**Nama: Adnan Ghazi Raushan Noor**

**No. Absen: 5**

**Tugas: Perulangan Pada JavaScript**

1. **For Loop**

For Loop digunakan untuk menjalankan blok kode berulang kali sampai kondisi tertentu terpenuhi.

sintaks umum dari For Loop dalam JavaScript:

for ([inisialisasi]; [kondisi]; [increment]){

// kode yang dijalankan

}

* Inisialisasi: Bagian ini untuk mengatur variabel counter untuk loop. Bagian ini hanya dijalankan sekali pada awal loop.
* Kondisi: Loop akan terus berjalan selama kondisi ini bernilai true. Setelah setiap iterasi, kondisi diperiksa lagi. Jika kondisi masih true, loop tersebut akan menjalankan lagi, jika false, loop akan berhenti.
* Increment: Bagian ini memperbarui nilai variabel counter. Ini biasanya digunakan untuk memperbarui atau mengincrement variabel counter.

contoh For Loop:

for (let i = 1; i <= 5; i++) {

console.log(i);

}

Di sini, for mendeklarasikan variabel ‘i‘ dan menginisialisasinya ke 1. Memeriksa nilai ‘i’ lebih kecil dari 5, dan menambahkan nilai ‘i’ 1 setiap putarannya. Maka output yang dihasilkannya adalah: 1, 2, 3, 4, 5.

1. **While Loop**

While Loop akan mengeksekusi blok kode berulang kali selama kondisi yang ditentukan bernilai true.

sintaks umum dari For Loop dalam JavaScript:

while (kondisi) {

// kode yang akan dijalankan

}

While diikuti oleh kondisi dalam tanda kurung (). Selama kondisi ini bernilai true, kode dalam tanda kurung kurawal {} akan terus dijalankan. Jika kondisi tersebut diganti menjadi false, sistem akan keluar dari loop dan melanjutkan pada kode berikutnya.

Contoh While Loop:

let i = 0;

while (i < 5) {

console.log("Nomor: " + i);

i++;

}

variabel ‘i’ diatur nilai awalnya menjadi 0. Loop tersebut akan berjalan selama ‘i’ < 5. Di dalam setiap iterasi, kode akan mencetak “Nomor: ” diikuti oleh nilai ‘i’ saat ini, kemudian ‘i’ akan ditambahkan 1. Jadi, outputnya akan:

"Nomer: 0"

"Nomer: 1"

"Nomer: 2"

"Nomer: 3"

"Nomer: 4"

Hindari perulangan tak terhingga. Pastikan kondisi dalam perulangan pada akhirnya menjadi “false” jika tidak, perulangan tidak akan pernah berakhir.

1. **Do While**

Do While akan selalu melaksanakan blok kode sekali, kemudian akan memeriksa kondisi, dan melanjutkan perulangan selama kondisinya bernilai true.

Berikut penulisan sintaks do while dalam JavaScript:

do {

// blok kode untuk dijalankan

} while (kondisi);

Contoh Do While:

var i = 0;

do {

console.log(i);

i++;

} while (i < 5);

Pada contoh di atas, selama ‘i’ masih kurang dari 5, blok kode dalam perulangan akan terus dijalankan. Kode ini akan mencetak angka 0 hingga 4 ke dalam console.

Perbedaan **do while** dan **while:**

* Perulangan do while akan selalu menjalankan blok kode sekurang-kurangnya satu kali, kemudian baru memeriksa kondisi.
* Perulangan while akan memeriksa kondisi terlebih dahulu baru menjalankan blok kode.

1. **Loop For .. in & For .. of**

* **For..in**

For..in loop yang khusus dibuat untuk iterasi object.

Berikut adalah sintaks dasar dari Loop For..in:

for (variabel of object) {

// kode untuk setiap item

}

Contoh:

let mobil = {model: 'Toyota', warna: 'hitam', tahun: 1998};

for (let x in mobil) {

console.log(x, ':' , mobil[x]);

}

Dengan loop for...in, bisa memperoleh properti (dalam hal ini x) dan nilainya.

* **For..of**

Sebuah Loop For...of membuat loop melalui objek iterable, seperti Array, String, Map, Set, dan sejenisnya.

Berikut adalah sintaks dasar dari Loop For..of:

for (variabel of iterable) {

// kode untuk setiap item

}

Di dalam loop for of, variabel akan mendapatkan nilai dari setiap elemen dalam iterable, satu per satu.

Contoh pada array:

let buah = ['apel', 'jeruk', 'mangga'];

for (let item of buah) {

console.log(item);

}

// Output: apel

// jeruk

// mangga

Contoh pada string:

let teks = 'JavaScript';

for (let karakter of teks) {

console.log(karakter);

}

Outputnya akan menampilkan perhuruf: J, a,v,a,S,c,r,i,p,t

1. **Loop Break & Continue**

* **Break**

statement break digunakan untuk “memecah” atau “menghentikan” eksekusi dari loop yang sedang berjalan, dan mengontrol program untuk berlanjut ke pernyataan setelah loop.

Contoh:

for (let i = 0; i < 10; i++) {

if (i === 5) {

break; //berhenti

}

console.log(i);

}

loop akan berhenti atau break ketika i memiliki nilai 5.

*Catatan:*

*Break dan Continue dapat digunakan bersama dengan label. Label adalah pengenal unik yang dilampirkan ke loop. Ketika break digunakan dengan label, itu akan “memecah” loop yang ditandai dengan label tersebut, bukan loop tempat break itu sendiri berada. Misalnya:*

*outerLoop: for (let i = 0; i < 5; i++) {*

*innerLoop: for (let j = 0; j < 5; j++) {*

*if (i === 3 && j === 3) {*

*break outerLoop;*

*}*

*console.log(`i = ${i}, j = ${j}`);*

*}*

*}*

* **Continue**

Continue dalam JavaScript membuat alur eksekusi melompati bagian sisa dari perulangan saat ini dan berlanjut langsung ke iterasi berikutnya dari perulangan.

Contoh:

for (let i = 0; i < 10; i++) {

if (i === 5) {

Continue //melompati perulangan

}

console.log(i);

}

angka 5 tidak akan dicetak ke konsol. Itu karena saat i bernilai 5, pernyataan ‘continue’ dijalankan dan bagian sisa dari loop saat ini (yaitu console.log(i)) dilewati. Kemudian, eksekusi melanjutkan iterasi berikutnya dari perulangan.

1. **Nested Loop**

Nested loop, atau loop bersarang, adalah konsep dimana suatu loop berjalan di dalam loop lain. Jadi, saat satu iterasi dari loop luar berjalan, semua iterasi dari loop dalam akan berjalan seluruhnya.

Tetapi penggunaan nested loop bisa mempengaruhi performa aplikasi negatif jika tidak digunakan dengan hati-hati, karena setiap penambahan loop di dalam loop lain akan mengalikan jumlah iterasi.

Contoh membuat segitiga bintang:

let rows = 5;

for(let i=1; i<=rows; i++){

let str = '';

for(let j=1; j<=i; j++){

str += '\*';

}

console.log(str);

}