UST

# HTML (Hyper Text Markup Language)

# INTRODUCTION

HTML stands for Hyper Text Markup Language

It is the standard markup language for creating Web pages

It describes the structure of a Web page

It consists of a series of elements

HTML elements tell the browser how to display the content

HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

# Example

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

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

</body>
</html>
```

# Explanation of Example

- 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 <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

# HTML Element

- An HTML element is defined by a start tag, some content, and an end tag:

```
<tagname>Content goes here...</tagname>
```

- The HTML element is everything from the start tag to the end tag:

```
<h1>My First Heading</h1>

<p>My first paragraph.</p>
```

| Start tag | Element content | End tag |
|---|---|---|
| <h1> | My First Heading | </h1> |
|  | My first paragraph. |  |
| <br> | *none* | *none* |

- Note: Some HTML elements have no content (like the <br> element). These elements are called empty elements. Empty elements do not have an end tag!

# HTML Page Structure

```
<html>
  <head>
    <title>Page title</title>
  </head>
  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph.</p>
    <p>This is another paragraph.</p>
  </body>
</html>
```

# HTML Elements

- The <html> element is the root element, and it defines the whole HTML document.
- It has a start tag <html> and an end tag </html>.
- Then, inside the <html> element there is a <body> element:

```
<body>

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

</body>
```

- The <body> element defines the document's body.
- It has a start tag <body> and an end tag </body>.
- Then, inside the <body> element there are two other elements: <h1> and <p>:

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



- The <h1> element defines a heading.
- The <p> element defines a paragraph.

# HTML Attributes

- All HTML elements can have attributes
- Attributes provide additional information about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"

# The href Attribute

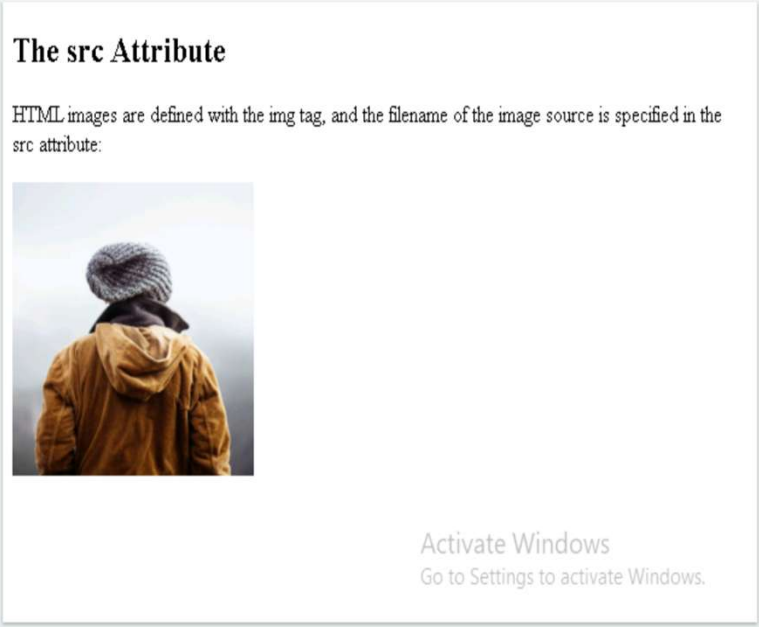
The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to

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

<h2>The href Attribute</h2>

<p>HTML links are defined with the a tag. The link address is specified in
the href attribute:</p>

<a href="https://www.w3schools.com">Visit W3Schools</a>

</body>
</html>
```

# 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:

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

<h2>The src Attribute</h2>
<p>HTML images are defined with the img tag, and the filename of the image
source is specified in the src attribute:</p>

<img src="img_girl.jpg" width="200" height="200">

</body>
</html>
```



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

1. **Absolute URL**

Links to an external image that is hosted on another website.

Example: src="https://www.w3schools.com/images/img_girl.jpg".

2. **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 width and height Attribute

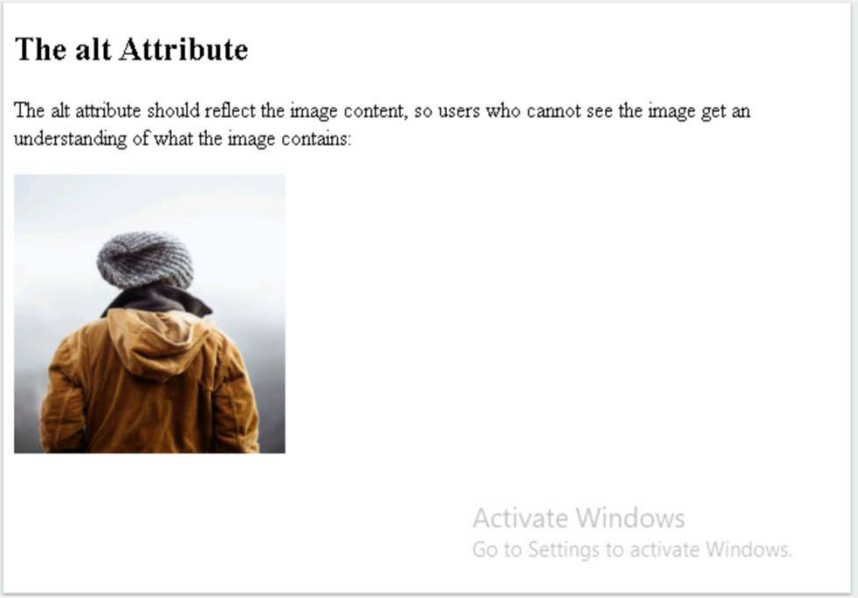
- The <img> tag should also contain the width and height attributes, which specify the width and height of the image (in pixels):

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

<h2>Width and Height Attributes</h2>

<p>The width and height attributes of the img tag, defines the width and
height of the image:</p>

<img src="img_girl.jpg" width="200" height="200">

</body>
</html>
```

# The alt Attribute

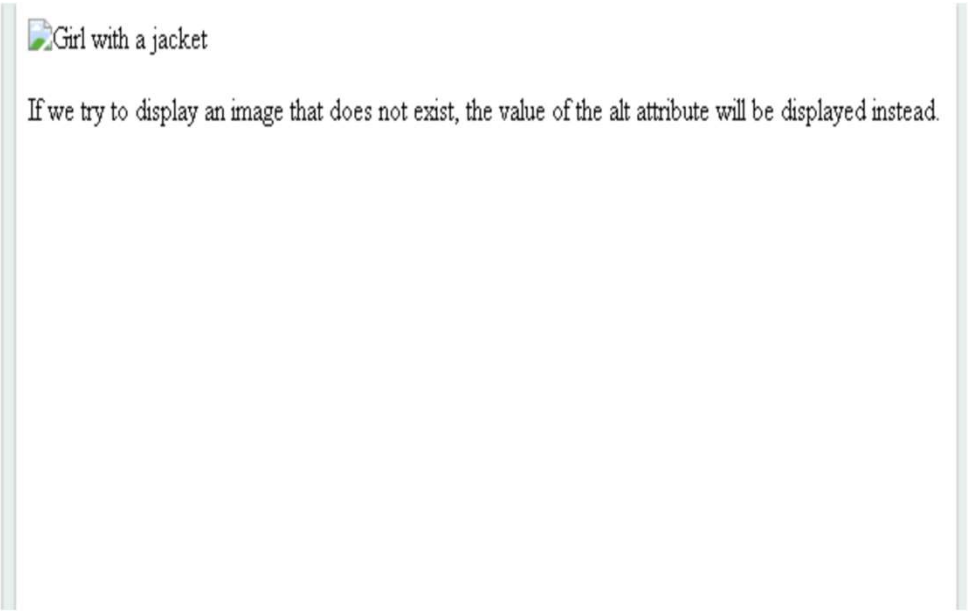
- The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed. This can be due to a slow connection, or an error in the src attribute, or if the user uses a screen reader.

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

<h2>The alt Attribute</h2>
<p>The alt attribute should reflect the image content, so users who cannot
see the image get an understanding of what the image contains:</p>

<img src="img_girl.jpg" alt="Girl with a jacket" width="200" height="200">

</body>
</html>
```



# What happens if we try to display an image that does not exist?

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

<img src="img_typo.jpg" alt="Girl with a jacket">

<p>If we try to display an image that does not exist, the value of the alt
attribute will be displayed instead. </p>

</body>
</html>
```



# The style Attribute

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

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

<h2>The style Attribute</h2>
<p>The style attribute is used to add styles to an element, such as color:
</p>

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

</body>
</html>
```

# The lang Attribute

- Should always include the lang attribute inside the <html> tag, to declare the language of the Web page.

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

# The lang Attribute

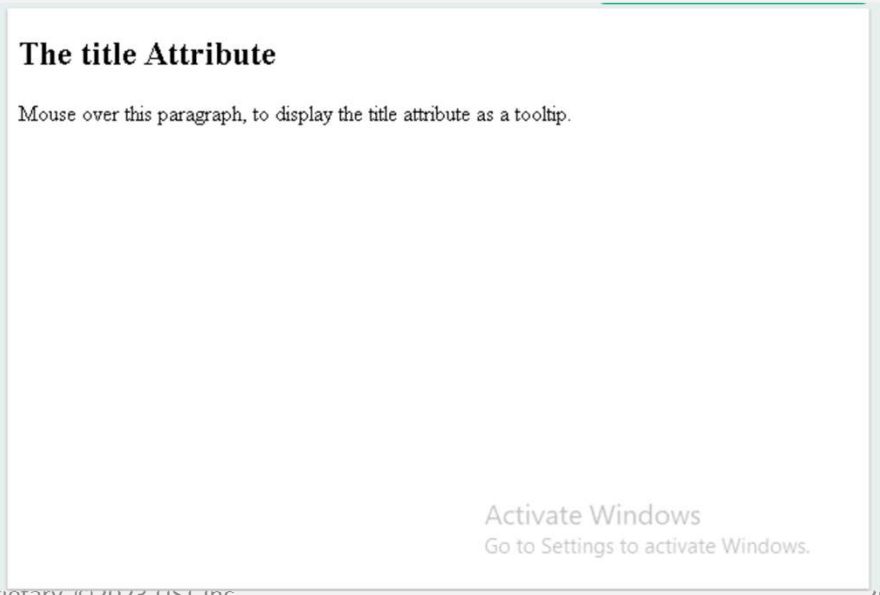
- Country codes can also be added to the language code in the lang attribute. So, the first two characters define the language of the HTML page, and the last two characters define the country.

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

# The title Attribute

- The title attribute defines some extra information about an element.
- The value of the title attribute will be displayed as a tooltip when you mouse over the element:

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

<h2 title="I'm a header">The title Attribute</h2>

<p title="I'm a tooltip">Mouse over this paragraph, to display the title
attribute as a tooltip.</p>

</body>
</html>
```



# Summary

- All HTML elements can have attributes
- The href attribute of <a> specifies the URL of the page the link goes to
- The src attribute of <img> specifies the path to the image to be displayed
- The width and height attributes of <img> provide size information for images
- The alt attribute of <img> provides an alternate text for an image
- The style attribute is used to add styles to an element, such as color, font, size, and more
- The lang attribute of the <html> tag declares the language of the Web page
- The title attribute defines some extra information about an element

# HTML Headings

- HTML headings are defined with the <h1> to <h6> tags.
- <h1> defines the most important heading. <h6> defines the least important heading.
- Example:

```
<!DOCTYPE html>
<html>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
</body>
</html>
```

# HTML Paragraphs

- The HTML <p> element defines a paragraph.
- Example:

```
<!DOCTYPE html>
<html>
<body>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

# HTML Styles

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

```
<!DOCTYPE html>
<html>
<body>
<p>I am normal</p>
<p style="color:red;">I am red</p>
<p style="color:blue;">I am blue</p>
<p style="font-size:50px;">I am big</p>
</body>
</html>
```

- **Background Color**

- Setting the style of an HTML element, can be done with the `style` attribute.
- The CSS `background-color` property defines the background color for an HTML element.
- Example:

```
<!DOCTYPE html>
<html>
<body style="background-color:powderblue;">
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

- **Text Color**

- The CSS `color` property defines the text color for an HTML element.

- Example:

```
<!DOCTYPE html>
<html>
<body>
<h1 style="color:blue;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>
</body>
</html>
```

- **Fonts**

- The CSS `font-family` property defines the font to be used for an HTML element:
- Example:

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

```
<h1 style="font-family:verdana;">This is a heading</h1>
<p style="font-family:courier;">This is a paragraph.</p>
</body>
</html>
```

- **Text Size**

- The CSS `font-size` property defines the text size for an HTML element
- Example:

```
<h1 style="font-size:300%;">This is a heading</h1>
<p style="font-size:160%;">This is a paragraph.</p>
```

- Text Alignment

- The CSS `text-align` property defines the horizontal text alignment for an HTML element:
- Example:

```
<h1 style="text-align:center;">Centered Heading</h1>
<p style="text-align:center;">Centered paragraph.</p>
```

# HTML Text Formatting

- **HTML Formatting Elements**

- Formatting elements were designed to display special types of text:
- `<b>` - Bold text
- `<strong>` - Important text
- `<i>` - Italic text
- `<em>` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `<del>` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

# HTML Quotation and Citation Elements

- **HTML <blockquote> for Quotations**

- The HTML `<blockquote>` element defines a section that is quoted from another source.
- Browsers usually indent `<blockquote>` elements.
- Example:
- ```
  Here is a quote from WWF's website:
  <blockquote cite="http://www.worldwildlife.org/who/index.html">
  For 60 years, WWF has worked to help people and nature thrive. As the world's leading
  conservation organization, WWF works in nearly 100 countries. At every level, we
  collaborate with people around the world to develop and deliver innovative solutions that
  protect communities, wildlife, and the places in which they live.
  </blockquote>
  ```
  ```

- **HTML <q> for Short Quotations**

- The HTML <q> tag defines a short quotation.
- Browsers normally insert quotation marks around the quotation.
- Example
- `<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".

- **HTML <address> for Contact Information**

- The HTML `<address>` tag defines the contact information for the author/owner of a document or an article.
- The text in the `<address>` element usually renders in *italic,* and browsers will always add a line break before and after the `<address>` element.
- Example:
- ```
  <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 `&lt;cite&gt;` tag defines the title of a creative work (e.g. a book, a poem, a song, a movie, a painting, a sculpture, etc.).
- The text in the `&lt;cite&gt;` element usually renders in *italic*.
  ```

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

# COMMENTS

- You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

- Comments can be used to hide content.
- This can be helpful if you hide content temporarily:

# Color

- HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values
- Background color:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

- Text-Color:

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

Border Color

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

- RGB Color value

- In HTML, a color can be specified as an RGB value, using this formula:
- **rgb(*red, green, blue*)**
- Each parameter (red, green, and blue) defines the intensity of the color with a value between 0 and 255.
- This means that there are 256 x 256 x 256 = 16777216 possible colors!

# • Hex Color Value

- In HTML, a color can be specified using a hexadecimal value in the form:
- ***#rrggbb***
- Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).
- For example, #ff0000 is displayed as red, because red is set to its highest value (ff), and the other two (green and blue) are set to 00.

# • HSL Color Value

- In HTML, a color can be specified using hue, saturation, and lightness (HSL) in the form:
- **hsl(*hue, saturation, lightness*)**
- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

# HTML CSS

- Cascading Style Sheets (CSS) is used to format the layout of a webpage.
- CSS can be added to HTML documents in 3 ways:
- **Inline** - by using the `style` attribute inside HTML elements
- `<h1 style="color:blue;">A Blue Heading</h1>`
- **Internal** - by using a `<style>` element in the `<head>` section

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-color: powderblue;}
h1   {color: blue;}
p    {color: red;}
</style>
</head>
<body>

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

</body>
</html>
```

- **External** - by using a <link> element to link to an external CSS file

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

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

</body>
</html>
```

"styles.css":

```css
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

# Css Color,Font,Size

- The CSS `color` property defines the text color to be used.
- The CSS `font-family` property defines the font to be used.
- The CSS `font-size` property defines the text size to be used.
-

```
<style>
h1 {
  color: blue;
  font-family: verdana;
  font-size: 300%;
}
p {
  color: red;
  font-family: courier;
  font-size: 160%;
}
</style>
```

# HTML Links

- HTML links are hyperlinks.
- You can click on a link and jump to another document.
- When you move the mouse over a link, the mouse arrow will turn into a little hand.
- `<a href="url">link text</a>`
- 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

# • HTML Color Link

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

```
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;
}
```

# HTML Images

- Images can improve the design and the appearance of a web page.
- The <img> tag has two required attributes.
- src - Specifies the path to the image
- alt - Specifies an alternate text for the image
- Syntax- <img src="url" alt="alternatetext">
- You can use the style attribute to specify the width and height of an image.
- Example : <img src="img_girl.jpg" alt="Girl in a jacket" style="width:500px;height:600px;">

# Image File Formats

- Different file formats are commonly used for web images:
- JPEG: Suitable for photographs and complex images.
- PNG: Supports transparency and ideal for graphics and logos.
- GIF: Supports animation and limited color palette.
- Specify the image dimensions using the width and height attributes.
- Use CSS to set dimensions or apply responsive design techniques.
- Example: <img src="image.jpg" alt="Image" width="500" height="300">

# Image Size and Dimensions

- Specify the image dimensions using the width and height attributes.
- Use CSS to set dimensions or apply responsive design techniques.
- Example: <img src="image.jpg" alt="Image" width="500" height="300">
- Use CSS to control image alignment and positioning within the page.
- Common CSS properties include float, margin, and display.
- Example: <img src="image.jpg" alt="Image" style="float: left; margin-right: 10px;">

# Responsive images

- Ensure images adapt to different screen sizes and devices.
- Use CSS media queries, viewport units, or frameworks like Bootstrap.
- Example: <img src="image.jpg" alt="Image" style="max-width: 100%; height: auto;">
- Compress and optimize images for faster loading.
- Tools like Adobe Photoshop, TinyPNG, or online image compressors can help.
- Consider lazy loading techniques to improve page performance.

# HTML Image Maps

- Used to create clickable areas on an image.
- An HTML image map is a way to define clickable areas on an image.
- It allows you to assign different links or actions to specific regions of an image.
- Image maps are widely used to create interactive and engaging web content.

# Types of Image Maps

- Client-side image maps are the most common type.
- They are created using HTML and CSS without the need for server-side processing.
- Server-side image maps involve using server-side scripting languages like PHP.
- Server-side image maps provide more flexibility and dynamic functionality.

# Creating an Image Map

- To create an image map in HTML, use the <map> and <area> elements.
- The <map> element defines the entire image map, and the <area> element defines each clickable area.
- Each <area> element is associated with a specific shape (rect, circle, or poly) and coordinates that define its position.

# Shape Options

- Rectangular areas (rect) are defined by specifying the top-left and bottom-right coordinates.
- Circular areas (circle) are defined by specifying the center coordinates and the radius.
- Polygonal areas (poly) are defined by providing a series of coordinates that form the shape.

# Coordinating and Mapping

- Coordinating the clickable areas with image positions is crucial for accurate mapping.
- Ensure that the image and the clickable areas have the same dimensions and aspect ratio.
- Use image editing tools or online image map generators to easily obtain the coordinates.

# Linking to Web Pages

- Use the href attribute in the <area> element to define the destination URL for each clickable area.
- You can link to other web pages, specific sections within a page, or external resources.

# Styling Image Maps

- Apply CSS styles to enhance the appearance of image maps.
- Customize the look of the clickable areas using background colors, borders, and hover effects.
- Use CSS classes or inline styles to target specific areas and apply different styles.

# How to give Image Map

- The image is inserted using the <img> tag. The only difference from other images is that you must add a usemap attribute.
- <img src="workplace.jpg" alt="Workplace" usemap="#workmap">
- The <map> element is used to create an image map, and is linked to the image by using the required name attribute
- <map name="workmap">

# HTML Image Tags

| Tag | Description |
|---|---|
| <img> | Defines an image |
| <map> | Defines an image map |
| <area> | Defines a clickable area inside an image map |
| <picture> | Defines a container for multiple image resources |

# HTML Background Images

- Definition of HTML background images
- Importance and benefits of using background images in web design
- Overview of how background images are implemented in HTML

# Background Image Property

- Explanation of the CSS background-image property
- How the property is used to set a background image for an HTML element
- Syntax example: background-image: url("image.jpg");

# Inline Background Images

- How to set an inline background image using the style attribute
- Example of inline background image syntax:
- 

# Background Image Size and Position

- Controlling the size and positioning of background images
- Overview of the background-size and background-position properties
- Examples of different background image sizing and positioning options

# Background Image Repeat

- Understanding how background images repeat by default
- Introduction to the background-repeat property
- Examples of different repeat options: repeat, repeat-x, repeat-y, and no-repeat

# Background Image Attachment

- Exploring the background-attachment property
- Differentiating between scroll and fixed attachment options
- Example of fixed attachment syntax: background-attachment: fixed;

# Background Image Shorthand

- The shorthand background property for setting multiple background properties
- Syntax example: background: url("image.jpg") no-repeat center center/cover;

# Background Image Best Practices

- Tips for selecting and optimizing background images
- Ensuring image file sizes are optimized for web performance
- Considering responsiveness and mobile devices in background image design

# Background Image Examples

- Showcase a few examples of websites or web elements with creative and effective background images
- Discuss the impact of background images on the overall design and user experience

# HTML <picture> Element

- Brief overview of the HTML <picture> element
- Explanation of its purpose and benefits
- Importance of responsive images in web design

# The Need for Responsive Images

- Discuss the challenges posed by different device resolutions and screen sizes
- Importance of delivering appropriately sized images for optimal user experience
- Introduction to the concept of responsive web design

# Introducing the <picture> Element

- Definition and purpose of the <picture> element
- Explanation of how it works in conjunction with the <source> and <img> elements
- Its role in providing different image sources for different conditions

# Syntax of the <picture> Element

- Example code demonstrating the basic structure of the <picture> element
- Usage of the <source> and <img> elements within the <picture> element

# Multiple Source Options with <source>

- Explanation of the <source> element's attributes: srcset, media, and type
- Demonstration of providing multiple image sources for different scenarios
- Examples of using different file formats (e.g., JPEG, PNG, WebP) for various browser support

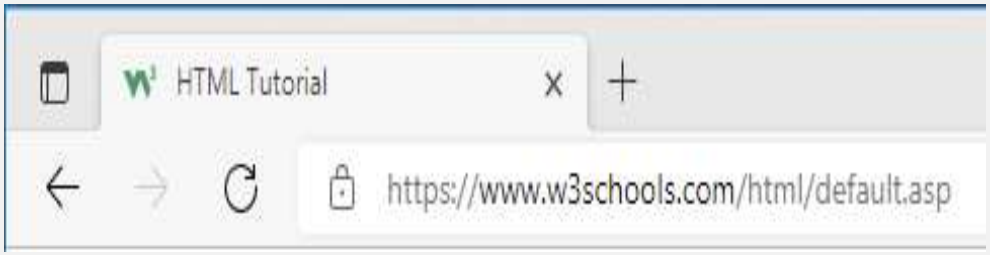
# Fallback <img> for Compatibility

- Importance of including a fallback <img> element within the <picture> element
- Explanation of the src attribute in the <img> element as a fallback source
- Example code showcasing the proper usage of the fallback <img> element

# HTML Favicon

.A favicon is a small image displayed next to the page title in the browser tab.

## How To Add a Favicon in Html




A favicon image is displayed to the left of the page title in the browser tab, like this:

To add a favicon to your website, either save your favicon image to the root directory of your webserver, or create a folder in the root directory called images, and save your favicon image in this folder. A common name for a favicon image is "favicon.ico".

Example

```
<!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 Page Title

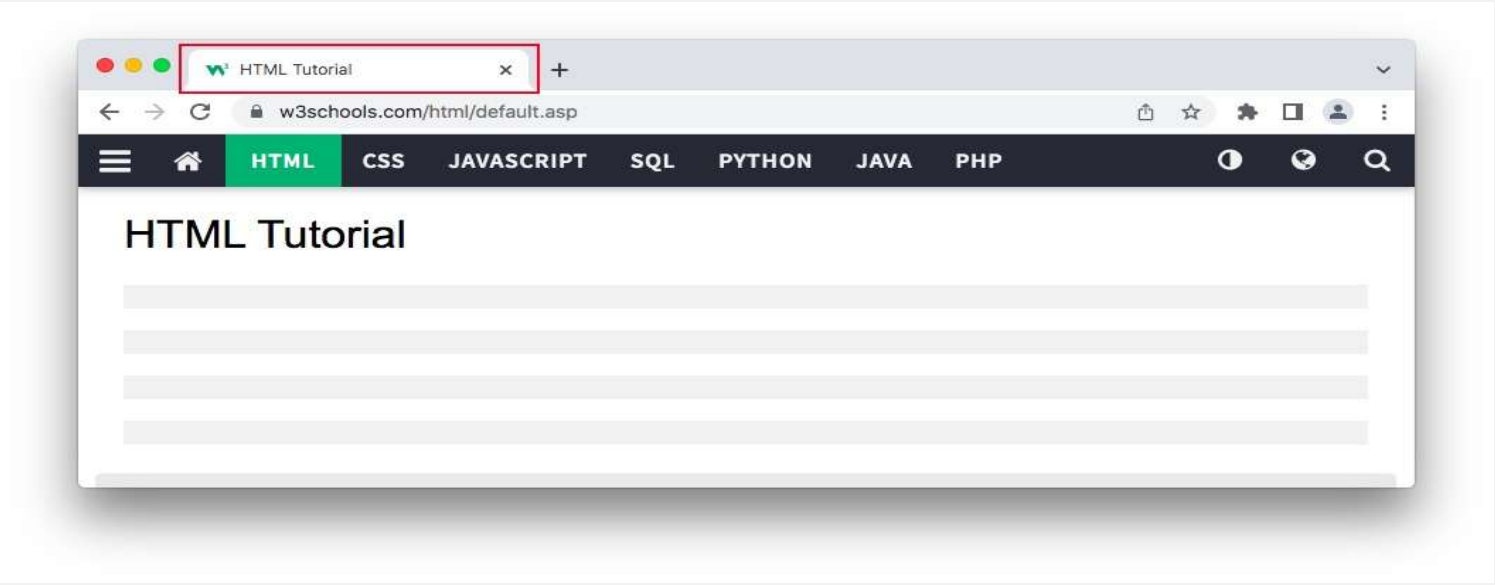
Every web page should have a page title to describe the meaning of the page.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML
Tutorial</title>
</head>
<body>

The content of the
document......

</body>
</html>
```

The title is shown in the browser's title bar:

.




The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine-results

# HTML Tables

HTML tables allow web developers to arrange data into rows and columns.

| Company | Contact | Country |
|---|---|---|
| Alfreds Futterkiste | Maria Anders | Germany |
| Centro comercial Moctezuma | Francisco Chang | Mexico |
| Ernst Handel | Roland Mendel | Austria |
| Island Trading | Helen Bennett | UK |
| Laughing Bacchus Winecellars | Yoshi Tannamuri | Canada |
| Magazzini Alimentari Riuniti | Giovanni Rovelli | Italy |

## Table Cells

Each table cell is defined by a <td> and a </td> tag.

td stands for table data.

Example

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td
>
    <td>Linus</td>
  </tr>
</table>
```

# HTML Page Title

Every web page should have a page title to describe the meaning of the page.

Example

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML
Tutorial</title>
</head>
<body>

The content of the
document......

</body>
</html>
```

# HTML Lists

HTML lists allow web developers to group a set of related items in list

Example

An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

## Unordered HTML List

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

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

Example

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

## Ordered HTML List

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

The list items will be marked with numbers by default:

Example

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

## HTML Block and Inline Elements

Every HTML element has a default display value, depending on what type of element it is
There are two display values: block and inline.

## Block-level Elements

Two commonly used block elements are: <p> and .
The <p> element defines a paragraph in an HTML document.
The  element defines a division or a section in an HTML document.

Example

```
<p>Hello World</p>
<div>Hello World</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.

Example

```
<span>Hello World</span>
```

# HTML class Attribute

- The HTML class attribute is used to specify a class for an HTML element.
- It can also be used by a JavaScript to access and manipulate elements with the specific class name.
- In the following example we have two  elements with a class attribute with the value of "note".
  Both  elements will be styled equally according to the .note style definition in the head section:

```
<!DOCTYPE html>
<html>
<head>
<style>
.note {
  font-size: 120%;
  color: red;
}
</style>
</head>
<body>

<h1>My <span class="note">Important</span> He
ading</h1>
<p>This
is some <span class="note">important</span> text
.</p>

</body>
</html>
```

# HTML id Attribute

- The HTML id attribute is used to specify a unique id for an HTML element.
- The id attribute is used to point to a specific style declaration in a style sheet
- In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

# HTML Iframes

- An HTML iframe is used to display a web page within a web page.
- The HTML <iframe> tag specifies an inline frame.
- An inline frame is used to embed another document within the current HTML document.

Syntax

- <iframe src="*url*" title="*description*"></iframe>

## Iframe - Set Height and Width

- Use the height and width attributes to specify the size of the iframe.

```
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>
```

## Iframe - Remove the Border

- By default, an iframe has a border around it.
- To remove the border, add the style attribute and use the CSS border property:

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe Example"></iframe>
```

# HTML JavaScript

- JavaScript makes HTML pages more dynamic and interactive.

The HTML <script> Tag

- The HTML <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains script statements, or it points to an external script file through the src attribute.
- To select an HTML element, JavaScript most often uses the document.getElementById() method.

```
<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>
```

## The HTML <noscript> Tag

- The HTML <noscript> tag defines an alternate content to be displayed to users that have disabled scripts in their browser or have a browser that doesn't support scripts:
- <script>
  document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
  <noscript>Sorry, your browser does not support JavaScript!</noscript>

# HTML File Paths

- A file path describes the location of a file in a web site's folder structure.
- File paths are used when linking to external files, like:

-Web pages

-Images

-Style sheets

-JavaScripts

## Absolute File Paths

- An absolute file path is the full URL to a file:

<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">

## Relative File Paths

- A relative file path points to a file relative to the current page.
- In the following example, the file path points to a file in the images folder located at the root of the current web:

```
<img src="/images/picture.jpg" alt="Mountain">
```

# 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 <title> element is required in HTML documents!

The <title> element:

- defines a title in the browser toolbar
- provides a title for the page when it is added to favorites
- displays a title for the page in search engine-results

- **The HTML <style> Element**

The <style> element is used to define style information for a single HTML page:

Example

```
<style>
  body {background-color: powderblue;}
  h1 {color: red;}
  p {color: blue;}
</style>
```

- **The HTML <link> Element**

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

The <link> tag is most often used to link to external style sheets:

Example

```
<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.

Examples

**Define the character set used:**

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

- **The HTML <script> Element**

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

The following JavaScript writes "Hello JavaScript!" into an HTML element with id="demo":

Example

```
<script>
  function myFunction() {
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  }
  </script>
```

- **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.

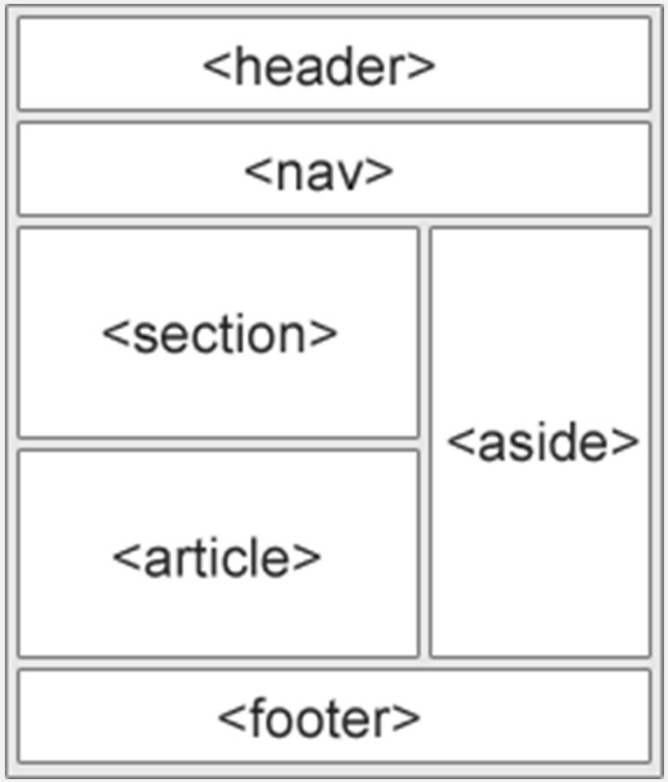
There can only be one single <base> element in a document!

Example

Specify a default URL and a default target for all links on a page:

```
<head>
<base href="https://www.w3schools.com/" target="_blank">
</head>

<body>
<img src="images/stickman.gif" width="24" height="39" alt="Stickman">
<a href="tags/tag_base.asp">HTML base Tag</a>
</body>
```

# HTML Layout Elements and Techniques




- **<header> - Defines a header for a document or a section**
- **<nav> - Defines a set of navigation links**
- **<section> - Defines a section in a document**
- **<article> - Defines an independent, self-contained content**
- **<aside> - Defines content aside from the content (like a sidebar)**
- **<footer> - Defines a footer for a document or a section**
- **<details> - Defines additional details that the user can open and close on demand**
- **<summary> - Defines a heading for the <details> element**

# HTML Layout Techniques

There are four different techniques to create multicolumn layouts. Each technique has its pros and cons:

- CSS framework
- CSS float property
- CSS flexbox
- CSS grid

- CSS Float Layout
- It is common to do entire web layouts using the CSS float property. Float is easy to learn - you just need to remember how the float and clear properties work. **Disadvantages:** Floating elements are tied to the document flow, which may harm the flexibility.
- CSS Flexbox Layout
- Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices.
- CSS Grid Layout
- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

# HTML Responsive Web Design

Responsive web design is about creating web pages that look good on all devices!

A responsive web design will automatically adjust for different screen sizes and viewports.

**What is Responsive Web Design?**

Responsive Web Design is about using HTML and CSS to automatically resize, hide, shrink, or enlarge, a website, to make it look good on all devices (desktops, tablets, and phones):

**Setting The Viewport**

To create a responsive website, add the following <meta> tag to all your web pages:

Example

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

This will set the viewport of your page, which will give the browser instructions on how to control the page's dimensions and scaling.

Here is an example of a web page *without* the viewport meta tag, and the same web page *with* the viewport meta tag:

Without the viewport meta tag:


With the viewport meta tag:

- **Responsive Images**

Responsive images are images that scale nicely to fit any browser size.

Using the width Property

If the CSS width property is set to 100%, the image will be responsive and scale up and down:

Example

```
<img src="img_girl.jpg" style="width:100%;">
```

- **Using the max-width Property**

If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

```
<img src="img_girl.jpg" style="max-width:100%;height:auto;">
```

- **Show Different Images Depending on Browser Width**

The HTML <picture> element allows you to define different images for different browser window sizes.

Resize the browser window to see how the image below changes depending on the width:

Example

```
<picture>
  <source srcset="img_smallflower.jpg" media="(max-width: 600px)">
  <source srcset="img_flowers.jpg" media="(max-width: 1500px)">
  <source srcset="flowers.jpg">
  <img src="img_smallflower.jpg" alt="Flowers">
</picture>
```

- **Responsive Text Size**

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

That way the text size will follow the size of the browser window:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Left Menu

# HTML Computer Code Elements

HTML contains several elements for defining user input and computer code.

Example:

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

- **HTML <kbd> For Keyboard Input**

The HTML <kbd> element is used to define keyboard input. The content inside is displayed in the browser's default monospace font.

Example

Define some text as keyboard input in a document:

```
<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>
```

- **HTML <samp> For Program Output**

The HTML <samp> element is used to define sample output from a computer program. The content inside is displayed in the browser's default monospace font.

Example

Define some text as sample output from a computer program in a document:

```
<p>Message from my computer:</p>
<p><samp>File not found.<br>Press F1 to continue</samp></p>
```

- **HTML <code> For Computer Code**

The HTML <code> element is used to define a piece of computer code. The content inside is displayed in the browser's default monospace font.

Example

Define some text as computer code in a document:

```
<code>
x = 5;
y = 6;
z = x + y;
</code>
```

- **HTML <var> For Variables**

The HTML <var> element is used to define a variable in programming or in a mathematical expression. The content inside is typically displayed in italic.

Example

Define some text as variables in a document:

```
<p>The area of a triangle is: 1/2 x <var>b</var> x <var>h</var>,
where <var>b</var> is the base, and <var>h</var> is the vertical height.</p>
```

# HTML Semantic Elements

What are Semantic Elements?

A semantic element clearly describes its meaning to both the browser and the developer.

Examples of **non-semantic** elements:  and  - Tells nothing about its content.

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

## Semantic Elements in HTML

Many web sites contain HTML code like:    to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

<article>

<aside>

<details>

<figcaption>

<figure>

<footer>

<header>

<main>

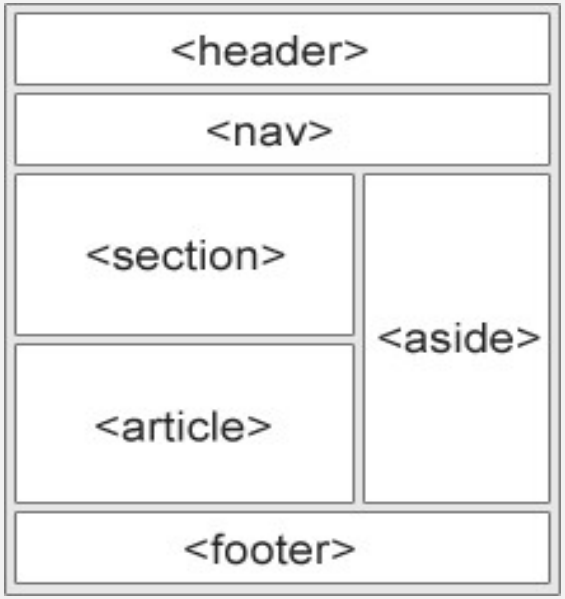
<mark>

<nav>

<section>

<summary>

<time>



- **HTML <section> Element**

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

According to W3C's HTML documentation: "A section is a thematic grouping of content, typically with a heading."

Examples of where a <section> element can be used:

- Chapters
- Introduction
- News items
- Contact information

- **HTML <article> Element**

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

An article should make sense on its own, and it should be possible to distribute it independently from the rest of the web site.

Examples of where the <article> element can be used:

- Forum posts
- Blog posts
- User comments
- Product cards
- Newspaper articles

- **Nesting <article> in <section> or Vice Versa?**

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

The <section> element defines section in a document.

Can we use the definitions to decide how to nest those elements? No, we cannot!

So, you will find HTML pages with <section> elements containing <article> elements, and <article> elements containing <section> elements.

- **HTML <header> Element**

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

A <header> element typically contains:

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

- **HTML <footer> Element**

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

A <footer> element typically contains:

- authorship information
- copyright information
- contact information
- sitemap
- back to top links
- related documents
- **HTML <nav> Element**

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

- **HTML <aside> Element**

The <aside> element defines some content aside from the content it is placed in (like a sidebar).

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

- **HTML <figure> and <figcaption> Elements**

The <figure> tag specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.

The <figcaption> tag defines a caption for a <figure> element.
The <figcaption> element can be placed as the first or as the last child of a <figure> element.

# HTML Entities

- Character entities are used to display reserved characters in HTML.

A character entity looks like this:

&*entity_name*;

OR

&#*entity_number*;

A commonly used entity in HTML is the non-breaking space: ** ** A non-breaking space is a space that will not break into a new line.It is used to prevent browsers from truncating spaces in HTML pages.

Examples:

- § 10
- 10 km/h
- 10 PM

| Character | Entity Name | Entity Number |
|---|---|---|
| Greater than sign (>) | > | > |
| Less than sign (<) | < | < |
| Ampersand (&) | & | & |
| Quotation mark (") | " | " |
| Apostrophe (') | ' | ' |
| Euro sign (€) | € | € |
| Registered trademark symbol (®) | ® | ® |
| Copyright symbol (©) | © | © |
| Bullet point symbol (•) | • | • |
| Em dash (—) | — | — |

Example:-

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Entities Example - &euro;</title>
</head>
<body>
   <h1>HTML Entities Example - &euro;</h1>

   <p>The price for this item is €19.99.</p>
</body>
</html>
```

HTML Entities Example - €
The price for this item is €19.99.

# HTML Symbols

- Many mathematical, technical, and currency symbols, are not present on a normal keyboard. To add such symbols to an HTML page, you can use the entity name or the entity number (a decimal or a hexadecimal reference) for the symbol.
- Example

Display the euro sign, €, with an entity name, a decimal, and a hexadecimal value:

```
<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>
```

Will display as:

I will display €
I will display €
I will display €

| Symbol | Entity Name | Entity Number |
|---|---|---|
| & | Ampersand | &amp; |
| < | Less than | &lt; |
| > | Greater than | &gt; |
| " | Quotation mark | &quot; |
| ' | Apostrophe | &apos; |
| | Non-breaking space |   |
| © | Copyright symbol | &copy; |
| ® | Registered trademark symbol | &reg; |
| ™ | Trademark symbol | &trade; |
| ♥ | Heart symbol | &hearts; |
| ♣ | Club symbol | &clubs; |
| ♦ | Diamond symbol | &diams; |
| ♠ | Spade symbol | &spades; |
| † | Dagger symbol | &dagger; |
| § | Section symbol | &sect; |

Example:-

```
<!DOCTYPE html>
<html>
<head>
   <title>HTML Entities Example - &copy;</title>
</head>
<body>
   <h1>HTML Entities Example - &copy;</h1>

   <p>This website is protected by &copy; 2023 MyCompany. All rights reserved.</p>
</body>
</html>
```

HTML Entities Example - ©

This website is protected by © 2023 MyCompany. All rights reserved.

# HTML Charset

- To display an HTML page correctly, a web browser must know which character set to use.

From ASCII to UTF-8

- ASCII was the first character encoding standard. ASCII defined 128 different characters that could be used on the internet: numbers (0-9), English letters (A-Z), and some special characters like ! $ + - ( ) @ < > .
- ISO-8859-1 was the default character set for HTML 4. This character set supported 256 different character codes. HTML 4 also supported UTF-8.
- ANSI (Windows-1252) was the original Windows character set. ANSI is identical to ISO-8859-1, except that ANSI has 32 extra characters.
- To display an HTML page correctly, a web browser must know the character set used in the page. This is specified in the <meta> tag:

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

# HTML URL Encode

- A URL can be composed of words  or an Internet Protocol (IP) address (e.g. 192.68.20.50).
- Web browsers request pages from web servers by using a URL.A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.
- URLs can only be sent over the Internet using the ASCII character-set. If a URL contains characters outside the ASCII set, the URL has to be converted.
- URL encoding converts non-ASCII characters into a format that can be transmitted over the Internet.
- URL encoding replaces non-ASCII characters with a "%" followed by hexadecimal digits.
- URLs cannot contain spaces. URL encoding normally replaces a space with a plus (+) sign, or %20.

Example:-

```
<!DOCTYPE html>
<html>
<head>
    <title>URL Encoding Example</title>
</head>
<body>
    <h1>URL Encoding Example</h1>

    <p>Original URL: https://example.com/?search=hello world&category=example</p>

    <p>URL Encoded: https://example.com/?search=hello%20world&amp;category=example</p>
</body>
</html>
```

# HTML vs XHTML

| HTML | XHTML |
|---|---|
| HTML is based on SGML (Standard Generalized Markup Language) | XHTML is based on XML (eXtensible Markup Language) |
| HTML allows some errors and inconsistencies | XHTML is stricter and requires well-formed XML syntax |
| Tags and attributes are case-insensitive | Tags and attributes are case-sensitive in XHTML |
| HTML has deprecated elements and attributes | XHTML removes deprecated elements and attributes |
| Inline event handlers are commonly used in HTML | Inline event handlers are not recommended in XHTML |
| HTML supports both HTML-style and XML-style comments | XHTML only supports XML-style comments |
| Empty elements do not require closing tags in HTML | Empty elements must be properly closed with a self-closing tag in XHTML |
| HTML does not require a DOCTYPE declaration | XHTML requires a DOCTYPE declaration at the beginning of the document |

# HTML Graphics

## HTML Canvas Graphics

The HTML <canvas> element is used to draw graphics on a web page.

The graphic to the left is created with <canvas>. It shows four elements: a red rectangle, a gradient rectangle, a multicolor rectangle, and a multicolor text.

# What is HTML Canvas?

- The HTML <canvas> element is used to draw graphics, on the fly, via JavaScript.
- The <canvas> element is only a container for graphics. You must use JavaScript to actually draw the graphics.
- Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

# Browser Support

- The numbers in the table specify the first browser version that fully supports the <canvas> element.

# Canvas Examples

- A canvas is a rectangular area on an HTML page. By default, a canvas has no border and no content.
- The markup looks like this:
- <canvas id="myCanvas" width="200" height="100"></canvas>
- **Note:** Always specify an id attribute (to be referred to in a script), and a width and height attribute to define the size of the canvas. To add a border, use the style attribute.

Here is an example of a basic, empty canvas:

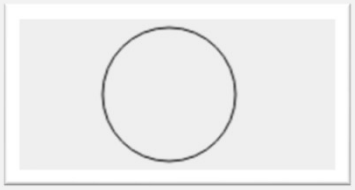
```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
Your browser does not support the HTML canvas tag.
</canvas>
</body>
</html>
```

# Add a JavaScript

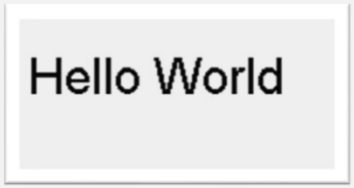
- After creating the rectangular canvas area, you must add a JavaScript to do the drawing.
- Here are some examples:

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>

</body>
</html>
```

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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```



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

<canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
Your browser does not support the HTML canvas tag.</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello World",10,50);
</script>

</body>
</html>
```



- <!DOCTYPE html>
- <html>
- <body>

- <canvas id="myCanvas" width="200" height="100" style="border:1px solid #d3d3d3;">
- Your browser does not support the HTML canvas tag.</canvas>

- <script>
- var c = document.getElementById("myCanvas");
- var ctx = c.getContext("2d");
- // Create gradient
- var grd = ctx.createLinearGradient(0,0,200,0);
- grd.addColorStop(0,"red");
- grd.addColorStop(1,"white");

```
// Fill with gradient
ctx.fillStyle = grd;
ctx.fillRect(10,10,150,80);
</script>

</body>
</html>
```

# HTML SVG Graphics

## What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web

## The HTML <svg> Element

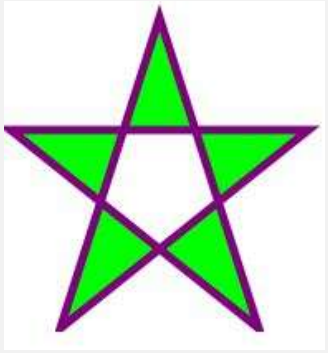
- The HTML <svg> element is a container for SVG graphics.
- SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

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

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40"
  stroke="green" stroke-width="4" fill="yellow" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

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

<svg width="400" height="100">
  <rect width="400" height="100"
  style="fill:rgb(0,0,255);stroke-width:10;stroke:rgb(0,0,0)" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

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

<svg width="400" height="180">
  <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
  style="fill:red;stroke:black;stroke-width:5;opacity:0.5" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

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

<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78 160,198"
  style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```



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

<svg height="130" width="500">
  <defs>
    <linearGradient id="grad1" x1="0%" y1="0%" x2="100%" y2="0%">
      <stop offset="0%"
      style="stop-color:rgb(255,255,0);stop-opacity:1" />
      <stop offset="100%"
      style="stop-color:rgb(255,0,0);stop-opacity:1" />
```

```
  </linearGradient>
  </defs>
  <ellipse cx="100" cy="70" rx="85" ry="55" fill="url(#grad1)" />
  <text fill="#ffffff" font-size="45" font-family="Verdana"
  x="50" y="86">SVG</text>
Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>
```

# HTML Geolocation API

- The getCurrentPosition() method is used to return the user's position.
- The second parameter of the getCurrentPosition() method is used to handle errors.
- The getCurrentPosition() method returns an object on success.
- watchPosition() - Returns the current position of the user and continues to return updated position as the user moves (like the GPS in a car).
- clearWatch() - Stops the watchPosition() method.

```
<!DOCTYPE html>
<html>
<body>
<p>Click the button to get your coordinates.</p>
<button onclick="getLocation()">Try It</button>
<p id="demo"></p>
<script>
var x = document.getElementById("demo");
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}
```

```
function showPosition(position) {
 x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}
</script>
</body>
</html>
```

# HTML Drag and Drop API

- Drag and drop is a very common feature. It is when you "grab" an object and drag it to a different location.
- To make an element draggable, set the draggable attribute to true:

```
<img draggable="true">
```

- The ondragstart attribute calls a function, drag(event), that specifies what data to be dragged.
- The ondragover event specifies where the dragged data can be dropped.

```
<!DOCTYPE HTML>
<html>
<head>
<script>
function allowDrop(ev) {
  ev.preventDefault();
}
function drag(ev) {
  ev.dataTransfer.setData("text", ev.target.id);
}
function drop(ev) {
  ev.preventDefault();
  var data = ev.dataTransfer.getData("text");
  ev.target.appendChild(document.getElementById(data));
}
</script>
</head>
```

```
<body>

<div id="div1" ondrop="drop(event)" ondragover="allowDrop(eve
nt)"></div>

<img id="drag1" src="img_logo.gif" draggable="true" ondragsta
rt="drag(event)" width="336" height="69">

</body>
</html>
```

# HTML Web Storage API

- With web storage, web applications can store data locally within the user's browser.
- HTML web storage provides two objects for storing data on the client:

  window.localStorage - stores data with no expiration date

  window.sessionStorage - stores data for one session (data is lost when the browser tab is closed)
- if (typeof(Storage) !== "undefined") {
  *}* else {
  }

# Web Worker

A web worker is a JavaScript that runs in the background, independently of other scripts, without affecting the performance of the page. You can continue to do whatever you want: clicking, selecting things, etc., while the web worker runs in the background.

## Check Web Worker Support

```
if (typeof(Worker) !== "undefined") {
    // Yes! Web worker support!
        // Some code.....
    } else {
    // Sorry! No Web Worker support..
}
```

## Create a Web Worker File

- Now, let's create our web worker in an external JavaScript.
- Here, we create a script that counts. The script is stored in the "demo_workers.js" file:

```
var i = 0;
function timedCount() {
    i = i + 1;
    postMessage(i);
    setTimeout("timedCount()",500);
}
timedCount();
```

## Create a Web Worker Object

- The following lines checks if the worker already exists, if not - it creates a new web worker object and runs the code in "demo_workers.js":

```
if (typeof(w) == "undefined") {
    w = new Worker("demo_workers.js");
}
```

## Terminate a Web Worker

- When a web worker object is created, it will continue to listen for messages (even after the external script is finished) until it is terminated.
- To terminate a web worker, and free browser/computer resources, use the terminate() method:

```
w.terminate();
```

# Server-Sent Events - One Way Messaging

A server-sent event is when a web page automatically gets updates from a server.

This was also possible before, but the web page would have to ask if any updates were available. With server-sent events, the updates come automatically.

Examples: Facebook/Twitter updates, stock price updates

## Receive Server-Sent Event Notifications

Example

```
var source = new EventSource("demo_sse.php");
source.onmessage = function(event) {
document.getElementById("result").innerHTML += event.data + "<br>";
};
```

# Check Server-Sent Events Support

```
if(typeof(EventSource) !== "undefined") {
    // Yes! Server-sent events support!
    // Some code.....
     } else {
    // Sorry! No server-sent events support..
    }
```

## Server-Side Code Example

```
<?php
header('Content-Type: text/event-stream');
header('Cache-Control: no-cache');

$time = date('r');
echo "data: The server time is: {$time}\n\n";
flush();
?>
```

# HTML MEDIA

**Multimedia:**

- Multimedia comes in many different formats. It can be almost anything you can hear or see, like images, music, sound, videos, records, films, animations, and more.
- Web pages often contain multimedia elements of different types and formats.

**Browser Support:**

- The first web browsers had support for text only, limited to a single font in a single color.
- Later came browsers with support for colors, fonts, images, and multimedia!

**Multimedia Formats:**

- Multimedia elements (like audio or video) are stored in media files.
- The most common way to discover the type of a file, is to look at the file extension.
- Multimedia files have formats and different extensions like: .wav, .mp3, .mp4, .mpg, .wmv, and .avi.


## Common Video Formats:

There are many video formats out there.

The MP4, WebM, and Ogg formats are supported by HTML.

The MP4 format is recommended by YouTube.

**Common Audio Formats:**

- MP3 is the best format for compressed recorded music. The term MP3 has become synonymous with digital music.
- If your website is about recorded music, MP3 is the choice.

# HTML Video

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

  The HTML <video> Element
- To show a video in HTML, use the <video> element:

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

- **How it Works:**
- The controls attribute adds video controls, like play, pause, and volume.
- It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.
- The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.
- The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

**HTML <video> Autoplay :**

- To start a video automatically, use the autoplay attribute:
- Example
- ```
  <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>
  ```
  ```

- HTML Video - Methods, Properties, and Events
- The HTML DOM defines methods, properties, and events for the <video> element.
- This allows you to load, play, and pause videos, as well as setting duration and volume.
- There are also DOM events that can notify you when a video begins to play, is paused, etc.
- HTML Video Tags

| Tag | Description |
| --- | --- |
| <video> | Defines a video or movie |
| <source> | Defines multiple media resources for media elements, such as <video> and <audio> |
| <track> | Defines text tracks in media players |

# HTML Audio

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

- The HTML <audio> Element
- To play an audio file in HTML, use the <audio> element:
- Example
- ```
  <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:
- Example
- ```
  <audio controls autoplay>
    <source src="horse.ogg" type="audio/ogg">
    <source src="horse.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
  </audio>
  ```
  ```

## HTML Audio - Methods, Properties, and Events

- The HTML DOM defines methods, properties, and events for the <audio> element.
- This allows you to load, play, and pause audios, as well as set duration and volume.
- There are also DOM events that can notify you when an audio begins to play, is paused, etc.
- For a full DOM reference, go to our HTML Audio/Video DOM Reference.

# HTML Plug-ins

- Plug-ins
- Plug-ins were designed to be used for many different purposes:
- To run Java applets
- To run Microsoft ActiveX controls
- To display Flash movies
- To display maps
- To scan for viruses
- To verify a bank id

**The <object> Element:**

- The <object> element is supported by all browsers.
- The <object> element defines an embedded object within an HTML document.
- It was designed to embed plug-ins (like Java applets, PDF readers, and Flash Players) in web pages, but can also be used to include HTML in HTML:

**Example:**

- <object width="100%" height="500px" data="snippet.html"></object>

- Try it Your

- Or images if you like:
- Example
- <object data="audi.jpeg"></object>

**The <embed> Element:**

- The <embed> element is supported in all major browsers.
- The <embed> element also defines an embedded object within an HTML document.
- Web browsers have supported the <embed> element for a long time. However, it has not been a part of the HTML specification before HTML5.

Example:

- <embed src="audi.jpeg">
- The <embed> element can also be used to include HTML in HTML:

Example:

- <embed width="100%" height="500px" src="snippet.html">

# HTML YouTube Videos

- Converting videos to different formats can be difficult and time-consuming.
- An easier solution is to let YouTube play the videos in your web page.

**YouTube Video Id:**

- YouTube will display an id (like tgbNymZ7vqY), when you save (or play) a video.
- You can use this id, and refer to your video in the HTML code.

- **Playing a YouTube Video in HTML:**
- To play your video on a web page, do the following:
- Upload the video to YouTube
- Take a note of the video id
- Define an <iframe> element in your web page
- Let the src attribute point to the video URL
- Use the width and height attributes to specify the dimension of the player
- Add any other parameters to the URL (see below)
- Example
- <iframe width="420" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY"> </iframe>

- YouTube Autoplay + Mute
- You can let your video start playing automatically when a user visits the page, by adding autoplay=1 to the YouTube URL. However, automatically starting a video is annoying for your visitors!
- **Note:** Chromium browsers do not allow autoplay in most cases. However, muted autoplay is always allowed.
- Add mute=1 after autoplay=1 to let your video start playing automatically (but muted).
- YouTube - Autoplay + Muted
- <iframe width="420" height="315"
  src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&
  mute=1">
  </iframe>

**YouTube Playlist:**

- A comma separated list of videos to play (in addition to the original URL).

**YouTube Loop:**

- Add loop=1 to let your video loop forever.
- Value 0 (default): The video will play only once.
- Value 1: The video will loop (forever).

**YouTube - Loop**

- <iframe width="420" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop=1"> </iframe>

- Try it Your

**YouTube Controls:**

- Add controls=0 to not display controls in the video player.
- Value 0: Player controls does not display.
- Value 1 (default): Player controls display.

**YouTube – Controls:**

- <iframe width="420" height="315" src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0"> </iframe>

# HTML FORMS

An HTML form is used to collect user input. The user input is most often sent to a server for processing.

The <form> Element

The HTML <form> element is used to create an HTML form for user input:

```
<form>
.
form elements
.
</form>
```

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

# The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.

## The <label> Element

The <label> tag defines a label for many form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focuses on the input element.

The <label> element also helps users who have difficulty clicking on very small regions (such as radio buttons or checkboxes) - because when the user clicks the text within the <label> element, it toggles the radio button/checkbox.

The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

# HTML Form Elements

**The < select > element**

The <select> element defines a drop-down list:

Example:

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars">
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

## < option > element

The <option> elements defines an option that can be selected.

By default, the first item in the drop-down list is selected.

- To define a pre-selected option, add the 'selected' attribute to the option
- Use the 'size' attribute to specify the number of visible values
- Use the multiple attribute to allow the user to select more than one value:

Example:

```
<label for="cars">Choose a car:</label>
<select id="cars" name="cars" size="4" multiple>
  <option value="volvo">Volvo</option>
  <option value="saab">Saab</option>
  <option value="fiat">Fiat</option>
  <option value="audi">Audi</option>
</select>
```

## The <textarea> Element

The < textarea > element defines a multi-line input field (a text area):

< textarea name="message" rows="10" cols="30" >
The cat was playing in the garden.
< / textarea >

The rows attribute specifies the visible number of lines in a text area.

The cols attribute specifies the visible width of a text area.

## The <button> Element

The <button> element defines a clickable button:

<button type="button" onclick="alert('Hello World!')">Click Me!</button>

## The < fieldset > and < legend > Elements

The <fieldset> element is used to group related data in a form.

The <legend> element defines a caption for the <fieldset> element.

```
<form action="/action_page.php">
    <fieldset>
      <legend>Personalia:</legend>
      <label for="fname">First name:</label><br>
      <input type="text" id="fname" name="fname" value="John"><br>
      <label for="lname">Last name:</label><br>
      <input type="text" id="lname" name="lname" value="Doe"><br><br>
      <input type="submit" value="Submit">
    </fieldset>
</form>
```

## The <datalist> Element

The <datalist> element specifies a list of pre-defined options for an <input> element.

Users will see a drop-down list of the pre-defined options as they input data.

The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

```
<form action="/action_page.php">
     <input list="browsers">
     <datalist id="browsers">
       <option value="Internet Explorer">
       <option value="Firefox">
       <option value="Chrome">
       <option value="Opera">
       <option value="Safari">
     </datalist>
   </form>
```

## The <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

Example

Perform a calculation and show the result in an <output> element:

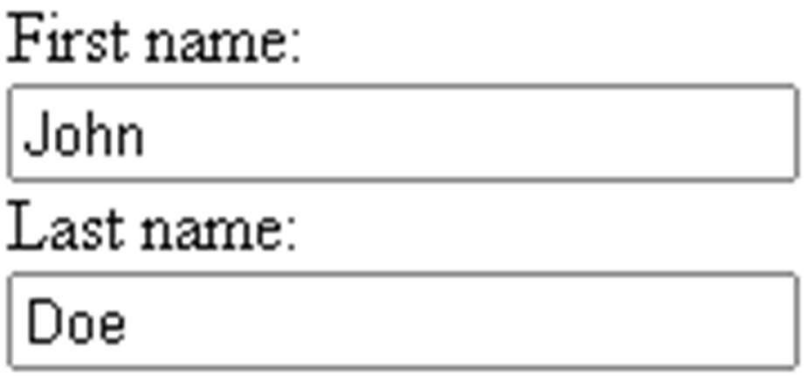
```
< form action="/action_page.php"
     oninput="x.value=parseInt(a.value)+parseInt(b.value)">
     0
     <input type="range"  id="a" name="a" value="50">
     100 +
     <input type="number" id="b" name="b" value="50">
     =
     <output name="x" for="a b"></output>
     <br><br>
     <input type="submit">
   </form>
```

# HTML Input Attributes

## The value Attribute

The input value attribute specifies an initial value for an input field.

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```



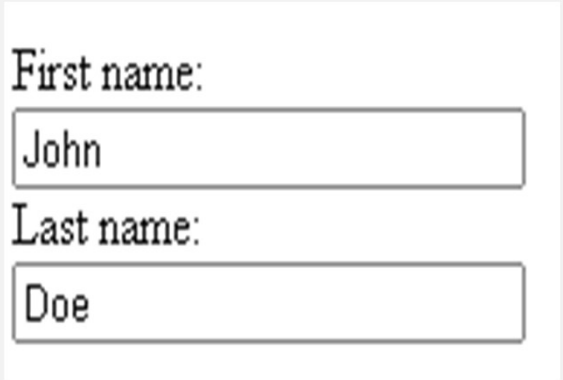
## The readonly Attribute

The input readonly attribute specifies that an input field is read-only.

A read-only input field cannot be modified (however, a user can tab to it, highlight it, and copy the text from it).

The value of a read-only input field will be sent when submitting the form.

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" readonly><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```



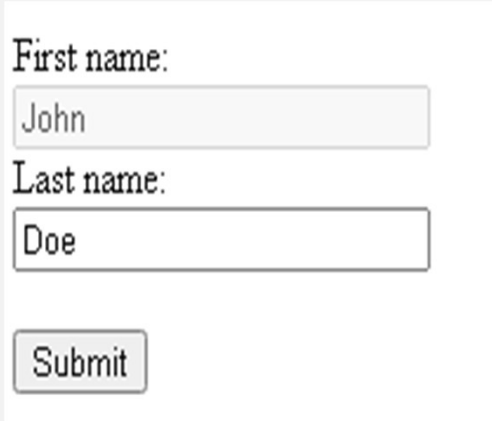
## The disabled Attribute

The input disabled attribute specifies that an input field should be disabled.

A disabled input field is unusable and un-clickable.

The value of a disabled input field will not be sent when submitting the form!

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John" disabled><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
```



## The size Attribute

The input size attribute specifies the visible width, in characters, of an input field.

The default value for size is 20.

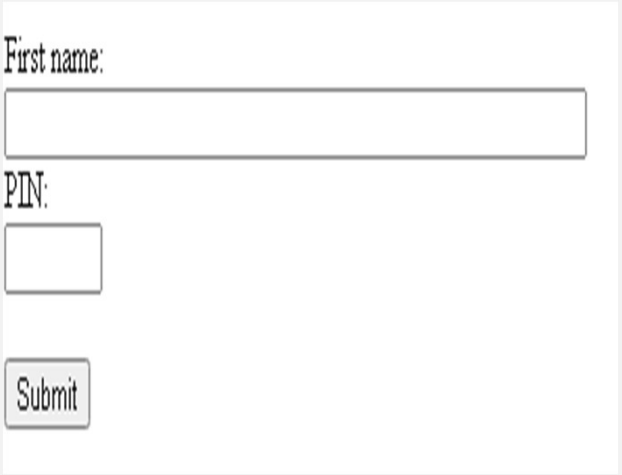
The size attribute works with the following input types: text, search, tel, url, email, and password.

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" size="4">
</form>
```

## The maxlength Attribute

The input maxlength attribute specifies the maximum number of characters allowed in an input field.

```
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="text" id="pin" name="pin" maxlength="4" size="4">
</form>
```



# The multiple Attribute

The input multiple attribute specifies that the user is allowed to enter more than one value in an input field.

The multiple attribute works with the following input types: email, and file.

```
<form>
  <label for="files">Select files:</label>
  <input type="file" id="files" name="files" multiple>
</form>
```

# The pattern Attribute

The input pattern attribute specifies a regular expression that the input field's value is checked against, when the form is submitted.

The pattern attribute works with the following input types: text, date, search, url, tel, email, and password.

```
<form>
  <label for="country_code">Country code:</label>
  <input type="text" id="country_code" name="country_code"
  pattern="[A-Za-z]{3}" title="Three letter country code">
</form>
```

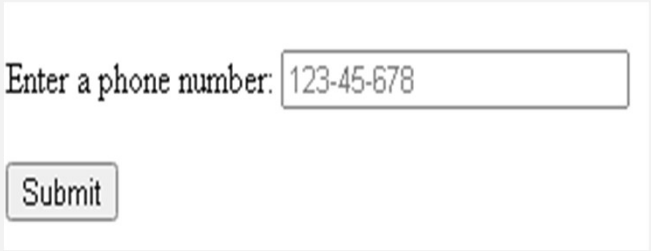
# The placeholder Attribute

The input placeholder attribute specifies a short hint that describes the expected value of an input field (a sample value or a short description of the expected format).

The short hint is displayed in the input field before the user enters a value.

The placeholder attribute works with the following input types: text, search, url, tel, email, and password.

```
<form>
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone"
  placeholder="123-45-678"
  pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}">
</form>
```



# The required Attribute

The input required attribute specifies that an input field must be filled out before submitting the form.

The required attribute works with the following input types: text, search, url, tel, email, password, date pickers, number, checkbox, radio, and file.

```
<form>
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>
</form>
```

# The list Attribute

The input list attribute refers to a <datalist> element that contains pre-defined options for an <input> element.

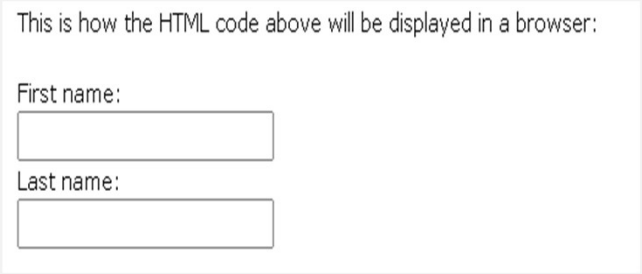
```
<form>
  <input list="browsers">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
</form>
```

# HTML INPUT TYPES

## ➢Input Type Text

It defines a single-line text input field.

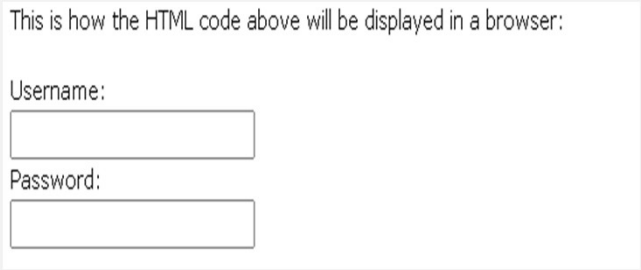
```
<form>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname">
  </form>
```



- Input Type Password

It defines a **password field.**

```
<form>
  <label for="username">Username:</label><br>
  <input type="text" id="username" name="username"><br>
  <label for="pwd">Password:</label><br>
  <input type="password" id="pwd" name="pwd">
</form>
```



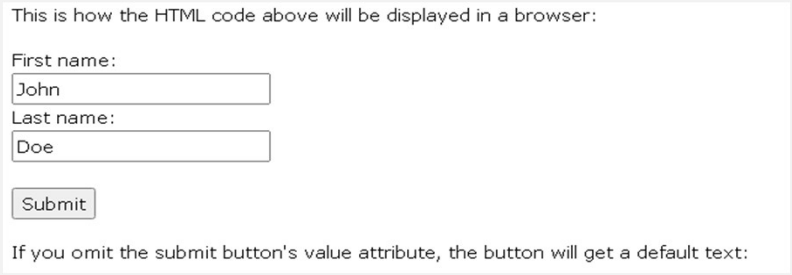
## ➢Input Type Submit

It defines a button for **submitting** form data to a **form-handler**.

The form-handler is typically a server page with a script for processing input data.

The form-handler is specified in the form's `action` attribute:

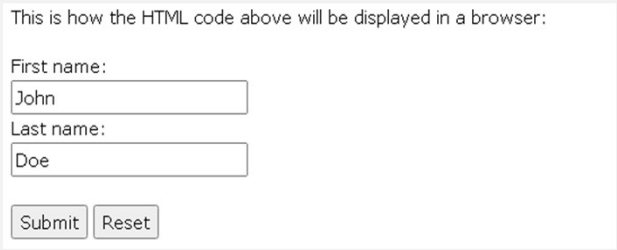
```
<form action="/action_page.php">
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </form>
```



## ➢Input Type Reset

- It defines a **reset button** that will reset all form values to their default values.

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
  <input type="reset">
</form>
```



## ➢Input Type Radio

It defines a **radio button**.

Radio buttons let a user select ONLY ONE of a limited number of choices.

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

This is how the HTML code above will be displayed in a browser:

○ HTML
○ CSS
○ JavaScript

## ➢Input Type Checkbox

It defines a **checkbox**.

Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label>
</form>
```

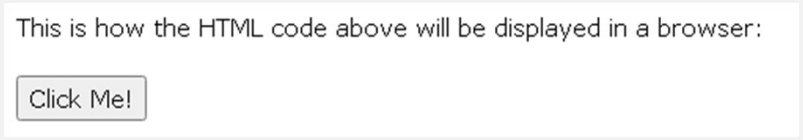
This is how the HTML code above will be displayed in a browser:

☐ I have a bike
☐ I have a car
☐ I have a boat

## ➢ Input Type Button

It defines a **button.**

```
<input type="button" onclick="alert('Hello World!')" value="Click Me!">
```




## ➢ Input Type Color

It is used for input fields that should contain a color.
Depending on browser support, a color picker can show up in the input field.

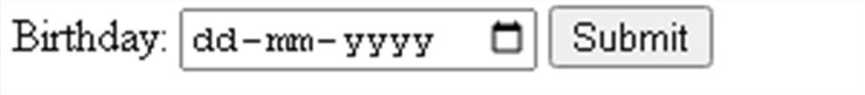
```
<form>
  <label for="favcolor">Select your favorite color:</label>
  <input type="color" id="favcolor" name="favcolor">
</form>
```

## ➢ Input Type Date

It is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

```
<form>
    <label for="birthday">Birthday:</label>
    <input type="date" id="birthday" name="birthday">
  </form>
```

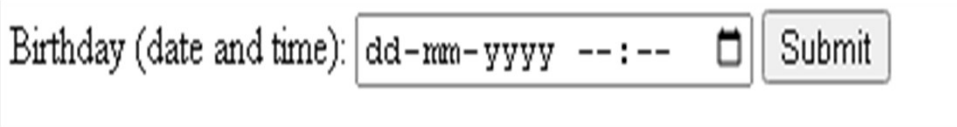



## ➢ Input Type Datetime-local

It specifies a date and time input field, with no time zone.

Depending on browser support, a date picker can show up in the input field.

```
<form>
 <label for="birthdaytime">Birthday (date and time):</label>
 <input type="datetime-local" id="birthdaytime" name="birthdaytime">
</form>
```



# ➢Input Type Email

It is used for input fields that should contain an e-mail address.

Depending on browser support, the e-mail address can be automatically validated when submitted.

Some smartphones recognize the email type and add ".com" to the keyboard to match email input.

```
<form>
     <label for="email">Enter your email:</label>
     <input type="email" id="email" name="email">
   </form>
```

# ➢Input Type Image

It defines an image as a submit button.

The path to the image is specified in the src attribute.

```
<form>
   <input type="image" src="img_submit.gif" alt="Submit" width="48" height="48">
   </form>
```

# Input Type File

It defines a file-select field and a "Browse" button for file uploads.

```
<form>
     <label for="myfile">Select a file:</label>
     <input type="file" id="myfile" name="myfile">
   </form>
```

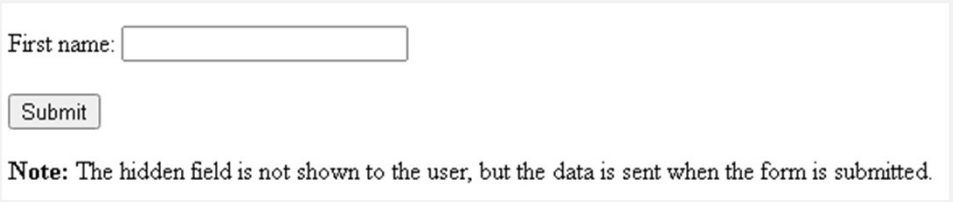
# ➢Input Type Hidden

It  defines a hidden input field (not visible to a user).

A hidden field lets web developers include data that cannot be seen or modified by users when a form is submitted.

A hidden field often stores what database record that needs to be updated when the form is submitted.

```
<form>
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
```



# ➢Input Type Range

It defines a control for entering a number whose exact value is not important (like a slider control). Default range is 0 to 100. However, you can set restrictions on what numbers are accepted with the `min`, `max`, and `step` attributes:

```
<form>
     <label for="vol">Volume (between 0 and 50):</label>
     <input type="range" id="vol" name="vol" min="0" max="50">
   </form>
```

## ➢ Input Type Search

It is used for search fields (a search field behaves like a regular text field).

```
<form>
  <label for="gsearch">Search Google:</label>
  <input type="search" id="gsearch" name="gsearch">
</form>
```

# ➢ Input Type Url

It is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted.

Some smartphones recognize the url type, and adds ".com" to the keyboard to match url input.

```
<form>
     <label for="homepage">Add your homepage:</label>
     <input type="url" id="homepage" name="homepage">
   </form>
```

# HTML Input form Attributes

## The form Attribute

The input form attribute specifies the form the <input> element belongs to.
The value of this attribute must be equal to the id attribute of the <form> element it belongs to.

Example:
An input field located outside of the HTML form (but still a part of the form):

```
 <form action="/action_page.php" id="form1">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
</form>

<label for="lname">Last name:</label>
<input type="text" id="lname" name="lname" form="form1">
```

# The input form attribute

The form attribute specifies the form an input element belongs to.

First name:

Submit

The "Last name" field below is outside the form element, but still part of the form.

Last name:

## The formaction Attribute

The input formaction attribute specifies the URL of the file that will process the input when the form is submitted.

This attribute overrides the action attribute of the <form> element.
The formaction attribute works with the following input types: submit and image.

Example

An HTML form with two submit buttons, with different actions:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formaction="/action_page2.php" value="Submit as Admin">
</form>
```

# The input formaction attribute

The formaction attribute specifies the URL of a file that will process the input when the form is submitted.

First name:

Last name:

Submit | Submit as Admin

## Submitted Form Data

### Your input was received as:

fname=&lname=

The server has processed your input and returned this answer.

**Note:** This tutorial will not teach you how servers are processing input. Processing input is explained in our PHP tutorial.

## The formenctype Attribute

The input formenctype attribute specifies how the form-data should be encoded when submitted (only for forms with method="post").

This attribute overrides the enctype attribute of the <form> element.
The formenctype attribute works with the following input types: submit and image.

Example

A form with two submit buttons. The first sends the form-data with default encoding, the second sends the form-data encoded as "multipart/form-data":

```
<form action="/action_page_binary.asp" method="post">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formenctype="multipart/form-data"
  value="Submit as Multipart/form-data">
</form>
```

# Binary Post Data

Input was received as:

------WebKitFormBoundarybq52NSAI9l0lqWUM Content-Disposition: form-data; name="fname" ------WebKitFormBoundarybq52NSAI9l0lqWUM--

This page was returned to you from the server. The server has processed your input and returned this answer.

# The input formenctype attribute

The formenctype attribute specifies how the form data should be encoded when submitted.

First name:

Submit | Submit as Multipart/form-data

## The formtarget Attribute

The input formtarget attribute specifies a name or a keyword that indicates where to display the response that is received after submitting the form.

Example
A form with two submit buttons, with different target windows:

```
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <label for="lname">Last name:</label>
  <input type="text" id="lname" name="lname"><br><br>
  <input type="submit" value="Submit">
  <input type="submit" formtarget="_blank" value="Submit to a new
window/tab">
</form>
```

## The novalidate Attribute

The novalidate attribute is a <form> attribute.
When present, novalidate specifies that all of the form-data should not be validated when submitted.

Example
Specify that no form-data should be validated on submit:

```
<form action="/action_page.php" novalidate>
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email"><br><br>
  <input type="submit" value="Submit">
</form›
```

UST

# THANK YOU

ust.com