah value atau variable.  
hasil dari operator typeof adalah string tipe datanya.

|  |  |
| --- | --- |
| **Type** | **Hasil operator typeof** |
| Undefined | “undefined” |
| Null | “object” |
| Boolean | “boolean” |
| Number | “number” |
| BitInt | “bigint” |
| String | “string” |
| Symbol | “symbol” |
| Function | “function” |
| Lainnya | “object” |

**Ternary Operator (?),** adalah operator sederhana dari if statement, terdiri dari kondisi yang dievaluasi, jika menghasilkan true, maka nilai pertama yg diambil, jika false, maka nilai kedua yg diambil.

**Nullish Coalescing Operator**  
Nullist value adalah nill dan undefined.  
Nullist coalescing operator (**??**) mirip dengan ternary operator, yg berbeda adalah kondisi, jika bernilai null atau undefined, maka value defaultnya yg diambil.

**Optional Chaining**  
operator yg digunakan untuk mengamankan ketika ingin mengakses properti dari sebuah object dari data nullish, jika tidak menggunakan optional chaining maka akan terjadi error. Simbolnya (**?**).

**Falsy dan Truthy**  
**Falsy** adalah value yang dalam konteks boolean dianggap false. Dan, **truthy** kebalikan dari falsy, dimana data nya dianggap true (bukan bernilai falsy).

|  |  |
| --- | --- |
| **Data Falsey** | **Keterangan** |
| false | Boolean false |
| 0, -0 | number 0 dan -0 dianggap false |
| “”, ‘’, `` | semua string kosong dianggap false |
| null | null dianggap false |
| undefined | undefined dianggap false |
| NaN | not a number dianggap false |

**Operator Logika di Non Boolean**  
di JS bisa menggunakan operator logika AND (**&&**) dan OR (**||**) untuk tipe data non boolean.  
AND, mengambil data falsy yg paling kiri (awal), bila tidak ada maka ambil data yg paling terakhir.  
OR, mengambil data truthy yg paling kiri (awal), bila tidak ada maka ambil data yg paling terakhir.

**For Loop**for adalahh salah satu kunci yg bisa digunakan untuk melakukan operasi perulangan. Perulangan akan terus dilakukan selama kondisi pada for terpenuhi.  
Sintak perulangan for adalah :  
**for(init statement; kondisi; post statement) {  
 //block perulangan  
}**

* Init statement akan dieksekusi hanya sekali diawal, sebelum perulangan
* Kondisi akan dilakukan pengecekan dalam setiap perulangan, jika true perulakan akan dilakukan, jika false maka akan berhenti.
* Post statement akan dieksekusi setiap kali diakhir perulangan
* Init statement, konsidi dan post statement tidak wajib diisi, jika kondisi tidak diisi, berarti kondisi selalu bernilai true.

**While Loop**  
merupakan versi perulangan yang lebih sederhana, hanya terdapat kondisi perulangan, tanpa ada init statement dan post statement. Blok fungsi While loop :  
**While(kondisi){  
 //blok perulangan  
}**

**Do While Loop**

* Perulangan mirip dengan while, berbeda pada pengecekan kondisi yg biasanya dilakukan diawal perulangan dilakukan, pada do while loop dilakukan setelah perulangan dilakukan.
* Oleh karena itu, pada do while loop, minimal perulangan dilakukan sekali walaupun kondisi tidak bernilai true.

Blok fungsi Do while loop :

**//Parameter**

**do {**

**//blok perulangan  
}  
while (kondisi)**

**Break dan Continue**Break digunakan untuk menghentikan perulangan, sedangkan Continue digunakan untuk menghentikan perulangan saat ini dan melanjutkan ke perulangan selanjutnya.

**Foor In**merupakan perulangan for yang digunakan untuk mengiterasi seluruh data property di object atau index array. Walaupun bisa digunakan untuk array, namun tidak direkomendasikan (lebih baik menggunakan for of).

**For Of**digunakan untuk melakukan iterasi terhadap isi value dari iterable tipe data, seperti array, string dll (kecuali object).

**Function**  
merupakan blok kode program yang akan berjalan saat kita panggil, dan bisa dipanggil berkali-kali (di bahasa pemrograman lain function disebut method).  
membuat function :  
***function* namaFunction() {  
 //kondisi  
}**

Memanggil function :  
**namaFunction( )**

**Function Parameter**kita bisa mengirim informasi ke function yang ingin kita panggil, dengan cara menambahkan parameter atau argumen pada function yg kita buat.  
Parameter ditempatkan di dalam kurung **()** di deklarasi method. Parameter bisa lbih dari satu, caranya dengan dipisah oleh tanda koma (**,**).

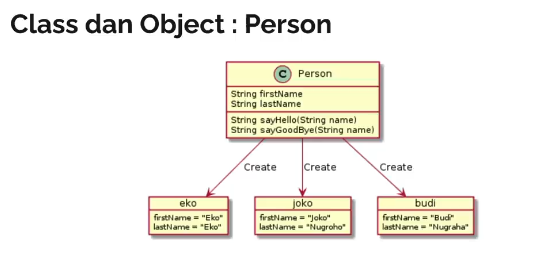
**Function Return Value**  
Secara default, return tidak menghasilkan value apapun, namun dapat digunakan untuk menyimpan value dari function tsb.  
Function hanya bisa mengembalikan satu data, bila ingin banyak data maka gunakan array sebagai return valuenya.  
Return dapat digunakan untuk menghentikan eksekusi pada sebuah function.

**Destructuring**

**Restrict**

**Debugger**

**Javascript Object Oriented Programming**

**Object Oriented Programming (OOP)**OOP adalah sudut pandang bahasa pemrograman yang berkonsep ‘objek’, beberapa istilah dalam OOP antara lain Object dan Class.  
**Object,** data yg berisi field / properties / attributes dan method / function / behaviour.  
**Class,** blue print, prototype atau cetakan untuk membuat object. Class berisi deklarasi semua properties dan functions yg dimiliki object.   
note : > Setiap object selalu terdiri / dibuat dari class  
 > Sebuah class bisa membuat object tanpa ada batasan  


Class

Object

Properties

Ket. : > Class : Person (berisi 3 object dengan nama, eko, joko, dan budi )  
> Tiap object berisi 2 properties, firstName dan lastName  
> Ketiga object tidak ada kaitannya satu sama lain (isolate satu sama lain)

**Constructor Function,** merupakan fungsi yang biasa digunakan untuk membuat class, dengan huruf awal pada nama functionnya uppercase (kapital).   
Untuk membuat object pada class/constructor function, maka gunakan kata kunci ‘**new**’, lalu diikuti dengan nama constructor functionnya.   
Untuk menambahkan properti pada constructor function, maka menggunakan ‘**this**’.  
Sama seperti tipe data object lainnya, jika kita menambahkan method di constructor function, secara otomatis object yang dibuat akan memiliki method tsb.  
Karena dalam JS class adalah function, maka kita bisa menambahkan parameter didalamnya, sehingga memungkinkan untuk mengirim langsung data lewat parameter di constructor tsb.  
Kode Constructor Function :  
**function NamaConstructorFunction(ParameterConstructorFunction) {  
 this.namaProperti = value ;  
 this.namaProperti = ParameterConstructorFunction ; //parameter bisa menjadi value di properti  
 this.namaFunction = function(namaParameter) { //ini adalah method pada constructor function  
 console.info(`${namaParameter} ${this.namaProperti}`)  
 }  
}  
const NamaVariable = new NamaFunction(ParameterConstructorFunction);**

**Constructor Inheritance**Didalam constractor function, kita bisa memanggil constractor function lainnya, sehingga nilai dr constractor lain tsb dapat diakses/digunakan. Untuk memanggilnya, dapat dengan :  
 **NamaConstructor.call (this, parameter) ; //pemisah antara this dan parameter adalah koma (,)**

**Prototype,** JS dikenal sebagai dengan bahasa pemrograman berbasis prototype.

**Prototype Inheritance**  
saat membuat object dari constructor function, object tsb disebut **instance**, semua properti (value atau method) akan berada didalam instance objectnya.  
setiap membuat sebuah constructor function, maka otomatis akan dibuatkan prototypnya. (ex : constractor function ‘Person’, maka akan ada **Person.prototype**).  
saat membuat object instance, secara otomatis object tsb adalah turunan dari constructor prototypenya.  
untuk mengakses prototype pada sebuah instance, bisa menggunakan **\_\_proto\_\_** .

**Menambah Property ke Prototype**  
prototype mirip object, dimana kita bisa menambah property baik value atau method.  
Saat menambah property ke prototype, secara otomatis semua object instance yg turunan dari prototype itu akan memiliki property tsb.