UNIT-1

1.Demonstrate DOCTYPE, Metadata Element, Division and Span elements in HTML with example program. (BTL 3 – Apply, CO1)

DOCTYPE Definition and Usage

All HTML documents must start with a <!DOCTYPE> declaration.

The declaration is not an HTML tag. It is an "information" to the browser about what document type to expect.

In HTML 5, the declaration is simple:
<!DOCTYPE html>

Metadata Element

The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.

<meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.

Metadata will not be displayed on the page, but is machine parsable.

Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<p>All meta information goes in the head section...</p>
</body>
</html>
```

DIV tag:

The tag defines a division or a section in an HTML document.

The tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.

The tag is easily styled by using the class or id attribute.

Any sort of content can be put inside the tag!

```
<html>
<head>
<style>
.myDiv {
  border: 5px outset red;
  background-color: lightblue;
  text-align: center;
}
</style>
</head>
<body>
<div class="myDiv">
  <h2>This is a heading in a div element</h2>
  <p>This is some text in a div element.</p>
</div>
</body>
</html>
```

SPAN TAG:

The tag is an inline container used to mark up a part of a text, or a part of a document.

The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.

The tag is much like the element, but is a block-level element and is an inline element.

```
<p>My mother has <span style="color:blue">blue</span> eyes.</p>
```

2.Demonstrate about Sectioning Elements (aside, header, footer, main, section, article, nav) in HTML with example program. Or Using HTML 5 tags design a simple web page. (BTL 3 – Apply, CO1)

ASIDE TAG:

The <aside> tag defines some content aside from the content it is placed in.

The aside content should be indirectly related to the surrounding content.

Tip: The <aside> content is often placed as a sidebar in a document.

```
<aside>
<h4>Epcot Center</h4>
<p>Epcot is a theme park at Walt Disney World Resort featuring exciting attractions, international pavilions, award-winning fireworks and seasonal special events.</p>
</aside>
```

HEADER TAG:

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

- one or more heading elements (<h1> - <h6>)
- logo or icon
- authorship information

```
<article>
  <header>
    <h1>A heading here</h1>
    <p>Posted by John Doe</p>
    <p>Some additional information here</p>
  </header>
  <p>Lorem Ipsum dolor set amet....</p>
</article>
```

FOOTER TAG:

The <footer> tag defines a footer for a document or section.

A <footer> element typically contains:

- authorship information
- copyright information
- contact information
- sitemap

- back to top links
- related documents

You can have several <footer> elements in one document.

```
<footer>
  <p>Author: Hege Refsnes</p>
  <p><a href="mailto:hege@example.com">hege@example.com</a></p>
</footer>
```

MAIN TAG:

The <main> tag specifies the main content of a document.

The content inside the <main> element should be unique to the document. It should not contain any content that is repeated across documents such as sidebars, navigation links, copyright information, site logos, and search forms.

Note: There must not be more than one <main> element in a document. The <main> element must NOT be a descendant of an <article>, <aside>, <footer>, <header>, or <nav> element.

SECTION TAG:

The <section> tag defines a section in a document.

```
<section>
<h2>WWF History</h2>
<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>
```

ARTICLE TAG:

The <article> tag specifies independent, self-contained content.

```
<article>
<h2>Google Chrome</h2>
<p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p>
</article>
```

NAV TAG:

The <nav> tag defines a set of navigation links.

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
```

```
  <a href="/python/">Python</a>
</nav>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta name="author" content="John Doe">
    <title>Document</title>
</head>
<body>
    <header>This is header tag</header>
    <nav>This is nav tag element</nav>
    <main>
        <section>
            This is section
            <article>
                this is some artical <aside>this is aside tag</aside>
            </article>
        </section>
    </main>
    <footer>This is footer</footer>
</body>
</html>
```

3.Determine different types of lists with an example program. (BTL 3 – Apply, CO1)

Unordered HTML List

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items will be marked with bullets (small black circles) by default:

----------→Disc,circle,square

Example:

```
<ul style="list-style-type:circle;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Ordered HTML List

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

------→’A’,’a’,’1’,’I’,’i’

Example:

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

HTML Description Lists

HTML also supports description lists.

A description list is a list of terms, with a description of each term.

The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example:

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```

4.Illustrate links, images with an example program. (BTL 3 – Apply, CO1)

HTML Links - Hyperlinks

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand

HTML Links - Syntax

The HTML <a> tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

The target attribute specifies where to open the linked document.

Visit W3Schools!

HTML IMAGES:

The HTML <img> tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

<img src="*url*" alt="*alternatetext*">

<img src="img_chania.jpg" alt="Flowers in Chania">

6.Using Table Elements (table, th, tr, td) and attributes (Colspan/Rowspan, border, cellspacing and cellpadding) design a static web page that displays an employee information table with three rows and three columns as shown below: (BTL 3 – Apply, CO1)

EID	ENAME	MOBILE
1	ABC	0123456789 9876543210
2	XYZ	9876543210 0123456789

Cell padding is the space between the cell edges and the cell content.
Cell spacing is the space between each cell.
To make a cell span over multiple columns, use the colspan attribute
To make a cell span over multiple rows, use the rowspan attribute

```
<!DOCTYPE html>
<html>
<head>
	<title>Document</title>
	<style>
		table {
			 border-spacing: 10px;
			border-padding:5px;
```

```
                }
        </style>
</head>
<body>
        <table border='1'>
                <tr>
                        <th>EID</th>
                        <th>ENAME</th>
                        <th colspan="2" >MOBILE</th>
                </tr>
                <tr>
                        <td>1</td>
                        <td>ABC</td>
                        <td>0123456789</td>
                        <td>9876543210</td>
                </tr>
                <tr>
                        <td>2</td>
                        <td>XYZ</td>
                        <td>9876543210</td>
                        <td>0123456789</td>
                </tr>
        </table>
</body>
</html>
```

EID	ENAME	MOBILE	
1	ABC	0123456789	9876543210
2	XYZ	9876543210	0123456789

7.Apply audio and video elements to insert media elements into a web page. (BTL 3 – Apply, CO1)

AUDIO TAG:

The HTML <audio> element is used to play an audio file on a web page.

```
<audio controls>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

To start an audio file automatically, use the autoplay attribute

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

VEDIO TAG:

The HTML <video> element is used to show a video on a web page.

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

To start a video automatically, use the autoplay attribute:

```
<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

UNIT-2

1.Discuss how to embed Javascript in a web page with example program. (K2 – Understand, CO2)

What is JavaScript

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document

The `<script>` tag is used to embed a client-side script (JavaScript).

The `<script>` element either contains scripting statements, or it points to an external script file through the src attribute.

Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content

```
<!DOCTYPE html>
<html>
<body>
<h1>The script element</h1>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
</body>
</html>
```

2.Explain about I/O statements with example programs. (K2 – Understand, CO2)

In JavaScript, we use the `prompt()` function to ask the user for input. As a parameter, we input the text we want to display to the user. Once the user presses “ok,” the input value is returned. We typically store user input in a variable so that we can use the information in our program.

```
var name = prompt("What is your name?");
```

```
println("Hello " + name + "!");
println("Name is a " + typeof name);
```

`readLine(prompt)`: Reads a string value from a user
`readInt(prompt)`: Reads an integer value from a user
`readFloat(prompt)`: Reads a float value from a user

```
var name = readLine("What is your name?");
var num = readInt("What is your favorite number? ");

println("Hello " + name + "!");
println(num + "?! That's my favorite number too!");
```

JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using innerHTML

To access an HTML element, JavaScript can use the document.getElementById(id) method.

The id attribute defines the HTML element. The innerHTML property defines the HTML content

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

<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
```

Using document.write()

For testing purposes, it is convenient to use document.write():

```
<script>
document.write(5 + 6);
</script>
```

Using window.alert()

You can use an alert box to display data:

```
<script>
window.alert(5 + 6);
</script>
```

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

```
<script>
console.log(5 + 6);
</script>
```

3.Illustrate control statements (conditional and loops) with syntax and examples. (K3 – Apply, CO2)

a.Program to find whether a given number is Prime or not

```
<!DOCTYPE html>
<html>
<body>
<script>
        function primeNumber(num) {
                for (var i = 2; i < num; i++) {
                        if (num % i === 0) {
                                return false;
                        }
                }
                return num > 1;
        }
        x = 7;
        document.write(primeNumber(x));
</script>
</body>
</html>
```

b.Take a number in between 0 to 999 in a text field and display in words in another text field

5.Discuss the advantage of Arrow function with example program. (K2 – Understand, CO2)

As we may able to visualize that the traditional function syntax, the use of curly braces are there, followed by function keyword and also it requires more number of executable lines, but as we may see the fact that arrow functions allow any user to shorten the syntax of writing the traditional function in a much more compact way.

Syntax for writing an Arrow Function: Following is the simple syntax of writing an arrow function:

```
let name_of_function = (parameters) => ...
```

We may also include any number of parameters inside the round braces

Advantages of using Arrow Function: The following points will describe the list of advantages which are associated with using Arrow functions instead of normal functions –

- This arrow function reduces lots of code and makes the mode more readable.
- Arrow function syntax automatically binds "this" to the surrounding code's context.
- Writing the arrow => is more flexible as compared with the writing **function** keyword.

- We may use arrow function syntax with our method associated with the array, like **map(), reduce(), filter()** since by using arrow function syntax instead of using normal function syntax one could easily read and understand as well as write the code more effectively.
- If we may use arrow functions while declaring promises and callbacks then it would be much easier for any user to understand the concept behind them otherwise by using traditional function syntax concepts like callback hells, promise chaining would eventually become more difficult to understand, or even writing would become a little complex.

```
<script>
   let checkNumber = (num) => num > 10 ?
      console.log("Yes") : console.log("No");

   checkNumber(5);
   checkNumber(10);
   checkNumber(20);
   checkNumber(32);
</script>
```

Output: The output of the above code snippet is as follows:

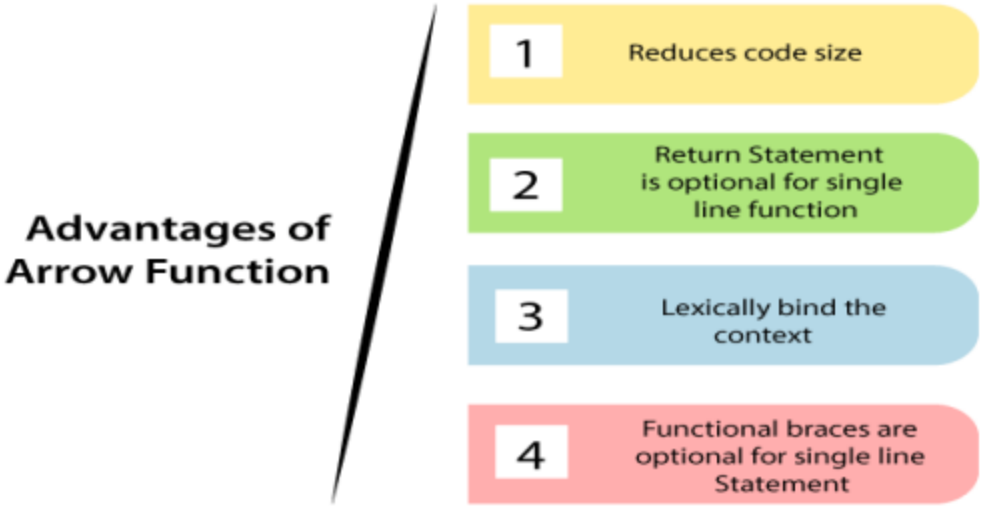
```
No

No

Yes

Yes
```

8.Discuss about Array creation, destructuring and accessing arrays with syntax and example programs. (K2 – Understand, CO2)



8.Discuss about Array creation, destructuring and accessing arrays with syntax and example programs. (K2 – Understand, CO2)

Creating an Array

Using an array literal is the easiest way to create a JavaScript Array.

Syntax:

```
const array_name = [item1, item2, ...];
```

It is a common practice to declare arrays with the const keyword.

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

```
const cars = [];
cars[0]= "Saab";
cars[1]= "Volvo";
cars[2]= "BMW";
```

Using the JavaScript Keyword new

The following example also creates an Array, and assigns values to it:

```
const cars = new Array("Saab", "Volvo", "BMW");
```

Accessing Array Elements

You access an array element by referring to the **index number**:

```
const cars = ["Saab", "Volvo", "BMW"];
let car = cars[0];
```

Destructuring means to break down a complex structure into simpler parts. With the syntax of destructuring, you can extract smaller fragments from objects and arrays. It can be used for assignments and declaration of a variable.

Destructuring is an efficient way to extract multiple values from data that is stored in arrays or objects. When destructuring an array, we use their positions (or index) in an assignment.

Let us try to understand the array destructuring by using some illustrations:

Example

```
var arr = ["Hello", "World"]


// destructuring assignment
var [first, second] = arr;


console.log(first); // Hello
console.log(second); // World
```

9.Illustrate Asynchronous programming using callback, promises, async, await and fetch with syntax and example program. (K3 – Apply, CO2)

Callback

In JavaScript, you can also pass a function as an argument to a function. This function that is passed as an argument inside of another function is called a callback function.

```
// function
function greet(name, callback) {
    console.log('Hi' + ' ' + name);
    callback();
}

// callback function
function callMe() {
    console.log('I am callback function');
}

// passing function as an argument
greet('Peter', callMe);
```

```
Hi Peter
I am callback function
```