HTML

Welcome to HTML!

HTML stands for **H**yper **T**ext **M**arkup Language.

Unlike a scripting or programming language that uses scripts to perform functions, a markup language uses tags to identify content.

Here is an example of an HTML tag:

What does a markup language use to identify content?

tags

The Web Structure

The ability to code using HTML is essential for any web professional. Acquiring this skill should be the starting point for anyone who is learning how to create content for the web.

Modern Web Design

HTML: Structure

CSS: Presentation

JavaScript: Behavior

PHP or similar: Backend

CMS: Content Management

The <html> Tag

Although various versions have been released over the years, HTML basics remain the same.

The structure of an HTML document has been compared with that of a sandwich. As a sandwich has two slices of bread, the HTML document has opening and closing HTML tags.

These tags, like the bread in a sandwich, surround everything else:

<html>

...

</html>

The <head> Tag

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.

<html>

<head>

</head>

</html>

The <body> Tag

The

body

tag follows the head tag.

All visual-structural elements are contained within the body tag.

Headings, paragraphs, lists, quotes, images, and links are just a few of the elements that can be contained within the body tag.

Basic HTML Structure:

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

The HTML File

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

Notepad

The HTML File

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>
```

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

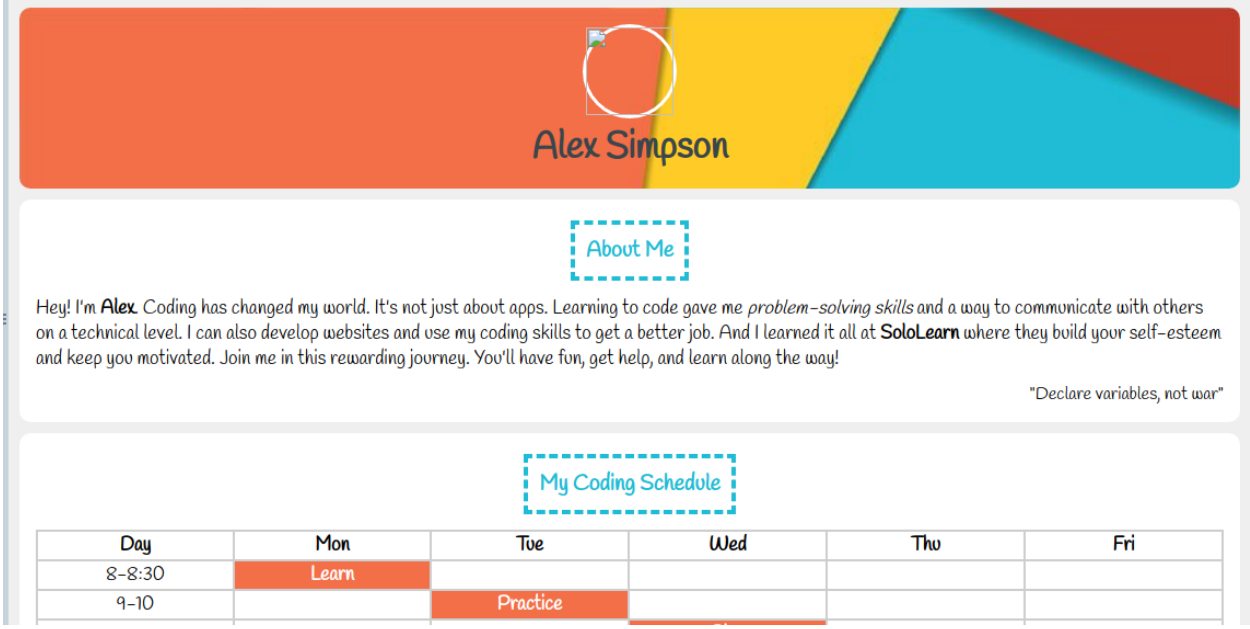
The <title> Tag

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>
```

Creating a Blog

Throughout this course, we'll help you practice and create your own unique blog project, so you'll retain what you've learned and be able to put it to use. Just keep going and follow the instructions in the



The <p> Element

To create a paragraph, simply type in the

<p>

element with its opening and closing tags:

```
<html>

   <head>

      <title>first page</title>

   </head>

   <body>
      <p>This is a paragraph. </p>
      <p>This is another paragraph. </p>

   </body>
</html>
```

Single Line Break

Use the

**
**

tag to add a single line of text without starting a new paragraph:

```
<body>

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

   <p>This is another paragraph. </p>

   <p>This is <br /> a line break </p>

</body>
```

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

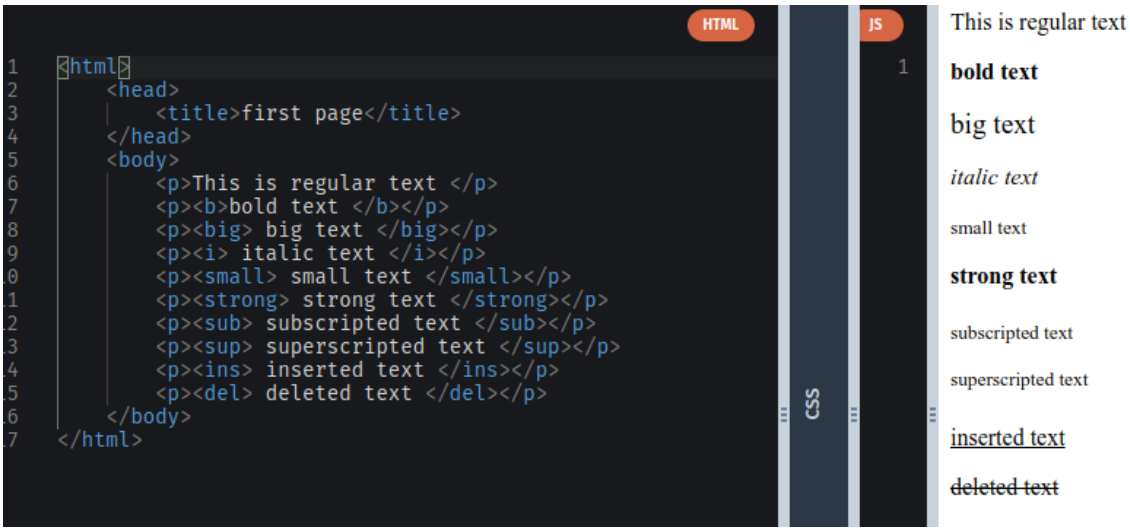
Opening the HTML file in the browser shows that a single line break has been added to the paragraph:

The
 element has no end tag.

Formatting Elements

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

Formatting elements were designed to display special types of text:



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

Formatting Elements

Each paragraph in the example is formatted differently to demonstrate what each tag does:

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

However, the meanings of these tags differ: <b> and <i> define bold and italic text, respectively, while and indicate that the text is " **important"**

HTML Headings

HTML includes six levels of headings, which are ranked according to importance.

These are

<h1>

,

<h2>

,

<h3>

,

<h4>

,

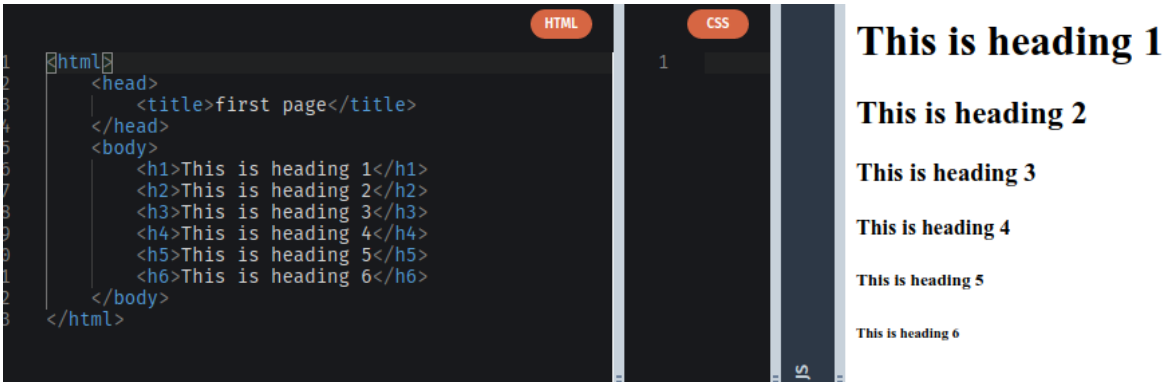
<h5>

, and

<h6>

.

The following code defines all of the headings:

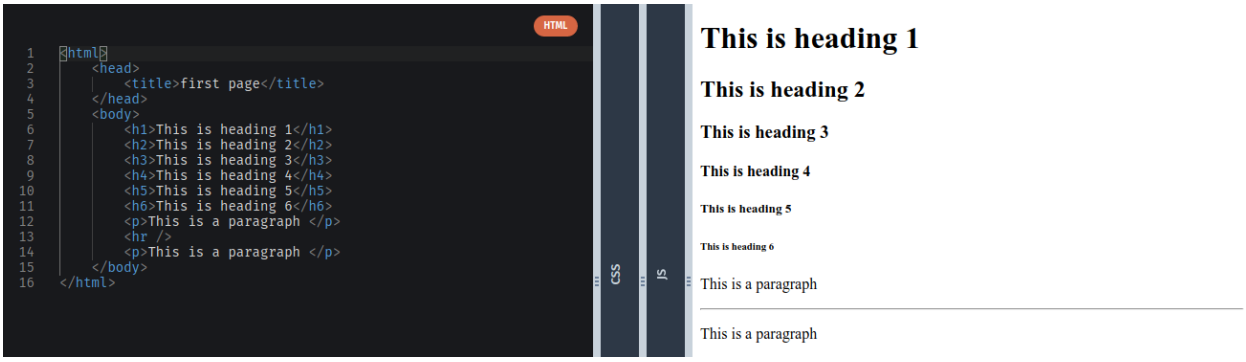


Horizontal Lines

To create a horizontal line, use the

<hr />

tag.



3ents

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

<!-- Your comment goes here →

```
<body>

   <p>This is a paragraph </p>

   <hr />

   <p>This is a paragraph </p>
   <!-- This is a comment -->

</body>
```

HTML 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. In the example below, the body element includes the <p> tags, the
 tag and the content, "This is a paragraph".

```
<html>

   <head>

      <title>first page</title>

   </head>

   <body>

      <p>This is a paragraph <br /></p>

   </body>
</html>
```

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 el

HTML Attributes

Attributes provide

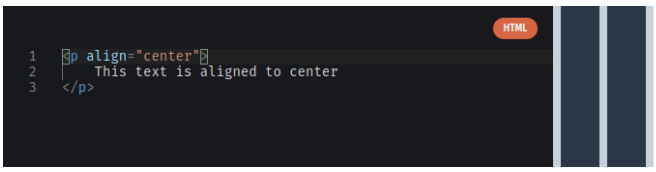
additional information

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>
```



This text is aligned to center

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

Attribute Measurements

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

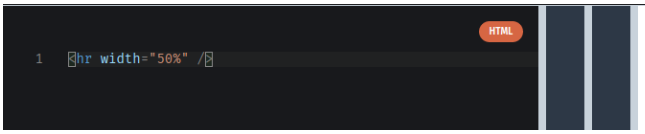
This can be done by using the width attribute:

This can be done by using the width attribute:

<hr width="50px" />

An element's width can also be defined using percentages:

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



What measurement units can be used for the width attribute?

pixel and %

What attribute is used to align the contents of an element to the right, center or left?

align

The <img> Tag

The

<img>

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 HTML image syntax looks like this:

The alt attribute specifies an alternate text for an image.

Image Location

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:

```
<html>

  <head>

    <title>first page</title>

  </head>

  <body>
      <img src="tree.jpg" alt="" />
  </body>
</html>
```

HTML

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**.

Image Resizing

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

The value can be specified in

pixels or as a **percentage:**

```
<html>
  <head>
    <title>first page</title>
  </head>
  <body>
    <img src="tree.jpg" height="150px" width="150px" alt="" />
    <!-- or -->
    <img src="tree.jpg" height="50%" width="50%" alt="" />
  </body>
</html>
```

HTML

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

```html
<html>
    <head>
        <title>first page</title>
    </head>
    <body>
        <img src="http://www.sololearn.com/images/tree.jpg"
        alt="" width="50%" height="50%" >
    </body>
</html>
```

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.

Image Resizing

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

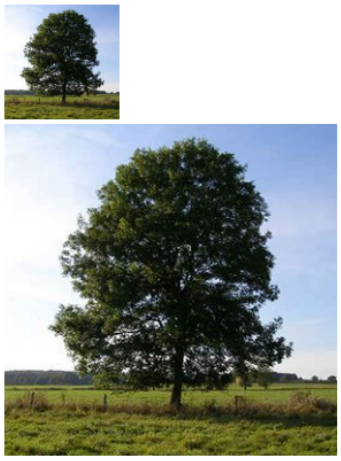
The value can be specified in

pixels

or as a

percentage:

```html
<html>
    <head>
        <title>first page</title>
    </head>
    <body>
        <img src="http://www.sololearn.com/images/tree.jpg"
        height="100px" width="100px" alt="" />
        <br />
        <img src="http://www.sololearn.com/images/tree.jpg"
        height="50%" width="50%" alt="" />
    </body>
</html>
```



What two attributes can be used to resize images inside HTML code?

height and width

Image Border

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

Try it Yourself

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

The <a> Tag

Links are also an integral part of every web page. You can add links to text or images that will enable the user to click on them in order to be directed to another file or webpage.

In HTML, links are defined using the **<a>** tag.

Use the **href** attribute to define the link's destination address:

<a **href**="">

To link an image to another document, simply nest the <img> tag inside <a> tags.

The

target

attribute specifies where to open the linked document.

Giving a

_blank

value to your attribute will have the link open in a new window or new tab:

```
<a href="http://www.sololearn.com" target="_blank">

  Learn Playing
</a>
```

A visited link is underlined and purple.

HTML Ordered Lists

An ordered list starts with the

<ol>

tag, and each list item is defined by the

<li>

tag.

Here is an example of an

ordered list

```
<html>

  <head>

    <title>first page</title>
```

</head>

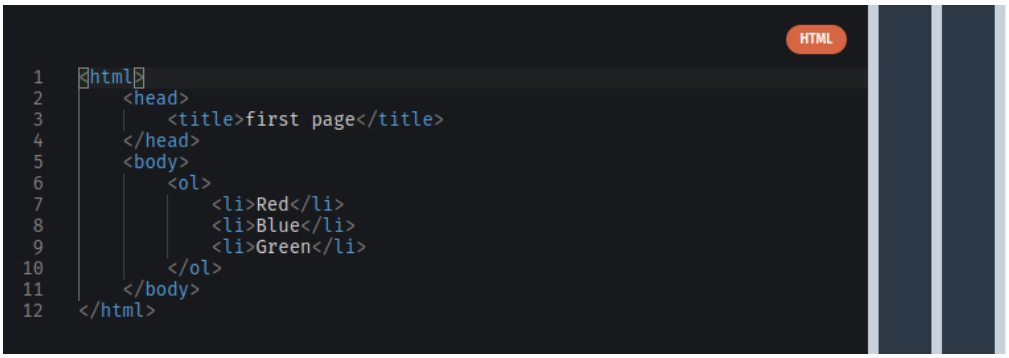
<body>
<ol>

**
Red
**

**
Blue
**

**
Green
**

**
**
</body>
</html>



HTML Unordered List

An unordered list starts with the

<ul>

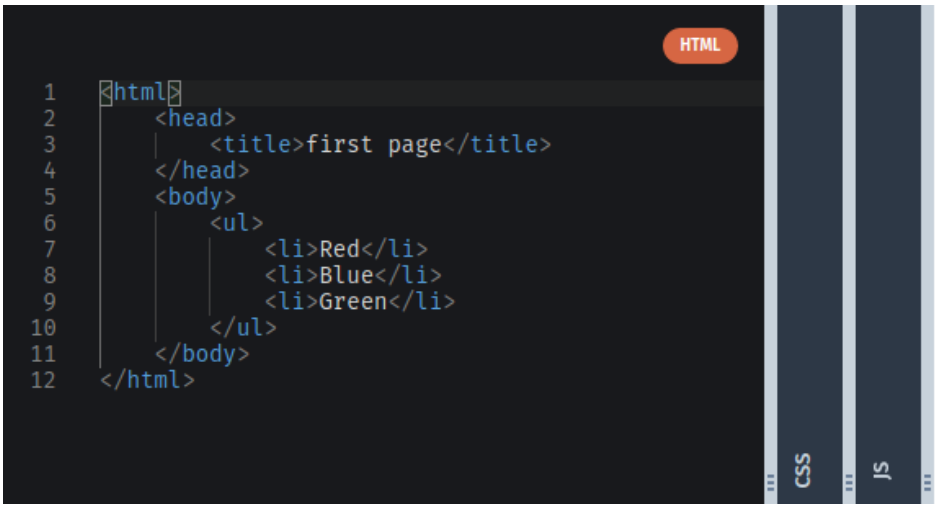
tag.

<html>

<head>

<title>first page</title>

```
</head>

<body>
 <ul>
 <li>Red</li>
 <li>Blue</li>
 <li>Green</li>
 </ul>
</body>
</html>
```



Creating a Table

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 a**nd **three columns:**

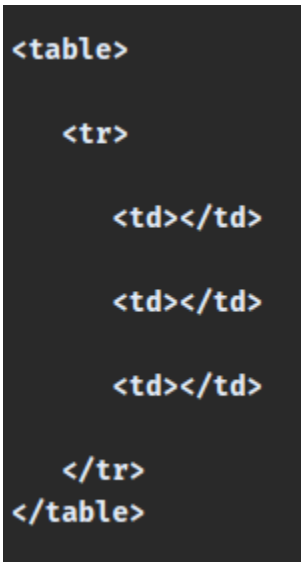


Table data tags <td> act as data containers within the table.

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

The border and colspan Attributes

The border and colspan Attributes

A border can be added using the **border** attribute:

<table **border**="2">

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>
```

Colspan Color

The example below demonstrates the **colspan** attribute in action:

```
<table border="2">
  <tr>
    <td>Red</td>
    <td>Blue</td>
    <td>Green</td>
  </tr>
  <tr>
    <td>Yellow</td>
    <td colspan="2">Orange</td>
  </tr>
</table>
```

HTML

Types of Elements

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

Block level elements start from a new line.

For example

: <h1>, <form>, <li>, <ol>, <ul>, <p>, <pre>, <table>, , etc.

Inline elements are normally displayed without line breaks.

For example

: <b>, <a>, , <img>, <input>, , , etc.

The

** **

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 element can be used to style blocks of content:

```
<html>

  <body>

    <h1>Headline</h1>
<div style="background-color:green; color:white; padding:20px;">
      <p>Some paragraph text goes here.</p>

      <p>Another paragraph goes here.</p>
</div>
  </body>
</html>
```

Types of Elements

Other elements can be used either as block level elements or inline elements. This includes the following elements:

APPLET- embedded Java applet

IFRAME- Inline frame

INS- inserted text

MAP- image map

OBJECT- embedded object

SCRIPT- script within an HTML document

You can insert inline elements inside block elements. For example, you can have multiple **** elements inside a ** ** element.

Inline elements cannot contain any block level elements.

The <form> Element

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.

```
<form action="http://www.sololearn.com">
</form>
```

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

The method and name Attributes

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">
```

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

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

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.

The example below shows a form requesting a username and password:

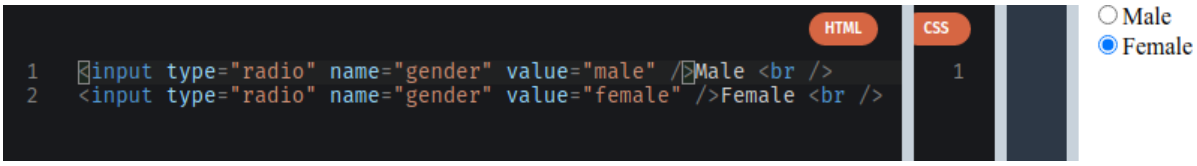
```
<form>

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

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

Form Elements

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

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

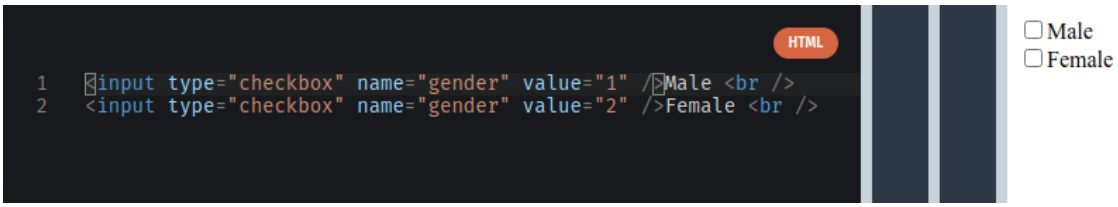
<input type="**radio**" name="gender" value="**female**" /> Female



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

<input type="**checkbox**" name="gender" value="1" /> Male

<input type="**checkbox**" name="gender" value="2" /> Female



The <input> tag has no end tag.

Form Elements

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.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

As you can see, there are 16 values there, 0 through F. Zero represents the lowest value, and F represents the highest.

HTML Color Model

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 hex characters.

As shown in the picture below, the circles overlap, forming new colors:

RGB color values are supported in all browsers.

Color Values

All of the possible

red,**green**, and **blue** combinations potentially number over 16 million.

Here are only a few of them:

We can mix the colors to form additional colors.

Background and Font Colors

The **bgcolor** attribute can be used to change the web page's background color.

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

<html>

<head>

<title>first page</title>

</head>

<body bgcolor="#000099">

<h1>

 White headline

</h1>

</body>

</html>

The color attribute specifies the color of the text inside a <font> element.

The <frame> Tag

A page can be divided into frames using a special frame document. The **<frame>** tag defines one specific window (frame) within a **<frameset>** . Each <frame> in a <frameset> can have different attributes, such as border, scrolling, the ability to resize, etc.

The <frameset> element specifies the number of columns or rows in the frameset, as well as what percentage or number of pixels of space each of them

occupies.

<**framesetcols**="100, 25%, *"><**/frameset**>

<**frameset rows**="100, 25%, *"><**/frameset**>

HTML

The <frameset> tag is not supported in HTML5.

Use the **<noresize>** attribute to specify that a user cannot resize a <frame> element:

<frame **noresize**="**noresize**">

HTML

Frame content should be defined using the **src** attribute. Lastly, the **<noframes>**

element provides a way for browsers that do not support frames to view the page. The element can contain an alternative page, complete with a body tag and any other elements.

```
<frameset cols="25%,50%,25%">

  <frame src="a.htm" />

  <frame src="b.htm" />

  <frame src="c.htm" />

  <noframes>Frames not supported!</noframes>
</frameset>

<!-- Media section start -->
<div class="section" style="background-color:teal">
```

```
<h1><span>My Media</span></h1>
<iframe height="250" width="500"
src="https://www.youtube.com/embed/Q6_5InVJZ88"
allowfullscreen frameborder="0" >
</iframe>
</div>
<!-- Media section end -->
```

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">

New Elements in HTML5

<article>, <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <footer>, <header>, <nav>, <output>, <progress>, <section>, <video>, and even more!

The default character encoding in HTML5 is UTF-8.

New in HTML5

Forms

- The Web Forms 2.0 specification allows for creation of more powerful forms and more compelling user experiences.

- 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 (Application Programming Interfaces)

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

The List of 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 behavior of the rest of the content. These elements are found in the

head of the document.

Elements: <base>, <link>, <meta>, <noscript>, <script>, <style>, <title>

Embedded : Content that imports other resources into the document.

Elements:<audio>, <video>, <canvas>, <iframe>, <img>, <math>, <object>, <svg>

Interactive: Content specifically intended for user interaction.

Elements:<a>, <audio>, <video>, <button>, <details>, <embed>, <iframe>, <img>, <input>, <label>, <object>, <select>, <textarea>

Heading: Defines a section header.

Elements:<h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <hgroup>

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

Elements:<img>, , , <label>,
, <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.

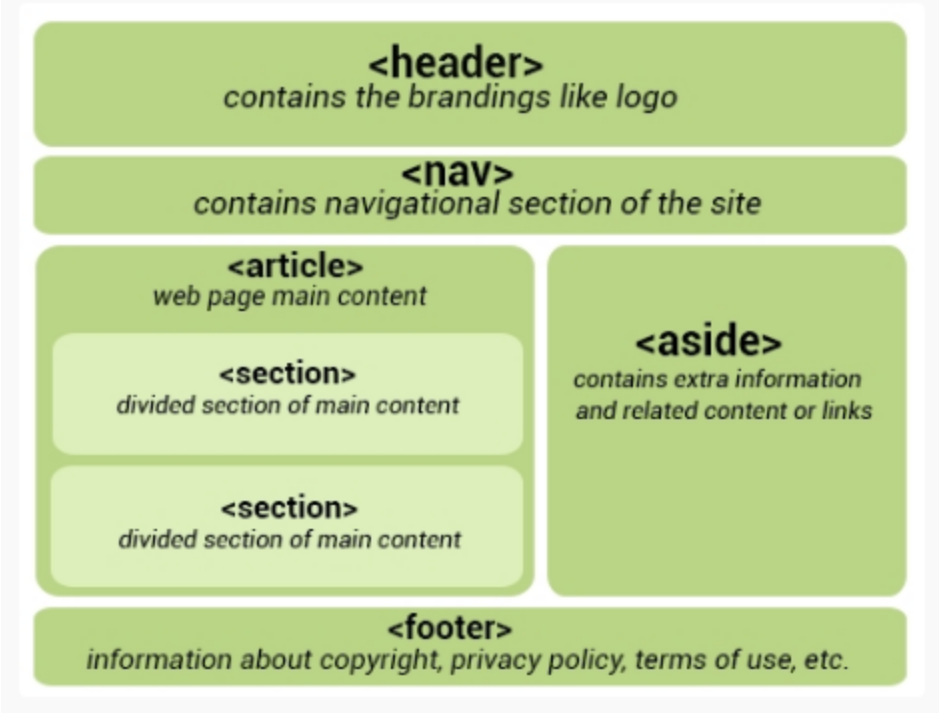
Sectioning content: Defines the scope of headings, content, navigation, and footers.

Elements: <article>, <aside>, <nav>, <section>

The various content models overlap in certain areas, depending on how they are being used.

Page Structure in HTML5

A generic HTML5 page structure looks like this:



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

The <header> Element

In HTML4, we would define a header like this:

HTML

In HTML5, a simple

<header>

tag is used, instead.

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

<!DOCTYPE html>

<html>

<head></head>

<body>

<header>

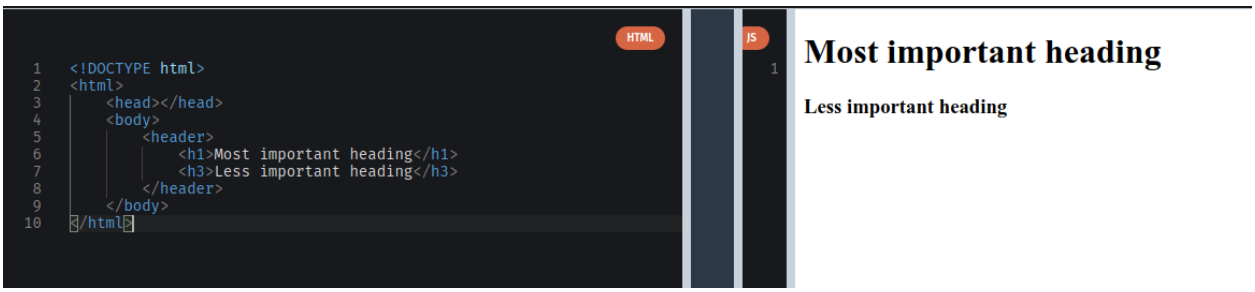
<h1>Most important heading</h1>

<h3>Less important heading</h3>

</header>

</body>

</html>



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

The <footer> Element

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.

<footer></footer>

HTML

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

The <nav> Element

This 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:

Not all of the links in a 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.

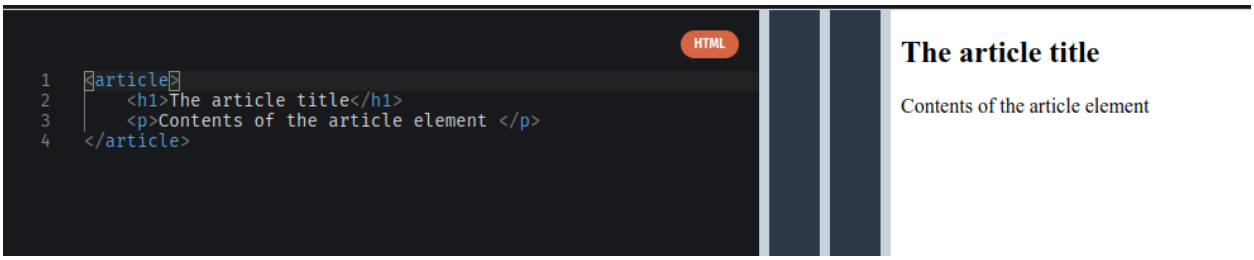
The <article> Element

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 forum 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 element that was widely used in HTML4, along with an id or class.

When an <article> element is nested, the inner element represents an article related to the outer element. For example, blog post comments can be <article> elements nested in the <article> representing the blog post.



The <section> Element

<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.

If it makes sense to separately syndicate the content of a <section> element, use an <article> element instead.

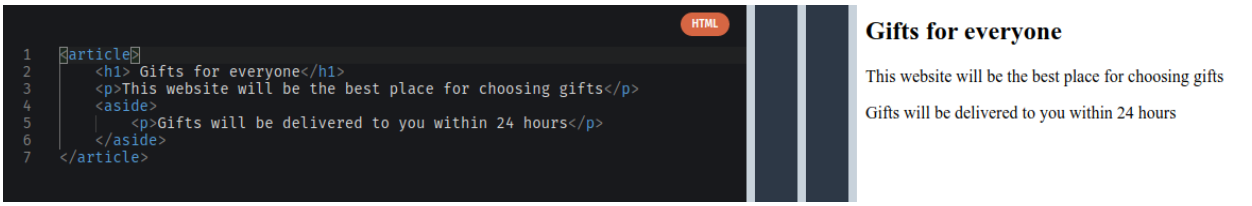
The <aside> Element

<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.



When an <aside> tag is used outside of an <article> tag, its content should be related to the surrounding content.

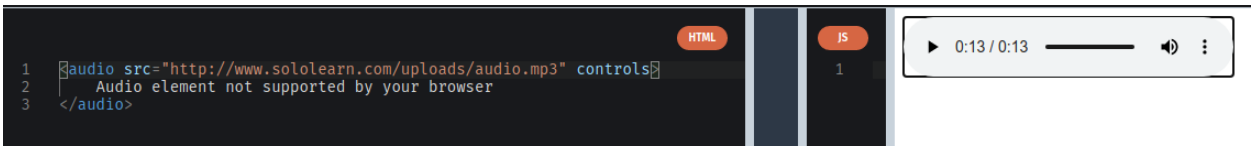
Audio on the Web

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

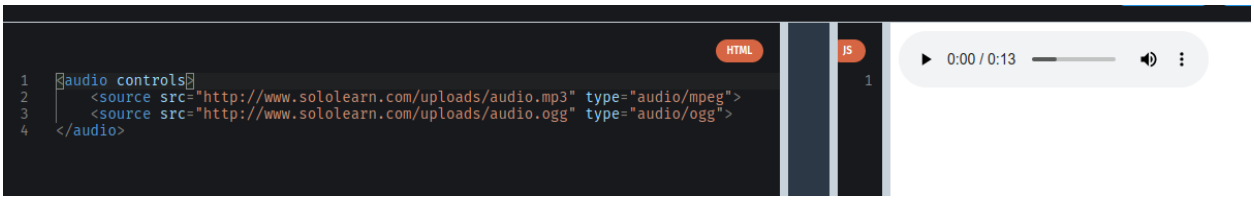
The HTML5 <audio> element specifies a standard for embedding audio in a web page.

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

```
<audio src="http://www.sololearn.com/uploads/audio.mp3" controls>
Audio element not supported by your browser
</audio>
```




```
<audio controls>
<source src="http://www.sololearn.com/uploads/audio.mp3" type="audio/mpeg">
<source src="http://www.sololearn.com/uploads/audio.ogg" type="audio/ogg">
</audio>
```

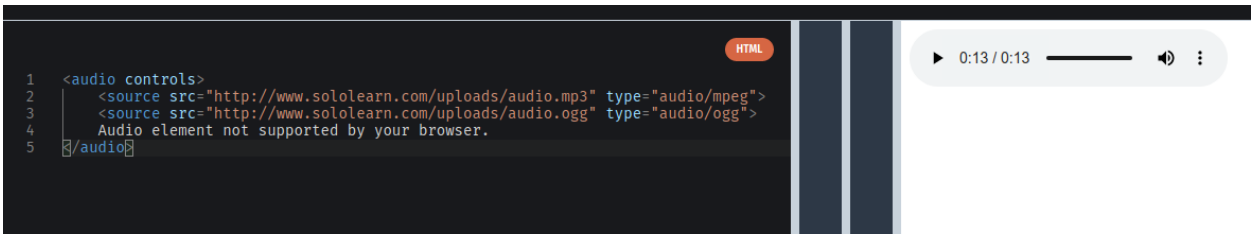


Multiple <source> elements can be linked to different audio files. The browser will use the first recognized format.

Audio on the Web

The <audio> element creates an audio player inside the browser.

```
<audio controls>
<source src="http://www.sololearn.com/uploads/audio.mp3" type="audio/mpeg">
<source src="http://www.sololearn.com/uploads/audio.ogg" type="audio/ogg">
Audio element not supported by your browser.
</audio>
```



The text in between the audio tags is: for not supported browsers

Attributes of <audio>

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>
<source src="http://www.sololearn.com/uploads/audio.mp3" type="audio/mpeg">
Audio element not supported by your browser.
</audio>
```

loop

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

```
<!-- Audio -->

<audio controls autoplay loop>
<source src="apple_ring.mp3" type="audio/mpeg">
Not supported audio
</audio>
```

Currently, there are three supported file formats for the <audio> element: MP3, WAV, and OGG.

Videos in HTML

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 -->
<video controls>
<source src="video.mp4" type="video/mp4" >
not supporded video
</video>
```

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.

Attributes of <video>

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 -->
<video controls autoplay loop>
<source src="video.mp4" type="video/mp4" >
not supporded video
</video>
```

Progress Bar

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.

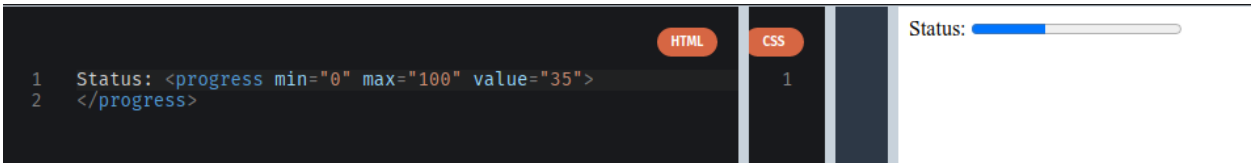
Progress Element Attributes

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

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

Example:

```
<h4> Status: <progress max="100" value="80"></progress></h4>
```



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

HTML5 Web Storage

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 Web Storage

- 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.

Before HTML5, application data was stored in: cookies

Types of Web Storage Objects

There are two types of web storage objects:

- **sessionStorage()**
- **localStorage()**

Local vs. Session

- 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.

Working with Values

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: Javascript

localStorage.**setItem**("key1", "value1"); →**JS**

Getting a Value:

//this will print the value

alert(localStorage.**getItem**("key1")); →**JS**

Removing a Value:

localStorage.**removeItem**("key1"); →**JS**

Removing All Values:

localStorage.**clear**(); →**JS**

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

What is the 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.

Using HTML Geolocation

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

navigator.geolocation.getCurrentPosition();

JS

Parameters:

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.

Presenting Data

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:

Attribute	Geodetic	Civic
Position	59.3, 18.6	Stockholm
Elevation	10 meters	4 th floor
Heading	234 degrees	City center
Speed	5km / h	Walking
Orientation	45 degrees	North-East

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

Making Elements Draggable

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:

 →**HTML**

Any HTML element can be draggable.

The API for HTML5 drag and drop is event-based.

Example:

```
<!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="box" ondrop="drop(event)"
ondragover="allowDrop(event)"

   style="border:1px solid black;

   width:200px; height:200px"></div>

   <img id="image" src="sample.jpg" draggable="true"
  ondragstart="drag(event)" width="150" height="50" alt="" />

</body>
</html>
```

What to Drag

When the element is dragged, the **ondragstart** attribute calls a function, drag(event), which specifies what data is to be dragged.

The **dataTransfer.setData()** method sets the data type and the value of the dragged data:

```
function drag(ev) {

    ev.dataTransfer.setData("text", ev.target.id);
}
```

In our example, the data type is "text" and the value is the ID of the draggable element ("image").

Where to Drop

The **ondragover** event specifies where the dragged data can be dropped. By default, data and elements cannot be dropped in other elements. To allow a drop, we must prevent the default handling of the element. This is done by calling the event. **preventDefault** () method for the **ondragover** event.

```
function drop(ev) {
ev.preventDefault();
var data = ev.dataTransfer.getData("text");
ev.target.appendChild(document.getElementById(data));
}
```

The **preventDefault()** method prevents the browser's default handling of the data (default is open as link on drop). The dragged data can be accessed with the **dataTransfer.getData()** method. This method will return any data that was set to the same type in the setData() method. The dragged data is the ID of the dragged element ("image"). At the end, the dragged element is appended into the drop element, using the appendChild() function.

Basic knowledge of JavaScript is required to understand and use the API.

Drawing Shapes

SVG stands for **S**calable **V**ector **G**raphics, 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.

Inserting SVG Images

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

```
<img src="image.svg" alt="" height="300" />
```

HTML

SVG defines vector-based graphics in XML format.

Drawing a Circle

To draw shapes with SVG, you first need to create an

SVG

element tag with two attributes: width and height.

```
svg width="1000" height="1000"></svg>
```

HTML

To create a circle, add a

<circle>

tag:

```
<svg width="2000" height="2000">
```

```
  <circle cx="80" cy="80" r="50" fill="green" />
</svg>
```

HTML

Try it Yourself

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 - adds an outline to the circle

Result:

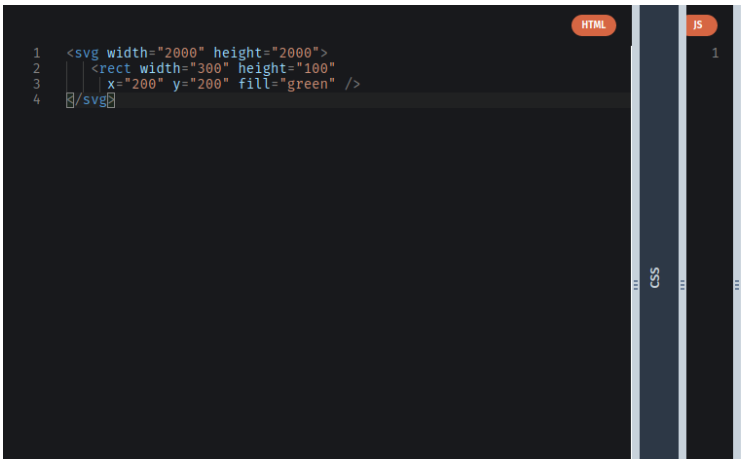
Every element and every attribute in SVG files can be animated.

Other Shape Elements

<rect>

defines a rectangle:

```
<svg width="2000" height="2000">

  <rect width="300" height="100"

    x="20" y="20" fill="green" />
</svg>
```



<line> defines a line segment:

```
<svg width="400" height="410">

    <line x1="10" y1="10" x2="200" y2="100"

        style="stroke:#000000; stroke-linecap:round;

        stroke-width:20"  />
</svg>
```

(x1, y1) define the start coordinates(x2, y2) define the end coordinates.

<polyline>

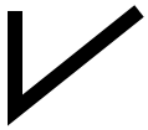
defines shapes built from multiple line definitions:

```
<svg width="2000" height="500">

    <polyline style="stroke-linejoin:miter; stroke:black;

        stroke-width:12; fill: none;"

        points="100 100, 150 150, 200 100" />
</svg>
```

Points are the polyline's coordinates.

The code below will draw a black check sign:

```
<svg width="2000" height="500">
    <polyline style="stroke-linejoin:miter; stroke:black;
        stroke-width:12; fill: none;"
        points="100 100, 100 180, 200 100" />
</svg>
```



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

<ellipse> and <polygon>

Ellipse

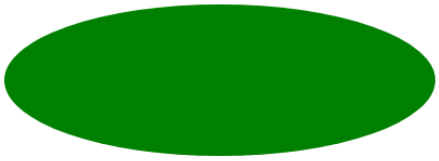
The <**ellipse**> is similar to the <**circle**>, with one exception:

You can independently change the horizontal and vertical axes of its radius, using the

rx and **ry** attributes.

```
<svg height="500" width="1000">

  <ellipse cx="200" cy="100" rx="150" ry="70" style="fill:green" />
</svg>
```

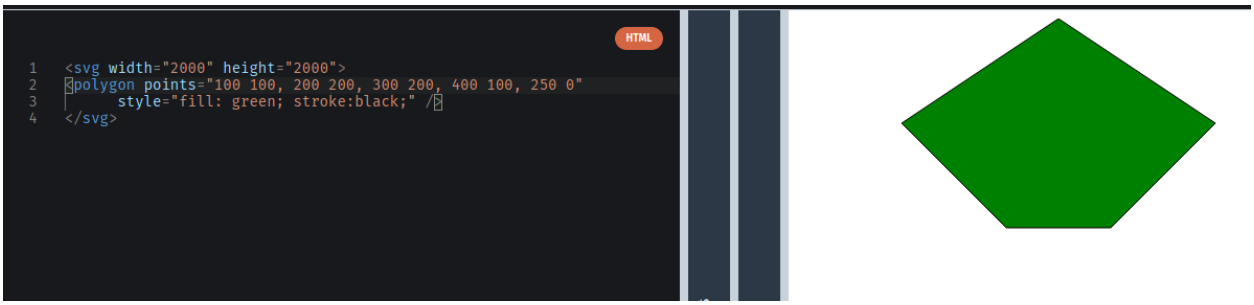
```
<svg height="500" width="1000">
  <ellipse cx="200" cy="100" rx="200" ry="70" style="fill:green" />
</svg>
```



Polygon

The **<polygon>** element 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.

```
<svg width="2000" height="2000">

<polygon points="100 100, 200 200, 300 0"

    style="fill: green; stroke:black;" />
</svg>
```



Polygon comes from Greek. "Poly" means "many" and "gon" means "angle."

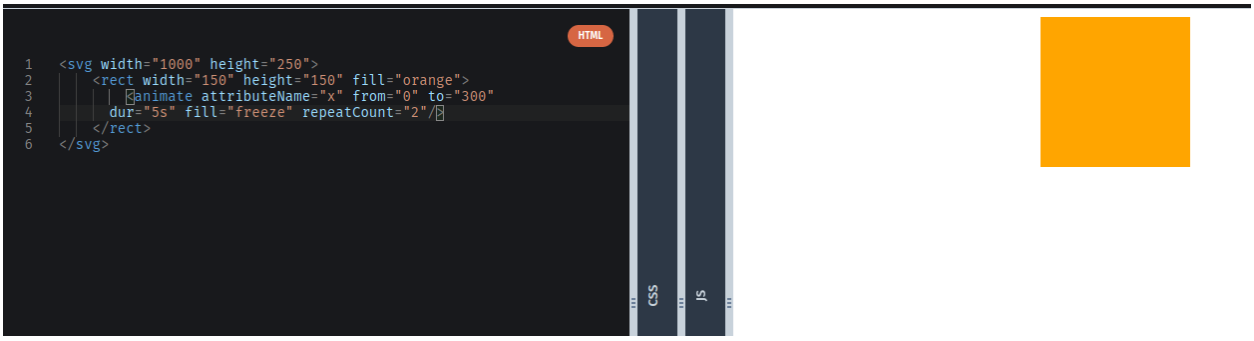
Shape Animations

SVG animations can be created using the <animate> element.

The example below creates a rectangle that will change its position in 3 seconds and will then repeat the animation twice:

```
<svg width="1000" height="250">

<rect width="150" height="150" fill="orange">
  <animate attributeName="x" from="0" to="300"
   dur="3s" fill="freeze" repeatCount="2"/>
```

</rect>
</svg>



attributeName : Specifies which attribute will be affected by the animation

from : Specifies the starting value of the attribute

to : Specifies the ending value of the attribute

dur : Specifies how long the animation runs (duration)

fill : Specifies whether or not the attribute's value should return to its initial value when the animation is finished (Values: "remove" resets the value; "freeze" keeps the “to value”)

repeatCount : Specifies the repeat count of the animation

In the example above, the rectangle changes its x attribute from 0 to 300 in 3 seconds.

To repeat the animation indefinitely, use the value "indefinite" for the repeatCount attribute.

Paths

The <path> element is used to define a path.

The following commands are available for path data:

M: moveto

L: lineto

H: horizontal lineto

V:vertical lineto

C: curveto

S: smooth curveto

Q: quadratic Bézier curve

T: smooth quadratic Bézier curveto

A:elliptical Arc

Z: close path Define a path using the **d** attribute:

```
<svg width="500" height="500">

<path d="M 0 0 L200 200 L200 0 Z" style="stroke:#000;  fill:none;" />
</svg>
```

M places our "virtual pen" down at the position 0, 0. It then moves 200px down and to the right, then moves up to the position 200, 0. The Z command closes the shape, which results in a hypotenuse:

All of the above commands can also be expressed with lower case letters. When capital letters are used, it indicates absolute position; lower case indicates relative position.

The <canvas> Element

The HTML canvas is used to draw graphics that include everything from simple lines to complex graphic objects.

The <canvas> element is defined by:

```
<canvas id="canvas1" width="200" height="100">

</canvas>
```

The <canvas> element is only a container for graphics. You must use a script to actually draw the graphics (usually JavaScript).

The <canvas> element must have an

id attribute

so it can be referred to by JavaScript:

```
<html>

  <head></head>

  <body>
<canvas id="canvas1"

  width="400" height="300"></canvas>

  <script>

    var can = document.getElementById("canvas1");

    var ctx = can.getContext("2d");

  </script>

  </body>
</html>
```

getContext() returns a drawing context on the canvas.

Basic knowledge of JavaScript is required to understand and use the Canvas.

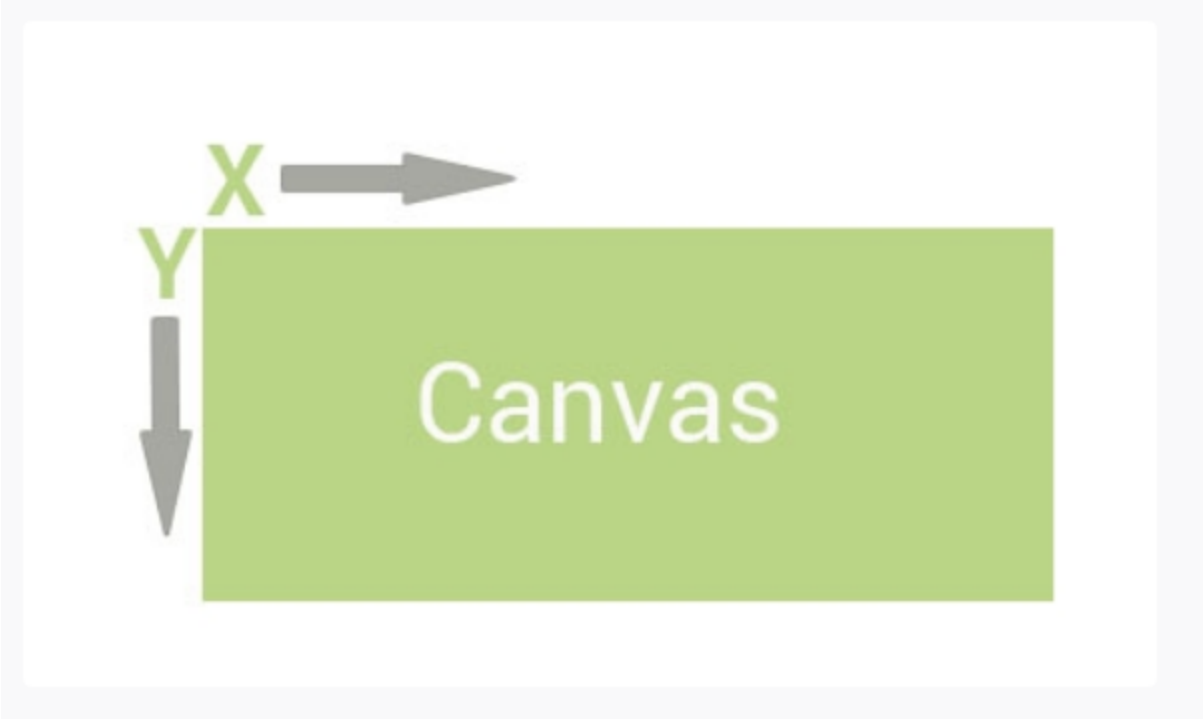
Canvas Coordinates

The HTML canvas is a two-dimensional grid.

The upper-left corner of the canvas has the coordinates (0,0).

X coordinate increases to the right.

Y coordinate increases toward the bottom of the canvas.



The <canvas> element is only a container for graphics.

Drawing Shapes

The **fillRect(x, y, w, h)** method draws a "filled" rectangle, in which w indicates width and h indicates height. The default fill color is black.

A black 100*100 pixel rectangle is drawn on the canvas at the position (20, 20):

The

fillStyle

property is used to set a color, gradient, or pattern to fill the drawing.

Using this property allows you to draw a green-filled rectangle.

```
var canvas=document.getElementById("canvas1");

var ctx=canvas.getContext("2d");

ctx.fillStyle ="rgba(0, 200, 0, 1)";
ctx.fillRect (36, 10, 22, 22);
```

The canvas supports various other methods for drawing:

Draw a Line

moveTo(x,y): Defines the starting point of the line.

lineTo(x,y): Defines the ending point of the line.

Draw a Circle

beginPath(): Starts the drawing.

arc(x,y,r,start,stop): Sets the parameters of the circle.

stroke(): Ends the drawing.

Gradients

createLinearGradient(x,y,x1,y1): Creates a linear gradient.

createRadialGradient(x,y,r,x1,y1,r1): Creates a radial/circular gradient.

Drawing Text on the Canvas

Font: Defines the font properties for the text.

fillText(text,x,y): Draws "filled" text on the canvas.

strokeText(text,x,y): Draws text on the canvas (no fill).

There are many other methods aimed at helping to draw shapes and images on the canvas.

Canvas vs. SVG

Canvas

- Elements are drawn programmatically
- Drawing is done with pixels
- Animations are not built in
- High performance for pixels-based drawing operations
- Resolution dependent
- No support for event handlers
- You can save the resulting image as .png or .jpg
- Well suited for graphic-intensive games

SVG

- Elements are part of the page's DOM (Document object model)
- Drawing is done with vectors
- Effects, such as animations are built in
- Based on standard XML syntax, which provides better accessibility
- Resolution independent
- Support for event handlers
- Not suited for game applications
- Best suited for applications with large rendering areas (for example, Google Maps)

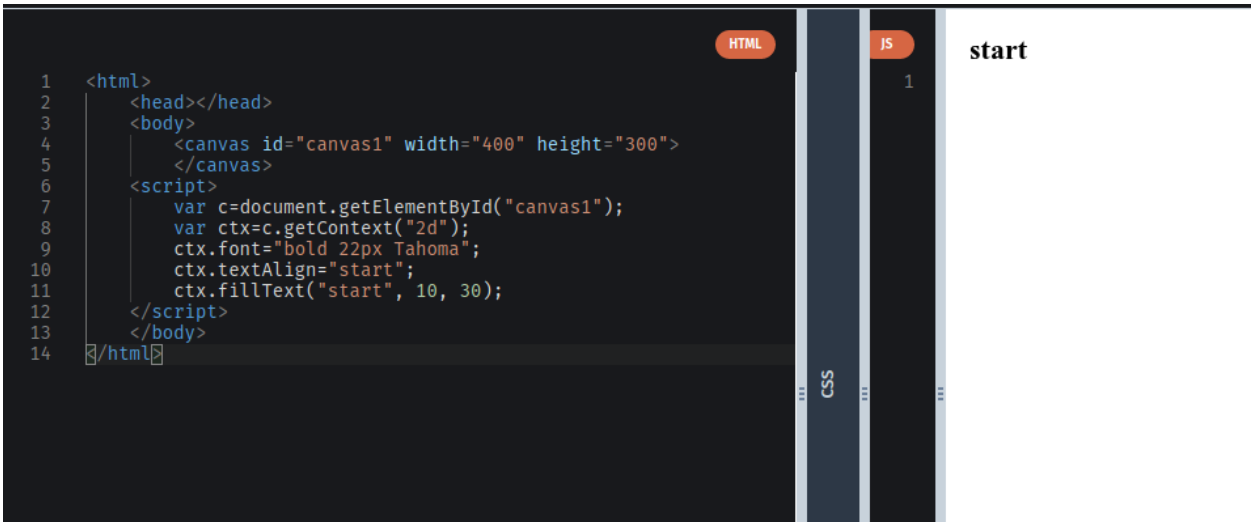
You can actually use both SVG and canvas on the same page, if needed.

However, you cannot just draw SVG onto a canvas, or vice-versa.

Working with Canvas

The Canvas element can be transformed. As an example, a text is written on the canvas at the coordinates (10, 30).

```
ctx.font="bold 22px Tahoma";

ctx.textAlign="start";

ctx.fillText("start", 10, 30);
```



The

translate(x,y)

method is used to move the Canvas.

x indicates how far to move the grid horizontally, and y indicates how far to move the grid vertically.

```
ctx.translate(100, 150);
ctx.fillText("after translate", 10, 30);
```

In this example, the canvas is moved 100px to the right, and 150px down.

Canvas has several methods for drawing paths, boxes, circles, text, and adding images.

The rotate() Method

The

rotate()

method is used to rotate the HTML5 Canvas. The value must be in radians, not degrees.

Here is an example that draws the same rectangle before and after rotation is set:

```
ctx.fillStyle = "#FF0000";

ctx.fillRect(10,10, 100, 100);

ctx.rotate( (Math.PI / 180) * 25);  //rotate 25 degrees.

ctx.fillStyle = "#0000FF";
ctx.fillRect(10,10, 100, 100);
```

The rotation will only affect drawings made after the rotation is done.

The scale() Method

The

scale()

method scales the current drawing. It takes two parameters:

- The number of times by which the image should be scaled in the X-direction.
- The number of times by which the image should be scaled in the Y-direction.

```
var canvas = document.getElementById('canvas1');

ctx =canvas.getContext('2d');

ctx.font="bold 22px Tahoma";

ctx.textAlign="start";

ctx.fillText("start", 10, 30);

ctx.translate(100, 150);

ctx.fillText("after translate", 0, 0);
```

ctx.rotate(1);

ctx.fillText("after rotate", 0, 0);

ctx.**scale(1.5, 4)**;
ctx.fillText("after scale", 0,20);

This will scale the canvas 1.5 times in the X-direction, and 4 times in Y-direction:

If you scale a drawing, all future drawings will also be scaled.

What method is used to increase or decrease the size of the current drawing?
scale

HTML5 Forms

HTML5 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.

New Attributes

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 for="email">Your e-mail address: </label>

  <input type="text" name="email" placeholder="email@example.com" />
</form>
```

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

```
<form>

  <label for="email">Your e-mail address: </label>

  <input type="text" name="email" autofocus/>
</form>
```

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

Forms with Required Fields

The "

required

" attribute is used to make the input elements required.

```
<form autocomplete="off">

  <label for="e-mail">Your e-mail address: </label>

  <input name="Email" type="text" required />

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

The

autocomplete

attribute specifies whether a form or 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 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 (regexp)
- placeholder
- required
- step

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

Creating a Search Box

The new

search

input type can be used to create a search box:

<input id="mysearch" name="searchitem" type="**search**" />

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

Search Options

The

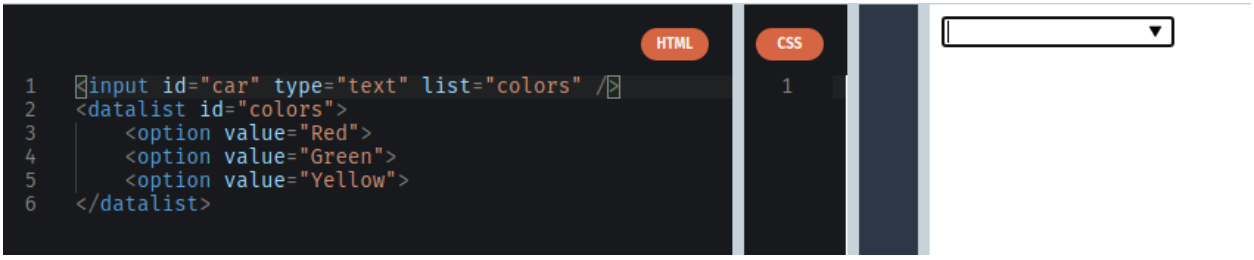
<datalist>

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>
```



<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.

Creating More Fields

Some other new input types include

email ,, **url** , and **tel**

```
<input id="email" name="email" type="email"
placeholder="example@example.com" />

<br />

<input id="url" name="url" type="url" placeholder="example.com" />
```

```
<br />
<input id="tel" name="tel" type="tel" placeholder="555.555.1211" />
```

These are especially useful when opening a page on a modern mobile device, which recognizes 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.