USN | | | | | | | | | | |

Internal Assessment Test 2 – June 2022

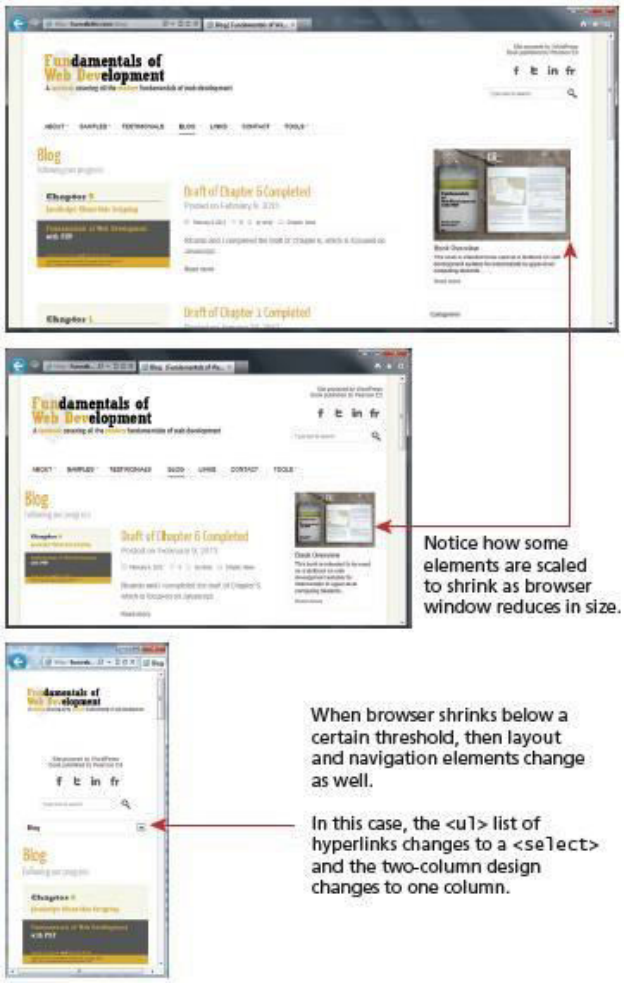
Sub:	WEB TECHNOLOGY AND ITS APPLICATIONS					Sub Code:	18CS63	Branch:	ISE	
Date:	09/06/2022	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI A, B & C		OBE	

	Answer any FIVE FULL Questions	MARKS	CO	RBT
1	Explain Responsive design in detail.	10	CO 2	L2
2	Illustrate the concepts of floating multiple items side by side and floating within a container.	10	CO 2	L2
3	List four different event types in JavaScript events and explain any two in detail.	10	CO 3	L2
4	Demonstrate the comparison of pass by value and pass by reference in PHP with suitable diagram.	10	CO 3	L2
5(a)	Demonstrate the comparison of client-side execution and server script execution with suitable diagram.	4	CO 3	L2
5(b)	Explain in detail the fundamental syntax of JavaScript.	6	CO 3	L2
6	Write the XHTML code to create the form with the following capabilities a) A text widget to collect the users name b) Four check boxes, one each for the following items i) Four 100 watt light bulbs for Rs. 20=39 ii) Eight 100 watt light bulbs for Rs 40=20 iii) Four 100 watt long life light bulbs for Rs. 30=95 iv) Eight 100 watt long life light bulbs for Rs 70=49 c) A collection of 3 radio buttons that are labeled as follows i) Visa ii) Master Card	10	CO 3	L3

Faculty Signature CCI Signature HOD Signature

1. Explain Responsive design in detail.

In a **responsive design**, the page "responds" to changes in the browser size that go beyond the width scaling of a liquid layout.
One of the problems of a liquid layout is that images and horizontal navigation elements tend to take up a fixed size, and when the browser window shrinks to the size of a mobile browser, liquid layouts can become unusable. In a responsive layout, images will be scaled down and navigation elements will be replaced as the browser shrinks, as shown in the figure below.



There are four key components that make responsive design work. They are:

1. Liquid layouts
2. Scaling images to the viewport size
3. Setting viewports via the <meta> tag
4. Customizing the CSS for different viewports using media queries

Responsive designs begin with a liquid layout, in which most elements have their widthsspecified as percentages. Making images scale in size is done as follows:

```
img {
max-width: 100%;
}
```

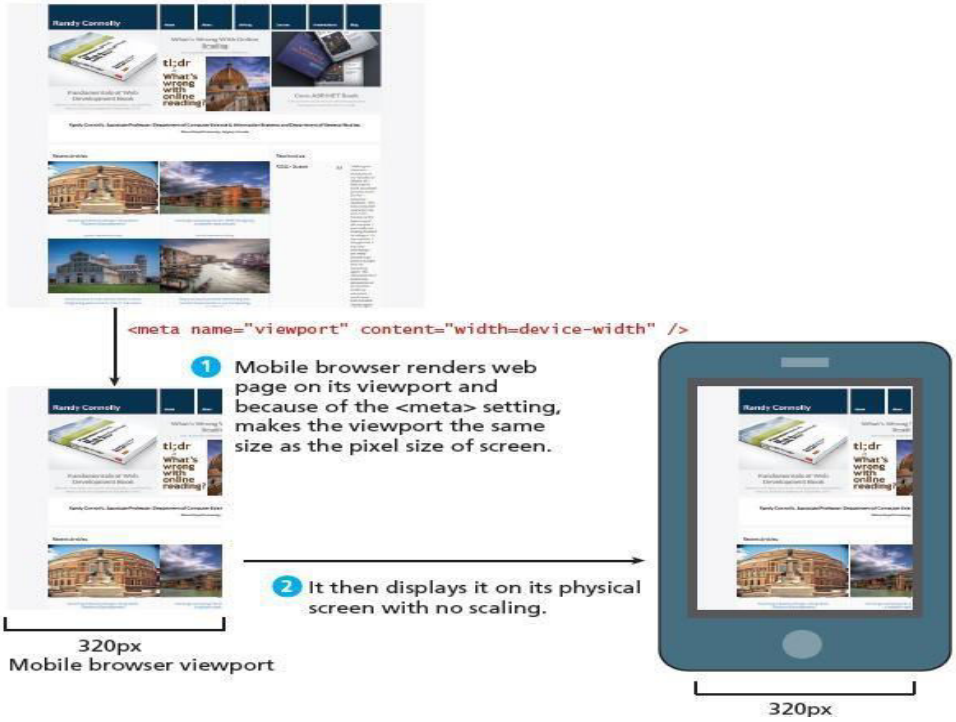
But this does not change the downloaded size of the image; it only shrinks or expands its visualdisplay to fit the size of the browser window, never expanding beyond its actual dimensions.

Setting Viewports

A key technique in creating responsive layouts is the ability of current mobile browsers to shrink or grow the web page to fit the width of the screen. The mobile browser renders the page on a canvas called the **viewport**. On iPhones, for instance, the viewport width is 980 px, and then thatviewport is scaled to fit the current width of the device. The mobile Safari browser introduced the viewport <meta> tag as a way for developers to control the size of that initial viewport.

```
<html>
<head>
<meta name="viewport" content="width=device-width" />
```

By setting the viewport as above, the page is telling the browser that no scaling is needed, and to make the viewport as many pixels wide as the device screen width. This means that if the device has a screen that is 320 px wide, the viewport width will be 320 px; if the screen is 480 px, then the viewport width will be 480 px.

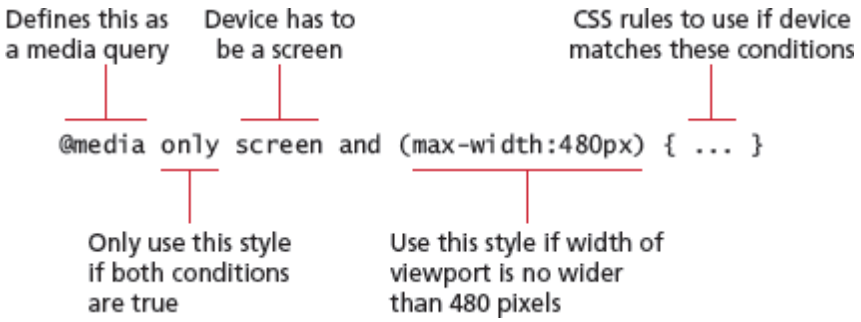


Media Queries

The other key component of responsive designs is **CSS media queries**. A media query is a way to apply style rules based on the medium that is displaying the file. Use these queries to look at the capabilities of the device, and then define CSS rules to target that device.

Example of media query

@media only screen and (max-width: 480px) {.....}



These queries are Boolean expressions and can be added to your CSS files or to the <link> element to conditionally use a different external CSS file based on the capabilities of the device.

w elements of the browser features that can be examined with media queries are –

Feature	Description
`width`	Width of the viewport
`height`	Height of the viewport
`device-width`	Width of the device
`device-height`	Height of the device
`orientation`	Whether the device is portrait or landscape
`color`	The number of bits per color

Contemporary responsive sites will typically provide CSS rules for phone displays first, then tablets, then desktop monitors, an approach called **progressive enhancement**. The media queries can be within your CSS file or within the <link> element; the later requires more HTTP requests but results in more manageable CSS files.

2. Illustrate the concepts of floating multiple items side by side and floating within a container.

It is possible to displace an element out of its position in the normal flow via the CSS float**property**. An element can be floated to the left or floated to the right.

When an item is floated, it is moved all the way to the far left or far right of its containing block and the rest of the content is "**re-flowed**" around the floated element.

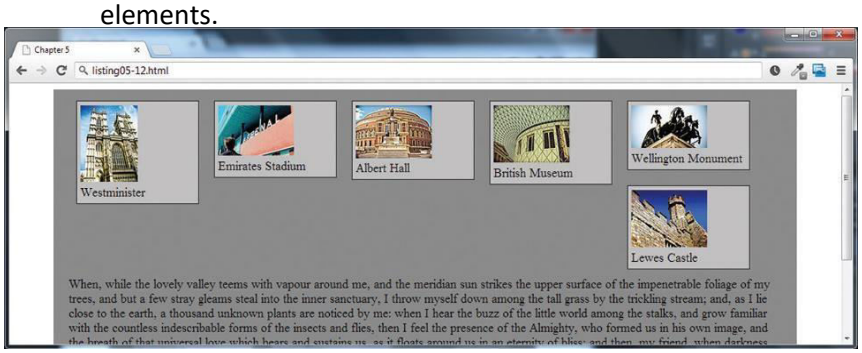
Notice that a floated block-level element must have a width specified; otherwise, the width will be set to auto, which will mean it implicitly fills the entire width of the containing block, and there will be no room available to flow content around the floated item.

Floating within a Container

It should be reiterated that a floated item moves to the left or right of its container. The floated figure contained within an <article> element that is indented from the browser's edge. The relevant margins and padding areas are color coded to help make it clearer how the float interacts with its container.

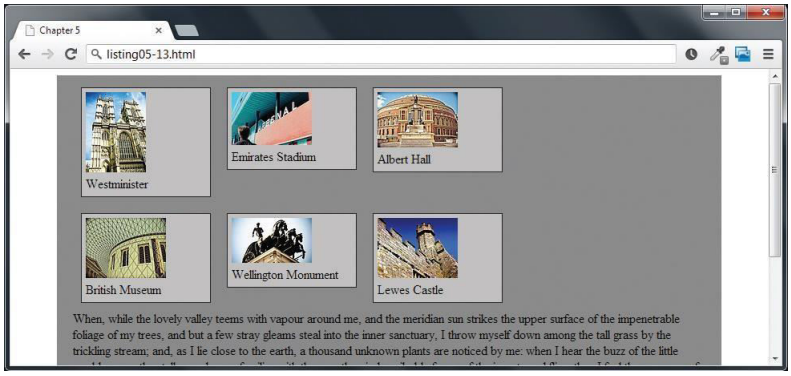
Floating Multiple Items Side by Side

A common use of float property is to place multiple items side by side on the same line. When multiple items are floated, each element will be nestled up beside the previously floated item. All other content in the containing block will flow around all the floated elements.



This arrangement of images floated changes as the browser window size changes. If suppose any element has to be stopped from flowing around a floated element, it can be done by using the **clear** CSS **property**.

By setting the clear property of third image to left, it means that there should be no elements toits left. { clear : left;}



The other values for clear property are described below –

Value	Description
left	The left-hand edge of the element cannot be adjacent to another element.
right	The right-hand edge of the element cannot be adjacent to another element.
both	the left-hand and right-hand edges of the element cannot be adjacent
none	The element can be adjacent to other elements.

Overlaying and Hiding Elements

One of the more common design tasks with CSS is to place two elements on top of each other, or to selectively hide and display elements. Positioning is important to both of these tasks. Positioning is often used for smaller design changes, such as moving items relative to other elements within a container.

An image that is the same size as the underlying one is placed on top of the other image using absolute positioning.

There are in fact two different ways to hide elements in CSS: using the display property and using the visibility property. The display property takes an item **out of the flow**: it is as if the element no longer exists. The visibility property just **hides the** element, but the space for that element remains.

3. List four different event types in JavaScript events and explain any two in detail.

There are several classes of event, with several types of event within each class. The classes are mouse events, keyboard events, form events, and frame events.

Mouse Events

Mouse events are defined to capture a range of interactions driven by the mouse. These can be further categorized as mouse click and mouse move events. The possible mouse events are –

Events	Description
onclick	The mouse was clicked on an element
ondblclick	The mouse was double clicked on an element
onmousedown	The mouse was pressed down over an element
onmouseup	The mouse was released over an element
onmouseover	The mouse was moved (not clicked) over an element
onmouseout	The mouse was moved off of an element
onmousemove	The mouse was moved while over an element

Keyboard Events

Keyboard events occur when taking inputs from the keyboard. These events are most useful within input fields. The possible keyboard events

Events	Description
onkeydown	The user is pressing a key (this happens first)
onkeypress	The user presses a key (this happens after onkeydown)
onkeyup	The user releases a key that was down (this happens last)

Eg: to validate an email address, on pressing any key
<input type="text" id="keyExample">

The input box above, for example, could be listened to and each key pressed echoed back to the user as an alert.

```
document.getElementById("keyExample").onkeydo
wn = function myFunction(e){
var keyPressed=e.keyCode; //get the raw key code
var character=String.fromCharCode(keyPressed); //convert to string

alert("Key " + character + " was pressed");
}
```

Form Events

Forms are the main means by which user input is collected and transmitted to the server. The events triggered by forms allow us to do some timely processing in response to user input. The most common JavaScript listener for forms is the onsubmit event. The different form events are -

Events	Description
onblur	A form element has lost focus (that is, control has moved to a different element), perhaps due to a click or Tab key press.
onchange	Some <input>, <textarea>, or <select> field had their value change. This could mean the user typed something, or selected a new choice

onfocus	This is triggered when an element gets focus (the user clicks in the field or tabs to it).
onreset	HTML forms have the ability to be reset. This event is triggered when reset (cleared).
onselect	When the users selects some text. This is often used to try and prevent copy/paste.
onsubmit	When the form is submitted this event is triggered. We can do some prevalidation when the user submits the form in JavaScript before sending the data on to the server

Eg: If the password field (with id pw) is blank, we prevent submitting to the

server
```
document.getElementById("loginForm").onsubmit = function(e){
        var pass =
        document.getElementById("pw").value;
        if(pass=="")
        {
                alert ("enter a
                password");
                e.preventDefault();
        }
}
```

Frame Events

Frame events are the events related to the browser frame that contains the web page. The most important event is the onload event, which triggers when an object is loaded. The frame events are -

Events	Description
onabort	An object was stopped from loading
onerror	An object or image did not properly load
onload	When a document or object has been loaded
onresize	The document view was resized
onscroll	The document view was scrolled
onunload	The document has unloaded

4. Demonstrate the comparison of pass by value and pass by reference in PHP with suitable diagram.

Parameter Default Values

In PHP, **parameter default values** can be set for any parameter in a function. However, once you start having default values, all subsequent parameters must also have defaults.

```
<?php
function getSum($x,$y=10)
{
                $z= $x+$y;
                return $z; // return $x+$y;
}
Echo “sum of 5 and6 is “.getSum(5,6);
Echo “<br />sum of 15 and 10 is
“.getSum(15); Echo “<br />sum of 12 and
14 is “.getSum(12,14); Echo “<br />sum of
25 and 10 is “.getSum(25);
?>
```

Output:
sum of 5 and6 is 11
sum of 15 and 10 is 25
sum of 12 and 14 is 26
sum of 25 and 10 is 35

Even if 2nd parameter value is not passed to the function, the default value 10 is taken. If the value is passed, the default will be overridden by whatever that value was.

Passing Parameters by Reference

By default, arguments passed to functions are **passed by value** in PHP. This means that PHP passes a copy of the variable so if the parameter is modified within the function, it does not change the original.
In the below example, notice that even though the function modifies the parameter value, the contents of the variable passed to the function remain unchanged after the function has been called.

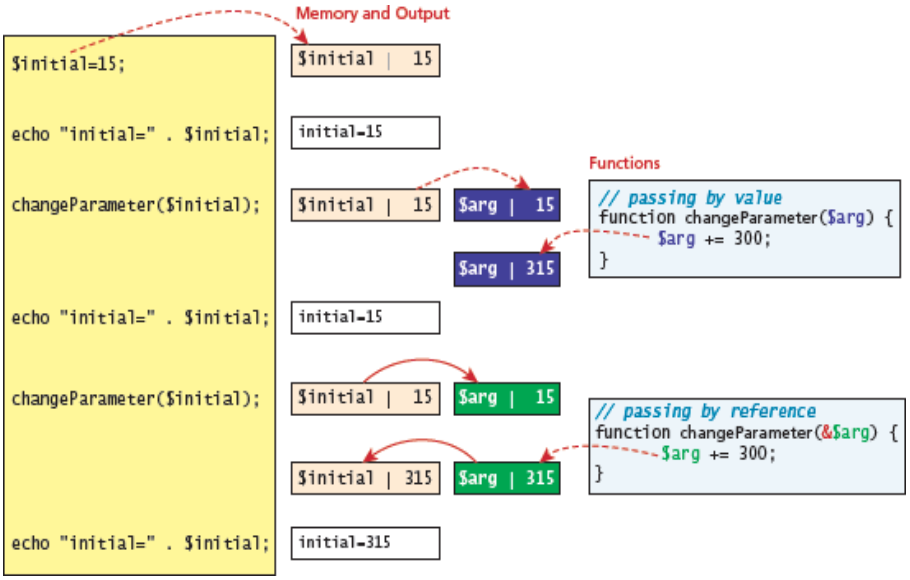
```
function changeParameter($arg) {
$arg += 300;
echo "arg=" . $arg;
}
$initial = 15;
echo "<br/>initial=" .
$initial;
changeParameter($initi
al);
echo "<br/>initial=" . $initial; // value not changed
```

PHP allows arguments to functions to be **passed by reference**, which will allow a function to change the contents of a passed variable. A parameter passed by reference points the local variable to the same place as the original, so if the function changes it, the original variable is changed as well. The mechanism in PHP to specify that a parameter is passed by reference is to add an ampersand (&) symbol next to the parameter name in the function definition.

Eg:

```
function changeParameter(&$arg) {
$arg += 300;
echo "arg=" . $arg;
}
$initial = 15;
echo "<br/>initial=" . $initial;
changeParameter($initial);
echo "<br/>initial=" . $initial; // value changed
```

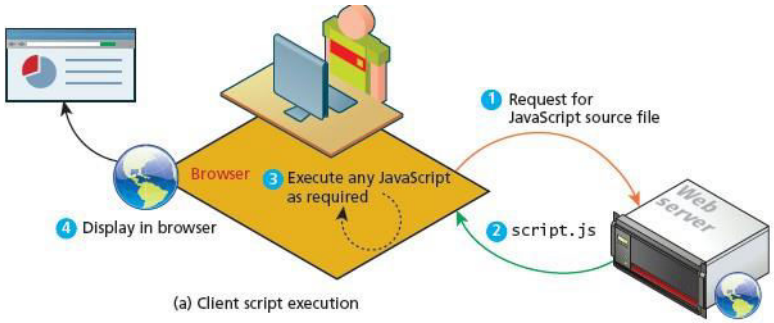
The memory differences between pass-by value and pass-by reference is illustrated below.



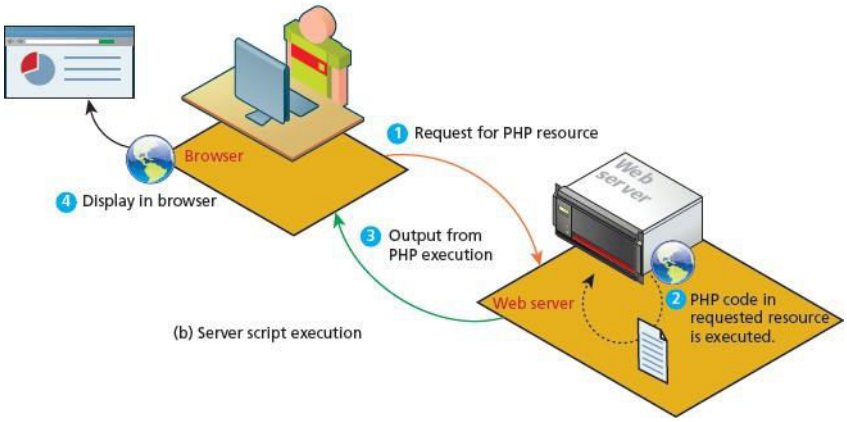
5. (a) Demonstrate the comparison of client-side execution and server script execution with suitable diagram.

Difference between Client and Server Scripts

Client-side Scripting	Server-side Scripting
The code is executed on the client browser	The code is executed on the web server
The requested script file is downloaded to the client.	The requested script file is hidden from the client as it is processed on the server
The requested file is downloaded at the client and executed.	The requested file is executed at the server and the output is sent to the client.
The client receives the script file.	The client will never see the script file, just the HTML output from the script, is sent to the client.
Client scripts can manipulate the HTML or DOM of a page in the client browser	Server scripts cannot manipulate the HTML or DOM of a page in the client browser
Client script cannot access resources(like database) present on the web server	Server script can access resources on the web server



(a) Client script execution



(b) Server script execution

(b) Explain in detail the fundamental syntax of JavaScript.

3.1 Syntax

Some of the features of javascript are:

- Everything is type sensitive, including function, class, and variable names.
- The scope of variables in blocks is not supported. This means variables declared inside a loop may be accessible outside of the loop.
- There is a === operator, which tests not only for equality but type equivalence.
- Null and undefined are two distinctly different states for a variable.
- Semicolons are not required, but are permitted.
- There is no integer type, only number, which means floating-point rounding errors are prevalent even with values intended to be integers.

Variables

Variables in JavaScript are dynamically typed, it means a variable can be an integer, and then later a string, then later an object. The variable type can be changed dynamically. The variable declarations do not require the type fields like *int*, *char*, and *String*.

The 'var' keyword is used to declare the variable, Eg: **var count;**
As the value is not defined the default value is undefined.

Assignment can happen at declaration-time by appending the value to the declaration, or at run time with a simple right-to-left assignment.

```
Eg: var
    count
    = 0;
    Var
    initial
    =2;
    Var b="Abhilash";
```

In addition, the conditional assignment operator, can also be used to assign based on condition. Eg: x= (y==1)? "yes": "no";

Variable x is assigned "yes", if value of y is 1, otherwise the value of x is "no".

Comparison Operators

The expressions upon which statement flow control can be based include primitive values, relational expression, and compound expressions. Result of evaluating a control expression is boolean value true or false.
In Javascript the comparison of two values are done by using the following operators-

Operator	Description	Matches (for x=9)
==	Equals	(x==9) is true (x=="9") is true

===	Exactly equals, including type	(x==="9") is false (x===9) is true
< , >	Less than, greater than	(x<5) is false
<= , >=	Less than or equal, greater than or equal	(x<=9) is true
!=	Not equal	(4!=x) is true
!==	Not equal in either value or type	(x!=="9") is true (x!==9) is false

Logical Operators

Many comparisons are combined together, using logical operators. Using logical operators, two or more comparisons are done in a single conditional checking operation. The logical operators used in Javascript are - && (and), || (or), and ! (not).

A	B	A && B
T	T	T
T	F	F
F	T	F
F	F	F

AND Truth Table

A	B	A || B
T	T	T
T	F	T
F	T	T
F	F	F

OR Truth Table

A	! A
T	F
F	T

NOT Truth Table

Functions

- A function definition consists of the function's header and a compound statement that describes the actions of the function. This compound statement is called the body of the function.
- A function header consists of the **reserved word function**, the function's name, and a parenthesized list of parameters (optional).
- The parentheses are required even if there are no parameters.
- Since JavaScript is dynamically typed, functions do not require a return type, nor do the parameters require type.
- A return statement returns control from the function in which it appears to the function's caller. Optionally, it includes an expression, whose value is returned to the caller. A function body may include one or more return statements. If there are no return statements in a function or if the specific return that is executed does not include an expression, the value returned is undefined. This is also the case if execution reaches the end of the function body without executing a return statement (an action that is valid).
- Syntactically, a call to a function with no parameters states the function's name followed by an empty pair of parentheses. A call to a function that returns undefined is a standalone statement. A call to a function that returns a useful value appears as an operand in an expression

Therefore a function to raise x to the yth power is
defined as: function power(x,y)
{

```
        var pow=1;
        for (var i=0;i<y;i++)
        {
                pow = pow*x;
        }
        return pow;
}
```

It is called as

power(2,10);

6. Write the XHTML code to create the form with the following capabilities

a) A text widget to collect the users name

b) Four check boxes, one each for the following items

i) Four 100 watt light bulbs for Rs. 20=39

ii) Eight 100 watt light bulbs for Rs 40=20

iii) Four 100 watt long life light bulbs for Rs. 30=95

iv) Eight 100 watt long life light bulbs for Rs 70=49

c) A collection of 3 radio buttons that are labeled as follows

i) Visa

ii) Master Card

```
<html>
<head>
<title>order form</title>
</head>
<body>
<form method="POST" action="bulbs.php">
user name<input type="text" name="myname" size="30"/><br>
select the items:<br>
<input type="checkbox" name="box" value="20.39" checked="checked"/>
fore100 watt light bulbs </br>
<input type="checkbox" name="box" value="40.20">
eight 100 watt light bulbs </br>
<input type="checkbox" name="box" value="30.95">
four 100 watt long life bulbs </br>
<input type="checkbox" name="box" value="70.49">
eight100 watt long life bulbs </br>
Select the mode of the poyment:</br>
<input type ="radio" name="paymode" value="visa" checked="checked"/>Visa<br>
<input type ="radio" name="paymode" value="Master card"/>Master card<br>
<input type ="radio" name="paymode" value="Discover"/>Discover<br>
<input type="submit" value="submit order"/>
<input type="reset" value="Clear the form"/> </body>
</html>
```