AIT ASSIGNMENT 1

1. *What are HTML tags? How many tags are required to create a web page in HTML5?*

- HTML tags are elements used for structuring and formatting content in web pages.
- HTML5 has no strict requirements for the number of tags needed, as it varies based on page complexity.
- At a minimum, typical HTML5 web pages require:
  <!DOCTYPE html>: Declares HTML5 document type.
  <html>: Defines the root element.
  <**head>**: Contains meta-information.
  <title>: Sets the title displayed in the browser.
  <body>: Contains visible content.
- Additional tags are used to structure content, apply styles, and include multimedia elements.

```
<!DOCTYPE html>
<html>
<head>
    <title>My Web Page</title>
</head>
<body>
    <h1>Hello, World!</h1>
    <p>This is a simple web page.</p>
</body>
</html>
```

2. *Explain various formatting tags in HTML5.*

Text Formatting Tags:
<b>: Makes text bold.
****: Indicates strong emphasis, typically displayed as bold.
<i>: Renders text in italic style.
****: Indicates emphasis, typically displayed in italic style.
<u>: Underlines text.
<s>: Renders text with a strikethrough style.
<sup>: Displays text as superscript.
<sub>: Displays text as subscript.

Heading Tags:
<h1> to <h6>: Defines six levels of headings, where <h1> is the highest level and <h6> is the lowest.

Paragraph Tags:
<p>: Defines a paragraph.

Line Break Tags:
**
**: Inserts a single line break.

Horizontal Rule Tag:
<hr>: Inserts a horizontal line, typically used to separate content sections.

Preformatted Text Tag:
<pre>: Preserves white space and line breaks within the element, displaying text exactly as written in the HTML code.

Quotation Tags:
<blockquote>: Indicates a block of quoted text.
<q>: Indicates inline quoted text.

Citation and Definitions Tags:
<cite>: Represents the title of a work being cited.
<dfn>: Indicates the definition of a term.

Abbreviation and Acronym Tags:
<abbr>: Represents an abbreviation or acronym.
<acronym>: Represents an acronym (not supported in HTML5, but still used).

Code and Variable Tags:
<code>: Defines a piece of computer code.
<var>: Represents a variable in computer programming or mathematics.

Address Tag:
<address>: Defines contact information for the author or owner of a document.

Marked Text Tag:
<mark>: Highlights text for reference or notation purposes.

3. ***How many ways to integrate CSS in a web page? Explain.***

3 main ways to integrate CSS in web pages:

1. **Inline CSS:**
   Inline CSS is applied directly within the HTML elements using the `style` attribute. For example:

```
<p style="color: blue; font-size: 16px;">This is a paragraph with inline CSS.</p>
```

2. **Internal CSS:**
   Internal CSS is placed within the `<style>` element in the head section of the HTML document. It affects the styling of the entire document or specific elements. For example:

```
<head>
    <style>
        p {
            color: blue;
            font-size: 16px;
        }
    </style>
</head>
<body>
    <p>This is a paragraph with internal CSS.</p>
</body>
```

3. **External CSS:**
   External CSS is defined in a separate CSS file and then linked to the HTML document using the <link> element. This method allows for the reuse of styles across multiple HTML pages. For example:

```
<head>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

4. What do you mean by CSS Selector?

- CSS selectors are patterns used to select HTML elements for styling.
- They target elements based on various criteria like element type, class, ID, attributes, and relationships.

1. Element Selector:

```
p {
    color: blue;
}
```

2. Class Selector:

```
.highlight {
    background-color: yellow;
}
```

3. ID Selector:

```
#header {
    font-size: 24px;
}
```

4. Attribute Selector:

```
input[type="text"] {
    border: 1px solid #ccc;
}
```

5. Descendant Selector:

```
div p {
    font-style: italic;
}
```

5. Explain how to add audio to a web page.

<audio> tag is used to embed audio content into an HTML document.
Attributes:

- **src**: Specifies the URL of the audio file.
- **controls**: Displays built-in controls like play, pause, and volume.
- **autoplay**: Automatically starts playing the audio when the page loads.
- **loop**: Loops the audio playback.

- **preload**: Specifies if and how the audio should be loaded when the page loads (none, metadata, or auto).

```
<audio src="audio_file.mp3" controls autoplay loop preload="auto">
        <p>Your browser does not support the audio element.</p>
</audio>
```

6. ***Draw a square in HTML5 SVG, fill that square with yellow color, and make a 5px blue stroke width.***

```
<svg width="100" height="100">
  <rect x="10" y="10" width="80" height="80" fill="yellow" stroke="blue"
stroke-width="5" />
</svg>
```

- **<svg>**: Defines the SVG container.
- **<rect>**: Draws a rectangle.
- **x, y**: Position of the top-left corner of the rectangle.
- **width, height**: Width and height of the rectangle.
- **fill**: Fills the rectangle with yellow color.
- **stroke**: Specifies the stroke color (blue).
- **stroke-width**: Specifies the width of the stroke (5 pixels).

7. ***What is a module in NodeJS? Explain its types.***
   - A Module is a reusable piece of code encapsulating related functionality.
   - Helps organize code into separate files.
   - Can be imported into other files using require() or ES6 import statements.
   - Types: inbuilt, user-defined, community created

1. **Core Modules:**
   - Built-in modules provided by Node.js.
   - Included in Node.js distribution.
   - Accessed using require() without installation.

```
const fs = require('fs');
```

2. **User Defined Modules:**
   - Developer-created modules stored as .js files.
   - Encapsulate specific functionality.
   - Imported into other files using require().

```
.js
module.exports = {
    add: (a, b) => a + b,
    subtract: (a, b) => a - b
};
// app.js
const math = require('./math.js');
Third
```

3. **Third-party Modules:**
    - Created by the community and hosted on npm.
    - Installed using npm.
    - Imported into Node.js projects using require().

```
npm install express
```

```
const express = require('express');
```

8. Explain REPL in detail with an example.

REPL Definition:

- REPL stands for Read-Eval-Print Loop.
- It's an interactive programming environment in Node.js.
- Allows executing JavaScript code line by line.

Usage:

- Ideal for testing small code snippets.
- Useful for experimenting with Node.js features.

Features:

- Reads user input (Read).
- Evaluates the input (Eval).
- Prints the result (Print).
- Repeats the process (Loop).

Accessing REPL:

- Run node command in the terminal without any arguments.

```
$ node
> 1 + 2
3
> let message = "Hello, World!"
undefined
> message
'Hello, World!'
> message.toUpperCase()
'HELLO, WORLD!'
>
```

9. *What is NPM? What are its features?*

- npm(Node Package Manager) is a package manager for Node.js, used to install, manage, and share JavaScript packages.

Installing package:

```
npm i express
```

Managing dependencies via **package.json**:

```
// package.json
{
    "dependencies": {
        "express": "^4.17.1"
    }
}
```

Features of NPM:
Package Management:
Installs package: npm install package-name
Manages dependencies: npm install

Versioning:
Specifies versions: "express": "^4.17.1" in package.json

Script Execution:
Runs scripts: "start": "node server.js" in package.json

Installation:
Global: npm install -g package-name
Local: npm install package-name

Publishing:
Publishes packages: npm publish

Security:
Audits for vulnerabilities: npm audit

Search:
Searches for packages: npm search keyword

10. Step-by-step demonstration of web server creation.

1. **Initialize Project**: Create a new directory and initialize a Node.js project:

```
mkdir my-server
cd my-server
npm init -y
```

2. **Install Express.js framework:**

```
npm install express
```

3. **Create server.js:**

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
    res.send('Hello, World!');
});

app.listen(port, () => {
    console.log(`Server is running at http://localhost:${port}`);
});
```

4. **Run Server:**

```
node server.js
```

5. **Access Web Server:**

Open a web browser and go to http://localhost:3000. You should see "Hello, World!" displayed.