Nama : Rizal Andian Rudiarko

NIM : L200140159

1. Daftar tipe data dalam Javascript

a. Data Primitive

Data nilai primitive adalah satu data nilai sederhana dengan tidak ada tambahan sifat dan metode. Jenis operator yang bisa kembali satu dari macam tipe primitif ini:

No	Tipe	Keterangan

1	**String**	**String** di dalam **JavaScript** adalah tipe data yang terdiri dari kumpulan karakter yang berurutan. Atau di dalam penggunaan sehari-hari string adalah tipe data yang menampung nilai **text** atau kalimat.
		Untuk mendeklarasikan tipe **string** dapat dilakukan dengan cara menuliskan string diantara tanda petik tunggal (') atau tanda petik ganda ("). Contoh :
		`var A = 'ini adalah pendeklarasian string';`
		`var B = 'ini juga sama string';`

2	**Number**	Ada dua macam tipe**number**, yaitu bilangan bulat dan bilangan real. Untuk bilangan bulat, kalian bisa merepresentasikan dengan basis desimal, oktal, dan heksadesimal. Contoh :
		`var A = 100;`
		`var A = 0x2F;`
		Untuk pendeklarasian tipe bilangan real kalian bisa menggunakan tanda titik atau notasi ilmiah (notasi **E**). Contoh :
		`var a = 123.456`
		`var b = 1.234567E+3`

3	**Boolean**	**Boolean** adalah tipe data yang hanya mempunyai dua nilai, yakni benar (**True**) atau salah (**False**). Tipe ini biasanya digunakan untuk mengecek suatu kondisi tertentu. Contoh : `var a = (b>50);`

4	**Null**	Tipe Null kata kunci (*keyword*) khusus yang berarti '*tidak memiliki nilai',* digunakan untuk merepresentasikan variabel yang tidak diberi nilai awal (inisialisasi).
5	**Undifined**	**Undefined** adalah variabel global di dalam ***javascript***, dan bukan merupakan objek khusus seperti **null**.

b. Data Komplex

Tipe dari operator yang bisa kembali dari dua tipe komplex:

No	Tipe	Keterangan

1	**Function**	Function javascript didefinisikan dengan kata kunci **function.** Contoh :
		`function myFunction(a, b) {`
		`  return a * b;`
		`}`

2	**Object**	Objek javascript ditulis dengan {}. Sifat dari objek ditulis dengan nama: nilai pasang, yang dipisahkan dengan tanda kutip. `var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};`

c. Tipe data Array

1.	**Array**	**Array** adalah **tipe data** yang berisi kumpulan dari nilai atau tipe data lain. Nilai di dalam array disebut dengan **elemen,** dan setiap *elemen* memiliki '*nomorurut'* yang dikenal dengan istilah **index**.
		Array di dalam **JavaScript** bersifat *dinamis*, dan kita tidak perlu mendefenisikan berapa ukuran **array** pada saat membuat variabel. Jumlah elemen dapat ditambah dan dikurang setiap saat.
		`<script>`
		`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`
		
		`//buat object objek1`
		`var arr4 = new Array();  // array kosong, 0 elemen`
		`var arr5 = new Array(1,2,3,4,5)  // array dengan 5 elemen`
		`var arr6 = new Array(3,4.1,"belajar","JavaScript")  // array 4 elemen`

2. Daftar nama fungsi dan kelompoknya

Fungsi pada *JavaScript* merupakan serangkaian kode yang dirancang untuk melaksanakan suatu tugas tertentu. Fungsipada *JavaScript* memiliki peranan yang sama dimana ia tidak akan dieksekusi secara langsung sampai dilakukan pemanggilan terhadap fungsi tersebut. Proses pemanggilan dapat juga dilakukan melalui suatu tombol saat diklik atau dibuat agar secara otomatis memanggil dirinya sendiri.

Tabel berikut memperlihatkan daftar fungsi bawaan JavaScript, beserta keterangan singkatnya.

No	Function	Keterangan
1	**Eval**	Digunakan untuk mengevaluasi kode JavaScript
2	**IsFinite**	Digunakan untuk mengevaluasi apakah suatu bilangan finite atau infinite
3	**IsNaN**	Digunakan untuk memeriksa apakah suatu nilai tergolong bilangan atau tidak
4	**Parseint**	Menghasilkan suatu nilai bilangan integer dari masukan
5	**parseFloat**	Menghasilkan suatu nilai bilangan floating dari masukan berupa sebuah nilai string
6	**Number**	Mengembalikan referensi sebuah object menjadi bentuk bilangan
7	**String**	Mengembalikan referensi sebuah object menjadi bentuk string
8	**Escape**	Mengembalikan kode heksadesimal dari suatu karakter yang menjadi masukan
9	**Unescape**	Mengembalikan sebuah karakter ASCII dari suatu kode heksadesimah yang menjadi masukan (lawan dari function escape)
10	**EncodeURI**	Mengkodekan suatu URI (Uniform Resource Identifier) menjadi suatu bentuk yang sesuai dengan standar UTF-8
11	**DecodeURI**	Mengkodekan balik hasil dari function encodeURI
12	**EncodeURIComponent**	Mengkodekan komponen-komponen URI sehingga sesuai dengan standar UTF-8.
13	**DecodeURIComponent**	Mengkodekan balik hasil dari function

		encodeURIComponent

3. Bagaimana cara membuat sebuah fungsi dalam Javascript

Fungsi adalah sebuah blok kode yang mengeksekusi hanya bila Anda mengatakan itu untuk mengeksekusi.

Hal ini dapat terjadi ketika sebuah peristiwa, seperti ketika pengguna mengklik tombol, atau dari panggilan dalam naskah Anda, atau dari panggilan dalam fungsi lain.

Fungsi dapat ditempatkan baik di head dan di bagian <body> dokumen, pastikan bahwa fungsi ada, ketika panggilan dilakukan.

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. Setiap kali kita menciptakan fungsi, pada dasarnya kita membuat sebuah objek `Function` baru, dengan nama yang kita berikan.

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*.