```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

The `<!DOCTYPE html>` declaration defines that this document is an

HTML5 document

The `<html>` element is the root element of an HTML page

The `<head>` element contains meta information about the HTML page

The <head> tag in HTML is used to define the head section of a web page, which contains meta-information about the document and other elements that are not displayed on the page itself. The content within the <head> tag is not directly visible to the user but is essential for various purposes, including search engine optimization (SEO), browser functionality, and linking external resources. Here are some common uses of the <head> tag:

Metadata:

The <head> section typically contains metadata about the document, such as the document's title, character encoding, and viewport settings.

Example: <title>My Website</title>, <meta charset="UTF-8">, <meta name="viewport" content="width=device-width, initial-scale=1.0">.

Linking External Resources:

The <head> section is used to link external resources such as CSS stylesheets, JavaScript files, web fonts, and favicon images.

Example: <link rel="stylesheet" href="styles.css">, <script src="script.js"></script>, <link rel="icon" href="favicon.ico">.

SEO Optimization:

Search engines use certain elements within the `<head>` section to understand and index the content of a web page effectively.

Meta tags like `<meta name="description" content="Description of the page">` and `<meta name="keywords" content="keyword1, keyword2">` help search engines understand the page's content.

Structured data markup, such as JSON-LD or microdata, can be included to provide additional context to search engines

Viewport Configuration:

The `<meta name="viewport">` tag within the `<head>` section is used to control the layout and scaling of a web page on different devices, especially mobile devices.

It allows developers to specify the width, initial scale, and behaviour of the viewport to ensure proper display and usability across various screen sizes.

The `<title>` element specifies a title for the HTML page (which is

shown in the browser's title bar or in the page's tab)

The `<body>` element defines the document's body, and is a container.

for all the visible contents, such as headings, paragraphs, images,

hyperlinks, tables, lists, etc.

The `<h1>` element defines a large heading

The `<p>` element defines a paragraph

Some HTML elements have no content (like the
 element).

These elements are called empty elements. Empty elements do

not have an end tag!

what is the better p.lace to to use link to js , head tag or body tag:-

Placing <script> tags in the <body> tag (preferred):

Advantages:

Faster rendering: Placing <script> tags at the end of the <body> tag allows the browser to render the HTML content before loading and executing JavaScript files. This can lead to faster initial page load times and improved perceived performance.

Progressive rendering: Users can start interacting with the page's content sooner, even if JavaScript files are still loading or executing in the background.

Improved script execution: Deferred loading ensures that scripts do not block the parsing and rendering of HTML content, preventing potential delays in page rendering.

Placing `<script>` tags in the `<head>` tag:

- **Advantages**:

<[redacted]te>

- Control over script execution order: Placing `<script>` tags in the `<head>` tag allows developers to explicitly control the order in which scripts are executed, ensuring dependencies are loaded and executed in the correct sequence.
- Access to document content: Scripts in the `<head>` tag can manipulate the DOM (Document Object Model) before the page content is rendered, enabling early initialization or modification of page elements.

Disadvantages:

- Slower page load times: Scripts in the `<head>` tag can delay the rendering of HTML content since browsers must wait for scripts to download and execute before rendering the page.
- Blocking behavior: Scripts in the `<head>` tag can block the parsing and rendering of HTML content, leading to longer perceived load times and poorer user experience, especially on slower connections.

HTML Block and Inline Elements

Block-level Elements

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Two commonly used block elements are: `<p>` and `<div>`.

Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

This is a element inside a paragraph.

Inline elements in HTML can be broadly categorized into two main types based on their purpose and usage:

Text-Level Inline Elements

Text-level inline elements are primarily used to manipulate or style individual portions of text within a larger block of text.

Examples of text-level inline elements include:

- `<span>`: A generic inline container used for styling or scripting specific sections of text.
- `<a>`: Defines hyperlinks, allowing users to navigate to other web pages or resources.
- `<strong>`: Indicates strong importance or emphasis, often displayed as bold text.
- `<em>`: Emphasizes text, typically displayed in italic or emphasized style.
- `<b>`: Renders text in bold style, but semantically less meaningful than `<strong>`.
- `<i>`: Renders text in italic style, but semantically less meaningful than `<em>`.
- `<u>`: Renders text with an underline, though it's often discouraged for stylistic reasons.
- `<mark>`: Highlights text with a different background color, typically used for indicating search results or highlighted sections.
- `<code>`: Represents a fragment of computer code, typically rendered in monospace font.
- `<cite>`: Indicates the title of a work (e.g., a book, article, or film), often rendered in italic style.

2. **Inline Replaced Elements**

Inline replaced elements are inline-level elements that replace the content of the element with some external resource, typically an image, video, or other multimedia content.

Examples of inline replaced elements include:

- `<img>`: Inserts an image into the document.
- `<input>`: Defines an input control element, such as a text field, checkbox, or radio button.
- `<audio>`: Embeds an audio clip into the document.
- `<video>`: Embeds a video clip into the document.
- `<iframe>`: Embeds another HTML document into the current document, often used for embedding content from external websites or web applications.

what is box model

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

The box model is a fundamental concept in CSS (Cascading Style Sheets) that describes how elements are rendered on a web page by defining the layout and sizing of each element. It consists of four main components:

Content:

- The content area of an element is where the actual content, such as text, images, or other media, is displayed. It is surrounded by padding, borders, and margins.

Padding:

- Padding is the space between the content area and the element's border. It provides additional spacing around the content, helping to separate it from the element's border.

Border:

- The border is a line that surrounds the padding and content area of an element. It defines the boundary of the element and can have a specified width, style, and color.

Margin:

- Margin is the space outside the element's border, creating space between the element and adjacent elements. It controls the spacing between elements in a layout.

rectangular shaped elements:-

Rectangular-shaped elements are a fundamental aspect of web design, forming the building blocks of layouts and interfaces on web pages. These elements are defined by the Box Model in CSS, which describes their structure and properties. Here are some common rectangular-shaped elements used in web design:

Divs (`<div>`):

- `<div>` elements are versatile container elements used for grouping and structuring content on a web page.
- They are commonly used to create sections, columns, or blocks within a layout.
- `<div>` elements have no inherent styling or semantics and are typically styled using CSS to achieve desired layouts and designs.

position ppty:-

The position property in CSS is used to specify the positioning method of an element on a web page. There are several values for the position property, including static, relative, absolute, fixed, and sticky.

Static is the default position all html elements have initially.

relative acts like static, but we can move left right top and bottom position of element.

it’s removed from document flow.

Absolute completely removes the element from document flow, it acts like element doesn’t exist at all,

fixed, fixed position element is they stay in same place when we scroll.

sticky it is combination of relative and fixed position, it acts static when scrolls it becomes fixed position.

what is the difference between display and visibility CSS: -

The display property determines how an element is rendered in the document flow, affecting its layout and visibility.

It specifies the type of box used for an HTML element, influencing its behaviour in terms of rendering, spacing, and interaction.

Common values for the display property include none, block, inline, inline-block, flex, grid, and others.

- The `visibility` property controls whether an element is visible or hidden without affecting its layout or positioning in the document flow.
- It accepts two main values: `visible` (default) and `hidden`.
- When set to `hidden`, the element remains in the document flow and takes up space as if it were visible, but it is not displayed on the screen. The element's space is preserved, and it still affects the layout of surrounding elements.
- Unlike the `display` property, changing the `visibility` property does not alter the layout or positioning of other elements on the page.

. hidden {

visibility: hidden;

}

`display`: Affects the layout and rendering of elements, with values like `none` completely removing the element from the document flow.

- `visibility`: Controls whether an element is visible or hidden without affecting its layout, with values like `hidden` hiding the element while preserving its space in the layout.

Horizontal Alignment with `justify-content:-`

The `justify-content` property aligns flex items along the main axis (horizontally by default, in a row-based layout). Here are some common values:

- `flex-start`: Aligns items to the start of the container.
- `flex-end`: Aligns items to the end of the container.
- `center`: Centers items in the container.
- `space-between`: Distributes items evenly, with the first item at the start and the last item at the end.
- `space-around`: Distributes items evenly with equal space around them.
- `space-evenly`: Distributes items evenly with equal space between them.

Vertical Alignment with `align-items`

The `align-items` property aligns flex items along the cross axis (vertically by default, in a row-based layout). Here are some common values:

- `stretch`: Stretches items to fill the container (default).
- `flex-start`: Aligns items to the start of the container.
- `flex-end`: Aligns items to the end of the container.
- `center`: Centers items in the container.
- `baseline`: Aligns items such that their baselines align.

difference between CSS and scss:-

CSS (Cascading Style Sheets) and SCSS (Sassy CSS) are both stylesheet languages used to style HTML documents, but they have differences in terms of syntax, features, and workflow.

Syntax:

CSS: CSS uses a simple syntax with selectors, properties, and values. Styles are written in plain text, and each rule is terminated by a semicolon (;). CSS does not support variables, nesting, or mixins.

SCSS: SCSS is an extension of CSS with a more advanced syntax. It uses curly braces ({}) to define blocks of styles and semicolons (;) to terminate property-value pairs, similar to CSS. SCSS supports features like variables, nesting, mixins, inheritance, and more, making it more powerful and flexible than plain CSS.

```
/* CSS Example */
.container {
    width: 100%;
    margin: 0 auto;
}

.container p {
    color: #333;
    font-size: 16px;
}
/* SCSS Example */
```

```
$primary-color: #333;
$font-size: 16px;

.container {
    width: 100%;
    margin: 0 auto;

    p {
        color: $primary-color;
        font-size: $font-size;
    }
}
```

Nesting:

- **CSS**: CSS does not support nesting of selectors. Selectors are written independently, leading to repetitive and verbose code, especially when styling nested elements.
- **SCSS**: SCSS allows selectors to be nested within one another, mirroring the structure of HTML elements. Nesting makes it easier to target and style nested elements, reducing the need for repetitive and redundant selectors.

```
/* CSS Example */
.container {
    width: 100%;
    margin: 0 auto;
}

.container p {
    color: #333;
    font-size: 16px;
}
/* SCSS Example */
.container {
    width: 100%;
```

```
    margin: 0 auto;

    p {
        color: #333;
        font-size: 16px;
    }
}
```

Variables:

- **CSS**: CSS does not natively support variables. Styles are written with hard-coded values, which can lead to repetition and inconsistency in large projects.
- **SCSS**: SCSS introduces the concept of variables, allowing developers to define reusable values that can be used throughout the stylesheet. Variables help maintain consistency, improve readability, and make it easier to update styles across multiple elements.

```
/* CSS Example */
.button {
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
}
/* SCSS Example */
$primary-color: #007bff;

.button {
    background-color: $primary-color;
    color: #fff;
    padding: 10px 20px;
}
```

Mixins and Functions:

- **CSS**: CSS does not have built-in support for mixins or functions. Reusable styles must be manually duplicated or abstracted using classes.
- **SCSS**: SCSS introduces mixins and functions, which are reusable blocks of styles that can be included in multiple rules. Mixins allow developers to encapsulate and reuse common styles, improving maintainability and reducing code duplication.

```
/* CSS Example */
.button-primary {
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
}

.button-secondary {
    background-color: #6c757d;
    color: #fff;
    padding: 10px 20px;
}

/* SCSS Example */
@mixin button-styles($bg-color, $text-color) {
    background-color: $bg-color;
    color: $text-color;
    padding: 10px 20px;
}

.button-primary {
    @include button-styles(#007bff, #fff);
}

.button-secondary {
    @include button-styles(#6c757d, #fff);
}
```

Inheritance:

- **CSS**: CSS does not support inheritance of styles between selectors. Each rule must explicitly define all properties and values.
- **SCSS**: SCSS supports inheritance through the `@extend` directive, allowing one selector to inherit styles from another. Inheritance helps create a more modular and maintainable stylesheet by promoting code reuse and consistency.

```
/* CSS Example */
.button-base {
    border: none;
    cursor: pointer;
}

.button-primary {
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
}
/* SCSS Example */
.button-base {
    border: none;
    cursor: pointer;
}

.button-primary {
    @extend .button-base;
    background-color: #007bff;
    color: #fff;
    padding: 10px 20px;
}
```

what are the various ways to integrate css in web page:-

Inline CSS :-

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

Internal CSS :-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Internal CSS Example</title>
    <style>
        p {
            color: blue;
            font-size: 18px;
        }
    </style>
</head>
<body>
    <p>This is a paragraph with internal CSS.</p>
</body>
</html>
```

External CSS :-

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>External CSS Example</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <p>This is a paragraph with external CSS.</p>
</body>
</html>
```

Importing CSS :-

```
/* styles.css */
@import url('reset.css');
```

```
p {
   color: blue;
   font-size: 18px;
}
```

CSS Preprocessors :-

```
// styles.scss
$primary-color: blue;

p {
   color: $primary-color;
   font-size: 18px;
}
```

CSS preprocessors like Sass (SCSS) and Less provide additional features and functionality to CSS, such as variables, mixins, nesting, and more.

what is a css selector:-

A CSS selector is a pattern used to select and target HTML elements in order to apply styles to them. Selectors are a fundamental part of CSS, allowing developers to specify which elements should be styled and how they should be styled. Selectors target elements based on various criteria, such as element type, class, ID, attributes, and more.

Element Selector:

- Selects all elements of a specific type.
- Example: `p` selects all `<p>` elements.

```
p {
   color: red;
}
```

Class Selector:

Selects elements based on their class attribute.
Example: `.button` selects all elements with the class "button" (`<div class="button">`, `<button class="button">`, etc.).

```
.button {
   background-color: blue;
}
```

ID Selector:

- Selects a single element based on its ID attribute.
- Example: `#header` selects the element with the ID "header" (`<div id="header">`, `<header id="header">`, etc.).

```
#header {
  border: 1px solid green;
}
```

1. **Attribute Selector**:
   - Selects elements based on their attributes.
   - Example: `[type="submit"]` selects all elements with a "type" attribute equal to "submit" (`<input type="submit">`, `<button type="submit">`, etc.).

```
[type="submit"] {
background-color: yellow;}
```

1. **Descendant Selector**:
   - Selects elements that are descendants of another element.
   - Example: `div p` selects all `<p>` elements that are descendants of `<div>` elements.

```
div p {
  text-decoration: underline;
}
```

1. **Child Selector**:
   - Selects elements that are direct children of another element.
   - Example: `ul > li` selects all `<li>` elements that are direct children of `<ul>` elements.

```
ul > li {
  font-weight: bold;
}
```

1. **Adjacent Sibling Selector**:
   - Selects an element that is immediately preceded by another element.
   - Example: `h2 + p` selects all `<p>` elements that are immediately preceded by `<h2>` elements.

```
h2 + p {
```

```
    font-style: italic;
}
```

1. **General Sibling Selector**:
    - Selects elements that are siblings of another element.
    - Example: `h2 ~ p` selects all `<p>` elements that are siblings of `<h2>` elements.

```
h2 ~ p {
    color: gray;
}
```

File Extension:

- **CSS**: CSS files typically have a `.css` extension.
- **SCSS**: SCSS files have a `.scss` extension, distinguishing them from regular CSS files and indicating that they contain SCSS syntax.

explain universal selector:-

The universal selector, denoted by an asterisk (*), is a CSS selector that matches any element in an HTML document. It is one of the most basic and powerful selectors in CSS, allowing you to apply styles to all elements regardless of their type or attributes.

difference between class selector and id selector:-

Styles applied using ID selectors have higher precedence and can override styles applied using class selectors.

Pseudo classes:-

Pseudo-classes in CSS are keywords added to selectors that specify a special state of the selected elements. They allow you to style elements based on various criteria that cannot be targeted using standard CSS selectors alone. Pseudo-classes are preceded by a colon (:) and are added to the end of a selector.

:hover:

- Applies styles when an element is being hovered over by the mouse cursor.
- Example: `a:hover { color: red; }`

:active:

- Applies styles to an element when it is being activated by the user, typically when clicked.
- Example: `button:active { background-color: gray; }`

:focus:

- Applies styles to an element when it gains focus, such as when it is clicked or selected using the keyboard.
- Example: `input:focus { border-color: blue; }`

:first-child:

- Selects the first child element of its parent.
- Example: `ul li:first-child { font-weight: bold; }`

:last-child:

- Selects the last child element of its parent.
- Example: `ul li:last-child { color: green; }`

:nth-child():

- Selects elements based on their position within their parent element. It accepts an argument that specifies the pattern for selecting elements.
- Example: `ul li:nth-child(odd) { background-color: lightgray; }`

:nth-of-type():

- Similar to `:nth-child()`, but only matches elements of a specified type.
- Example: `p:nth-of-type(even) { color: blue; }`

:not():

- Selects elements that do not match a specified selector.
- Example: `input:not([type="text"]) { opacity: 0.5; }`

:checked:

- Applies styles to a checked checkbox or radio button.
- Example: `input[type="checkbox"]:checked + label { text-decoration: line-through; }`

:disabled:

- Applies styles to disabled form elements.
- Example: `input:disabled { background-color: lightgray; }`

:nth-last-child():

- Similar to `:nth-child()`, but counts from the end of the parent's children.
- Example: `ul li:nth-last-child(2) { font-style: italic; }`

what are pseudo elements:-

Pseudo-elements in CSS allow you to style certain parts of an element's content without needing to add extra HTML markup.

used to create effects that are not achievable with regular CSS selectors alone.

```
p::before {
    content: "Before ";
    color: red;
}
```
Inserts content before the content of the selected element.

```
p::after {
    content: " After";
    color: blue;
}
```
Inserts content after the content of the selected element.

```
p::first-letter {
    font-size: 150%;
    color: purple;
}
```
Styles the first letter of the text content within the selected element.

```
p::first-line {
    font-weight: bold;
    text-transform: uppercase;
}
```
Styles the first line of text content within the selected element.

Styles the placeholder text of input and textarea elements.

```
input::placeholder {
    color: gray;
    font-style: italic;
}
```

z-index:-

Sure, the z-index property is a powerful tool in CSS for controlling the stacking order of elements on a webpage. It is particularly useful in situations where elements overlap.

The z-index property specifies the stack order of an element, and only applies to elements that have a position value other than static (i.e., relative, absolute, fixed, or sticky). Elements with a higher z-index value will appear in front of elements with a lower z-index value.

```
<style>
  .box1 {
    position: absolute;
    width: 100px;
    height: 100px;
    background-color: red;
    top: 50px;
    left: 50px;
    z-index: 1;
  }
  .box2 {
    position: absolute;
    width: 100px;
    height: 100px;
    background-color: blue;
    top: 100px;
    left: 100px;
    z-index: 2;
  }
</style>
<div class="box1"></div>
<div class="box2"></div>
```

Elements with higher `z-index` values will be stacked on top of elements with lower values.

In this example, the blue box (.box2) will appear on top of the red box (.box1) because it has a higher z-index value (2 vs. 1).

Web Browsers

The purpose of a web browser (Chrome, Edge, Firefox, Safari) is to

read HTML documents and display them correctly.

A browser does not display the HTML tags, but uses them to

determine how to display the document:

The <!DOCTYPE> declaration represents the document type, and helps browsers to display web pages correctly.

```
<!DOCTYPE html>
```

HTML Links

```
<a href="https://www.w3schools.com">This is a link</a>
```

HTML Images

HTML images are defined with the <img> tag.

The source file (src), alternative text (alt), width, and height are provided as attributes:

```
<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">
```

Empty HTML Elements

HTML elements with no content are called empty elements.

The
 tag defines a line break, and is an empty element without a closing tag:

HTML is Not Case Sensitive

HTML tags are not case sensitive: <P> means the same as <p>.

HTML Attributes

The href Attribute

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to:

The src Attribute

The `<img>` tag is used to embed an image in an HTML page. The `src` attribute specifies the path to the image to be displayed:

There are two ways to specify the URL in the `src` attribute:

. **Absolute URL** - Links to an external image that is hosted on another website. Example: src="https://www.w3schools.com/images/img_girl.jpg".

Relative URL - Links to an image that is hosted within the website. Here, the URL does not include the domain name. If the URL begins without a slash, it will be relative to the current page. Example: src="img_girl.jpg". If the URL begins with a slash, it will be relative to the domain. Example: src="/images/img_girl.jpg".

The alt Attribute

The required `alt` attribute for the `<img>` tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to a slow connection, or an error in the `src` attribute, or if the user uses a screen reader.

The style Attribute

The `style` attribute is used to add styles to an element, such as color, font, size, and more.

```
<p style="color:red;">This is a red paragraph.</p>
```

The lang Attribute

You should always include the `lang` attribute inside the `<html>` tag, to declare the language of the Web page. This is meant to assist search engines and browsers.

```
<!DOCTYPE html>
<html lang="en">
<body>
...
</body>
</html>
```

Country codes can also be added to the language code in the `lang` attribute. So, the first two characters define the language of the HTML page, and the last two characters define the country.

```
<!DOCTYPE html>
<html lang="en-US">
<body>
...
</body>
</html>
```

HTML Paragraphs

The HTML <p> element defines a paragraph.

A paragraph always starts on a new line, and browsers automatically add

some white space (a margin) before and after a paragraph.

HTML Horizontal Rules

The <hr> tag defines a thematic break in an HTML page, and is most often displayed as a horizontal rule.

The <hr> element is used to separate content (or define a change) in an HTML page:

```
<h1>This is heading 1</h1>
<p>This is some text.</p>
<hr>
<h2>This is heading 2</h2>
<p>This is some other text.</p>
<hr>
```

HTML Line Breaks

The HTML
 element defines a line break.

Use
 if you want a line break (a new line) without starting a new

paragraph:

The
 tag is an empty tag, which means that it has no end tag.

HTML <i> and Elements

The HTML <i> element defines a part of text in an alternate voice or mood.

The content inside is typically displayed in italic.

The HTML element defines emphasized text. The content inside is typically displayed in italic.

```
<em>This text is emphasized</em>
```

HTML <small> Element

The HTML <small> element defines smaller text:

HTML <mark> Element

The HTML <mark> element defines text that should be marked or highlighted:

HTML <del> Element

The HTML <del> element defines text that has been deleted from a document. Browsers will usually strike a line through deleted text:

```
<p>My favorite color is <del>blue</del>  red.</p>
```

HTML <ins> Element

The HTML <ins> element defines a text that has been inserted into a document. Browsers will usually underline inserted text:

```
<p>My favorite color is <del>blue</del>  <ins>red</ins>.</p>
```

HTML <sub> Element

The HTML <sub> element defines subscript text. Subscript text appears half a character below the normal line, and is sometimes rendered in a smaller font. Subscript text can be used for chemical formulas, like H_2O:

HTML <sup> Element

The HTML <sup> element defines superscript text. Superscript text appears half a character above the normal line, and is sometimes rendered in a smaller font.

HTML <blockquote> for Quotations

The HTML <blockquote> element defines a section that is quoted from another source.

HTML <q> for Short Quotations

The HTML <q> tag defines a short quotation.

Browsers normally insert quotation marks around the quotation.

```
<p>WWF's goal is to: <q>Build a future where people live in harmony
```

```
with nature.</q></p>
```

HTML <abbr> for Abbreviations

The HTML `<abbr>` tag defines an abbreviation or an acronym, like "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

Marking abbreviations can give useful information to browsers, translation systems and search-engines.

HTML <address> for Contact Information

The HTML `<address>` tag defines the contact information for the author/owner of a document or an article.

The contact information can be an email address, URL, physical address, phone number, social media handle, etc.

The text in the `<address>` element usually renders in *italic,* and browsers will always add a line break before and after the `<address>` element.

```
<address>
Written by John Doe.<br>
Visit us at:<br>
Example.com<br>
Box 564, Disneyland<br>
USA
</address>
```

HTML <cite> for Work Title

The HTML `<cite>` tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).

Note: A person's name is not the title of a work.

The text in the `<cite>` element usually renders in *italic*.

```
<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>
```

HTML <bdo> for Bi-Directional Override

BDO stands for Bi-Directional Override.

The HTML `<bdo>` tag is used to override the current text direction:

```
<bdo dir="rtl">This text will be written from right to left</bdo>
```

HTML Links - The target Attribute

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

The `target` attribute specifies where to open the linked document.

The `target` attribute can have one of the following values:

- `_self` - Default. Opens the document in the same window/tab as it was clicked
- `_blank` - Opens the document in a new window or tab
- `_parent` - Opens the document in the parent frame
- `_top` - Opens the document in the full body of the window

```
<a href="https://www.w3schools.com/" target="_blank">Visi
ccccccccccccccccccccccccccc2t W3Schools!</a>
```

Absolute URLs vs. Relative URLs

Both examples above are using an **absolute URL** (a full web address) in the `href` attribute.

A local link (a link to a page within the same website) is specified with a

relative URL (without the "https://www" part):

```
<h2>Absolute URLs</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>

<h2>Relative URLs</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
```

HTML Links - Use an Image as a Link

To use an image as a link, just put the `<img>` tag inside the `<a>` tag:

```
<a href="default.asp">
<img src="smiley.gif" alt="HTML tutorial" style="width:42px;height:42px;">
</a>
```

Link Titles

The `title` attribute specifies extra information about an element. The information is most often shown as a tooltip text when the mouse moves over the element.

```
<a href="https://www.w3schools.com/html/" title="Go to W3Schools HTML section">Visit our HTML Tutorial</a>
```

HTML Link Colors

By default, a link will appear like this (in all browsers):

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

You can change the link state colors, by using CSS:

```
<style>
a:link {
  color: green;
  background-color: transparent;
  text-decoration: none;
}

a:visited {
  color: pink;
  background-color: transparent;
  text-decoration: none;
}

a:hover {
  color: red;
  background-color: transparent;
  text-decoration: underline;
}

a:active {
  color: yellow;
  background-color: transparent;
  text-decoration: underline;
}
</style>
<style>
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 15px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
}
```

```
a:hover, a:active {
  background-color: red;
}
</style>
```

HTML Links - Create Bookmarks

HTML links can be used to create bookmarks, so that readers can jump to specific parts of a web page.

Create a Bookmark in HTML

Bookmarks can be useful if a web page is very long.

To create a bookmark - first create the bookmark, then add a link to it.

When the link is clicked, the page will scroll down or up to the location with the bookmark.

```
<h2 id="C4">Chapter 4</h2>

<a href="#C4">Jump to Chapter 4</a>
```

How To Add a Favicon in HTML

```
<!DOCTYPE html>
<html>
<head>
 <title>My Page Title</title>
 <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

HTML Iframes

An HTML iframe is used to display a web page within a web page.

HTML - The Head Element

The HTML <head> Element

The <head> element is a container for metadata (data about data) and is placed between the <html> tag and the <body> tag.

HTML metadata is data about the HTML document. Metadata is not displayed.

Metadata typically define the document title, character set, styles, scripts, and other meta information.

The HTML <title> Element

The <title> element defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The content of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The HTML <link> Element

The <link> element defines the relationship between the current document and an external resource.

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

The HTML <meta> Element

The <meta> element is typically used to specify the character set, page description, keywords, author of the document, and viewport settings.

The metadata will not be displayed on the page, but is used by browsers (how to display content or reload page), by search engines (keywords), and other web services.

Define the character set used:

```
<meta charset="UTF-8">
```

Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

Define a description of your web page:

```
<meta  name="description" content="Free Web tutorials">
```

Define the author of a page:

```
<meta  name="author" content="John Doe">
```

Refresh document every 30 seconds:

```
<meta http-equiv="refresh" content="30">
```

Setting the viewport to make your website look good on all devices:

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

Setting The Viewport

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following `<meta>` element in all your web pages:

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

This gives the browser instructions on how to control the page's dimensions and scaling.

The `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).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

The HTML <script> Element

The `<script>` element is used to define client-side JavaScripts.

The HTML <base> Element

The `<base>` element specifies the base URL and/or target for all relative URLs in a page.

The `<base>` tag must have either an href or a target attribute present, or both.

There can only be one single `<base>` element in a document!

Responsive Text Size

The text size can be set with a "vw" unit, which means the "viewport width".

Never Skip the <title> Element

The `<title>` element is required in HTML.

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

HTML SVG Graphics

SVG (Scalable Vector Graphics)

SVG defines vector-based graphics in XML, which can be directly embedded in HTML pages.

SVG graphics are scalable, and do not lose any quality if they are zoomed or resized:

SVG stands for Scalable Vector Graphics

SVG is used to define vector-based graphics for the Web

SVG defines graphics in XML format

Each element and attribute in SVG files can be animated

SVG is a W3C recommendation

SVG integrates with other standards, such as CSS, DOM, XSL and JavaScript

HTML Video

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

```
<!DOCTYPE html>
<html>
<body>


<video width="400" controls>
  <source src="mov_bbb.mp4" type="video/mp4">
  <source src="mov_bbb.ogg" type="video/ogg">
  Your browser does not support HTML video.
</video>
<p>
Video courtesy of
<a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>.
</p>
</body>
</html>
```

HTML <video> Autoplay

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>
```

Add `muted` after `autoplay` to let your video start playing automatically (but muted):

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

HTML Audio

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

To play an audio file in HTML, use the `<audio>` element:

```
<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>
```

HTML Audio - How It Works

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.

HTML <audio> Autoplay

To start an audio file automatically, use the `autoplay` attribute:

Add `muted` after `autoplay` to let your audio file start playing automatically (but muted):

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

Semantic HTML

Semantic HTML means using correct HTML elements for their correct purpose as much as possible. Semantic elements are elements with a meaning; if you need a button, use the `<button>` element (and not a `<div>` element).

Examples of **non-semantic** elements: `<div>` and `<span>` - Tells nothing about its content.

Examples of **semantic** elements: `<form>`, `<table>`, and `<article>` - Clearly defines its content.

: Explain the concept of HTML5 Web Storage and its differences from cookies.

HTML5 Web Storage provides a way for web applications to store data locally in the user's browser, allowing persistent storage of larger amounts of data compared to cookies. Web Storage includes two mechanisms: localStorage and sessionStorage.

localStorage stores data with no expiration date, and the data persists even after the browser is closed and reopened.

sessionStorage stores data for the duration of the page session, meaning it's cleared when the browser tab or window is closed

Expiration: Unlike session cookies, data stored in `localStorage` has no expiration date and remains available indefinitely until explicitly removed by the web application or cleared by the user through browser settings.

Capacity: The storage capacity of `localStorage` varies between web browsers but is typically around 5-10 MB per origin. This capacity is significantly larger than traditional cookies, which are limited to about 4 KB per domain.

// Storing data in localStorage

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

// Retrieving data from localStorage

const value = localStorage.getItem('key');

// Removing data from localStorage

localStorage.removeItem('key');

// Clearing all data from localStorage

localStorage.clear();

Expiration: Data stored in `sessionStorage` is automatically cleared when the browser tab or window is closed, ensuring that it is only available for the current browsing session.

Capacity: Similar to `localStorage`, the storage capacity of `sessionStorage` also varies between web browsers but is typically around 5-10 MB per origin.

// Storing data in sessionStorage

sessionStorage.setItem('key', 'value');

// Retrieving data from sessionStorage

const value = sessionStorage.getItem('key');

// Removing data from sessionStorage

```
sessionStorage.removeItem('key');


// Clearing all data from sessionStorage
sessionStorage.clear();
```

Unlike cookies, Web Storage data is not sent to the server with every HTTP request, reducing network traffic and improving performance. Additionally, Web Storage has a larger storage capacity (usually 5-10 MB per domain) compared to cookies, which are limited to about 4 KB.

How does HTML5 differ from previous versions of HTML?

HTML5 introduced several new features and elements that enhance the functionality and semantics of web pages. Some key differences include:

New semantic elements like <header>, <footer>, <nav>, and <article>.

Support for audio, video, and canvas elements for multimedia content.

Built-in support for offline web applications using the Application Cache (AppCache) and local storage (localStorage) API.

Simplified and standardized syntax with features like the `<!DOCTYPE html>` declaration and self-closing tags.

Local Storage: This read-only interface property provides access to the document's local storage object; the stored data is stored across browser sessions. Similar to sessionStorage, except that sessionStorage data gets cleared when the page session ends—that is, when the page is closed. It is cleared when the last "private" tab of a browser is closed (localStorage data for a document loaded in a private browsing or incognito session).

Local Storage	Session Storage	Cookies
The storage capacity of local storage is 5MB/10MB	The storage capacity of session storage is 5MB	The storage capacity of Cookies is 4KB
As it is not session-based, it must be deleted via java script or manu-ally	It's session-based and works per window or tab. This means that data	Cookies expire based on the setting and working per tab and window

Local Storage	Session Storage	Cookies
The storage capacity of local storage is 5MB/10MB	is stored only for the duration of a session, i.e., until the browser (or tab) is closed The storage capacity of session storage is 5MB	The storage capacity of Cookies is 4KB
The client can read and write local storage	The client can read and write local storage	Both clients and servers can read and write the cookies
There is no transfer of data to the server	There is no transfer of data to the server	Data transfer to the server is exist
Supported by all browsers, including older ones.	Supported by all browsers, including older ones	It is supported by all the browser including older browser