CLIENT SIDE SCRIPTING

THEORIES TO WRITE IN PRACTICAL BOOK

Practical No. 02: Decision making and looping statements

I] DECISION MAKING STATEMENTS:

- if
- if-else
- if-else-if ladder
- nested if
- switch-case

1. if:

   Used to execute block of code only when specified condition is true.

   ```
   if(condition)
   {
        //code
   }
   ```

2. if-else:

   When the if condition is false, the block written under else{} is executed instead.

3. if-else-if ladder:

   When there to be checked multiple conditions after prior condition returns false, we write multiple if(){...}else if(){...}………

   This structure is called as a ladder.

   ```
   if(condition){
        //statements1
   }
   else if(condition2){
        //statements2
   }
   else if(…)………
   ```

4. nested if:

   When there to be checked multiple conditions after prior condition returns true OR when the code to be executed requires multiple conditions to be true, we use nested if, that means if under if under if and so on…

   ```
   if(condition){
        if(condition2){
             if(condition3){
                  //statement will only execute when above all
   conditions return true;
             }
        }
   ```

```
}
```

5. switch case:

An alternate to of statements. It is used to perform different actions based on different conditions. It evaluates an expression and matches its value against a series of case clauses. When a match is found, the code associated with that case is executed. If no match is found, the default clause is executed, if it is provided.

```
switch(expression){
      case <value>:
            //statements
            break;
      case <value2>:
            //statements
            break;
      .
      .
      .
      case <valuen>:
            //statements
            break;
      default:
            //statements
}
```

II] LOOPING STATEMENTS:

- while
- do-while
- for
- for in
- for of

1. while:

While loop is used to execute a block of code repeatedly till the condition specified is true. There's a need to update the variable used in condition so as to make condition false at some point. Otherwise, it will be an infinite loop.

```
while(condition){
          //statements;
}
```

2. do-while:

Do-while loop is an exit-controlled loop unlike while which is entry-controlled. Do-while at least execute the block once even if the condition is false. It first executes, then check the condition.

```
do{
        //statements;
}while(condition);
```

3. for:

A combined statement of initialization, condition and update. It executes until the condition is true. Condition and update field can be left vacant.

```
for(initialize; condition; update){
        //statements;
}
```

4. for in:

A variant of for loop. It's used to iterate through keys of an object or an array without the need of knowing number of keys an object has or length of array.

```
const obj = {
          key1:value,
          key2:value,
};
for(let x in obj){
        console.log(x); //it will print keys of object obj
}
```

5. for of:

A variant of for loop. It is used to iterate through String characters and arrays.

```
let cities = ["Mumbai", "Delhi", "Bangalore"];
for(let city of cities){
        console.log(city); //it will print all values if array cities
}
```

III] BRANCHING STATEMENTS:

1. continue:

Used to skip an iteration; often written under a decision making statement.

2. break:

Used to break/exit the loop before the exit condition is met; often written under a decision making statement.

WARNING: Do not run any infinite loop code in JS, it may lag your browser!