Nama : Topan Ramadhan

NIM : L200140138

Kelas : A

**TUGAS 1**

**Sistem Informasi Geografi**

1. **Tipe Data**
2. String

Digunakan untuk menuliskan serangkaian karakter yang berada didalam tanda kutip tunggal ( ‘…..’ ) atau tanda kutip ganda ( “….. “).

Contoh :

var str = ‘Hello World!’;

var str = “Hello World!”;

1. Numbers

Javascript hanya memiliki satu jenis numbers yaitu ditulis dengan atau tanpa decimal dan dapat juga ditulis dengan ilmiah (eksponensial).

Contoh :

var x1 = 21,14;

var x2 = 23;

var y = 1812e-5;

1. Booleans

Tipe ini hanya memiliki 2 nilai yaitu true atau false.

Contoh :

var  Y = (X > 96);

contoh diatas menunjukkan bahwa jika X lebih besar dari 96 maka Y akan bernilai True.

1. Array

Jenis ini ditulis dengan tanda kurung persegi dan item nya dipisahkan dengan tanda koma.

Contoh :

var motor = [“beat”, “vario”, “scoopy”];

1. Obyek

Tipe ini ditulis dengan kurung kurawal, terdapat pasangan nilai dan dipisahkan dengan koma.

Contoh :

var orang = {firstName : “Novilia”, lastName : “Putri”, usia : 20};

1. **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() : 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() : 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 w3!"));

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() : 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() : 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() : 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() : 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. **Cara membuat sebuah fungsi dalam javascript**

Sebuah fungsi pada Javascript dibuat dengan cara seperti berikut:

|  |  |
| --- | --- |
| 1  2  3  4 | 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:

|  |  |
| --- | --- |
| 1  2  3  4 | 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.