Nama : Ari Agung Prasetiyo

Kelas : A

NIM : L200140109

Tugas :

1. Buatlah daftar Type Data.
2. Daftar nama Fungsi dan Kelompoknya
3. Bagaimana cara membuat sebuah fungsi dalam javascript

Jawab:

1. Daftar type data dalam Javascript
2. JavaScript Type are Dynamic

JavaScript memiliki tipe dinamis. Ini berarti bahwa variabel yang sama dapat digunakan untuk menyimpan berbagai jenis data.

Contoh:

var x;               // Now x is undefined  
var x = 5;           // Now x is a Number  
var x = "John";      // Now x is a String

1. Javascript Strings

Sebuah String (atau string teks) adalah serangkaian karakter.String ditulis dengan tanda kutip. Anda dapat menggunakan tanda kutip tunggal atau ganda.

Contoh:

var carName = "Volvo XC60";   // Using double quotes  
var carName = 'Volvo XC60';   // Using single quotes

Anda dapat menggunakan tanda kutip dalam string, selama mereka tidak cocok dengan kutipan string sekitarnya.

Contoh:

var answer = "It's alright";             // Single quote inside double quotes  
var answer = "He is called 'Johnny'";    // Single quotes inside double quotes  
var answer = 'He is called "Johnny"';    // Double quotes inside single quotes

1. JavasScript Numbers

JavaScript hanya memiliki satu jenis nomor.

Nomor dapat ditulis dengan, atau tanpa desimal.

Contoh :

var x1 = 34.00;     // Written with decimals  
var x2 = 34;        // Written without decimals

1. JavaScript Booleans

Booleans hanya dapat memiliki dua nilai: Benar atau salah.

Contoh:

var x = true;  
var y = false;

1. JavaScript Array

Ditulis dengan tanda kurung persegi. Item array dipisahkan dengan koma.

Contoh:

Kode berikut mendeklarasikan sebuah array disebut mobil, yang berisi tiga item (nama mobi):

var cars = ["Saab", "Volvo", "BMW"];

1. JavaScript Object

Ditulis dengan kurung kurawal. Properti objek ditulis dengan nama: Pasangan nilai, dipisahkan dengan koma.

Contoh:

var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};

1. The typ eof Operator

Anda dapat menggunakan JavaScript operator untuk menemukan jenis variabel JavaScript.

Operator type mengembalikan jenis variabel atau ekspresi

Contoh:

typeof ""                  // Returns "string"  
typeof "John"              // Returns "string"  
typeof "John Doe"          // Returns "string"

1. Primitive Data

Nilai data primitif adalah nilai data sederhana tunggal tanpa sifat tambahan dan metode.

Tipenya:

1. string
2. number
3. boolean
4. null
5. undefined

Contoh:

typeof "John"              // Returns "string"   
typeof 3.14                // Returns "number"  
typeof true                // Returns "boolean"  
typeof false               // Returns "boolean"

1. Daftar nama Fungsi dan Kelompoknya
2. encodeUR()

fungsi yang digunakan untuk mengkodekan URI

contoh:

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

1. [encodeURIComponent()](http://www.w3ii.com/id/jsref/jsref_encodeuricomponent.html)

mengkodekan komponen URI, untuk mengkodekan karakter khusus dan karakter ( , / ? : @ & = + $ # )

Contoh:

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 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.

Contoh:

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.

Contoh:

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. 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.