VTU 4th SEMESTER MCA EXAMINATION JUNE JULY 2018

CBCS SCHEME

USN 1 C R 1 7 M C A 7 6

16MCA42

Fourth Semester MCA Degree Examination, June/July 2018
Advanced Web Programming

Time: 3 hrs. Max. Marks: 80

Note: Answer any FIVE full questions, choosing one full question from each module.

2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-1

1 a. How jQuery is structured? **(06 Marks)**

b. Which versions of jQuery useful for governmental websites and company local network? **(02 Marks)**

c. Write the situation in which jQuery document ready handler is useful. **(04 Marks)**

d. Describe the situation in which jQuery universal selector can applied. **(04 Marks)**

OR

2 a. Explain the basic selectors of jQuery with suitable examples. **(10 Marks)**

b. Create a custom filter to select elements having only numbers, letters or the underscore (-) as their text. **(06 Marks)**

Module-2

3 a. Explain any three PHP lexical structure with suitable example. **(06 Marks)**

b. Write a PHP code to handle the following form data:
(i) Name field (ii) Date field (iii) Gender (o male o female)
(iv) Contact number (Provide dummy values) **(10 Marks)**

OR

4 a. Discuss string handling functions in PHP (Any 3) with suitable examples. **(06 Marks)**

b. Write a PHP program to demonstrate
(i) Cookie (ii) Session. **(10 Marks)**

Module-3

5 a. Explain Input and Output statements in ruby with suitable examples. **(04 Marks)**

b. Write a ruby program to count the words and their frequencies in the given string. **(04 Marks)**

c. Explain the form handling in rails with example program. **(08 Marks)**

OR

6 a. Explain built-up methods for arrays and lists in ruby. **(08 Marks)**

b. Give an example how dynamic documents are generated in Ruby on rails. **(08 Marks)**

Module-4

7 a. Discuss how the rich browser experience is made possible. **(06 Marks)**

b. What is JSON? Explain different types of literals used in JSON with examples. **(10 Marks)**

OR

8 a. What is Web 2.0? Explain. **(04 Marks)**

b. Explain REST with sample REST call. **(06 Marks)**

c. Write a note on SOAP and WSDL. **(06 Marks)**

Module-5

9	a.	Write a note on d3.js	(04 Marks)
	b.	Explain d3.js selects method.	(04 Marks)
	c.	Write a program to draw the bar graph for the following data set [2, 10, 12, 50].	(08 Marks)

OR

10	a.	Explain any four d3.js scales.	(08 Marks)
	b.	Assume any data set and write a d3.js program to draw a Pie chart.	(08 Marks)

* * * * *

1. jQuery is just a **JavaScript library**.What jQuery does essentially is let you **manipulate the DOM**.It handles the browser quirks but doesn't add any features to the browser or to JavaScript itself.
2. the jQuery function isn't a function constructor.It's just a function that returns an object,a function that returns a call to function constructor.

The jQuery JavaScript library has become one of the most widely utilized open source software (OSS) tools for web developers since its release in 2006. The extendable JavaScript library lets developers create a custom user interaction in websites by simply using less code. Developers can create dynamic websites with the client-side scripting of HTML across multiple web browsers and CSS manipulation. The multi-browser capabilities allow developers to manipulate Document Object Model (DOM) elements, add animation and effects to websites, standalone widgets, and implement many more innovative techniques. Here at Segue developers deploy jQuery on government and commercial web applications. The features are most noticed in the user interface as site visitor's click through menu items. One of my favorite Segue designed sites, utilizes a jQuery photo gallery to display fan photos as they devour the company's perfectly crafted cheeseburgers and delicious fries.

2.JQuery 1.x and 2.x

3. A page can't be manipulated safely until the document is "ready." jQuery detects this state of readiness for you. Code included inside $(document).ready() will only run once the page Document Object Model (DOM) is ready for JavaScript code to execute. Code included inside $(window).on("load", function() { ... }) will run once the entire page (images or iframes), not just the DOM, is ready. The ready event occurs when the DOM (document object model) has been loaded.

Because this event occurs after the document is ready, it is a good place to have all other jQuery events and functions. Like in the example above.

The ready() method specifies what happens when a ready event occurs.

4. The universal selector selects all the elements available in the document.

- Like any other jQuery selector, this selector also returns an array filled with the found elements. **$('*')** selects all the elements available in the document.

Following example would select all the elements and will apply yellow color to their background. Try to understand that this selector will select every element including head, body etc.

```
<html>

   <head>

      <title>The Selecter Example</title>

      <script type = "text/javascript"

         src =
"https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery
.min.js">

      </script>


      <script type = "text/javascript" language =
"javascript">

         $(document).ready(function() {

            /* This would select all the elements */
```

```
            $("*").css("background-color", "yellow");
         });
      </script>
   </head>

   <body>
      <div class = "big" id = "div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class = "medium" id = "div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class = "small" id = "div3">
         <p>This is third division of the DOM</p>
      </div>
   </body>
</html>
```

2 a.

Selector	**Example**	**Selects**

*	$("*")	All elements
#id	$("#lastname")	The element with id="lastname"
.class	$(".intro")	All elements with class="intro"
.class,.class	$(".intro,.demo")	All elements with the class "intro" or "demo"
element	$("p")	All <p> elements
el1,el2,el3	$("h1,div,p")	All <h1>, and <p> elements
:first	$("p:first")	The first <p> element
:last	$("p:last")	The last <p> element

:even	$("tr:even")	All even <tr> elements
:odd	$("tr:odd")	All odd <tr> elements
:first-child	$("p:first-child")	All <p> elements that are the first child of their parent
:first-of-type	$("p:first-of-type")	All <p> elements that are the first <p> element of their parent
:last-child	$("p:last-child")	All <p> elements that are the last child of their parent
:last-of-type	$("p:last-of-type")	All <p> elements that are the last <p> element of their parent
:nth-child(*n*)	$("p:nth-child(2)")	All <p> elements that are the 2nd child of their parent

:nth-last-child(*n*)	$("p:nth-last-child(2)")	All <p> elements that are the 2nd child of their parent, counting from the last child
:nth-of-type(*n*)	$("p:nth-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent
:nth-last-of-type(*n*)	$("p:nth-last-of-type(2)")	All <p> elements that are the 2nd <p> element of their parent, counting from the last child
:only-child	$("p:only-child")	All <p> elements that are the only child of their parent
:only-of-type	$("p:only-of-type")	All <p> elements that are the only child, of its type, of their parent
parent > child	$("div > p")	All <p> elements that are a direct child of a element

parent descendant	$("div p")	All <p> elements that are descendants of a element
element + next	$("div + p")	The <p> element that are next to each elements
element ~ siblings	$("div ~ p")	All <p> elements that are siblings of a element

b.

```
$(document).ready(function(){
  $(p).click(function(){
    var sd=$(this).text();
    var num = sd.match(/[\d\.]+/g);
    if (num != null){
        var number = num.toString();
        alert(number );
    }
  });
});
```

Module - 2

3a. PHP lexical structure

Computer languages, like human languages, have a lexical structure. A source code of a PHP script consists of tokens. Tokens are atomic code elements. In PHP language, we have comments, variables, literals, operators, delimiters, and keywords.

PHP comments

Comments are used by humans to clarify the source code. All comments in PHP follow the #character.

```
<?php

# comments.php
# Author Jan Bodnar
# ZetCode 2016

echo "This is comments.php script\n";

?>
```

Everything that follows the # character is ignored by the PHP interpreter.

```
// comments.php
// author Jan Bodnar
// ZetCode 2016

/*
 comments.php
 author Jan Bodnar
 ZetCode 2016
*/
```

PHP also recognizes the comments from the C language.

PHP white space

White space in PHP is used to separate tokens in PHP source file. It is used to improve the readability of the source code.

```
public $isRunning;
```

White spaces are required in some places; for example between the access specifier and the variable name. In

other places, it is forbidden. It cannot be present in variable identifiers.

```
$a=1;
$b = 2;
$c  =  3;
```

The amount of space put between tokens is irrelevant for the PHP interpreter. It is based on the preferences and the style of a programmer.

```
$a = 1;
$b = 2; $c = 3;
$d
  =
   4;
```

We can put two statements into one line. Or one statement into three lines. However, source code should be readable for humans. There are accepted standards of how to lay out your source code.

PHP semicolon

A semicolon is used to mark the end of a statement in PHP. It is mandatory.

```
$a = 34;
$b = $a * 34 - 34;
echo $a;
```

Here we have three different PHP statements. The first is an assignment. It puts a value into the $avariable. The second one is an expression. The expression is evaluated and the output is given to the $b variable. The third one is a command. It prints the $a variable.

PHP variables

A variable is an identifier, which holds a value. In programming we say that we assign a value to a variable. Technically speaking, a variable is a reference to a

computer memory, where the value is stored. In PHP language, a variable can hold a string, a number, or various objects like a function or a class. Variables can be assigned different values over time.

Variables in PHP consist of the $ character, called a sigil, and a label. A label can be created from alphanumeric characters and an underscore _ character. A variable cannot begin with a number. The PHP interpreter can then distinguish between a number and a variable more easily.

```
$Value
$value2
$company_name
```

These were valid PHP identifiers.

```
$12Val
$exx$
$first-name
```

These were examples of invalid PHP identifiers.

The variables are *case sensitive*. This means that $Price, $price, and $PRICE are three different identifiers.

case.php

```
<?php

$number = 10;
$Number = 11;
$NUMBER = 12;

echo $number, $Number, $NUMBER;

echo "\n";

?>
```

In our script, we assign three numeric values to three variables and print them. However, for clarity reasons, it

is not recommended to create variables which differ only in case; it is considered a poor practice.

```
$ php case.php
101112
```

This is the output of the script.

PHP constants

A constant is an identifier for a value which cannot change during the execution of the script. By convention, constant identifiers are always uppercase.

constants.php

```
<?php

define("SIZE", 300);
define("EDGE", 100);

#SIZE = 100;

echo SIZE;
echo EDGE;

echo "\n";

?>
```

In the script, we define two constants.

```
define("SIZE", 300);
define("EDGE", 100);
```

Constants are created with the define() function.

```
#SIZE = 100;
```

Constants differ from variables; we cannot assign a different value to an existing constant. The script will fail if we uncomment the line.

```
echo SIZE;
echo EDGE;
```

Constants do not use the dollar sigil character.

```
$ php constants.php
300100
```

This is the output of the constants script.

The following is a list of PHP compile time constants.

```
__CLASS__     __DIR__         __FILE__     __FUNCTION__
__METHOD__    __NAMESPACE__
```

PHP literal

A literal is any notation for representing a value within the PHP source code. Technically, a literal is assigned a value at compile time, while a variable is assigned at runtime.

```
$age = 29;
$nationality = "Hungarian";
```

Here we assign two literals to variables. Number 29 and string "Hungarian" are literals.

literals.php

```
<?php

$name1 = "Jane ";
$age1 = 17;

$name2 = "Rose ";
$age2 = 16;

echo "Patrick 34\n";
echo "Luke 22\n";

echo $name1, $age1, "\n";
echo $name2, $age2, "\n";
```

```
?>
```

If we do not assign a literal to a variable, there is no way how we can work with it—it is dropped.

```
$ php literals.php
Patrick 34
Luke 22
Jane 17
Rose 16
```

This is the output of the literals.php script.

PHP operators

An operator is a symbol used to perform an action on some value.

```
+    -    *    /    %    ++    --
=    +=   -=   *=   /=   .=    %=
==   !=   ><   >   <   >=   <=
&&   ||   !   xor   or
&    ^    |   ~   .   <<   >>
```

These are PHP operators. We will talk about operators later in the tutorial.

PHP delimiters

A delimiter is a sequence of one or more characters used to specify the boundary between separate, independent regions in plain text or other data stream.

```
$a = "PHP";
$b = 'Java';
```

The single and double characters are used to mark the beginning and the end of a string.

```
function setDate($date) {
    $this->date = $data;
```

```
}

if ( $a > $b) {
    echo "\$a is bigger than \$b";
}
```

Parentheses are used to mark the function signature. The signature is the function parameters. Curly brackets are used to mark the beginning and the end of the function body. They are also used in flow control.

```
$a = array(1, 2, 3);
echo $a[1];
```

The square brackets are used to mark the array index.

```
/*
 Author Jan Bodnar
 January 2016
 ZetCode
*/
```

/* */ delimiters are used to provide C style comments in PHP.

```
<?php
// PHP code
?>
```

The <?php and ?> delimiters are used to delimit PHP code in a file.

PHP keywords

A keyword is a reserved word in the PHP programming language. Keywords are used to perform a specific task in a computer program; for example, print a value, do repetitive tasks, or perform logical operations. A programmer cannot use a keyword as an ordinary variable.

The following is a list of PHP keywords.

abstract and array() as break

```
case        catch        class       clone       const
continue    declare      default     do          else
elseif      enddeclare   endfor      endforeach  endif
endswitch   endwhile     extends     final       for
foreach     function     global      goto        if
implements  interface    instanceof  namespace   new
or          private      protected   public      static
switch      throw        try         use         var
while       xor
```

Next we have other language constructs.

```
die()           echo()          empty()     exit()
eval()
include()       include_once()  isset()     list()
require()
require_once()  return()        print()     unset()
```

3b. <?php

```
If($_POST[“textfieldname”]=”value”)

Echo “ correct”;
```

```
$unixTime = Time();
print date("m/d/y h:i:s a", $unixTime);
if (isset($gender) && $gender=="male") echo "checked";

if (is_numeric($element)) {
        echo var_export($element, true) . " is numeric", P
HP_EOL;
    }
```

4a.

Function	**Description**

addcslashes()	Returns a string with backslashes in front of the specified characters
addslashes()	Returns a string with backslashes in front of predefined characters
bin2hex()	Converts a string of ASCII characters to hexadecimal values
chop()	Removes whitespace or other characters from the right end of a string
chr()	Returns a character from a specified ASCII value
chunk_split()	Splits a string into a series of smaller parts
convert_cyr_string()	Converts a string from one Cyrillic character-set to another
convert_uudecode()	Decodes a uuencoded string

convert_uuencode()	Encodes a string using the uuencode algorithm
count_chars()	Returns information about characters used in a string
crc32()	Calculates a 32-bit CRC for a string
crypt()	One-way string hashing
echo()	Outputs one or more strings
explode()	Breaks a string into an array

4b.

This is a suitable method for tracking users. Web servers are stateless entities.

Cookies :-

Cookies provide a way for a server to store information about a user on the user's machine. This enables to maintain the user's visit to the site , so that they can track their movement through the site , or to store information such as their user name .

Cookies allow data to be stored in the form of a name/value pair. Both the name and the value are set at choice.

It contains the following syntax

Name=value;expires=expiration date gmt; path=url(optional)

To create a cookie

Setcookie(name,value,expires,path);

Cookie name/value pair: The first section of the cookie defines the name of the cookie and the value assigned to the cookie. Both the name and value settings can be any value as per the users choice.

Cookie Expiration time: The optional expires= section specifies the date on which the associated cookie should expire. The PHP time() can be used to obtain and manipulate dates for this purpose

Example:

```
<?php
Setcookie('username','abcd',time()+4800);
Echo "cookie has been set";
?>
```

The cookie will expire after 4800 seconds.

Cookies are sent in the HTTP headers in pages sent by the browser . Once the cookies has been set they can be accessed on the next page load with the $_COOKIE array. This is an associative array where the name of the cookie provides the index into the array to extract the corresponding value of the name/value pair.

Example:

```
<?php
Echo “cookie value is “.$_COOKIE[‘username’];
?>
```

To delete a cookie

Cookies are deleted by calling the setcookie() function with the cookie name , a null for the value and an expiration date in the past.

```
<?php
Setcookie(“username”,””,time()-4800);
?>
```

Sessions

A session is a way to store information (in variables) to be used across multiple pages. A session creates a file in a temporary directory on the server where registered

session variables and their values are stored. The data will be available to all pages on the site during the visit.

A session like a cookie provides a way to track data for a user over a series of pages.

The main difference between cookie and session is that cookie stores the data on the client side in the web browser whereas the session data is stored on the server.

Sessions are generally more secure, because the data is not transmitted back and forth between the client and server repeatedly.

Sessions let you store more information than you can in cookie.

When session starts, PHP generates a random session id , a reference to that particular session and its stored data.

To start a session

```
Session_start();
```

Session variables are set with the PHP global variables $_SESSION.

```
<?php
```

```
Session_start();
$_SESSION[“username”] = “abc”;
?>
```

To delete the session variables , we unset $_SESSION array

```
Unset($_SESSION[“username”]);
```

Module – 3

5a.

Input

```
 Gets
gets.to_i
Gets.chomp
```

Output

```
Puts
```

5b.

```
the_file='/Users/Al/DD/Ruby/GettysburgAddress.txt'
h = Hash.new
f = File.open(the_file, "r")
f.each_line { |line|
  words = line.split
```

```
  words.each { |w|
    if h.has_key?(w)
      h[w] = h[w] + 1
    else
      h[w] = 1
    end
  }
}


# sort the hash by value, and then print it in this sorted
order
h.sort{|a,b| a[1]<=>b[1]}.each { |elem|
  puts "\"#{elem[0]}\" has #{elem[1]} occurrences"
}
```

Here's a little more discussion of the program:

1. Create a String to store the file name
2. Create a new Hash
3. Open the file in read-only mode
4. Read each line in the file, one line at a time
5. Split each line into words (words separated by spaces)

6. Put the word and the word frequency into the Hash (the word is the key, the frequency is the value)
7. Print the hash, with the results sorted by the hash value

It may help to understand the program, so I'll show the last 10 lines of the output here:

```
"have" has 5 occurrences

"not" has 5 occurrences

"can" has 5 occurrences

"and" has 6 occurrences

"--" has 7 occurrences

"a" has 7 occurrences

"we" has 8 occurrences

"to" has 8 occurrences

"the" has 9 occurrences

"that" has 13 occurrences
```

5c.

Form

To create a form tag with the specified action, and with POST request, use the following syntax –

```
<%= form_tag :action => 'update', :id => @some_object %>
```

```
<%= form_tag( { :action => :save, }, { :method => :post })
%>
```

ext Fields

To create a text field use the following syntax −

```
<%= text_field :modelname, :attribute_name, options %>
```

Have a look at the following example −

```
<%= text_field "person", "name", "size" => 20 %>
```

This will generate following code −

```
<input type = "text" id = "person_name" name =
"person[name]"

   size = "20" value = "<%= @person.name %>" />
```

To create hidden fields, use the following syntax;

```
<%= hidden_field ... %>
```

To create password fields, use the following syntax;

```
<%= password_field ... %>
```

To create file upload fields, use the following syntax;

```
<%= file_field ... %>
```

Text Area

To create a text area, use the following syntax −

```
<%= text_area ... %>
```

Have a look at the following example −

```
<%= text_area "post", "body", "cols" => 20, "rows" => 40%>
```

This will generate the following code −

```
<textarea cols = "20" rows = "40" id = "post_body" name ="
post[body]">

   <%={@post.body}%>

</textarea>
```

Radio Button

To create a Radio Button, use the following syntax −

```
<%= radio_button :modelname, :attribute, :tag_value,
options %>
```

Have a look at the following example −

```
radio_button("post", "category", "rails")

radio_button("post", "category", "java")
```

This will generate the following code −

```
<input type = "radio" id = "post_category" name =
"post[category]"

   value = "rails" checked = "checked" />

<input type = "radio" id = "post_category" name =
"post[category]" value = "java" />
```

Checkbox Button

To create a Checkbox Button use the following syntax −

```
<%= check_box :modelname,
:attribute,options,on_value,off_value%>
```

Have a look at the following example −

```
check_box("post", "validated")
```

This will generate the following code −

```
<input type = "checkbox" id = "post_validate" name =
"post[validated]"

   value = "1" checked = "checked" />

<input name = "post[validated]" type = "hidden" value =
"0" />
```

Let's check another example −

```
check_box("puppy", "gooddog", {}, "yes", "no")
```

This will generate following code −

```
<input type = "checkbox" id = "puppy_gooddog" name =
"puppy[gooddog]" value = "yes" />

<input name = "puppy[gooddog]" type = "hidden" value =
"no" />
```

Options

To create a dropdopwn list, use the following syntax −

```
<%= select
:variable,:attribute,choices,options,html_options%>
```

Have a look at the following example −

```
select("post", "person_id", Person.find(:all).collect {|p|
[ p.name, p.id ] })
```

This could generate the following code. It depends on what value is available in your database. −

```
<select name = "post[person_id]">

   <option value = "1">David</option>

   <option value = "2">Sam</option>

   <option value = "3">Tobias</option>
```

```
</select>
```

Date Time

Following is the syntax to use data and time −

```
<%= date_select :variable, :attribute, options %>

<%= datetime_select :variable, :attribute, options %>
```

Following are examples of usage −

```
<%=date_select "post", "written_on"%>

<%=date_select "user", "birthday", :start_year => 1910%>

<%=date_select "user", "cc_date", :start_year => 2005,

   :use_month_numbers => true, :discard_day => true,
:order => [:year, :month]%>

<%=datetime_select "post", "written_on"%>
```

End Form Tag

Use following syntax to create </form> tag −

```
<%= end_form_tag %>
```

6a.

Array Built-in Methods

We need to have an instance of Array object to call an Array method. As we have seen, following is the way to create an instance of Array object −

```
Array.[](...) [or] Array[...] [or] [...]
```

This will return a new array populated with the given objects. Now, using the created object, we can call any available instance methods.

```
#!/usr/bin/ruby
```

```
digits = Array(0..9)

num = digits.at(6)

puts "#{num}"
```

This will produce the following result –

```
6
```

Hash Built-in Methods

We need to have an instance of Hash object to call a Hash method. As we have seen, following is the way to create an instance of Hash object –

```
Hash[[key =>|, value]* ] or


Hash.new [or] Hash.new(obj) [or]

Hash.new { |hash, key| block }
```

This will return a new hash populated with the given objects. Now using the created object, we can call any available instance methods.

Module 4

7a. Rich User Experience

Traditional web are built with HTML and CSS、CGI and had been offered as a static page . On the other hand Web 2.0 uses Ajax (Asynchronous JavaScript + XML) presenting dynamic , rich user experience to users .

For example, Google Provided Google Maps and Google Suggest

7b. JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following:

 Data is represented in name/value pairs.

 Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).

 Square brackets hold arrays and values are separated by ,(comma)

```
{
"book": [
{
"id":"01",
"language": "Java",
"edition": "third",
"author": "Herbert Schildt"
},
{
"id":"07",
"language": "C++",
"edition": "second"
"author": "E.Balagurusamy"
}]
}
```

JSON supports the following two data structures:
 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.

8a. Web 2.0 describes World Wide Web sites that emphasize user-generated content, usability, and interoperability. A Web 2.0 site may allow users to interact and collaborate with each other in a social media dialogue as creators of user-generated content in a virtual community, in contrast to Web sites where people are limited to the passive viewing of content. Examples of Web 2.0 include social networking sites, blogs, wikis, folksonomies, video sharing sites, hosted services, Web applications,

and mashups.

Web 2.0 is the current state of online technology as it compares to the early days of the Web, characterized by greater user interactivity and collaboration, more pervasive network connectivity and enhanced communication channels. One of the most significant differences between Web 2.0 and the traditional World Wide Web (WWW, retroactively referred to as Web 1.0) is greater collaboration among Internet users, content providers and enterprises. Originally, data was posted on Web sites, and users simply viewed or downloaded the content. Increasingly, users have more input into the nature and scope of Web content and in some cases exert real-time control over it. The social nature of Web 2.0 is another major difference between it and the original, static Web. Increasingly, websites enable community-based input, interaction, content-sharing and collaboration. Types of social media sites and applications include forums, microblogging, social networking, social bookmarking, social curation, and wikis.

8 b. Explain REST?

Representational State Transfer (REST) is an architectural **style** that specifies constraints, such as the uniform interface, that if applied to a web service induce desirable properties, such as performance, scalability, and modifiability, that enable services to work best on the Web.

RESTful Web Services are REST architecture based web services. In REST Architecture everything is a resource. RESTful web services are light weight, highly scalable and maintainable and are very commonly used to create APIs for web based applications.

In REST architecture, a REST Server simply provides access to resources and REST client accesses and

presents the resources. Here each resource is identified by URIs/ global IDs. REST uses various representations to represent a resource like text, JSON and XML. Now a days JSON is the most popular format being used in web services.

HTTP Methods

Following well known HTTP methods are commonly used in REST based architecture.

- **GET** - Provides a read only access to a resource.
- **PUT** - Used to create a new resource.
- **DELETE** - Used to remove a resource.
- **POST** - Used to update a existing resource or create a new resource.

Properties of a REST Application

The REST style is characterized by the following properties:

- Communication takes place on call. The Client is active and requests a representation from the passive server and/or modifies a resource.
- A resource can be addressed by an unique URI.
- The client can request the representation of a resource in form of a document.
- Representations can refer to further resources.
- The server does not monitor the status of its clients. Each query to the server must contain all information that are necessary for interpreting itself.
- Caching is supported. The server can mark its answers as cacheable or not cacheable.

8c.

SOAP

SOAP is an XML-based protocol for exchanging information between computers.

- SOAP is a communication protocol.
- SOAP is for communication between applications.
- SOAP is a format for sending messages.
- SOAP is designed to communicate via Internet.
- SOAP is platform independent.
- SOAP is language independent.
- SOAP is simple and extensible.
- SOAP allows you to get around firewalls.
- SOAP will be developed as a W3C standard.

WSDL

WSDL is an XML-based language for describing web services and how to access them.

- WSDL stands for Web Services Description Language.
- WSDL was developed jointly by Microsoft and IBM.
- WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
- WSDL is the standard format for describing a web service.
- WSDL definition describes how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of UDDI, an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.

9a. D3 is a JavaScript library The D3 library allows us to manipulate elements of a web page in the context of a data set. These elements can be HTML, SVG, or Canvas elements, and can be introduced, removed, or edited according to the contents of the data set. So, for example, to create a scatter graph, we use D3 to arrange SVG circle elements such that their cx and cy attributes are set to the x- and y-values of the elements in a data set, scaled to map from their natural units into pixels. Instead of creating a traditional visualization toolkit, which typically places a heavy wrapper between the designer and the web page, D3 is focused on providing helper functions to deal with

mundane tasks, such as creating axes and axis ticks, or advanced tasks such as laying out graph visualizations or chord diagrams. This means that, once over D3's initial learning curve, the designer is opened up to a very rich world of modern, interactive and animated data visualization.

9b. **D3.js Select Method**

The first part of the JavaScript code that we wrote is **.select("body").**

The D3.js Select method uses CSS3 selectors to grab DOM elements. To learn more about CSS3 selectors please check this out => CSS3 Selectors

D3 looks at the document and selects the first descendant DOM element that contains the tag **body.**

Once an element is selected, D3.js allows you to apply **operators** to the element you have selected.

These operators can get or set things like "attributes", "properties", "styles", "HTML", and "text content".

9c.

```
<!DOCTYPE html>

<html>

<head>

    <script type = "text/javascript" src =
"d3.min.js"></script>

</head>

<body>

  <script>

     var data = [2,10,12,50]

     var width = 50, barHeight = 20, margin = 1;

     console.log(barHeight * data.length) //120
```

```
var scale = d3.scaleLinear()
    .domain([d3.min(data), d3.max(data)])
    .range([100, 400]);

var svg = d3.select("body")
    .append("svg")
    .attr("width", width)
    .attr("height", barHeight * data.length);

var g = svg.selectAll("g")
    .data(data)
    .enter()
    .append("g")
    .attr("transform", function (d, i) {
    //console.log("translate(0," + i * barHeight +
")") //0,0 0,20 0,40 0,60 0,80 0,100
        return "translate(0," + i * barHeight + ")";
});

g.append("rect")
.attr("width", function (d) {
//console.log(scale(d))
    return scale(d);
})
```

```
      .attr("height", barHeight - margin)
      g.append("text")
      .attr("x", function (d) { return (scale(d)); })
      .attr("y", barHeight / 2)
      .text(function (d) { return d; });
  </script>
   </body>
</html>
```

10a.

D3 provides the following important scaling methods for different types of charts. Let us understand then in detail.
d3.scaleLinear – Constructs a continuous linear scale where we can input data maps to the specified output range.
d3.scaleIdentity – Construct a linear scale where the input data is the same as the output.
d3.scaleTime – Construct a linear scale where the input data is in the dates and the output in numbers.
d3.scaleLog – Construct a logarithmic scale.
d3.scaleSqrt – Construct a square root scale.
d3.scalePow – Construct an exponential scale.
d3.scaleSequential – Construct a sequential scale where output range is fixed by interpolator function.

10b.

```
<body>
<svg width="300" height="200"> </svg>
<script>
```

```
    var data = [2, 4, 8, 10];

    var svg = d3.select("svg"),
        width = svg.attr("width"),
        height = svg.attr("height"),
        radius = Math.min(width, height) / 2,
        g = svg.append("g").attr("transform", "translate("
+ width / 2 + "," + height / 2 + ")");

    var                        color                        =
d3.scaleOrdinal(['#4daf4a','#377eb8','#ff7f00','#984ea3','
#e41a1c']);

    // Generate the pie
    var pie = d3.pie();

    // Generate the arcs
    var arc = d3.arc()
                .innerRadius(0)
                .outerRadius(radius);

    //Generate groups
    var arcs = g.selectAll("arc")
                .data(pie(data))
                .enter()
                .append("g")
                .attr("class", "arc")

    //Draw arc paths
    arcs.append("path")
        .attr("fill", function(d, i) {
            return color(i);
        })
        .attr("d", arc);
</script>
</body>
```