

## New Elements in HTML5

Semantics is the study of the meanings of words and phrases in language.

Semantic elements are elements with a meaning.

What are Semantic Elements?

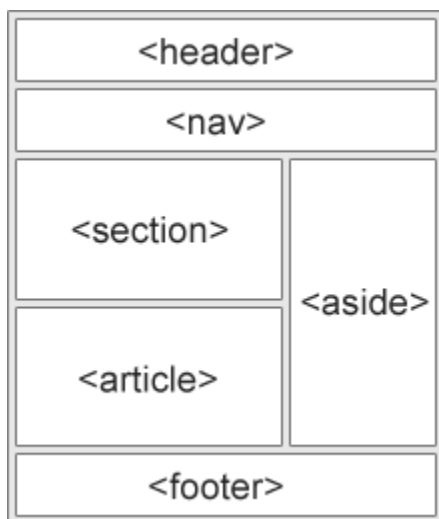
A semantic element clearly describes its meaning to both the browser and the developer.

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

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

## New Semantic Elements in HTML5

HTML5 offers new semantic elements to define different parts of a web page:



## HTML5 `<section>` Element

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

"A section is a thematic grouping of content, typically with a heading."

A Web site's home page could be split into sections for introduction, content, and contact information.

- Example

```
<section>
  <h1>WWF</h1>
  <p>The World Wide Fund for Nature (WWF) is....</p>
</section>
```

## HTML5 <article> Element

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

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

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

- Forum post
- Blog post
- Newspaper article

### Example

```
<article>
  <h1>What Does WWF Do?</h1>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

## HTML5 <header> Element

The <header> element specifies a header for a document or section.

The <header> element should be used as a container for introductory content.

You can have several <header> elements in one document.

The following example defines a header for an article:

- Example

```
<article>
  <header>
    <h1>What Does WWF Do?</h1>
    <p>WWF's mission:</p>
  </header>
  <p>WWF's mission is to stop the degradation of our planet's natural environment,
  and build a future in which humans live in harmony with nature.</p>
</article>
```

## HTML5 <footer> Element

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

A <footer> element should contain information about its containing element.

A footer typically contains the author of the document, copyright information, links to terms of use, contact information, etc.

You can have several <footer> elements in one document.

### Example

```
<footer>
  <p>Posted by: vinayak</p>
  <p>Contact information: <a href="mailto:someone@example.com">
  someone@example.com</a>.</p>
</footer>
```

## HTML5 <nav> Element

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

The <nav> element is intended for large blocks of navigation links. However, not all links in a document should be inside a <nav> element!

### Example

```
<nav>
  <a href="/html/">HTML</a> |
  <a href="/css/">CSS</a> |
  <a href="/js/">JavaScript</a> |
  <a href="/jquery/">jQuery</a>
</nav>
```

## HTML5 <aside> Element

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

The aside content should be related to the surrounding content.

- Example

```
<p>My family and I visited The Epcot center this summer.</p>
```

```
<aside>
  <h4>Epcot Center</h4>
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>
</aside>
```

## HTML5 <figure> and <figcaption> Elements

In books and newspapers, it is common to have captions with images.

The purpose of a caption is to add a visual explanation to an image.

With HTML5, images and captions can be grouped together in <figure> elements:

### Example

```
<figure>
  
  <figcaption>Fig1. - The Pulpit Rock, Norway.</figcaption>
</figure>
```

The <img> element defines the image, the <figcaption> element defines the caption.

## Why Semantic HTML5 Elements?

With HTML4, developers used their own favorite attribute names to style page elements:

header, top, bottom, footer, menu, navigation, main, container, content, article, sidebar, topnav, ...

This made it impossible for search engines to identify the correct web page content.

With HTML5 elements like: `<header>` `<footer>` `<nav>` `<section>` `<article>`, this will become easier.

According to the W3C, a Semantic Web:

"Allows data to be shared and reused across applications, enterprises, and communities."

## Semantic Elements in HTML5

**Below is an alphabetical list of the new semantic elements in HTML5.**

Tag	Description
<a href="#"><code>&lt;article&gt;</code></a>	Defines an article
<a href="#"><code>&lt;aside&gt;</code></a>	Defines content aside from the page content
<a href="#"><code>&lt;details&gt;</code></a>	Defines additional details that the user can view or hide
<a href="#"><code>&lt;figcaption&gt;</code></a>	Defines a caption for a <code>&lt;figure&gt;</code> element
<a href="#"><code>&lt;figure&gt;</code></a>	Specifies self-contained content, like illustrations, diagrams, photos, co
<a href="#"><code>&lt;footer&gt;</code></a>	Defines a footer for a document or section

<a href="#"><u>&lt;header&gt;</u></a>	Specifies a header for a document or section
<a href="#"><u>&lt;main&gt;</u></a>	Specifies the main content of a document
<a href="#"><u>&lt;mark&gt;</u></a>	Defines marked/highlighted text
<a href="#"><u>&lt;nav&gt;</u></a>	Defines navigation links
<a href="#"><u>&lt;section&gt;</u></a>	Defines a section in a document
<a href="#"><u>&lt;summary&gt;</u></a>	Defines a visible heading for a <details> element
<a href="#"><u>&lt;time&gt;</u></a>	Defines a date/time

## New Semantic/Structural Elements

### HTML5 offers new elements for better document structure:

Tag	Description
<article>	Defines an article in the document
<aside>	Defines content aside from the page content
<bdi>	Defines a part of text that might be formatted in a different direction from oth
<details>	Defines additional details that the user can view or hide
<dialog>	Defines a dialog box or window

<figcaption>	Defines a caption for a <figure> element
<figure>	Defines self-contained content, like illustrations, diagrams, photos, code listing
<footer>	Defines a footer for the document or a section
<header>	Defines a header for the document or a section
<main>	Defines the main content of a document
<mark>	Defines marked or highlighted text
<menuitem>	Defines a command/menu item that the user can invoke from a popup menu
<meter>	Defines a scalar measurement within a known range (a gauge)
<nav>	Defines navigation links in the document
<progress>	Defines the progress of a task
<rp>	Defines what to show in browsers that do not support ruby annotations
<rt>	Defines an explanation/pronunciation of characters (for East Asian typography)
<ruby>	Defines a ruby annotation (for East Asian typography)
<section>	Defines a section in the document
<summary>	Defines a visible heading for a <details> element

<time>	Defines a date/time
--------	---------------------

<wbr>	Defines a possible line-break
-------	-------------------------------

## New Form Elements

Tag	Description
<datalist>	Defines pre-defined options for input controls
<keygen>	Defines a key-pair generator field (for forms)
<output>	Defines the result of a calculation

## HTML5 <datalist> Element

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

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

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

### Example

An **<input>** element with pre-defined values in a **<datalist>**:

```
<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>
```

## HTML5 <keygen> Element

The purpose of the <keygen> element is to provide a secure way to authenticate users.

The <keygen> element specifies a key-pair generator field in a form.

When the form is submitted, two keys are generated, one private and one public.

The private key is stored locally, and the public key is sent to the server.

The public key could be used to generate a client certificate to authenticate the user in the future.

### Example

#### A form with a keygen field:

```
<form action="action_page.php">
  Username: <input type="text" name="user">
  Encryption: <keygen name="security">
  <input type="submit">
</form>
```

## HTML5 <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>
```

## New Input Types

### New Input Types

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

### New Input Attributes

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max
- multiple

- week
- pattern (regexp)
- placeholder
- required
- step

## Input Type: number

The `<input type="number">` is used for input fields that should contain a numeric value.

You can set restrictions on the numbers.

Depending on browser support, the restrictions can apply to the input field.

### Example

```
<form>
```

Quantity (between 1 and 5):

```
<input type="number" name="quantity" min="1" max="5">
</form>
```

## Input Restrictions

Here is a list of some common input restrictions (some are new in HTML5):

Attribute	Description
disabled	Specifies that an input field should be disabled
max	Specifies the maximum value for an input field
maxlength	Specifies the maximum number of character for an input field
min	Specifies the minimum value for an input field

pattern	Specifies a regular expression to check the input value against
readonly	Specifies that an input field is read only (cannot be changed)
required	Specifies that an input field is required (must be filled out)
size	Specifies the width (in characters) of an input field
step	Specifies the legal number intervals for an input field
value	Specifies the default value for an input field

#### Example

```
<form>  
  Quantity:  
  <input type="number" name="points" min="0" max="100" step="10" value="30"  
>  
</form>
```

## Input Type: date

The **<input type="date">** is used for input fields that should contain a date.

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

#### Example

```
<form>  
  Birthday:  
  <input type="date" name="bday">  
</form>
```

You can add restrictions to the input:

#### Example

```
<form>
  Enter a date before 1980-01-01:
  <input type="date" name="bday" max="1979-12-31"><br>
  Enter a date after 2000-01-01:
  <input type="date" name="bday" min="2000-01-02"><br>
</form>
```

## Input Type: color

The `<input type="color">` is used for input fields that should contain a color.

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

### Example

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

## Input Type: range

The `<input type="range">` is used for input fields that should contain a value within a range.

Depending on browser support, the input field can be displayed as a slider control.

### Example

```
<form>
  <input type="range" name="points" min="0" max="10">
</form>
```

You can use the following attributes to specify restrictions: min, max, step, value.

## Input Type: month

The `<input type="month">` allows the user to select a month and year.

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

#### Example

```
<form>  
  Birthday (month and year):  
  <input type="month" name="bdaymonth">  
</form>
```

## Input Type: week

The `<input type="week">` allows the user to select a week and year.

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

#### Example

```
<form>  
  Select a week:  
  <input type="week" name="week_year">  
</form>
```

## Input Type: time

The `<input type="time">` allows the user to select a time (no time zone).

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

#### Example

```
<form>  
  Select a time:  
  <input type="time" name="usr_time">  
</form>
```

## Input Type: datetime

The `<input type="datetime">` allows the user to select a date and time (with time zone).

### Example

```
<form>  
  Birthday (date and time):  
  <input type="datetime" name="bdaytime">  
</form>
```

The input type datetime is removed from the HTML standard. Use datetime-local instead.

## Input Type: datetime-local

The `<input type="datetime-local">` allows the user to select a date and time (no time zone).

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

### Example

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

## Input Type: email

The `<input type="email">` 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 adds ".com" to the keyboard to match email input.

#### Example

```
<form>  
  E-mail:  
  <input type="email" name="email">  
</form>
```

## Input Type: search

The **<input type="search">** is used for search fields (a search field behaves like a regular text field).

#### Example

```
<form>  
  Search Google:  
  <input type="search" name="googlesearch">  
</form>
```

## Input Type: tel

The **<input type="tel">** is used for input fields that should contain a telephone number.

The tel type is currently supported only in Safari 8.

#### Example

```
<form>  
  Telephone:  
  <input type="tel" name="usrtel">  
</form>
```



## Input Type: url

The `<input type="url">` 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.

### Example

```
<form>
```

Add your homepage:

```
<input type="url" name="homepage">
```

```
</form>
```

## HTML5 Attributes

HTML5 added the following attributes for `<input>`:

- autocomplete
- autofocus
- form
- formaction
- formenctype
- formmethod
- formnovalidate
- formtarget
- height and width
- list
- min and max

- multiple
- pattern (regexp)
- placeholder
- required
- step

and the following attributes for <form>:

- autocomplete
- novalidate

## The autocomplete Attribute

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

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

**Tip:** It is possible to have autocomplete "on" for the form, and "off" for specific input fields, or vice versa.

The autocomplete attribute works with <form> and the following <input> types: text, search, url, tel, email, password, datepickers, range, and color.

### Example

An HTML form with autocomplete on (and off for one input field):

```
<form action="action_page.php" autocomplete="on">  
  First name:<input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  E-mail: <input type="email" name="email" autocomplete="off"><br>  
  <input type="submit">  
</form>
```

**Tip:** In some browsers you may need to activate the autocomplete function for this to work.

## The novalidate Attribute

The novalidate attribute is a <form> attribute.

When present, novalidate specifies that form data should not be validated when submitted.

### Example

Indicates that the form is not to be validated on submit:

```
<form action="action_page.php" novalidate>  
  E-mail: <input type="email" name="user_email">  
  <input type="submit">  
</form>
```

## The autofocus Attribute

The autofocus attribute is a boolean attribute.

When present, it specifies that an <input> element should automatically get focus when the page loads.

### Example

Let the "First name" input field automatically get focus when the page loads:

First name:<input type="text" name="fname" autofocus>

## The form Attribute

The form attribute specifies one or more forms an <input> element belongs to.

**Tip:** To refer to more than one form, use a space-separated list of form ids.

### Example

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

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

Last name: <input type="text" name="lname" form="form1">

## The formation Attribute

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

The formation attribute overrides the action attribute of the <form> element.

The formation attribute is used with type="submit" and type="image".

### Example

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

```
<form action="action_page.php">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit"><br>  
  <input type="submit" formation="demo_admin.asp"  
  value="Submit as admin">  
</form>
```

## The formenctype Attribute

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

The formenctype attribute overrides the enctype attribute of the <form> element.

The formenctype attribute is used with type="submit" and type="image".

### Example

Send form-data that is default encoded (the first submit button), and encoded as "multipart/form-data" (the second submit button):

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

## The formmethod Attribute

The formmethod attribute defines the HTTP method for sending form-data to the action URL.

The formmethod attribute overrides the method attribute of the <form> element.

The formmethod attribute can be used with type="submit" and type="image".

### Example

The second submit button overrides the HTTP method of the form:

```
<form action="action_page.php" method="get">  
  First name: <input type="text" name="fname"><br>  
  Last name: <input type="text" name="lname"><br>  
  <input type="submit" value="Submit">  
  <input type="submit" formmethod="post" formaction="demo_post.asp"  
  value="Submit using POST">
```

`</form>`

## The formnovalidate Attribute

The novalidate attribute is a boolean attribute.

When present, it specifies that the `<input>` element should not be validated when submitted.

The formnovalidate attribute overrides the novalidate attribute of the `<form>` element.

The formnovalidate attribute can be used with `type="submit"`.

### Example

A form with two submit buttons (with and without validation):

```
<form action="action_page.php">  
  E-mail: <input type="email" name="userid"><br>  
  <input type="submit" value="Submit"><br>  
  <input type="submit" formnovalidate value="Submit without validation">  
</form>
```

## The formtarget Attribute

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

The formtarget attribute overrides the target attribute of the `<form>` element.

The formtarget attribute can be used with `type="submit"` and `type="image"`.

### Example

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

```
<form action="action_page.php">
```

```
First name: <input type="text" name="fname"><br>
Last name: <input type="text" name="lname"><br>
<input type="submit" value="Submit as normal">
<input type="submit" formtarget="_blank"
value="Submit to a new window">
</form>
```

## The height and width Attributes

The height and width attributes specify the height and width of an <input> element.

The height and width attributes are only used with <input type="image">.

Always specify the size of images. If the browser does not know the size, the page will flicker while images load.

### Example

Define an image as the submit button, with height and width attributes:

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

## The list Attribute

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

### Example

An <input> element with pre-defined values in a <datalist>:

```
<input list="browsers">

<datalist id="browsers">
  <option value="Internet Explorer">
```

```
<option value="Firefox">  
<option value="Chrome">  
<option value="Opera">  
<option value="Safari">  
</datalist>
```

## The min and max Attributes

The min and max attributes specify the minimum and maximum value for an `<input>` element.

The min and max attributes work with the following input types: number, range, date, datetime, datetime-local, month, time and week.

### Example

`<input>` elements with min and max values:

Enter a date before 1980-01-01:

```
<input type="date" name="bday" max="1979-12-31">
```

Enter a date after 2000-01-01:

```
<input type="date" name="bday" min="2000-01-02">
```

Quantity (between 1 and 5):

```
<input type="number" name="quantity" min="1" max="5">
```

## The multiple Attribute

The multiple attribute is a boolean attribute.

When present, it specifies that the user is allowed to enter more than one value in the `<input>` element.

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



## Example

A file upload field that accepts multiple values:

Select images: `<input type="file" name="img" multiple>`

## The pattern Attribute

The pattern attribute specifies a regular expression that the `<input>` element's value is checked against.

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

**Tip:** Use the global [title](#) attribute to describe the pattern to help the user.

**Tip:** Learn more about [regular expressions](#) in our JavaScript tutorial.

## Example

An input field that can contain only three letters (no numbers or special characters):

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

## The placeholder Attribute

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

The 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.

## Example

An input field with a placeholder text:

```
<input type="text" name="fname" placeholder="First name">
```

## The required Attribute

The required attribute is a boolean attribute.

When present, it 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.

### Example

A required input field:

Username: `<input type="text" name="username" required>`

## The step Attribute

The step attribute specifies the legal number intervals for an `<input>` element.

Example: if `step="3"`, legal numbers could be -3, 0, 3, 6, etc.

**Tip:** The step attribute can be used together with the max and min attributes to create a range of legal values.

The step attribute works with the following input types: number, range, date, datetime, datetime-local, month, time and week.

### Example

An input field with a specified legal number intervals:

```
<input type="number" name="points" step="3">
```

## HTML5 - New Attribute Syntax

HTML5 allows four different syntaxes for attributes.

This example demonstrates the different syntaxes used in an `<input>` tag:

Type	Example
Empty	<code>&lt;input type="text" value="John" disabled&gt;</code>
Unquoted	<code>&lt;input type="text" value=John&gt;</code>
Double-quoted	<code>&lt;input type="text" value="John Doe"&gt;</code>
Single-quoted	<code>&lt;input type="text" value='John Doe'&gt;</code>

In HTML5, all four syntaxes may be used, depending on what is needed for the attribute.

## HTML5 Graphics

Tag	Description
<code>&lt;canvas&gt;</code>	Defines graphic drawing using JavaScript
<code>&lt;svg&gt;</code>	Defines graphic drawing using SVG

What is HTML Canvas?

The HTML `<canvas>` element is used to draw graphics, on the fly, via scripting (usually JavaScript).

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

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

## 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:

### Basic Canvas Example

```
<canvas id="myCanvas" width="200" height="100" style="border: 1px solid  
#000000;">  
</canvas>
```

### Drawing with JavaScript

```
var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.fillStyle = "#FF0000";  
ctx.fillRect(0,0,150,75);
```

### Draw a Line

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

### Draw a Circle

```
var c = document.getElementById("myCanvas");
```

```
var ctx = c.getContext("2d");  
ctx.beginPath();  
ctx.arc(95,50,40,0,2*Math.PI);  
ctx.stroke();
```

### Draw a Text

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

### Stroke Text

```
var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
ctx.font = "30px Arial";  
ctx.strokeText("Hello World",10,50);
```

### Draw Linear Gradient

```
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);
```

### Draw Circular Gradient

```
var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");
```

```
// Create gradient
```

```
var grd = ctx.createRadialGradient(75,50,5,90,60,100);  
grd.addColorStop(0,"red");  
grd.addColorStop(1,"white");
```

```
// Fill with gradient
```

```
ctx.fillStyle = grd;  
ctx.fillRect(10,10,150,80);
```

### Draw Image

```
var c = document.getElementById("myCanvas");  
var ctx = c.getContext("2d");  
var img = document.getElementById("scream");  
ctx.drawImage(img,10,10);
```

## What is SVG?

- SVG stands for Scalable Vector Graphics
- SVG is used to define graphics for the Web
- SVG is a W3C recommendation

### The HTML <svg> Element

The HTML <svg> element (introduced in HTML5) is a container for SVG graphics.

SVG has several methods for drawing paths, boxes, circles, text, and graphic images.

### SVG Circle

#### Example

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

```
<body>

<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-
width="4" fill="yellow" />
</svg>

</body>
</html>
```

## SVG Rectangle

### Example

```
<svg width="400" height="100">
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
width:10;stroke:rgb(0,0,0)" />
</svg>
```

## SVG Rounded Rectangle

### Example

```
<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" />
</svg>
```

## SVG Star

### Example

```
<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;" />
</svg>
```

## SVG Logo

### Example

```
<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>
```

<path> element is used to draw a connected straight lines.

## Declaration

Following is the syntax declaration of **<path>** element. We've shown main attributes only.

```
<path
```



```
d="data" >
</path>
```

## Attributes

Sr.No.	Name&Description
1	<b>d</b> – path data, usually a set of commands like moveto, lineto etc.

<path> element is used to define any path. Path element uses Path data which comprises of number of commands. Commands behaves like a nip of pencil or a pointer is moving to draw a path.

Sr.No.	Command & Description
1	<b>M</b> – moveto – move from one point to another point.
2	<b>L</b> – lineto – create a line.
3	<b>H</b> – horizontal lineto – create a horizontal line.
4	<b>V</b> – vertical lineto – create a vertical line.
5	<b>C</b> – curveto – create a curve.
6	<b>S</b> – smooth curveto – create a smooth curve.
7	<b>Q</b> – quadratic Bezier curve – create a quadratic Bezier curve.
8	<b>T</b> – smooth quadratic Bezier curveto – create a smooth quadratic Bezier curve
9	<b>A</b> – elliptical Arc – create a elliptical arc.

10	<b>Z</b> – closepath – close the path.
----	--

As above commands are in Upper case, these represents absolute path. In case their lower case commands are used, then relative path is used.

### Example

*testSVG.htm*

```
<html>
  <title>SVG Path</title>
  <body>

    <h1>Sample SVG Path Image</h1>

    <svg width="570" height="320">
      <g>
        <text x="0" y="10" fill="black" >Path #1: Without
opacity.</text>

        <path d="M 100 100 L 300 100 L 200 300 z"
stroke="black" stroke-width="3" fill="rgb(121,0,121)">
</path>
      </g>
    </svg>

  </body>
</html>
```

In above example, in first shape, M 100 100 moves drawing pointer to (100,100), L 300 100 draws a line from (100,100) to (300,100), L 200 300 draws a line from (300,100) to (200,300) and z closes the path.

## Output

Open textSVG.htm in Chrome web browser. You can use Chrome/Firefox/Opera to view SVG image directly without any plugin. Internet Explorer 9 and higher also supports SVG image rendering.

## Path with opacity

```
<html>
  <title>SVG Path</title>
  <body>
```

```
<h1>Sample SVG Path Image</h1>

<svg width="800" height="800">
  <g>
    <text x="0" y="15" fill="black" >Path #2: With opacity
  </text>

    <path d="M 100 100 L 300 100 L 200 300 z"
    style="fill:rgb(121,0,121);stroke-width:3;
    stroke:rgb(0,0,0);stroke-opacity:0.5;"> </path>
  </g>
</svg>

</body>
</html>
```

## Output

Open textSVG.htm in Chrome web browser. You can use Chrome/Firefox/Opera to view SVG image directly without any plugin. Internet Explorer 9 and higher also supports SVG image rendering.

## Differences Between SVG and Canvas

- SVG is a language for describing 2D graphics in XML.
- Canvas draws 2D graphics, on the fly (with a JavaScript).
- SVG is XML based, which means that every element is available within the SVG DOM. You can attach JavaScript event handlers for an element.
- In SVG, each drawn shape is remembered as an object. If attributes of an SVG object are changed, the browser can automatically re-render the shape.
- Canvas is rendered pixel by pixel. In canvas, once the graphic is drawn, it is forgotten by the browser. If its position should be changed, the entire scene needs to be redrawn, including any objects that might have been covered by the graphic.

## Comparison of Canvas and SVG

The table below shows some important differences between Canvas and SVG:

Canvas	SVG
<ul style="list-style-type: none"> <li>• Resolution dependent</li> <li>• No support for event handlers</li> <li>• Poor text rendering capabilities</li> <li>• You can save the resulting image as .png or .jpg</li> <li>• Well suited for graphic-intensive games</li> </ul>	<ul style="list-style-type: none"> <li>• Resolution independent</li> <li>• Support for event handlers</li> <li>• Best suited for applications with large Maps)</li> <li>• Slow rendering if complex (anything will be slow)</li> <li>• Not suited for game applications</li> </ul>

## New Media Elements

Tag	Description
<audio>	Defines sound or music content
<embed>	Defines containers for external applications (like plug-ins)
<source>	Defines sources for <video> and <audio>
<track>	Defines tracks for <video> and <audio>
<video>	Defines video or movie content

## 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>
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<div style="text-align:center">
```

```
  <button onclick="playPause()">Play/Pause</button>
```

```
  <button onclick="makeBig()">Big</button>
```

```
  <button onclick="makeSmall()">Small</button>
```

```
  <button onclick="makeNormal()">Normal</button>
```

```
  <br><br>
```

```
  <video id="video1" width="420">
```

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

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

```
    Your browser does not support HTML5 video.
```

```
  </video>
```

```
</div>
```

```
<script>
```

```
var myVideo = document.getElementById("video1");
```

```
function playPause() {  
    if (myVideo.paused)  
        myVideo.play();  
    else  
        myVideo.pause();  
}
```

```
function makeBig() {  
    myVideo.width = 560;  
}
```

```
function makeSmall() {  
    myVideo.width = 320;  
}
```

```
function makeNormal() {  
    myVideo.width = 420;  
}
```







```
</script>
```

```
<p>Video courtesy of <a href="http://www.bigbuckbunny.org/"  
target="_blank">Big Buck Bunny</a>.</p>
```

```
</body>
```

```
</html>
```

## Version Compatibility for HTML 5

						
Browser Support	IE 11 onwards	Firefox 90-91 onwards	Edge 91 onwards	Edge 92-94 onwards	Opera-76 onwards	Safari 15-TP onwards
	Complete support	Partial support	Partial support	Complete support	Complete support	Partial support
OS Support	Windows✓	Windows✓	Windows✓	Windows✓	Windows✓	
	Android✓				Android✓	Android✓
	Mac✓					Mac✓
	Linux✓	Linux✓				

## Basic example

To get a `CanvasRenderingContext2D` instance, you must first have an HTML `<canvas>` element to work with:

```
<canvas id="my-house" width="300" height="300"></canvas>
```

Copy to Clipboard

To get the canvas' 2D rendering context, call `getContext()` on the `<canvas>` element, supplying `'2d'` as the argument:



```
const canvas = document.getElementById('my-house');  
  
const ctx = canvas.getContext('2d');
```

### Copy to Clipboard

With the context in hand, you can draw anything you like. This code draws a house:

```
// Set line width  
  
ctx.lineWidth = 10;  
  
// Wall  
  
ctx.strokeRect(75, 140, 150, 110);  
  
// Door  
  
ctx.fillRect(130, 190, 40, 60);  
  
// Roof  
  
ctx.beginPath();  
  
ctx.moveTo(50, 140);  
  
ctx.lineTo(150, 60);  
  
ctx.lineTo(250, 140);  
  
ctx.closePath();  
  
ctx.stroke();
```

The resulting drawing looks like this:

