Part Two: CSS

What You Should Already Know

- Before you continue you should have a basic understanding of the following:
  - HTML

---

1. What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

Styles Solved a Big Problem

- HTML was never intended to contain tags for formatting a document.
- HTML was intended to define the content of a document, like:

```
<h1>This is a heading</h1>
<p>This is a paragraph.</p>
```

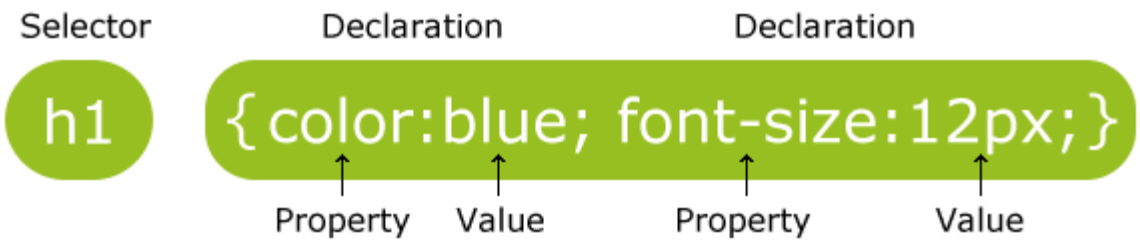
- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file.
- All browsers support CSS today.

CSS Saves a Lot of Work!

- CSS defines HOW HTML elements are to be displayed.
- Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file!

CSS Syntax

- A CSS rule set consists of a selector and a declaration block:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a property name and a value, separated by a colon.

CSS Example

- A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly braces:

```
p
{
color:red;text-align:center;
}
```

- To make the CSS code more readable, you can put one declaration on each line, like this:

Example

```
p
{
   color: red;
   text-align: center;
}
```

CSS Comments

- Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.
- A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

Example

```
p {
   color: red;
   /* This is a single-line comment */
   text-align: center;
}
/* This is
a multi-line
comment */
```

The class Selector

- The class selector finds elements with the **specific class**.
- The class selector uses the HTML **class attribute**.
- To find elements with a **specific class**, write a period character, followed by the name of the class:
- In the example below, all HTML elements with class="myp" will be center-aligned:

Example

```
.myp
 {
   text-align: center;
   color: red;
}
```

- You can also specify that only specific HTML elements should be affected by a class.
- In the example below, all p elements with class="myp" will be center-aligned:

Example

```
p.myp
 {
   text-align: center;
   color: red;
}
```

- Do **NOT** start a class name with a number!

Grouping Selectors

- In style sheets there are often elements with the same style:

```
h1 {
   text-align: center;
   color: red;
}
h2 {
   text-align: center;
   color: red;
}
p {
   text-align: center;
   color: red;
}
```

- To minimize the code, you can group selectors.
- To group selectors, **separate each selector with a comma**.
- In the example below we have grouped the selectors from the code above:

Example

```
h1, h2, p
{
   text-align: center;
   color: red;
}
```

---

2. CSS How to...

- When a browser reads a style sheet, it will format the document according to the information in the style sheet.

Three Ways to Insert CSS

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

- An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing just one file.
- Each page must include a link to the style sheet with the <link> tag. The <link> tag goes inside the head section:

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

- An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension. An example of a style sheet file is shown below:

"**myStyle.css**":

```
Body
```

```
{
  Background-color: lightblue;
}
h1
 {
  color: navy;
  margin-left: 20px;
}
```

- Do not add a space between the property value and the unit (such as margin-left: **20 px**;). The correct way is: margin-left: **20p**x;

Internal Style Sheet

- An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, inside the <style> tag, like this:

Example

```
<head>
<style>
body {
  background-color: linen;
}
h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
```

Inline Styles

- An inline style loses many of the advantages of a style sheet (by mixing content with presentation). Use this method carefully!
- To use inline styles, add the style attribute to the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a h1 element:

Example

```
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
```

Multiple Style Sheets

- If some properties have been set for the same selector in different style sheets, the values will be inherited from the **more specific style sheet**.
- For example, assume that an **external style sheet** has the following properties for the h1 selector:

```
h1 {
  color: navy;
  margin-left: 20px;
}
```

- Then, assume that an **internal style sheet** also has the following property for the h1 selector:

```
h1 {
    color: orange;
}
```

- If the page with the internal style sheet also links to the external style sheet the properties for the h1 element will be:

```
color: orange;
margin-left: 20px;
```

- The left margin is inherited from the external style sheet and the color is replaced by the internal style sheet.

Multiple Styles Will Cascade into One

Styles can be specified:

- Inside an HTML element
- Inside the head section of an HTML page
- In an external CSS file

Tip: Even multiple external style sheets can be referenced inside a single HTML document.

Cascading order

- What style will be used when there is more than one style specified for an HTML element?
- Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:
    1. Browser default
    2. External style sheet
    3. Internal style sheet (in the head section)
    4. Inline style (inside an HTML element)
- So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).
- **Note:** If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet

---

3. CSS Background

- CSS background properties are used to define the background effects of an element.
- CSS properties used for background effects:
    - background-color
    - background-image
    - background-repeat
    - background-attachment
    - background-position

Background Color

- The background-color property specifies the background color of an element.
- The background color of a page is defined in the body selector:

Example

```
body
```

```
{
   background-color: #b0c4de;
}
```

With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a color name - like "red"

Background Image

- The background-image property specifies an image to use as the background of an element.
- By default, the image is repeated so it covers the entire element.
- The background image for a page can be set like this:

Example

```
body
{
   background-image: url("paper.gif");
}
```

Background Image - Repeat Horizontally or Vertically

- By default, the background-image property repeats an image both horizontally and vertically.

Example

```
body
{
   background-image: url("gradient_bg.png");
}
```

- If the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
Body
 {
   background-image: url("gradient_bg.png");
   background-repeat: repeat-x;
}
```

Background Image - Set position and no-repeat

- When using a background image, use an image that does not disturb the text
- Showing the image only once is specified by the background-repeat property:

Example

```
body {
   background-image: url("img_tree.png");
   background-repeat: no-repeat;
}
```

- In the example above, the background image is shown in the same place as the text.
- We want to change the position of the image, so that it does not disturb the text too much.
- The position of the image is specified by the background-position property:

Example

```
body {
   background-image: url("img_tree.png");
   background-repeat: no-repeat;
   background-position: right top;
   background-attachment: scroll;
}
```

Background - Shorthand property

- As you can see from the examples above, there are many properties to consider when dealing with backgrounds.
- To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property.
- The shorthand property for background is simply "background":

Example

```
body {
   background: #ffffff url("img_tree.png") no-repeat right top;
}
```

When using the shorthand property the order of the property values is:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the ones that are present are in this order.

---

4. CSS Text

Text Color

- The color property is used to set the color of the text.
- The default color for a page is defined in the body selector.

Example

```
body {
   color: blue;
}
h1 {
   color: #00ff00;
}
h2 {
   color: rgb(255,0,0);
}
```

- **Note:** For W3C compliant CSS: If you define the color property, you must also define the background-color property

---

Text Alignment

- The text-align property is used to set the horizontal alignment of a text.
- Text can be centered, or aligned to the left or right, or justified.

Example

```
h1 {
  text-align: center;
}
p.date {
  text-align: right;
}
p.main {
  text-align: justify;
}
```

Text Decoration

- The text-decoration property is used to set or remove decorations from text.
- The text-decoration property is mostly used to remove underlines from links for design purposes:

Example

```
a {
  text-decoration: none;
}
```

- It can also be used to decorate text:

Example

```
h1 {
  text-decoration: overline;
}
h2 {
  text-decoration: line-through;
}
h3 {
  text-decoration: underline;
}
```

- **Note:** It is not recommended to underline text that is not a link, as this often confuses users.

Text Transformation

- The text-transform property is used to specify uppercase and lowercase letters in a text.
- It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example

```
p.uppercase {
  text-transform: uppercase;
}
p.lowercase {
  text-transform: lowercase;
}
p.capitalize {
  text-transform: capitalize;
}
```

Text Indentation

- The text-indent property is used to specify the indentation of the first line of a text.

Example

```
p {
   text-indent: 50px;
}
```

Text direction

- The direction property specifies the text direction/writing direction.

Example

- Set the text direction to "right-to-left":

```
div {
   direction: rtl;
}
```

Text Shadow

- The CSS3 text-shadow property applies shadow to text.
- In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Example

- Text shadow effect!

```
h1 {
   text-shadow: 2px 2px;
}
```

- Next, add a color to the shadow:

```
h1 {
   text-shadow: 2px 2px red;
}
```

- Then, add a blur effect to the shadow:

```
h1 {
   text-shadow: 2px 2px 5px red;
}
```

- The following example shows a white text with black shadow:

```
h1 {
   color: white;
   text-shadow: 2px 2px 4px #000000;
}
```

CSS Font

- CSS font properties define the font family, boldness, size, and the style of a text.

Font Family

- The font family of a text is set with the font-family property.
- The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.
- Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

Note: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

- More than one font family is specified in a comma-separated list:

Example

```
p {
  font-family: "Times New Roman", Times, serif;
}
```

Font Style

- The font-style property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {
  font-style: normal;
}
p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
```

Font Size

- The font-size property sets the size of the text.
- Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an absolute or relative size.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size with Pixels

- Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {
   font-size: 40px;
}
h2 {
   font-size: 30px;
}
p {
   font-size: 14px;
}
```

Tip: However, you can still use the zoom tool to resize the entire page.

Set Font Size with Em

- To allow users to resize the text (in the browser menu), many developers use em instead of pixels.
- The em size unit is recommended by the W3C.
- 1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.
- The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {
   font-size: 2.5em; /* 40px/16=2.5em */
}
h2 {
   font-size: 1.875em; /* 30px/16=1.875em */
}
p {
   font-size: 0.875em; /* 14px/16=0.875em */
}
```

- In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.
- Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Font-weight

- The font-weight property sets how thick or thin characters in text should be displayed.

Example

- Set different font weight for three paragraphs:

```
p.normal {
   font-weight: normal;
}

p.thick {
   font-weight: bold;
}
p.light {
   font-weight: lighter;
}
```

```
p.thicker {
   font-weight: 900;
}
```

5. CSS Links

- Links can be styled with any CSS property (e.g. color, font-family, background, etc.).
- In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

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

Example

```
/* unvisited link */
a:link {
   color: #FF0000;
}
/* visited link */
a:visited {
   color: #00FF00;
}
/* mouse over link */
a:hover {
   color: #FF00FF;
}
/* selected link */
a:active {
   color: #0000FF;
}
```

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**

Common Link Styles

- In the example above the link changes color depending on what state it is in.
- Lets go through some of the other common ways to style links:

Text Decoration

- The text-decoration property is mostly used to remove underlines from links:

Example

```
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:

Example

```
a:link {
   background-color: #B2FF99;
}
a:visited {
   background-color: #FFFF85;
}
a:hover {
   background-color: #FF704D;
}
a:active {
   background-color: #FF704D;
}
```

6. CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

Different List Item Markers

- The type of list item marker is specified with the list-style-type property:

Example

```
ul.a {
   list-style-type: circle;
}
ul.b {
   list-style-type: square;
}
ol.c {
   list-style-type: upper-roman;
}
ol.d {
   list-style-type: lower-alpha;
}
```

- Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

- To specify an image as the list item marker, use the **list-style-image** property:

Example

```
ul {
  list-style-image: url('sqpurple.gif');
}
```

- The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.
- If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

List - Shorthand property

- It is also possible to specify all the list properties in one, single property. This is called a shorthand property.
- The shorthand property used for lists, is the list-style property:

Example

```
ul {
  list-style: square url("sqpurple.gif");
}
```

When using the shorthand property, the order of the values are:

- list-style-type
- list-style-position (for a description, see the CSS properties table below)
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

7. CSS Tables

- The look of an HTML table can be greatly improved with CSS:

Table Borders

- To specify table borders in CSS, use the border property.
- The example below specifies a black border for table, th, and td elements:

Example

```
table, th, td {
  border: 1px solid black;
}
```

- 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 collapsed into a single border or separated:

Example

```
table {
  border-collapse: collapse;
}
table, th, td {
  border: 1px solid black;
}
```

Table Width and Height

- Width and height of a table is defined by the width and height properties.
- The example below sets the width of the table to 100%, and the height of the th elements to 50px:

Example

```
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:

Example

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

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

Example

```
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:

Example

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

Table Color

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

Example

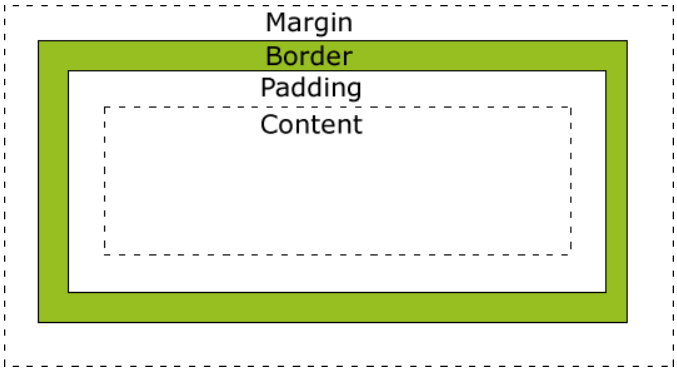
```
table, td, th {
  border: 1px solid green;
}
th {
  background-color: green;
  color: white;
}
```

---

8. CSS Box Model

The CSS Box Model

- All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around HTML elements, and it consists of: **margins**, **borders**, **padding**, and the **actual content**.
- The box model allows us to add a border around elements, and to define space between elements.
- The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

Example

```
div {
    width: 300px;
    padding: 25px;
    border: 25px solid navy;
    margin: 25px;
}
```

Width and Height of an Element

- In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add the padding, borders and margins

- Let's make a div element with a total width of 350px:

Example

```
div {
    width: 320px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0;
}
```

Let's do the math:
320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px

- The total width of an element should be calculated like this:
- Total element width = width + left padding + right padding + left border + right border + left margin + right margin
- The total height of an element should be calculated like this:
- Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

9. CSS Border

CSS Border Properties:

- The CSS border properties allow you to specify the **style**, **size**, and **color** of an element's border.

Border Style

- The border-style property specifies what kind of border to display.

Note: None of the border properties will have ANY effect unless the **border-style** property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders is the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Border Width

- The border-width property is used to set the width of the border.
- The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.

Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one {
    border-style: solid;
    border-width: 5px;
}
p.two {
    border-style: solid;
    border-width: medium;
}
```

Border Color

- The border-color property is used to set the color of the border.
- You can also set the border color to "transparent".
- If the border color is not set it is inherited from the color property of the element.
- **Note:** The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one {
    border-style: solid;
    border-color: red;
}
p.two {
    border-style: solid;
    border-color: #98bf21;
}
```

Border - Individual sides

- In CSS it is possible to specify different borders for different sides:

Example

```
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
```

- The example above can also be set with a single property:
- The border-style property can have from one to four values.
  `border-style: dotted solid double dashed;`
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

border-style: dotted solid double;
- top border is dotted
- right and left borders are solid
- bottom border is double

border-style: dotted solid;
- top and bottom borders are dotted
- right and left borders are solid

border-style: dotted;
- all four borders are dotted

- The border-style property is used in the example above. However, it also works with border-width and border-color.

10. CSS Margin

- The CSS margin properties define the space around elements.
- The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.
- The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

Note: It is also possible to use negative values, to overlap content

Margin - Individual sides

- In CSS, it is possible to specify different margins for different sides of an element:

Example

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 50px;
}
```

Margin - Shorthand property

- To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.
- The shorthand property for all the margin properties is "margin":

Example

```
p {
  margin: 100px 50px;
}
```

The margin property can have from one to four values.

- **margin: 25px 50px 75px 100px;**

  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- **margin: 25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- **margin: 25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px
- **margin: 25px;**
  - all four margins are 25px

11. CSS Padding

- The CSS padding properties define the space between the element border and the element content.
- The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.
- The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all puddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

- In CSS, it is possible to specify different padding for different sides:

Example

```
p {
  padding-top: 25px;
  padding-right: 50px;
  padding-bottom: 25px;
  padding-left: 50px;
}
```

Padding - Shorthand property

- To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.
- The shorthand property for all the padding properties is "padding":

Example

```
p {
  padding: 25px 50px;
}
```

The padding property can have from one to four values.

- **padding: 25px 50px 75px 100px;**

  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px
- **padding: 25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px
- **padding: 25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

padding: 25px;

  - all four paddings are 25px

12. CSS Display and Visibility

- The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

Hiding an Element - display:none or visibility:hidden

- Hiding an element can be done by setting the display property to "none" or the visibility property to "hidden". However, notice that these two methods produce different results:
- visibility:hidden hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

Example

```
h1.hidden {
    visibility: hidden;
}
```

- display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {
    display: none;
}
```

CSS Display - Block and Inline Elements

- A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-
-

An inline element only takes up as much width as necessary, and does not force line breaks.
Examples of inline elements:

-
- <a>

Changing How an Element is Displayed

- Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards.
- The following example displays <li> elements as inline elements:

Example

```
li {
    display: inline;
}
```

- The following example displays elements as block elements:

Example

```
span {
    display: block;
}
```

- **Note:** Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with **display: block** is not allowed to have other block elements inside of it.

13. CSS Positioning

- Positioning can be tricky sometimes!
- Decide which element to display in front!
- Elements can overlap!

Positioning

- The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.
- Elements can be positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the positioning method.
- There are four different positioning methods.

Static Positioning

- HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.
- Static positioned elements are not affected by the top, bottom, left, and right properties.

Fixed Positioning

- An element with fixed position is positioned relative to the browser window.
- It will not move even if the window is scrolled:

Example

```
p.pos_fixed {
    position: fixed;
    top: 30px;
    right: 5px;
}
```

- **Note:** IE7 and IE8 support the fixed value only if a !DOCTYPE is specified

- Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.
- Fixed positioned elements can overlap other elements.

Relative Positioning

- A relative positioned element is positioned relative to its normal position.

Example

```
h2.pos_left {
    position: relative;
    left: -20px;
}
h2.pos_right {
    position: relative;
    left: 20px;
}
```

- The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

Example

```
h2.pos_top {
    position: relative;
    top: -50px;
}
```

- Relatively positioned elements are often used as container blocks for absolutely positioned elements.

Absolute Positioning

- An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

Example

```
h2 {
    position: absolute;
    left: 100px;
    top: 150px;
}
```

- Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.
- Absolutely positioned elements can overlap other elements.

Overlapping Elements

- When elements are positioned outside the normal flow, they can overlap other elements.
- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element can have a positive or negative stack order:

Example

```
img {
    position: absolute;
    left: 0px;
    top: 0px;
```

```
    z-index: -1;
}
```

An element with greater stack order is always in front of an element with a lower stack order.
Note: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top

---

How Elements Float

- With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.
- Float is very often used for images, but it is also useful when working with layouts
- Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.
- A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.
- The elements after the floating element will flow around it.
- The elements before the floating element will not be affected.
- If an image is floated to the right, a following text flows around it, to the left:

Example

```
img {
    float: right;
}
```

Floating Elements Next to Each Other

- If you place several floating elements after each other, they will float next to each other if there is room.
- Here we have made an image gallery using the float property:

Example

```
.thumbnail {
    float: left;
    width: 110px;
    height: 90px;
    margin: 5px;
}
```

Turning off Float - Using Clear

- Elements after the floating element will flow around it. To avoid this, use the clear property.
- The clear property specifies which sides of an element other floating elements are not allowed.
- Add a text line into the image gallery, using the clear property:

Example

```
.text_line {
    clear: both;
}
```

---

14. CSS Horizontal Align

- In CSS, several properties are used to align elements horizontally.

Aligning Block Elements

- A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
-

In this tutorial we will show you how to horizontally align block elements for layout purposes.

Center Aligning Using the margin Property

- Block elements can be center-aligned by setting the left and right margins to "auto".
- **Note:** Using margin: auto; will not work in IE8 and earlier, **unless a !DOCTYPE is declared**
- Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element:

Example

```
.center {
    margin-left: auto;
    margin-right: auto;
    width: 70%;
    background-color: #b0e0e6;
}
```

Tip: Center-aligning has no effect if the width is 100%.

Left and Right Aligning Using the position Property

- One method of aligning elements is to use absolute positioning:

Example

```
.right {
    position: absolute;
    right: 0px;
    width: 300px;
    background-color: #b0e0e6;
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Left and Right Aligning Using the float Property

- One method of aligning elements is to use the float property:

Example

```
.right {
    float: right;
    width: 300px;
    background-color: #b0e0e6;
}
```

15. CSS Combinatory

- A combinatory is something that explains the relationship between the selectors.
- A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinatory.

There are four different combinatory in CSS3:

- descendant selector
- child selector
- adjacent sibling selector
- general sibling selector

Descendant Selector

- The descendant selector matches all elements that are descendants of a specified element.
- The following example selects all <p> elements inside elements:

Example

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

Child Selector

- The child selector selects all elements that are the immediate children of a specified element.
- The following example selects all <p> elements that are immediate children of a element:

Example

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

Adjacent Sibling Selector

- The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.
- Sibling elements must have the same parent element, and "adjacent" means "immediately following".
- The following example selects all <p> elements that are placed immediately after elements:

Example

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

General Sibling Selector

- The general sibling selector selects all elements that are siblings of a specified element.
- The following example selects all <p> elements that are siblings of elements:

Example

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

16. CSS Pseudo-classes

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 is over it
- Style visited and unvisited links differently

Syntax

- The syntax of pseudo-classes:

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

Anchor Pseudo-classes

- Links can be displayed in different ways:

Example

```
/* unvisited link */
a:link {
    color: #FF0000;
}
/* visited link */
a:visited {
    color: #00FF00;
}
/* mouse over link */
a:hover {
    color: #FF00FF;
}
/* selected link */
a:active {
    color: #0000FF;
}
```

- **Note:** a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!! a:active MUST come after a:hover in the CSS definition in order to be effective!! Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

- Pseudo-classes can be combined with CSS classes:

Example

```
a.highlight: hover
 {
    color: #ff0000;
}
```

- When you hover over the link in the example, it will change color.

CSS - The :first-child Pseudo-class

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

Note: For :first-child to work in IE8 and earlier, a <!DOCTYPE> must be declared

Match the first <p> element

- In the following example, the selector matches any <p> element that is the first child of any element:

Example

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

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

- In the following example, the selector matches the first <i> element in all <p> elements:

Example

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

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

- In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

Example

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

All CSS Pseudo Classes/Elements

Selector	Example	Example description
:link	a:link	Selects all unvisited links
:visited	a:visited	Selects all visited links
:active	a:active	Selects the active link
:hover	a:hover	Selects links on mouse over
:focus	input:focus	Selects the input element which has focus
::first-letter	p::first-letter	Selects the first letter of every <p> element
::first-line	p::first-line	Selects the first line of every <p> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
::before	p::before	Insert content before every <p> element
::after	p::after	Insert content after every <p> element
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"

---

17. CSS Image Opacity / Transparency

- Creating transparent images with CSS is easy.
- The CSS opacity property is a part of the W3C CSS3 recommendation.

Example 1 - Creating a Transparent Image

- The CSS3 property for transparency is **opacity**.
- First we will show you how to create a transparent image with CSS.
- Look at the following CSS:

Example

```
img {
    opacity: 0.4;
    filter: alpha(opacity=40); /* For IE8 and earlier */
}
```

- IE9, Firefox, Chrome, Opera, and Safari use the property **opacity** for transparency. The opacity property can take a value from 0.0 - 1.0. A lower value makes the element more transparent.
- IE8 and earlier use **filter:alpha(opacity=x)**. The x can take a value from 0 - 100. A lower value makes the element more transparent.

Example 2 - Image Transparency - Hover Effect

- Mouse over the images:
- The CSS looks like this:

Example

```
img {
    opacity: 0.4;
    filter: alpha(opacity=40); /* For IE8 and earlier */
}
img:hover {
    opacity: 1.0;
    filter: alpha(opacity=100); /* For IE8 and earlier */
}
```

- The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hover over one of the images. In this case we want the image to NOT be transparent when the user hover over it.

18. CSS 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

Syntax

- The syntax of pseudo-elements:

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

The ::first-line Pseudo-element

- The ::first-line pseudo-element is used to add a special style to the first line of a text.
- The ::first-line pseudo-element can only be applied to block elements.

Example

- Format the first line of the text in p elements:

```
p::first-line {
   color: #ff0000;
   font-variant: small-caps;
}
```

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

The ::first-letter Pseudo-element

- The ::first-letter pseudo-element is used to add a special style to the first letter of a text.
- The ::first-letter pseudo-element can only be applied to block elements.

Example

- Format the first letter of the text in p elements:

```
p::first-letter {
   color: #ff0000;
   font-size: xx-large;
}
```

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and CSS Classes

- Pseudo-elements can be combined with CSS classes:

Example

```
p.intro::first-letter {
   color: #ff0000;
   font-size:200%;
}
```

- The example above will display the first letter of paragraphs with class="intro", in larger size, and red.

Multiple Pseudo-elements

- Several pseudo-elements can also be combined.
- In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

Example

```
p::first-letter {
   color: #ff0000;
   font-size: xx-large;
}
p::first-line {
   color: #0000ff;
   font-variant: small-caps;
}
```

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 each <h1> element:

Example

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

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 each <h1> element:

Example

```
h1::after {
   content: url(smiley.gif
```

---

19. CSS Navigation Bar

- Having easy-to-use navigation is important for any web site.
- With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

- A navigation bar needs standard HTML as a base.
- In our examples we will build the navigation bar from a standard HTML list.

- A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

- Now let's remove the bullets and the margins and padding from the list:

Example

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
}
```

Example explained:

- List-style-type: none - Removes the bullets. A navigation bar does not need list markers
- Setting margins and padding to 0 to remove browser default settings
- The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

Vertical Navigation Bar

- To build a vertical navigation bar we only need to style the <a> elements, in addition to the code above:

Example

```
a {
    display: block;
    width: 60px;
}
```

Example explained:

- display: block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- width: 60px - Block elements take up the full width available by default. We want to specify a 60 px width
- **Note:** Always specify the width for <a> elements in a vertical navigation bar. If you omit the width, IE6 can produce unexpected results.

Horizontal Navigation Bar

- There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.
- Both methods work fine, but if you want the links to be the same size, you have to use the floating method.

Inline List Items

- One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code above:

Example

```
li {
   display: inline;
}
```

Example explained:

- display: inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Floating List Items

- In the example above the links have different widths.
- For all the links to have an equal width, float the <li> elements and specify a width for the <a> elements:

Example

```
li {
   float: left;
}
a {
   display: block;
   width: 60px;
}
```

Example explained:

- float: left - use float to get block elements to slide next to each other
- display: block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- Width: 60px - Since block elements take up the full width available, they cannot float next to each other. We specify the width of the links to 60px

20. CSS Image Gallery

- CSS can be used to create an image gallery.
- The following image gallery is created with CSS:

Example

```
<html>
<head>
<style>
div.img {
   margin: 5px;
   padding: 5px;
   border: 1px solid #0000ff;
   height: auto;
   width: auto;
   float: left;
   text-align: center;
}
div.img img {
   display: inline;
   margin: 5px;
   border: 1px solid #ffffff;
}
div.img a:hover img {
   border:1px solid #0000ff;
```

```
}
div.desc {
   text-align: center;
   font-weight: normal;
   width: 120px;
   margin: 5px;
}
</style>
</head>
<body>
<div class="img">
  <a target="_blank" href="klematis_big.htm">
    <img src="klematis_small.jpg" alt="Klematis" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis2_big.htm">
    <img src="klematis2_small.jpg" alt="Klematis" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis3_big.htm">
    <img src="klematis3_small.jpg" alt="Klematis" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
<div class="img">
  <a target="_blank" href="klematis4_big.htm">
    <img src="klematis4_small.jpg" alt="Klematis" width="110" height="90">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
</body>
</html>
```

21. CSS Image Sprites

- An image sprite is a collection of images put into a single image.
- A web page with many images can take a long time to load and generates multiple server requests.
- Using image sprites will reduce the number of server requests and save bandwidth.

Image Sprites - Simple Example

- Instead of using three separate images, we use this single image ("img_navsprites.gif"):

- With CSS, we can show just the part of the image we need.

- In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:

Example

```
#home {
    width: 46px;
    height: 44px;
    background: url(img_navsprites.gif) 0 0;
}
```

Example explained:

- <img id="home" src="img_trans.gif"> - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- width: 46px; height: 44px; - Defines the portion of the image we want to use
- background: url(img_navsprites.gif) 0 0; - Defines the background image and its position (left 0px, top 0px)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

Image Sprites - Create a Navigation List

- We want to use the sprite image ("img_navsprites.gif") to create a navigation list.
- We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist {
    position: relative;
}
#navlist li {
    margin: 0;
    padding: 0;
    list-style: none;
    position: absolute;
    top: 0;
}
#navlist li, #navlist a {
    height: 44px;
    display: block;
}
#home {
    left: 0px;
    width: 46px;
    background: url('img_navsprites.gif') 0 0;
}
#prev {
    left: 63px;
    width: 43px;
    background: url('img_navsprites.gif') -47px 0;
}
#next {
    left: 129px;
```

```
    width: 43px;
    background: url('img_navsprites.gif') -91px 0;
}
```

Example explained:

- #navlist {position:relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li {margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a {height:44px;display:block;} - the height of all the images are 44px

Now start to position and style for each specific part:

- #home {left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home {background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev {left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev {background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next {left:129px;width:43px;}- Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next {background:url('img_navsprites.gif') -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

- Now we want to add a hover effect to our navigation list.
- **The :hover selector is used to select elements when you mouse over them. Tip:** The :hover selector can be used on all elements, not only on links.
- Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:

- Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.
- We only add three lines of code to add the hover effect:

Example

```
#home a:hover {
    background: url('img_navsprites_hover.gif') 0 -45px;
}
#prev a:hover {
    background: url('img_navsprites_hover.gif') -47px -45px;
}
#next a:hover {
    background: url('img_navsprites_hover.gif') -91px -45px;
}
```

explained:

- #home a:hover {background: transparent url('img_navsprites_hover.gif') 0 -45px;} - For all three hover images we specify the same background position, only 45px further down

22. CSS Attribute Selectors

Style HTML Elements With Specific Attributes

- It is possible to style HTML elements that have specific attributes, not just class and id.
- **Note:** IE7 and IE8 support attribute selectors only if a !DOCTYPE is specified

CSS [attribute] Selector

- The [attribute] selector is used to select elements with the specified attribute.
- The following example selects all <a> elements with a target attribute:

Example

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

CSS [attribute=value] Selector

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

Example

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

CSS [attribute~=value] Selector

- The [attribute~=value] selector is used to select elements with an attribute value containing a specified word.
- The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

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

- The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|=value] Selector

- The [attribute|=value] selector is used to select elements with the specified attribute starting with the specified value.
- The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|="top"] {
  background: yellow;
}
```

CSS [attribute^=value] Selector

- The [attribute^=value] selector is used to select elements whose attribute value begins with a specified value.
- The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example

```
[class^="top"] {
  background: yellow;
}
```

CSS [attribute$=value] Selector

- The [attribute$=value] selector is used to select elements whose attribute value ends with a specified value.
- The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example

```
[class$="test"] {
  background: yellow;
}
```

CSS [attribute*=value] Selector

- The [attribute*=value] selector is used to select elements whose attribute value contains a specified value.
- The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {
  background: yellow;
}
```

Styling Forms

- The attribute selectors can be useful for styling forms without class or ID:

Example

```
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}
input[type="button"] {
  width: 120px;
  margin-left: 35px;
```

```
    display: block;
}
```

23.CSS Media Types

- By using the @media rule, a website can have a different layout for screen, print, mobile phone, tablet, etc.
- Some CSS properties are only designed for a certain media. For example the "voice-family" property is designed for aural user agents. Some other properties can be used for different media types. For example, the "font-size" property can be used for both screen and print media, but perhaps with different values. A document usually needs a larger font-size on a screen than on paper, and sans-serif fonts are easier to read on the screen, while serif fonts are easier to read on paper.

The @media Rule

- The @media rule allows different style rules for different media in the same style sheet.
- The style in the example below tells the browser to display a 14 pixels Verdana font on the screen. But if the page is printed, it will be in a 20 pixels font, and in a red color:

Example

```
@media screen {
    p {
        font-family: verdana,sans-serif;
        font-size: 14px;
    }
}
@media print {
    p {
        font-size: 20px;
        color: red;
    }
}
```

Other Media Types

Media Type	Description
all	Used for all media type devices
aural	Used for speech and sound synthesizers
braille	Used for braille tactile feedback devices
embossed	Used for paged braille printers
handheld	Used for small or handheld devices
print	Used for printers
projection	Used for projected presentations, like slides
screen	Used for computer screens
tty	Used for media using a fixed-pitch character grid, like teletypes and terminals
tv	Used for television-type devices

24.CSS Complete Reference

https://www.w3schools.com/cssref/default.asp