SLIIT
Discover Your Future

[redacted]

IT1100 Internet and Web technologies

Lecture 05

JavaScript – Part II

SLIIT
FACULTY OF COMPUTING

JavaScript Strings

[redacted]

- JavaScript strings are used for storing and manipulating text.
- zero or more characters written inside quotes , using single or double quate.
- var Description= “That's alright"
  var Description = "He is called ‘Kuma’"
  var Description = 'He is called “Mahela“’
- You can use quotes inside a string, as long as they don't match the quotes surrounding the String

Length of a String

var txt = "lets watch legend playing";

var Length=txt.length;

JavaScript String Methods

[redacted]

- The indexOf() method returns the index of (the position of) the first occurrence
- The lastIndexOf() method returns the index of the last occurrence of a specified text in a string
- Both indexOf(), and lastIndexOf() return -1 if the text is not found

JavaScript String Methods

[redacted]

var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
	var str = "Please locate where 'locate' occurs!";
	var pos = str.lastIndexOf("locate");

var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("John");
	var str = "Please locate where 'locate' occurs!";
	var pos = str.lastIndexOf("locate", 15);

JavaScript Numerical Methods

[redacted]

Number(" 10 "); // returns 10
Number("10.33"); // returns 10.33
Number("10,33");
	parseInt("10"); // returns 10
	parseInt("10.33"); // returns 10
	parseInt("10 20 30"); // returns 10
	parseInt("10 years"); // returns 10
	parseInt("years 10"); // returns NaN

parseFloat("10.33"); // returns 10.33
parseFloat("10 20 30"); // returns 10
parseFloat("10 years"); // returns 10
parseFloat("years 10"); // returns NaN
	Number. MAX_VALUE returns the largest possible number in JavaScript.
	Number.MIN_VALUE;

JavaScript Arrays

[redacted]

- An array is a special variable, which can hold more than one value at a time.
  - var *array_name* = [*item1*, *item2*, ...];
  - var cars = [“Toyota", "Volvo"];

  How to insert new elements
  - Cars[2]=“BMW”;

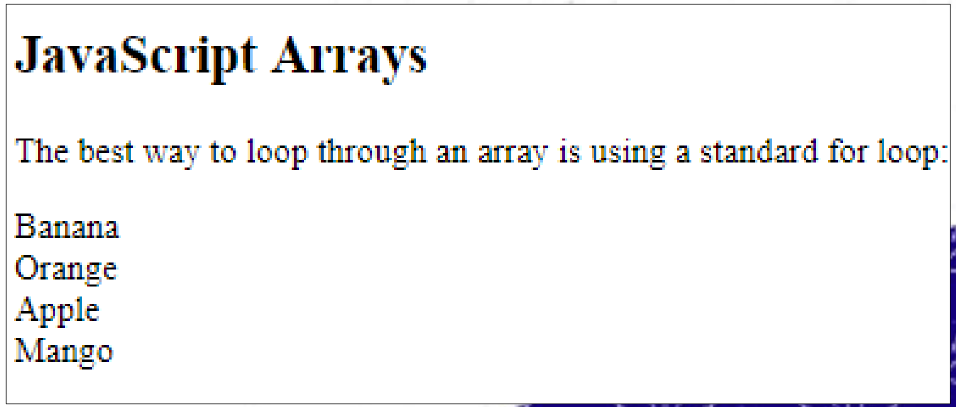
  How to display element ant it’s value
  - Document.write(Cars[0]);

SLIIT FACULTY OF COMPUTING

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>The best way to loop through an array is using a standard for loop:</p>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;

for (i = 0; i < fLen; i++) {
  document.write(fruits[i]+"</br>");
}
</script>
</body>
</html>
```

[redacted]

output

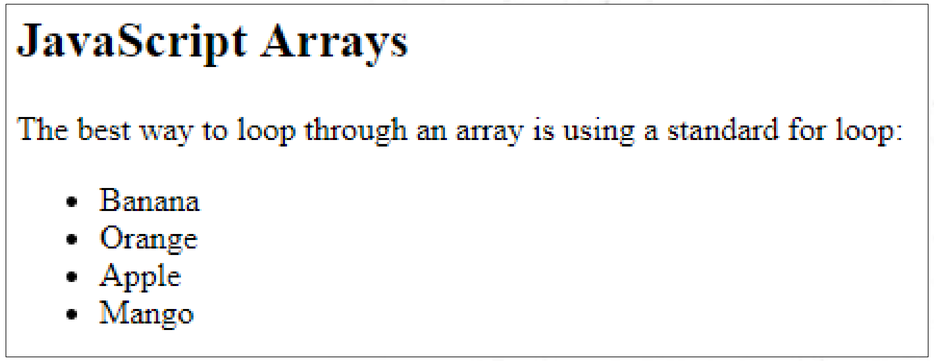


```
JavaScript Arrays

The best way to loop through an array is using a standard for loop:

Banana
Orange
Apple
Mango
```

[redacted]

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Arrays</h2>
<p>The best way to loop through an array is using a standard for loop:</p>
<p id="demo"></p>
<script>
var fruits, text, fLen, i;
fruits = ["Banana", "Orange", "Apple", "Mango"];
fLen = fruits.length;
text = "<ul>";
for (i = 0; i < fLen; i++) {
  text += "<li>" + fruits[i] + "</li>";
}
text += "</ul>";
document.getElementById("demo").innerHTML = text;
</script>
</body>
</html>
```

output

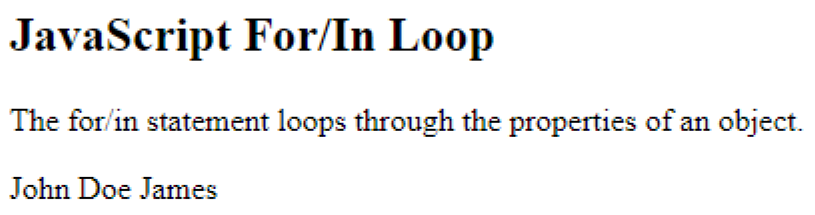


```
JavaScript Arrays

The best way to loop through an array is using a standard for loop:

  • Banana
  • Orange
  • Apple
  • Mango
```

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript For/In Loop</h2>
<p>The for/in statement loops through the properties of an
object.</p>

<script>
var txt = "";
var person = ["John","Doe","James"];
var x;
for (x in person) {
  txt = txt + person[x] + " ";
}
document.write(txt);
</script>
</body>
</html>
```

[redacted]

output



JavaScript For/In Loop

The for/in statement loops through the properties of an object.

John Doe James

FUNCTIONS

[redacted]

- A function is a group of reusable code which can be called anywhere in your program.

- This eliminates the need of writing the same code again and again.

- It helps programmers in writing modular codes.

- Functions allow a programmer to divide a big program into a number of small and manageable functions.

[redacted]

FUNCTIONS - Function Definition

- Before we use a function, we need to define it.
- The most common way to define a function in JavaScript is by using the **function** keyword, followed by a unique function name, a list of parameters (that might be empty), and a statement block surrounded by curly braces.

```
<script type="text/javascript">
function function_name (parameter-list)
{
      statement(s)
}
</script>
```

[redacted]

FUNCTIONS - Calling a Function

- To invoke a function somewhere later in the script, you would simply need to write the name of that function as shown in the code.

```
<html>
<head>
<script type="text/javascript">
function sayHello()
{
   document.write ("Hello  there!");
}
</script>
</head>
<body>
<script type="text/javascript">
    sayHello();
</script>
</body>
</html>
```

Output

```
Hello there!
```

FUNCTIONS - **Function Parameters**

- Till now, we have seen functions without parameters.
- But there is a facility to pass different parameters while calling a function.
- These passed parameters can be captured inside the function
- Any manipulation can be done over those parameters.
- A function can take multiple parameters separated by comma.

FUNCTIONS - **Function Parameters**

[redacted]

```
<html>
<head>
<script type="text/javascript">
function sayHello(name, age)
{
document.write (name + " is " + age + " years old.");
}
</script></script>
</head>
<body>
<script type="text/javascript">
    sayHello('Zara', 7);
</script>
</body>
</html>
```

Output

```
Zara is 7 years old.
```

FUNCTIONS - **The return Statement**

[redacted]

- A JavaScript function can have an optional **return** statement.
- This is required if you want to return a value from a function.
- <u>**This statement should be the last statement in a function**</u>.
- For example, you can pass two numbers in a function and then you can expect the function to return their multiplication in your calling program.

```
<html>
<head>
<script type="text/javascript">
function concatenate(first, last)
{
var full;
full = first + last;
return full;
}
function secondFunction()
{
var result;
result = concatenate('Zara ', 'Ali  Khan');
document.write (result );
}
</script>
</head>
<body>
<script type="text/javascript">
secondFunction();
</script>
</body>
</html>
```

The return Stateme[redacted]

Output

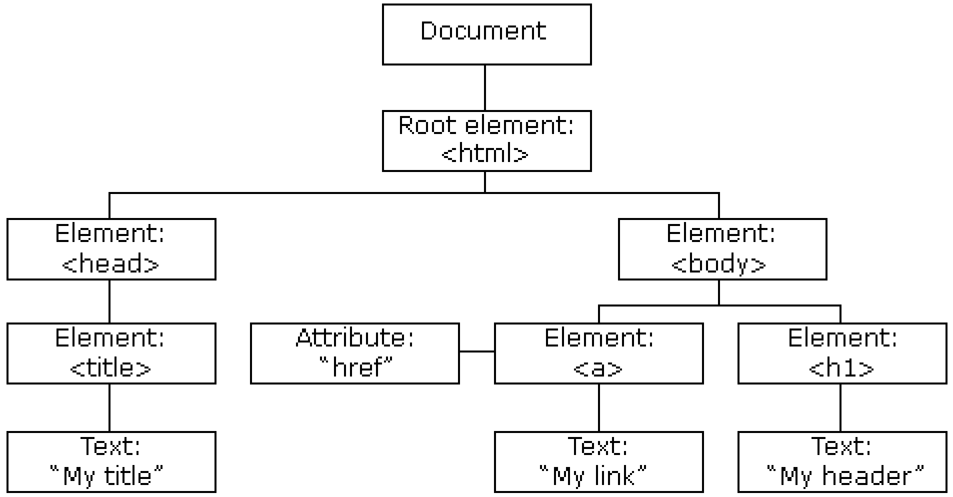
```
Zara Ali Khan
```

5. Document Object Methods

[redacted]

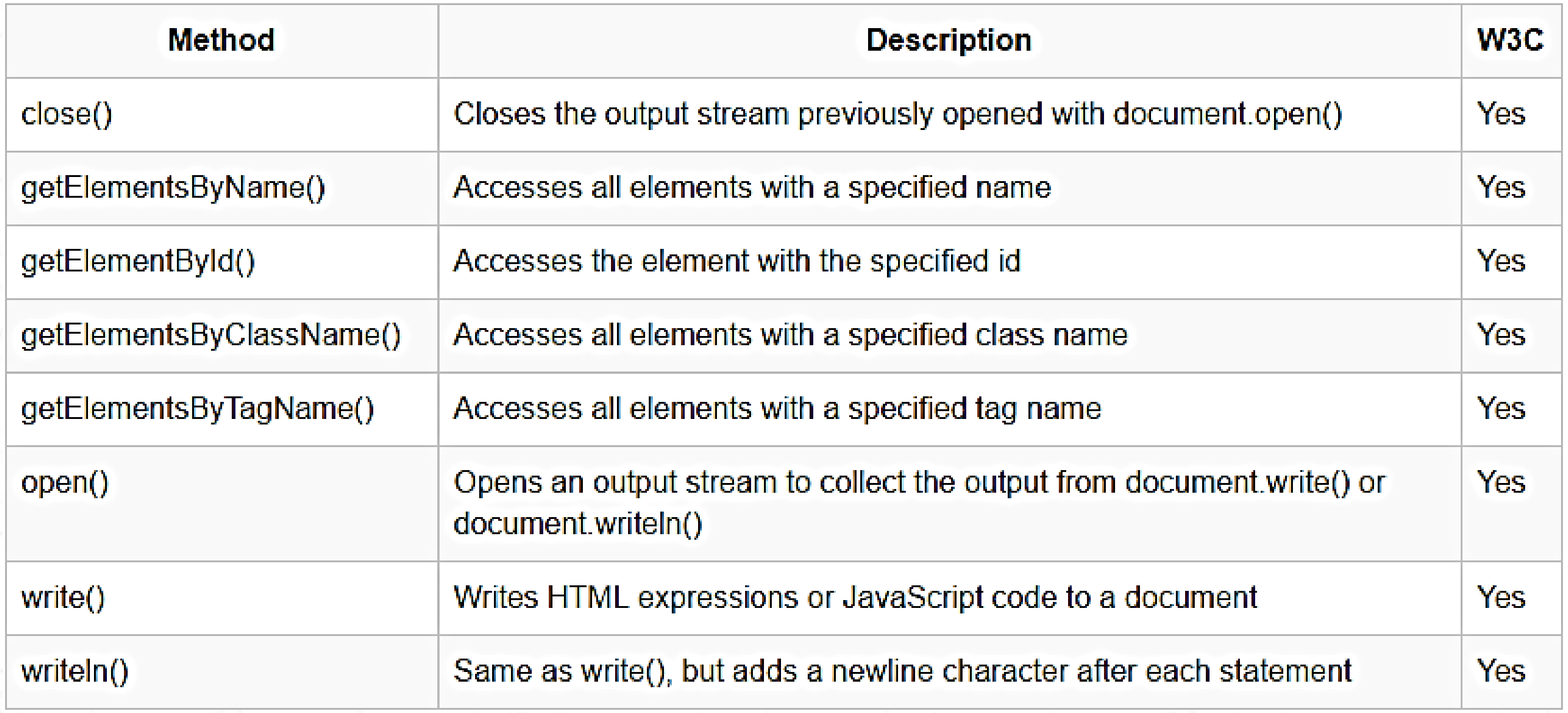
When a web page is loaded, the browser creates a Document Object Model of the page.

The HTML DOM model is constructed as a tree of Objects:



[redacted]

5. Document Object Methods



Method	Description	W3C
close()	Closes the output stream previously opened with document.open()	Yes
getElementsByName()	Accesses all elements with a specified name	Yes
getElementById()	Accesses the element with the specified id	Yes
getElementsByClassName()	Accesses all elements with a specified class name	Yes
getElementsByTagName()	Accesses all elements with a specified tag name	Yes
open()	Opens an output stream to collect the output from document.write() or document.writeln()	Yes
write()	Writes HTML expressions or JavaScript code to a document	Yes
writeln()	Same as write(), but adds a newline character after each statement	Yes

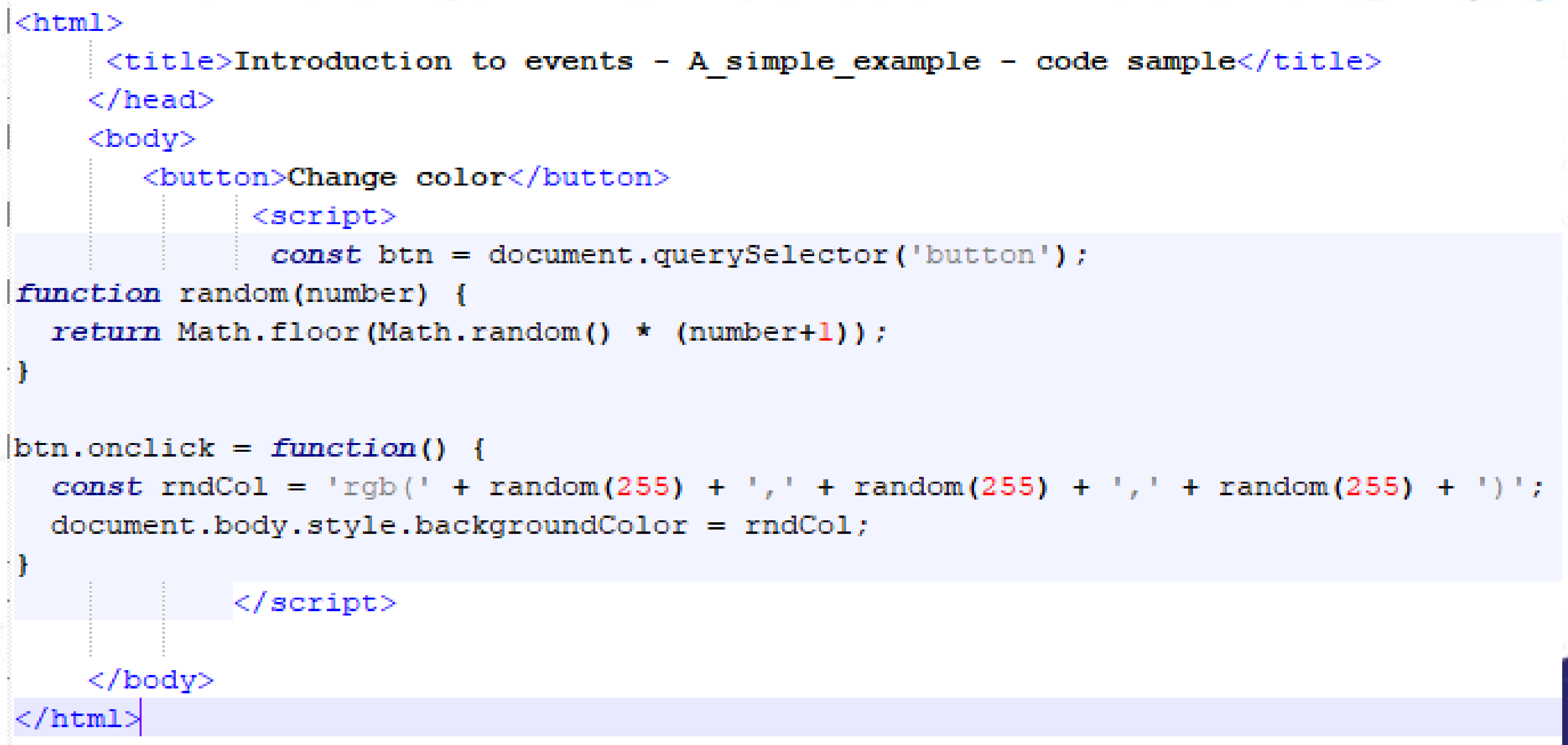
5. DOM API

[redacted]

```
<form>
        <input type=“text” id=“txtName”>
        <div id=“divOutput”></div>
</form>
----------------------------------------------------
//Read the value
var name = document.getElementById(" txtName ").value;

//Display output
document.getElementById(" divOutput").innerHTML = “Hello “+name;
```

[redacted]

querySelector() …… method returns the first element that matches a specified CSS selector(s)



```
<html>
     <title>Introduction to events - A_simple_example - code sample</title>
    </head>
    <body>
       <button>Change color</button>
             <script>
              const btn = document.querySelector('button');
function random(number) {
  return Math.floor(Math.random() * (number+1));
}

btn.onclick = function() {
  const rndCol = 'rgb(' + random(255) + ',' + random(255) + ',' + random(255) + ')';
  document.body.style.backgroundColor = rndCol;
}
            </script>

    </body>
</html>
```

6. Event Handling

[redacted]

- Event handling is used to implement responses for the user events
    - Click, type, select, drag and drop, etc…

Ex:

- Read form values and validate before submitting the form and display proper error messages

SLIIT FACULTY OF COMPUTING

6. Event Handling

[redacted]

- Event handlers are used to handle the events, when the events are triggered
- There 2 main ways of developing event handlers in JS
  1. DOM level 0 inline event handlers setting
  2. Event registration using the **addEventListener()** function

6. Event Handling
6.1 DOM level 0 inline event handlers

[redacted]

- HTML event attributes are used.
  - **onclick, onload, etc…**

EX: Find all the HTML attributes available for event handling

```
<button onclick="alert('Hello');">Try it</button>
```

6. Event Handling
6.1 DOM level 0 inline event handlers

[redacted]

- If there is more code to write, it is good to implement a function and call that function in the event handler

```
<button onclick="myFunction();">Try it</button>
<script>
function myFunction() {
        alert("Do whatever needed in this function");
}
</script>
```

6. Event Handling
6.1 DOM level 0 inline event handlers

[redacted]

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {

   document.getElementById("demo").innerHTML = document.getElementById("inTxt").value;
}
</script>
</head>
<body>

<input type="text" id="inTxt">

<button onclick="myFunction()">Click Me</button>

<p id="demo"></p>

</body>
</html>
```

6. Event Handling

6.2 Event registration using addEventListener()

[redacted]

- It is good the separate the JS from HTML as much as possible, towards increasing the modifiability.
- By using the **addEventListener()** function, we can eliminate the HTML event attributes

[redacted]

6. Event Handling

6.2 Event registration using addEventListener()

```
<button id="btnTest">Try it</button>
<script>
var btn = document.getElementById("btnTest");
btn.addEventListener("click", function() {
    alert("Do whatever needed in this function");
}
);
</script>
```

6. Event Handling
6.2 Event registration using addEventListener()

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>

<input type="text" id="inTxt">

<button id="myBtn">Click Me</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").addEventListener("click", function(){
    document.getElementById("demo").innerHTML = document.getElementById("inTxt").value;
});
</script>

</body>
</html>
```

[redacted]

JS summary

1. JavaScript Arrays
2. String and Numerical methods
3. DOM API
4. Event handling