# HTML5, CSS3

## Unit - 1 : Content

**HTML 5**

- Introduction
- Semantic Elements
- Web Storage API
- HTTP Status Codes

**Cascading Style Sheets (CSS) 3**

- Types of CSS: inline, internal, and external
- CSS Selectors
- Box model
- Grid
- Flexbox
- Responsive web design using media
- Viewport
- Transition
- Animation

## Introduction

- HTML stands for Hyper Text Markup Language.
- It is used to design web pages using a markup language.
- Its functioning depends on various tags that are categorized in two ways including opening tags and closing tags.
  **Example:**

```
    <b>and closing tags </b>.
```

**HTML5:**

- HTML 5 is the fifth and current version of HTML.
- It has improved the markup available for documents.
- It has introduced application programming interfaces (API) and Document Object Model (DOM).

## Introduction

**Features:**

- It has introduced new multimedia features like `<audio>` and `<video>` tags
- There are new graphics elements including vector graphics.
- Enrich semantic content by including semantic tags like `<header>` `,<footer> etc.,`
- It helps to locate the geographical location of a client.
- Allows drawing various shapes like triangle, rectangle, circle, etc.
- Capable of handling incorrect syntax.

## Introduction

**Advantages:**

- All browsers supported.
- More device friendly.
- Easy to use.
- HTML 5 in integration with CSS, JavaScript, helps us to build beautiful websites.

## Semantic Elements

A semantic element clearly describes its meaning to both the browser and the developer.
Examples of semantic elements in HTML: `<form>`, `<table>` - Clearly defines its content.
**HTML 5 Semantic Elements:**
In HTML 5 there are few more Semantic Elements added to make a web page more easy to read and understand and those are.

| Tag | Description |
|---|---|
| `<header>` | Defines a header for a document or section. |
| `<footer>` | Defines a footer for a document or section. |
| `<hgroup>` | Groups a set of `<h1>` to `<h6>` elements when a heading has multiple level. |
| `<mark>` | Defines marked/highlighted text. |
| `<section>` | Defines a section in a document. |

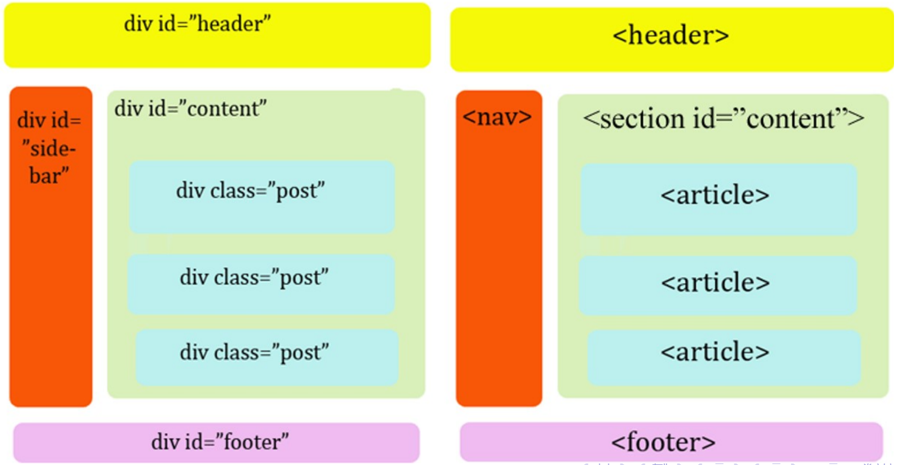
## Semantic Elements

Few more...

| Tag | Description |
|---|---|
| `<main>` | Defines the main content of a document. |
| `<nav>` | Defines navigation links. |
| `<wbr>` | Defines a possible line-break. |
| `<article>` | Defines an article. |
| `<aside>` | Defines content aside from the page content. |
| `<figure>` | Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc. |
| `<figcaption>` | Defines a caption for a `<figure>` element |

## Semantic Elements

Structure of a Webpage with HTML and HTML 5 tags...



## Semantic Elements

Example using Semantic Elements:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <title>HTML5 Semantic Elements Example</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            margin: 0;
            padding: 0;
        }
        header, nav, section, article, aside, footer {
            padding: 1em;
            border: 1px solid #ccc;
            margin: 10px;
```

## Semantic Elements

Example continued using Semantic Elements:

```
        }
        header, footer {
            background: #f4f4f4;
        }
        nav {
            background: #eee;
        }
    </style>
</head>
<body>
    <header>
        <h1>My Website</h1>
        <p>Welcome to my website!</p>
    </header>
    <nav>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
```

## Semantic Elements

Example continued using Semantic Elements:

```
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
    <main>
        <section>
            <h2>About Us</h2>
            <p>This is the about us section where we
    describe our company.</p>
        </section>
        <section>
            <h2>Services</h2>
            <article>
                <h3>Web Design</h3>
                <p>We provide web design services to help
    you build your online presence.</p>
            </article>
            <article>
                <h3>SEO</h3>
```

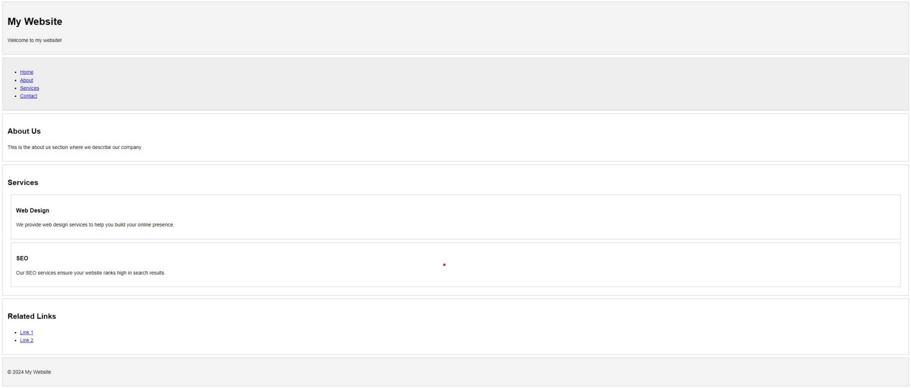
## Semantic Elements

Example continued using Semantic Elements:

```
                <h3>SEO</h3>
                <p>Our SEO services ensure your website
    ranks high in search results.</p>
            </article>
        </section>
        <aside>
            <h2>Related Links</h2>
            <ul>
                <li><a href="#">Link 1</a></li>
                <li><a href="#">Link 2</a></li>
            </ul>
        </aside>
    </main>
    <footer>
        <p>&copy; 2024 My Website</p>
    </footer>
</body>
</html>
```

## Semantic Elements

Output:



My Website

Welcome to my website!

- Home
- About
- Services
- Contact

About Us

This is the about us section where we describe our company.

Services

Web Design

We provide web design services to help you build your online presence.

SEO

Our SEO services ensure your website ranks high in search results.

Related Links

- Link 1
- Link 2

© 2024 My Website

## Semantic Elements

**One More Example: index.html**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <title>Photo Gallery</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>SNIST Photo Gallery</h1>
    </header>

    <main>
        <section class="gallery">
            <figure>
```

## Semantic Elements

```
        <img src="photo1.jpg" alt="Description of photo 1">
        <figcaption>Photo 1 - Beautiful </figcaption>
            </figure>
            <figure>
        <img src="photo2.jpg" alt="Description of photo 2">
        <figcaption>Photo 2 - Skyline</figcaption>
            </figure>
            <figure>
        <img src="photo3.jpg" alt="Description of photo 3">
        <figcaption>Photo 3 - Top View</figcaption>
            </figure>
            <figure>
        <img src="photo4.jpg" alt="Description of photo 4">
        <figcaption>Photo 4 - Over All</figcaption>
            </figure>
            <!-- Add more photos as needed -->
        </section>
    </main>
```

## Semantic Elements

```
    <footer>
        <p>&copy; 2024 Photo Gallery. All rights reserved.</
   p>
    </footer>
</body>
</html>
```

**styles.css:**

```
    /* External CSS Styles */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    background-color: #f4f4f4;
}
header {
    background-color: #333;
    color: white;
```

## Semantic Elements

```
    text-align: center;
    padding: 1rem;
}
header h1 {
    margin: 0;
    font-size: 2.5rem;
}
main {
    padding: 2rem;
    max-width: 1200px;
    margin: auto;
}
.gallery {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1
   fr));
    gap: 1rem;
}
figure {
```

## Semantic Elements

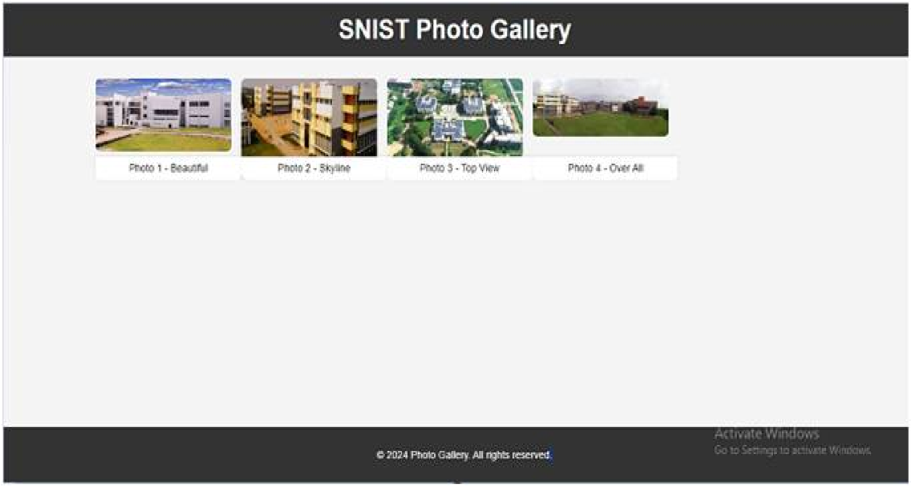
```
    margin: 0;
    position: relative;
}
figure img {
    width: 100%;
    height: auto;
    display: block;
    border-radius: 8px;
}
figcaption {
    background-color: #fff;
    padding: 0.5rem;
    text-align: center;
    border-radius: 0 0 8px 8px;
    position: absolute;
    bottom: 0;
    width: 100%;
    box-shadow: 0 -1px 4px rgba(0, 0, 0, 0.1);
}
```

## Semantic Elements

```
footer {
    background-color: #333;
    color: white;
    text-align: center;
    padding: 1rem;
    position: fixed;
    bottom: 0;
    width: 100%;
}
@media (max-width: 768px) {
    .gallery {
        grid-template-columns: repeat(auto-fill, minmax(150
   px, 1fr));
    }

    header h1 {
        font-size: 2rem;
    }
}
```

## Semantic Elements

**Output:**



SNIST Photo Gallery

Photo 1 - Beautiful

Photo 2 - Skyline

Photo 3 - Top View

Photo 4 - Over All

© 2024 Photo Gallery. All rights reserved.

## Web Storage API

- Before HTML5, application data had to be stored in cookies, which is affecting website performance.
- Web storage is introduced in HTML 5 is more secure, and large amounts of data can be stored locally.
- Web storage does not affect the Website performance.
- The storage limit is far larger (at least 5MB) and information is never transferred to the server.

`HTML web storage provides two objects for storing data on the client:`

1. Local Storage: Stores data with no expiration date.
2. Session Storage: Stores data for one session (data is lost when the browser tab is closed)

## Web Storage API

Before we use either Local or Session storage we need to check whether browser supports web storage or not.

**Code to check for the support:**

```
if (typeof(Storage) !== "undefined") {
  console.log('Supports web storage');
} else {
  console.log('Does not Support web storage');
}
```

**Local Storage:**

It is used to store data on the client side. It has no expiration time, so the data in the LocalStorage exists always till the user manually deletes it.

## Web Storage API

**Local storage provides following methods:**

- setItem():

  `This method takes two parameters one is key and another one is value.  It is used to add/update the value in a particular location with the name of the key.`

  **Syntax:**

  ```
  localStorage.setItem(key, value);
  ```

  **Example:**

  ```
  localStorage.setItem("FirstName", "Vijay");
  ```

## Web Storage API

- getItem():

  This method returns the value of the specified key name

  **Syntax:**

  `localStorage.getItem(key);`

  **Example:**

  `localStorage.getItem("FirstName");`

- removeItem():

  This method removes that key from the storage

  **Syntax:**

  `localStorage.removeItem(key);`

  **Example:**

  `localStorage.removeItem("FirstName");`

## Web Storage API

- clear():

  `This method empty all key out of the storage`

  **Syntax:**

  `localStorage.clear();`
- key(n):

  `This method gives the name of key at the specified index.`

  **Syntax:**

  `localStorage.key(n);`

  **Example:**

  `localStorage.key(0);`

**Example:**

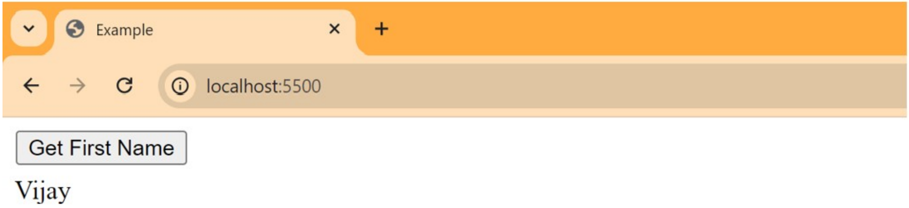
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
```

## Web Storage API

```
width, initial-scale=1.0">
        <title>Example</title>
    <script type="text/javascript">
        if(typeof(Storage)!="undefined"){
    localStorage.setItem("FirstName","Vijay");
        }
        else{
    console.log('Does not Supports web storage');
        }
        function getvalue(){
            document.getElementById("p1").innerText=
   localStorage.getItem("FirstName");
        }
    </script>
</head>
<body>
    <input type="button" onclick="getvalue()" value="Get
   First Name"/>
    <p id="p1"></p></body></html>
```

## Web Storage API

**Example Continued...:**
**Output:**



Vijay

**length:** It is the property of localStorage Object that Returns the number of data items stored in the Storage object.

## Web Storage API

**Session Storage:**

- It is used to store data on the client side.
- SessionStorage object stores data for one session.
- The data is deleted when the browser is closed.

**Session storage provides following methods same as Local Storage:**

- setItem():
  `This method takes two parameters one is key and another one is value. It is used to add/update the value in a particular location with the name of the key.`
  **Syntax:**
  ```
  sessionStorage.setItem(key, value);
  ```
  **Example:**
  ```
  sessionStorage.setItem("FirstName", "Vijay");
  ```

## Web Storage API

- getItem():
  `This method returns the value of the specified key name`
  **Syntax:**
  `sessionStorage.getItem(key);`
  **Example:**
  `sessionStorage.getItem("FirstName");`
- removeItem():
  `This method removes that key from the storage`
  **Syntax:**
  `sessionStorage.removeItem(key);`
  **Example:**
  `sessionStorage.removeItem("FirstName");`

## Web Storage API

- clear():

  This method empty all key out of the storage

  **Syntax:**

  `sessionStorage.clear();`
- key(n):

  This method gives the name of key at the specified index.

  **Syntax:**

  `sessionStorage.key(n);`

  **Example:**

  `localStorage.key(0);`

**Example:**

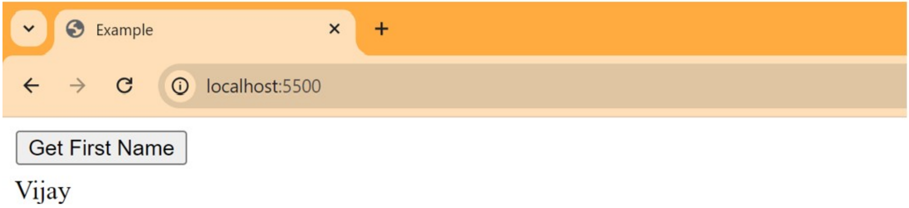
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
```

## Web Storage API

```
width, initial-scale=1.0">
        <title>Example</title>
    <script type="text/javascript">
        if(typeof(Storage)!="undefined"){
    sessionStorage.setItem("FirstName","Vijay");
        }
        else{
    console.log('Does not Supports web storage');
        }
        function getvalue(){
            document.getElementById("p1").innerText=
   sessionStorage.getItem("FirstName");
        }
    </script>
</head>
<body>
    <input type="button" onclick="getvalue()" value="Get
   First Name"/>
    <p id="p1"></p></body></html>
```

## Web Storage API

**Example Continued...:**
**Output:**



Vijay

**length:** `It is the property of localStorage Object that Returns the number of data items stored in the Storage object.`

## Web Storage API

**Key Differences between Local Storage and Web Storage:**

| **Local Storage** | **Session Storage** |
|---|---|
| The storage capacity of local storage is 5MB/10MB | The storage capacity of session storage is 5MB. |
| It must be deleted manually | The data is stored only for the duration of a session, i.e., until the browser (or tab) is closed. |

## HTTP Status Codes

- The Status-Code element is a server response.
- It is a 3-digit integer where the first digit of the Status-Code defines the class of response, last two digits do not have any categorization role.

| Code | Description |
|---|---|
| `1xx: Informational` | This means request has been received and the process is continuing. |
| `2xx: Success` | This means the action was successfully received, understood, and accepted. |

## HTTP Status Codes

| **Code** | **Description** |
|---|---|
| `3xx: Redirection` | This means further action must be taken in order to complete the request. |
| `4xx: Client Error` | This means the request contains incorrect syntax or cannot be fulfilled. |
| `5xx: Server Error` | This means the server failed to fulfill an apparently valid request. |

## HTTP Status Codes

**1xx: Information:**

| **Message** | **Description** |
|---|---|
| `100 Continue` | Only a part of the request has been received by the server, but as long as it has not been rejected, the client should continue with the request. |
| `101 Switching Protocols` | This means the request contains incorrect syntax or cannot be fulfilled. |

## HTTP Status Codes

**2xx: Success:**

| Message | Description |
|---|---|
| 200 OK | The request is OK. |
| 201 Created | The request is complete, and a new resource is created. |
| 202 Accepted | The request is accepted for processing, but the processing is not complete. |
| 203 Non-authoritative Information | The information in the entity header is from a local or third-party copy, not from the original server. |
| 204 No Content | A status code and a header are given in the response, but there is no entity-body in the reply. |

## HTTP Status Codes

**2xx: Success:**

| **Message** | **Description** |
|---|---|
| 205 Reset Content | The browser should clear the form used for this transaction for additional input. |
| 206 Partial Content | The server is returning partial data of the size requested. |

## HTTP Status Codes

**3xx: Redirection:**

| Message | Description |
|---|---|
| `300 Multiple Choices` | A link list. The user can select a link and go to that location. Maximum five addresses. |
| `301 Moved Permanently` | The requested page has moved to a new url. |
| `302 Found` | The requested page has moved temporarily to a new url. |
| `303 See Other` | The requested page can be found under a different url. |
| `304 Not Modified` | This Specifies where the URL has not been modified since the specified date. |

## HTTP Status Codes

**3xx: Redirection:**

| **Message** | **Description** |
|---|---|
| `305 Use Proxy` | The requested URL must be accessed through the proxy mentioned in the Location header. |

**4xx: Redirection:**

| **Message** | **Description** |
|---|---|
| `400 Bad Request` | The server did not understand the request. |
| `401 Unauthorized` | The requested page needs a username and a password. |
| `402 Payment Required` | You can not use this code yet. |
| `403 Forbidden` | Access is forbidden to the requested page. |

## HTTP Status Codes

**4xx: Client Error:**

| **Message** | **Description** |
|---|---|
| `404 Not Found` | The server can not find the requested page. |
| `405 Method Not Allowed` | The method specified in the request is not allowed. |
| `406 Not Acceptable` | The server can only generate a response that is not accepted by the client. |

## HTTP Status Codes

**5xx: Server Error:**

| Message | Description |
|---|---|
| `500 Internal Server Error` | The request was not completed. The server met an unexpected condition. |
| `501 Not Implemented` | The request was not completed. The server did not support the functionality required. |
| `502 Bad Gateway` | The request was not completed. The server received an invalid response from the upstream server. |

## HTTP Status Codes

**5xx: Server Error:**

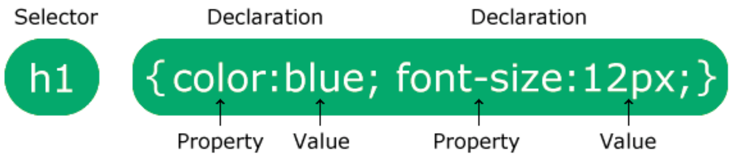
| **Message** | **Description** |
|---|---|
| `503 Service Unavailable` | The request was not completed. The server is temporarily overloading or down. |
| `504 Gateway Timeout` | The gateway has timed out. |
| `505 HTTP Version Not Supported` | The server does not support the "http protocol" version. |

## CSS Introduction

CSS (Cascading Style Sheets) is used to style the appearance of HTML elements on a webpage. There are three main types of CSS:

- Inline CSS
- Internal CSS
- External CSS

**CSS Structure:**



## Inline CSS

Inline CSS is applied directly to an HTML element using the ‘style‘ attribute.

- Example:

```
<p style="color: blue; font-size: 16px;">Hello</p>
```

- Inline CSS affects only the specific element it is applied to and overrides any external or internal CSS.
- It is useful for applying unique styles to individual elements.

## Internal CSS

Internal CSS is defined within the <style> element in the <head> section of an HTML document.

- Example:

```
    <style>
        p {
            color: red;
            font-size: 18px;
        }
    </style>
```

- Internal CSS affects all elements with the specified selectors within the same HTML document.
- It is useful for applying styles consistently across multiple elements within a single document.

## External CSS

External CSS is defined in a separate CSS file and linked to an HTML document using the `<link>` element.

- Example:

```
<link rel="stylesheet" href="styles.css">
```

- External CSS affects all elements with the specified selectors across multiple HTML documents.
- It allows for better organization and maintenance of styles, especially for large websites with multiple pages.

## HTML Example with CSS I

Below is an HTML example that demonstrates the use of different CSS selectors and properties with inline, internal, and external CSS:

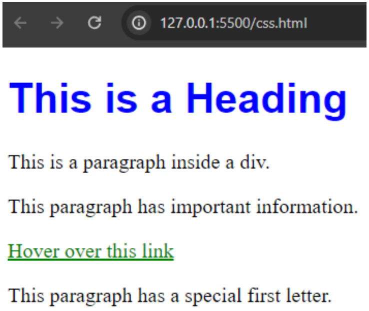
```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CSS Example</title>
    <!-- External CSS -->
    <link rel="stylesheet" href="styles.css">
    <style>
        /* Internal CSS */
        h1 {
            color: blue;
        }
    </style>
</head>
```

## HTML Example with CSS II

```
<body>
    <!-- Inline CSS -->
    <h1 style="font-family: Arial;">This is a Heading</h1>
    <div>
        <p>This is a paragraph inside a div.</p>
    </div>
    <div>
        <p data-info="important">This paragraph has
   important information.</p>
    </div>
    <a href="#" style="color: green;">Hover over this link</
   a>
    <p>This paragraph has a special first letter.</p>
</body>
</html>
```

## HTML Example with CSS III

Output:



127.0.0.1:5500/css.html

This is a Heading

This is a paragraph inside a div.

This paragraph has important information.

Hover over this link

This paragraph has a special first letter.

## CSS Selectors

- CSS selectors are used to target HTML elements.
- CSS properties are used to define the visual style of targeted elements.

Different types of CSS 3 selectors

- Basic selectors
- Combinatory selectors
- Attribute selectors
- Pseudo-class selectors
- Pseudo-element selectors

## CSS Selectors: Basic

- Type Selector: Selects all elements of a specified type.
  - Syntax: `element`
  - Example: `p { color: blue; }`
- Universal Selector: Selects all elements on the page.
  - Syntax: `*`
  - Example: `* { margin: 0; padding: 0; }`
- Class Selector: Selects elements with a specific class attribute.
  - Syntax: `.classname`
  - Example: `.highlight { background-color: yellow; }`
- ID Selector: Selects a single element with a specific ID attribute.
  - Syntax: `#idname`
  - Example: `#header { font-size: 24px; }`
- Attribute Selector: Selects elements with a specific attribute.
  - Syntax: `[attribute]`
  - Example: `[type="text"] { border: 1px solid black; }`

## CSS Selectors: Combinatory

- Combinatory selectors allow you to select elements based on their relationship to other elements.
- Common combinators:
  - Descendant (`<space>`): Selects all elements that are descendants of a specified element.
  - Child (>) : Selects all direct child elements specified by a parent element.
  - Adjacent sibling (+): Selects the next sibling element that follows a specified element.
  - General sibling (˜): Selects all sibling elements that follow a specified element.

## CSS Selectors: Attribute

- Attribute selectors allow you to select elements based on their attributes and attribute values.
- Common attribute selectors:
  - `[attribute]`: Selects all elements with the specified attribute.
  - `[attribute=value]`: Selects all elements with the specified attribute and value.
  - `[attribute =value]`: Selects all elements with an attribute value containing a specified word.
  - `[attribute|=value]`: Selects all elements with an attribute value starting with a specified value.

## CSS Selectors: Pseudo-class and Pseudo-element

- Pseudo-classes are used to define the special state of an element.
  - `:hover` - Selects an element when the mouse pointer is over it.
  - `:active` - Selects an element when it is being activated by the user.
  - `:focus` - Selects an element when it has focus.
  - `:nth-child(n)` - Selects every nth child element.
- Pseudo-elements allow you to style certain parts of an element.
  - `::before` - Inserts content before the content of an element.
  - `::after` - Inserts content after the content of an element.
  - `::first-line` - Selects the first line of text within an element.
  - `::first-letter` - Selects the first letter of text within an element.

## HTML Example with CSS Selectors I

Below is an HTML example that demonstrates the use of different CSS selectors and properties:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>CSS Example</title>
    <style>
        /* Basic selector */
        h1 {
            color: blue;
        }

        /* Combinatory selector */
        div p {
            font-size: 18px;
        }
```

## HTML Example with CSS Selectors II

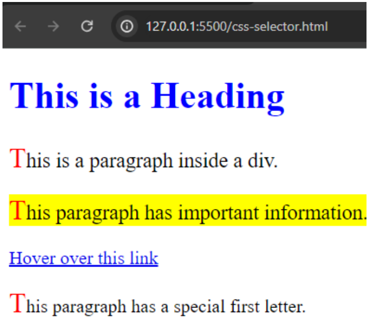
```
        /* Attribute selector */
        [data-info="important"] {
            background-color: yellow;
        }

        /* Pseudo-class selector */
        a:hover {
            text-decoration: underline;
        }

        /* Pseudo-element selector */
        p::first-letter {
            font-size: 24px;
            color: red;
        }
    </style>
</head>
```

## HTML Example with CSS Selectors III

```
<body>
    <h1>This is a Heading</h1>
    <div>
        <p>This is a paragraph inside a div.</p>
    </div>
    <div>
        <p data-info="important">This paragraph has
    important information.</p>
    </div>
    <a href="#">Hover over this link</a>
    <p>This paragraph has a special first letter.</p>
</body>
</html>
```

## HTML Example with CSS Selectors IV

Output:


127.0.0.1:5500/css-selector.html

This is a Heading

This is a paragraph inside a div.

This paragraph has important information.

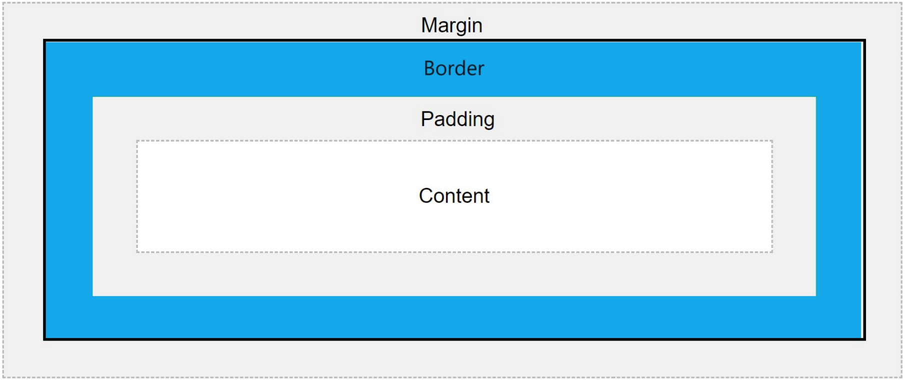
Hover over this link

This paragraph has a special first letter.

## Box Model

- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of: content, padding, borders and margins.

**Box model Example:**



## Box Model

- Content: The content of the box, where text and images appear
- Padding: Clears an area around the content. The padding is transparent.
- Border: A border that goes around the padding and content
- Margin: Clears an area outside the border. The margin is transparent

**Example CSS of a box model:**

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

## Box Model

**Example:**

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <title>Example</title>
<style type="text/css">
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>
<div>SNIST Has Remarkable Placements Every Year! <br> <br>
```

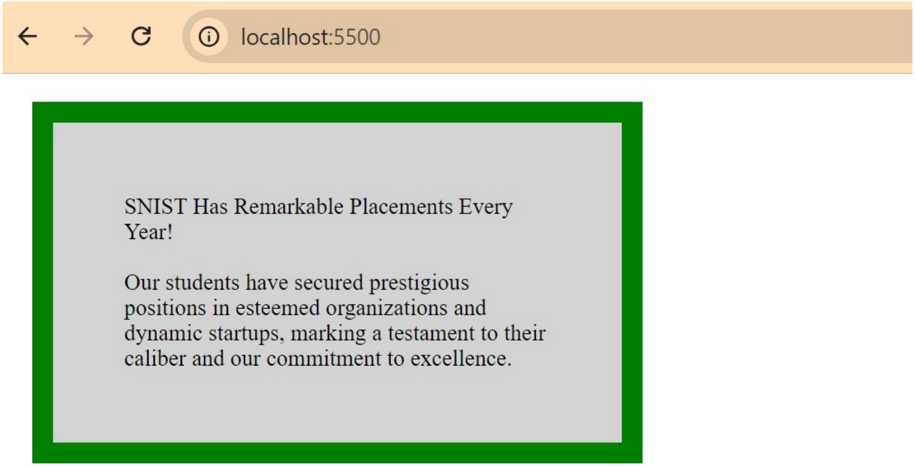
## Box Model

**Example Continued:**

```
<div>SNIST Has Remarkable Placements Every Year! <br> <br>

Our students have secured prestigious positions in esteemed
    organizations and dynamic startups, marking a testament
    to their caliber and our commitment to excellence.</div>
</body>
</html>
```

## Box Model

**Output:**



SNIST Has Remarkable Placements Every Year!

Our students have secured prestigious positions in esteemed organizations and dynamic startups, marking a testament to their caliber and our commitment to excellence.

## Grid

Grid is a two-dimensional grid-based layout system with rows and columns

The majorly used grid layout properties are:

| **Property** | **Description** |
|---|---|
| `display: grid` | It offers a grid-based layout system, with rows and columns. |
| `grid-template-columns` | It is used to set the number of columns and size of the columns of the grid. |
| `grid-template-rows` | It is used to set the number of rows and height of the rows in a grid. |
| `gap` | It is used to set the spacing also caller gutter between the rows and columns. |

## Grid

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <style>
        .grid-container {
            display: grid;
            grid-template-columns: 1fr 2fr 1fr;  /* Defines
   the columns */
            grid-template-rows: 100px 100px auto;  /*
   Defines the rows */
            gap: 10px;  /* Space between grid items */
            padding: 10px;
            background-color: green;
        }
        .grid-item {
```

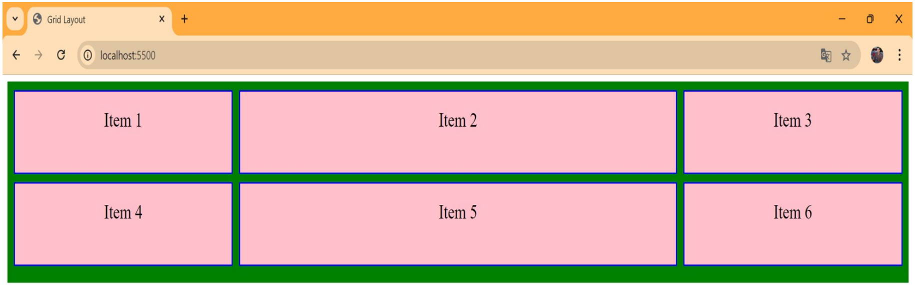
## Grid

**Example Continued.. :**

```
            background-color: pink;
            border: 2px solid blue;
            text-align: center;
            padding: 20px;
            font-size: 1.5em;
        }
    </style>
    <title>CSS Grid Layout Example</title>
</head>
<body>
    <div class="grid-container">
        <div class="grid-item">Item 1</div>
        <div class="grid-item">Item 2</div>
        <div class="grid-item">Item 3</div>
        <div class="grid-item">Item 4</div>
        <div class="grid-item">Item 5</div>
        <div class="grid-item">Item 6</div>
    </div></body></html>
```

## Grid

**Output :**


Grid Layout
localhost:5500
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6

## Flex Box

- FlexBox is a two-dimensional layout system with either row or column
- It provides a more flexible way to design responsive layouts compared to traditional CSS layout methods.

`The majorly used Flex Box Parent properties are:`

| **Property** | **Description** |
|---|---|
| `display:flex` | Flex Container becomes Flexible by setting the property display to flex |
| `flex-direction` | It defines in which direction the container wants to display the flex items. It has values as row,column,column-reverse and row-reverse |
| `flex-wrap` | It specifies whether the flex items should wrap or not. It has values as wrap,nowrap and wrap-reverse. |

## Flex Box

| Property | Description |
|---|---|
| `flex-flow` | It is a shorthand property for setting both the flex-direction and flex-wrap properties |
| `justify-content` | It is the property used to align the flex items. It has values as center,flex-start,flex-end,space-around and space-between |
| `align-items` | It is the property is used to align the flex items. It has values as center,flex-start,flex-end, stretch and baseline |
| `align-content` | It Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines. It has values as center,flex-start,flex-end, stretch, space-between and space-around. |

## Flex Box

The majorly used Flex Box Child properties are:

| Property | Description |
|---|---|
| `flex` | A shorthand property for the flex-grow, flex-shrink, and the flex-basis properties. |
| `align-self` | It Specifies the alignment for a flex item (overrides the flex container's align-items property) |
| `flex-basis` | It Specifies the initial length of a flex item. |
| `flex-grow` | This Specifies how much a flex item will grow relative to the rest of the flex items inside the same container. |
| `flex-shrink` | This Specifies how much a flex item will shrink relative to the rest of the flex. |
| `order` | Specifies the order of the flex items inside the same container. |

## Flex Box

### Flex Box Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <title>Flexbox</title>
<style>
.container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    justify-content: space-between;
    align-items: center;
    align-content: space-around;
    width: 80%;
    background-color: #f0f0f0;
    padding: 20px;
```

## Flex Box

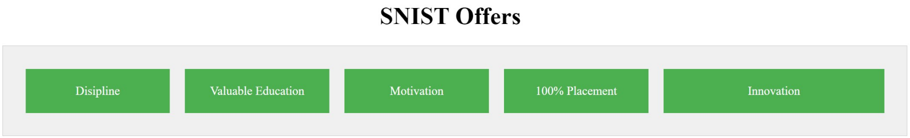
### Flex Box Example:

```
    border: 1px solid #ccc;
}
.item {
    background-color: #4CAF50;
    color: white;
    padding: 20px;
    margin: 10px;
    text-align: center;
    flex: 1 1 100px; /* flex-grow, flex-shrink, flex-basis
    */
}
.item:nth-child(3) {
    order: 1;
    flex-grow: 2;
    flex-shrink: 1;
    flex-basis: 150px;
    align-self: flex-end;
}
```

## Flex Box

**Flex Box Example:**

```
</style>
</head>
<body>
  <h1 style="margin-left: 500px;">SNIST Offers</h1>
    <div class="container">
      <div class="item">Disipline</div>
      <div class="item">Valuable Education</div>
      <div class="item">Innovation</div>
      <div class="item">Motivation</div>
      <div class="item">100% Placement</div>
    </div>
</body>
</html>
```



## Responsive Web Design

- As Web pages can be viewed using many different devices: desktops, tablets, and phones. It makes a web page look good on all devices.
- It Gives Best Experience For All Users.
- It uses only HTML and CSS.

**We often achieve responsive design by using several key techniques and tools like:**

- Media Queries
- Fluid Layouts
- Flexible Grid Layouts
- Viewport Meta Tag
- Responsive Images
- etc.,

## Responsive Web Design

### Example: Using Media Queries

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <title>Responsive Design Example</title>
   <style>
.nav {
    background-color: #4CAF50;
    padding: 1em;
}
.nav ul {
    list-style-type: none;
    padding: 0;
    margin: 0;
    display: flex;
    flex-direction: column;
```

## Responsive Web Design

```
    align-items: center;
}
.nav ul li {
    margin: 0.5em 0;
}
.nav ul li a {
    text-decoration: none;
    color: white;
    font-size: 1.2em;
}
/* Media query for tablets and larger screens */
@media (min-width: 600px) {
  .nav {
    background-color:gray;
    padding: 1em;
}
```

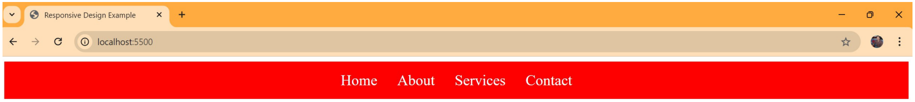
## Responsive Web Design

```
    .nav ul {
        flex-direction: row;
        justify-content: center;
    }
    .nav ul li {
        margin: 0 1em;
    }
}
/* Media query for desktops and larger screens */
@media (min-width: 900px) {
   .nav {
    background-color:red;
    padding: 1em;
}
    .nav ul li a {
        font-size: 1.5em;
    }
}
   </style>
</head>
```

## Responsive Web Design

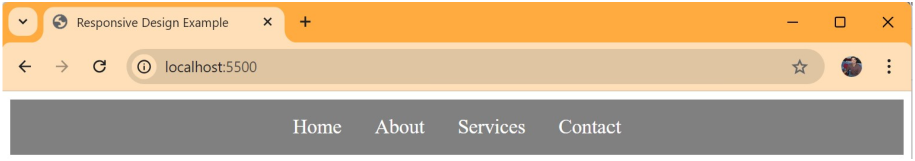
```
<body>
  <nav class="nav">
      <ul>
          <li><a href="#">Home</a></li>
          <li><a href="#">About</a></li>
          <li><a href="#">Services</a></li>
          <li><a href="#">Contact</a></li>
      </ul>
  </nav>
</body>
</html>
```

**Desktop Output:**

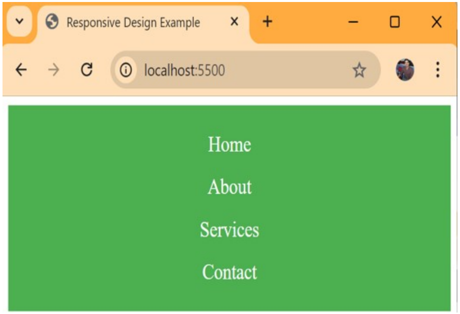


## Responsive Web Design

**Tablet Output:**



**Mobile Output:**



## Responsive Web Design

### Example: Using Responsive Images

```
   <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Responsive Images Example</title>
    <style>
        img {
            max-width: 100%;
            height: auto;
        }
        .container {
            width: 80%;
            margin: 0 auto;
        }
    </style>
</head>
```

## Responsive Web Design

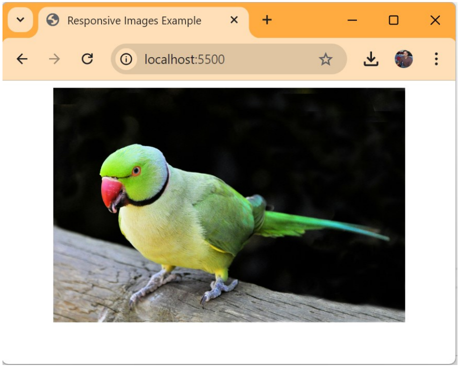
**Example: Using Responsive Images**

```
<body>
    <div class="container">
        <img src="parrot-small.jpg"
             srcset="parrot-small.jpg 480w,
             parrot-medium.jpg 800w,
             parrot-large.jpg 1200w"
             sizes="(max-width: 600px) 100vw,
                    (max-width: 1200px) 50vw,
                    33vw"
             alt="A responsive example image">
    </div>
</body>
</html>
```

## Responsive Web Design

We need to have different resolution images like parrot-small.jpg,parrot-medium.jpg and parrot-large.jpeg in the project repository as we used above

**Mobile Output:**



## View Port

- The viewport is the user's visible area of a web page.
- The viewport does not have the same size.
- HTML5 introduced a method to let web designers take control over the viewport, through the ¡meta¿ tag.

**Syntax:**

```
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
```

```
Here,
```

- width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

## Transition

- CSS transitions enable web developers to control the smooth transition between two states of an element.
- It enhances user experience and interactivity by making changes visually appealing.

**CSS Transition Properties:**

| **Property** | **Description** |
|---|---|
| `transition-property` | The transition-property specifies the CSS property (or properties) to which the transition will apply |
| `transition-duration` | The transition-duration defines how long the transition takes to complete. You can specify the duration in seconds (s) or milliseconds (ms). |
| `transition-timing-function` | It defines the speed curve of the transition effect. It controls how the intermediate states of the transition are calculated. |

## Transition

| Property | Description |
|---|---|
| `transition-delay` | The transition-delay specifies the amount of time to wait before starting the transition. You can specify the delay in seconds (s) or milliseconds (ms) |

**transition-timing-function**

| Value | Description |
|---|---|
| `ease` | Starts slow, speeds up, then slows down. |
| `linear` | Constant speed from start to finish. |
| `ease-in` | Starts slow, then speeds up. |
| `ease-out` | Starts fast, then slows down. |
| `ease-in-out` | Starts slow, speeds up, then slows down. |
| `cubic-bezier(n,n,n,n)` | Custom timing function defined by a cubic Bézier curve. |

## Transition

**Example:**

```
    <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Transition Example</title>
</head>
<style type="text/css">
.box {
  width: 100px;
  height: 100px;
  background-color: blue;
  transition-property: background-color; /* Only apply
   transition to the background-color */
  transition-duration: 5s;
  transition-timing-function: ease;
```

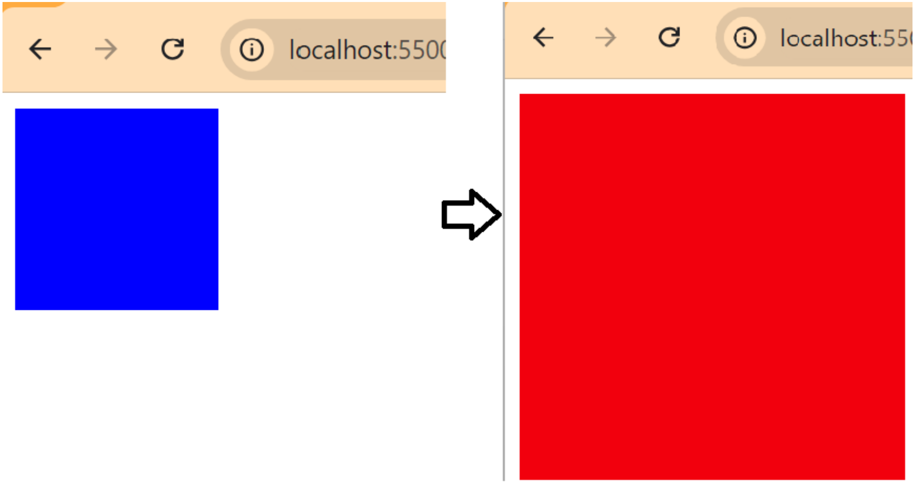
## Transition

**Example: (Continued..)**

```
  transition-timing-function: ease;/*Starts slow, speeds up,
     then slows down.*/
  transition-delay:5s;
}
.box:hover {
  background-color: red; /* Only this property will
    transition smoothly */
  width: 200px; /* This will change instantly without
    transition */
  height: 200px;
}
</style>
<body>
    <div class="box">

    </div>
</body>
</html>
```

## Transition

**Output:**



## Animation

- CSS Animations change the appearance and behavior of various elements in web pages.
- They are defined using the @keyframes rule.
- @keyframes specifies animation properties of the element and the specific time intervals at which those properties should change.

**CSS Animation Properties:**

| **Property** | **Description** |
|---|---|
| `@keyframes` | The @keyframes rule in CSS is used to specify the animation rule |
| `animation-name` | It is used to specify the name of the @keyframes describing the animation. |
| `animation-duration` | It is used to specify the time duration it takes animation to complete one cy-cle. |

## Animation

| Property | Description |
|---|---|
| `animation-delay` | It specifies the delay of the start of an animation. |
| `animation-iteration-count` | This specifies the number of times the animation will be repeated. |
| `animation-direction` | It defines the direction of the animation. animation direction can be normal, reverse, alternate, and alternate-reverse. |
| `animation-fill-mode` | It defines how styles are applied before and after animation. The animation fill mode can be none, forwards, backwards, or both. |

## Animation

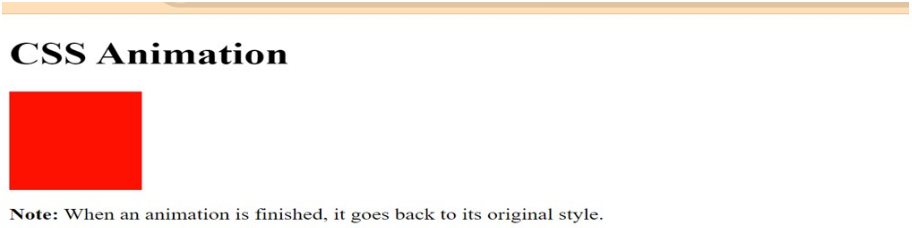
**Example:**

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
width: 100px;
height: 100px;
background-color: red;
animation-name: example;
animation-duration: 4s;
animation-iteration-count:infinite;//we specify like 2 or 3
}
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}
</style>
</head>
```

## Animation

```
<body>
<h1>CSS Animation</h1>
<div></div>
<p><b>Note:</b> When an animation is finished, it goes back
    to its original style.</p>
</body>
</html>
```

**Output:**



CSS Animation

**Note:** When an animation is finished, it goes back to its original style.

**Note:**Animation Starts with the default color red and animate from yellow to red and it iterates infinitely.

## Questionnaire

**Short Answer Questions**

1. What is the main purpose of HTML5 in web development?
2. How does the Web Storage API differ from traditional cookies?
3. What is the CSS box model, and what are its key components?
4. What is the role of media queries in responsive web design?
5. Explain the difference between CSS transitions and CSS animations.

## Questionnaire

**Long Answer Questions**

1. Explain the importance of semantic elements in HTML5. How do elements like `<header>`, `<footer>`, `<article>`, `<section>`, and `<nav>` improve the structure and accessibility of a webpage? Provide examples to illustrate their use.
2. Examine the role of HTTP status codes in web communication. How do different status codes (e.g., 200, 301, 404, 500) inform both the client and server of the state of a request? Discuss the importance of correctly handling these codes in web applications.
3. Explain the CSS box model in detail. How do properties such as margin, padding, border, and content interact within this model? Discuss how understanding the box model is crucial for solving layout issues and achieving precise control over element spacing.
4. How can CSS animations be used to create complex motion effects on a webpage? Discuss key properties such as @keyframes, animation-duration, animation-timing-function, and animation-iteration-count. Provide examples of how these properties can be combined to create intricate animations.

**Thank you**