T

***Nama : Hanif Amrullah Kelas : A***

***Nim : L200140026 Tugas : Tugas 1 Sistem Informasi Geografis***

1. Buatlah daftar TIPE DATA di Java Script.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipe data dasar** (primitif) | ***Fungsi*** | ***Ciri - Ciri*** | ***Keterangan*** |
| String | Untuk membuat teks/string atau meyimpan data teks/string pada variable(var) | Teks/string diantara tanda petik tunggal (’) atau tanda petik ganda (”) | Contoh :  var carName ="Volvo XC60";   // Using double quotes var carName = 'Volvo XC60';   // Using single quotes |
| Number/  Numerik/Bilangan | Untuk membuat atau menyimpan data berupa bilangan | Tanpa tada petik dan berupa bilangan atau angka | * Dibedakan menjadi 2 yaitu tipe bilangan bulat(decimal, octal, heksadesimal) dan tipe bilangan titik mengambang/riil (bilangan pecahan) * Operator berdasarkan prioritas : perkalian(\*), pembagian(/) dan modulus(%) yang memiliki prioritas diatas penambahan(+) dan pengurangan(-). Jika tingkat prioritas sama maka di utamakan operasi dari kiri ke kanan dan juga tanda kurung di operasikan terlebih dahulu. * Contoh :   var x1 = 34.00;     // Written with decimals var x2 = 34;        // Written without decimals  var y = 123e5;      // 12300000 var z = 123e-5;     // 0.00123 |
| Booleans | Sering dimanfaatkan dalam logika kondisional  (if/while) | **true** berarti benar  **false** berarti salah | Contoh :  var x = true; var y = false; |
| Null | Untuk menyatakan nilai khusus yang berarti nilanya tidak ada/tidak pernah diberi nilai | Menggunakan kata “null” | Contoh  var person = null;         // Value is null, but type is still an object |
| Undefined | Untuk membuat variable tanpa nilai | Memiliki nilai dan type tidak terdefinisi | Contoh:  var person;                // Value is undefined, type is undefined |
| Empty Value | Untuk membuat variable sebagai string namun tidak didefinisikan isi dari string | Petik tunggal atau ganda dengan string kosong atau tak terdefinisi diantara petik tersebut | Contoh:  var car = "";              // The value is "", the typeof is "string" |
| Simbol | Sebagai jenis data baru di Standar JavaScript ECMAScript Edition 6 | unik dan nilai primitif immutable juga memungkinkan digunakan untuk properti Objek | Symbol dijuluki sebagai atoms |
| **Tipe data objek** | ***Fungsi*** | ***Ciri - Ciri*** | ***Keterangan*** |
| Array | berisi kumpulan dari nilai atau tipe data lain | array JavaScript ditulis dengan tanda kurung persegi dengan penomoran indexs dimulai dari 0/nol dan bersifat dinamis | * Array di dalam **JavaScript** tidak bertipe (untyped array). **Elemen** dari array bisa bertipe data **string**, **number** dan **boolean** dalam sebuah array yang sama, bahkan elemen dari array bisa berupa **objek** atau **array** yang lain. * Contoh :   var cars = ["Saab", "Volvo", "BMW"];  var arr1 = []                               // array kosong, 0 elemen  var arr2 = [1,2,3,4,5]                      // array dengan 5 elemen  var arr3 = [3,4.1,"belajar","JavaScript"]   // array dengan 4 elemen |
| Date | Untuk memudahkan penanganan tanggal dan waktu |  | Contoh pendeklarasian objek Date:  var d = new Date(); // tangal dan waktu sekarang var d = new Date(milliseconds); // miliseconds dimulai 1970/01/01 var d = new Date(dateString); var d = new Date(year, month, day, hours, minutes, seconds, milliseconds); |
| Function | Menjalankan perintah perintah yang di definikan oleh function tersebut | Kata kunci : function | Contoh pendeklarasian funsi :  function myFunc(){} |
| Objek | Bersifat dinamis dan cara untuk pemrograman OOP di javascript | Terdiri dari properti yang digunakan untuk mendeskripsikan objek tersebut | Contoh :  var objek\_kosong = {};  var mobil = {  "warna-badan": "merah",  "nomor-polisi": "BK1234AB"  }; |

1. Buatlah daftar nama fungsi dan kelompoknya.
2. decodeURI() : fungsi yang digunakan untuk memecahkan kode URI

var uri = "my test.asp?name=stale&car=saab";  
var enc = encodeURI(uri);  
var dec = decodeURI(enc);  
var res = enc + "<br>" + dec;

1. [decodeURIComponent()](http://www.w3ii.com/id/jsref/jsref_decodeuricomponent.html) : fungsi untuk mengkodekan komponen URI

var uri = "http://w3ii.com/my test.asp?name=stale&car=saab";  
var uri\_enc = encodeURIComponent(uri);  
var uri\_dec = decodeURIComponent(uri\_enc);  
var res = uri\_enc + "<br>" + uri\_dec;

1. [encodeURI()](http://www.w3ii.com/id/jsref/jsref_encodeuri.html) : fungsi yang digunakan untuk mengkodekan URI.

var uri = "my test.asp?name=stale&car=saab";  
var res = encodeURI(uri);

1. [encodeURIComponent()](http://www.w3ii.com/id/jsref/jsref_encodeuricomponent.html) : fungsi mengkodekan komponen URI, untuk mengkodekan karakter khusus dan karakter ( , / ? : @ & = + $ # )

var uri = "http://w3ii.com/my test.asp?name=stale&car=saab";  
var res = encodeURIComponent(uri);

1. [escape()](http://www.w3ii.com/id/jsref/jsref_escape.html) : fungsi ini membuat string portabel, sehingga dapat ditransmisikan melalui jaringan apapun kepada komputer yang mendukung karakter ASCII dan karakter khusus dengan pengecualian ( \* @ - \_ + . / )

document.write(escape("Need tips? Visit w3ii!"));

1. eval() : fungsi mengevaluasi atau mengeksekusi argumen.Jika argumen adalah ekspresi, eval () mengevaluasi ekspresi. Jika argumen adalah satu atau lebih pernyataan JavaScript, eval () mengeksekusi pernyataan.

var x = 10;  
var y = 20;  
var a = eval("x \* y") + "<br>";  
var b = eval("2 + 2") + "<br>";  
var c = eval("x + 17") + "<br>";  
  
var res = a + b + c;

1. [isFinite()](http://www.w3ii.com/id/jsref/jsref_isfinite.html) : Fungsi ini mengembalikan false jika nilai adalah + infinity, infinity, atau NaN (Not-a-Number), jika tidak maka kembali benar.

var a = isFinite(123) + "<br>";  
var b = isFinite(-1.23) + "<br>";  
var c = isFinite(5-2) + "<br>";  
var d = isFinite(0) + "<br>";  
var e = isFinite("123") + "<br>";  
var f = isFinite("Hello") + "<br>";  
var g = isFinite("2005/12/12");  
  
var res = a + b + c + d + e + f + g;

1. [isNaN()](http://www.w3ii.com/id/jsref/jsref_isnan.html) : Fungsi ini mengembalikan nilai true jika nilai setara dengan NaN. Jika tidak maka kembali palsu.

isNaN(123) //false  
isNaN(-1.23) //false  
isNaN(5-2) //false  
isNaN(0) //false  
isNaN('123') //false  
isNaN('Hello') //true  
isNaN('2005/12/12') //true  
isNaN('') //false  
isNaN(true) //false  
isNaN(undefined) //true  
isNaN('NaN') //true  
isNaN(NaN) //true  
isNaN(0 / 0) //true

1. [Number()](http://www.w3ii.com/id/jsref/jsref_number.html) : fungsi mengkonversi argumen objek untuk nomor yang mewakili nilai objek.

var x1 = true;  
var x2 = false;  
var x3 = new Date();  
var x4 = "999";  
var x5 = "999 888";  
  
var n =   
Number(x1) + "<br>" +   
Number(x2) + "<br>" +   
Number(x3) + "<br>" +   
Number(x4) + "<br>" +   
Number(x5);

1. [parseFloat()](http://www.w3ii.com/id/jsref/jsref_parsefloat.html) : fungsi mem-parsing string dan mengembalikan sejumlah floating point.

var a = parseFloat("10") + "<br>";  
var b = parseFloat("10.00") + "<br>";  
var c = parseFloat("10.33") + "<br>";  
var d = parseFloat("34 45 66") + "<br>";  
var e = parseFloat(" 60 ") + "<br>";  
var f = parseFloat("40 years") + "<br>";  
var g = parseFloat("He was 40") + "<br>";  
  
var n = a + b + c + d + e + f + g;

1. [parseInt()](http://www.w3ii.com/id/jsref/jsref_parseint.html) : fungsi mem-parsing string dan mengembalikan integer.

var a = parseInt("10") + "<br>";  
var b = parseInt("10.00") + "<br>";  
var c = parseInt("10.33") + "<br>";  
var d = parseInt("34 45 66") + "<br>";  
var e = parseInt(" 60 ") + "<br>";  
var f = parseInt("40 years") + "<br>";  
var g = parseInt("He was 40") + "<br>";  
  
var h = parseInt("10",10)+ "<br>";  
var i = parseInt("010")+ "<br>";  
var j = parseInt("10",8)+ "<br>";  
var k = parseInt("0x10")+ "<br>";  
var l = parseInt("10",16)+ "<br>";  
  
var n = a + b + c + d + e + f + g + "<br>" + h + i + j + k +l;

1. [String()](http://www.w3ii.com/id/jsref/jsref_string.html) : fungsi mengkonversi nilai objek untuk string.

var x1 = Boolean(0);  
var x2 = Boolean(1);  
var x3 = new Date();  
var x4 = "12345";  
var x5 = 12345;  
  
var res =  
String(x1) + "<br>" +  
String(x2) + "<br>" +  
String(x3) + "<br>" +  
String(x4) + "<br>" +  
String(x5);

1. [unescape()](http://www.w3ii.com/id/jsref/jsref_unescape.html) : fungsi decode string disandikan.

var str = "Need tips? Visit w3ii!";  
var str\_esc = escape(str);  
document.write(str\_esc + "<br>")  
document.write(unescape(str\_esc))

1. Sebuah fungsi pada Javascript dibuat dengan cara seperti berikut.

**function** tambah(a, b) {

hasil = a + b;

**return** hasil;

}

Cara penulisan fungsi seperti ini dikenal dengan nama *function declaration*, atau deklarasi fungsi. Terdapat empat komponen yang membangun fungsi yang baru kita definisikan di atas, yaitu:

1. Kata kunci function, yang memberitahu Javascript bahwa kita akan membuat fungsi.
2. Nama fungsi, dalam contoh di atas adalah tambah. Dengan memberikan sebuah fungsi nama maka kita dapat merujuk ke fungsi tersebut dengan nama yang diberikan. Harus diingat bawa nama fungsi bersifat *opsional*, yang berarti **fungsi pada Javascript tidak harus diberi nama**. Kita akan membahas tentang hal ini lebih dalam nanti.
3. Daftar parameter fungsi, yaitu a, b pada contoh di atas. Daftar parameter ini selalu dikelilingi oleh tanda kurung (()). Parameter boleh kosong, tetapi tanda kurung wajib tetap dituliskan. Parameter fungsi akan secara otomatis didefinisikan menjadi variabel yang hanya bisa dipakai di dalam fungsi. Variabel pada parameter ini diisi dengan nilai yang dikirimkan kepada fungsi secara otomatis.
4. Sekumpulan perintah yang ada di dalam kurung kurawal ({}). Perintah-perintah ini dikenal dengan nama badan fungsi. Badan fungsi dieksekusi secara berurut ketika fungsi dijalankan.

Penulisan deklarasi fungsi (*function declaration*) seperti di atas merupakan cara penulisan fungsi yang umumnya kita gunakan pada bahasa pemrograman imperatif dan berorientasi objek. Tetapi selain deklarasi fungsi Javascript juga mendukung cara penulisan fungsi lain, yaitu dengan memanfaatkan ekspresi fungsi (*function expression*). Ekspresi fungsi merupakan cara pembuatan fungsi yang memperbolehkan kita melewatkan nama fungsi. Fungsi yang dibuat tanpa nama dikenal dengan sebutan fungsi anonim atau fungsi lambda. Berikut adalah cara membuat fungsi dengan ekspresi fungsi:

**var** tambah = **function** (a, b) {

hasil = a + b;

**return** hasil;

};

Terdapat hanya sedikit perbedaan antara ekspresi fungsi dan deklarasi fungsi:

1. Penamaan fungsi. Pada deklarasi fungsi, kita langsung memberikan nama fungsi sesuai dengan sintaks yang disediakan Javascript. Menggunakan ekspresi fungsi kita pada dasarnya menyimpan sebuah fungsi anonim ke dalam variabel, dan nama fungsi adalah nama variabel yang kita buat. Perlu diingat juga bahwa pada dasarnya ekspresi fungsi *adalah* fungsi anonim. Penyimpanan ke dalam variabel hanya diperlukan karena kita akan memanggil fungsi nantinya.
2. Ekspresi fungsi dapat dipandang sebagai sebuah ekspresi atau perintah standar bagi Javascript, sama seperti ketika kita menuliskan kode var i = 0;. Deklarasi fungsi merupakan konstruksi khusus untuk membuat fungsi. Hal ini berarti pada akhir dari ekspresi fungsi kita harus menambahkan ;, sementara pada deklarasi fungsi hal tersbut tidak penting.

Karena pada Javascript sebuah fungsi juga adalah sekaligus sebuah objek, maka pada buku ini kita akan hampir selalu menggunakan ekspresi fungsi. Setiap kali kita menciptakan fungsi, pada dasarnya kita membuat sebuah objek Function baru, dengan nama yang kita berikan. Karenanya, secara eksplisit menuliskan bahwa kita membuat objek baru dan memperlakukan objek tersebut sama seperti perintah-perintah lain dalam program akan menyederhanakan kode program kita, yang pada akhirnya akan mempermudah kita mengerti kode kita nantinya.

Aturan pembuatan fungsi, baik ekspresi fungsi maupun deklarasi fungsi, sama dengan aturan penulisan ekspresi. Di mana kita dapat menuliskan ekspresi, kita dapat mendefinisikan fungsi juga. Karena aturan ini, maka kita juga dapat mendefinisikan fungsi di dalam fungsi lainnya. Fungsi yang berada di dalam fungsi lainnya memiliki akses terhadap semua variabel yang ada pada fungsi penampungnya. Keterhubungan fungsi di dalam fungsi ini dikenal dengan nama *closure*. Kita akan membahas tentang *closure* dan melihat bagaimana *closure* memberikan kemampuan ekspresi yang sangat besar kepada pengembang pada bagian berikutnya. Berikut adalah sedikit contoh fungsi built-in yang sering digunakan dalam pemrograman di google Maps beserta penjelasannaya.

Fungsi **Map()**

Berfungsi sebagai function dari kelas javascript yang memuat gambar peta dari Google Maps API.

Kode ini mendefinisikan variabel (bernama map) dan memberikan variabel itu ke objek Map baru. Fungsi Map() dikenal sebagai *konstruktor* dan definisinya ditampilkan di bawah ini.

Map(mapDiv:Node, opts?:MapOptions)

* mapDiv:Node/ letak peta di halaman HTML menggunakan fungsi **document.getElementById()**
* opts?:[MapOptions](https://developers.google.com/maps/documentation/javascript/tutorial?hl=id#MapOptions) / Opsi peta memiliki 3 opsi yaitu

center,zoom,dan jenis peta. Dalam center menggunakan funsi

**LatLng(titik\_latitude, titik\_langitude**), range zoom maksimal adalah 1 sampai 21.sedangkan jenis peta menggunakan objekMapTypeId.ROADMAP (memperlihatkan tampilan peta jalan default. Ini adalah tipe peta default), MapTypeId.SATELLITE (menampilkan citra satelit Google Earth), MapTypeId.HYBRID (menampilkan kombinasi tampilan normal dan satelit), MapTypeId.TERRAIN (menampilkan peta fisik berdasarkan informasi medan).

* Fungsi **setMapTypeId()**

sebenarnya fungsi tidak mengatur tipe peta ini secara langsung, melainkan mengatur mapTypeId untuk mereferensikan MapType menggunakan identifier. Maps JavaScript API V3 menggunakan registry tipe peta, yang dijelaskan di bawah ini, untuk mengelola semua referensi ini.

Contoh :

map.setMapTypeId(google.maps.MapTypeId.TERRAIN);

* Fungsi **setTilt()** dan **getTilt()**

setTilt(value/derajat) : memberikan kemiringan yang ditampilkan pada peta, sedangkan getTilt()untuk mendapatkan pencerminan nilai kemiringan peta.

* Fungsi **setHeading()**

mengarahkan gambar ke salah satu arah mata angin

* Fungsi **addDomListener(**)

Berfungsi sebagai untuk mendengarkan dan mengikat kejadian DOM.

addDomListener(instance:Object, eventName:string, handler:Function)

* instance:Object : Anggota hierarki DOM seperti window atau document.body.myform
* eventName:string : nama kejadian missal ‘load’
* handler:Function : memanggil fungsi yang akan diload

Meskipun kode di atas adalah kode Maps JavaScript API, metode addDomListener() akan diikat ke objek window browser dan memungkinkan API berkomunikasi dengan objek di luar domain normal API.