INTERNET & WEB TECHNOLOGIES

UNIT 2

Introduction to DHTML

DHTML, or Dynamic HTML, is a combination of HTML, JavaScript, and Cascading Style Sheets (CSS) used to create interactive and dynamic web pages. It is an extension of standard HTML that allows web developers to create highly interactive websites and applications that can respond to user actions without the need for server-side scripting. In this article, we will cover the basic concepts of DHTML and its key components.

Key Components of DHTML:

- ***HTML*** - HTML is the markup language used to define the structure of a web page. It provides the basic framework for creating web pages and includes elements such as headings, paragraphs, and lists.
- ***JavaScript -*** JavaScript is a client-side scripting language used to add interactivity to web pages. With JavaScript, you can change the behavior and appearance of a web page in response to user actions such as mouse clicks, button presses, and form submissions.
- ***Cascading Style Sheets (CSS)*** - CSS is a language used to define the visual style of a web page. It allows developers to create sophisticated page layouts and control the presentation of HTML elements. By using CSS, developers can separate the presentation of a web page from its content, making it easier to maintain and update.

Benefits of DHTML:

- ***Interactivity*** - DHTML provides the ability to create highly interactive web pages that can respond to user actions in real-time. This allows web developers to create more engaging and dynamic user experiences that can increase user engagement and satisfaction.
- ***Faster Performance*** - DHTML can improve the performance of web pages by reducing the need for server-side scripts. By using client-side scripting, DHTML can reduce the amount of data sent between the server and client, resulting in faster page loading times.
- ***Improved User Experience*** - DHTML can be used to create more engaging and interactive user experiences that can improve the overall satisfaction of website visitors. With DHTML, developers can create animations, menus, and other interactive elements that can make a website more appealing and easy to use.
- ***Cross-browser Compatibility*** - DHTML is designed to work across all major web browsers, ensuring that your website will be accessible to the widest possible audience.

Limitations of DHTML:

- ***Browser Support*** - DHTML can be limited by the support of different browsers. Different browsers may support different versions of HTML, CSS, and JavaScript, which can result in compatibility issues.
- ***Accessibility*** - DHTML can make web pages more difficult to navigate for users who rely on assistive technologies such as screen readers.
- ***Security*** - DHTML can create security vulnerabilities in web pages if not implemented properly. Developers must be careful when using client-side scripting to ensure that their code is secure and cannot be exploited by attackers.

Key features of DHTML

DHTML, or Dynamic HTML, is a combination of HTML, CSS, and JavaScript that allows for interactive and animated web pages. Some of the key features of DHTML include:

- ***Dynamic Content***: With DHTML, web developers can create web pages that are more interactive and engaging. This is achieved through the use of scripting languages such as JavaScript, which allows for the creation of dynamic content.
- ***Animation*** and Effects: DHTML allows for the creation of animations and effects on web pages, making them more visually appealing and engaging. This can be achieved through the use of CSS and JavaScript, which provide a range of animation and transition effects.
- ***Browser Compatibility***: DHTML is designed to be compatible with all major web browsers, making it accessible to a wide range of users. This ensures that web pages created using DHTML can be viewed and used on any device.
- ***Cross-platform Compatibility***: DHTML is also cross-platform compatible, meaning that it can be used on different operating systems, including Windows, Mac, and Linux. This allows web developers to create web pages that can be accessed from any device.
- ***Dynamic User Interface:*** DHTML allows for the creation of dynamic user interfaces that can change based on user interactions. This can be achieved through the use of JavaScript and CSS, which can be used to create interactive forms and menus.
- ***Improved Performance:*** DHTML can improve the performance of web pages by reducing the number of server requests needed to load content. This is achieved through the use of JavaScript and CSS, which can be used to load content dynamically, reducing the need for page refreshes.

Overall, DHTML is a powerful tool for web developers, providing a range of features that can be used to create interactive and engaging web pages. With its dynamic content, animation and effects, browser compatibility, and cross-platform compatibility, DHTML is a popular choice for web development.

CSS:

Types of Style sheets:

CSS stands for Cascading Style Sheets and it is used to describe how HTML elements are displayed on a web page. There are three types of style sheets in CSS:

- Inline Style Sheet
- Internal Style Sheet
- External Style Sheet

Let's discuss each one of them in detail.

Inline Style Sheet:

Inline Style Sheet is defined within the HTML element itself using the "style" attribute. It is used to apply styles to a single element on the web page.

Example of Inline Style Sheet:

```
<p style="color: red; font-size: 24px;">This text is styled using an inline style sheet</p>
```

In this example, we have applied two styles to the "p" element using the "style" attribute. The "color" property is set to "red" and the "font-size" property is set to "24px".

Internal Style Sheet:

Internal Style Sheet is defined within the head section of an HTML document using the "style" tag. It is used to apply styles to multiple elements on the web page.

Example of Internal Style Sheet:

```
<!DOCTYPE html>
<html>
<head>
    <title>Example of Internal Style Sheet</title>
    <style>
```

```
            p {    color: blue;
                    font-size: 18px;
                }
            h1 {
                    color: green;
                    font-size: 30px;
            }
        </style>
</head>
<body>
        <h1>This is a heading</h1>
        <p>This is a paragraph</p>
</body>
</html>
```


In this example, we have defined two styles using the "style" tag. The first style applies to all "p" elements on the web page, and the second style applies to all "h1" elements on the web page.

External Style Sheet:

External Style Sheet is a separate file that contains all the styles that are used on a web page. It is linked to the HTML document using the "link" tag. It is used to apply styles to multiple web pages on a website.

Example of External Style Sheet:

```
<!DOCTYPE html>
<html>
<head>
        <title>Example of External Style Sheet</title>
        <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

```
<body>
    <h1>This is a heading</h1>
    <p>This is a paragraph</p>
</body>
</html>
```

this example, we have linked an external style sheet "styles.css" to the HTML document using the "link" tag. The "styles.css" file contains all the styles that are applied to the web page.

codechamp.online example:

CodeChamp.online website uses external style sheets to apply styles to its web pages. The website has a main style sheet called "style.css" which contains all the styles that are used on the web pages. Here is an example of the "style.css" file:

```
/* Global Styles */
body {
  font-family: 'Montserrat', sans-serif;
  font-size: 16px;
  line-height: 1.5;
  color: #333;
  background-color: #f2f2f2;
}

a {
  color: #333;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
```

```
}


/* Header Styles */

header {

  background-color: #fff;

  padding: 20px;

  box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);

}

.header-logo {

  display: inline-block;

  margin: 0;

  font-size: 24px;

  font-weight: bold;

  color
```


Different elements of Style sheets:

Style sheets refer to the set of rules and guidelines that govern how the visual elements of a website are presented. In website development, style sheets can be created using different languages such as CSS, SASS, and LESS.

Selectors:

CSS selectors are used to select and style specific elements on a web page. They can be based on the element type, class, ID, and attribute. Selectors are written followed by the curly braces {} that contain the property-value pairs that define the style.

Example:

```
/* Element Selector */

p {

  color: #333;

  font-size: 16px;
```

```
}


/* Class Selector */
.navbar {
    background-color: #f8f8f8;
}


/* ID Selector */
#header {
    height: 100px;
}


/* Attribute Selector */
a[target="_blank"] {
    color: blue;
}
```


Properties:

CSS properties define the visual appearance of the selected elements. They are written in property-value pairs within the curly braces {} of the selector. There are numerous properties, including color, font-size, background-color, padding, margin, and many more.

Example:/* Color Property */

```
p {
    color: #333;
}


/* Font-size Property */
```

```
h1 {
   font-size: 32px;
}

/* Background-color Property */
body {
   background-color: #fff;
}

/* Padding Property */
.navbar {
   padding: 10px;
}
```


Values:

CSS values define the specific aspect of the selected property. For instance, the color property can have values such as hexadecimal color codes (#fff, #333), RGB color values (rgb(255,255,255)), or color keywords (red, blue).

Example:

```
/* Hexadecimal Color Value */
p {
   color: #333;
}

/* RGB Color Value */
body {
   background-color: rgb(255, 255, 255);
```

```
}


/* Color Keyword */

h1 {

  color: red;

}
```

Inheritance:

In CSS, styles can be inherited from parent to child elements. For example, a font style set on the body element will be inherited by all the child elements unless specifically overridden.

Example:

```
/* Parent Element */

body {

  font-family: Arial, sans-serif;

}


/* Child Element */

h1 {

  font-size: 32px;

}
```


Specificity:

Specificity refers to the weight of a selector, which determines which style will be applied if there are conflicting styles. The more specific selector has a higher weight and overrides less specific selectors. For example, an ID selector has a higher specificity than a class selector.

Example:

```
/* Class Selector */

.navbar {

  background-color: #f8f8f8;

}
```

```
/* ID Selector */

#header {

   background-color: #333;

}
```

Media Queries:

Media queries allow you to apply different styles based on the screen size or device. This allows for responsive design where the website will adapt to different devices.

Example:

```
/* Media Query */

@media only screen and (max-width: 768px) {

   /* Styles for screens with max width of 768px */

   .navbar {

      background-color: #333;

   }

}
```


IFrame:

An IFrame (Inline Frame) is an HTML document embedded within another HTML document. It is a container for a web page that is embedded within another web page. IFrames allow you to display another website within your own website.

Example of using IFrame:

```
<iframe src="https://www.codechamp.online" width="100%" height="400"></iframe>
```

This code will embed the website "https://www.codechamp.online" within your website, with a width of 100% and a height of 400 pixels.

DIV:

A DIV is an HTML tag that is used to create a container for HTML elements. It is used to group and organize content on a web page. You can use the DIV tag to style and position different elements on a web page.

Example of using DIV:

```
<div>
  <h1>Heading</h1>
  <p>Paragraph</p>
  <img src="image.jpg">
</div>
```

This code will create a container for the Heading, Paragraph, and Image elements on your web page.

Layer Tag:

The Layer tag was used in HTML 4 to position content on a web page. It has since been deprecated and is no longer used in HTML 5. The DIV tag is now used for positioning content on a web page.

Example of using Layer tag:

```
<div style="position: absolute; top: 50px; left: 50px; width: 200px; height: 200px; background-color: red;">
  This is a layer
</div>
```


This code will create a red box on your web page, positioned 50 pixels from the top and 50 pixels from the left. The box will be 200 pixels wide and 200 pixels high. The text "This is a layer" will appear within the box.

XML (Extensible Markup Language)

XML (Extensible Markup Language) is a markup language that is designed to store, transport, and exchange data between different systems. XML is a flexible and open standard that is widely used in web development, data integration, and application programming interfaces

Syntax of XML

XML uses a set of rules and syntax to define the structure of data. The basic syntax of XML includes:

Elements: An element is a building block of XML, and it contains a start tag, end tag, and content. For example:

```
<book>
  <title>Harry Potter and the Philosopher's Stone</title>
```

<author>J.K. Rowling</author>

<year>1997</year>

</book>

Tags: A tag is used to define the start and end of an element. Tags are enclosed in angle brackets (< >). For example:

<book>

Attributes: An attribute provides additional information about an element. Attributes are added to the start tag of an element. For example:

<book category="fiction">

Entities: Entities are used to represent special characters such as &, <, and > in XML. For example:

< represents < (less than)

> represents > (greater than)

& represents & (ampersand)

Structure of XML

XML has a hierarchical structure, which means that data is organized in a tree-like structure. The root element is the top-level element, and all other elements are nested inside it. For example:

<library>

<book>

<title>Harry Potter and the Philosopher's Stone</title>

<author>J.K. Rowling</author>

<year>1997</year>

</book>

<book>

<title>The Lord of the Rings</title>

<author>J.R.R. Tolkien</author>

<year>1954</year>

```
  </book>
</library>
```

In the above example, the root element is <library>, and it contains two <book> elements. Each <book> element contains a <title>, <author>, and <year> element.

Usage of XML(APIs)

XML has a wide range of applications, including:

- ***Data exchange:*** XML is used to exchange data between different systems and applications. For example, XML is used in web services to exchange data between web applications.
- ***Data integration***: XML is used to integrate data from different sources and systems. For example, XML is used in ETL (Extract, Transform, Load) processes to transform and load data into a data warehouse.
- ***Configuration files:*** XML is used to store configuration information for applications and systems. For example, XML is used in Spring framework to configure application contexts.
- ***Web development***: XML is used in web development to define and structure data. For example, XML is used in XHTML (Extensible Hypertext Markup Language) to define web page content.

SGML

SGML is a markup language that was developed in the 1980s to standardize the representation of text-based documents. It is a powerful language that allows for the creation of complex and flexible document structures. SGML is used as the basis for many other markup languages, including HTML, XML, and LaTeX.

SGML is an application of a document markup language (DML), a family of markup languages that includes HTML, XML, and LaTeX. It uses tags to define the structure and content of a document. SGML documents are made up of elements, which can be nested to create complex document structures.

SGML uses a document type definition (DTD) to define the structure of a document. The DTD specifies the allowed elements, attributes, and their relationships. A DTD can be used to validate a document, ensuring that it conforms to the specified structure.