The for attribute is used in labels. It refers to the id of the element this label is associated with.

For example:

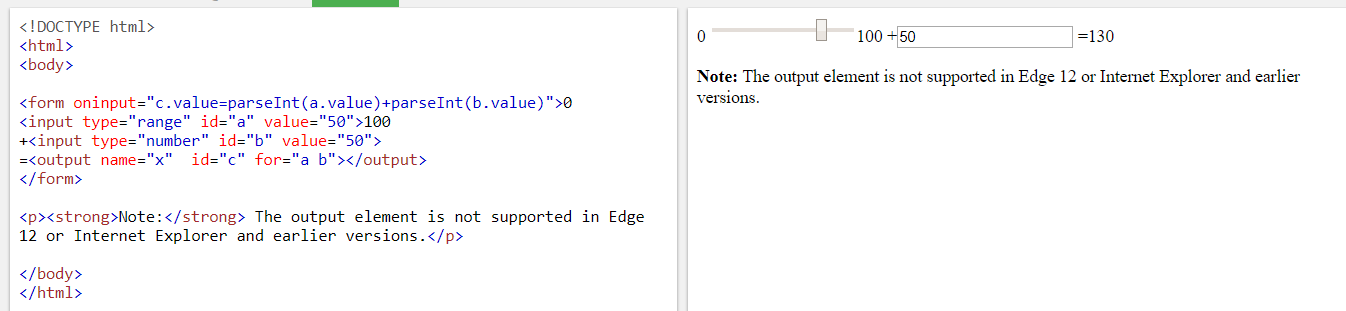
<label for="username">Username</label>

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

Now when the user clicks with the mouse on the username text the browser will automatically put the focus in the corresponding input field. This also works with other input elements such as <textbox> and <select>.

Quote from the [specification](http://www.w3.org/TR/html401/interact/forms.html#h-17.9.1):

This attribute explicitly associates the label being defined with another control. When present, the value of this attribute must be the same as the value of the id attribute of some other control in the same document. When absent, the label being defined is associated with the element's contents.

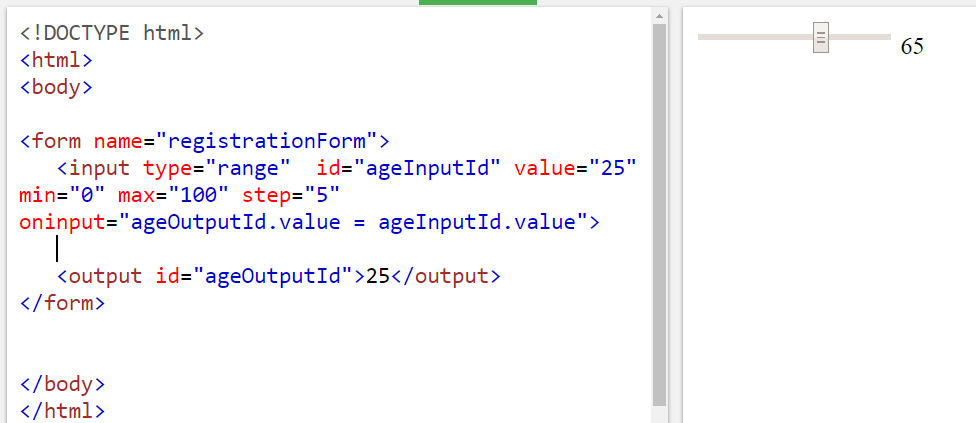


Id vs name attributes in HTML

1. The difference is that a "name" transfers from the browser to the server and can be different than the "id".
2. To put it very informally, id is what your frontend (CSS, JS) works with, while name is what your server receives and can then process.
3. It might be better to say: *The name attribute is****required*** when sending data... instead of: *The name attribute is****used****when sending data...* since any form data missing the **name** attribute will not be transmitted (or indeed will not be processed at all according to the HTML spec)
4. id is used to **identify the HTML element through the Document Object Model** (via JavaScript or styled with CSS). id is expected to be unique within the page.

name corresponds to the form element and **identifies what is posted back to the server**.

## <input type=”range”>

<form name="registrationForm">

<input type="range" id="ageInputId" value="25" min="0" max="100" step="5" oninput="ageOutputId.value = ageInputId.value">

<output id="ageOutputId">25</output>

</form>

## <pre> tag

The <pre> tag defines preformatted text.

Text in a <pre> element is displayed in a fixed-width font (usually Courier), and it preserves both spaces and line breaks.

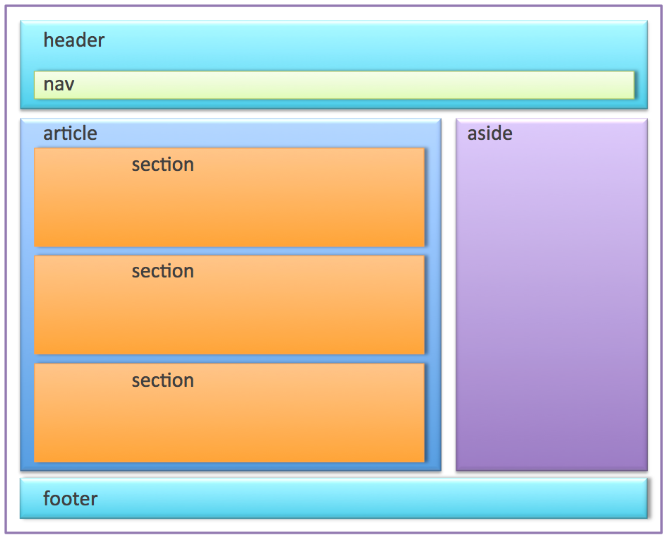


Not supported in HTML5.

1. Are HTML tags case sensitive?

Short and Sweet Answer : “NO”.

1. HTML5 semantic elements: <header>, <article>, <section>, <footer>.



**Header and footer elements**

 Each article could have it’s own header and footer. The header could contain the title and author of the article, the footer could contain links related to the article, contact info for the author, or something else. In this case, the header would be the header of the parent article element rather than the header of the whole page.

Therefore, several header and footer elements can co-exist in a page, each related to it’s parent element.

## Article

Section

## Aside

“The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography. The element can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page.”

**Nav**

Nav elements generally contain website **nav**igation links. These links are typically structured in the form of an unordered list (ul).

<nav>

    <h1>Navigation Menu</h1>

    <ul>

        <li><a href="page1.html">page1</a></li>

        <li><a href="page2.html">page2</a></li>

        <li><a href="page3.html">page3</a></li>

    </ul>

</nav>

**Sectioning elements**

In HTML5, article, section, aside and nav are known as **sectioning elements**. Note that header and footer do not belong to this category.

Can a <section> contain <article> elements? Can an <article> contain <section>elements? Provide usage examples.

The answer to both questions is yes; i.e., a <section> can contain <article> elements, and an <article> can contain <section>elements.

For example, a personal dashboard page might contain a <section> for social network interactions as well as a <section> for the latest news articles, the latter of which could contain several <article> elements.

Conversely, an <article> might contain a <section> at the end for reader comments.

Can a web page contain multiple <header> elements? What about <footer> elements?

Yes to both. In fact, both the <header> and <footer> tags are designed to serve their respective purposes in relation to whatever their parent “section” may be. So not only can the page <body> contain a header and a footer, but so can every <article> and <section> element. In fact, a <header> should be present for all of these, although a <footer> is not always necessary.

Describe the relationship between the <header> and <h1> tags in HTML5.

In previous specifications of HTML, only one <h1> element was typically present on a page, used for the heading of the entire page. HTML5 specifies that <h1> represents the top-level heading of a “section”, whether that be the page <body>, or an <article> or <section> element. In fact, every <header> element should at least contain an <h1> element. If there is no natural heading for the section, it is a good indication it should not use an <article> or <section> tag.

Give a simple implementation of the <video> tag to embed a video stored at http://www.example.com/amazing\_video.mp4. Give the video a width of 640 pixels by 360 pixels. Provide the user with controls.

Here is one simple implementation:

<video src="http://www.example.com/amazing\_video.mp4" width="640" height="360" controls></video>

Alternatively, the source file may be indicated with a separate <source> tag inside the <video> element, as in:

<video width="640" height="360" controls>

<source src="http://www.example.com/amazing\_video.mp4">

</video>

Write the code necessary to create a 300 pixel by 300 pixel <canvas>. Within it, paint a blue 100 pixel by 100 pixel square with the top-left corner of the square located 50 pixels from both the top and left edges of the canvas.

Here is one simple implementation:

<canvas id="c" width="300" height="300"></canvas>

<script>

var canvas = document.getElementById( "c" );

var drawing\_context = canvas.getContext( "2d" );

drawing\_context.fillStyle = "blue";

drawing\_context.fillRect( 50, 50, 100, 100 );

</script>

What is HTML5 Web Storage? Explain localStorage and sessionStorage. With HTML5, web pages can store data locally within the user’s browser.

Earlier, this was done with cookies. However, Web Storage is more secure and faster. The data is not included with every server request, but used ONLY when asked for.

The data is stored in name/value pairs, and a web page can only access data stored by itself. Unlike cookies, the storage limit is far larger (at least 5MB) and information is never transferred to the server.

The difference between localStorage and sessionStorage involves the lifetime and scope of the storage.

Data stored through localStorage is permanent: it does not expire and remains stored on the user’s computer until a web app deletes it or the user asks the browser to delete it. SessionStorage has the same lifetime as the top-level window or browser tab in which the script that stored it is running. When the window or tab is permanently closed, any data stored through sessionStorage is deleted.

Both forms of storage are scoped to the document origin so that documents with different origins will never share the stored objects. But sessionStorage is also scoped on a per-window basis. If a user has two browser tabs displaying documents from the same origin, those two tabs have separate sessionStorage data: the scripts running in one tab cannot read or overwrite the data written by scripts in the other tab, even if both tabs are visiting exactly the same page and are running exactly the same scripts.

What is the difference between span and div?

div is a block element, span is inline.

This means that to use them semantically, divs should be used to wrap sections of a document, while spans should be used to wrap small portions of text, images, etc.

For example:

<div>This a large main division, with <span>a small bit</span> of spanned text!</div>

Note that it is illegal to place a block level element within an inline element, so:

<div>Some <span>text that <div>I want</div> to mark</span> up</div>

...is illegal.

What is the Geolocation API in HTML5?

HTML5’s Geolocation API lets users share their physical location with chosen web sites. JavaScript can capture a user’s latitude and longitude and can send it to the back-end web server to enable location-aware features like finding local businesses or showing their location on a map.

Today, most browsers and mobile devices support the Geolocation API. The Geolocation API works with a new property of the global navigator object.

A Geolocation object can be created as follows:

var geolocation = navigator.geolocation;

The geolocation object is a service object that allows widgets to retrieve information about the geographic location of the user’s device.

What’s one main result if you do not specify a doctype in an HTML page?

New HTML5-specific tags will not be interpreted by the browser.

What’s the difference between the <svg> and <canvas> elements?

SVG is like a "draw" program. The drawing is specified as drawing instructions for each shape and any part of any shape can be changed. Drawings are shape-oriented.

Canvas is like a "paint" program. Once the pixels hit the screen, that is your drawing. You cannot change shapes except by overwriting them with other pixels. Paintings are pixel-oriented.

Being able to change drawings is very important for some programs; e.g. drafting apps, diagramming tools, etc. So SVG has an advantage here.

Being able to control individual pixels is important for some artistic programs.

Getting great animation performance for user-manipulation via mouse drags is easier with Canvas than SVG.

A single pixel on the computer screen will often consume 4 bytes of information and a computer screen these days takes several megabytes. So Canvas might be inconvenient if you want to let the user edit an image and then upload it again.

By contrast, drawing a handful of shapes that cover the entire screen using SVG takes up few bytes, downloads quickly, and can be uploaded again easily with the same advantages going in that direction as when it comes down on the other direction. So SVG can be faster than Canvas.

Google implemented Google Maps with SVG. That gives the web app its zippy performance and smooth scrolling.

———————————————————————————————-

Now if i will not explain then it might be possible that my answer will be collapsed.

So, Follow below steps for example:

<html>

<body>

<h1>Hello , I think you will upvote my answer.</h1>

</body>

</html>

copy code in notepad and save as “first.html”

Open the file in your browser.

Now Copy below code :

<HTML>

<BODY>

<H1>Hello, I Think you will upvote my answer.</H1>

</BODY>

</HTML>

4. Save this code in notepad and save as “second.html”

5. Open the file in you browser.

Now, you can see both gives exactly same result.

Hence proved!!

1. Saving html file naming convention – index.html file is picked up by the browser automatically
2. HTML colors: can be specified in three ways – 1-colour name, 2-hexadecimal value (starts with ‘#’ and contains 6 letters [a-f] and [0-9]) 3- RGB code (e.g. rgb(0, 102,153)).
3. Anchor tag for site navigation: <a href =”page2.html”>Click here to go to Page 2</a>.
4. Target attribute with anchor tag,

target=”\_blank” (whenever you click on the link a new tab opens up => no. of clicks = no. of new tabs) ,

target=”new” (when you first click on the link a new tab opens up => on second click same new tab will not open up)

1. Img tag: <img src=”image/pic.jpg” height=”150” width=”150”/> {no end tag required }.
2. Link styles:

<head>

<style>

a: link { color: pink; text-decoration: none // for removing under line }

a: visited {backgorund-color: yellow}

a: hover

a: active // rule: this order is necessary to be maintained

</style>

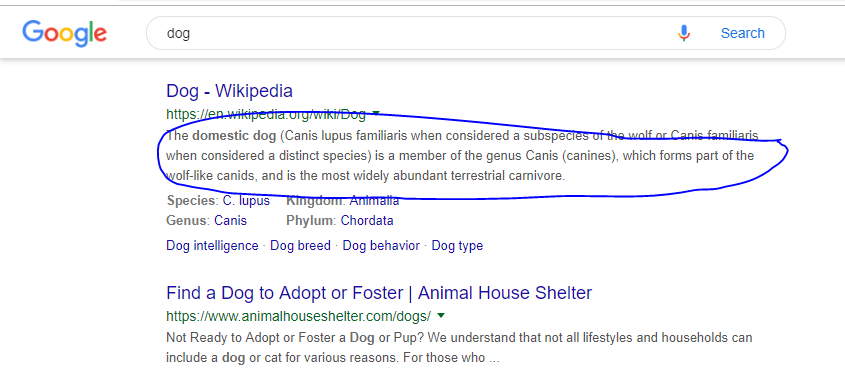
</head>

<body>

<a href=”page3.html” >Click here to navigate </a>

</body>

1. Comments tag: anything inside comment tag is not seen by the browser. /\* .. \*/
2. Difference between head and body tag.
3. Meta tag:

<head>  
  <meta charset="UTF-8">  
  <meta name="description" content="Free Web tutorials"> // is visible when we search on google   
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">  
  <meta name="author" content="John Doe">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

(description, keyword, author, viewport)

<meta http-equiv="refresh" content="30"> //**Refresh document every 30 seconds**  
</head>

1. Audio tag

<audio controls>

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

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

Your browser does not support the audio element.

</audio>

1. Video tag

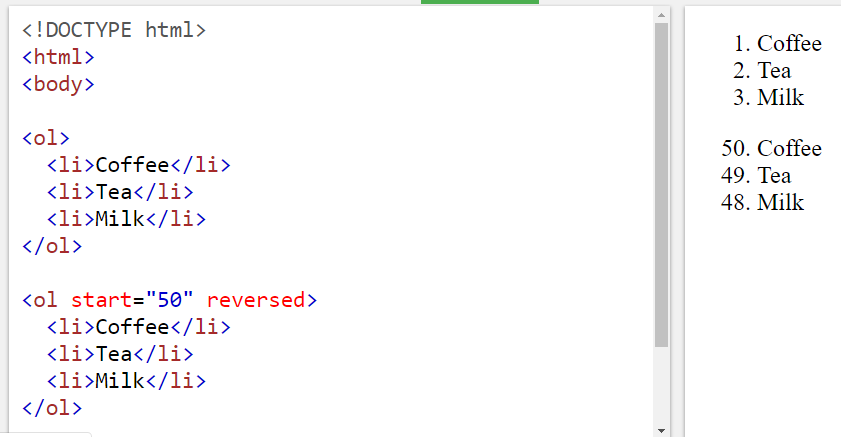
<video width="320" height="240" controls>

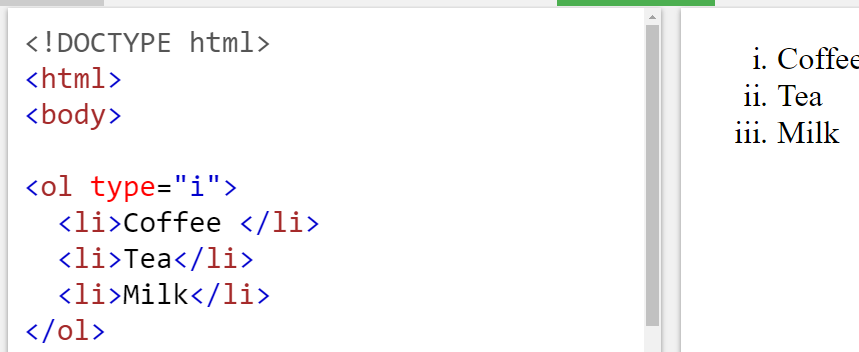
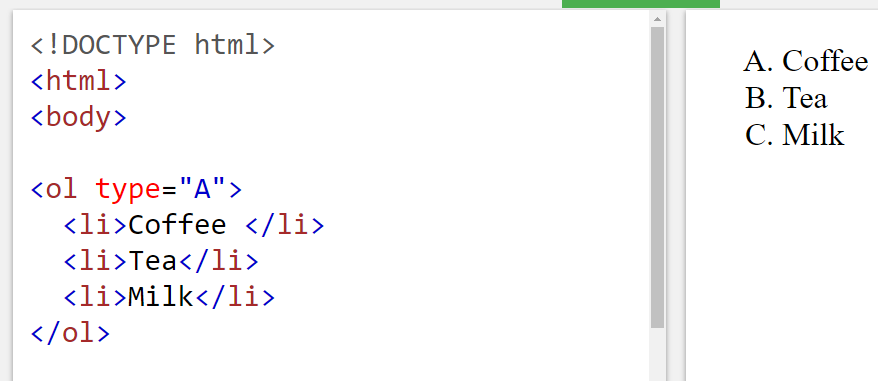
<source src="movie.mp4" type="video/mp4">

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

Your browser does not support the video tag.

</video>

1. Ordered list: 

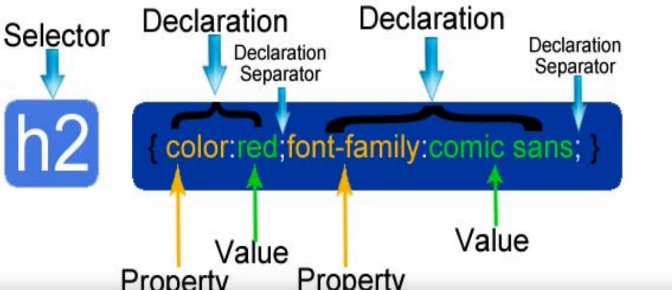


1. horizontal rule <hr> tag ,  tag has no end tag

CSS

1. All CSS syntax is case-insensitive within the ASCII range (i.e., [a-z] and [A-Z] are equivalent), except for parts that are not under the control of CSS.
2. Inline CSS - <h1 **style**="color:blue;"> - it is placed inside html tag
3. Internal CSS – In head section – inside <style> tags





1. External CSS:

**<head>**

**<link** rel="stylesheet" type="text/css" href="mystyle.css"**>**

**</head>**

1. CSS Selector
   1. **Universal selector :** \* { color: green; font-size: 10px }
   2. **ID selector :** <div id= “container” ></div> | #container { width: 90px; margin: 0 auto; }
   3. **Element type selector :** body { color: green; font-size: 10px }
   4. **Class selector:** declared with dot: <div class=”box”></div> | .box {padding : 20px; margin: 20px}
   5. **Descendent combinatory:** it is used to combine two or more selectors,

Html: <div id=”container”> <div class=”box”> </div> </div>

CSS: #container .box { float: left }

e.g. –HTML <p class=”fname”> </p>

CSS: p.fname { align: “center” } //for <p> tag and class =”fname”

1. Change Font size: three ways to set the size – pt (point), px (pixels), em (m)

e.g. – h1 { font-size: 18px }

1. A <span> element used to color a part of a text:

<p>My mother has <span style="color:blue">blue</span> eyes.</p>

1. Use the <mark> tag [defines marked text] if you want to highlight parts of your text.

e.g. <style>

mark {

background-color: red; /\* default is yellow \*/

color: black;

}

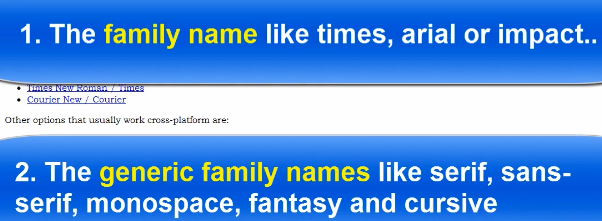
</style>

</head>

<body><mark>Highlighted text!!</mark> </body>

Fonts:

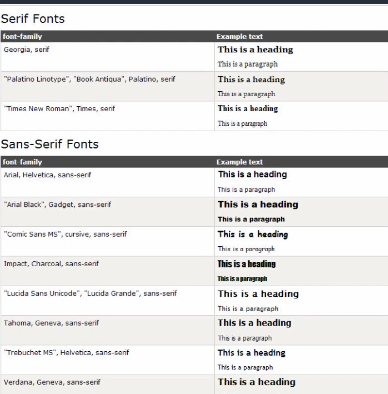




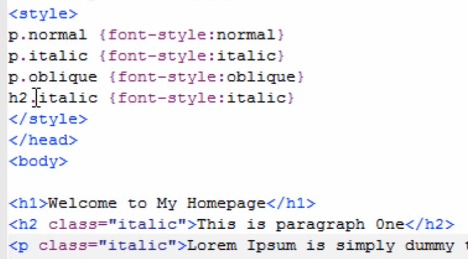


Fallback fonts are used when the primary font is not available on use computer.

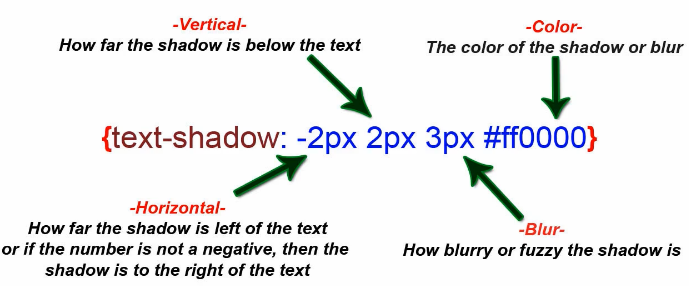
Use “ “ when the font name contains more than one word,e.g. “Times New Roman”



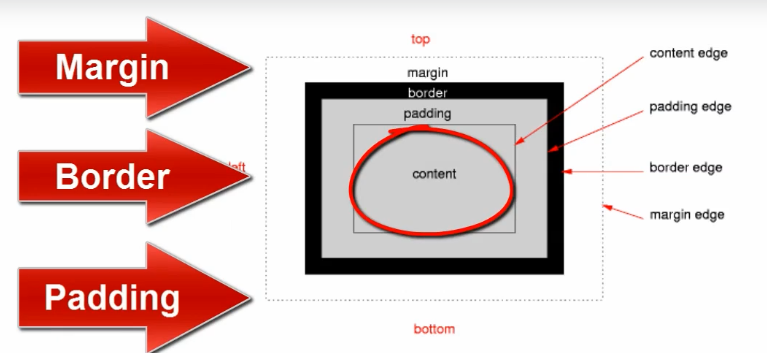
Font-styles



Add shadow effect

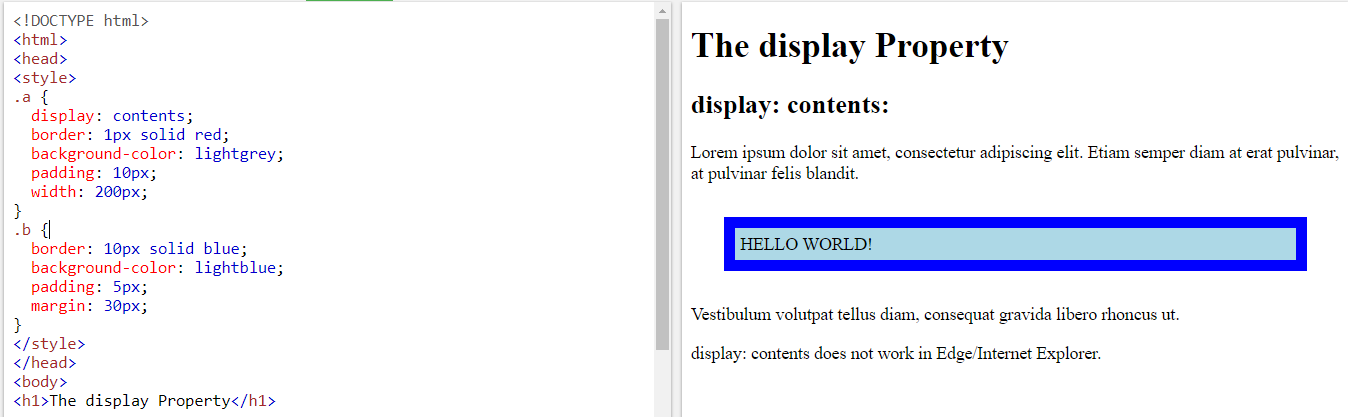


Content Box:

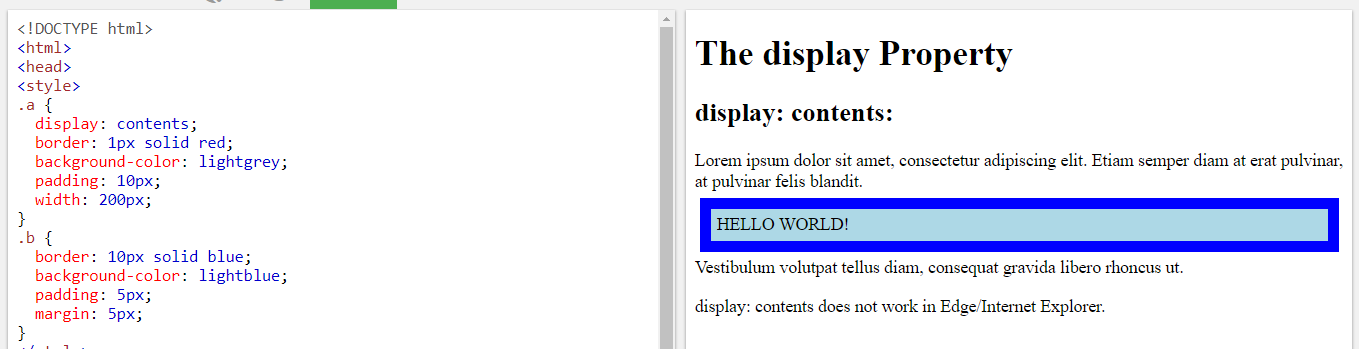


## Margin: is the space beyond border

margin: 30px;



margin: 5px;



Display: this property positions the element

p.ex1 {display: none;}

p.ex2 {display: inline;}

p.ex3 {display: block;}

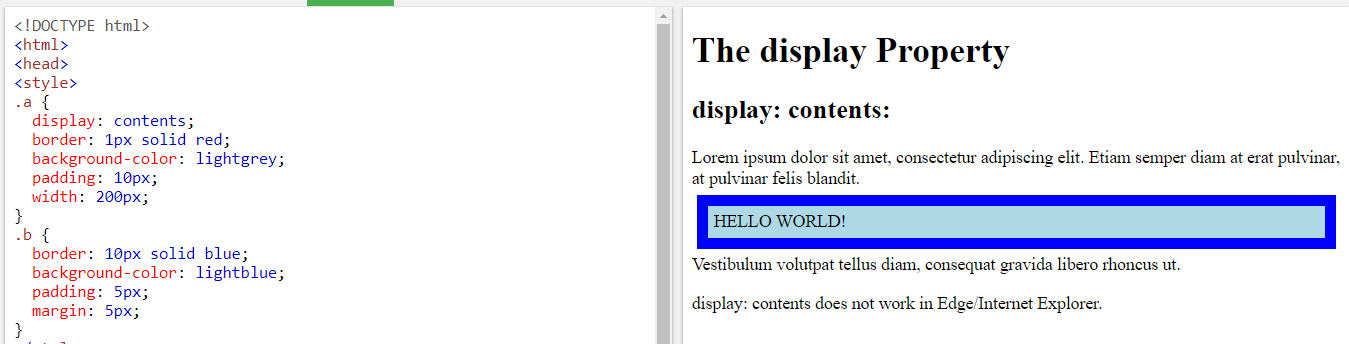
p.ex4 {display: inline-block;}

Padding: it is the space between border and content

padding: 50px;



Padding: 5px;



## Display

The CSS display property defines the type of rendering box to use for an element.

## Syntax

The syntax for the display CSS property is:

display: value;

### **Parameters or Arguments**

**value**

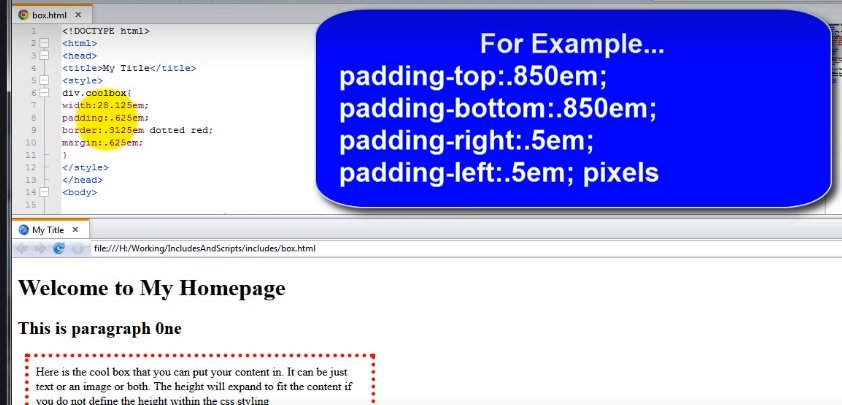
The type of rendering box. It can be one of the following:

| Value | Description |
| --- | --- |
| none | Turn off display of an element div { display: none; } |
| inline | Element will generate an inline element box div { display: inline; } |
| block | Element will generate a block element box div { display: block; } |
| inline-block | Element will generate a block element box that behaves like an inline box div { display: inline-block; } |
| list-item | Element will generate a block element box for the content and a separate list-item inline box div { display: list-item; } |
| table | Element will behave like the [HTML <table> element](https://www.techonthenet.com/html/elements/table_tag.php) div { display: table; } |
| table-caption | Element will behave like the [HTML <caption> element](https://www.techonthenet.com/html/elements/caption_tag.php) div { display: table-caption; } |
| table-column | Element will behave like the HTML <col> element div { display: table-column; } |
| table-column-group | Element will behave like the HTML <colgroup> element div { display: table-column-group; } |
| table-header-group | Element will behave like the HTML <thead> element div { display: table-header-group; } |
| table-row-group | Element will behave like the HTML <tbody> element div { display: table-row-group; } |
| table-footer-group | Element will behave like the HTML <tfoot> element div { display: table-footer-group; } |
| table-row | Element will behave like the [HTML <tr> element](https://www.techonthenet.com/html/elements/tr_tag.php) div { display: table-row; } |
| table-cell | Element will behave like the [HTML <td> element](https://www.techonthenet.com/html/elements/td_tag.php) div { display: table-cell; } |
| inherit | Element will inherit the display from its parent element div { display: inherit; } |

## Flex

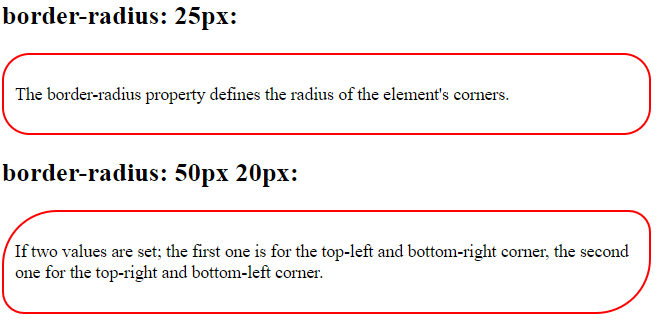
Flex is basically used to manage the HTML element width when there is change in screen size and the element is supposed to shrink or grow.

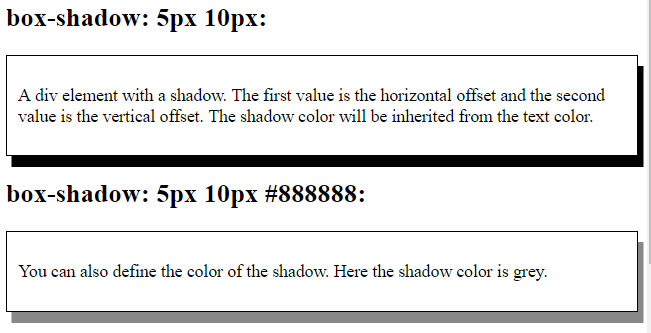
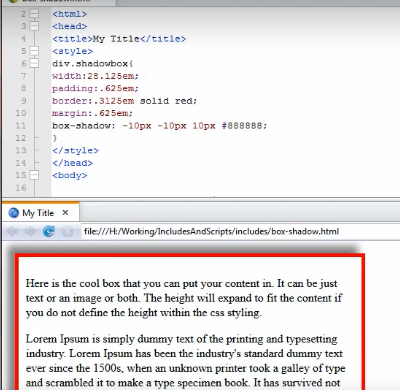






Rounded box:



7/25/2019

In HTML5, the <hr> tag defines a thematic break.

In HTML 4.01, the <hr> tag represents a horizontal rule.

However, the <hr> tag may still be displayed as a horizontal rule in visual browsers, but is now defined in semantic terms, rather than presentational terms.

All the layout attributes are removed in HTML5. Use CSS instead.

HTML <option> tag

It is used to create a dropdown. It is used with <select> tag. <select> enclosed list of option elements.

<!DOCTYPE html>

<html>

<body>

<select>

<option>Choose an option</option>

<option value=”Volvo”>volvo</option>

<option value=”saab”>Saab</option>

<option value=”opel”>Opel</option>

<option value=”audi”>Audi</option>

</select>

</body>

</html>



Value attribute

The value attribute specifies the value of an <input> element.

The value attribute in [HTML](http://learnwebdevelopment.info/html/)is used differently for different input types:

* For "button", "reset", and "submit" - it defines the text on the button
* For "text", "password", and "hidden" - it defines the initial (default) value of the input field
* For "checkbox", "radio", "image" - it defines the value associated with the input (this is also the value that is sent on submit)