

Cascading Style Sheet

CSS

Content

- Introduction
- CSS Syntax
- CSS Selectors
- `<div>` tag

Introduction

- CSS is the language we use to style a Web page
- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

Why to use CSS

- CSS is used to define
 - Styles for web pages
 - The design, layout and variations in display for different devices and screen sizes
- CSS removed the style formatting from the HTML page
- The style definitions are normally saved in external .css files.
- With an external stylesheet file, you can change the look of an entire website by changing just one file!

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}

</style>
</head>

<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph</p>

</body>
</html>
```

CSS Example

My First CSS Example

This is a paragraph.

CSS Syntax

- A CSS rule consists of a selector and a declaration block.

h1 { color:blue; font-size:12px; }

- The selector points to the HTML element you want to style.
 - h1 is selector
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
 - Properties are color, font-size
 - The values are blue, 12px

Syntax

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

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

</style>
</head>

<body>

<h1> Hello World! </h1>
<p>These paragraphs are styled with CSS</p>

</body>
</html>
```

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value
- text-align is a property, and center is the property value

CSS Syntax

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

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

</style>
</head>

<body>

<p> Hello World! </p>
<p>These paragraphs are styled with CSS</p>

</body>
</html>
```

Hello World!

These paragraphs are styled with CSS.

CSS Selectors

CSS selectors are used to "find" or select the HTML elements we want to style.

CSS selectors are divided into five categories:

1. **Simple selectors:** Select elements based on name, id, class
2. **Combinator selectors:** Select elements based on a specific relationship between them
3. **Pseudo-class selectors:** Select elements based on a certain state
4. **Pseudo-elements selectors:** Select and style a part of an element
5. **Attribute selectors:** Select elements based on an attribute or attribute value

CSS Selectors

The CSS Element Selector

- The element selector selects HTML elements based on the element name.
- Here, all <p> elements on the page will be center-aligned, with a red text color

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

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

</style>
</head>

<body>

<p>Every paragraph will be affected by the
style</p>
<p id = "para1"> Me too !</p>
<p>And me</p>

</body>
</html>
```

Every paragraph will be affected by the style.

Me too!

And me!

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

#para1 {
  color: red;
  text-align: center;
}

</style>
</head>

<body>

<p>This paragraph is not affected by the style</p>
<p id = "para1"> Hello World</p>

</body>
</html>
```

CSS Selectors

The CSS id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.
- An id name can't start with a number

CSS Selectors

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

#para1 {
  color: red;
  text-align: center;
}

</style>
</head>

<body>

<p>This paragraph is not affected by the
style</p>
<p id = "para1"> Hello World</p>

</body>
</html>
```

This paragraph is not affected by the style.

Hello World!

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>

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

</style>
</head>

<body>

<h1 class = "center" > Red and Center aligned
Heading </h1>
<p class = "center" > Red and Center aligned
Paragraph </p>

</body>
</html>
```

CSS Selectors

The CSS class Selector

- The class selector selects HTML elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the class name.
- In this example all HTML elements with class="center" will be red and center-aligned:

Red and center-aligned heading

Red and center-aligned paragraph.

CSS Selectors

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>
```

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

```
</style>
</head>
```

```
<body>
```

```
<h1 class = "center" > This Heading will not be
affected </h1>
```

```
<p class = "center" > Red and Center aligned
Paragraph </p>
```

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

The CSS class Selector

- Only specific HTML elements can be specified that should be affected by a class.
- In this example only <p> elements with class="center" will be red and center-aligned

This heading will not be affected

This paragraph will be red and center-aligned.

```
<!DOCTYPE html>
<html lang = "en-US">
<head>
<style>
```

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

```
p.large {
  font-size: 300%;
}
```

```
</style>
</head>
```

```
<body>
```

```
<h1 class = "center" > This Heading will not be affected </h1>
<p class = "center" > This paragraph will be red and center-
aligned </p>
<p class = "center large" > This paragraph will be red, center-
aligned and in large font-size </p>
```

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

CSS Selectors

The CSS class Selector

- HTML elements can also refer to more than one class.
- In this example the <p> element will be styled according to class="center" and to class="large"
- A class name cannot start with a number!

This heading will not be affected

This paragraph will be red and center-aligned.

This paragraph will be red, center-aligned, and in a large font-size.

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

CSS Selectors

The CSS Universal Selector

- The universal selector (*) selects all HTML elements on the page.
- The CSS rule below will affect every HTML element on the page:

Hello world!

Every element on the page will be affected by the style.

Me too!

And me!


```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```

CSS Selectors

The CSS Grouping Selector

- The grouping selector selects all the HTML elements with the same style definitions.
- In the CSS code, the h1, h2, and p elements have the same style definitions

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

```
<!DOCTYPE html>

<html>

<head>

<style>

h1, h2, p {

    text-align: center;

    color: red;

}

</style>

</head>

<body>


<h1>Hello World!</h1>

<h2>Smaller heading!</h2>


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


</body>

</html>
```

CSS Selectors

The CSS Grouping Selector

- It will be better to group the selectors, to minimize the code.
- 

Hello World!

Smaller heading!

This is a paragraph.

CSS Selectors

All CSS Simple Selectors

Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,... elements	div, p	Selects all <div> elements and all <p>

The <div> tag

- The <div> tag defines a division or a section in an HTML document.
- The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.
- The <div> tag is easily styled by using the class or id attribute.
- Any sort of content can be put inside the <div> tag!
- By default, browsers always place a line break before and after the <div> element.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<style>
```

```
.myDiv {
```

```
  border: 5px outset red;
```

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

```
  text-align: center;
```

```
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1>The div element</h1>
```

```
<div class="myDiv">
```

```
  <h2>This is a heading in a div element</h2>
```

```
  <p>This is some text in a div element.</p>
```

```
</div>
```

```
<p>This is some text outside the div element.</p>
```

```
</body>
```

```
</html>
```

<div> Element

The div element

This is a heading in a div element

This is some text in a div element.

This is some text outside the div element.

Cascading Style Sheet

CSS

Content

- How to insert CSS?

How to insert CSS?

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

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

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

</body>
</html>
```

```
body {
  background-color: lightblue;
}

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

mystyle.css

- With an external style sheet, we can change the look of an entire website by changing just one file
- Each HTML page must include a reference to the external style sheet file inside the <link> element
- It is written inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="mystyle.css">
</head>
<body>

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

</body>
</html>
```

```
body {
  background-color: lightblue;
}

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

mystyle.css

External CSS

- With an external style sheet, we can change the look of an entire website by changing just one file
- Each HTML page must include a reference to the external style sheet file inside the <link> element

This is a heading

This is a paragraph.

Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

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

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

</body>
</html>
```

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

Internal CSS

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: linen;
}

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

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

</body>
</html>
```

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

This is a heading

This is a paragraph.

Inline CSS

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

<h1 style="color:blue;text-align:center;">This is a
heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element.
- The style attribute can contain any CSS property.
- An inline style loses many of the advantages of a style sheet

Inline CSS

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

<h1 style="color:blue;text-align:center;">This is a
heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element.
- The style attribute can contain any CSS property.
- An inline style loses many of the advantages of a style sheet

This is a heading

This is a paragraph.

Multiple Style Sheets

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css"
>
<style>
h1 {
  color: orange;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The style of this document is a combination of an
external stylesheet, and internal style</p>

</body>
</html>
```

```
h1 {
  color: navy;
}
```

mystyle.css

- If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

This is a heading

The style of this document is a combination of an external stylesheet, and internal style

Multiple Style Sheets

- If some properties have been defined for the same selector (element) in different style sheets, the value from the **last read**

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css"
>
</head>
<body>

<h1>This is a heading</h1>
<p>The style of this document is a combination of an
external stylesheet, and internal style</p>

</body>
</html>
```

This is a heading

The style of this document is a combination of an external stylesheet, and internal style

```
h1 {
  color: navy;
}
```

mystyle.css

Cascading Order

If there is more than one style specified for an HTML element then cascading order will be defined

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

An inline style has the highest priority, and will override external and internal styles

Rule Cascading

The cascade takes

- **Input:** An unordered list of declared values for a given property on a given element
- **Process:** It sorts them by their declaration's precedence
- **Output:** a single cascaded value

Cascade Sorting Order

The cascade sorts declarations according to the following criteria, in descending order of priority:

1. Origin and Importance
2. Specificity
3. Order of Appearance

The output of the cascade is a (potentially empty) sorted list of declared values for each property on each element.

Cascading Origin

Each style rule has a cascade origin, which determines where it enters the cascade.

CSS defines three core origins:

1. Author Origin

- The author specifies style sheets for a source document according to the conventions of the document language.
- For instance, in HTML, style sheets may be included in the document or linked externally.

2. User Origin

- The user may be able to specify style information for a particular document.
- Example 1: the user may specify a file that contains a style sheet
- Example 2: the user agent may provide an interface that generates a user style sheet (or behaves as if it did).

3. User-Agent Origin

- Conforming user agents must apply a default style sheet (or behave as if they did).
- A user agent's default style sheet should present the elements of the document language in

ways that satisfy general presentation expectations for the document language.

Cascading Origin

Extensions to CSS define the following additional origins:

4. Animation Origin

CSS Animations [css-animations-1] generate “virtual” rules representing their effects when running.

5. Transition Origin

Like CSS Animations, CSS Transitions generate “virtual” rules representing their effects when running.

Important Declarations

- It is mentioned using the **!important** annotation
- CSS attempts to create a balance of power between author and user style sheets.
- By default, rules in an author's style sheet override those in a user's style sheet, which override those in the user-agent's default style sheet.
- For balancing overriding mechanism, a declaration can be marked important
- It increases its weight in the cascade and inverts the order of precedence.
- A declaration is important if it has a !important annotation
- If the last two (non-whitespace, non-comment) tokens in its value are the delimiter token ! followed by the identifier token important.
- All other declarations are normal (non-important).

Example:

```
[hidden] { display: none !important; }
```

Important Declarations

- An important declaration takes precedence over a normal declaration.
- Author and user style sheets may contain important declarations, with user-origin important declarations overriding author-origin important declarations.
- This CSS feature improves accessibility of documents by giving users with special requirements (large fonts, color combinations, etc.) control over presentation.
- Important declarations from all origins take precedence over animations.
- This allows authors to override animated values in important cases. (Animated values normally override all other rules.)
- User-agent style sheets may also contain important declarations. These override all author and user

Example

```
/* From the user's style sheet */
```

```
p { text-indent: 1em !important }
```

```
p { font-style: italic !important }
```

```
p { font-size: 18pt }
```

```
/* From the author's style sheet */
```

```
p { text-indent: 1.5em !important }
```

```
p { font: normal 12pt sans-serif !important }
```

```
p { font-size: 24pt }
```

Property	Winning value
text-indent	1em
font-style	italic
font-size	12pt
font-family	sans-serif

Specificity

- The Selectors describes how to compute the specificity of a selector.
- Each declaration has the same specificity as the style rule it appears in.
- For the purpose of this step, declarations that do not belong to a style rule (such as the **contents of a style attribute**) are considered to have a **specificity higher than any selector**.
- The declaration with the highest specificity wins.

Highest to lowest specificity are

1. ID selectors
2. Class and pseudo-class selectors
3. Descendant and type selectors (the more element type names, the more specific)
4. Universal selectors

Order of Appearance

The last declaration in document order wins.

For this purpose:

- Declarations from imported style sheets are ordered as if their style sheets were substituted in place of the @import rule.
- Declarations from style sheets independently linked by the originating document are treated as if they were concatenated in linking order, as determined by the host document language.
- Declarations from style attributes are ordered according to the document order of the element the style attribute appears on, and are all placed after any style sheets.

Example

Here the style rulesets are effectively in the order

- p { color:red }
- p { color:blue }
- p { color:green }
- p { color:yellow }

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

<style type="text/css">
p { color:red }
</style>

<link rel="stylesheet" type="text/css" href="imp1.css" />

<style type="text/css">
p { color:yellow }
</style>

</head>
<body>

<h1>This is a heading</h1>
<p>The style of this document is a combination of an external
stylesheet, and internal style</p>

</body>
</html>
```

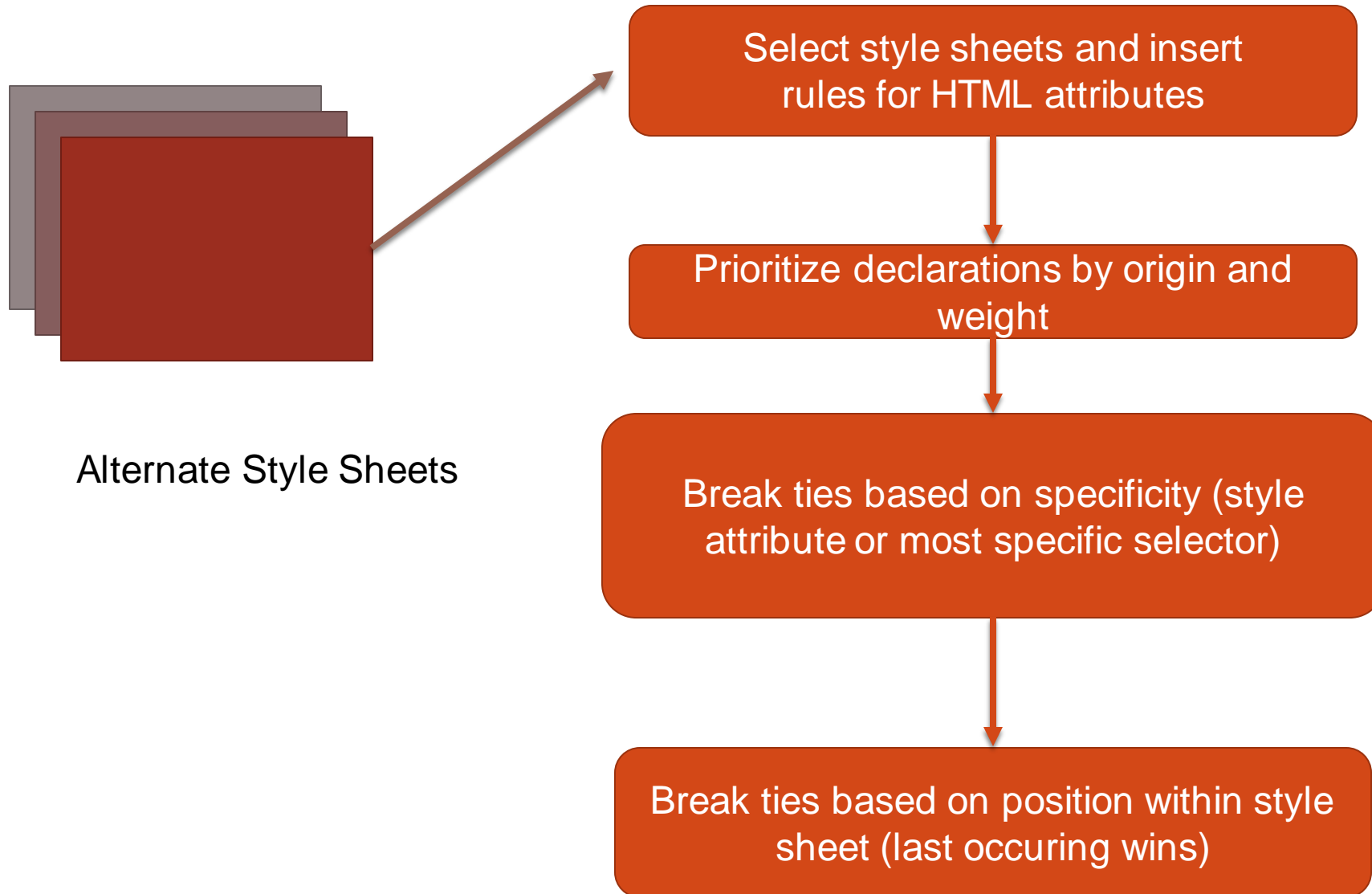
```
@import url("imp2.css");
p { color:green }
```

imp1.css

```
p { color:blue }
```

imp2.css

Rule Cascading



Style Inheritance

- Inheritance is based on the tree structure of the document itself.
- An element inherits a value for one of its properties from its parent's value.
- The parent may in turn inherit its property value from its parent, and so on.

The searching mechanism can be described as:

- To inherit a property value, an element will **search upward** through its tree of ancestor elements
- The searching begin with its parent and ending either at the root html element or at the first element that has a value for the property.
- If the search ends at an element with a value for the property, that value will be used as its property value.
- If no ancestor element has a value for the property, then value specified for each property by the **CSS specification will be used (property's initial value)**

```

<!DOCTYPE html>
<html>
<head>

<title>
Inherit.html
</title>

<style type="text/css">
body { font-weight:bold }
li { font-style:italic }
p { font-size:larger }
span { font-weight:normal }
</style>

</head>

<body>
<ul>
    <li>List item outside and <span>inside </span> a span.
        <p>Embedded paragraph outside and <span>inside</span> a span.
    </p>
    </li>
</ul>
</body>
</html>

```

Style Inheritance

- **For span element**, font-weight is specified by an author rule
- Elements inside **body element**: font-weight property value will be inherited from body element
- span element do inherit other properties from body element (Other than font-weight)
- The first span inherits italicization from its parent li element
- The second span inherits a larger font size from its p element parent and italicization from its li element grandparent.

Style Inheritance

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

<title>
Inherit.html
</title>

<style type="text/css">
body { font-weight:bold }
li { font-style:italic }
p { font-size:larger }
span { font-weight:normal }
</style>

</head>

<body>
<ul>
    <li>List item outside and <span>inside</span> a span.
        <p>Embedded paragraph outside and <span>inside</span> a
span.</p>
    </li>
</ul>
</body>
</html>
```

- *List item outside and inside a span.*

Embedded paragraph outside and inside a span.

Cascading Style Sheet

CSS

Content

- Background
- Border Images
- Color
- shadows

CSS Background

The CSS background properties are used to add background effects for elements.

Different Properties:

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

CSS background-color

- The background-color property specifies the background color of an element.
- With CSS, a color is most often specified by:
 1. a valid color name - like "red"
 2. a HEX value - like "#ff0000"
 3. an RGB value - like "rgb(255,0,0)"

Example:

```
body {  
    background-color: lightblue;  
}
```

Opacity/Transparency

- The opacity property specifies the opacity/transparency of an element.
- It can take a value from 0.0 - 1.0.
- The lower value, the more transparent

Example:

```
div {  
  background-color: green;  
  opacity: 0.3;  
}
```

- When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency.
- This can make the text inside a fully transparent element hard to read.

Transparency using RGBA

- If we do not want to apply opacity to child elements use RGBA color values.
- The following example sets the opacity for the background color and not the text:

```
div {  
    background: rgba(0, 128, 0, 0.3)    /* Green background with 30% opacity */  
}
```

- In addition to RGB, we can use an RGB color value with an alpha channel (RGBA) - which specifies the opacity for a color.
- An RGBA color value is specified with: `rgba(red, green, blue, alpha)`.
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

CSS 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.

Example:

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

- The background image can also be set for specific elements, like the <p> element

Example:

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

CSS background-repeat

- By default, the background-image property repeats an image both horizontally and vertically.
- Some images should be repeated only horizontally or vertically
- To repeat image only horizontally set the value as:
background-repeat: repeat-x;
- To repeat image only vertically set the value as:
background-repeat: repeat-y;
- Showing the background image only once is also specified
background-repeat: no-repeat;

CSS background-position

- The background-position property is used to specify the position of the background image.

- Example:

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


CSS background-attachment

- The background-attachment property specifies whether the background image should scroll or be fixed
- If it is fixed then it will not scroll with the rest of the page
- Example:

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

CSS background shorthand property

- To shorten the code, it is also possible to specify all the background properties in one single property.
- This is called a shorthand property.
- 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 other

CSS background shorthand property

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

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

CSS border-image property

- The border-image property allows you to specify an image to be used as the border around an element.
- The border-image property is a shorthand property for:
 - border-image-source
 - border-image-slice
 - border-image-width
 - border-image-outset
 - border-image-repeat
- Omitted values are set to their default values.

CSS border-image property

- Property Values

Value	Description
border-image-source	The path to the image to be used as a border
border-image-slice	How to slice the border image
border-image-width	The width of the border image
border-image-outset	The amount by which the border image area extends beyond the border box
border-image-repeat	Whether the border image should be repeated, rounded or stretched
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element. Read about inherit

CSS Colors

Colors are specified using

- predefined color names
- RGB
- HEX
- HSL
- RGBA
- HSLA

RGB Colors

- An RGB color value represents **RED, GREEN and BLUE** light sources.
- In CSS, a color can be specified as an RGB value, using this formula:

rgb (red, green, blue)

- Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example:

- rgb (255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.
- To display black, set all color parameters to 0, like this: rgb(0, 0, 0).
- To display white, set all color parameters to 255, like this: rgb(255, 255, 255).
- Shades of gray are often defined using equal values for all the 3 light sources

RGBA values

- RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
- An RGBA color value is specified with:
`rgba(red, green, blue, alpha)`
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all)

HEX Colors

- A hexadecimal color is specified with: #RRGGBB,
- where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.
- In CSS, a color can be specified using a hexadecimal value in the form:

#rrggbb

Here rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example,

- **#ff0000** is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).
- Shades of gray are often defined using equal values for all the 3 light sources

HEX Colors

3 Digit HEX value

- Sometimes you will see a 3-digit hex code in the CSS source.
- The 3-digit hex code is a shorthand for some 6-digit hex codes.
- The 3-digit hex code has the following form:

`#rgb`

Where r, g, and b represents the red, green, and blue components with values between 0 and f.

- The 3-digit hex code can only be used when both the values (RR, GG, and BB) are the same for each components.
- So, if we have `#ff00cc` it can be written like this: `#f0c`

HSL Colors

- HSL stands for **hue, saturation, and lightness**.

HSL Value

- In CSS, a color can be specified using hue, saturation, and lightness (HSL)

in the form:

`hsl(hue, saturation, lightness)`

- Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

- Saturation is a percentage value. 0% means a shade of gray, and 100% is

HSL Colors

Saturation

- Saturation can be described as the **intensity of a color**.
- 100% is pure color, no shades of gray
- 50% is 50% gray, but you can still see the color.
- 0% is completely gray, you can no longer see the color.

Lightness

- The lightness of a color can be described as **how much light you want to give the color**,
- where 0% means no light (black),
- 50% means 50% light (neither dark nor light)
- 100% means full lightness (white)
- Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

HSL Colors

HSLA Value

- HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.
- An HSLA color value is specified with:

`hsla(hue, saturation, lightness, alpha)`

- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

Cascading Style Sheet

CSS

Content

- Shadow
- Text
- Transformation
- Transition
- Animation

CSS Shadow

With CSS shadow can be added to text and to elements.

We will learn two type of shadow effects:

- text-shadow
- box-shadow

CSS Text Shadow

```
<!DOCTYPE html>
<html>
<head>
<style>

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

</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

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

Text-shadow effect!

CSS Text Shadow

- Adding Color to shadow

```
<!DOCTYPE html>
<html>
<head>
<style>

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

</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

Text-shadow effect!

CSS Text Shadow

```
<!DOCTYPE html>
<html>
<head>
<style>

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

</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

- Adding blur effect and Color to shadow

Text-shadow effect!

CSS Text Shadow

- Multiple Shadows

To add more than one shadow to the text, you can add a comma-separated list of shadows

Text-shadow effect!

```
<!DOCTYPE html>
<html>
<head>
<style>

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

</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

CSS Text Shadow

```
<!DOCTYPE html>
<html>
<head>
<style>

h1 {
  color: yellow;
  text-shadow: -1px 0 black, 0 1px black, 1px 0
black, 0 -1px black;
}

</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```

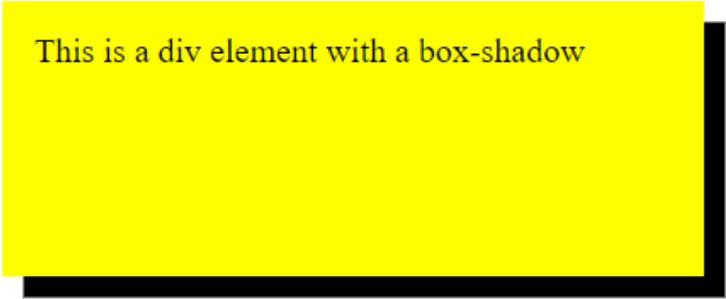
- Create plain border using text-shadow property

Border around text!

CSS BOX Shadow

- The CSS box-shadow property applies shadow to elements.
- In its simplest use, you only specify the horizontal shadow and the vertical

The box-shadow Property



This is a div element with a box-shadow

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: yellow;
  box-shadow: 10px 10px;
}
</style>
</head>
<body>

<h1>The box-shadow Property</h1>

<div>This is a div element with a box-shadow</div>

</body>
</html>
```

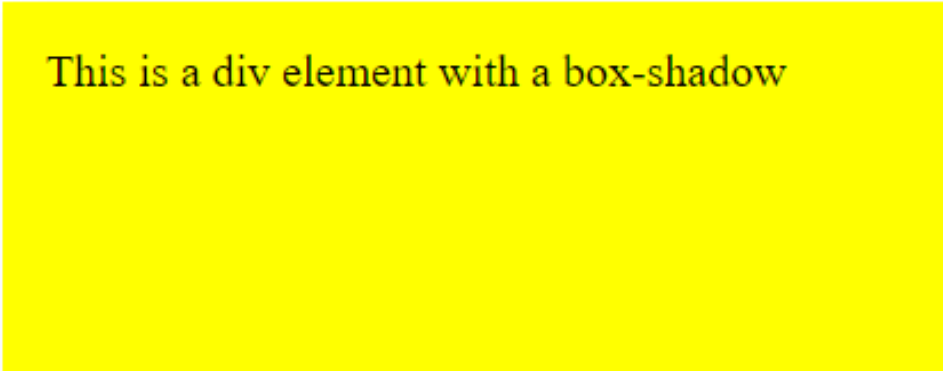
CSS BOX Shadow

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: yellow;
  box-shadow: 10px 10px 20px blue;
}
</style>
</head>
<body>

<div>This is a div element with a box-shadow</div>

</body>
</html>
```

- Add color and blur effect to shadows

A yellow rectangular box with a blue box shadow. The shadow is a blurred blue rectangle positioned below and to the right of the yellow box. The text "This is a div element with a box-shadow" is written in black inside the yellow box.

This is a div element with a box-shadow

```

<!DOCTYPE html>
<html>
<head>
<style>
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;
}
div.header {
  background-color: #4CAF50;
  color: white;
  padding: 10px;
  font-size: 40px;
}
}
div.container {
  padding: 10px;
}
}
</style>
</head>
<body>
<h2>Cards</h2>
<p>The box-shadow property can be used to create paper-like cards:</p>
<div class="card">
  <div class="header">
    <h1>1</h1>
  </div>
  <div class="container">
    <p>January 1, 2016</p>
  </div>
</div>
</body>
</html>

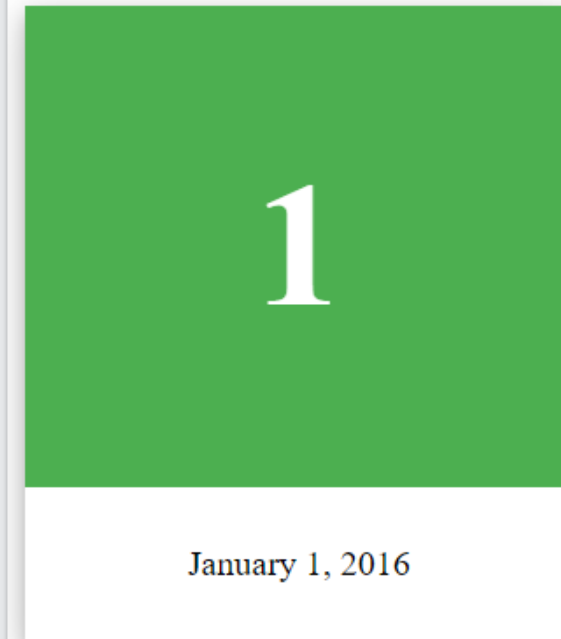
```

CSS BOX Shadow

- Box-shadow property can be used to create paper-like cards

Cards

The box-shadow property can be used to create paper-like cards:



CSS Text

CSS has a lot of properties for formatting text.

- Color
 - To specify the color of text: **color**
 - To specify the color of background: **background-color**
- Alignment:
 - The **text-align** property: set the horizontal alignment of a text
 - Possible values of text-align property: center, left, right, justify
 - For **justify** property, each line is stretched so that every line has equal width, and the left and right margins are straight
 - The **vertical-align** property: Sets the vertical alignment of an element
 - Possible Values: baseline, text-top, text-bottom, sub, super
- Decoration
 - The **text-decoration** property is used to set or remove decorations from text.
 - The value text-decoration: none; is often used to remove underlines from links
 - Possible values are: overline, line-through, underline

CSS Text

CSS has a lot of properties for formatting 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
 - Possible values: **uppercase, lowercase, capitalize**
- Spacing
 - The **text-indent** property: To specify the indentation of the first line of a text
 - The **letter-spacing** property: To specify the space between the characters in a text.
 - The **line-height** property: To specify the space between lines
 - The **word-spacing** property: To specify the space between the words in a text(Values: wrap/nowrap)

CSS 2D Transformation

CSS transforms allow to move, rotate, scale, and skew elements.

The CSS transform property can use the following 2D transformation methods

1. The `translate()` method

It moves an element from its current position according to the parameters given for the X-axis and the Y-axis

```
div {  
  transform: translate(50px, 100px);  
}
```

2. The `rotate()` method

It rotates an element clockwise or counter-clockwise according to a given degree

```
div {  
  transform: rotate(20deg);  
}
```

Using negative values will rotate the element counter-

CSS 2D Transformation

3. The scale() method

- It increases or decreases the size of an element (according to the parameters given for the width and height).
- The example increases the <div> element to be two times of its original width, and three times of its original height

```
div {  
    transform: scale(2, 3);  
}
```

```
div {  
    transform: scaleX(2);  
}
```

4. The scaleX() method

The scaleX() method increases or decreases the width of an element.

```
div {  
    transform: scaleY(2);  
}
```

CSS 2D Transformation

6. The skewX() method

It skews an element along the X-axis by the given angle.

```
div {  
    transform: skewX(20deg);  
}
```

7. The skewY() method

It skews an element along the Y-axis by the given angle.

```
div {  
    transform: skewY(20deg);  
}
```

8. The skew() method

```
div {  
    transform: skew(20deg, 10deg);  
}
```

CSS 3D Transformation

The CSS transform property you can use the following 3D transformation methods:

- rotateX(): rotates an element around its X-axis at a given degree
- rotateY(): rotates an element around its Y-axis at a given degree
- rotateZ(): rotates an element around its Z-axis at a given degree

```
div {  
    transform: rotateX(20deg);  
}
```

```
div {  
    transform: rotateY(20deg);  
}
```

```
div {  
    transform: rotateZ(20deg);  
}
```

CSS Transition

- CSS transitions allows you to change property values smoothly, over a given duration.
- To create a transition effect, you must specify two things:
 - The CSS property you want to add an effect to
 - The duration of the effect
- If the duration part is not specified, the transition will have no effect, because the default value is 0.

Example:

- The transition effect will start when the specified CSS property (width) changes value.
- specify a new value for the width property when a user mouses over the <div> element
- When the cursor mouses out of the element, it will gradually change back to its original style.

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s;  
}
```

```
div:hover {  
  width: 300px;  
}
```

Change Several Property Values:

The example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height:

```
div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s height 4s;  
}
```


Speed Curve of Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- **ease** - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- **linear** - specifies a transition effect with the same speed from start to end
- **ease-in** - specifies a transition effect with a slow start
- **ease-out** - specifies a transition effect with a slow end
- **ease-in-out** - specifies a transition effect with a slow start and end
- **cubic-bezier(n,n,n,n)** - lets you define your own values in a cubic-bezier function

The transition-delay property specifies a delay (in seconds) for the transition effect.

CSS Animation

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify **some keyframes** for the animation.
- **Keyframes** hold what styles the element will have at certain times.

```
/* The animation code */
```

```
@keyframes example {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}
```

```
/* The element to apply the animation to */
```

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

CSS Animation

- When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.
- To get an animation to work, you must bind the animation to an element.
- The example binds the "example" animation to the <div> element.
- The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element

from "red" to "yellow".

```
/* The animation code */
```

```
@keyframes example {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}
```

```
/* The element to apply the animation to */
```

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

CSS Animation

- The animation-duration property defines how long an animation should take to complete.
- If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).
- In the example we have specified when the style will change by using the keywords "from" and "to" which represents 0% (start) and 100%

CSS Animation

```
/* The animation code */
```

```
@keyframes example {  
  0%   {background-color: red;}  
  25%  {background-color: yellow;}  
  50%  {background-color: blue;}  
  100% {background-color: green;}  
}
```

```
/* The element to apply the animation to */
```

```
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

- The example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

CSS Animation

```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: 2s;  
  animation-iteration-count: 3;  
}
```

Delay in animation

- The animation-delay property specifies a delay for the start of an animation.
- Negative values are also allowed.
- If using negative values, the animation will start as if it had already been playing for N seconds.

Animation Count

The animation-iteration-count property specifies the number of times an

Run Animation in Reverse Direction or Alternate Cycles

The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- **normal** - The animation is played as normal (forwards). This is default
- **reverse** - The animation is played in reverse direction (backwards)
- **alternate** - The animation is played forwards first, then backwards
- **alternate-reverse** - The animation is played backwards first, then forwards

Speed Curve of the Animation

The animation-timing-function property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- **ease** - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- **linear** - Specifies an animation with the same speed from start to end
- **ease-in** - Specifies an animation with a slow start
- **ease-out** - Specifies an animation with a slow end

- **ease-in-out** - Specifies an animation with a slow start and end

Fill-mode for the Animation

- CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played.
- The **animation-fill-mode** property can override this behavior.
- The **animation-fill-mode** property specifies a style for the target element when the animation is not playing before it starts, after it ends, or both
- The animation-fill-mode property can have the following values:
 - **none** - Default value. Animation will not apply any styles to the element before or after it is executing
 - **forwards** - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
 - **backwards** - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period

CSS Animation shorthand property

```
div {
```

```
  animation-name: example;
```

```
  animation-duration: 5s;
```

```
  animation-timing-function:
```

```
linear;
```

```
  animation-delay: 2s;
```

```
  animation-iteration-count:
```

```
infinite;
```

```
div {
```

```
  animation: example 5s linear 2s infinite
```

```
  alternate;
```

```
}
```