SLIIT

Discover Your Future

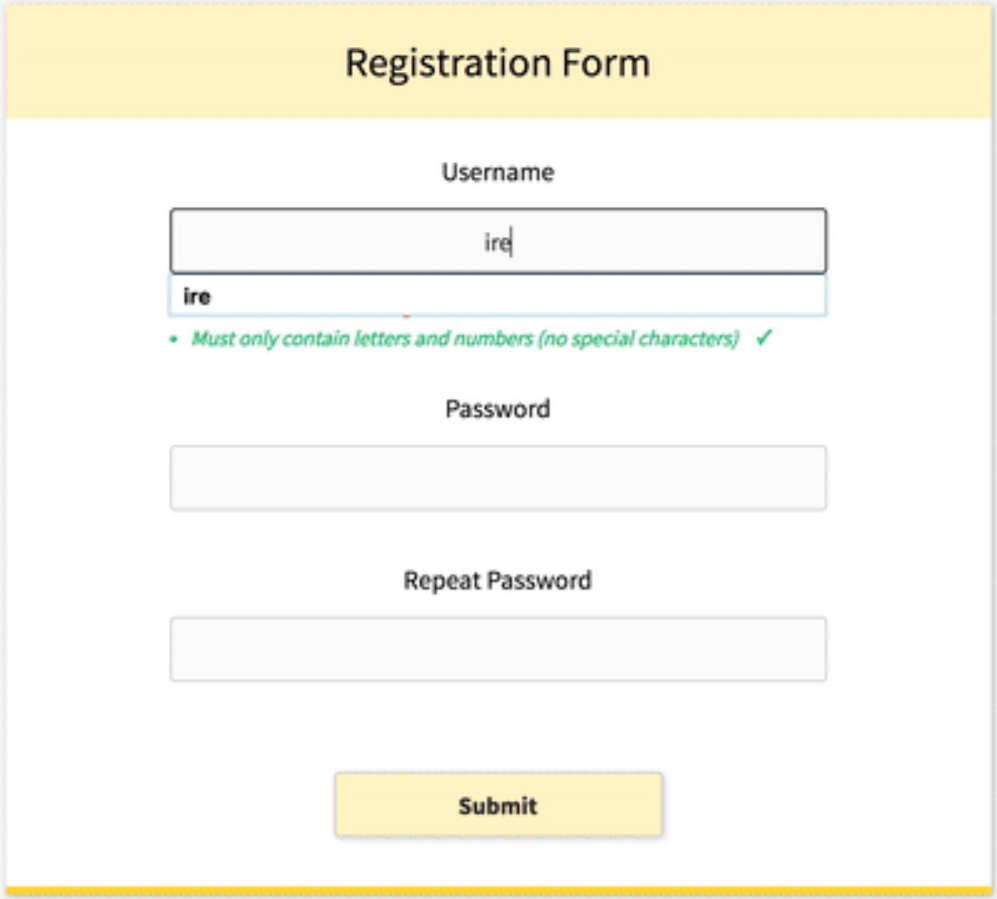
IT1100 Internet and Web technologies

Lecture 04

JavaScript

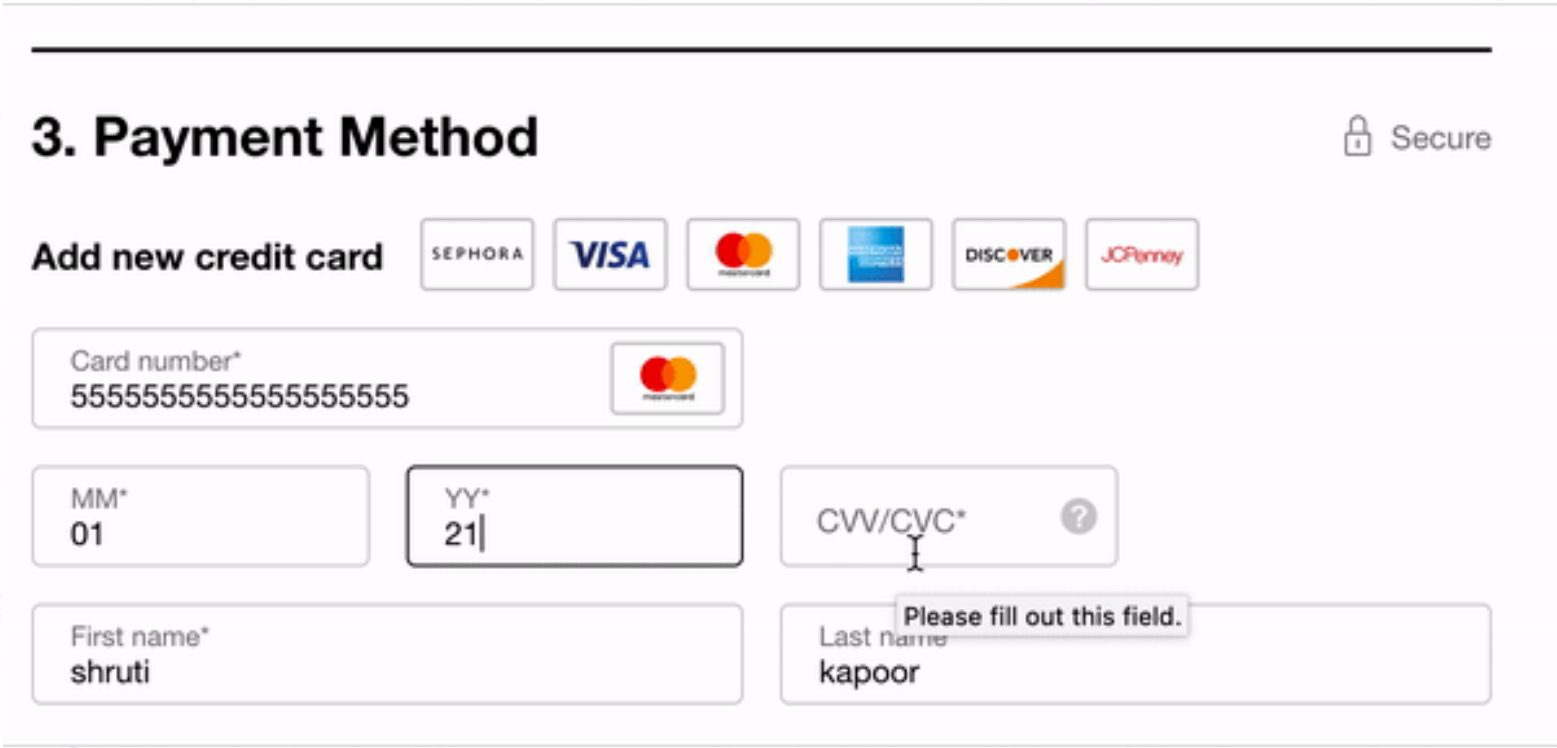
Why JavaScript…….

Why Javascript.....



Why JavaScript…….

Why JavaScript.....



Content – **JavaScript**

Today

1. Introduction to the JavaScript
2. Variables in JS
3. Operators in JS
4. Control structures in JS

Next weeks

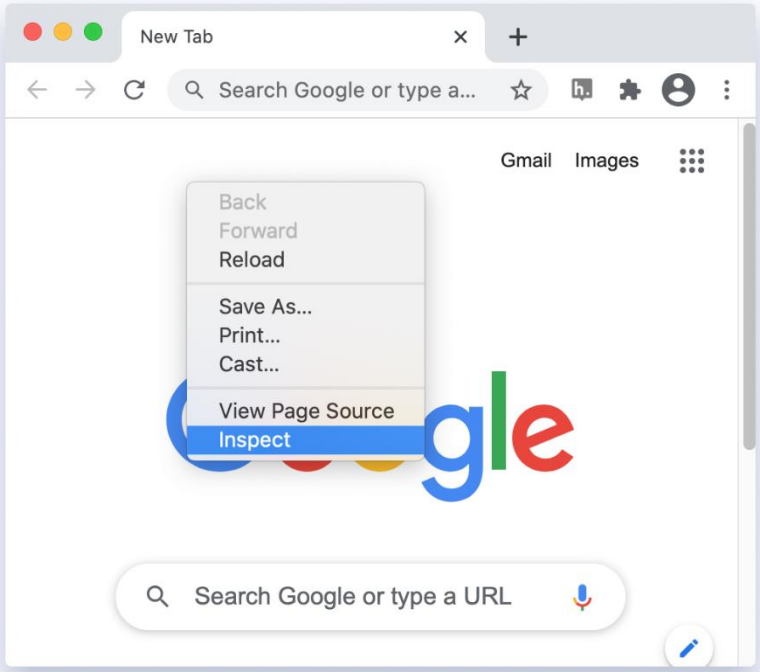
1. Arrays
2. Functions
3. DOM
4. Functions and Forms

Content – **JavaScript**

1. Introduction to the JavaScript
2. Variables in JS
3. Operators in JS
4. Control structures in JS
5. Arrays

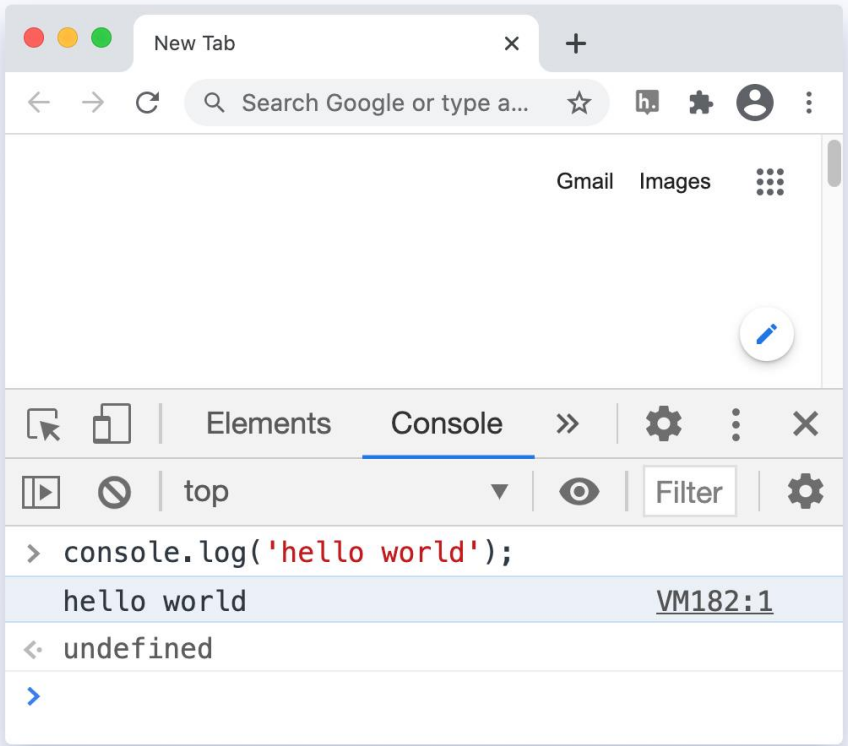
Using Console Tab of Web Browsers

- All the popular web browsers have built-in JavaScript engines. Hence, you can run JavaScript on a browser. To run JavaScript on a browser,
- Open your favorite browser (here we use Google Chrome).
- Open the developer tools by right clicking on an empty area and select Inspect. Shortcut: F12.



Using Console Tab of Web Browsers

- On the developer tools, go to the console tab. Then, write JavaScript code and press enter to run the code.



1. Introduction to the JavaScript

Ways of using JavaScript

1. Internal
   - Scripts in <body> (inline JS)
   - Scripts in <head> (Internal script)
   - Scripts in both <head> and <body>

2. External script files

1. Introduction to the JavaScript

Ways of using JavaScript

Internal script

- JavaScript is embedded into the HTML document using the script element

```
<script >

//JS code

</script>
```

1. Introduction to the JavaScript

Ways of using JavaScript

```
<!DOCTYPE html>
<html>
    <head>
    <style>
        h3{
        color:#FFC300;
        }
    </style>
    <title>JavaScript Page</title>

        <script type="text/javascript">
            document.write('<h3>This is from internal Java script</h3>');
            </script>

    </head>
<body>
        <h3>Hello world</h3>
</body>
</html>
```

output

This is from internal Java script

Hello world

1. Introduction to the JavaScript

Ways of using JavaScript

```
<!DOCTYPE html>
<html>
    <head>
    <style>
        h3{
        color:red;
        }
    </style>
    <title>JavaScript Page</title>
    </head>
<body>
    <script type="text/javascript">
            document.write('<h3>This is from internal Java script</h3>');
            </script>
     <h3>Hello world</h3>
</body>
</html>
```

output

This is from internal Java script

Hello world

1. Introduction to the JavaScript

Ways of using JavaScript

External script files

- The JS file should use the extension .js
- External file is linked to the web page in head using the script element
- The script element uses the **src** attribute to specify the URL of the source JS file
  - ***src***=*"<Location>/<FileName>.js"*

1. Introduction to the JavaScript
Ways of using JavaScript

External script files

```
<head>
    <script src="../ClientScripts/MainJS.js"></script>
</head>
```

- The same JS file can be linked to multiple pages

External JavaScript

external_java_script.html

```
1  <html>
2      <head>
3      <title>JavaScript Page</title>
4          <script type="text/javascript" src="ext.js">
5          </script>
6      </head>
7  <body>
8          <h1>Hello world</h1>
9  </body>
10 </html>
```

external.js

```
1  document.write('This is from External Javascript');
```

output

This is from internal JSscript

Hello world

2. Variables in JS

Data types

JS is a weakly typed language

The keyword **var** **and** **let** is used to declare a variable

1. Numerical
   - Integers – 1, 2, 3, -56, -135, 3464
   - Floating point/Decimal – 34.46, -65.135
2. Strings
   - Single characters – “a”, “b”, “c”, “2”, “7”
   - Multiple characters – “abc”, “12/04/2012”, “34”
3. Boolean – true / false
4. Null
5. Undefined

2. Variables in JS

Data types

Note:

- JavaScript does not make a distinction between integer values and floating point values.
- All numbers in JavaScript are represented as floating-point values.

2. Variables in JS

Variable declaration

Standars for the variable name

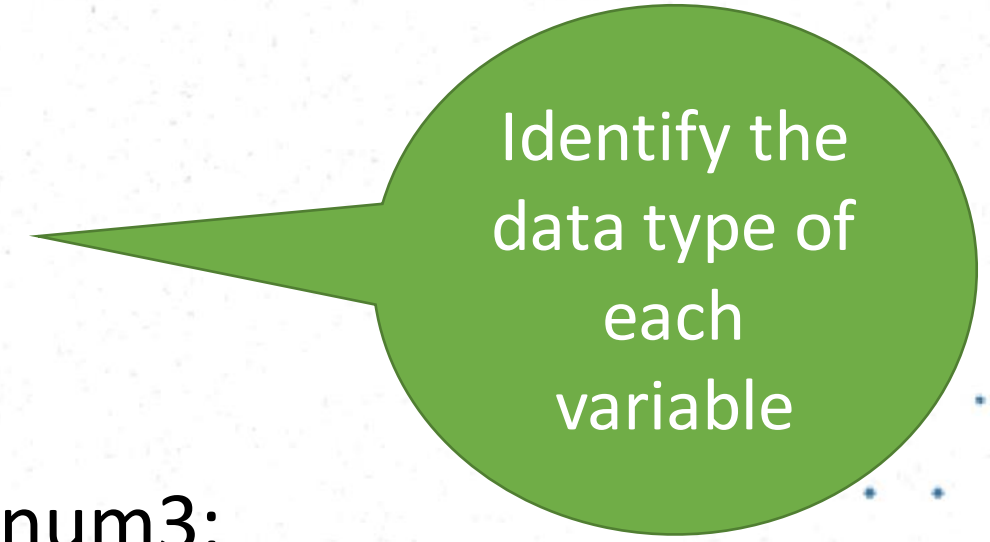
- You should not use any of the JavaScript **reserved keywords** as a variable name.
- No spaces
- Meaningful
- Use camel case

abstract	else	Instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

2. Variables in JS

Variable declaration

- Examples
  - var age;
  - var smallNumber;
  - var initial
  - var name
  - var isPassed;
  - var num1, num2, num3;



2. Variables in JS

Variable Initialization

Initialize the variables

```
var age = 20;
var height = 5.5;
var initial = "K";
var name = "Kamal";
var isPassed = true;
```

2. Variables in JS

Variable declaration

Assign values to the variables

```
age = 20;
height = 5.5;
initial = "K";
name = "Kamal";
isPassed = true;
```

2. Variables in JS

Java Script Constants

JavaScript Constants

The const keyword was also introduced in the **ES6(ES2015)** version to create constants. For example,

```
const x = 5; x = 10; // Error! constant cannot be changed. console.log(x)
```

Also, you cannot declare a constant without initializing it. For example

```
const x; // Error! Missing initializer in const declaration. x = 5; console.log(x)
```

2. Variables in JS
Variable declaration

Read and use the variable value

```
var age = 20; //Declare and initialize the variable
document.write(age);


age = 25; //Assign a new value to the variable
document.write("<br>Modified age = " + age);
```

2. Variables in JS
Weak typing?

```
<html>
  <head>
  <title>JavaScript Page</title>
  </head>
        <body>
            <script type="text/javascript">
                document.write("4"/3);
                document.write("<br>");
                document.write("5" +5);
                document.write("<br>");
                document.write("5"- 3);
                document.write("<br>");
                document.write("5"* "5");
                document.write("<br>");
                document.write(4*3);
                document.write("<br>");
                document.write(5* "5");

    </script>
  <h1>Hello world</h1>
</body>
</html>
```

output

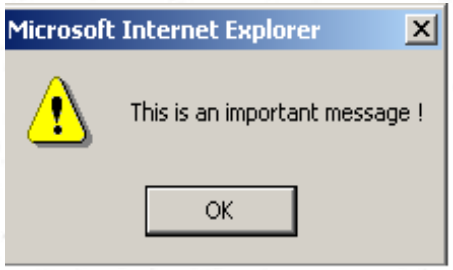
1.3333333333333333
55
2
25
12
25

Hello world

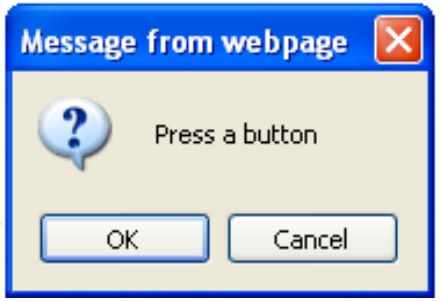
JavaScript popup boxes

- JavaScript has three kinds of popup boxes:

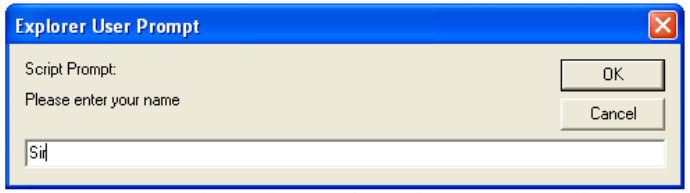
  - Alert box
    - alert (“This is an important message !”);



  - Confirm box
    - var response=confirm("Press a button");



  - Prompt box
    - var name=prompt("enter your name",“Sir");



3. Operators in JS

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators

3. Operators in JS
Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

3. Operators in JS
Assignment Operators

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

3. Operators in JS
Comparison Operators

Operator	Description	Example
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

http://www.w3schools.com/js/js_operators.asp

3. Operators in JS
Logical Operators

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

http://www.w3schools.com/js/js_operators.asp

4. Control Structures in JS

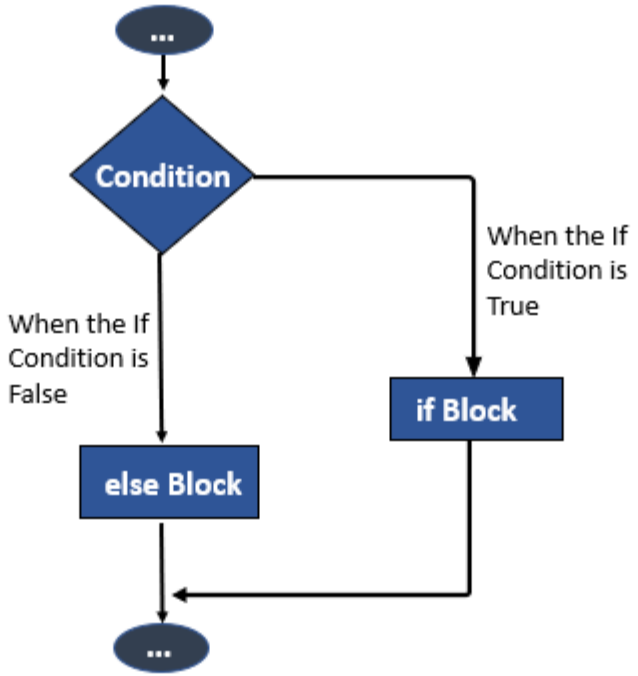
1. Sequence
2. Selection / Branching
   - Simple if-else
   - If-else ladder
   - Nested if-else
   - switch
3. Repetition / Iteration / Looping
   - While loop
   - For loop

4. Control Structures in JS
Selection / Branching

- Used to divide the algorithm execution path into branches based on conditions
- Conditions produce Boolean results
- **If** the condition is true – we can do something
- **Else** we can do some other thing

4. Control Structures in JS Selection / Branching

```
if (<Condition>)
{
        //Do something
}
else
{
        //Do some other thing
}
```



- Else is optional

4. Control Structures in JS Selection / Branching

Simple if-else

- User enters the mark for Maths.
  - If the mark is greater than or equals 50 then display a message “Pass”
  - Else display a message “Fail”

4. Control Structures in JS
4.1. Selection / Branching

Simple if-else

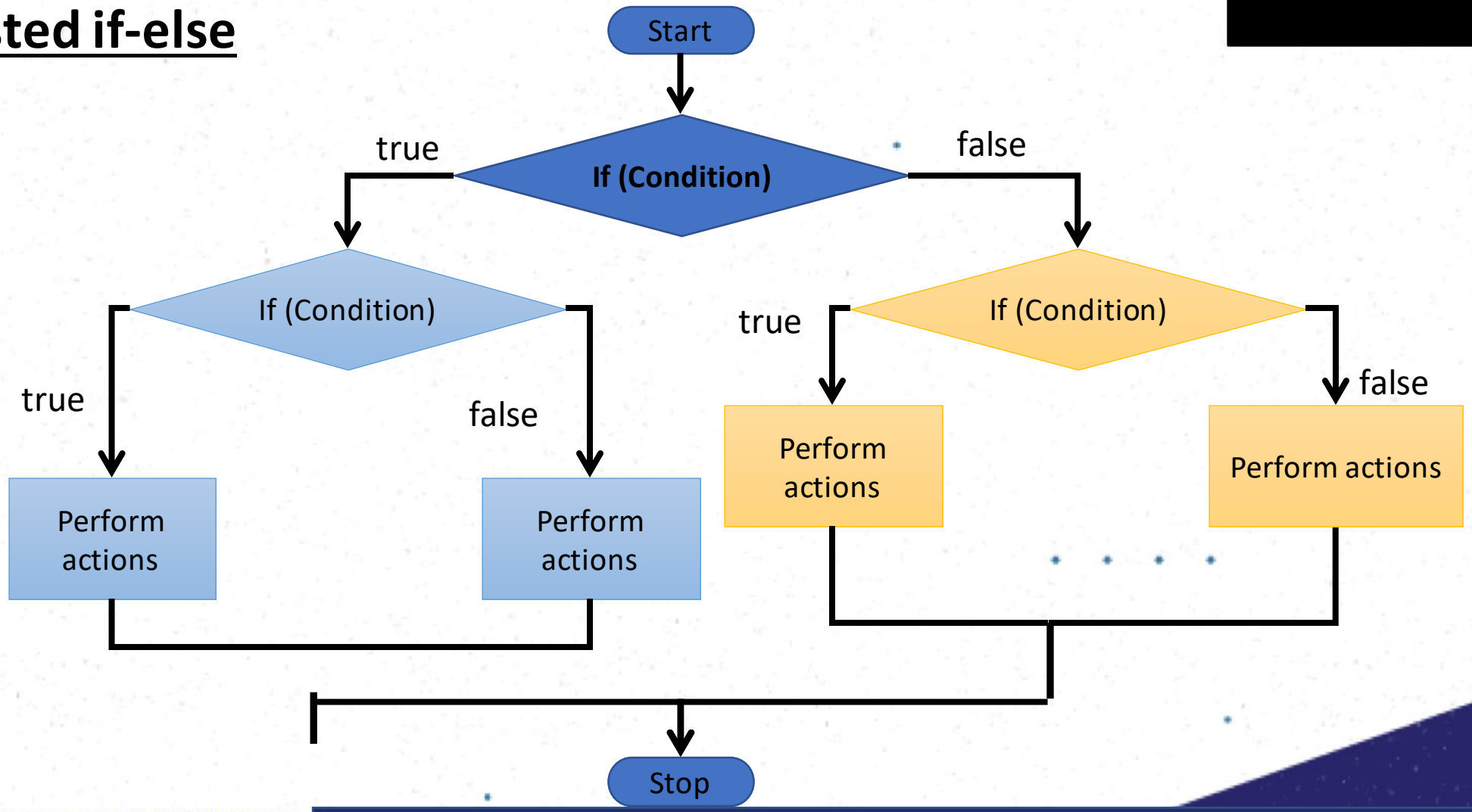
```
if(mark >= 50)
{
  document.write ("Pass");
}
else
{
        document.write ("Fail");
}
```

4. Control Structures in JS
4.1. Selection / Branching

```
// check is the number is positive or negative/zero
 const number = prompt("Enter a number: ");
// check if number is greater than 0
 if (number > 0) {
console.log("The number is positive");
 } // if number is not greater than 0
 else {
console.log("The number is either a negative number or 0");
 }
console.log("The if...else statement is easy");
```

4. Control Structures in JS
Selection / Branching

Nested if-else



4. Control Structures in JS
Selection / Branching

Nested if-else

```
if(<Condition1>)
{
        if(<Condition2>)  { //Actions  }
        else  { //Actions  }
}
else
{
        if(<Condition3>)  { //Actions  }
        else  { //Actions   }
}
```

4. Control Structures in JS Selection / Branching

- Are these equivalent?

```
if ( age < 12 ) {
   entry = "free";
} else if ( age < 18 ) {
   entry = "£10";
} else {
   entry = "£20";
}
```

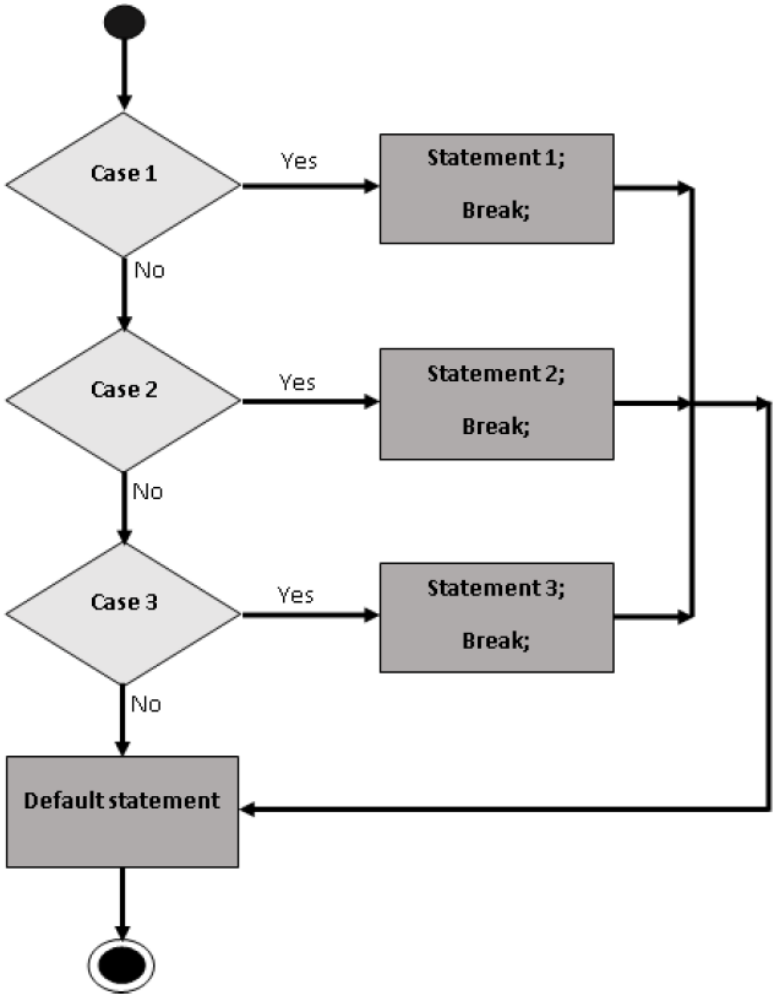
```
if ( age < 18 ) {
   entry = "£10" ;
} else if ( age < 12 ) {
   entry = "free";
} else {
   entry = "£20";
}
```

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
    document.write("<b>Good morning</b>");
}
else if (time>=10 && time<16)
        {
            document.write("<b>Good day</b>");
        }
        else
        {
            document.write("<b>Hello World!</b>");
        }
</script>
<p>
This example demonstrates the if..else if...else statement.
</p>
</body>
</html>
```

output

4. Control Structures in JS
Selection / Branching

Switch




```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different
  from case 1 and 2
}
```

4. Control Structures in JS Selection / Branching

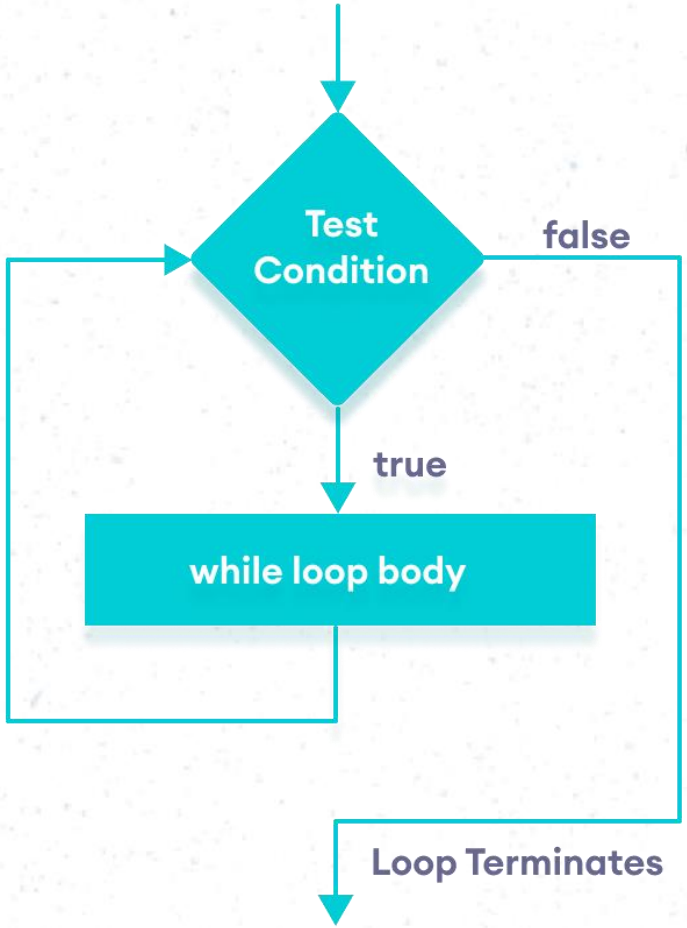
```
var grade="B";
switch (grade)
    {
    case "A":
        alert("Excellent");
        break;

    case "B":
        alert("Good");
        break;

    default:
        alert("Average");
        break;
}
```

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("<b>Finally  Friday</b>");
  break;
case 6:
  document.write("<b>Super  Saturday</b>");
  break;
case 0:
  document.write("<b>Sleepy  Sunday</b>");
  break;
default:
  document.write("<b>I'm  really looking forward to this weekend!</b>");
}
</script>
<p>This JavaScript will generate a different greeting based on what day it is. Note that Sunday=0,
   Monday=1, Tuesday=2, etc.</p>
</body>
</html>
```

SLIIT FACULTY

```
// program for a simple calculator
 let result;
// take the operator input
const operator = prompt('Enter operator ( either +, -, * or / ): ‘);
// take the operand input
const number1 = parseFloat(prompt('Enter first number: '));
const number2 = parseFloat(prompt('Enter second number: '));
switch(operator) {
case '+': result = number1 + number2;
console.log(`${number1} + ${number2} = ${result}`);
break;
case '-': result = number1 - number2;
console.log(`${number1} - ${number2} = ${result}`);
break; case '*': result = number1 * number2;
console.log(`${number1} * ${number2} = ${result}`);
 break; case '/': result = number1 / number2;
console.log(`${number1} / ${number2} = ${result}`);
break; default: console.log('Invalid operator’);
break; }
```

4. Control Structures in JS
Loops – while loop



- The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is **true**.
- Once the expression becomes **false**, the loop terminates.

4. Control Structures in JS
Loops – while loop

```
<html>
<body>
 <table border="5">
  <script>
   var i=1;
     while (i<=6)
       {
            document.write("<tr>”);
            document.write("<td>col 1 row " + i + "</td>")
            document.write("<td>col 2 row " + i + “</td>");
            document.write("</tr>");
           i++;
           }
  </script>
 </table>
</body>
</html>
```

output

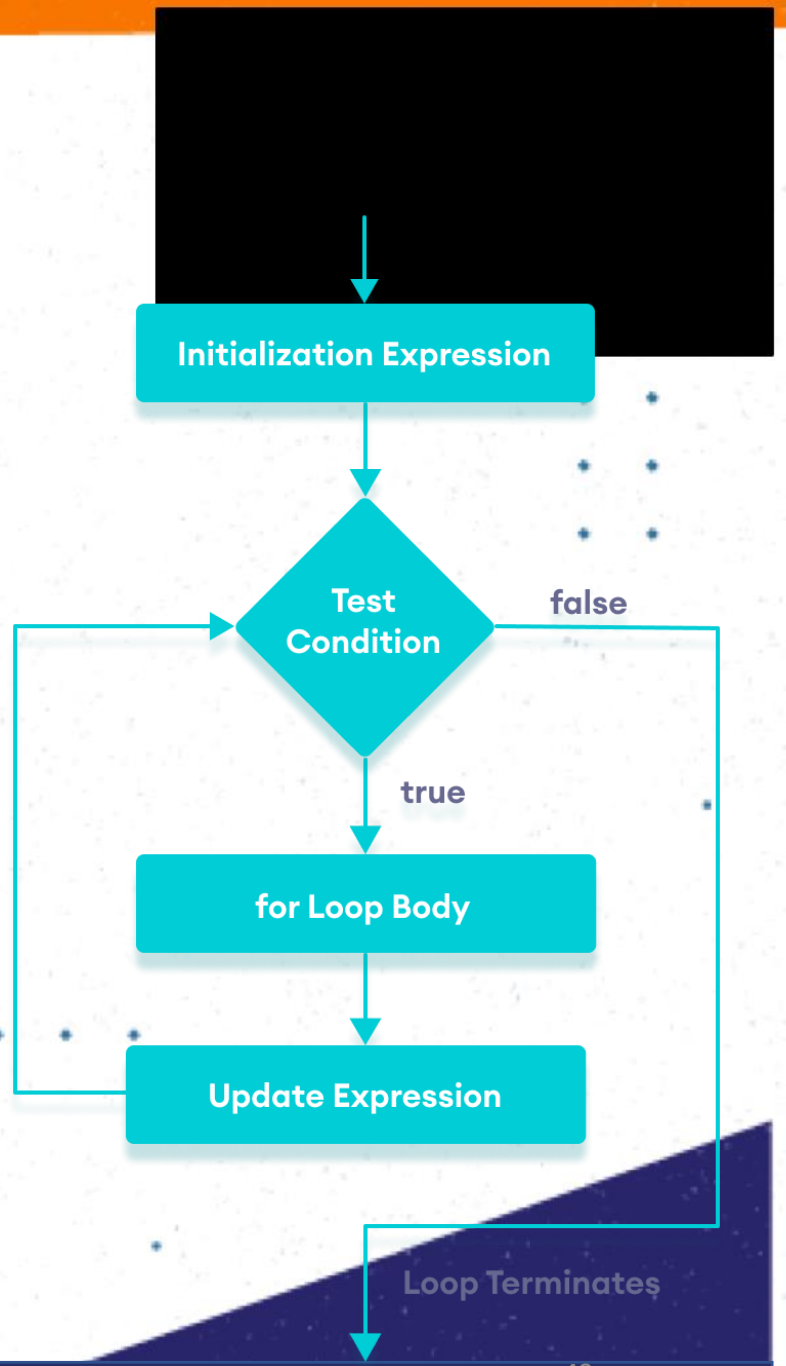
col 1 row 1	col 2 row1
col 1 row 2	col 2 row2
col 1 row 3	col 2 row3
col 1 row 4	col 2 row4
col 1 row 5	col 2 row5
col 1 row 6	col 2 row6

4. Control Structures in JS
Loops – while loop - Console

```
// program to find the sum of positive numbers //
 if the user enters a negative numbers, the loop ends //
 the negative number entered is not added to sum
let sum = 0; // take input from the user
let number = parseInt(prompt('Enter a number: '));
while(number >= 0) {
// add all positive numbers
sum += number;
// take input again if the number is positive
number = parseInt(prompt('Enter a number: '));
} // display the sum
console.log(`The sum is ${sum}.`);
```

4. Control Structures in JS
Loops – for loop

```
for (var=startvalue; var<=endvalue; var=var+increment)
{
        //code to be executed
}
```



4. Control Structures in JS
Loops – for loop

```
<html>
<body>
 <table border="5">
  <script>
            for (i=0;i<=6;i++)
        {
         document.write("<tr>");
         document.write("<td>col 1 row " + i + "</td>")
         document.write("<td>col 2 row " + i + "</td>");
         document.write("</tr>");
         //i++;
        }
  </script>
 </table>
</body>
</html>
```

output

col 1 row 0	col 2 row 0
col 1 row 1	col 2 row 1
col 1 row 2	col 2 row 2
col 1 row 3	col 2 row 3
col 1 row 4	col 2 row 4
col 1 row 5	col 2 row 5
col 1 row 6	col 2 row 6

4. Control Structures in JS
Loops – for loop - Console

```
// program to display the sum of natural numbers
let sum = 0;
const n = 100
 // looping from i = 1 to n // in each iteration, i is increased by 1
 for (let i = 1; i <= n; i++) {
sum += i; // sum = sum + I
 }
console.log('sum:', sum);
```

4. Control Structures in JS Loops

- **The break Statement**
  - The break statement will break the loop and continue executing the code that follows the loop (if any).

- **The continue Statement**
  - The continue statement will break the current loop and continue with the next iteration.

4. Control Structures in JS
Loops - Break

```
<!DOCTYPE html>
<html>
<body>
<script
for (i=0;i<=10;i++)
{
  if (i==8)
  {
    break;
  }
  document.write("The number is " + i);
  document.write("<br />");
}
document.write("Break....");
</script>
</body>
</html>
```

output

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
Break....

4. Control Structures in JS
Loops - Continue

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=0
for (i=0;i<=10;i++)
{
   if (i==3)
   {
      continue;
   }
  document.write("The number is " + i);
  document.write("<br />");
}
</script>
</body>
</html>
```

output

The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
The number is 10

Content – **JavaScript**

Today We Learnt

1. Introduction to the JavaScript
2. Variables in JS
3. Operators in JS
4. Control structures in JS

Next weeks

1. Arrays
2. Functions
3. DOM
4. Functions and Forms