CS 207: Applied Database Practicum
Week 2

Varun Dutt

School of Computing and Electrical Engineering
School of Humanities and Social Sciences
Indian Institute of Technology Mandi, India

Scaling the Heights

JavaScript

- JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g. having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of javascript such as Node.Js (which is discussed later).
- JavaScript contains a standard library of objects, such as Array, Date, and Math, and a core set of language elements such as operators, control structures, and statements.
- JavaScript is a interpreted programming language.

JavaScript - Syntax and Placement

- JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>**.
- JavaScript code can be placed anywhere but it is usually placed inside **<head>** tags.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

```
<html>
<head>
<script type="text/javascript">
document.write("JavaScript is a simple language");
</script>
</head>
<body>
</body>
</html>
```

Syntax and Placement

- Semicolons are optional after each statement if these statements are written in separate lines.
- Commenting done is done in similar way to C/C++.
  by //…..
  and by /*.... */

Variables

- Datatypes -
  Numbers, Strings, Boolean
  Other datatypes - null and undefined, each defines a single value.
  JavaScript also supports composite datatype **object**.
- Variables are declared with var keyword
- Variable can hold a value of any datatype

Variables-Example

```
<script type="text/javascript">
      var name = "Ali";
      var money
      money = 2000.50
      document.write(money)
      document.write("<br>")     //Leave this for now

      //notice how javascript automatically changes the datatype of variable
      money = "Two hundred rupees"
      document.write(money)
</script>
```

Output:

```
2000.5
Two hundred rupees
```

Variables scope: Global, local

Composite Data-type-Example

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Objects</h2>

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

<script>
var person = {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
};

document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
</script>

</body>
</html>
```

JavaScript Objects

John is 50 years old.

Operators

Same operators as in C/C++

- Arithmetic operators - eg: +, -, *
- Comparison operators - eg: ==, !=
- Logical operators - eg: &&, ||
- Bitwise operators - &, | , ^
- Assignment operators - eg: =, +=
- Conditional operator - eg: ?:
- typeof operator: tells datatype of operand

Operators: Example

```
<!DOCTYPE html>
<html>
<body>

<h2>Operators</h2>

<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<p id="demo4"></p>
<p id="demo7"></p>
<p id="demo8"></p>

<script>
var x = 5;
var y = 2;
var z = x + y;
//Arithmetic operator
document.getElementById("demo1").innerHTML = z;
//Comparision operator
document.getElementById("demo2").innerHTML = x < y;
//Logical operator
document.getElementById("demo3").innerHTML = (x < 10 && y > 1) + "<br>" + (x < 10 && y < 1);
//Bitwise operator
document.getElementById("demo4").innerHTML = (x > 10 | y < 1)+ "<br>" + (x < 10 | y > 1);
//typeof operator
document.getElementById("demo7").innerHTML = typeof(x)
//Conditional operator
if (new Date().getHours() < 18) {
    document.getElementById("demo8").innerHTML = "Good day!";
    }
</script>

</body>
</html>
```

Operators

7

false

true
false

0
1

number

Good day!

Controls

- Follow same syntax and meaning as in C
- Use of conditional statements - if, if-else, if-else-else if
- Use of switch-case
- Use of for, while loop
- Use of break, continue within the loops
- Labels:
  Syntax:

```
label:
statements
break labelname;

continue labelname;
```

Controls:Example

```
<!DOCTYPE html>
<html>
<body>

<p>Display "Good Day!" 10 times  if the hour is less than 18:00 else Good Evening 1 time</p>

<p id="demo">Good Evening!</p>

<script>
var text = "";
var i = 0;
if (new Date().getHours() < 18) { //if
    while (i < 10) { //while
    text += "<br>Good Day " + i;
    i++;
    }
} else { //else
    text += "<br>Good Evening " + i;
}
document.getElementById("demo").innerHTML = text;

</script>

</body>
</html>
```

Display "Good Day!" 10 times if the hour is less than 18:00 else Good Evening 1 time

Good Day 0
Good Day 1
Good Day 2
Good Day 3
Good Day 4
Good Day 5
Good Day 6
Good Day 7
Good Day 8
Good Day 9

Functions

We define 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.

Eg:

```
<script type="text/javascript">
      function sayHello()
      {
         alert("Hello there");
      }
</script>
```

This function when invoked from outside by sayHello() displays an alert box.

Events

JavaScript's interaction with HTML is handled through events that occur when the user or the browser manipulates a page.

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples include events like pressing any key, closing a window, resizing a window, etc.

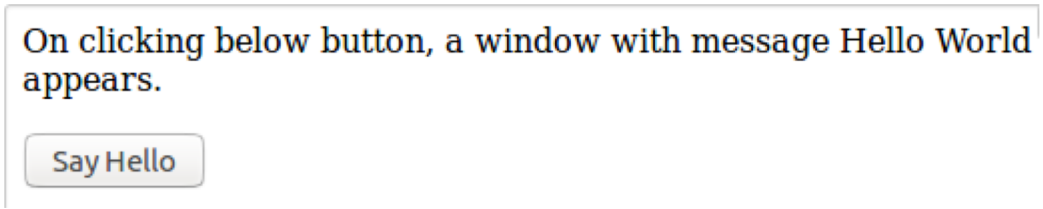
Developers can use these events to execute JavaScript coded responses, which cause buttons to close windows, messages to be displayed to users, data to be validated, and virtually any other type of response imaginable.

Events: Example

```
<html>
   <head>

      <script type="text/javascript">
            function sayHello() {
               alert("Hello World")
            }
      </script>

   </head>

   <body>
      <p>On clicking below button, a window with message Hello World appears.</p>

      <form>
         <input type="button" onclick="sayHello()" value="Say Hello" />
      </form>

   </body>
</html>
```

Above is an example of onclick event type

Output:



Events:Example

```
<!DOCTYPE html>
<html>
<body>

<p>When you submit the form, a function is triggered which alerts some text.</p>

<form action="/action_page.php" onsubmit="myFunction()">
  Enter name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

<script>
function myFunction() {
    alert("The form was submitted");
}
</script>

</body>
</html>
```

Above is the eg. of onsubmit event type - It is a event that occurs when you try to submit a form. You can put your form validation against this event type.If **myFunction()** function returns true, the form will be submitted, otherwise it will not submit the data.

HTML Document Object Model

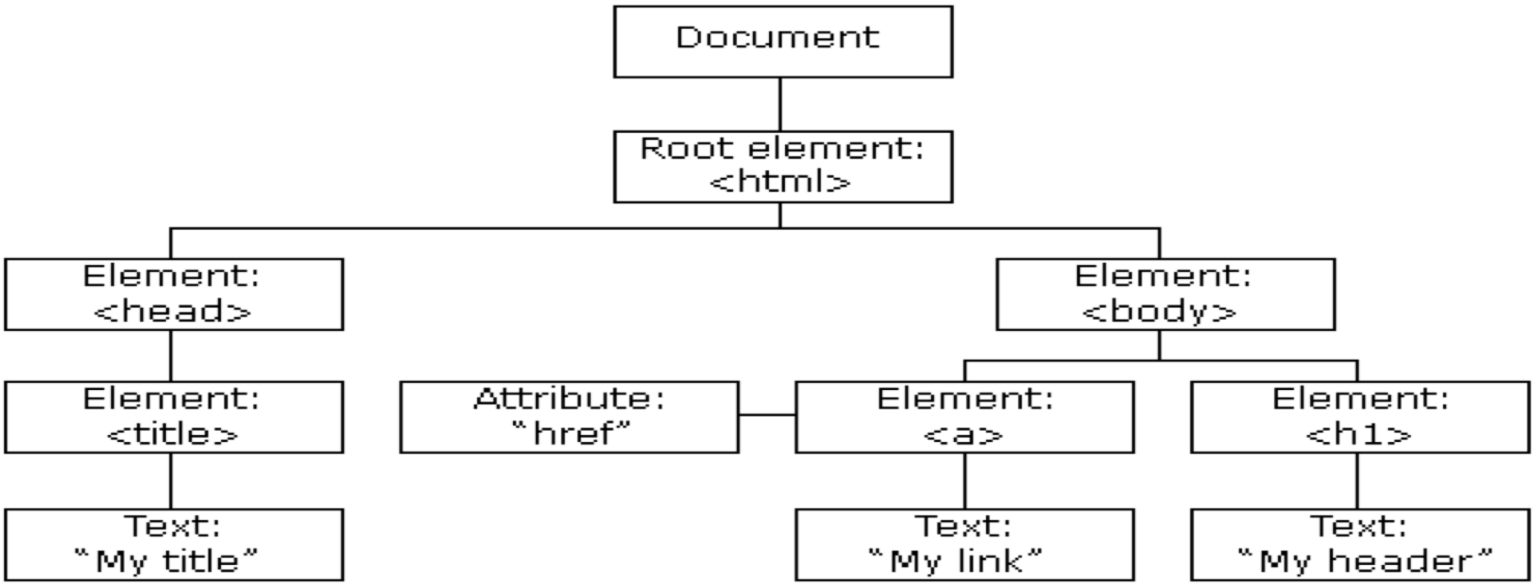
When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The DOM model is constructed as tree of objects.

The DOM represents the document as nodes and objects.

DOM

The DOM is an object-oriented representation of the web page, which can be modified with a scripting language such as JavaScript.



The HTML DOM Tree of Objects

Use of DOM in JS

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

DOM in JS

In the DOM, all HTML elements are defined as **objects**.
The programming interface is the properties and methods of each object.
A **property** is a value that you can get or set (like changing the content of an HTML element).
A **method** is an action you can do (like add or deleting an HTML element).
Below example shows how by the method **getElementById** we change the property, **innerHTML** of the element specified by the Id.

DOM in JS

```
<html>
<body>

<h2>My First Page</h2>

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

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

Output:

Examples of Methods used by JS

If you want to access any element in an HTML page, you always start with accessing the document object.
Below are some examples of how you can use the document object to access and manipulate HTML.

Finding HTML Elements-

Method		**Description**
document.getElementById	-	Find element by its Id
document.getElementsByTagName	-	Find element by tag

Examples of Methods used by JS

Changing HTML Elements-

Method	**Description**
element.innerHTML	- Change the inner HTML of an element
element.*attribute*	- Change the attribute value of an HTML element
element.style.*property*	- Change the style of an HTML element

Examples of Methods used by JS

Adding and deleting HTML Elements-

Method	**Description**
document.createElement	Create an HTML element
document.removeChild	Remove an HTML element
document.appendChild	Add an HTML element

Adding Event handlers-

document.getElementById(*id*).onclick = function(){*code*} - Adding event handler code to an onclick event

Example

```
<body>

<div id = "divId">
<p id="p1">Hello World!</p>
<p id="p2">Hello World!</p>
</div>

<script>
//use of element.attribute - change of attribute value
document.getElementById("p2").style.color = "blue";

//choosing by id and tag name and changing innerHTML
var x = document.getElementsByTagName("p");
document.getElementById("p2").innerHTML =
'The second paragraph (indexing-0) is: ' + x[1].innerHTML;

//use of parent-child relationship adding a child to p element
//and then p to div element
var para = document.createElement("p");
var node = document.createTextNode("Yet another paragraph");
para.appendChild(node);
var element = document.getElementById("divId");
element.appendChild(para);
</script>
</body>
</html>
```

Example:Output

Output:

Hello World!

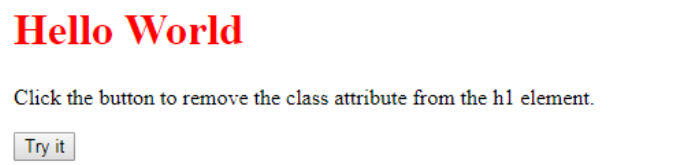
The second paragraph (indexing-0) is: Hello World!

Yet another paragraph

Element.attribute: Example

```html
<!DOCTYPE html>
<html>
<head>
<style>
.democlass {
    color: red;
}
</style>
</head>
<body>

<h1 class="democlass">Hello World</h1>

<p id="demo">Click the button to remove the class attribute from the h1 element.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    document.getElementsByTagName("H1")[0].removeAttribute("class");
}
</script>

</body>
</html>
```

Before:



After:

Document.removeChild: Example

```
<!DOCTYPE html>
<html>
<body>

<!-- Note that the <li> elements inside <ul> are not indented (whitespaces).
If they were, the first child node of <ul> would be a text node
-->
<ul id="myList"><li>Coffee</li><li>Tea</li><li>Milk</li></ul>

<p>Click the button to remove the first item from the list.</p>

<button onclick="myFunction()">Try it</button>

<script>
function myFunction() {
    var list = document.getElementById("myList");
    list.removeChild(list.childNodes[0]);
}
</script>

</body>
</html>
```

Before

- Coffee
- Tea
- Milk

Click the button to remove the first item from the list.

Try it

After

- Tea
- Milk

Click the button to remove the first item from the list.

Try it

Introduction to PHP

- PHP == ‘Hypertext Preprocessor’
- Open-source, server-side scripting language
- Used to generate dynamic web-pages
- PHP scripts reside between reserved PHP tags
- This allows the programmer to embed PHP scripts within HTML pages
- Interpreted language, scripts are parsed at run-time rather than compiled beforehand
- Executed on the server-side

PHP

- Source-code not visible by client
- ‘View Source’ in browsers does not display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

What does PHP code look like?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
    .....
?>
```

Comments in PHP

Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment

# Shell-style comments

/* C-style comments
These can span multiple lines */
```

Variables in PHP

- PHP variables must begin with a “$” sign
- Case-sensitive ($Foo != $foo != $fOo)
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - Form variables ($_POST, $_GET)
  - Server variables ($_SERVER)

Variable usage

```
<?php
$foo = 25; // Numerical variable
$bar = “Hello”; // String variable


$foo = ($foo * 7); // Multiplies foo by 7
$bar = ($bar * 7); // Invalid expression
?>
```

Echo

- The PHP command '**echo**' is used to output the parameters passed to it
  - The typical usage for this is to send data to the client's web-browser
- Syntax
  - void **echo** (string arg***1*** [, string arg***n***...])
  - In practice, arguments are not passed in parentheses since **echo** is a language construct rather than an actual function

Echo example

```
<?php
$foo = 25; // Numerical variable
$bar = “Hello”; // String variable


echo $bar; // Outputs Hello
echo $foo,$bar; // Outputs 25Hello
echo “5x5=”,$foo; // Outputs 5x5=25
echo “5x5=$foo”; // Outputs 5x5=25
echo ‘5x5=$foo’; // Outputs 5x5=$foo
?>
```

Arithmetic Operations

```
<?php
        $a=15;
        $b=30;
        $total=$a+$b;
        Print $total;
        Print “<p><h1>$total</h1>”;
        // total is 45
?>
```

- $a - $b // subtraction
- $a * $b // multiplication
- $a / $b // division
- $a += 5 // $a = $a+5 Also works for *= and /=

Escaping the Character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.

```
<?php
$heading="\"Computer Science\"";
Print $heading;
?>
```

"Computer Science"

PHP - GET & POST Methods

The GET Method

The GET method sends the encoded user information appended to the page request. The page and the encoded information are separated by the **?** character.

```
<?php
   if( $_GET["name"] || $_GET["age"] ) {
      echo "Welcome ". $_GET['name']. "<br />";
      echo "You are ". $_GET['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "GET">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

PHP - GET & POST Methods

The POST Method

The POST method transfers information via HTTP headers. The information is encoded as described in case of GET method and put into a header called QUERY_STRING.

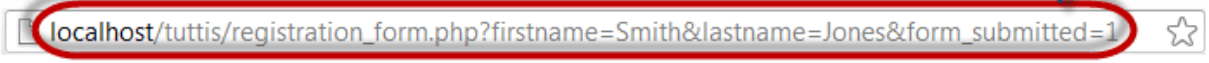
```
<?php
   if( $_POST["name"] || $_POST["age"] ) {
      if (preg_match("/[^A-Za-z'-]/",$_POST['name'] )) {
         die ("invalid name and name should be alpha");
      }
      echo "Welcome ". $_POST['name']. "<br />";
      echo "You are ". $_POST['age']. " years old.";

      exit();
   }
?>
<html>
   <body>

      <form action = "<?php $_PHP_SELF ?>" method = "POST">
         Name: <input type = "text" name = "name" />
         Age: <input type = "text" name = "age" />
         <input type = "submit" />
      </form>

   </body>
</html>
```

PHP - GET & POST Methods

FORM SUBMISSION GET METHOD

```
<form action="registration_form.php" method="GET">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

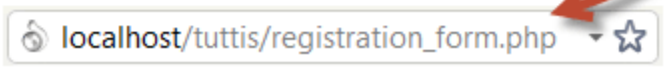
SUBMISSION URL SHOWS FORM VALUES



FORM SUBMISSION POST METHOD

```
<form action="registration_form.php" method="POST">
    First name: <input type="text" name="firstname"><br>
    Last name: <input type="text" name="lastname">
    <br>
    <input type="hidden" name="form_submitted" value="1"/>
    <input type="submit" value="Submit">
</form>
```

Submission URL does not show form values



What is JSON ?

- JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange.
- The official Internet media type for JSON is application/json.
- The JSON filename extension is .json.

Example of JSON Object

- Example of JSON object:

```
{
   "CS207_BTech_TA": [

      {
         "name":"AJ Ladhha",
         "enrollmentNumber": "B16004"
      },
      {
         "name":"Akul Gupta",
         "enrollmentNumber": "B16006"
      },
      {
         "name":"Aman Khandelwal",
         "enrollmentNumber": "B16007"
      },
      {
         "name":"Shashwat Garg",
         "enrollmentNumber": "B16034"
      },
      {
         "name":"Virendrasingh Suryavanshi",
         "enrollmentNumber": "B16037"
      }
   ]
}
```

More about JSON:

- Data structures supported by JSON
  - Collection of name/value pairs − This Data Structure is supported by different programming languages.
  - Ordered list of values − It includes array, list, vector or sequence etc.
- The attributes of a JSON Object are accessed by a '.' operator. For example:

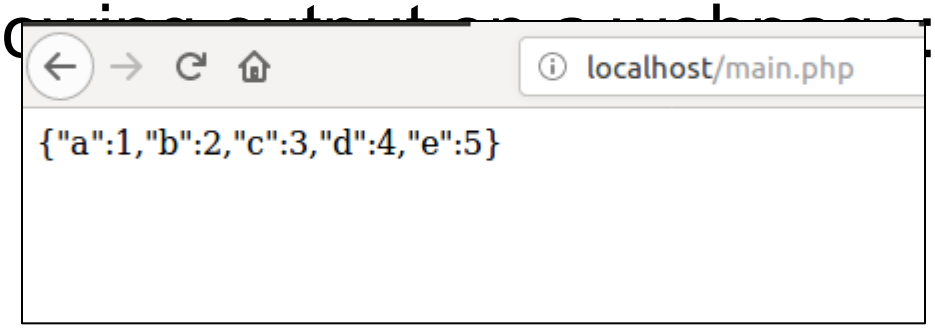
```
var JsonObj = {
    "username":"user",
    "password":"pass"
};

console.log(JsonObj.username);
```

Using JSON with PHP

- Encoding JSON in PHP (**json_encode**):
  - Example: creating a json object from

```php
<?php
   $arr = array('a' => 1, 'b' => '2', 'c' => 3, 'd' => 4, 'e' => 5);
   $jsonObject=json_encode($arr);
   echo $jsonObject;
?>
```

  - The above code in PHP gives the following output on a webpage:



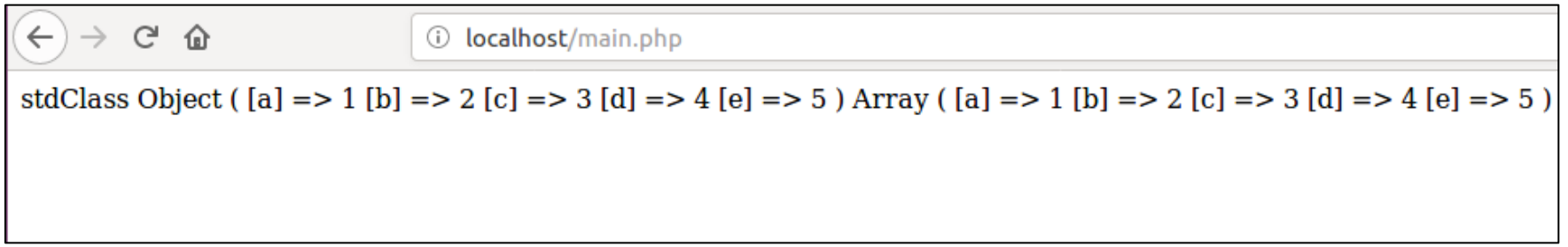
Using JSON with PHP

- Decoding JSON in PHP (json_decode):
  - Example: creating an object or array from stringified json object:

```php
<?php
    $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
    /* The attribute/property name should be inside double inverted commas,
       while the whole object must be inside single inverted commas. */
    $convertedObject = json_decode($json);
    // The function returns an object.
    print_r($convertedObject);
    // print_r() displays information about a variable in human readable form.
    $convertedArray = json_decode($json,true);
    // The flag "true" makes the function return an array.
    print_r($convertedArray);
?>
```

Using JSON with PHP

- The last code in PHP gives the following output on the webpage:



What is Node.JS

- Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.
- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.
- It is Asynchronous, and very fast, which makes it the first choice for developers all around the world for server-side scripting.

When to use Node.JS

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications
- It is **not advisable** to use Node.js for CPU intensive applications.

Prerequisites for running Node.JS on your system

- Text editor is required to save and edit a file of the extension .js
- Node.JS Runtime Distribution needs to be installed on the system. It can be downloaded from the official website: www.nodejs.org
- The Node Package Manager (NPM) provides many functionalities to make development easier. It comes bundled with Node.JS packages.

Creating Node.JS Application

Step 1: Import required Module

We use the “require” directive to load the http module and store the returned HTTP instance into an http variable as follows −

```
var http = require("http");
```

Creating Node.JS Application contd.

Step 2: Create Server

- We use the created http instance and call **http.createServer()** method to create a server instance and then we bind it to port 8081 using the **listen** method associated with the server instance. This creates an HTTP server which will listen, i.e., wait for a request over 8081 port on the local machine. Pass it a function with parameters **request** and **response**. Current sample implementation always returns “Hello World”:

```
http.createServer(function (request, response) {
   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World\n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Creating Node.JS Application contd.

Step 3: Testing Request and Response

Just combine the code in the above two steps to get the code shown below, and save it as main.js.

```
var http = require("http");

http.createServer(function (request, response) {
   // Send the HTTP header
   // HTTP Status: 200 : OK
   // Content Type: text/plain
   response.writeHead(200, {'Content-Type': 'text/plain'});

   // Send the response body as "Hello World"
   response.end('Hello World\n');
}).listen(8081);

// Console will print the message
console.log('Server running at http://127.0.0.1:8081/');
```

Creating Node.JS Application contd.

Now execute the main.js to start the server as follows −

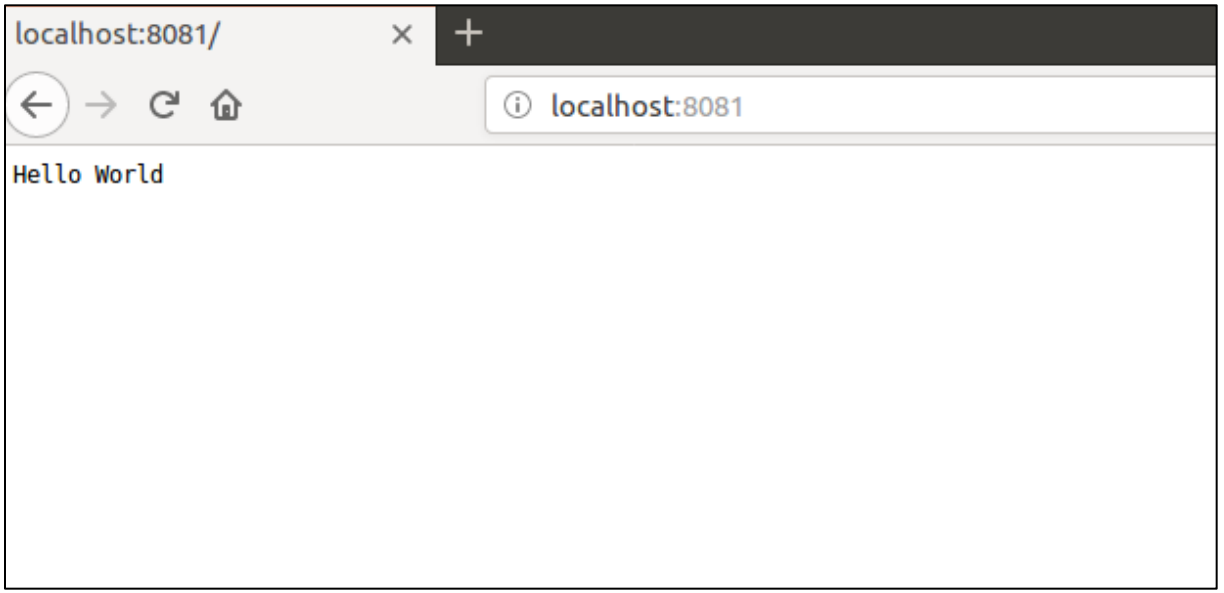
```
~$ node main.js
```

Verify the output that the server has started:

```
Server running at http://127.0.0.1:8081/
```

Creating Node.JS Application contd.

Make a request to the Node.JS server by simply opening http://localhost:8081/ in any browser and observe the following result:



References

- https://www.tutorialspoint.com
- http://php.net/manual/en/function.print-r.php
- http://www.w3schools.com
- https://developer.mozilla.org/en-US/docs/Web/JavaScript
- http://php.net/

- *Acknowledgements:* Thank you to Shashwat Garg, Suryavanshi Virendrasingh, Aman Khandelwal and Abhinav Choudhury for their help with the lecture, activity, and assignment materials for this week.