

HTML

Morteza Ghodoosi

What is HTML?

HTML is at the core of every web page. Behind every web page you've ever visited there is HTML code that your web browser uses to display elements like text, images and tables.

In this course, you'll work with HTML code to control the structure and the elements of web page.

HTML is the standard language for documents designed to be displayed in a web browser.

Web browsers read HTML documents(containing HTML code) to display the resulting web page.

What is HTML?

HTML is based on tags. You can tell the web browser that you want to include a paragraph of text in your web page by enclosing the text in paragraph tags `<p>`

For example:

```
<p>I'm a text paragraph.</p>
```

What is HTML?

The paragraph element requires both opening and closing tags.

`<p>` is the opening paragraph tag. It defines the start of the element.

`</p>` is the closing tag, which is used to define the end of the element with forward slash.

Many HTML elements require both opening and closing tags, also called container tags.

Creating your first HTML page

HTML files are text files, so you can use any text editor to create your first webpage.

There are some very nice HTML editors available; you can choose the one that works for you. For now let's write our examples in VSCode.

Creating your first HTML page

HTML structure has three main parts: `<html>`, `<head>` and `<body>` elements.

Add the basic HTML structure to the text editor with "This is a line of text" in the body section.

```
<html>
```

```
  <head>
```

```
  </head>
```

```
  <body>
```

```
    This is a line of text.
```

```
  </body>
```

```
</html>
```

Creating your first HTML page

In our example, the file is saved as first.html

When the file is opened, the following result is displayed in the web browser.

Don't forget to save the file. HTML file names should end in either .html or .htm

Immediately following the opening HTML tag, you'll find the head of the document, which is identified by opening and closing head tags.

The head of an HTML file contains all of the non-visual elements that help make the page work.

Creating your first HTML page

To place a title on the tab describing the web page, add a `<title>` element to your head section.

```
<html>
  <head>
    <title> first page </title>
  </head>
  <body>
    This is a line of text.
  </body>
</html>
```

The title element is important because it describes the page and is used by search engines.

HTML elements

Let's add other HTML elements to the page.

The example code below creates the structure of a basic web page with 3 HTML elements.

- A level 1 heading element - defined using the `<h1>` tag
- A paragraph - defined with the `<p>` tag
- An image - defined with the `` tag

HTML elements

you can use up to 6 levels of headings in HTML, and the tags for these heading elements are `<h1>`, `<h2>` ... `<h6>`

The different heading levels help to communicate the organization and hierarchy of the content on a page.

HTML elements

Many HTML elements require both opening and closing tags, also called **container tags**.

Headings are examples of container tags.

Container tags always come with content. The content is what we want the browser to display.

Container tags come in pairs and follow this structure:

opening tag + content + closing tag

HTML elements

You can add images to your web page with the `` tag. Images are not technically inserted into a web page, they are linked.

You need to tell the browser the path of location of the image file so it can be found and displayed.

This path or location is called the source (`src`) of the image and needs to be included in the image tag.

The image tag doesn't require a closing tag. Tags that do not contain closing tags are called **empty tags**.

Headings, Lines & Comments

The browser does not display comments, but they help document the HTML and add descriptions, reminders and other notes.

```
<!-- Your comment goes here -->
```

The comment is not displayed in the browser.

Headings, Lines & Comments

To create a paragraph, simply type in the `<p>` element with its opening and closing tags.

Browsers automatically add an empty line before and after a paragraph.

Use the `
` tag to add a single line of text without starting a new paragraph.

The `
` element is an empty HTML element. It has no end tag.

Text formatting

In HTML, there is a list of elements that specify text style.

Formatting elements were designed to display special types of text.

``, `<big>`, `<i>`, `<small>`

``, `<sub>`, `<sup>`

`<ins>`, ``

Text formatting

<body>

<p>This is regular text</p>

<p> bold text </p>

<p><big> big text </big></p>

<p><i> italic text </i></p>

<p><small> small text </small></p>

<p> strong text </p>

<p>_{subscripted text}</p>

<p>^{superscripted text}</p>

<p><ins> inserted text </ins></p>

<p> deleted text </p>

</body>

This is regular text

bold text

big text

italic text

small text

strong text

subscripted text

superscripted text

inserted text

~~deleted text~~

Text formatting

The `` tag is a phrase tag. It defines important text.

Browsers display `` as ``, and `` as `<i>`.

However, the meanings of these tags differ: `` and `<i>` define bold and italic text, respectively,

while `` and `` indicate that the text is "important".

Elements

HTML documents are made up of HTML elements.

An HTML element is written using a start tag and an end tag, and with the content in between.

HTML documents consist of nested HTML elements.

Some HTML elements (like the `
` tag) do not have end tags.

Some elements are quite small. Since you can't put contents within a break tag, and you don't have an opening and closing break tag,

it's a separate, single element.

So HTML is really scripting with elements within elements.

Attributes

Attributes provide additional information about an element or a tag, while also modifying them.

Most attributes have a value, the value modifies the attribute.

```
<p align="center">
```

This text is aligned to center

```
</p>
```

In this example, the value of "center" indicates that the content within the p element should be aligned to the center.

Attributes

Attributes are always specified in the start tag, and they appear in name="value" pairs.

As an example, we can modify the horizontal line so it has a width of 50 pixels.

This can be done by using the width attribute:

```
<hr width="50px" />
```

An element's width can also be defined using:

```
<hr width="50%" />
```

An element's width can be defined using pixels or percentages.

This text is aligned to center

Attributes

The align attribute is used to specify how the text is aligned.

```
<p align="center">
```

This text is aligned to center

```
</p>
```

we have a paragraph that is aligned to the center.

The align attribute of <p> is not supported in HTML5.

This text is aligned to center

Images

The `` tag is used to insert an image. It contains only attributes, and does not have a closing tag.

The image's URL (address) can be defined using the `src` attribute.

```

```

The `alt` attribute specifies an alternate text for an image. In case the image cannot be displayed, the `alt` attribute specifies an alternate text that describes the image in words. The `alt` attribute is required.

Images

you need to put in the image location for the src attribute that is between the quotation marks.

For example, if you have a photo named "tree.jpg" in the same folder as the HTML file, your code should look like this:

```
<body>
```

```
  
```

```
</body>
```


Images

To define the image size, use the width and height attribute.

The value can be specified in pixels or as a percentage:

```

```

<!-- or -->

```

```

Loading images takes time. Using large images can slow down your page, so use them with care.

Images

By default, an image has no borders. Use the border attribute within the image tag to create a border around the image.

```

```

By default, Internet Explorer 9, and its earlier versions, display a border around an image unless a border attribute is defined.

Lists

An ordered list starts with the `` tag, and each list items is defined by the `` tag.


``

`Red`

`Blue`

`Green`

``



```
1. Red  
2. Blue  
3. Green
```

The list items will be automatically marked with numbers.

Lists

An unordered list starts with the `` tag.

``

`Red`

`Blue`

`Green`

``

The list items will be marked with bullets.

- Red
- Blue
- Green

Tables

Tables are defined by using the `<table>` tag.

Tables are divided into table rows with the `<tr>` tag. Table rows are divided into table columns (table data) with the `<td>` tag.

Here is an example of a table with one row and three columns.

```
<table>
```

```
  <tr>
```

```
    <td>Sunday</td>
```

```
    <td>Monday</td>
```

```
    <td>Friday</td>
```

```
  </tr>
```

```
</table>
```

Sunday	Monday	Friday
--------	--------	--------

Tables

Table data tags `<td>` act as data containers within the table.

They can contain all sorts of HTML elements, such as text, images, lists, other tables and so on.

A border can be added using the border attribute.

```
<table border="2">
```

The border attribute is not supported in HTML5.

Tables

A table cell can span two or more columns.

```
<table border="2">
```

```
<tr>
```

```
<td>Red</td>
```

```
<td>Blue</td>
```

```
<td>Green</td>
```

```
</tr>
```

```
<tr>
```

```
<td><br /></td>
```

```
<td colspan="2"><br /></td>
```

```
</tr>
```

```
</table>
```

Red	Blue	Green

Tables

To change your table's position, use the align attribute inside your table tag.

```
<table align="center">
```

Now let's specify a background color of red for a table cell. To do that, just use the bgcolor attribute.

```
<td bgcolor="red">Red</td>
```

In the case of styling elements, CSS is more effective than HTML.

Links

The target attribute specifies where to open the linked document.

Given a _blank value to your attribute will have the linked

```
<a href="http://www.google.com"  
target="_blank">Learn Playing</a>
```

A visited link is underlined and purple.

Inline and Block elements

In HTML, most elements are defined as **block level** or **inline elements**.

Block level elements start from a new line.

Inline elements are normally displayed without line breaks.

The `<div>` element is a block-level element that is often used as a container for other HTML elements.

When used together with some CSS styling, the `<div>` element can be used to style blocks of content.

Inline and Block elements

Similarly, the `` element is an inline element that is often used as a container for some text.

When used together with CSS, the `` element can be used to style parts of the text.

`<h2>`Some

``Important``

message

`</h2>`

Some **Important** message

Inline and Block elements

Other elements can be used either as block level elements or inline elements.

You can insert inline elements inside block elements.

For example, you can have multiple `` elements inside a `<div>` element.

Inline elements cannot contain any block level elements.

Forms

HTML forms are used to collect information from the user.

Forms are defined using the `<form>` element, with its opening and closing tags.

```
<body>
```

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

```
</body>
```

Use the action attribute to point to a webpage that will load after the user submits the form.

Usually the form is submitted to a web page on a web server.

Forms

The method attribute specifies the HTTP method (GET or POST) to be used when forms are submitted (see below for description).

```
<form action="url" method="GET">
```

```
<for action="url" method="POST">
```

Note: When you use GET, the form data will be visible in the page address.

Use POST if the form is updating data, or includes sensitive information (password).

POST offers better security because the submitted data is not visible in the page address.

Forms

To take in user input, you need the corresponding form elements, such as text fields.

The `<input>` element has many variations, depending on the type attribute. It can be a text, password, radio, URL, submit, etc.

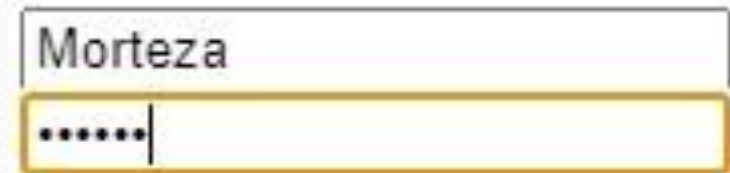
`<form>`

`<input type="text" name="username" />
`

`<input type="password" name="password" />`

`</form>`

The name attribute specifies a name for a form.



Morteza

.....|

Forms

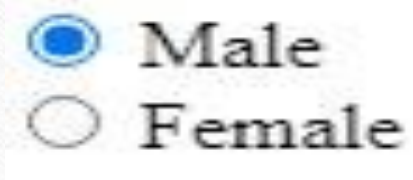
If we change the input type to radio, it allows the user select only one of a number of choices.

```
<form>
```

```
  <input type="radio" name="gender" value="male" /> Male <br />
```

```
  <input type="radio" name="gender" value="female" /> Female <br />
```

```
</form>
```



Forms

The type checkbox allows the user to select more than one option.

`<form>`

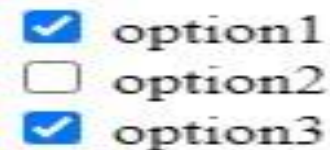
`<input type="checkbox" name="gender" value="1" /> option1
`

`<input type="checkbox" name="gender" value="2" /> option2
`

`<input type="checkbox" name="gender" value="3" /> option3
`

`</form>`

The `<input>` tag has no end tag.



Forms

The submit button submits a form to its action attribute.

```
<input type="submit" value="Submit" />
```

After the form is submitted, the data should be processed on the server using a programming language, such as PHP.

HTML colors

HTML colors are expressed as hexadecimal values.

Colors are displayed in combinations of red, green and blue light (RGB).

Hex values are written using the hashtag symbol (#), followed by either three or six characters.

RGB color and Hexadecimal color values are supported in all browsers.

All of the possible red, green and blue combinations potentially number over 16 million.

We can mix the colors to form additional colors.

HTML colors

The bgcolor attribute can be used to change the web background color.

This example would produce a dark blue background with a white headline.

```
<body bgcolor="#000099">
```

```
  <h1>
```

```
    <font color="#FFFFFF">White headline</font>
```

```
  </h1>
```

```
</body>
```

The color attribute specifies the color of the text inside a element.

Introduction to HTML5

When writing HTML5 documents, one of the first new features that you'll notice is the doc type declaration.

```
<!DOCTYPE HTML>
```

The character encoding (charset) declaration is also simplified.

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

Introduction to HTML5

Forms:

The Web Forms 2.0 specification allows for creation of more powerful form and more compelling user experience.

- Date pickers, color pickers, and numeric stepper controls have been added.
- Input field types now include email, search and URL.
- PUT and DELETE form methods are now supported.

Integrated API:

- Drag and DROP
- Audio and Video
- Offline Web Applications
- History
- Local Storage
- Geolocation
- Web Messaging

Content Models

In HTML, elements typically belonged in either the block level or inline content model.

HTML5 introduces seven main content models.

- Metadata
- Embedded
- Interactive
- Heading
- Phrasing
- Flow
- Sectioning

Content Models

In HTML, elements typically belonged in either the block level or inline content model.

HTML5 introduces seven main content models.

- Metadata
- Embedded
- Interactive
- Heading
- Phrasing
- Flow
- Sectioning

The HTML5 content models are designed to make the markup structure more meaningful for both the browser and the web designer.

Content Models

Metadata:

Content that sets up the presentation or behaviour of the rest of the content.

These elements are found in the head of the document.

Some Metadata tags:

base, link, meta, noscript, script, style, title

Content Models

Embedded:

Content that imports other resources into the document.

Some Embedded tags:

audio, video, canvas, iframe, img, math, object, svg

Content Models

Interactive:

Content specifically intended for user interaction.

Some Interactive tags:

a, audio, video, button, details, embed, iframe, img, input, label, object, select, textarea

Content Models

Heading:

Defines a section header.

Some heading tags:

h1, h2, ... , h6, hgroup

Content Models

Phrasing:

This model has a number of inline level elements in common with HTML4.

Some phrasing tags:

img, span, strong, label, br, small, sub and more.

Content Models

Flow content:

Contains the majority of HTML5 elements that would be included in the normal flow of the document.

Sectioning content:

Defines the scope of headings, content, navigation and footers.

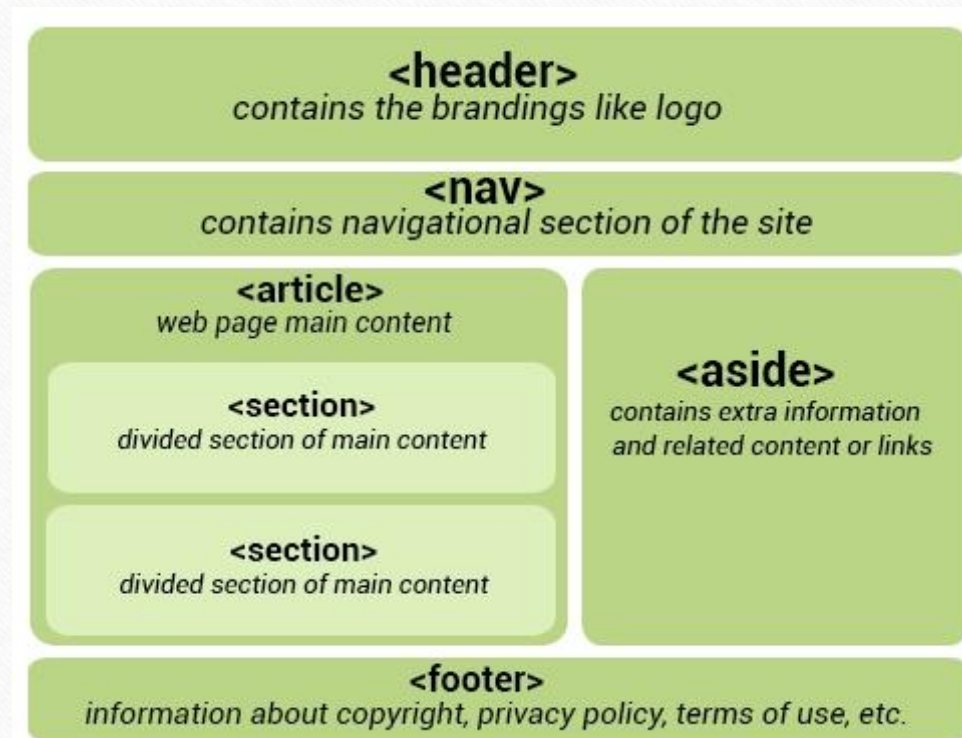
Some sectioning tags:

article, aside, nav, section

HTML5 page structure

A generic HTML5 page structure looks like this:

You may not need some of these elements, depending on your page structure.



Header, nav and footer

in HTML4, we would define a header like this:

```
<div id="header">
```

In HTML5, a simple `<header>` tag is used, instead.

The `<header>` element is appropriate for use inside the body tag.

```
<body>
```

```
  <header>
```

```
    <h1>Most Important heading.</h1>
```

```
  </header>
```

```
</body>
```

Note that the `<header>` is completely different from the `<head>` tag.

Most Important heading.

Header, nav and footer

The footer element is also widely used.

Generally we refer to a section located at the very bottom of the web page as the footer.

The following information is usually provided between these tags:

- Contact information
- Privacy policy
- Social media icons
- Terms of service
- Copyright information
- Sitemap and related documents.

Header, nav and footer

nav tag represents a section of a page that links to other pages or to certain sections within the page.

This would be a section with navigation links.

Here is an example of a major block of navigation links:

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Services</a></li>
    <li><a href="#">About me</a></li>
  </ul>
</nav>
```

- [Home](#)
- [Services](#)
- [About me](#)

Header, nav and footer

Not all of the links in the document should be inside a `<nav>` element.

The `<nav>` element is intended only for major blocks of navigation links.

Typically, the `<footer>` element often has a list of links that don't need to be in a `<nav>` element.

article, section and aside

Article is a self-contained, independent piece of content that can be used and distributed separately from the rest of the page or site.

This could be a form post, a magazine or newspaper article, a blog entry, a comment, an interactive widget or gadget, or any other independent piece of content.

The `<article>` element replaces the `<div>` element that was widely used in HTML4, along with an id or class.

```
<article>
```

```
  <h1>The article title</h1>
```

```
  <p>Contents of the article element</p>
```

```
</article>
```

article, section and aside

`<section>` is a logical container of the page or article.

Sections can be used to divide up content within an article.

For example, a homepage could have a section for introducing the company, another for news items, and still another for contact information.

Each `<section>` should be identified, typically by including a heading (h1-h6 element) as a child of the `<section>` element.

`<article>`

```
<h1>Welcome</h1>
```

```
<section>
```

```
<h1>heading</h1>
```

```
<p>content or image</p>
```

```
</section>
```

```
</article>
```

article, section and aside

`<aside>` is secondary or tangential content which could be considered separate from but indirectly related to the main content. This type of content is often represented in sidebars.

When an `<aside>` tag is used within an `<article>` tag, the content of the `<aside>` should be specifically related to that article.

`<article>`

`<h1>Gifts for everyone.</h1>`

`<p>This website will be the best place for choosing gifts</p>`

`<aside>`

`<p>Gifts will be delivered to you within 24 hours</p>`

`</aside>`

`</article>`

The audio element

Before HTML5, there was no standard for playing audio files on a web page.

The HTML5 `<audio>` element specifies a standard for embedding audio in a web page.

There are two different ways to specify the audio source file's URL. The first uses the source attribute.

`<audio src="audio.mp3" controls>Audio element not supported by your browser</audio>`

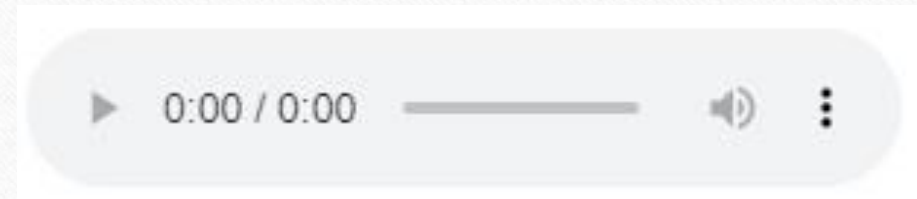
The second way used the `<source>` element inside the `<audio>` element.

`<audio controls>`

`<source src="audio.mp3" type="audio/mpeg">`

`<source src="audio.ogg" type="audio/ogg">`

`</audio>`



The audio element

Multiple `<source>` elements can be linked to different audio files. The browser will use the first recognized format.

The audio element creates an audio player inside the browser.

The text between the `<audio>` and `</audio>` tags will display in browsers that do not support the `<audio>` element.

The audio element

controls: Specifies that audio controls should be displayed (such as a play/pause button, etc)

autoplay: When this attribute is defined, audio starts playing as soon as it is ready, without asking for the visitor's permission.

<audio controls autoplay>

loop: This attribute is used to have the audio replay every time it is finished.

<audio controls autoplay loop>

Currently, there are three supported file formats for the <audio> element. MP4, WAV and OGG.

The video element

The video element is similar to the audio element.

You can specify the video source URL using an attribute in a video element, or using source elements inside the video element.

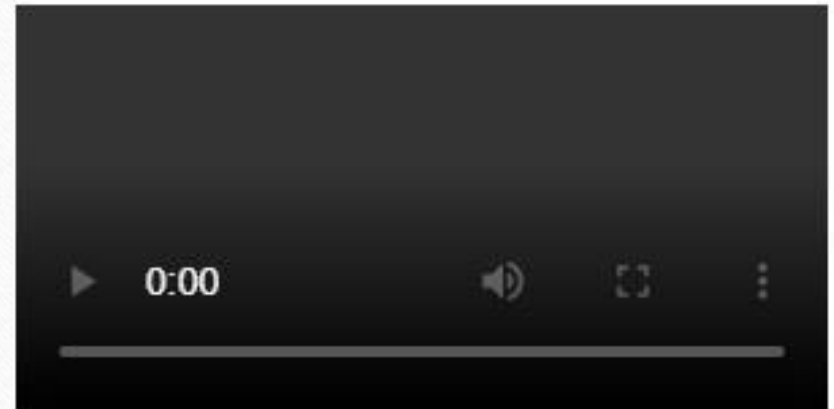
```
<video controls>
```

```
  <source src="video.mp4" type="video/mp4">
```

```
  <source src="video.ogg" type="video/ogg">
```

```
  Video is not supported by your browsers
```

```
</video>
```



The video element

Another aspect that the audio and video elements have in common is that the major browsers do not all support the same file types. If the browser does not support the first video type, it will try the next one.

Another aspect shared by both the audio and the video elements is that each has controls, autoplay and loop attributes.

In this example, the video will replay after it finishes playing.

```
<video controls autoplay loop>
```

```
  <source src="video.mp4" type="video/mp4">
```

```
  <source src="video.ogg" type="video/ogg">
```

```
  Video is not supported by your browsers
```

```
</video>
```

Currently, there are three supported video formats for the `<video>` element. MP4, WebM and OGG.

The progress element

The `<progress>` element provides the ability to create progress bars on the web.

The progress element can be used within headings, paragraphs, or anywhere else in the body.

Value: Specifies how much of the task has been completed.

Max: Specifies how much work the task requires in total.

Status: `<progress min="0" max="100" value="35"></progress>`

Use the `<progress>` tag in conjunction with **JavaScript** to dynamically display a task's progress.

Status: 

Web storage API

With HTML5 web storage, websites can store data on a user's local computer.

Before HTML5, we had to use JavaScript cookies to achieve this functionality.

The advantages of it are:

- More secure
- Faster
- Stores a larger amount of data
- Stored data is not sent with every server request

Local storage is per domain. All pages from one domain can store and access the same data.

Web storage API

There are two types of web storage objects:

- `sessionStorage()`
- `localStorage()`

Session Storage is destroyed once the user closes the browser.

Local Storage stores data with no expiration date.

You need to be familiar with basic **JavaScript** in order to understand and use the API.

Web storage API

The syntax for web storage for both local and session storage is very simple and similar. The data is stored as key/value pairs.

- Storing a Value: `localStorage.setItem("key1", "value1");`
- Getting a Value: `alert(localStorage.getItem("key1"));`
- Removing a Value: `localStorage.removeItem("key1");`
- Removing all Values: `localStorage.clear();`

The same syntax applies to the session storage, with one difference. Instead of `localStorage`, `sessionStorage` is used.

Geolocation API

In HTML5, the Geolocation API is used to obtain the user's geographical location.

Since this can compromise user privacy, the option is not available unless the user approves it.

Geolocation is much more accurate for devices with GPS, like smartphones and the like.

Geolocation API

The Geolocation API's main method is `getCurrentPosition`, which retrieves the current geographic location of the user's device.

```
navigator.geolocation.getCurrentPosition();
```

Geolocation API

`showLocation(mandatory)`: Defines the callback method that retrieves location information.

`ErrorHandler(optional)`: Defines the callback method that is invoked when an error occurs in processing the asynchronous call.

`Options(optional)`: Defines a set of options for retrieving the location information.

You need to be familiar with basic **JavaScript** in order to understand and use the API.

Geolocation API

User location can be presented in two ways: Geodetic and Civic.

1. The geodetic way to describe position refers directly to latitude and longitude.
2. The civic representation of location data is presented in a format that is more easily read and understood by the average person.

Each parameter has both a geodetic and a civic representation.

The `getCurrentPosition()` method returns an object if it is successful. The latitude, longitude and accuracy properties are always returned.

Geolocation API

Each parameter has both a geodetic and a civic representation.

Table 1

Attribute	Geodetic	Civic
Position	59.3, 18.6	Stockholm
Elevation	8 metres	Third floor
Heading	142 degrees	To the city
Speed	12 km/h	Running
Orientation	55 degrees	North-West

Drag&Drop API

The drag and drop feature lets you grab an object and drag it to a different location.

To make an element draggable, just set the draggable attribute to true.

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

Any HTML element can be draggable.

SVG

SVG stand for Scalable Vector Graphics, and is used to draw shapes with HTML-style markup.

It offers several methods for drawing paths, boxes, circles, text, and graphic images.

SVG is not pixel-based, so it can be magnified infinitely with no loss of quality.

An SVG image can be added to HTML code with just a basic image tag that includes a source attribute pointing to the image.

```

```

SVG

To draw shapes with SVG, you first need to create an SVG element tag with two attributes: width and height.

```
<svg width="1000" height="2000">  
  <circle cx="80" cy="80" r="50" fill="green" />  
</svg>
```

cx pushes the center of the circle further to the right of the screen.

cy pushes the center of the circle further down from the top of the screen.

r defines the radius

fill determines the color of our circle stroke add an outline to the circle



SVG

`<rect>` defines a rectangle

`<line>` defines a line segment

`<polyline>` defines shaped built from multiple line definitions

`<ellipse>` is similar to the `<circle>` with one exception.

You can independently change the horizontal and vertical axes of its radius, using `rx` and `ry` attributes.

`<polygon>` is used to create a graphic with at least three sides.

The polygon element is unique because it automatically closes off the shape for you.

The width and height attributes of the `<rect>` element define the height and the width of the rectangle.

HTML5 forms

HTML 5 brings many features and improvements to web form creation.

There are new attributes and input types that were introduced to help create better experiences for web users.

Form creation is done in HTML5 the same way as it was in HTML4.

```
<form>
```

```
  <label>Your name:</label>
```

```
  <input id="user" name="username" type="text">
```

```
</form>
```

Use the novalidate attribute to avoid form validation on submissions.

HTML5 forms

HTML5 has introduced a new attribute called placeholder.

On `<input>` and `<textarea>` elements, this attribute provides a hint to the user of what information can be entered into the field.

```
<form>
```

```
  <label>Your name:</label>
```

```
  <input id="user" name="username" type="text" placeholder="John Smith">
```

```
</form>
```

Your name:

HTML5 forms

The autofocus attribute makes the desired input focus when the form loads.

```
<form>
```

```
  <label>Your name:</label>
```

```
  <input id="user" name="username" type="text" placeholder="John Smith"
autofocus>
```

```
</form>
```


HTML5 forms

The required attribute tells the browser that the input is required.

The required attribute is used to make the input elements required.

```
<form>
```

```
  <label>Your name:</label>
```

```
  <input id="user" name="username" type="text" placeholder="John Smith"
  autofocus required>
```

```
</form>
```

The form will not be submitted without filling in the required fields.

HTML5 forms

The required attribute tells the browser that the input is required.

The required attribute is used to make the input elements required.

```
<form>
```

```
  <label>Your name:</label>
```

```
  <input id="user" name="username" type="text" placeholder="John Smith" autofocus required>
```

```
</form>
```

The form will not be submitted without filling in the required fields.

The autocomplete attribute specifies whether a form of input field should have autocomplete turned on or off.

When autocomplete is on, the browser automatically complete values based on values that the user has entered before.

HTML5 forms

HTML5 added several new input types:

- color
- date
- datetime
- datetime-local
- email
- month
- number
- range
- search
- tel
- time
- url
- week

New input attributes in HTML5:

- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple
- pattern(regex)
- placeholder
- required
- step

HTML5 forms

Input types that are not supported by old web browsers, will behave as input type text.

The new search input type can be used to create a search box.

```
<form>
```

```
  <label>My search:</label>
```

```
  <input id="mysearch" name="searchitem" type="search">
```

```
</form>
```

Note: You must remember to set a name for your input, otherwise nothing will be submitted.



HTML5 forms

The `<datalist>` tag can be used to define a list of pre-defined options for the search field.

```
<input id="car" type="text" list="colors" />
```

```
<datalist id="colors">
```

```
  <option value="Red">
```

```
  <option value="Green">
```

```
  <option value="yellow">
```

```
</datalist>
```



A browser rendering of an HTML5 form. It features a text input field with a dropdown arrow on the right. A dropdown menu is open, showing three options: 'Red', 'Green', and 'yellow'.

HTML5 forms

`<option>` defines the options in a drop-down list for the user to select.

The ID of the datalist element must match with the list attribute of the input box.

Some other new input types include email, url and tel are especially useful when opening a page on a modern mobile device, which recognize the input types and opens a corresponding keyboard matching the field's type.

These new types make it easier to structure and validate HTML forms.