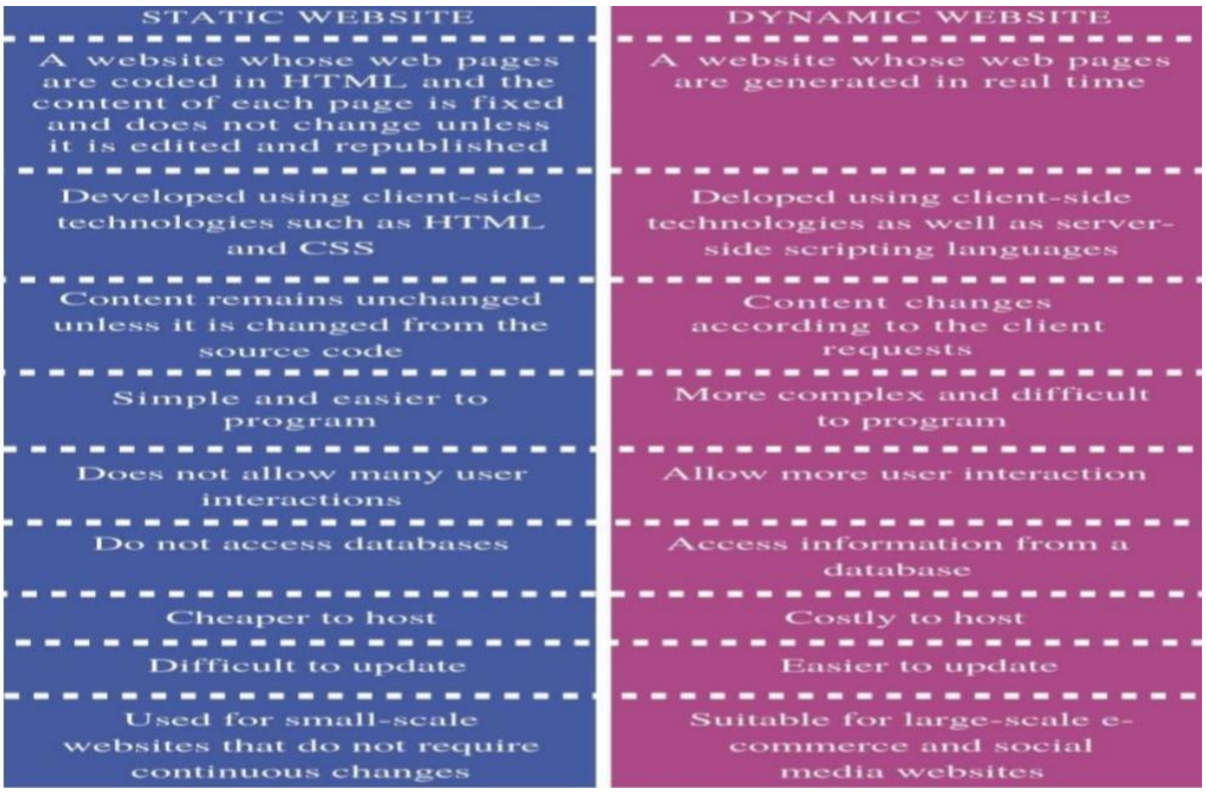
WDD FINAL ANSWERS

UNIT-1

1. Write any five differences between static and dynamic websites?



STATIC WEBSITE	DYNAMIC WEBSITE
A website whose web pages are coded in HTML and the content of each page is fixed and does not change unless it is edited and republished	A website whose web pages are generated in real time
Developed using client-side technologies such as HTML and CSS	Deloped using client-side technologies as well as server-side scripting languages
Content remains unchanged unless it is changed from the source code	Content changes according to the client requests
Simple and easier to program	More complex and difficult to program
Does not allow many user interactions	Allow more user interaction
Do not access databases	Access information from a database
Cheaper to host	Costly to host
Difficult to update	Easier to update
Used for small-scale websites that do not require continuous changes	Suitable for large-scale e-commerce and social media websites

2.What are the advantages of using HTML5?

Ans: HTML stands for Hyper Text Markup Language which is used for creating web pages and web applications. Let's

See what is meant by Hypertext Markup Language, and Web page.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click On a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or More web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting Conventions to a text document.

Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc

HTML5 advantages: The following are the advantages of HTML5.

1. Cleaner markup/ Improved code:

HTML5 will enable web designers to use cleaner, neater code. We can remove div tags and replace them with semantic HTML5 elements.

2. Consistency:

As websites will adopt the new HTML5 elements we will see more consistency in terms of HTML used to code a web page on one site compared to another. This will make it much easier for designers and developers to immediately understand how a web page is created.

3. Supports rich media elements:

HTML5 has an inbuilt capability to play audio and video and so we can bid goodbye to those plugin tags.

4. Offline Application Cache:

HTML5 offers an offline application cache facility which will load the page the user has visited even if the user is temporarily offline. This feature will help the files to load much faster and reduces load on server.

HTML is widely used

.Every browser supports HTML Language

.Easy to learn and use

.HTML is light weighted and fast to load

.Do not get to purchase any extra software because it’s by default in every window

.Easy to use

.Loose syntax (although, being too flexible won’t suit standards).

3.Define web hosting and its types.

Ans: Web hosting is a service of providing online space for storage of web pages. These web pages Are made available via World Wide Web. The companies which offer website hosting are Known as Web hosts.

The servers on which web site is hosted remain switched on 24 x7. These servers are run by Web hosting companies. Each server has its own IP address. Since IP addresses are difficult to Remember therefore, webmaster points their domain name to the IP address of the server their Website is stored on.

Types of Hosting :

Shared Hosting: In shared hosting, the hosting company puts thousands of websites on the same physical server. Each customer has their own allocation of physical web space and a set of bandwidth limit.

Virtual Private Server (VPS): It is also known as Virtual Dedicated Server. It is a server which is partitioned into Smaller servers. In this customer is given their own partition, which is installed with its own operating system.

Dedicated Server: In this kind of hosting, single dedicated server is setup for just one customer. It is commonly used By the businesses that need the power, control and security that a dedicated server offers.

Reseller Hosting: A reseller acts as a middle man and sells hosting space of someone else’s server.

Grid Hosting: Instead of utilizing one server, Grid Hosting spreads resources over a large number of servers. It is

Quite stable and flexible. The servers can be added or taken away from the grid without crashing the system.

4.Write a HTML program to create a time table?

5.Explain some of the common lists to design a web page.

Ans: HTML Lists: HTML Lists are used to specify lists of information. All lists may contain one or more list elements.

There are three different types of HTML lists:

1.Ordered List or Numbered List (ol)

2.Unordered List or Bulleted List (ul)

3.Description List or Definition List (dl)

HTML Ordered List or Numbered List:

In the ordered HTML lists, all the list items are marked with numbers by default. It is known as numbered list also.

The ordered list starts with <ol> tag and the list items start with <li> tag

The type Attribute

You can use type attribute for <ol> tag to specify the type of numbering you like. By default, it is a number. Following are the possible options –

<ol type = “1”> ,<ol type = “I”>,<ol type = “I”>,<ol type = “A”>,<ol type “a”>.

The start Attribute

You can use start attribute for <ol> tag to specify the starting point of numbering you need. Following are the Possible options –

<ol type = “1” start = “4”>

Example:

```
<html>
<head>
<title>HTML unordered List</title>
```

```
</head>
<body>
<ol type="A" start="3">
<li>SOE</li>
<li>SOA</li>
<li>SOMs</li>
<li>SOS</li>
</ol>
</body>
</html>
```

Output:

```
C. SOE
D. SOA
E. SOMs
F. SOS
```

HTML Unordered Lists

An unordered list is a collection of related items that have no special order or sequence. This list is created By using HTML <ul> tag. Each item in the list is marked with a bullet.

The type Attribute

You can use type attribute for <ul> tag to specify the type of bullet you like. By default, it is a disc.

Following are the possible options –

<ul type = “square”>,<ul type = “disc”>,<ul type = “circle”>

Example:

```
<html>
<head>
<title>HTML unordered List</title>
<ul type=”square”>
<li>SOE</li>
<li>SOA</li>
<li>SOMs</li>
<li>SOS</li>
</ul>
</body>
</html>
```

Output:

```
▯ SOE
▯ SOA
▯ SOMs
▯ SOS
```

HTML Definition Lists

HTML and XHTML supports a list style which is called definition listswhere entries are listed like in a dictionary or Encyclopedia. The definition list is the ideal way to present a glossary, list of terms, or other name/value list.

Definition List makes use of following three tags.

- **<dl> – Defines the start of the list**
- **<dt> – A term**
- **<dd> – Term definition**
- **</dl> – Defines the end of the list**

Example:

```
<html>
<head>
<title>HTML Definition List</title>
</head>
<body>
<dl>
<dt><b>HTML</b></dt>
<dd>This stands for Hyper Text Markup Language</dd>
<dt><b>HTTP</b></dt>
<dd>This stands for Hyper Text Transfer Protocol</dd>
</dl>
</body>
</html>
```

Output:

```
HTML
This stands for Hyper Text Markup Language
HTTP
This stands for Hyper Text Transfer Protocol
```

6.How do you validate a form in HTML with suitable example program?

Ans. An HTML form is a section of a document which contains controls such as text fields, password Fields, checkboxes, radio buttons, submit button, menus etc.

An HTML form facilitates the user to enter data that is to be sent to the server for processing Such as name, email address, password, phone number, etc.

HTML forms are required if you want to collect some data from of the site visitor.

HTML Form Syntax:

```
<form action=”server url” method=”get|post”>
//input controls e.g. textfield, textarea, radiobutton, button
</form>
```

The <form> Element

The HTML <form> element is used to create an HTML form for user input:

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.

HTML TextField :

The type=”text” attribute of input tag creates textfield control also known as single line textfield Control.

HTML <textarea> tag in form:

The <textarea> tag in HTML is used to insert multiple-line text in a form. The size of <textarea> Can be specify either using “rows” or “cols” attribute

Radio Button :

The radio button is used to select one option from multiple options. It is used for selection of Gender, quiz questions etc.

Checkbox :

The checkbox control is used to check multiple options from given checkboxes.

Submit button:

HTML <input type=”submit”> are used to add a submit button on web page. When user clicks On submit button, then form get submit to the server

HTML program for registration form:

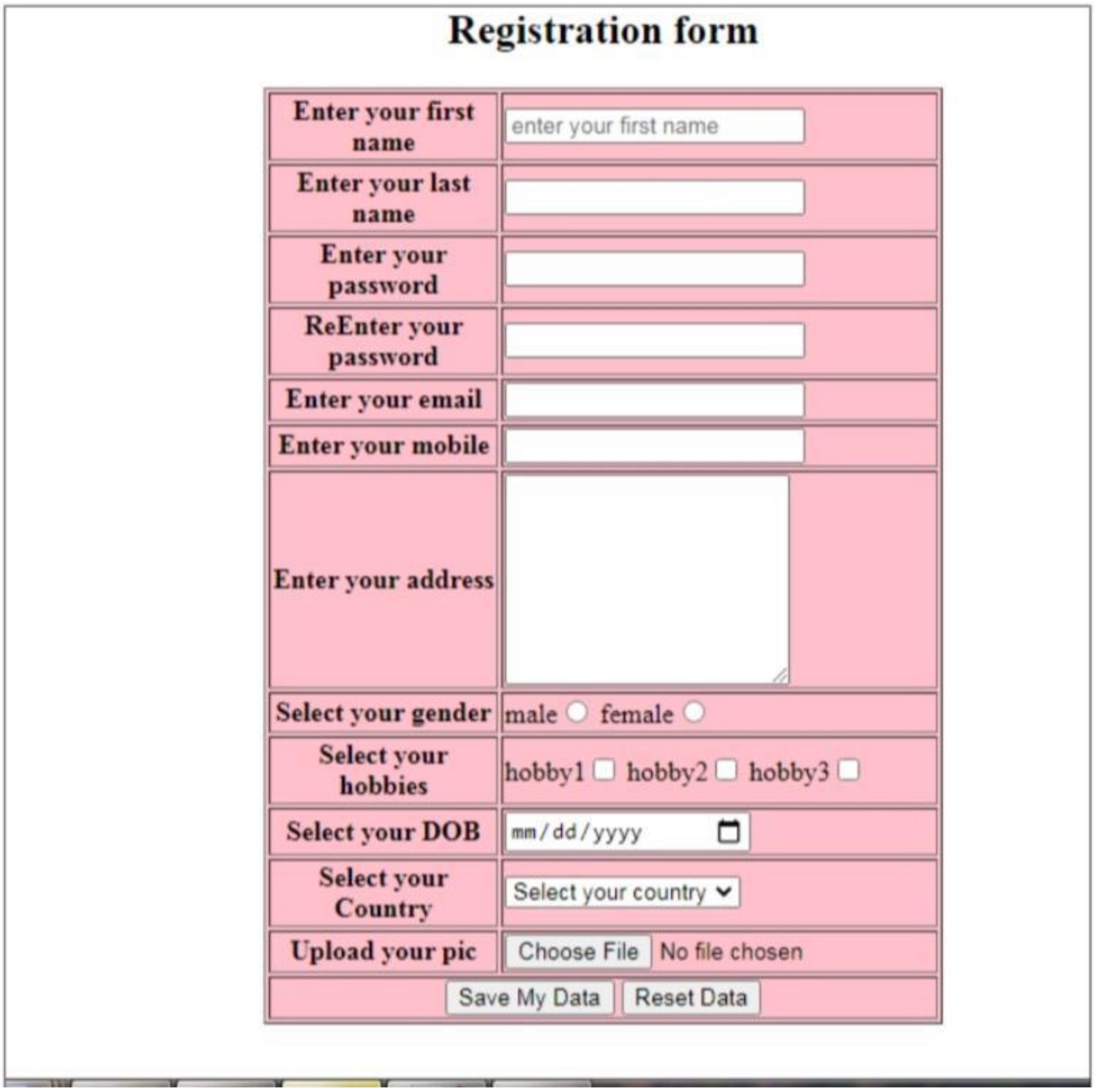
```
<html>
<head>
<title>HTML Table</title>
</head>
<body>
<form method=”” action=””>
<table border=”1” align=”center” width=”400” bgcolor=”pink” >
<caption><h2>Registration form</h2></caption>
<tr>
 <th>Enter your first name</th>
<td><input type=”text” name=”fn” id=”fn1” maxlength=”10”
Title=”enter your first name” placeholder=”enter your first name”
required/></td>
</tr>
<tr>
<th>Enter your last name</th>
 <td><input type=”text”/></td>
</tr>
```

```
<tr>
 <th>Enter your password</th>
<td><input type=”password”/></td>
</tr>
<tr>
<th>ReEnter your password</th>
<td><input type=”password”/></td>
</tr>
<tr>
<th>Enter your email</th>
<td><input type=”email”/></td>
</tr>
<tr>
<th>Enter your mobile</th>
<td><input type=”number”/></td>
</tr>
<tr>
<th>Enter your address</th>
<td><textarea rows=”8” cols=”20”></textarea></td>
</tr>
<tr>
<th>Select your gender</th>
<td>
Male<input type=”radio” name=”g” value=”m”/>
Female<input type=”radio” name=”g” value=”f”/>
</td>
</tr>
<tr>
```

```
<th>Select your hobbies</th>
<td>hobby1<input type=”checkbox” name=”x[]” value=”h”/>
Hobby2<input type=”checkbox” name=”x[]” value=”h2”/>
Hobby3<input type=”checkbox” name=”x[]” value=”h3”/>
</td>
</tr>
<tr>
<th>Select your DOB</th>
<td><input type=”date”/></td>
</tr>
<tr>
<th>Select your Country</th>
<td>
<select name=”country”><option value=”” selected=”selected”
disabled=”disabled”>Select your country</option>
<option value=”1”>India</option>
<option value=”2”>Pakistan</option>
</select>
</td>
</tr>
<tr>
<th>Upload your pic</th>
<td><input type=”file”/></td>
</tr>
<tr>
<td colspan=”2” align=”center”><input type=”submit” value=”Save My
Data”/><input type=”reset” value=”Reset Data”/>
</td>
```

```
</tr>
</table>
</form>
</body>
</html>
```

Output:



Registration form

Enter your first name	enter your first name
Enter your last name	
Enter your password	
ReEnter your password	
Enter your email	
Enter your mobile	
Enter your address	
Select your gender	male ○ female ○
Select your hobbies	hobby1 ☐ hobby2 ☐ hobby3 ☐
Select your DOB	mm/dd/yyyy
Select your Country	Select your country
Upload your pic	Choose File No file chosen
Save My Data Reset Data	

UNIT-2

1.Explain the different types of selectors in CSS with suitable example.

Ans: CSS Selectors

A CSS selector is the part of a CSS rule set that actually selects the content to which we want to apply the style. Let's Look at all the different kinds of selectors available, with a brief description of each.

(i)Universal Selector

The "universal selector" works like a wild card character, selecting all elements on a page. Every HTML page is built On HTML tags. Each set of tags represents an element on the page.

CSS example, which uses the universal selector:

```
<style>
.{color: green;
Font-size: 20px;
Line-height: 25px;
}</style>
```

(ii) Element Type Selector

Also referred to simply as a "type selector," this selector must match one or more HTML elements of the same Name.

Example: A selector of <ul> would match all HTML unordered lists, or <ul>elements. (i.e)

The following example uses An element type selector to match all <ul> elements:

```
ul {
List-style: none;
Border: solid 1px #fff;}
```

(iii)ID Selector

An ID selector is declared using “a hash, or pound symbol (#)” preceding a string of characters. The string of Characters is defined by the developer. This selector matches any HTML element that has an ID attribute with the Same value as that of the selector, but minus the hash symbol.Here’s an example:

```
#container {
Width: 960px;
Margin: 0 auto;
}
```

(iv)Class Selector

The class selector is the most useful of all CSS selectors. It’s declared with a dot preceding a string of one or more

Characters. He class selector also matches all elements on the page that have their class attribute set to the same

Value as the class, minus the dot. Take the following rule set:

```
.box {padding: 20px;
Margin: 10px;
Width: 240px;}
```

(v) Attribute Selector

The attribute selector targets elements based on the presence and/or value of HTML attributes, and is declared

Using square brackets:

Input[type="text"] {

Background-color: #444;

Width: 200px;}

There should not be a space before the opening square bracket unless you intend to use it along with a descendant Combinator.

2. How to add CSS in HTML pages to format the document according to information in the style sheet with types?

Ans: css

To apply the style and lay outs to the web pages — for example, to alter the font, colour, size and spacing of your content, split it into multiple columns, or add animations and other decorative features.For that purpose we can use "Cascading Style Sheets(CSS)".

->CSS can be added to HTML elements in 3 ways:

1. Inline CSS -

Def: Using the <style> attribute with in HTML elements or HTML tags is called "Inline CSS".

➔ An Inline CSS is used to apply a unique style to a single HTML element.

➔ This example sets the text color of the <h1> element to blue:

Example:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;">This is a Blue Heading</h1>
</body>
</html>
```

output:

```
This is a Blue Heading
```

2. Internal CSS Def: Using a <style> element in the <head> section is called "Internal CSS".

➔ An Internal CSS is used to define a style for "a single HTML page".

➔ An internal CSS is defined in the <head> section of an HTML page, within a <style> element.

Example:

```
<html>
<head>
<style>
body {background-color: powderblue;}
h1 {color: blue;}
p {color: red;}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
 </html>
```

Output:

```
This is a heading
This is a paragraph.
```

3. External CSS- Using a <style> element in an external CSS file is called "External CSS".

➔ An external style sheet is used to define the style for "many HTML pages".

➔ With an external style sheet, you can change the look of an entire web site, by changing one file.

Example:

To use an external style sheet, add a link to it in the <head> section of the HTML page:

```
<html>
<head>
<link rel="Stylesheet" href="Styles.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

➔ An external style sheet can be written in any text editor. The file must not contain any HTML code, and must be saved with a .css extension.

Here is how the "Styles.css" looks:

```
body {
background-color: red;}
h1 {color: blue;}
p {color: red;}
This is a heading
This is a paragraph.
```

3.To discuss the CSS background property is used to define the background effects on element and there are 5 CSS background properties that affect the HTML elements.

Ans. CSS background property is used to define the background effects on element. There are 5 CSS background properties that affects the HTML elements:

1. background-color

2. background-image
3. background-repeat
4. background-attachment
5. background-position

1) CSS background-color

The background-color property is used to specify the background color of the element.

You can set the background color like this:

```
<!DOCTYPE html>
<html>
<head>
<style>
h2,p{
    background-color: #b0d4de;
}
</style>
</head>
<body>
<h2>My first CSS page.</h2>
<p>Hello Javatpoint. This is an example of CSS background-color.</p>
</body>
</html>
```

My first CSS page.

Hello Javatpoint. This is an example of CSS background-color.

2) CSS background-image

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

background-repeat: repeat-x;

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("gradient_bg.png");
    background-repeat: repeat-x;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
</body>
</html>
```

3) CSS background-repeat

By default, the background-image property repeats the background image horizontally and vertically. Some images are repeated only horizontally or vertically.

The background looks better if the image repeated horizontally only.

background-repeat: repeat-x;

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
    background-image: url("gradient_bg.png");
    background-repeat: repeat-x;
}
</style>
</head>
<body>
<h1>Hello Javatpoint.com</h1>
</body>
</html>
```

4) CSS background-attachment

The background-attachment property is used to specify if the background image is fixed or scroll with the rest of the page in browser window. If you set fixed the background image then the image will not move during scrolling in the browser. Let?s take an example with fixed background image.

```
background: white url('bbb.gif');
background-repeat: no-repeat;
background-
attachment: fixed;
```

5) CSS background-position

The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

You can set the following positions:

1. center
2. top
3. bottom
4. left
5. right

```
1. background: white url('good-morning.jpg');
2. background-repeat: no-repeat;
3. background-attachment: fixed;
4. background-position: center;
```

4.What is the color property in CSS is used to set the color of HTML elements

Ans. Color Properties in CSS:

CSS color properties allows us to color the "Background and Foreground Color" on a Web Page. We can set CSS color on text, backgrounds, borders, and other parts of elements in a document.

I.CSS Background Color

(i)Set CSS body background color:

You can define the background color of a webpage by specifying its body

"background-color" property.

Example:

body {background-color:#C0C0C0;}

(ii)Set CSS Paragraph background color :

Example:

a. body

```
 {
background-color:#C0C0C0;
 }
```

b. p {background-color:#FFFFFF;}

(iii)Set CSS div back color :

```
body {background-color:#C0C0C0;}
p {background-color:#FFFFFF;}
div {background-color:#00FFFF;}
```

2.CSS Foreground Color

(i) Change CSS text color :

When we want to change the color of a text (foreground color) in an HTML document the term color is used to specify the CSS property.

Example:

```
body { color: #800000 }
```

(ii) Change the Font Color with CSS :

When you set foreground color , actually the font color will change.

Example:

```
h1 { color: green; }
```

The above code changes the font color inside the h1 tag.

3.BorderColor

(i)Set CSS border-color:

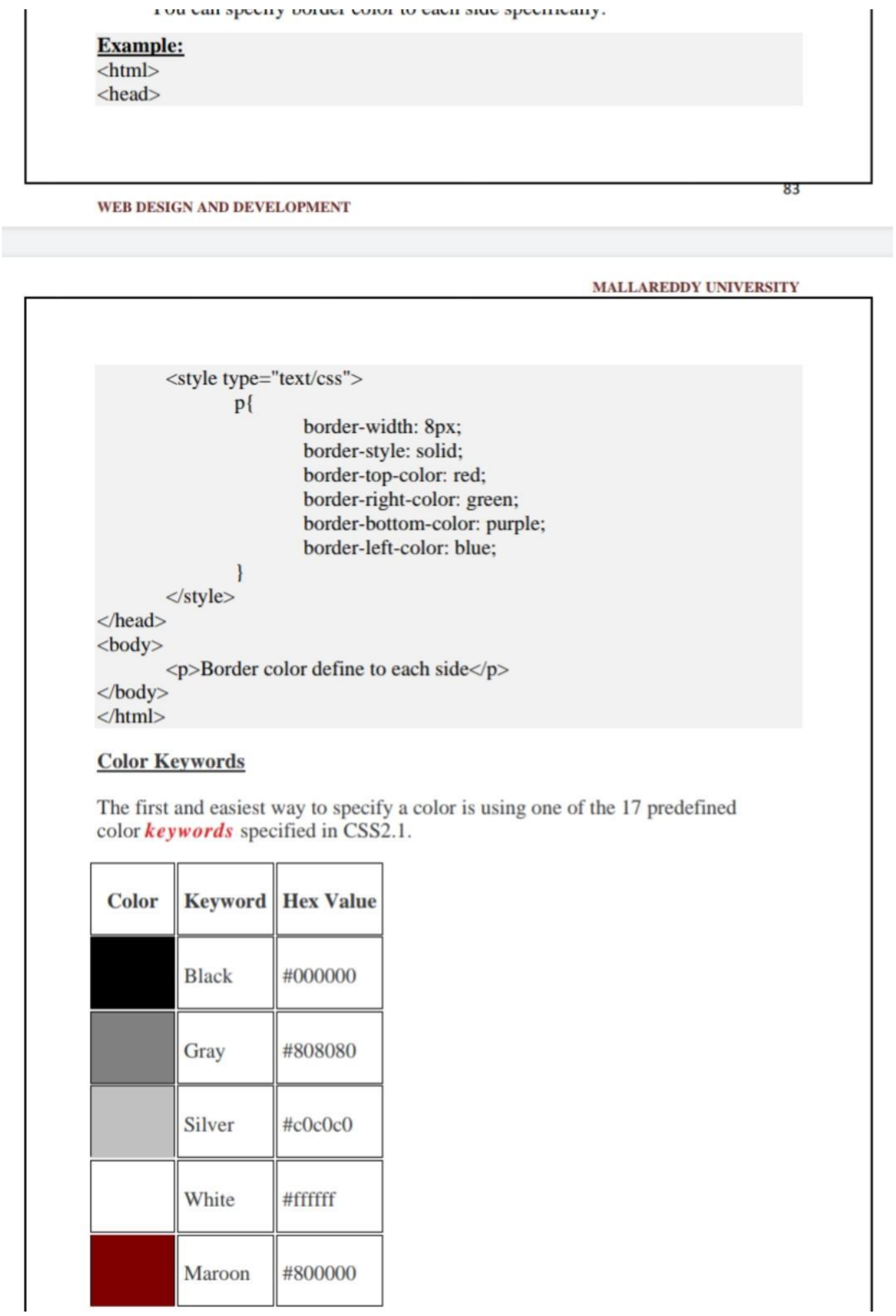
The "border-color" property specifies the border color for each side of the box.

Example:

```
P{
```

```
border-width: 2px;

border-color:red;

border-style: solid;

or

border:2px solid red;

 }
```

You can specify border color to each side specifically.



You can specify border color to each side specifically.

Example:

```
<html>
<head>
```

```
        <style type="text/css">
                p{
                        border-width: 8px;
                        border-style: solid;
                        border-top-color: red;
                        border-right-color: green;
                        border-bottom-color: purple;
                        border-left-color: blue;
                }
        </style>
</head>
<body>
        <p>Border color define to each side</p>
</body>
</html>
```

Color Keywords

The first and easiest way to specify a color is using one of the 17 predefined color ***keywords*** specified in CSS2.1.

Color	Keyword	Hex Value
⬛	Black	#000000
▨	Gray	#808080
▨	Silver	#c0c0c0
⬜	White	#ffffff
🟥	Maroon	#800000

5.Write short notes on CSS three types of gradients?

Ans. CSS Gradient:

CSS gradient is used to display smooth transition within two or more specified colors.

Why CSS Gradient

These are the following reasons to use CSS gradient.

o You don't have to use images to display transition effects.

o The download time and bandwidth usage can also be reduced.

o It provides better look to the element when zoomed, because the gradient is generated by

the browser.

1) CSS Linear Gradient

The CSS3 linear gradient goes up/down/left/right and diagonally. To create a CSS3 linear gradient, you must have to define two or more color stops. The color stops are the colors which are used to create a smooth transition. Starting point and direction can also be added along with the gradient effect.

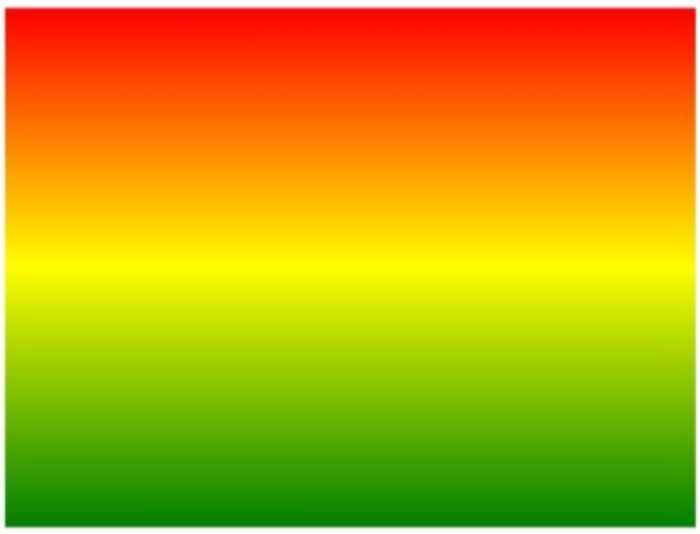
1. **background: linear-gradient (direction, color-stop1, color-stop2.....);**

Linear gradient :

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8">

<title>Example of Linear Gradients from Top to Bottom</title>

<style>

.gradient {

width: 400px;

height: 300px;

background: red;

background: linear-gradient(red, yellow, green);

}
```

```
</style>

</head>

<body>

 <div class="gradient"></div>

</body>

</html>
```

Output:

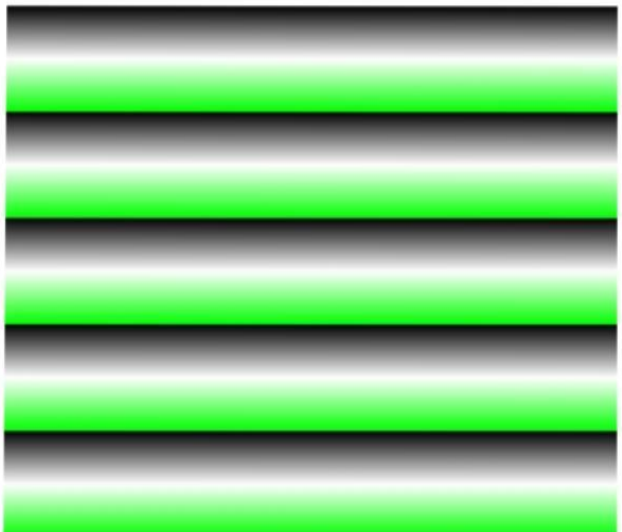


ii) Repeating linear gradient:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Example of Linear Gradients from Top to Bottom</title>
<style>
        .gradient {
                width: 400px;
                height: 300px;
                background: red;
                background: repeating-linear-gradient(black, white 10%, lime 20%);
        }
</style>
</head>
```

```
<body>
   <div class="gradient"></div>
</body>
</html>
```

Output:

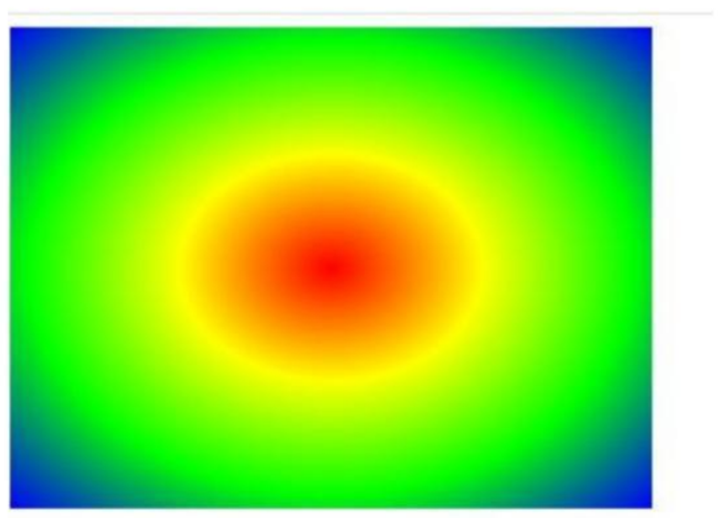


2) CSS Radial Gradient

You must have to define at least two color stops to create a radial gradient. It is defined by its

center.

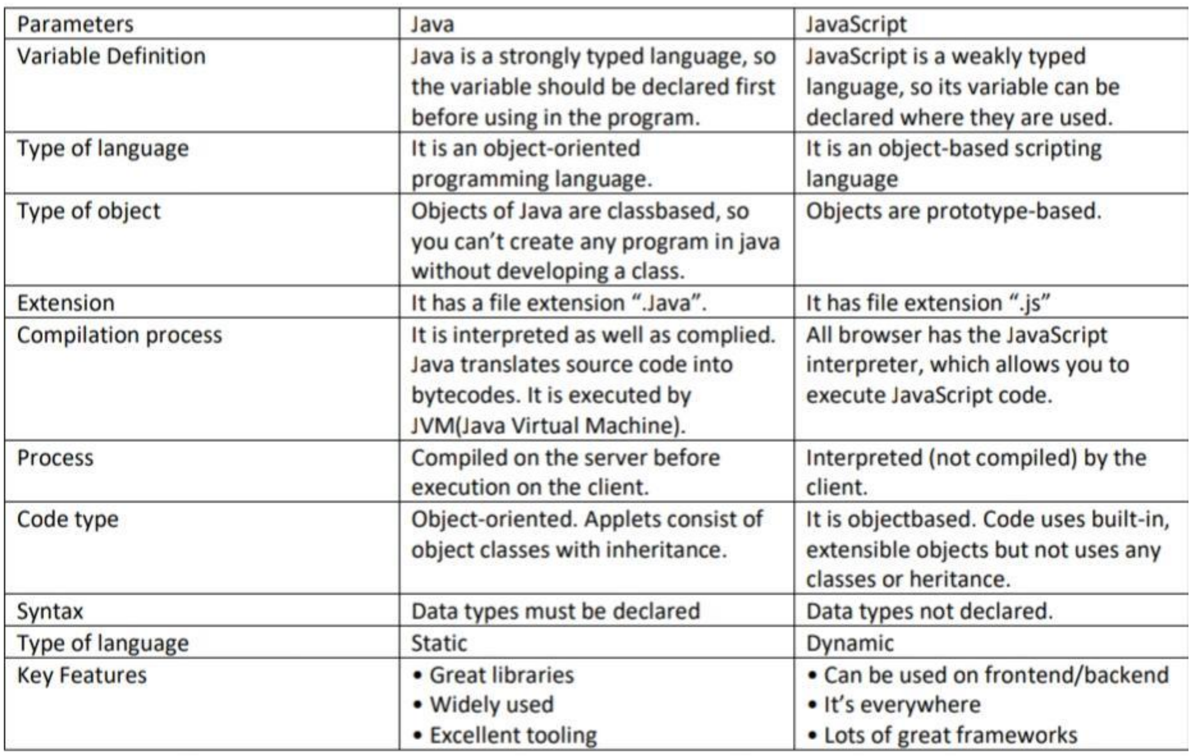
1. ***background: radial-gradient(shape size at position, start-color, ..., last-color);***
2. **<!DOCTYPE html>**
3. **<html lang="en">**
4. **<head>**
5. **<meta charset="utf-8">**
6. **<title>Example of Linear Gradients from Top to Bottom</title>**
7. **<style>**
8. **.gradient {**
9. **width: 400px**
10. **height: 300px;**
11. **background: red;**
12. **background: radial-gradient(red, yellow, lime, blue);**
13. **}**
14. **</style>**
15. **</head>**
16. **<body>**
17. **<div class="gradient"></div>**
18. **</body>**
19. **</html>**
20. **Output:**



UNIT-3

1.What is the difference between JavaScript and Java?

Ans.



Parameters	Java	JavaScript
Variable Definition	Java is a strongly typed language, so the variable should be declared first before using in the program.	JavaScript is a weakly typed language, so its variable can be declared where they are used.
Type of language	It is an object-oriented programming language.	It is an object-based scripting language
Type of object	Objects of Java are classbased, so you can't create any program in java without developing a class.	Objects are prototype-based.
Extension	It has a file extension ".Java".	It has file extension ".js"
Compilation process	It is interpreted as well as complied. Java translates source code into bytecodes. It is executed by JVM(Java Virtual Machine).	All browser has the JavaScript interpreter, which allows you to execute JavaScript code.
Process	Compiled on the server before execution on the client.	Interpreted (not compiled) by the client.
Code type	Object-oriented. Applets consist of object classes with inheritance.	It is objectbased. Code uses built-in, extensible objects but not uses any classes or heritance.
Syntax	Data types must be declared	Data types not declared.
Type of language	Static	Dynamic
Key Features	• Great libraries • Widely used • Excellent tooling	• Can be used on frontend/backend • It's everywhere • Lots of great frameworks

2.What is JavaScript? List some features of JavaScript.

Ans. JavaScript: JavaScript is a programming language commonly used in web development. It was originally developed by Netscape as a means to add dynamic and interactive elements to websites. While JavaScript is influenced by Java, the syntax is more similar to C and is based on ECMAScript, a scripting language developed by Sun Microsystems.

important features of JavaScript.

- Light Weight Scripting language.
- Dynamic Typing.
- Object-oriented programming support.
- Functional Style.
- Platform Independent.
- Prototype-based.
- Interpreted Language.

1. Light Weight Scripting Language

JavaScript is a lightweight scripting language because it is made for data handling at the browser only. Since it is not a general-purpose language so it has a limited set of libraries. Also as it is only meant for client-side execution and that too for web applications, hence the lightweight nature of JavaScript is a great feature.

2. Dynamic Typing

JavaScript supports dynamic typing which means types of the variable are defined based on the stored value. For example, if you declare a variable `x` then you can store either a string or a Number type value or an array or an object. This is known as dynamic typing.

3. Object-Oriented Programming Support

Starting from ES6, the concept of class and OOPs has been more refined. Also, in JavaScript, two important principles with OOP in JavaScript are Object Creation patterns (**Encapsulation**) and Code Reuse patterns (**Inheritance**). Although JavaScript developers rarely use this feature but its there for everyone to explore.

3. Functional Style

This implies that JavaScript uses a functional approach, even objects are created from the constructor functions and each constructor function represents a unique object-type. Also, functions in JavaScript can be used as objects and can be passed to other functions too.

4. Platform Independent

This implies that JavaScript is platform-independent or we can say it is portable; which simply means that you can simply write the script once and run it anywhere and anytime. In general, you can write your JavaScript applications and run them on any platform or any browser without affecting the output of the Script.

5. Prototype-based Language

JavaScript is a prototype-based scripting Language. This means javascript uses prototypes instead of classes or inheritance. In languages like Java, we create a class and then we create objects for those classes. But in JavaScript, we define object prototype and then more objects can be created using this object prototype.

7. Interpreted Language

JavaScript is an interpreted language which means the script written inside javascript is processed line by line. These Scripts are interpreted by JavaScript interpreter which is a built-in component of the Web browser.

3.What are the different data types present in JavaScript?

Ans. Javascript Data Types

JavaScript provides different data types to hold different types of values. There

are two types of data types in JavaScript.

1. Primitive data type

2. Non-primitive (reference) data type

JavaScript is a dynamic type language, means you don't need to specify type of

the variable because it is dynamically used by JavaScript engine. You need to

use var here to specify the data type. It can hold any type of values such as

numbers, strings etc. For example

1. var a=40;//holding number

2. var b="Rahul";//holding string

JavaScript primitive data types

There are five types of primitive data types in JavaScript. They are as follows:

Data -Type Description

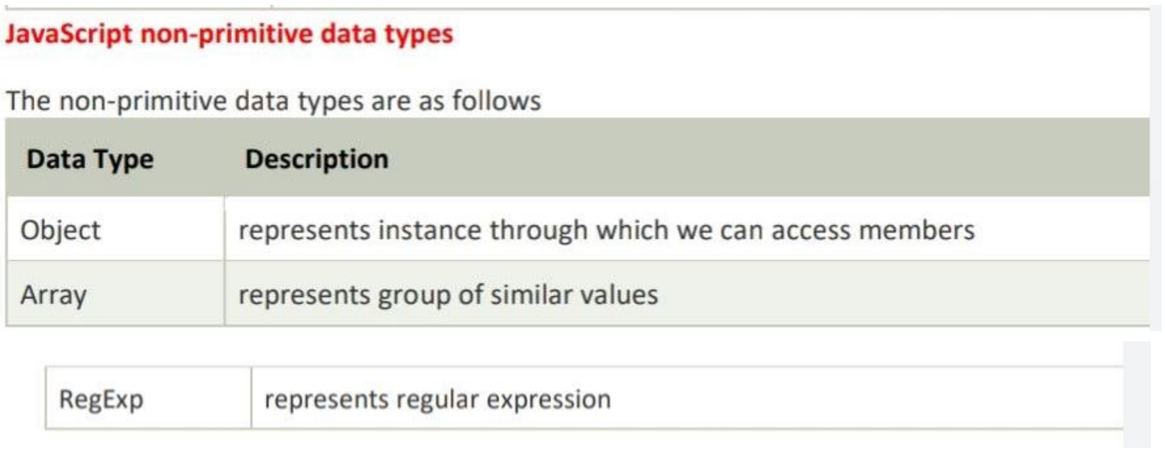
String - represents sequence of characters e.g. "hello"

Number- represents numeric values e.g. 100

Boolean - represents boolean value either false or true

Undefined - represents undefined value

Null - represents null i.e. no value at all



JavaScript non-primitive data types

The non-primitive data types are as follows

Data Type	Description
Object	represents instance through which we can access members
Array	represents group of similar values
RegExp	represents regular expression

4. Write short notes on :

a. Type Conversion

b. Operators

Ans

A.Type Conversion

JavaScript variables can be converted to a new variable and another data type:

- By the use of a JavaScript function
- **Automatically** by JavaScript itself

We can do the following operations

- Converting Strings to Numbers
- Converting Numbers to Strings
- Converting Dates to Numbers
- Converting Dates to String
- Converting Booleans to Numbers
- Converting Booleans to Strings

Converting Strings to Numbers

- The global method `Number()` can convert strings to numbers.
- Strings containing numbers (like "3.14") convert to numbers (like 3.14).
- Empty strings convert to 0.
- Anything else converts to `NaN` (Not a Number).

```
• Number("3.14")    // returns 3.14
  Number(" ")       // returns 0
  Number("")        // returns 0
  Number("99 88")   // returns NaN
```

Converting Numbers to Strings

The global method `String()` can convert numbers to strings.

Example

```
String(x)         // returns a string from a number variable x
String(123)       // returns a string from a number literal 123
```

Converting Dates to Numbers

The global method `Number()` can be used to convert dates to numbers.

```
d = new Date();
Number(d)          // returns 1404568027739
```

Converting Dates to Strings

The global method `String()` can convert dates to strings.

```
String(Date())  // returns "Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe
Daylight Time)"
```

Converting Booleans to Numbers

The global method `Number()` can also convert booleans to numbers.

```
Number(false)     // returns 0
Number(true)      // returns 1
```

Converting Booleans to Strings

The global method `String()` can convert booleans to strings.

```
String(false)      // returns "false"
String(true)       // returns "true"
```

B.operators

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator.

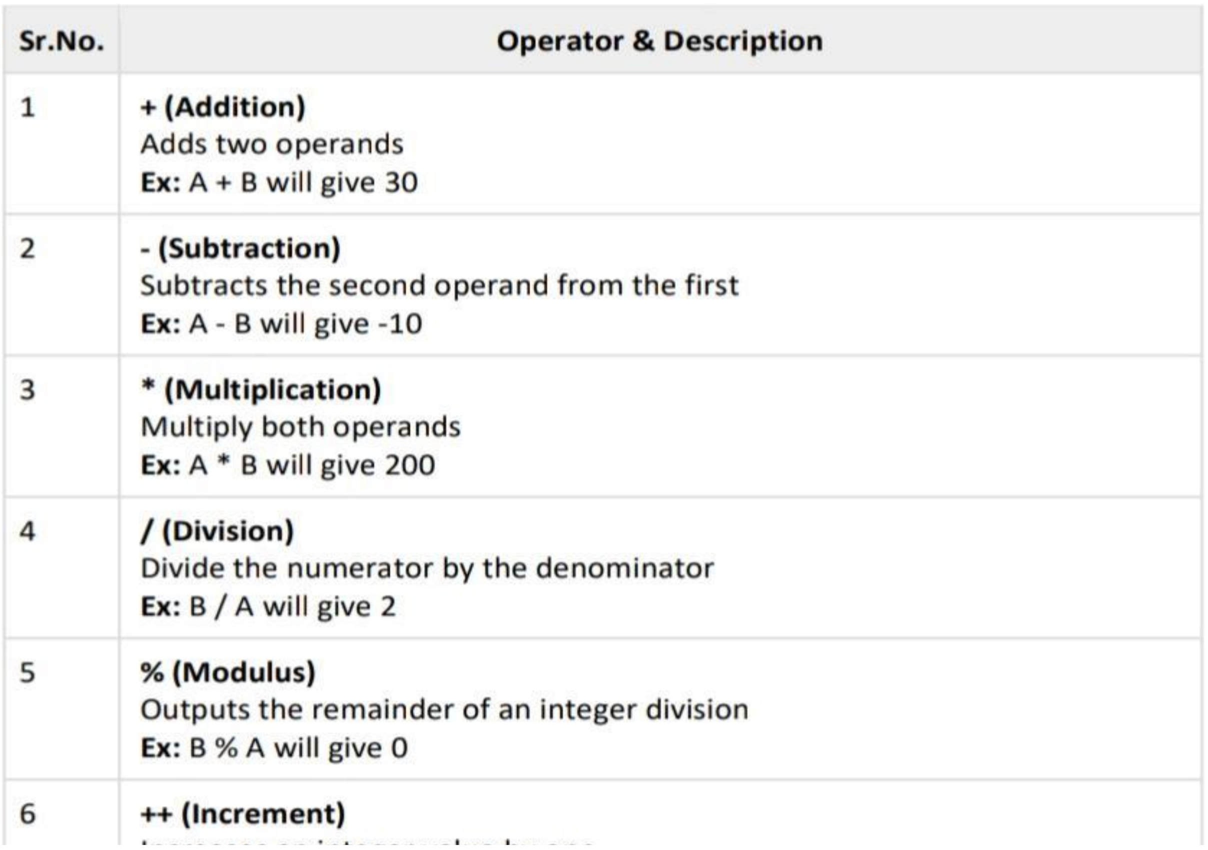
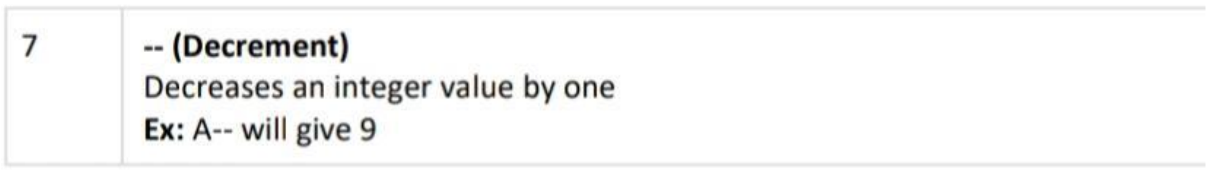
JavaScript supports the following types of operators.

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

Arithmetic Operators

JavaScript supports the following arithmetic operators –

Assume variable A holds 10 and variable B holds 20, then –

Sr.No.	Operator & Description
1	**+ (Addition)** Adds two operands **Ex:** A + B will give 30
2	**- (Subtraction)** Subtracts the second operand from the first **Ex:** A - B will give -10
3	*** (Multiplication)** Multiply both operands **Ex:** A * B will give 200
4	**/ (Division)** Divide the numerator by the denominator **Ex:** B / A will give 2
5	**% (Modulus)** Outputs the remainder of an integer division **Ex:** B % A will give 0
6	**++ (Increment)**
7	**-- (Decrement)** Decreases an integer value by one **Ex:** A-- will give 9

Comparison Operators

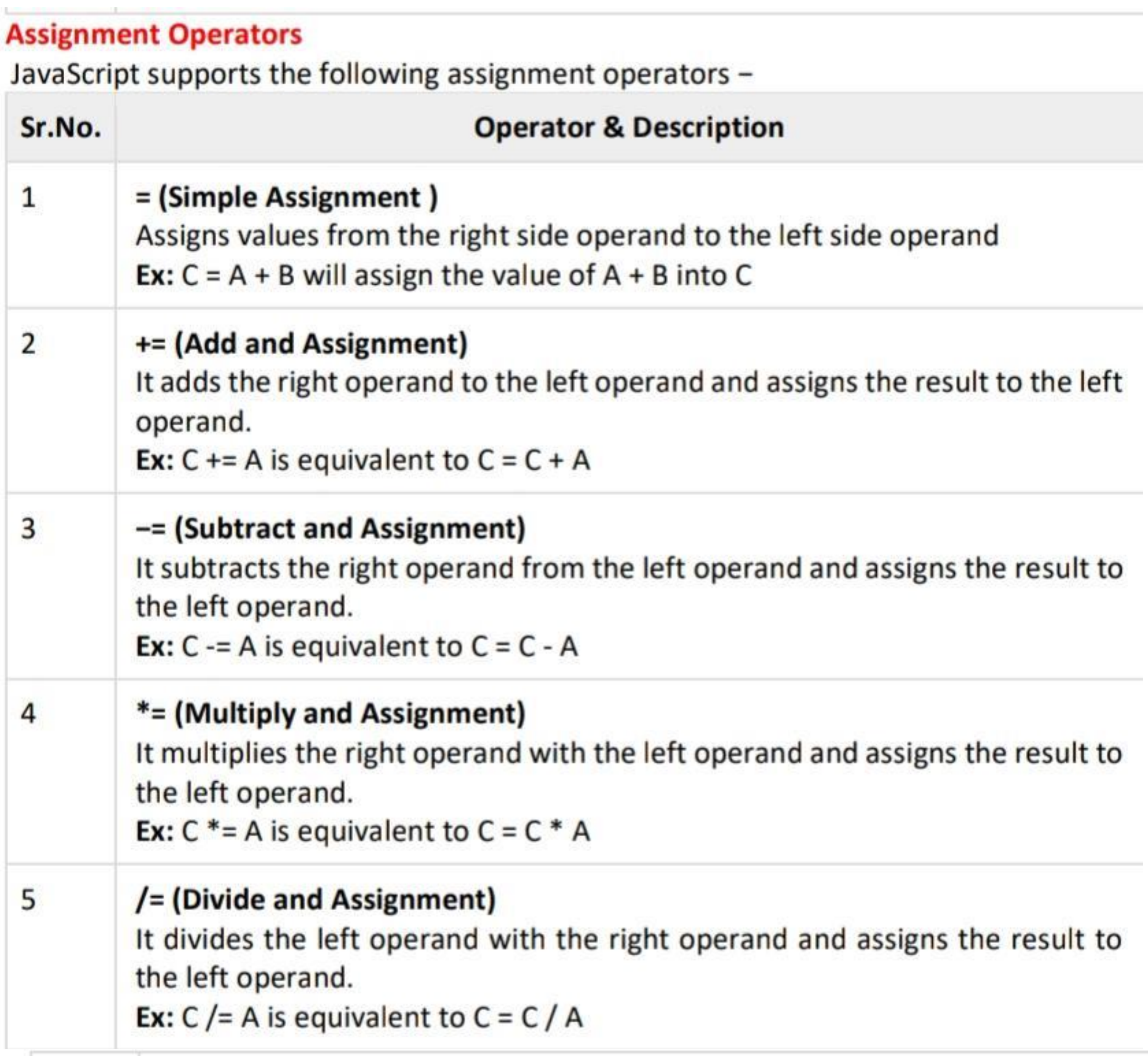
JavaScript supports the following comparison operators –
Assume variable A holds 10 and variable B holds 20, then –

Sr.No.	Operator & Description
1	**= = (Equal)** Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** (A == B) is not true.
2	**!= (Not Equal)** Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** (A != B) is true.
3	**> (Greater than)** Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A > B) is not true.
4	**< (Less than)** Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A < B) is true.
5	**>= (Greater than or Equal to)** Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A >= B) is not true.
6	**<= (Less than or Equal to)** Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A <= B) is true.

Logical Operators

JavaScript supports the following logical operators –
Assume variable A holds 10 and variable B holds 20, then –

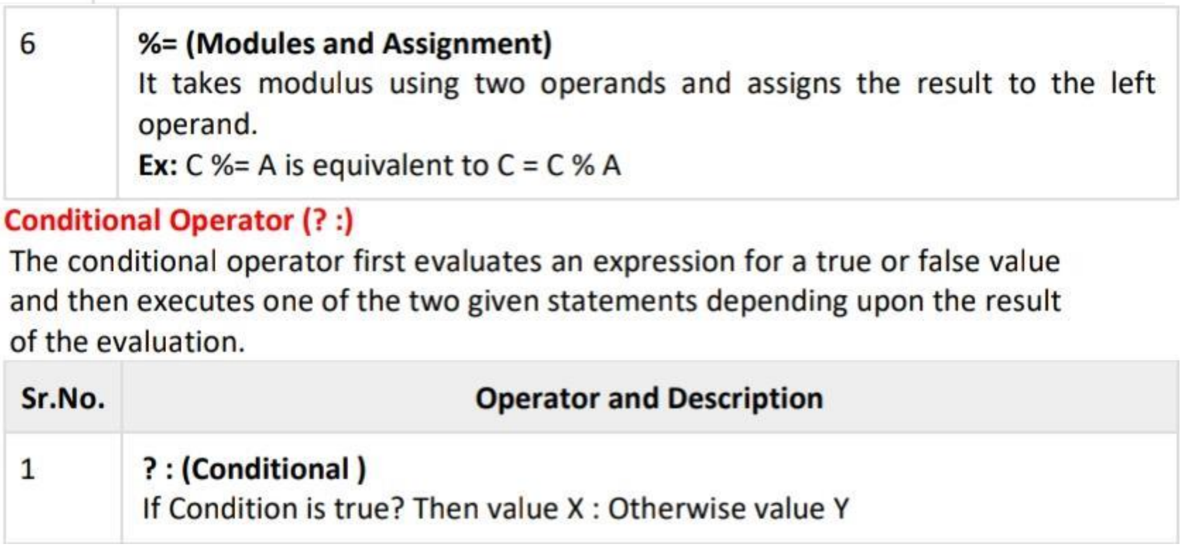
Sr.No.	Operator & Description
1	**&& (Logical AND)** If both the operands are non-zero, then the condition becomes true. **Ex:** (A && B) is true.
2	**|| (Logical OR)** If any of the two operands are non-zero, then the condition becomes true. **Ex:** (A || B) is true.
3	**! (Logical NOT)** Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Ex:** ! (A && B) is false.



Assignment Operators

JavaScript supports the following assignment operators –

Sr.No.	Operator & Description
1	**= (Simple Assignment)** Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C
2	**+= (Add and Assignment)** It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A
3	**−= (Subtract and Assignment)** It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C - A
4	***= (Multiply and Assignment)** It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A
5	**/= (Divide and Assignment)** It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A



6	**%= (Modules and Assignment)** It takes modulus using two operands and assigns the result to the left operand. **Ex:** C %= A is equivalent to C = C % A

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No.	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

5. What are the pop-up boxes available in JavaScript?

Ans. JavaScript has three kind of popup boxes:

Alert box, Confirm box, and Prompt box.

Alert Box

An alert box is often used if you want to make sure information comes through to the user.When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
window.alert("sometext");
```

The window.alert() method can be written without the window prefix.

Example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Alert</h2>
<button onclick="myFunction()">Try it</button>
<script>
function myFunction() {
 alert("I am an alert box!");
}
</script>
</body>
</html>
```

Output:

```
JavaScript Alert
Try it
```

Confirm Box:

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box

returns false.

Syntax:

```
window.confirm("sometext");
```

The window.confirm() method can be written without the window prefix.

Example:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Confirm Box</h2>
<button onclick="myFunction()">Try it</button>
<p id="demo"></p>
<script>
function myFunction() {
 var txt;
 if (confirm("Press a button!")) {
 txt = "You pressed OK!";
 } else {
 txt = "You pressed Cancel!";
 }
 document.getElementById("demo").innerHTML = txt;
}
</script>
</body>
</html>
```

Output:

```
JavaScript Confirm Box
```

Try it

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
window.prompt("sometext","defaultText");
```

The window.prompt() method can be written without the window prefix.

Example

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript</h2>
<p>Line-breaks in a popup box.</p>
<button onclick="alert('Hello\nHow are you?')">Try it</button>
</body>
</html>
```

Output:

```
JavaScript
Line-breaks in a popup box.
Try it
```

UNIT-IV

1.Brifely explain the conditional statements are in Java scrip

Ans. Conditional statements:

JavaScript supports conditional statements which are used to perform different actions based on different conditions. Here we will explain the if..else statement.

JavaScript supports the following forms of if..else statement –

- **if statement**
- **if...else statement**
- **if...else if... statement.**

if Statement:

The if statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax:

The syntax for a basic if statement is as follows −

```
if(expression)
{
Statement(s) to be executed if expression istrue
}
```

Examples:

```
<scripttype="text/javascript">
<!--
var age =20;
if( age >18){
document.write("<b>Qualifies for driving</b>");
}
```

```
//-->
</script>
<p>Set the variable to different value and then try...</p>
```

Output:

```
Set the variable to different value and then try...
```

if...else statement:

The 'if...else' statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax:

```
if(expression)
{
Statement(s) to be executed if expression istrue.
}
else{
Statement(s) to be executed if expression isfalse
}
```

Example:

```
<scripttype="text/javascript">
<!--
var age =15;
if( age >18){
document.write("<b>Qualifies for driving</b>");
}
else{
```

```
document.write("<b>Does not qualify for driving</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
```

Output:

```
Does not qualify for driving
Set the variable to different value and then try...
```

if...else if... statement:

The if...else if... statement is an advanced form of if…else that allows JavaScript to make a correct decision out of several conditions.

Syntax:

The syntax of an if-else-if statement is as follows −

```
if(expression 1)
{
Statement(s) to be executed if expression 1istrue
}
elseif(expression 2)
{
Statement(s) to be executed if expression 2istrue
}
elseif(expression 3)
{
Statement(s) to be executed if expression 3istrue
}
else{
Statement(s) to be executed ifno expression istrue
}
```

Example

```
<scripttype="text/javascript">
<!--
var book ="maths";
if( book =="history"){
document.write("<b>History Book</b>");
}
elseif( book =="maths"){
document.write("<b>Maths Book</b>");
}
elseif( book =="economics"){
document.write("<b>Economics Book</b>");
}
else{
document.write("<b>Unknown Book</b>");
}
//-->
</script>
<p>Set the variable to different value and then try...</p>
```

Output:

```
Maths Book
Set the variable to different value and then try...
```

JavaScript -Switch Case:

You can use multiple if...else…if statements, as in the previous chapter, to perform a multiway branch. However, this is not always the best solution, especially when all of the branches depend on the value of a single variable. Starting with JavaScript 1.2, you can use a switch statement which handles exactly this situation, and it does so more efficiently than repeated if...else if statements.

Syntax:

```
switch(expression)
{
case condition 1: statement(s)
break;
case condition 2: statement(s)
break;
...
case condition n: statement(s)
break;
default: statement(s)
}
```

Example

```
<scripttype="text/javascript">
<!--
var grade='A';
document.write("Entering switch block<br />");
switch(grade)
```

```
 {
 case'A':document.write("Good job<br />");
 break;
 case'B':document.write("Pretty good<br />");
 break;
 case'C':document.write("Passed<br />");
break;
case'D':document.write("Not so good<br />");
break;
case'F':document.write("Failed<br />");
break;
default:document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
<p>Set the variable to different value and then try...</p>
```

Output:

```
Entering switch block
Good job
Exiting switch block
Set the variable to different value and then try...
```

Break statements play a major role in switch-case statements. Try the following code that uses switch-case statement without any break statement.

```
<scripttype="text/javascript">
<!--
var grade='A';
document.write("Entering switch block<br />");
switch(grade)
{
case'A':document.write("Good job<br />");
case'B':document.write("Pretty good<br />");
case'C':document.write("Passed<br />");
case'D':document.write("Not so good<br />");
case'F':document.write("Failed<br />");
default:document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//-->
</script>
<p>Set the variable to different value and then try...</p>
```

Output:

```
Entering switch block
Good job
Pretty good
Passed
Not so good
Failed
Unknown grade
Exiting switch block
```

Set the variable to different value and then try..

2. What are the different types of looping statements available in JavaScript?

Ans: Loop Types in JavaScript:

1.for loop

2.while loop

3.do-while loop

1. for Loop

for loop is a most frequently used loop in javascript. It consists of 3 parts i.e.loop initializer, test statement and iteration statement. loop initializer is executed only once before the loop begins. test statement helps to determine whether a condition is true or not. If the condition is true, code inside of loop is executed else code inside loop is escaped. Lastly, iteration statement is used to increment or decrement the value of test statement. This is executed on every loop.

Syntax of for loop:

```
for (statement 1; statement 2; statement 3) {
 code block to be executed
}
```

Example of for loop:

```
var sum = 0;
for (var i = 1; i <= 20; i++) {
 sum = sum + i;
}
alert("Sum = " + sum);
//output: Sum = 210
```

2.while loop:while loop is also frequently used loop in javascript. While loop

executes a block of code as long as the condition specified is true.

syntax of while loop:

```
while (condition) {
 code block to be executed
}
```

Example of while loop:

```
var sum = 0;
var number = 1;
while (number <= 50) { // -- condition
  sum += number; // -- body
  number++; // -- updater
}
alert("Sum = " + sum);
//output: Sum = 1275
```

3. do-while loop

do-while loop is similar to while loop except that the condition check done at the end of loop. Due to this, block of code inside loop is executed at least once.

syntax of do-while loop:

```
do{
  code block to be executed
} while (condition);
```

Example of do-while loop:

Below is a simple example of do-while loop.

```
var sum = 0;
var number = 1;
do {
 sum += number; // -- body
 number++; // -- updater
} while (number <= 50); // -- condition
alert("Sum = " + sum);
//output: Sum = 1275
```

3.Define a named function in Java script and explain the functions parameters?

Ans: Defining a Function

Functions are defined, or declared, with the function keyword. Below is the syntax for a function in JavaScript.

```
function nameOfFunction()
{
// Code to be executed
}
```

The declaration begins with the function keyword, followed by the name of the function. Function names follow the same rules as variables — they can contain letters, numbers, underscores and dollar signs, and are frequently written in camel case. The name is followed by a set of parentheses, which can be used for optional parameters. The code of the function is contained in curly brackets, just like a for statement or an if statement.

In our first example, we'll make a function declaration to print a greeting statement to the console.

```
// Initialize greeting function
function greet()
{
console.log("Hello, World!");
}
// Invoke the function
greet();
```

Now we will put those together, defining our function and invoking it.

```
// Initialize greeting function
function greet()
{
console.log("Hello, World!");
}
// Invoke the function
greet();
```

With the call for greet();, the function will run and we will receive the Hello, World! as the program’s output.

Output

```
Hello, World!
```

Function Parameters

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

Example

```
<html>
<head>
<scripttype="text/javascript">
functionsayHello(name, age)
```

```
{
document.write(name +" is "+ age +" years old.");
}
</script>
</head>
<body>
<p>Click the following button to call the function</p>
<form>
<inputtype="button"onclick="sayHello('Zara',7)"value="Say Hello">
</form>
<p>Use different parameters inside the function and then try...</p>
</body>
</html>
```

Output:

```
functionsayHello(name, age){ document.write(name +" is "+ age +" years old."); }
Click the following button to call the function
Use different parameters inside the function and then try...
```

4. Write short notes on :

a. Call()

b. Apply()

c. Array

Ans. The JavaScript call() Method

The `call()` method is a predefined JavaScript method.

It can be used to invoke (call) a method with an owner object as an argument (parameter).

With `call()`, an object can use a method belonging to another object.

This example calls the **fullName** method of person, using it on **person1**:

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Functions</h2>
<p>This example calls the fullName method of person, using it on person1:
 </p>
 <p id="demo"></p>
<script>
const person = {
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
}
const person1 = {
  firstName:"John",
  lastName: "Doe"
}
const person2 = {
  firstName:"Mary",
  lastName: "Doe"
}
document.getElementById("demo").innerHTML = person.fullName.call(person1);
 </script>
 </body>
</html>
```

Output:

JavaScript Functions

This example calls the fullName method of person, using it on person1:

John Doe

The JavaScript apply() Method

The `apply()` method is similar to the `call()` method (previous chapter).

The difference is:

The `call()` method takes arguments **separately**.

The `apply()` method takes arguments as an **array**.

```
<!DOCTYPE html>
<html>
 <body>
 <h2>JavaScript Functions</h2>
 <p>In this example the fulllName method of person is <b>applied</b> on person1:</p>
  <p id="demo"></p>
  <script>
const person = {
  fullName: function(city, country) {
    return this.firstName + " " + this.lastName + "," + city + "," + country;
  }
}
const person1 = {
  firstName:"John",

  lastName: "Doe"

}

document.getElementById("demo").innerHTML = person.fullName.apply(person1,
["Oslo", "Norway"]);

</script>

</body>

</html>
```

Output:

JavaScript Functions

In this example the fulllName method of person is **applied** on person1:

John Doe,Oslo,Norway

JavaScript Arrays

An array is a special variable, which can hold more than one value

```
<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Arrays</h2>

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

<script>

const cars = ["Saab", "Volvo", "BMW"];

document.getElementById("demo").innerHTML = cars;

 </script>
```

</body>

</html>

Why Using an Array?

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
let car1 = "Saab";
let car2 = "Volvo";
let car3 = "BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The solution is an array!

An array can hold many values under a single name, and you can access the values by referring to an index number.

7.Briefly explain about arrays in Java script?

Ans: JavaScript array is an object that represents a collection of similar type of

elements.

There are 3 ways to construct array in JavaScript

□ By array literal

□ By creating instance of Array directly (using new keyword)

□ By using an Array constructor (using new keyword)

1) JavaScript array literal

The syntax of creating array using array literal is given below:

var arrayname=[value1,value2.....valueN];

As you can see, values are contained inside [ ] and separated by , (comma).

<html>

<body>

<script>

var emp=["Sono","Vimal","Ratan"];

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

Output:

```
Sonoo
Vimal
Ratan
```

2) JavaScript Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, new keyword is used to create instance of array.

Let's see the example of creating array directly.

```
<html>
<body>
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";

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

Output:

```
Arun
Varun
John
```

3) JavaScript array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

```
<html>
<body>
```

UNIT-4

1.What is Bootstrap? Explain the Responsive Web Design.

Ans. 1. Bootstrap is the most popular HTML, CSS and JavaScript framework for developing a responsive and mobile friendly website.

2.It is absolutely free to download and use.

3.It is a front-end framework used for easier and faster web development.

4.It includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many others.

5.It can also use JavaScript plug-ins.

6.It facilitates you to create responsive designs.

Responsive Web Design:

Responsive web design (RWD) is a web development approach that creates dynamic changes to the appearance of a website, depending on the screen size and orientation of the device being used to view it

Creating Containers with Bootstrap

Containers are the most basic layout element in Bootstrap and are required when using the grid system. Containers are basically used to wrap content with some padding. They are also used to align the content horizontally center on the page in case of fixed width layout.

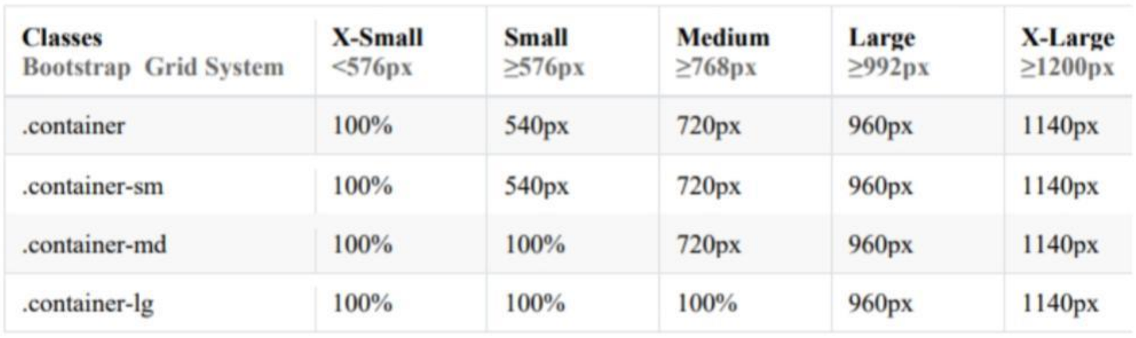
Bootstrap provides three different types containers:

• .container, which has a max-width at each responsive breakpoint.

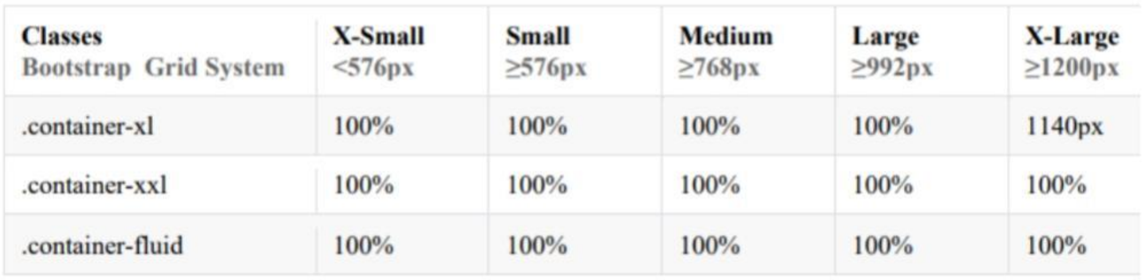
• .container-fluid, which has 100% width at all breakpoints.

• .container-{breakpoint}, which has 100% width until the specified breakpoint.

The table below illustrates how each container's max-width changes across each breakpoint.



Classes Bootstrap Grid System	X-Small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px
.container	100%	540px	720px	960px	1140px
.container-sm	100%	540px	720px	960px	1140px
.container-md	100%	100%	720px	960px	1140px
.container-lg	100%	100%	100%	960px	1140px



Classes Bootstrap Grid System	X-Small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px
.container-xl	100%	100%	100%	100%	1140px
.container-xxl	100%	100%	100%	100%	100%
.container-fluid	100%	100%	100%	100%	100%

Creating Responsive Fixed-width Containers

You can simply use the .container class to create a responsive, fixed-width

container. The width of the container will change at different breakpoints or screen sizes, as shown above.

EXAMPLE:

```
<div class="container">
<h1>This is a heading</h1>
<p>This is a paragraph of text.</p>
</div>
```

Creating Fluid Containers

You can use the .container-fluid class to create a full width container. The width of the fluid container will always be 100% irrespective of the devices or screen sizes.

EXAMPLE:

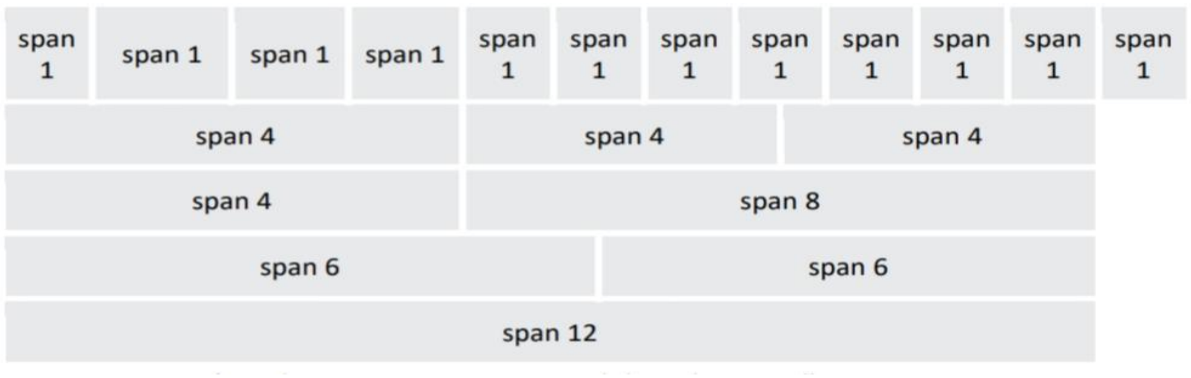
```
<div class="container-fluid">
<h1>This is a heading</h1>
<p>This is a paragraph of text.</p>
 </div>
```

2.Define Bootstrap grid and explain the types of bootstrap grid classes.

Ans. Bootstrap grid system is a very common technique used to create web layouts that recognize the size of the screen a visitor is using and adapt to it. This is extremely important nowadays when so many devices of various sizes can be easily connected to the Internet

Bootstrap's grid system allows up to 12 columns across the page.

If you do not want to use all 12 column individually, you can group the columns



together to create wider columns:

The Bootstrap grid system has four classes:

** xs (for phones - screens less than 768px wide)**

** sm (for tablets - screens equal to or greater than 768px wide)**

** md (for small laptops - screens equal to or greater than 992px wide)**

** lg (for laptops and desktops - screens equal to or greater than 1200px wide)**

The classes above can be combined to create more dynamic and flexible layouts.

Basic Structure of a Bootstrap Grid

The following is a basic structure of a Bootstrap grid:

```
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
```

```
<div class="row">

...

</div>
```

First; create a row (<div class="row">). Then, add the desired number of columns (tags with appropriate .col-*-* classes). Note that numbers in .col-*-* should always add up to 12 for each row.

Below we have collected some examples of basic Bootstrap grid layouts.

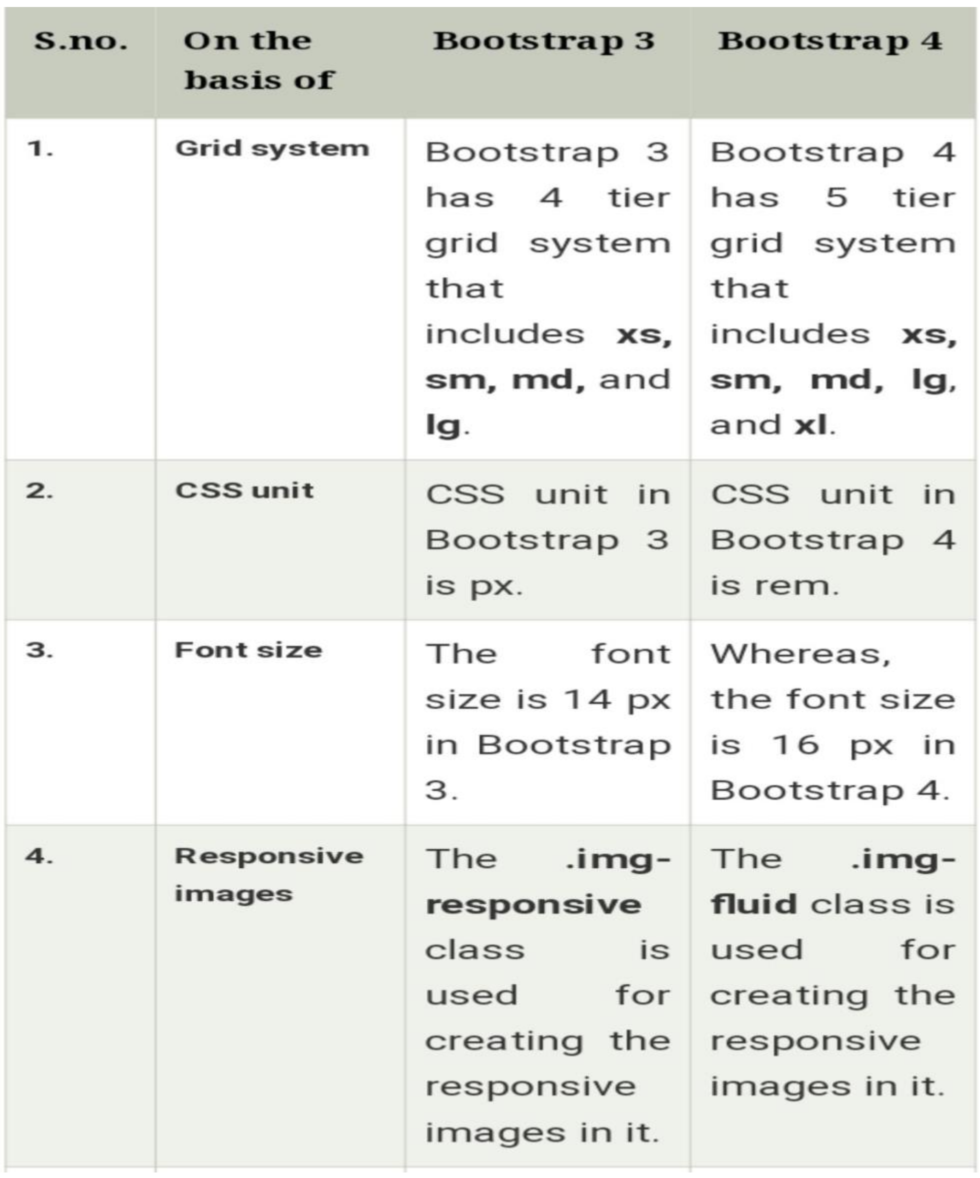
The following example shows how to get a three equal-width columns starting at tablets and scaling to large desktops. On mobile phones or screens that are less than 768px wide, the columns will automatically stack:

Example

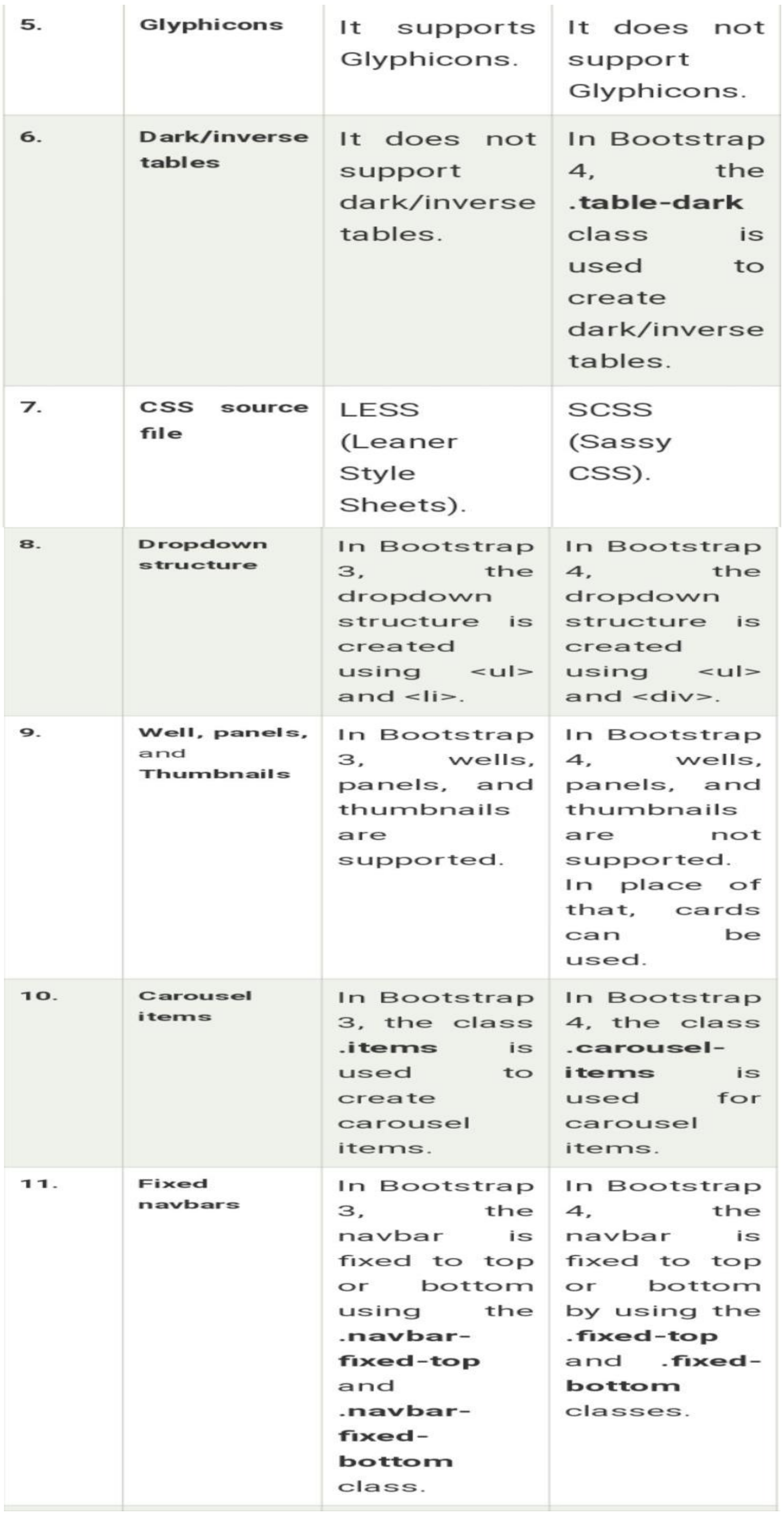
```
<div class="row">

<div class="col-sm-4">.col-sm-4</div>

<div class="col-sm-4">.col-sm-4</div>

<div class="col-sm-4">.col-sm-4</div>

</div>
```

3.What is the difference between Bootstrap 3 and Bootstrap 4?

Ans.



S.no.	On the basis of	Bootstrap 3	Bootstrap 4
1.	Grid system	Bootstrap 3 has 4 tier grid system that includes **xs, sm, md,** and **lg**.	Bootstrap 4 has 5 tier grid system that includes **xs, sm, md, lg**, and **xl**.
2.	CSS unit	CSS unit in Bootstrap 3 is px.	CSS unit in Bootstrap 4 is rem.
3.	Font size	The font size is 14 px in Bootstrap 3.	Whereas, the font size is 16 px in Bootstrap 4.
4.	Responsive images	The **.img-responsive** class is used for creating the responsive images in it.	The **.img-fluid** class is used for creating the responsive images in it.



5.	**Glyphicons**	It supports Glyphicons.	It does not support Glyphicons.
6.	**Dark/inverse tables**	It does not support dark/inverse tables.	In Bootstrap 4, the **.table-dark** class is used to create dark/inverse tables.
7.	**CSS source file**	LESS (Leaner Style Sheets).	SCSS (Sassy CSS).
8.	**Dropdown structure**	In Bootstrap 3, the dropdown structure is created using `<ul>` and `<li>`.	In Bootstrap 4, the dropdown structure is created using `<ul>` and `<div>`.
9.	**Well, panels,** and **Thumbnails**	In Bootstrap 3, wells, panels, and thumbnails are supported.	In Bootstrap 4, wells, panels, and thumbnails are not supported. In place of that, cards can be used.
10.	**Carousel items**	In Bootstrap 3, the class **.items** is used to create carousel items.	In Bootstrap 4, the class **.carousel-items** is used for carousel items.
11.	**Fixed navbars**	In Bootstrap 3, the navbar is fixed to top or bottom using the **.navbar-fixed-top** and **.navbar-fixed-bottom** class.	In Bootstrap 4, the navbar is fixed to top or bottom by using the **.fixed-top** and **.fixed-bottom** classes.

4. **How would you implement a carousel in bootstrap?**

Ans. Carousel:

The Carousel is a slideshow for cycling through elements

Create a Carousel

The following example shows how to create a basic carousel with indicators and controls:

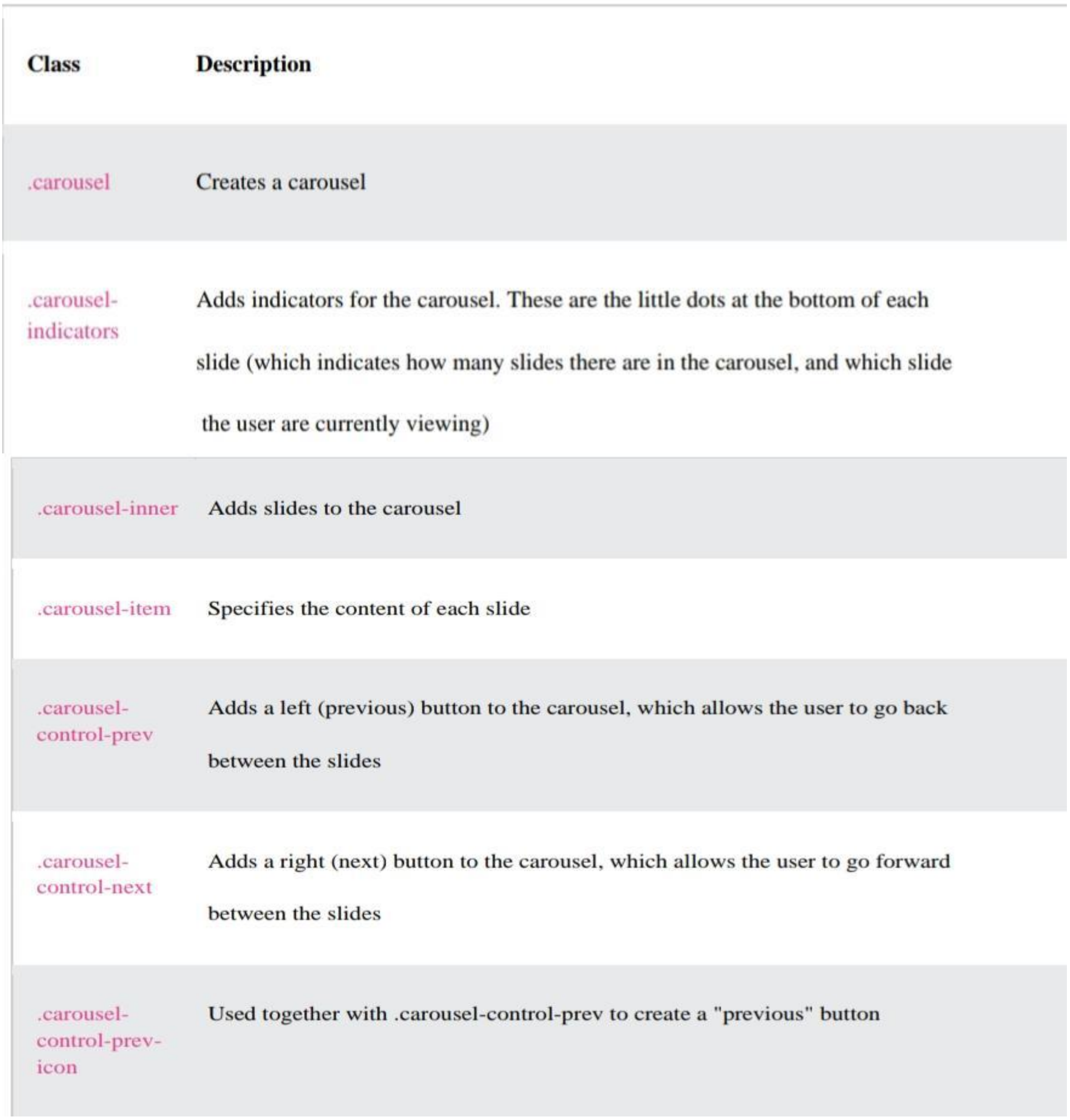
Example

```
<div id="demo" class="carousel slide" data-ride="carousel">

<!-- Indicators -->

<ul class="carousel-indicators">

<li data-target="#demo" data-slide-to="0" class="active"></li>

<li data-target="#demo" data-slide-to="1"></li>

<li data-target="#demo" data-slide-to="2"></li>

</ul>

<!-- The slideshow -->

<div class="carousel-inner">

<div class="carousel-item active">

<img src="la.jpg" alt="Los Angeles">

</div>

<div class="carousel-item">

<img src="chicago.jpg" alt="Chicago">

</div>

<div class="carousel-item">

<img src="ny.jpg" alt="New York">

</div>

</div>

<!-- Left and right controls -->
```

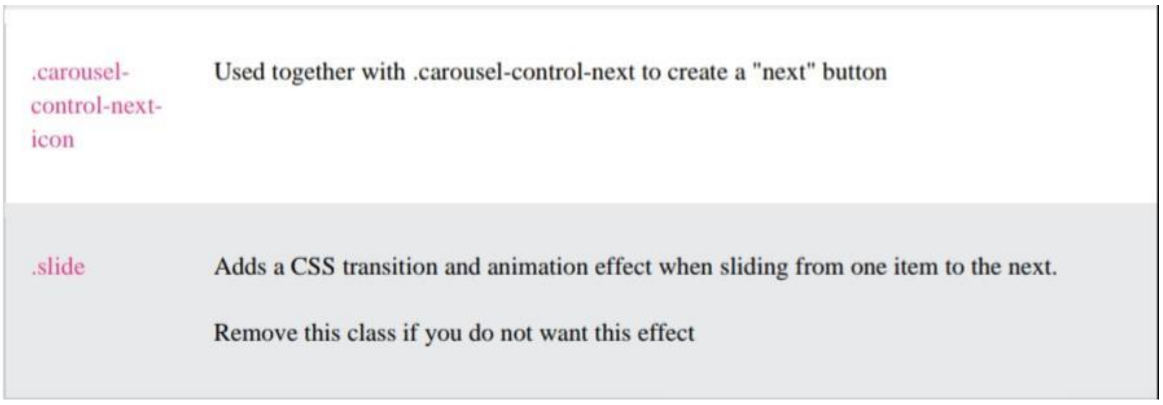
```
<a class="carousel-control-prev" href="#demo" data-slide="prev">
<span class="carousel-control-prev-icon"></span>
</a>
<a class="carousel-control-next" href="#demo" data-slide="next">
<span class="carousel-control-next-icon"></span>
</a>
</div>
```

Example explanation:

A description of what each class from the example above do:



Class	Description
.carousel	Creates a carousel
.carousel-indicators	Adds indicators for the carousel. These are the little dots at the bottom of each slide (which indicates how many slides there are in the carousel, and which slide the user are currently viewing)
.carousel-inner	Adds slides to the carousel
.carousel-item	Specifies the content of each slide
.carousel-control-prev	Adds a left (previous) button to the carousel, which allows the user to go back between the slides
.carousel-control-next	Adds a right (next) button to the carousel, which allows the user to go forward between the slides
.carousel-control-prev-icon	Used together with .carousel-control-prev to create a "previous" button



.carousel-control-next-icon	Used together with .carousel-control-next to create a "next" button
.slide	Adds a CSS transition and animation effect when sliding from one item to the next. Remove this class if you do not want this effect

Add Captions to Slides

Add elements inside <div class="carousel-caption"> within each <div class="carousel-item"> to create a caption for each slide:

Example

```
<div class="carousel-item">

<img src="la.jpg" alt="Los Angeles">

<div class="carousel-caption">

<h3>Los Angeles</h3>

<p>We had such a great time in LA!</p>

</div>

</div>
```

5.What is bootstrap pagination and how are they classified?

Ans. Pagination:

Basic Pagination

If you have a web site with lots of pages, you may wish to add some sort of pagination to each page

To create a basic pagination, add the .pagination class to an <ul> element. Then add the .page-item to each <li> element and a .page-link class to each link inside <li>:

The basic pagination can be specified using the following classes.

- The **.pagination** class is used to specify pagination on a list group.

- The **.page-item** class is used to specify each pagination item in the group.
- The **.page-link** class is used to specify the link in the pagination item.

Example

```
<ul class="pagination">
 <li class="page-item"><a class="page-link" href="#">Previous</a></li>
 <li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">Next</a></li>
</ul>
```

Active State

The .active class is used to "highlight" the current page

Example

```
<ul class="pagination">
<li class="page-item"><a class="page-link" href="#">Previous</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item active"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">Next</a></li>
</ul>
```

Disabled State

The .disabled class is used for un-clickable links

Example

```
<ul class="pagination">
```

```
<li class="page-item disabled"><a class="page-link" href="#">Previous</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">Next</a></li>
</ul>
```

Pagination Sizing

Pagination blocks can also be sized to a larger or a smaller size

Add class .pagination-lg for larger blocks or .pagination-sm for smaller blocks

Example

```
<ul class="pagination pagination-lg">
<li class="page-item"><a class="page-link" href="#">Previous</a></li>
<li class="page-item"><a class="page-link" href="#">1</a></li>
<li class="page-item"><a class="page-link" href="#">2</a></li>
<li class="page-item"><a class="page-link" href="#">3</a></li>
<li class="page-item"><a class="page-link" href="#">Next</a></li>
</ul>
```

Pagination Alignment

Use utility classes to change the alignment of the pagination

Example

```
<!-- Default (left-aligned) -->
<ul class="pagination" style="margin:20px 0">
<li class="page-item">...</li>
</ul>
<!-- Center-aligned -->
<ul class="pagination justify-content-center" style="margin:20px 0">
```

```
<li class="page-item">...</li>

</ul>

<!-- Right-aligned -->

<ul class="pagination justify-content-end" style="margin:20px 0">

<li class="page-item">...</li>

</ul>
```

THE END

BY

A.VAISHNAVI DEVI