**SET – I**

**Q. No 1. What is Web design? List and explain the common issues to be considered while designing the web site**

**Explanation for Web design**

**List and explain the common issues raised during web design**

**Explanation for Web design =>**

Web designing can be considered as a collection of skills and disciplines used to design effective websites. While designing a website, the designer can use his or her imagination and designing skills creatively on the screen. The most important aspect to keep in mind while designing a website is the goal of the website. The goal of a website refers to the purpose for which it is created. For example, if the business of an organization is to sell books, then the website must provide information about different books, including their bibliographical details, price, discount, payment mode, and shipping details. The homepage generally includes basic links such as Home, About Us, Our Brands, and Contact Us. Other required links should include options such as Shopping Cart, Order History, Help, etc. If the user clicks any of these links, he or she should be taken to the relevant location in the website containing information related to the clicked link. Each page of the website will be consistent in terms of its design. In other words, elements such as the colour scheme, and the placement and design of objects such as buttons, menus, fonts, etc., used in the website. should have a consistent appearance. The logo of an organisation appears on every page of its website. It should have the uniform size, colour, and design elements. This consistency in the design works as a unifying force and helps to make the website a cohesive entity.

**List and explain the common issues raised during web design =>**

Some of the most common issues that are considered while designing a website relate to content, page layout, usage of colours, styling (HTML and CSS), etc.

These issues can be briefly explained as follows:

* **Content:**

When a user visits any website for some information, he/she usually gets only 50-60% of the information. This is because most of the space of any Web page focuses on displaying advertisements and other similar elements. In addition, confusing or complex navigation followed by some websites makes it difficult for the user to find the information they are looking for, even if it is present on the website. Therefore, while designing a website, make sure that it provides relevant and sufficient information.

* **Page layout:**

This issue of page layout is closely related to the issue of content as page layout depends on the type of content we have. When a user visits a website, he or she expects to find the content he/she is looking for as quickly as possible. If the user does not find the content, or finds the process long, complicated, or tiresome, he/she is likely to leave the website. Therefore, as a developer, we need to ensure that the various elements on a Web page can be easily seen and understood, and provide the correct information quickly on clicking. In addition to this, a Web page must not be too cluttered as it might make the navigation of the page difficult or confusing. Therefore, it is best to keep the pages of the website simple and clean.

* **Colour:**

Some standard colours are used in designing the pages of a website. For example, blue is used for unclicked links and violet is used for clicked ones (that is, the ones that the user has already visited). Using any other colours to represent these links may confuse the user. Apart from this, use colours that are pleasing to the eyes on a Web page.

* **Hyper Text Markup Language (HTML):**

The appearance of a website is generally decided through the use of HTML code. Use correct and valid code to make sure that all parts of our website are fully functional and displaying the correct information. Also, make sure that our Web browser supports the latest version of HTML, that is, HTML 5.

* **Cascading Style Sheets (CSS):**

CSS or simply style sheets are text files that contain one or more rules in the form of property/value pairs to determine how the elements in a Web page should be displayed. In other words, CSS is used to style a Web page. Use CSS style properties carefully and make sure that the chosen style complements the content on the Web page. CSS is used to choose how the content needs to be displayed, what colours to use, and the appearance of the various page elements. The style should be attractive and help to present the information on the Web page as a single cohesive unit.

**Q. No 2. What is event in HTML. Explain the common JavaScript event and its role**

**Event in HTML**

**List and explain common JavaScript event**

**Event in HTML =>**

Events are another common and popular way to incorporate interactivity in a web page. They refer to a particular action performed on the web page by a user. For instance, an event can occur when the user clicks a hyperlink on the web page. There are some events that occur without the user’s participation, for example, an event occurs when a web page is fully loaded in the web browser.

**List and explain common JavaScript event =>**

JavaScript defines several in-built events to assist we in making the web pages dynamic and interactive. When any of these events occur, its event handler is called. An event handler is a special function that handles a particular event. In JavaScript, we can create an event handler and then associate it with the desired event. The event handlers are associated with the events through certain attributes of various HTML tags.

* **Click =>** Occurs when the user clicks an HTML element, such as form element, an image, or a link
* **Dblclick =>** Occurs when the user double-clicks an HTML element
* **Keydown =>** Occurs when the user has pressed a key on the keyboard
* **Keypress** **=>** Occurs when the user presses a key on the keyboard for a while
* **Keyup** **=>** Occurs when the user has released a key on the keyboard
* **Load** **=>** Occurs when a web page or an image is completely loaded in the web browser
* **Mousedown** **=>** Occurs when the user has just pressed a mouse button
* **Mouseover** **=>** Occurs when the user moves the mouse on top of a form element
* **Mouseup** **=>** Occurs when the user has just released a mouse button
* **Submit** **=>** Occurs when the user submits an HTML form by pressing the submit button

**Q. No 3. What is string class? Explain the constructors in string class**

**String class**

**Constructors in string class**

**String class =>**

Strings in Java are immutable, as once an object of the String class is created and initialized with a value, the value cannot be changed. However, we may sometimes need to manipulate the value assigned to a String object. For this, Java provides the StringBuffer class.

Strings are very important in the world of programming languages as they are used to process long textual/symbolic information. For example, when we visit a bank to open an account, we are asked to provide information, such as our name and address. The information we provide is in the form of an ordered sequence of characters and symbols. This ordered sequence of characters and symbols is known as a string.

In programming languages, such as C and C++, strings are considered fundamental data types. In Java, strings are not considered fundamental data types. Instead, in Java, the String class is provided to work with strings as well as with character arrays. In Java, we can either use character arrays or the String class as per our requirement.

**Constructors in string class =>**

In Java, the int primitive type stores integers and the float primitive type stores floating-point numbers, but there is no primitive type to store string data. We need objects of the String class to handle string data in Java. The String class is included in the java.lang package. As this package is imported by default, each class in Java can use the String class. Some of the most commonly used constructors of the String class are listed below:

* **String()**:

Constructs a String object with no value. For example, String str= new String();

* **String(String str)**:

Constructs and initializes a String object with the given string value. For example, String str=new String(“Hello”); The str object now contains Hello.

* **String(char charArr [ ]):**

Constructs and initializes a String object with the values of the given character array. For example, char item [ ] = {‘C’, ‘O’, ‘P’, ‘Y’}; String testString = new String(item);

The testString object now contains the value stored in the character array named item.

* **String(char charArr[ ], int startIndex, int count):**

Constructs and initializes a String object with a part of a character array. For example, char inputArray [ ] = {‘P’,’U’,’S’,’H’,’P’, ‘O’,’P’};

String testArr = new String(inputArray,4, 3);

The testArr object now contains the value POP because the starting index is 4 and the total number of characters to be copied is 3.

* **String(byte byteArr[ ]):**

Constructs and initializes a String object with an array having American Standard Code for Information Interchange (ASCII) values. Apart from constructing and initializing a String object, this constructor also decodes the given array values according to the system’s default character set.

* **String(byte byteArr[ ], int startIndex, int count):**

Constructs and initializes a String object with a part of an array having ASCII values. This constructor also decodes ASCII values according to the system’s default character set.

Apart from using a constructor to create and initialize a String object, we can use the following code snippet to directly create and initialize a String object:

String str = “Hello”;

The preceding code snippet creates a String object, str and initializes it with the value, Hello. We can also assign the value of a String object into another String object. Consider the following code snippet to assign value of a String object to another String object:

String str1=”How”;

String str2=”Are”;

str1=str2; // now the str1 contains the value Are

**Code to Create Strings Using a Constructor of the String:**

class StringConstructor{

public static void main(String args[]){

String str= new String();

str="Deepak";

System.out.println(" The value stored in str is " + str);

String str1= new String("Gupta");

System.out.println(" The value stored in str1 is " + str1);

char ch []= {'T','W'};

String str2 = new String(ch);

System.out.println(" The value stored in str2 is " + str2);

byte byteArr[]={95,65,74};

String str3=new String(byteArr);

System.out.println(" The value stored in str3 is " + str3);}

}

**SET - II**

**Q. No 1. Define DOM. Explain the following HTML Tags with its syntax and example**

**i. <head> tag**

**ii. <title> tag**

**iii. <body> tag**

**DOM**

**<head> tag**

**<title> tag**

**<body> tag**

**DOM =>**

DOM is a cross-platform, language-independent interface that allows programs and scripts to dynamically access and update the content, structure, and style of HTML or XML documents. Every element in an HTML document represents a DOM node. These nodes are related to each other through a parent-child relationship. Although DOM is designed to work independently of a programming language, it is often used with JavaScript to design dynamic and interactive HTML pages.

In DOM, all the properties, methods, and events that are available to a web developer for manipulating and creating web pages are organized into objects, called DOM objects. Depending on the type of objects, DOM is divided into three parts—Core, XML, and HTML. Core DOM includes the objects that are present in both XML and HTML documents. XML and HTML DOMs contain XML and HTML objects, respectively.

The structure of an HTML document is made of several elements and attributes such as !DOCTYPE, html, head, and body.

**<head> tag =>**

The <head> tag contains general information about the HTML document, such as its title, keywords for search engines, and a base address for URLs.

* **Attributes of the <head> Tag:**
* **class** **=>** Specifies the class for an element in an HTML document
* **id** **=>** Represents a unique alphanumeric identifier for an element
* **style** **=>** Represents the inline style for rendering an element
* **title** **=>** Specifies the title of the HTML document

**<title> tag =>**

The <title> tag contains the title of the HTML document. It appears in the title bar of the web browser and is used by search engines to refer to the document. Each <head> tag should contain a <title> tag. Try to keep the title text relatively short and to the point because some browsers find it difficult to handle titles that are longer than 256 characters.

* **Attributes of the <title> Tag:**
* **class** **=>** Represents the class of an element and is used to render content
* **id** **=>** Refers to a unique alphanumeric identifier for an element
* **lang** **=>** Represents the base language used for an element
* **style** **=>** Represents the inline style for rendering an element

**<body> tag =>**

The <body> tag contains the body of the HTML document and it includes the entire content that will appear in the web browser. It can also include text, images, and multimedia elements.

* **Attributes of the <body> Tag:**
* **alink** **=>** Specifies the colour of hyperlinks when they are clicked. We can set this attribute to a pre-defined colour name or value.
* **background** **=>** Represents the Uniform Resource Locator (URL) of a graphic file (image) that is used as the background of the web page.
* **bgcolor** **=>** Specifies the background colour of a web page. We can set this attribute to a pre-defined colour name or value.
* **class** **=>** Represents the class of an element and is used to render content.
* **id** **=>** Refers to a unique alphanumeric identifier for an element.
* **link** **=>** Specifies the colour of hyperlinks that have not yet been visited by a user. We can set this attribute to a pre-defined colour name or value.
* **style** **=>** Specifies whether a vertical scrollbar appears to the right of the document. we can set this attribute to either yes (which is the default value for the attribute) or no.
* **text** **=>** Specifies the colour of the text in a document. We can set this attribute to a pre-defined colour name or value.
* **Title** **=>** Holds additional information for an element, similar to tooltips.

**Q. No 2. What is function in Java. Explain the following concept of function**

**Passing parameter to methods**

**Returning value from Methods**

**Function in Java**

**Passing parameter to methods**

**Returning value from Methods**

**Function in Java =>**

Methods are functions that are built into the class, and therefore, built into the objects we create from that class. We usually divide methods into different categories on the basis of their access specifiers, that is, private, public, and protected. In private, the methods are intended to be used inside a class, public outside a class, and protected by the class from where they are derived.

Java supports two types of methods: class methods and instance methods. Instance methods are invoked by the objects of a class. Class methods, on the other hand, are invoked by a class.

* **Declaring Methods:**

A Java class consists of attributes and methods. Methods can have any name, such as disp and vol, and are considered as functions that are used to perform some specific tasks in an application. Depending on the requirement, many methods can be created in a program. The following

syntax shows us, how to create a method:

<return\_type> <method\_name> ([para\_list])

{

// method body;

}

In the preceding syntax, return type can be any data type depending on the type of values that a method is returning. If the method is not returning any value, then the data type is replaced with void. A method after its invocation returns a value through the return statement. In the syntax, method name is the name of the method, and para\_list represents a list of parameters that are passed in the method. In some cases, we may not need to use parameters in a method. In such cases, para\_list is not used, and the methods return a null value. The body of a method is written within curly braces.

**Passing parameter to methods =>**

When we declare a method, we can specify a comma-separated list of parameters that we want to pass to that method in parentheses following the method’s name, as shown in the following syntax:

[access] [static] type method1 ([type parameter\_name1 , [type

parameter\_name1...]]){

.

.

}

In the preceding syntax, the parameters passed to the method are accessible in the body of the method.

If we have more than one parameter to pass, we can specify multiple parameters in the parameter list, separated by commas, as shown in the below:

class Calculator{

int addem(int op1, int op2)

{

int result = op1 + op2;

.

.

}

}

We can call methods by passing literals, variables, arrays, or objects, as shown in the following statement:

calc.addem(1,int1, array1, obj1)

When we pass a simple variable or literal to a method, the value of the variable/literal is passed to the method. This process is called passing by value. On the other hand, when we pass an object or array, we are passing a reference to that object or array (in fact, when we store an array or object in a variable, what we are actually storing is a reference to the array or object. For this reason, the code in the called method has direct access to the original array or object, not a copy of it. Therefore, if that code changes some aspect of the array or object, such as an element in the array or a data member of the object, the original array or object is changed.

**Returning value from Methods =>**

We can use the return statement in a method to return a value from the method. We can also indicate the return type of a method when we declare the method, as shown in the following syntax:

[access] [static] type method1 ([type parameter\_name1 ,[ type

parameter\_name1...]]){

.

.

}

In the preceding syntax, the return type can be any type that Java recognizes—for example, int, float, double, the name of a class we have defined, int[ ] to return an integer array, or float[ ] to return a float array.

For example, in Listing 5.1, the class Calculator has a method named addem(), which takes two integer parameters, adds them, and returns the result:

Code to Add Two Numbers and Return their Sum from a Method:

class Example2{

int addem(int op1, int op2){

return op1 + op2;

}

}

public class Calculator{

public static void main(String[] args){

example2 calc = new example2();

System.out.println("addem(2, 2) = " + calc.addem(2, 2));}

}

**Q. No 3. Describe the following graphics layout Managers and its methods**

**i. GridLayout Manager**

**ii. CardLayout Manager**

**Grid Layout Manager**

**CardLayout Manager**

Java provides various layout managers that can be used by Java programmers. Layout managers are used to arranging the components, such as button and radio button, in a frame window. A layout manager implements the LayoutManager interface of the java.awt package to arrange the components in a frame window. The setLayout() method is used to set a particular layout for a window. For example, if we want to set the Border layout to arrange the components in a window, then we can apply the following code snippet:

setLayout(new BorderLayout());

**Grid Layout Manager =>**

The GridLayout manager divides a container into a two-dimensional grid that contains several rows and columns. In a GridLayout, each component is given the same size and dimension. The Java class, which is used to implement the GridLayout in frame window and applets, is known as the GridLayout class.

**Constructors of the GridLayout class:**

* **GridLayout ()**

Creates a grid layout having one column per component in one row.

* **GridLayout (int rows, int cols)**

Creates a grid layout with a specific number of rows and columns.

* **GridLayout (int rows, int cols, int hgap, int vgap)**

Creates a grid layout with specific number of rows and columns, along with specific horizontal and vertical gap between rows and columns.

**Methods of the GridLayout Class:**

* **void setHgap (int hgap)**

Sets the horizontal gap between components.

* **void setVgap (int vgap)**

Sets the vertical gap between components.

* **void setColumns (int cols)**

Sets the specified number of columns in the layout.

* **void setRows (int rows)**

Sets the specified number of rows in the layout.

**CardLayout Manager =>**

The CardLayout manager displays the containers we pass to it as cards. We give each card a name, then, we can move from card to card with the card layout’s show method. Besides the show(Container parent, String name) method, we can also display specific cards by using the first(Container parent), last(Container parent), next(Container parent), and previous(Container parent) methods of the CardLayout class.

**Constructors of the CardLayout Class:**

* **CardLayout ()** - Creates a new card layout.
* **CardLayout (int hgap, int vgap)** - Creates a new card layout with the given horizontal and vertical gaps.

**Methods of the CardLayout Class:**

* **void addLayoutComponent(Component comp, Object constraints)**

Adds the given component with the given name to the layout.

* **void first(Container parent)**

Goes to the first card of the container.

* **int getHgap()**

Gets the horizontal gap between components.

* **float getLayoutAlignmentX(Container parent)**

Gets the alignment along the x-axis.

* **float getLayoutAlignmentY(Container parent)**

Gets the alignment along the y-axis.

* **int getVgap()**

Gets the vertical gap between components.

* **void invalidateLayout(Container target)**

Invalidates the layout.

* **void last(Container parent)**

Goes to the last card of the container.

* **void layoutContainer(Container parent)**

Lays out the given container using this CardLayout.

* **Dimension maximumLayoutSize(Container target)**

Gets the maximum dimensions for the layout, given the components in the given target container.

* **Dimension minimumLayoutSize(Container parent)**

Calculates the minimum size for the given panel.

* **void next(Container parent)**

Goes to the next card of the given container.

* **Dimension preferredLayoutSize(Container parent)**

Determines the preferred size of the container argument by using this CardLayout.

* **void previous(Container parent)**

Goes to the previous card of the given container.

* **void removeLayoutComponent(Component comp)**

Removes the given component from the layout.

* **void setHgap(int hgap)**

Sets the horizontal gap between components.

* **void setVgap(int vgap)**

Sets the vertical gap between components.

* **void show(Container parent, String name)**

Goes to the component that was added to the layout with the given name.

* **String toString()**

Gets a string representation of this layout.