

List Properties

PropertyDescription

- **list-style-image** → Specifies an image as the list-item marker
- **list-style-position** → Specifies if the list-item markers should appear *inside* or *outside* the content flow
- **list-style-type** → Specifies the type of list-item marker
- **list-style** → Sets all the properties for a list in one declaration

list-style: *list-style-type* | *list-style-position* | *list-style-image*

```
<html> <head>
<style type="text/css">
ul.a {list-style-type:circle;}
ul.b {list-style-type:disc;}
ul.c {list-style-type:square;}
ol.f {list-style-type:decimal;}
ol.g {list-style-type:decimal-leading-
    zero;}
ol.n {list-style-type:lower-alpha;}
ol.q {list-style-type:lower-roman;}
ol.r {list-style-type:upper-alpha;}
```

```
ol.t {list-style-type:upper-roman;}
ol.u {list-style-type:none;}
ol.v {list-style-image:
    url("sqpurple.gif");}
</style> </head>
<body>
<ul class="a">
<li>Circle type</li> </ul>
<ul class="b">
<li>Disc type</li> </ul>
```

```
<ul class="c">
<li>Square type</li></ul>
<ol class="f">
<li>Decimal type</li></ol>
<ol class="g">
<li>Decimal-leading-zero type</li>
</ol>
<ol class="n">
<li>Lower-alpha type</li> </ol>
<ol class="q">
<li>Lower-roman type</li>
</ol>
```

```
<ol class="r">
<li>Upper-alpha type</li>
</ol>
<ol class="t">
<li>Upper-roman type</li>
</ol>
<ol class="u">
<li>None type</li>
</ol>
<ol class="v">
<li>Image type</li>
</ol>
</body> </html>
```

- **Syntax**
- The syntax of pseudo-classes:
 - `selector:pseudo-class {property: value}`
- CSS classes can also be used with pseudo-classes:
 - `selector.class:pseudo-class {property: value}`
- Ex:
 - `a:link {color: #FF0000} /* unvisited link */`
 - `a:visited {color: #00FF00} /* visited link */`
 - `a:hover {color: #FF00FF} /* mouse over link */`
 - `a:active {color: #0000FF} /* selected link */`

Link Properties

- The four links states are:

1. a:link - a normal, unvisited link
2. a:visited - a link the user has visited
3. a:hover - a link when the user mouse over it
4. a:active - a link the moment it is clicked

a:link {color:#FF0000;} /* unvisited link */

a:visited {color:#00FF00;} /* visited link */

a:hover {color:#FF00FF;} /* mouse over link */

a:active {color:#0000FF;} /* selected link */

- When setting the style for several link states, there are some order rules:

a:hover MUST come after a:link and a:visited

a:active MUST come after a:hover

Text Decoration

```
a:link {text-decoration:none;}  
a:visited {text-decoration:none;}  
a:hover {text-decoration:underline;}  
a:active {text-decoration:underline;}
```

Background Color

The background-color property specifies the background color for links:

```
a:link {background-color:#B2FF99;}
```

```

<html> <head>
<style type="text/css">
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}
a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}
a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}
a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}
a.five:link { color:#ff0000;
text-decoration:none;}

```

```

a.five:visited {color:#0000ff;
text-decoration:none;}
a.five:hover {text-decoration:underline;}
a.six:link,a.six:visited
{ display:block;
font-weight:bold;
color:#FFFFFF;
background-color:#98bf21;
width:120px;
text-align:center;
padding:4px;
text-decoration:none;}
a.six:hover,a.six:active
{ background-color:#7A991A;
text-decoration:underline;}
</style> </head>

```

```
<body>
```

```
<p>Mouse over the links to see them change layout.</p>
```

```
<p><b><a class="one" href="default.asp" target="_blank">This link  
changes color</a></b></p>
```

```
<p><b><a class="two" href="default.asp" target="_blank">This link  
changes font-size</a></b></p>
```

```
<p><b><a class="three" href="default.asp" target="_blank">This link  
changes background-color</a></b></p>
```

```
<p><b><a class="four" href="default.asp" target="_blank">This link  
changes font-family</a></b></p>
```

```
<p><b><a class="five" href="default.asp" target="_blank">This link  
changes text-decoration</a></b></p>
```

```
<p><b><a class="six" href="../tutorial.html" target="_blank">This  
link changes Box Color</a></b></p>
```

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

CSS Selectors

- CSS selectors are used to "find" (or select) the HTML elements you want to style.
- We can divide CSS selectors into five categories:
 - Simple selectors (select elements based on name, id, class)
 - Combinator selectors (select elements based on a specific relationship between them)
 - Pseudo-class selectors (select elements based on a certain state)
 - Pseudo-elements selectors (select and style a part of an element)
 - Attribute selectors (select elements based on an attribute or attribute value)

CSS Combinator Selector

- A CSS selector can contain more than one simple selector. Between the simple selectors, can include a combinator.
- There are four different combinators in CSS:
 - descendant selector (space)
 - child selector ($>$)
 - adjacent sibling selector ($+$)
 - general sibling selector (\sim)

Descendant and Child selector

- **Descendant Selector (space)** → matches all elements that are descendants of a specified element. <p> elements inside <div> elements:

```
div p { background-color: yellow; }
```

- **Child Selector(>)** → The child selector selects all elements that are the children of a specified element.

```
div > p {  
    background-color: yellow;  
}
```

Adjacent Sibling Selector (+) and General Sibling Selector (~)

- **Adjacent sibling selector(+)** → used to select an element that is directly after another specific element. Sibling elements must have the same parent element, and "adjacent" means "immediately following".

```
div + p { background-color: yellow; }
```

- **General sibling selector(~)** → selects all elements that are next siblings of a specified element.

```
div ~ p {  
    background-color: yellow;  
}
```

What are Pseudo-classes?

- A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
 - Style an element when a user mouse over it
 - Style visited and unvisited links differently
 - Style an element when it gets focus
- The syntax of pseudo-classes:
 - `selector:pseudo-class { property: value; }`

- **CSS - The :first-child Pseudo-class**

- The :first-child pseudo-class matches a specified element that is the first child of another element.

p:first-child { color: blue; }

- **Match the first <i> element in all <p> elements**

- the selector matches the first <i> element in all <p> elements:

p i:first-child { color: blue; }

- **Match all <i> elements in all first child <p> elements**

p:first-child i { color: blue; }

- The **:nth-child(*n*)** selector matches every element that is the *n*th child of its parent. *n* can be a number, a keyword (odd or even), or a formula (like $an + b$). *a* represents a cycle size, *n* is a counter (starts at 0), and *b* is an offset value.

```
/* Selects the second element of div siblings */  
div:nth-child(2) {  
  background: red;  
}
```

```
/* Selects the second li element in a list */  
li:nth-child(2) {  
  background: lightgreen;  
}
```

```
/* Selects every third element among any group of siblings */  
:nth-child(3) {  
  background: yellow;  
}
```

nth-child($an+b$)

- Using a formula ($an + b$). Description: a represents a cycle size, n is a counter (starts at 0), and b is an offset value.
- Here, we specify a background color for all p elements whose index is a multiple of 3 (will select the third, sixth, ninth, etc):

```
p:nth-child(3n+0) {  
  background: red;  
}
```

- The **:nth-last-child(*n*)** selector matches every element that is the *n*th child, regardless of type, of its parent, counting from the last child. *n* can be a number, a keyword, or a formula.

- a background color for every <p> element that is the second child of its parent, counting from the last child:

```
p:nth-last-child(2) { background: red;}
```

- **:nth-last-of-type(*n*)** selector matches every element that is the *n*th child, of a particular type, of its parent, counting from the last child. *n* can be a number, a keyword, or a formula.

- a background color for every <p> element that is the second p element of its parent, counting from the last child:

```
p:nth-last-of-type(2) {  
  background: red;  
}
```

CSS3 Attribute Selectors

- Style HTML Elements With Specific Attributes
- It is possible to style HTML elements that have specific attributes or attribute values.
- **CSS [attribute] Selector**
 - The [attribute] selector is used to select elements with a specified attribute.
 - Example: selects all <a> elements with a target attribute:

```
a[target] { background-color: yellow; }
```

- **CSS [attribute="value"] Selector**
 - The [attribute="value"] selector is used to select elements with a specified attribute and value.
 - for example: selects all <a> elements with a target="_blank" attribute:

```
a[target="_blank"] { background-color: yellow; }
```

CSS3 Attribute Selectors

- **CSS [attribute~="value"] Selector**
- The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

```
[title~="flower"] {  
    border: 5px solid yellow;  
}
```

What are Pseudo-Elements?

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
 - Style the first letter, or line, of an element
 - Insert content before, or after, the content of an element
- The syntax of pseudo-elements:

`selector::pseudo-element { property: value; }`

All CSS Pseudo Elements

Selector	Example	Example description
<u>::after</u>	p::after	Insert something after the content of each <p> element
<u>::before</u>	p::before	Insert something before the content of each <p> element
<u>::first-letter</u>	p::first-letter	Selects the first letter of each <p> element
<u>::first-line</u>	p::first-line	Selects the first line of each <p> element
<u>::marker</u>	::marker	Selects the markers of list items
<u>::selection</u>	p::selection	Selects the portion of an element that is selected by a user

The :first-line Pseudo-element

- The "first-line" pseudo-element is used to add special styles to the first line of the text in a selector:

p {font-size: 12pt}

p::first-line {color: #0000FF; font-variant: small-caps}

<p>Some text that ends up on two or more lines</p>

- The output could be something like this:

SOME TEXT THAT ENDS up on two or more lines

The :first-letter Pseudo-element

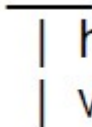
- The "first-letter" pseudo-element is used to add special style to the first letter of the text in a selector:

p {font-size: 12pt}

p::first-letter {font-size: 200%; float: left}

<p>The first words of an article.</p>


- The output could be something like this:

he first
words of an
article.

CSS - The ::before Pseudo-element

- The ::before pseudo-element can be used to insert some content before the content of an element.
- The following example inserts an image before the content of each `<h1>` element:

```
h1::before {  
    content: url(smiley.gif);  
}
```

 **This is a heading**

CSS - The ::after Pseudo-element

- The ::after pseudo-element can be used to insert some content after the content of an element.
- The following example inserts an image after the content of each `<h1>` element:

```
h1::after {  
    content: url(smiley.gif);  
}
```

This is a heading 

width and height property

- The height and width
 - *auto* (this is default. Means that the browser calculates the height and width),
 - *length values*, like px, cm, etc., or in percent (%) of the containing block.

```
<style>
```

```
div {
```

```
    height: 100px;
```

```
    width: 500px;
```

```
    background-color: powderblue;
```

```
}
```

```
</style>
```

```
<body>
```

```
<h2>Set the height and width of an element</h2>
```

```
<p>This div element has a height of 100px and a width of 500px:</p>
```

```
<div></div>
```

```
</body>
```

```
</html>
```

Table Properties

Table Borders

table, th, td

```
{  
border: 1px solid red;  
}
```

- Notice that the table in the example above has double borders. This is because both the table and the th/td elements have separate borders.
- To display a single border for the table, use the border-collapse property.

Collapse Borders

- The border-collapse property sets whether the table borders are *collapse* into a single border or *separated or initial*:

border-collapse: separate | collapse | initial

table

{

border-collapse:collapse;

}

table, td, th

{

border:1px solid red;

}

Table Width and Height

- Width and height of a table is defined by the width and height properties.

table

{

width:100%;

}

th

{

height:50px;

}

Table Text Alignment

- The text in a table is aligned with the text-align and vertical-align properties.
- The text-align property sets the horizontal alignment, like left, right, or center

```
td
{
text-align:right;
}
```

- The vertical-align property sets the vertical alignment, like top, bottom, or middle:

```
td
{
height:50px;
vertical-align:bottom;
}
```

Table Padding

- To control the space between the border and content in a table, use the padding property on td and th elements:

```
td
{
padding:15px;
}
```

Table Color

- The example below specifies the color of the borders, and the text and background color of th elements:

```
table, td, th
{
border:1px solid green;
}
th
{
background-color:green;
color:white;
}
```

Table border-spacing

- The distance between the borders of adjacent cells: **border-spacing: *length* | initial**

```
table.ex1 {  
border-collapse: separate;  
border-spacing: 10px;}  
table.ex2 {  
border-collapse: separate;  
border-spacing: 10px 50px;  
}
```

Table caption-side

- The **caption-side** property specifies the placement of a table caption.

caption-side: top | bottom | initial

- table, td, th**
{
border: 1px solid green;
}
th
{
background-color: green;
color: white;
}

Table empty-cells

- Sets whether or not to display borders and background on empty cells in a table

empty-cells: show | hide | initial

```
table {  
  border-collapse: separate;  
  empty-cells: hide;  
}
```

```

<html> <head>
<style type="text/css">
table
{
    border-collapse:collapse;
    width:50%; }
th
{
    height:50px;
    vertical-align:center; }
td
{
    text-align:right; }
table,th,td
{
    border:1px solid red;
}
</style> </head>

```

```

<body>
<table>
<tr>
    <th>Student Name</th>
    <th>Seminar Topic</th>
</tr>
<tr>
    <td>Madhusuthanan P</td>
    <td>Internet, Intranet and
        WWW</td>
</tr>
<tr>
    <td>Roopa S</td>
    <td>Internet Protocols</td>
</tr>
</table> </body> </html>

```

max-width

- improve the browser's handling of small windows.
- This is important when making a site usable on small devices

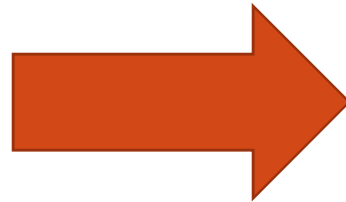
```
div.ex2 {  
    max-width: 500px;  
    margin: auto;  
    border: 3px solid #73AD21;  
}
```

CSS - outline

- The CSS outline properties specify the *style, color, and width* of an outline.
- An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".
- **outline-style** : *dotted, dashed, solid....*
- **outline-color**
- **outline-width** : *in px, em, cm, pt, thin, medium, or thick.*
- Shorthand
 - **outline**: outline-width outline-style (required) outline-color

Grouping Selectors

```
h1
{
color:green;
}
h2
{
color:green;
}
p
{
color:green;
}
```



- To minimize the code, you can group selectors.
- Separate each selector with a comma.

```
h1,h2,p
{
color:green;
}
```

Nesting Selectors

- It is possible to apply a style for a selector within a selector.

p → style is specified for all p elements

```
{  
color:blue;  
text-align:center;  
}
```

.marked → style is specified for all elements with class="marked"

```
{  
background-color:red;  
}
```

.marked p → style is specified only for p elements within elements with class="marked"

```
{  
color:white;  
}
```

```
<html> <head>
<style type="text/css">
p
{
    color:blue;
    text-align:center;
}
.marked
{
    background-color:red;
}
.marked p
{
color:white;
}
</style> </head>
```

```
<body>
<p>This is a blue, center-aligned
    paragraph.</p>
<div class="marked">
<h1> Heading 1 </h1>
<p>This p element should not be
    blue.</p>
</div>
<p>p elements inside a "marked"
    classed element keeps the
    alignment style, but has a
    different text color.</p>
</body>
</html>
```

overflow

- specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area

overflow:visible | hidden | scroll | auto

- visible - Default. The overflow is not clipped. It renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
- auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

float and clear

float

- be used to wrap text around images.
- The following example specifies that an image should float to the right in a text:

```
img {  
    float: right;  
    margin: 0 0 10px 10px;  
}
```

Clear

- used to control the behavior of floating elements.
- specifies on which sides of an element floating elements are not allowed to float:

clear: left | right

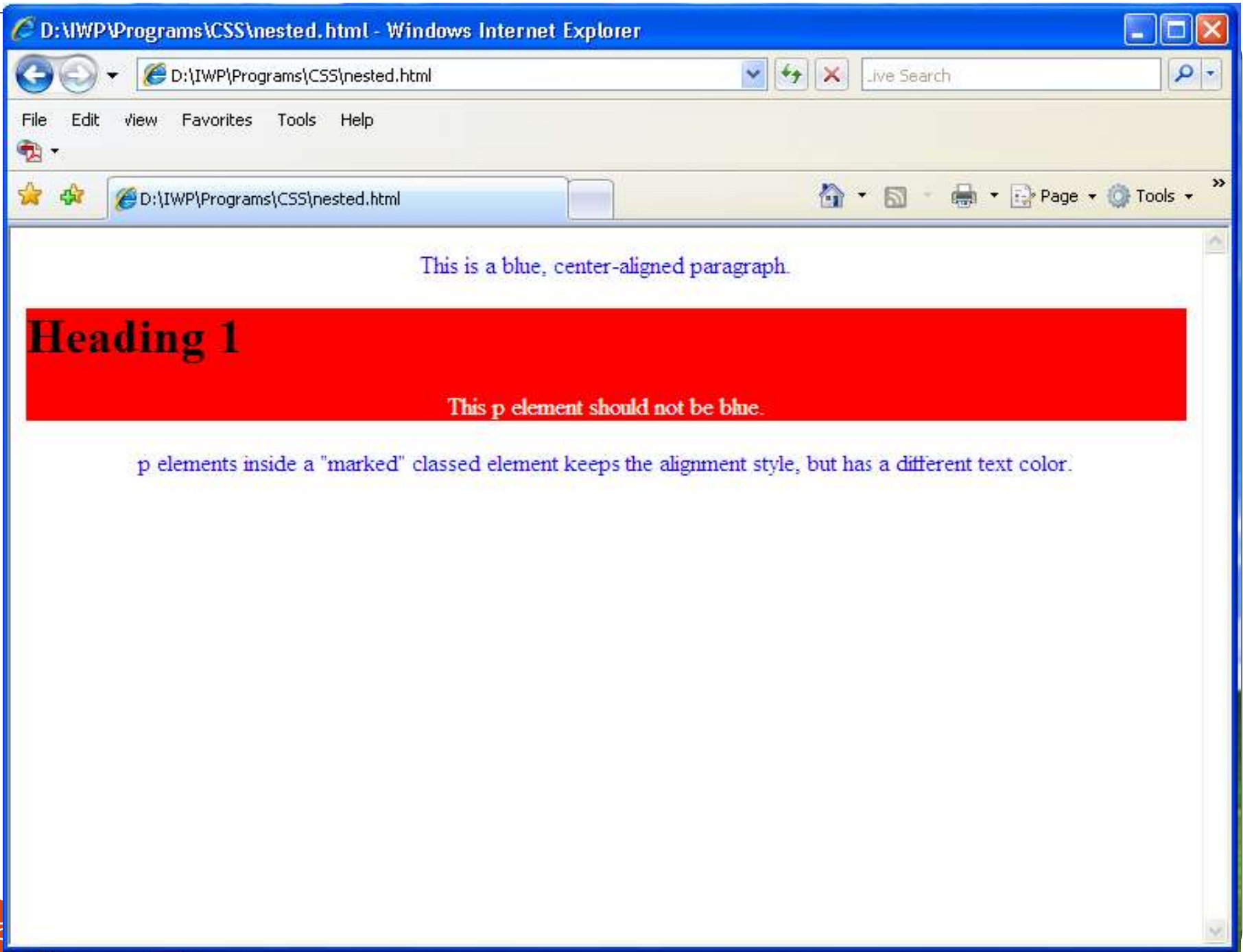
The display Property

- The display property specifies if/how an element is displayed.
- Every HTML element has a default display value depending on what type of element it is. The display value for most elements is **block**, **inline** and **none**.
 - block->A block-level element always starts on a new line and takes up the full width available.
 - inline->does not start on a new line and only takes up as much width as necessary. Any height and width properties will have no effect. This is default.
 - none->to hide and show elements without deleting and recreating them
 - inline-block→Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values

```

<head>
<style>
li { display: inline;}
span { display: block;}
h1.hidden{display:none;}
p{visibility:hidden;}
</style>
</head>
<body>
<h1 class="hidden">This is a hidden heading</h1>
<p>Display a list of links as a horizontal menu:</p>
<ul>
  <li><a href="/html/default.asp" target="_blank">HTML</a></li></ul>
  <span>A display property with a value of "block" results in</span> <span>a line break
  between the two elements.</span></body>
</html>

```



display property value

Value	Description
inline	Displays an element as an inline element
block	Displays an element as a block element
contents	Makes the container disappear, making the child elements children of the element the next level up in the DOM
flex	Displays an element as a block-level flex container
grid	Displays an element as a block-level grid container
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
inline-flex	Displays an element as an inline-level flex container
inline-grid	Displays an element as an inline-level grid container
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a <code></code> element
run-in	Displays an element as either block or inline, depending on context

table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element
none	The element is completely removed
initial	Sets this property to its default value
inherit	Inherits this property from its parent element

Navigation Bars

```
body { margin: 0; }
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}
li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}
li a.active {
  background-color: #4CAF50;
  color: white;}
```

Vertical



Horizontal



```
li a:hover:not(.active) {
  background-color: #555;
  color: white;
}
</style>
</head>
<body>

<ul>
  <li><a class="active"
href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
```

Navigation Bars

```
body { margin: 0; }
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  position: fixed;
  height: 100%;
  overflow: auto;
}
li a {
  display: block;
  color: #000;
  padding: 8px 16px;
  text-decoration: none;
}
li a.active {
  background-color: #4CAF50;
  color: white;}
```

Vertical



Horizontal



```
li a: hover: not(.active) {
  background-color: #555;
  color: white;
}
</style>
</head>
<body>

<ul>
  <li><a class="active"
href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li><a href="#about">About</a></li>
</ul>
```

text-shadow

- adds shadow to text
- `text-shadow: h-shadow v-shadow blur-radius color | none`

```
h1 {  
    text-shadow: 2px 2px 8px #FF0000;  
}
```

```
h1 {  
    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;  
}
```

box-shadow

- attaches one or more shadows to an element.
- `box-shadow: none | h-shadow v-shadow blur spread color`

```
div {  
  box-shadow: 10px 10px grey;  
}  
div.card {  
  width: 250px;  
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2), 0  
    6px 20px 0 rgba(0, 0, 0, 0.19);  
  text-align: center;  
}
```

Responsive web design with viewport

- The viewport is the user's visible area of a web page.
- The viewport varies with the device, and will be smaller on a mobile phone than on a computer screen

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

- This gives the browser instructions on how to control the page's dimensions and scaling.
- The **width=device-width** part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).
- The **initial-scale=1.0** part sets the initial zoom level when the page is first loaded by the browser.

What is a Media Query?

- Media query is a CSS technique introduced in CSS3.
- It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.
- **Example**
- If the browser window is 600px or smaller, the background color will be lightblue:

```
@media only screen and (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
}
```

```
<!DOCTYPE html>
```

```
<html><head>
```

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

```
<style>
```

```
body {
```

```
  background-color: lightgreen;
```

```
}
```

```
@media only screen and (max-width: 600px) {
```

```
  body {
```

```
    background-color: lightblue;
```

```
  }}</style></head><body>
```

```
<p>Resize the browser window. When the width of this document is 600 pixels or less, the background-color is "lightblue", otherwise it is "lightgreen".</p>
```

```
</body>
```

```
</html>
```