

DO NOT WEAR SCENTED PRODUCTS!

INFX 1606

INTRODUCTION TO
WEBSITE CREATION

HTML Comments

Raghav V. Sampangi

Instructor, Faculty of Computer Science, Dalhousie University

raghav@cs.dal.ca

HTML Comments

```
<html> <!-- An HTML comment -->
  <head> <!-- Another HTML comment -->
    <title>HTML</title>
  </head>
  <!-- Text inside a comment is not rendered on the web page. -->
  <body>
    <p>Paragraph text.</p>
  </body>
</html>
```

It's all about style! Cascading Style Sheets (CSS)

Raghav V. Sampangi

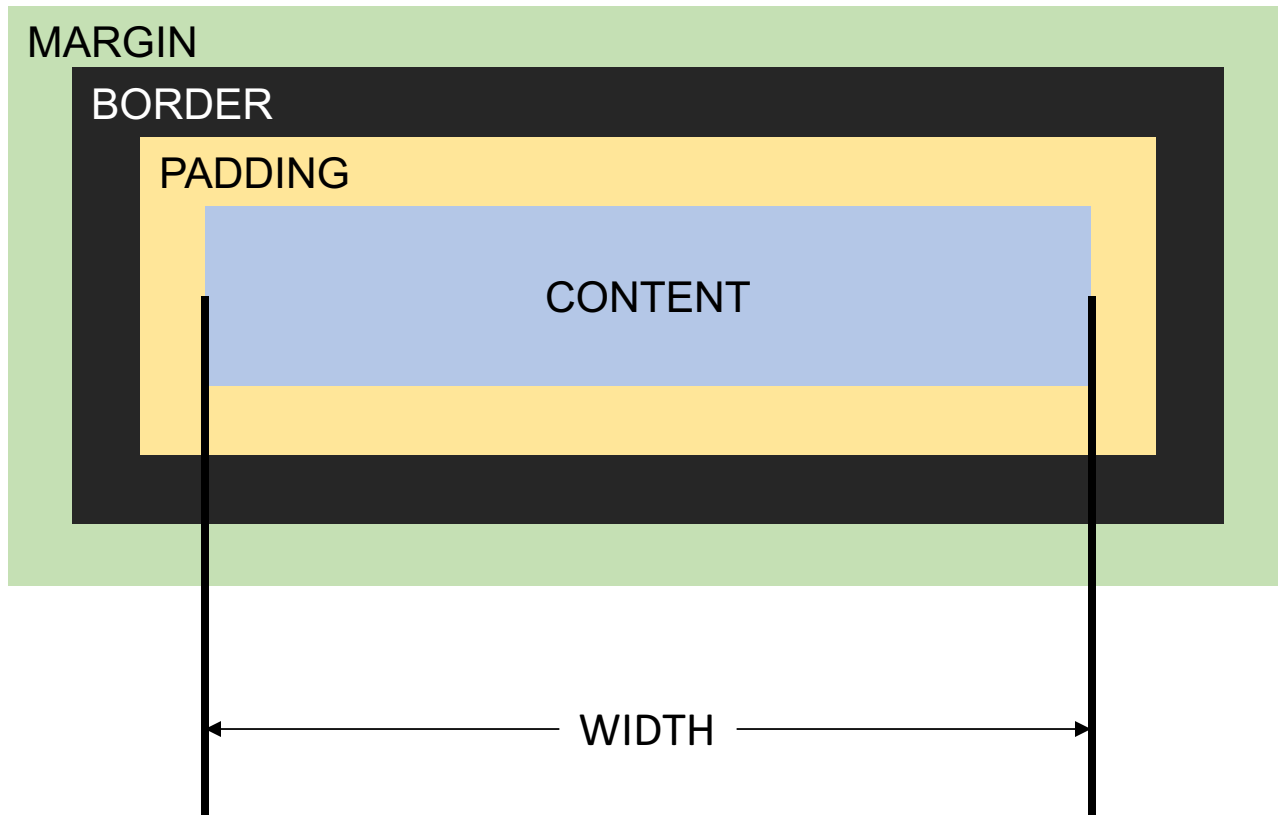
Instructor, Faculty of Computer Science, Dalhousie University

raghav@cs.dal.ca

The Box Model

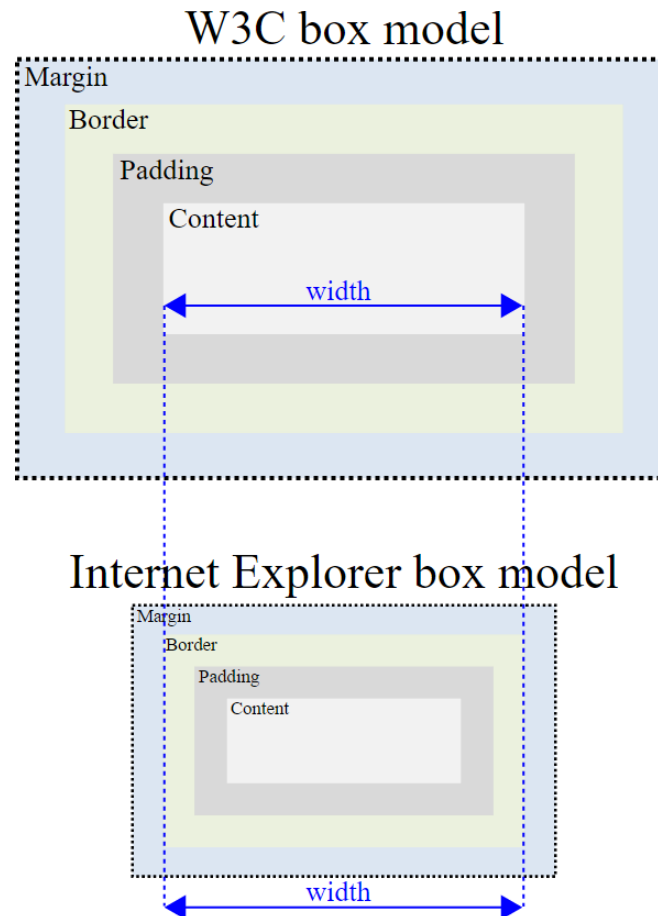
Every HTML element lives inside a box.

The “box” itself has several characteristics (*the W3C box model*):



The Box Model

The reason why Internet Explorer (version 8 and earlier) has issues with CSS, is because it did not conform to the W3C box model! IE had its own box model!



More info: https://en.wikipedia.org/wiki/Internet_Explorer_box_model_bug

The Box Model

Computing absolute dimensions (using the W3C box model)

If you want the content in a container (say, section) to have a **width of 480px**, with a padding of 10px on the left and right, and a border of 2px, with no margin, then, the total width will be:

480px	(content width)
+ 20px	(10 + 10 px padding on left and right)
+ 4px	(2 + 2 px border on left and right)
+ 0px	(0 px margin)

504px	(total width)

Note: We have not considered the top/bottom dimensions because we are just computing the width. If we are computing the height of the section, we will use those padding/border/margins as well.

Introducing CSS

CSS is a style sheet language, which helps express the presentation of structured documents, i.e. containing some standard way of presenting the structure and content of a document.

CSS is used to style content presented in HTML documents.

CSS is also used style content presented in other markup languages such as XHTML, SVG, etc.

Why is it called “style sheet”?

A style sheet is a collection of style rules.

There are several rules that are defined in CSS, which you can use to customize your web page.

Introducing CSS: Syntax of CSS Rules

```
selector
{
    property1: value1;

    property2: value2;
    /* this is a comment */
}
```

Introducing CSS: Syntax of CSS Rules

Selector.

Indicates the element to which the style rule applies.

selector

{

property1: value1;

property2: value2;

/ this is a comment */*

}

Declaration.

Indicates the way in which the elements in the selector should be styled.

Each declaration has a **style property** and a **value**, separated by a colon.

Introducing CSS: Syntax of CSS Rules

Selector.

Indicates the element to which the style rule applies.

selector

Property.

Indicates the aspects of the element that you intend to style or change.

property1: **value1;**

property2: **value2;**

/ this is a comment */*

}

Declaration.

Indicates the way in which the elements in the selector should be styled. Each declaration has a **style property** and a **value**, separated by a colon.

Value.

Indicates the settings for the chosen property.

Tags, attributes, styles, properties and values!

Too much jargon!!

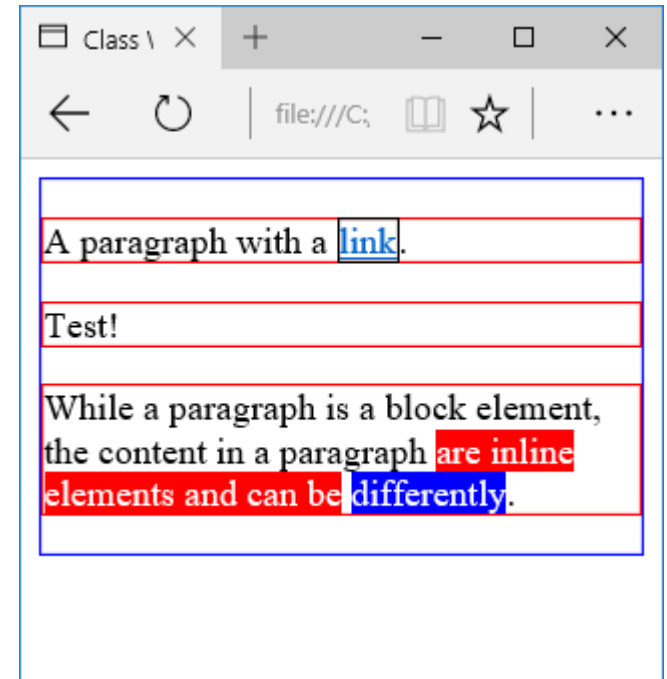
- In HTML:
 - An element has tags (opening and closing, most of the time and some are empty elements);
 - tags can have attributes (like ID, width and height, etc.), and can also have style definition (using style attribute).
- A style definition/declaration:
 - Has a property that you want to change, and,
 - How much or how you want to change it.
 - (Remember to always include semi-colons at the end of style definitions!)

Introducing CSS

Recall that *each HTML element “lives” inside its own box.*

When you apply any style on the HTML element, you are treating it as a box, and applying the style on that box (and all the contents of the box).

```
<body style="border: 1px solid Blue;">
  <p style="border: 1px solid Red;">
    A paragraph with a <a href=""
      style="border: 1px solid
        black;">link</a>.</p>
  <p style="border: 1px solid Red;">Test!</p>
  <p style="border: 1px solid Red;">
    While a paragraph is a block element, the
    content in a paragraph <span
      style="background-color: Red; color:
        White;">are inline elements and can
    be</span> <span style="background-color:
      Blue; color: White;"> differently
    </span>.</p>
</body>
```



Introducing CSS

Recall that ***each HTML element “lives” inside its own box.***

When you apply any style on the HTML element, you are treating it as a box, and applying the style on that box (and all the contents of the box).

Three ways of styling elements in a document:

- Using ***style*** attribute in the element: *only applicable to that element.*
- Defining a **<style>** section in the page head, and defining styles for individual elements or setting default styles for all elements.
- Including the style definitions in an external file.

Introducing CSS

Using ***style*** attribute in the element

Benefits:

- You can style any element independently
- Control over each element

Limitations:

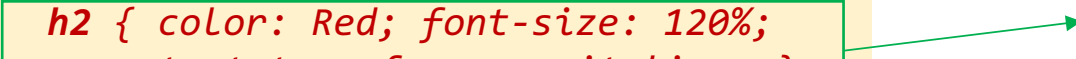
- Results in “messy” looking markup
- Making changes is cumbersome

```
<body style="border: 1px solid Blue;">
  <p style="border: 1px solid Red;">
    A paragraph with a <a href=""
      style="border: 1px solid
        black;">link</a>.</p>
  <p style="border: 1px solid Red;">Test!</p>
</body>
```

Introducing CSS

Defining a **<style>** section in the page head, and defining styles for individual elements or setting default styles for all elements.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      h2 { color: Red; font-size: 120%;
          text-transform: capitalize; }
      strong { color: Blue; font-style:
               italic; border-bottom: 1px
               solid #f2ac00; }
    </style>
  </head>
  <body>
    <h2>A heading</h2>
    <p>This is a simple paragraph. The
    content here doesn't matter, however, our
    focus is on using <strong>CSS</strong> for
    styling.</p>
  </body>
</html>
```

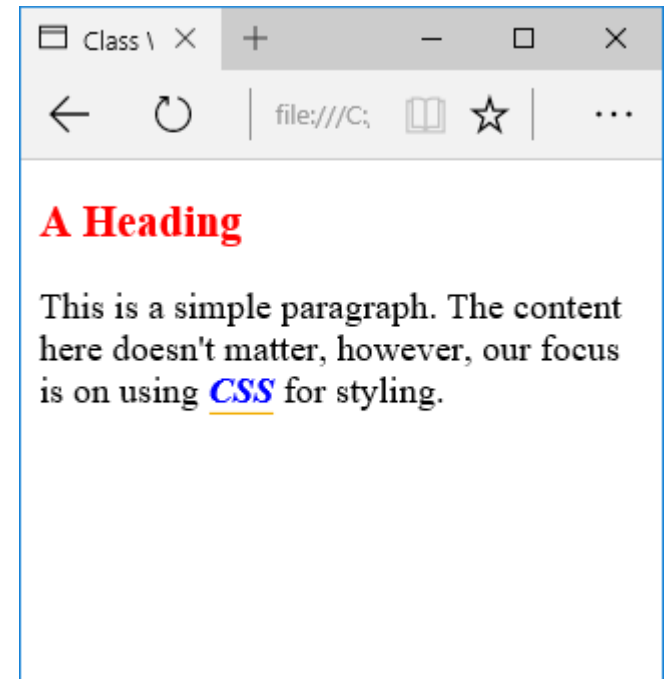


```
selector
{
  property1: value1;
  property2: value2;
  /* this is a comment */
}
```


Introducing CSS

Defining a **<style>** section in the page head, and defining styles for individual elements or setting default styles for all elements.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      h2 { color: Red; font-size: 120%;
          text-transform: capitalize; }
      strong { color: Blue; font-style:
              italic; border-bottom: 1px
              solid #f2ac00; }
    </style>
  </head>
  <body>
    <h2>A heading</h2>
    <p>This is a simple paragraph. The
    content here doesn't matter, however, our
    focus is on using <strong>CSS</strong> for
    styling.</p>
  </body>
</html>
```



Introducing CSS

Defining a **<style>** section in the page head, and defining styles for individual elements or setting default styles for all elements.

Benefits:

- You can style any element independently
- You can set “global” styles for elements
(e.g. all <p> elements can be set have 90% of normal font size)
- Makes markup look less “messy”

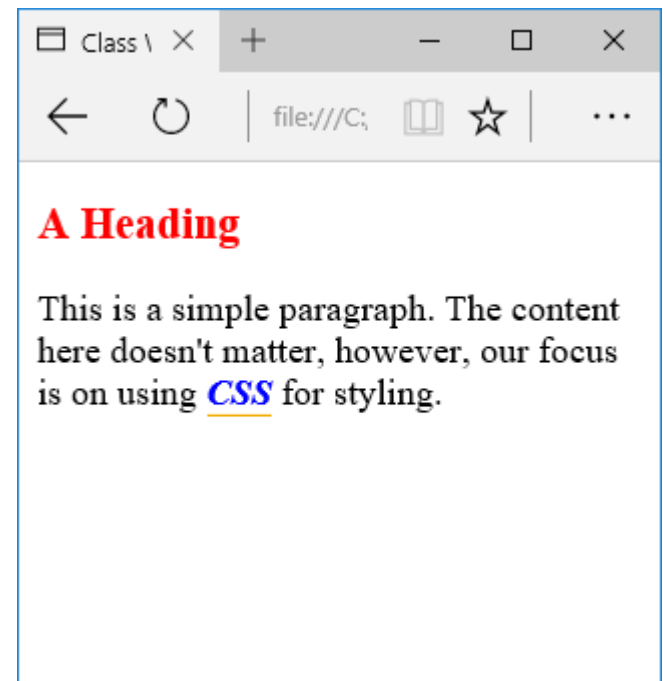
Limitations:

- Introduces redundancy – if you want to re-use same styles in other pages of the same website, you must “copy-paste” same <style> element in all pages!

Introducing CSS

Including the style definitions in an external file (importing styles from an external file).


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <link rel="stylesheet" href="styles.css" >
  </head>
  <body>
    <h2>A heading</h2>
    <p>This is a simple paragraph. The content
here doesn't matter, however, our focus is on
using <strong>CSS</strong> for styling.</p>
  </body>
</html>
```




The <link> element

The <link> element (empty element):


```
<link href="resource url" rel="resource_type" type="content_type">
```



*URL of the resource;
can be absolute or
relative path*



*Type of resource being linked;
e.g. "stylesheet", "icon"*



*MIME type, indicating
content type.
e.g. "text/css"*

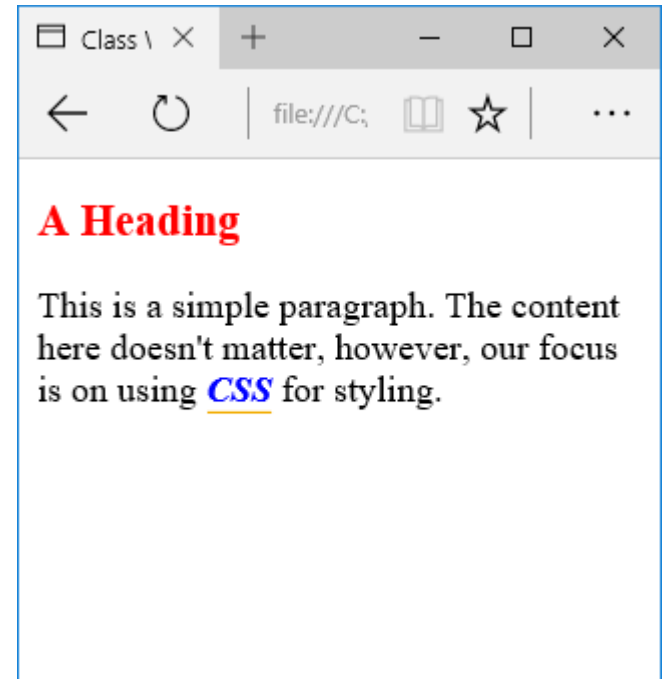
Example:

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

Introducing CSS

Back to our previous example... with the `<style>` element.

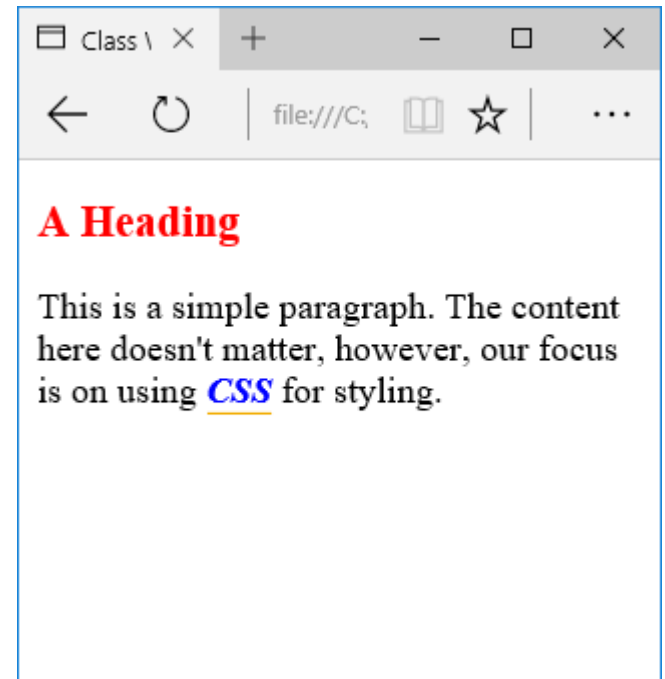
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      h2 { color: Red; font-size: 120%;
          text-transform: capitalize; }
      strong { color: Blue; font-style:
              italic; border-bottom: 1px
              solid #f2ac00; }
    </style>
  </head>
  <body>
    <h2>A heading</h2>
    <p>This is a simple paragraph. The
    content here doesn't matter, however, our
    focus is on using <strong>CSS</strong> for
    styling.</p>
  </body>
</html>
```



Introducing CSS

Including the style definitions in an external file.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <link rel="stylesheet" href="styles.css" >
  </head>
  <body>
    <h2>A heading</h2>
    <p>This is a simple paragraph. The content
here doesn't matter, however, our focus is on
using <strong>CSS</strong> for styling.</p>
  </body>
</html>
```



Contents of **styles.css**:

```
h2 { color: Red; font-size: 120%; text-transform: capitalize; }
strong { color: Blue; font-style: italic; border-bottom: 1px solid #f2ac00; }
```

Introducing CSS

Including the style definitions in an external file.

Benefits:

- You can style any element independently
- You can set “global” styles for elements
(e.g. all <p> elements can be set have 90% of normal font size)
- Makes markup look less “messy”
- You can ***set styles for all pages in your website in one file!***
(and re-use the file as and when necessary)

Limitations:

- Page rendering may be delayed since the CSS file has to be downloaded first, and styles applied next

How are page styles affected when...

...we have included an external style sheet AND a separate <style> element in the page head?

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <link rel="stylesheet" href="styles1.css" >
    <style>
      h2 { color: Blue; font-size: 150%; }
    </style>
  </head>
  <body>
    <h2>Heading with styles</h2>
    <h2 style="color: Brown;">Here's another
heading</h2>
  </body>
</html>
```


How are page styles affected when...

...we have included an external style sheet AND a separate <style> element in the page head?

The way page is rendered depends on the position of the <style> and <link> elements.

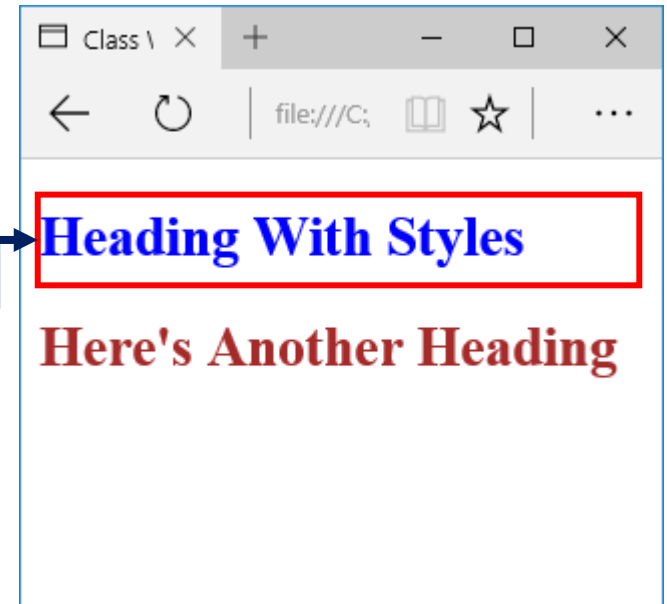
If <link> is placed **before** <style> in the markup, then the style definitions in the <style> element override the definitions in the external style sheet.

If <link> is placed **after** <style> in the markup, then the style definitions in the external style sheet override the definitions in the <style> element.

What happens when...

<link> is placed **before** <style> in the markup:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <link rel="stylesheet" href="styles1.css" >
    <style>
      h2 { color: Blue; font-size: 150%; }
    </style>
  </head>
  <body>
    <h2>Heading with styles</h2>
    <h2 style="color: Brown;">Here's another
heading</h2>
  </body>
</html>
```



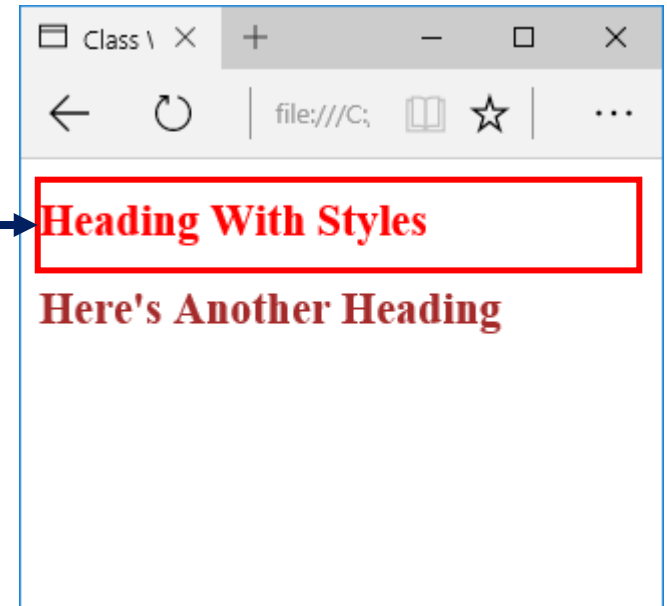
Contents of styles1.css:

```
h2 { color: Red; font-size: 120%; }
```

What happens when...

<link> is placed *after* <style> in the markup:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      h2 { color: Blue; font-size: 150%; }
    </style>
    <link rel="stylesheet" href="styles1.css" >
  </head>
  <body>
    <h2>Heading with styles</h2>
    <h2 style="color: Brown;">Here's another
heading</h2>
  </body>
</html>
```



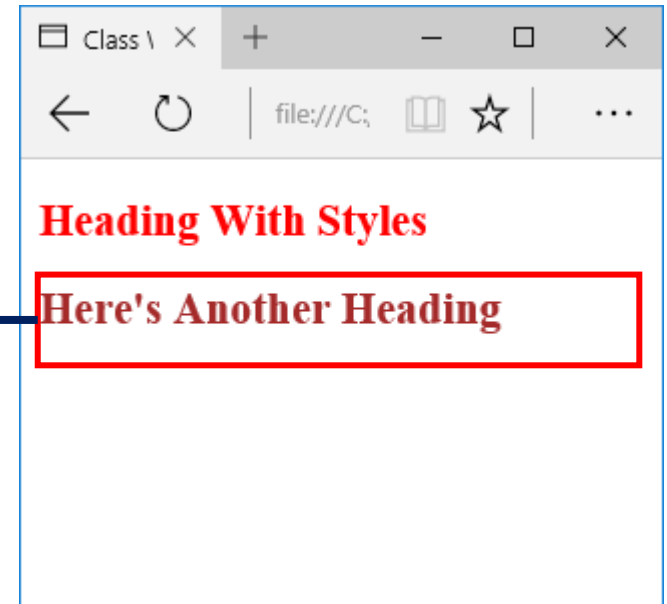
Contents of styles1.css:

```
h2 { color: Red; font-size: 120%; }
```

But...

Why did the colour of the second heading not change?

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      h2 { color: Blue; font-size: 150%; }
    </style>
    <link rel="stylesheet" href="styles1.css" >
  </head>
  <body>
    <h2>Heading with styles</h2>
    <h2 style="color: Brown;">Here's another
heading</h2>
  </body>
</html>
```



Contents of styles1.css:

```
h2 { color: Red; font-size: 120%; }
```

How does CSS work?

CSS defines the styles (i.e. presentation) for the web page, while HTML defines the content and the structure of the content.

After receiving data from the website, the browser performs the following actions:

- It converts the HTML into the ***Document Object Model (DOM)***
- It computes and applies the styles to the document
- It then displays the contents of the DOM (in the browser's viewport)

How does CSS work?

What is the Document Object Model (DOM)?

It is a programming interface for HTML (and XML) documents.

It represents the HTML documents in a structured manner.

It provides mechanisms to (programmatically) access and change the structure of the HTML document, its styles and content.

*The DOM lets you look at **each HTML element as an object**, and allows you to **create, change or remove (hide)** these elements, based on user interaction or some other condition (e.g. time or change in another web resource).*

Therefore, DOM is an **object-oriented representation of a web page**.

How does CSS work?

What is the Document Object Model (DOM)?

Each element in an HTML web page have attributes and these attributes can change based on various conditions, such as user behaviour, change in the data (in the database), etc.

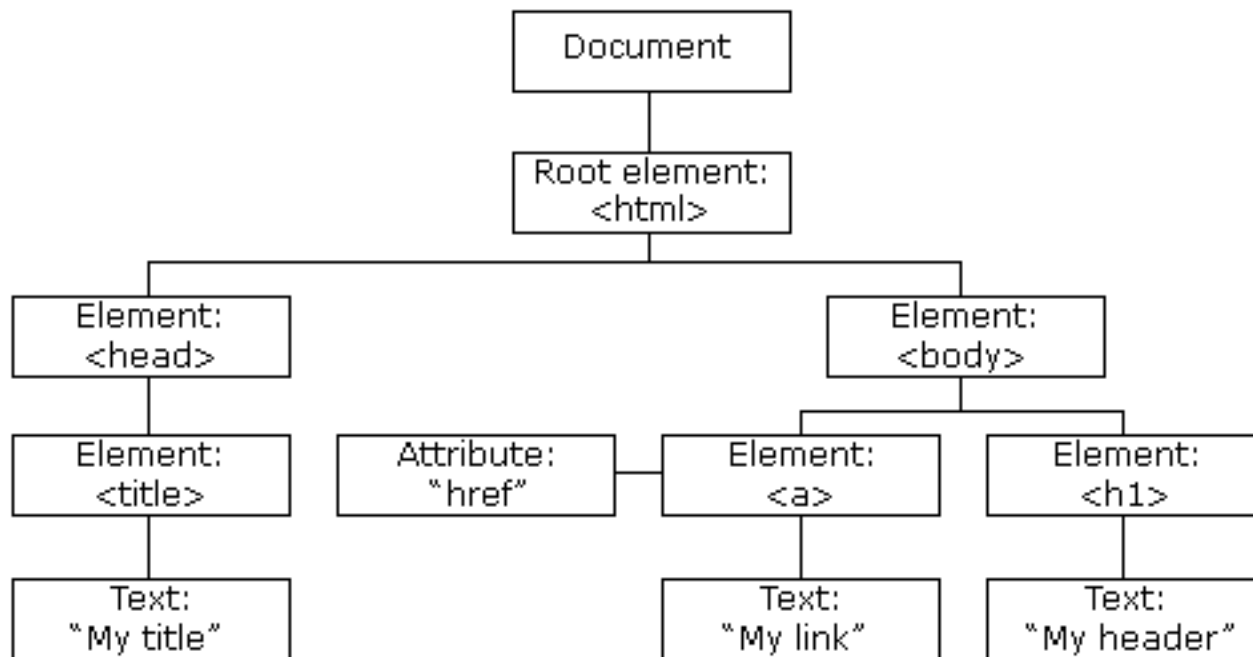
Typically, we use scripting languages (e.g. JavaScript) to access and modify the objects (or elements) in the DOM.

These elements or objects and their attributes + values become “nodes” in the ***tree structure*** of the HTML document.

How does CSS work?

What is the Document Object Model (DOM)?

Example tree structure of the DOM:



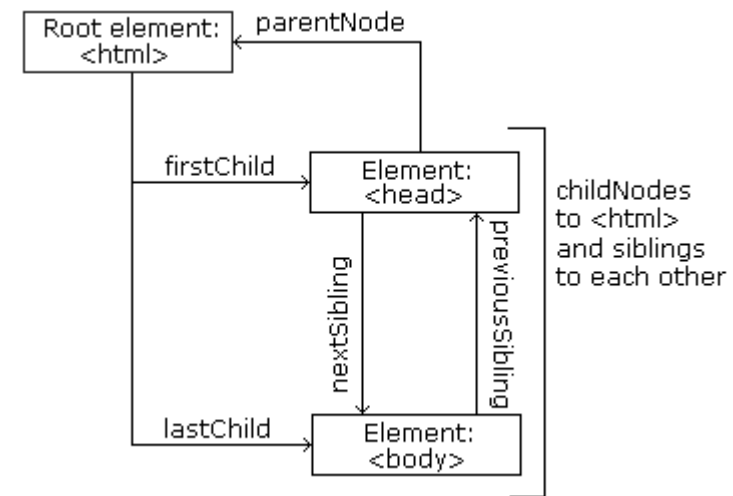
How does CSS work?

What is the Document Object Model (DOM)?

Each HTML node has a relationship with other nodes in the document.

```
<html> <!-- Root element -->
<head> <!-- First child element -->
  <!-- Child of first child element -->
  <title>HTML DOM</title>
</head>

<body> <!-- Second child element -->
  <!-- Children of second child element -->
  <h1>HTML DOM</h1>
  <p>Paragraph text.</p> <!-- Siblings -->
</body>
</html>
```



All elements are “child” nodes of the “root node” or the top node in the DOM tree. Each node has a parent-child node relationship with any other node that it might “contain”. Nodes with the same “parent node” are called “siblings”.

How does CSS work?

What is the Document Object Model (DOM)?

We need to understand the DOM to be able to apply changes to the document based on user interaction.

What is “cascading” about CSS?

The style information about a web page forms a “cascade”, of sorts...

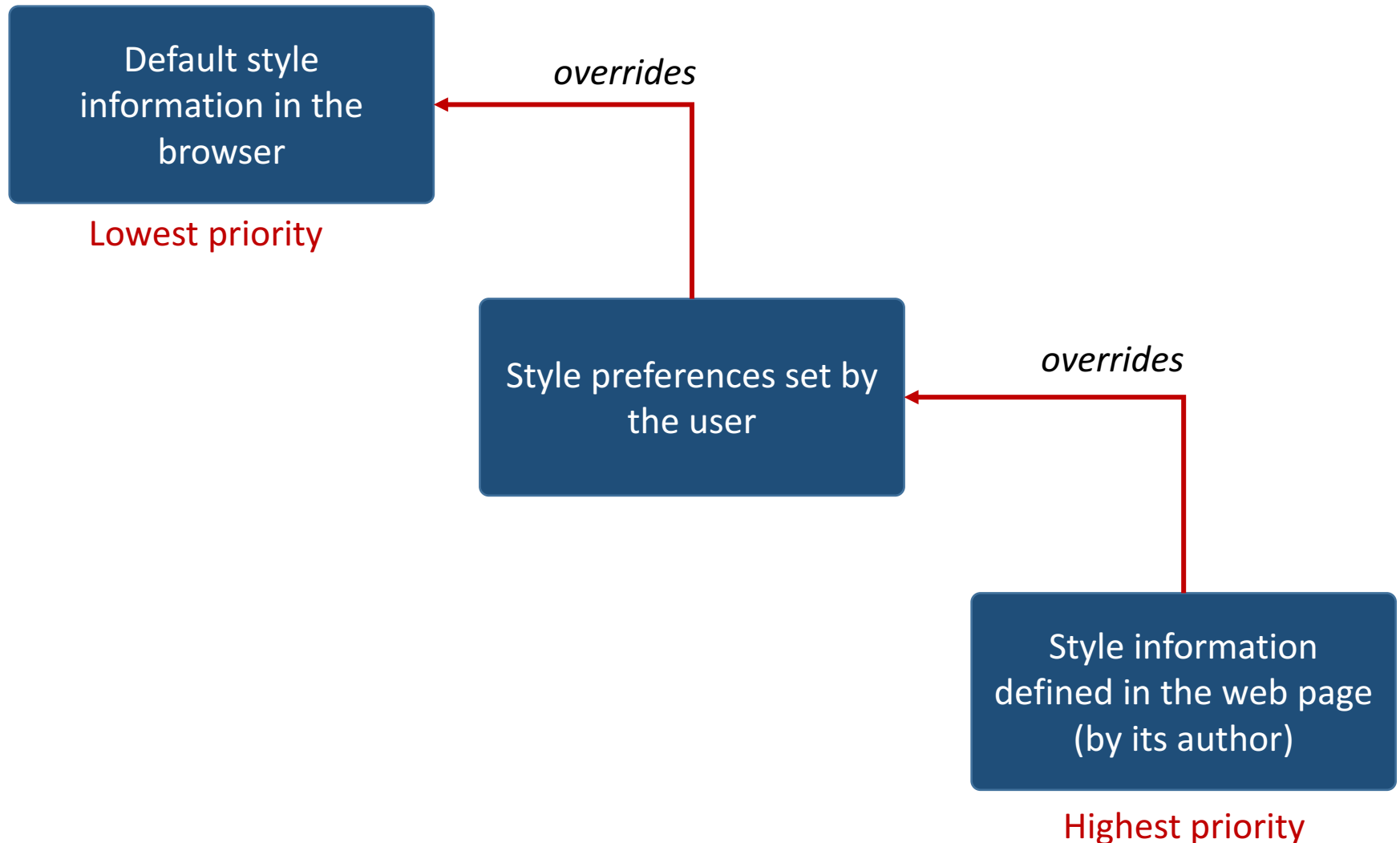


cbmw.org

- The browser has default styles for the markup
 - e.g. default ways to render headings, etc.
 - Have lowest priority
- The user (using the browser) may specify styles by updating style preferences in the browser
 - *this modifies the browser's default styles*
- The styles applied on the web page by the author (i.e. using external style sheets, style element or using style attribute).
 - *This further modifies the styles.*
 - *Has highest priority.*

What is “cascading” about CSS?

The style information about a web page forms a “cascade”, of sorts...

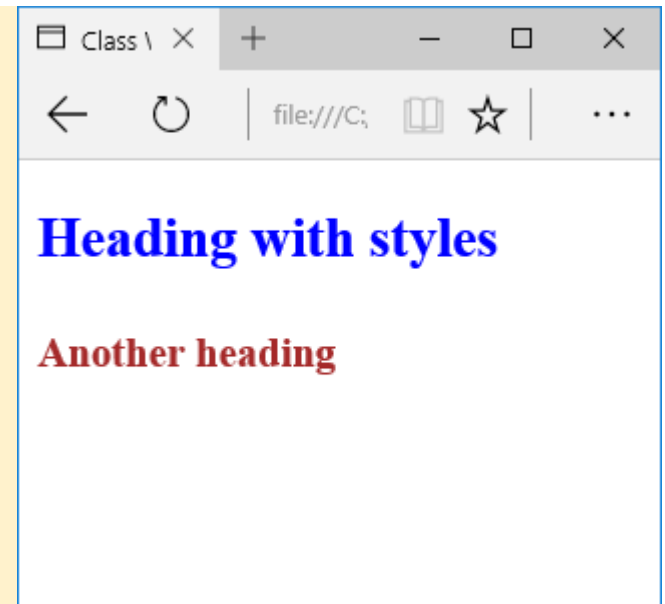


What is “cascading” about CSS?

This cascading behaviour is also observed in nested elements.

- In the example below, notice that the font type is Times New Roman, which is the default browser font. The font size is set to 150% of the default font size for h2 set by the browser.
- Also notice how the style for the second heading (set by the element) overrides the global values set by the <style> element.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <link rel="stylesheet" href="styles1.css" >
    <style>
      h2 { color: Blue; font-size: 150%; }
    </style>
  </head>
  <body>
    <h2>Heading with styles</h2>
    <h2><span style="color: Brown; font-size: 75%">
      Another heading</span></h2>
  </body>
</html>
```



What is “cascading” about CSS?

Let’s revisit our “how are page styles affected when...” example.

- We considered two cases:
 - If `<link>` is placed **before** `<style>` in the markup, then the style definitions in the `<style>` element override the definitions in the external style sheet.
 - If `<link>` is placed **after** `<style>` in the markup, then the style definitions in the external style sheet override the definitions in the `<style>` element.
- This behaviour is observed because:
 - HTML pages are read/rendered sequentially
 - Also, because the applied styles have a cascading effect
- Normally, we would place the `<style>` after `<link>` (if at all) – mainly because we are trying to create specific styles for that page that override global styles set by the external CSS.

What is “cascading” about CSS?

Nested elements **acquire** certain properties from the parent element.

This is why we see common characteristics between the parent and child elements, even if *some* attributes may have changed.

Inheritance in CSS

What is inheritance?

“An attribute acquired by a successor from an ancestor”

In CSS, inheritance allows child elements to ***inherit*** certain characteristics (attributes + their values) from parent elements.

E.g. If a portion of paragraph text is emphasized, only font-style of the emphasis (child) element is set to “italic”, while all other attributes remain the same as the paragraph (parent) element.

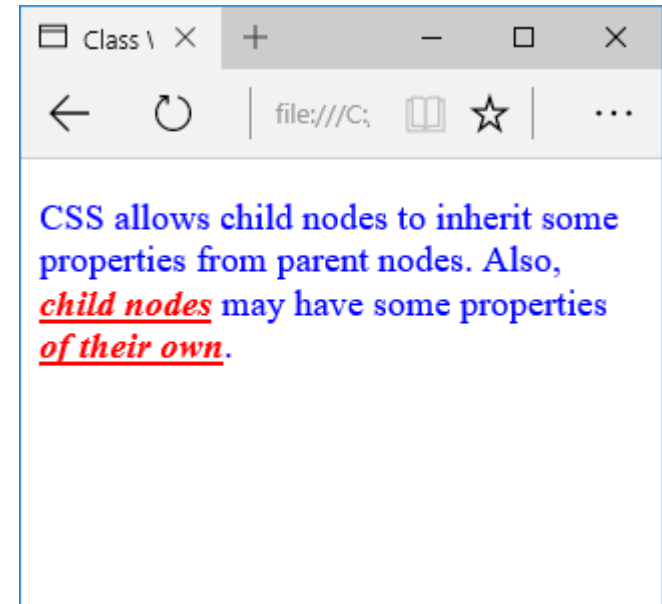
If no style rules are assigned for a child node, rules of the parent node are applied.

The child node's own style will have priority over the parent node's style.

Inheritance in CSS

Example.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Class Work</title>
    <style>
      p { color: Blue; }
      strong { color: Red; font-style: italic;
text-decoration: underline; }
    </style>
  </head>
  <body>
    <p>CSS allows child nodes to inherit some
      properties from parent nodes.
      Also, <strong>child nodes</strong> may
      have some properties <strong>of their
      own</strong>.
    </p>
  </body>
</html>
```



Inheritance in CSS

Specified, computed and actual values

We have discussed cascading and inheritance. But, how does the browser (or any other user agent) know what to do when no values are defined for several properties?

The final values assigned to properties are applied as follows:

1. First, the value is determined through specification (***specified value***):
 - Use values from the cascade (if available)
 - Otherwise, inherit the property from parent element (if element is not the root element)
 - Otherwise, use the property's default value in its definition
2. Second, values are ***resolved*** from specified values (***computed value***):
 - Convert all URIs to absolute paths
 - Convert all dimensions into pixels
 - This resolved value is used for inheritance

Inheritance in CSS

Specified, computed and actual values

3. Third, values are converted into absolute values, if necessary (*used value*):
 - Useful when some elements are “dependent” on values of other elements
 - E.g. if width of a **div** is set to 75% of its parent element, the user agent will not know the width of its parent element until it is rendered
 - In such cases, the value **used** will be determined as the page is being rendered
4. Finally, values are **transformed** as per rules of the local environment (*actual value*):
 - The final value that is applied to a property, taking into account any approximations or limitations of the user agent environment
 - E.g. if width of **div** is set to 75% and that 75% yields a floating point value (with decimal points), but the user agent can use only integer values, then, it will approximate the value to the nearest integer
 - E.g. if the user agent can only render black/white graphics, it will apply approximations on the colour

Inheritance in CSS

Specified, computed and actual values: summary

1. The value is determined through specification (*specified value*)
2. Values are **resolved** from specified values (*computed value*)
3. Values are converted into absolute values, if necessary (*used value*)
4. Values are **transformed** as per rules of the local environment (*actual value*)

Inheritance in CSS

The *inherit* value

Recall the syntax of a style definition from our discussion on the style rule syntax:

style-property: *value*;

E.g. **color: Red;**

When assigning values to attributes, you may set a value of *inherit*, i.e. you may **enforce** inheritance.

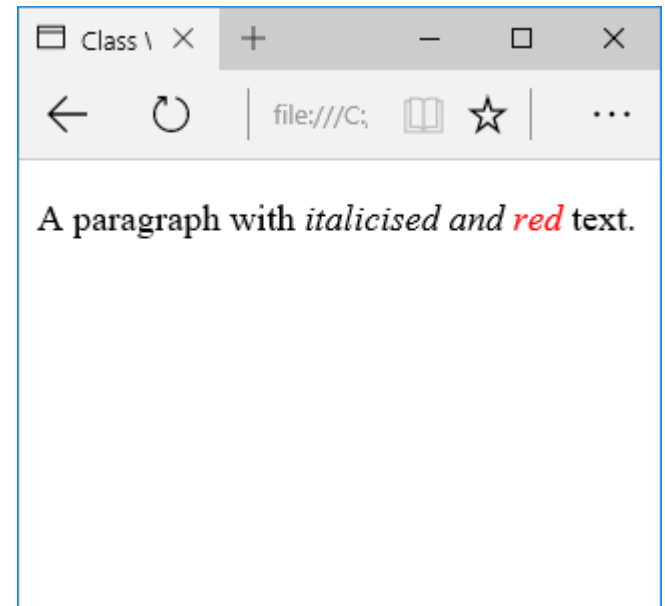
This means that for that style property, *the value will be inherited from its parent element.*

Inheritance in CSS

The *inherit* value

E.g.

```
<p>  
  A paragraph with  
  <span style="font-style:italic;">  
    italicised and  
    <span style="font-style:inherit; color:Red;">  
      red  
    </span>  
  </span>  
  text.  
</p>
```



Inheritance in CSS

!important rules

By default, rules in an author's style sheet override the style rules in the user's style sheet.

However, if you (as the user) wanted to override the styles set by the author (or may be disable an element – like an advertisement box), you may use the ***!important*** declaration.

The ***!*** symbol is called the “delimiter” and is followed by “important”.

This declaration tells the user agent that a particular value assigned to the style property has more importance over other values (which may have been computed in the cascade).

Usage: **font-style: italic !important;**

Inheritance in CSS

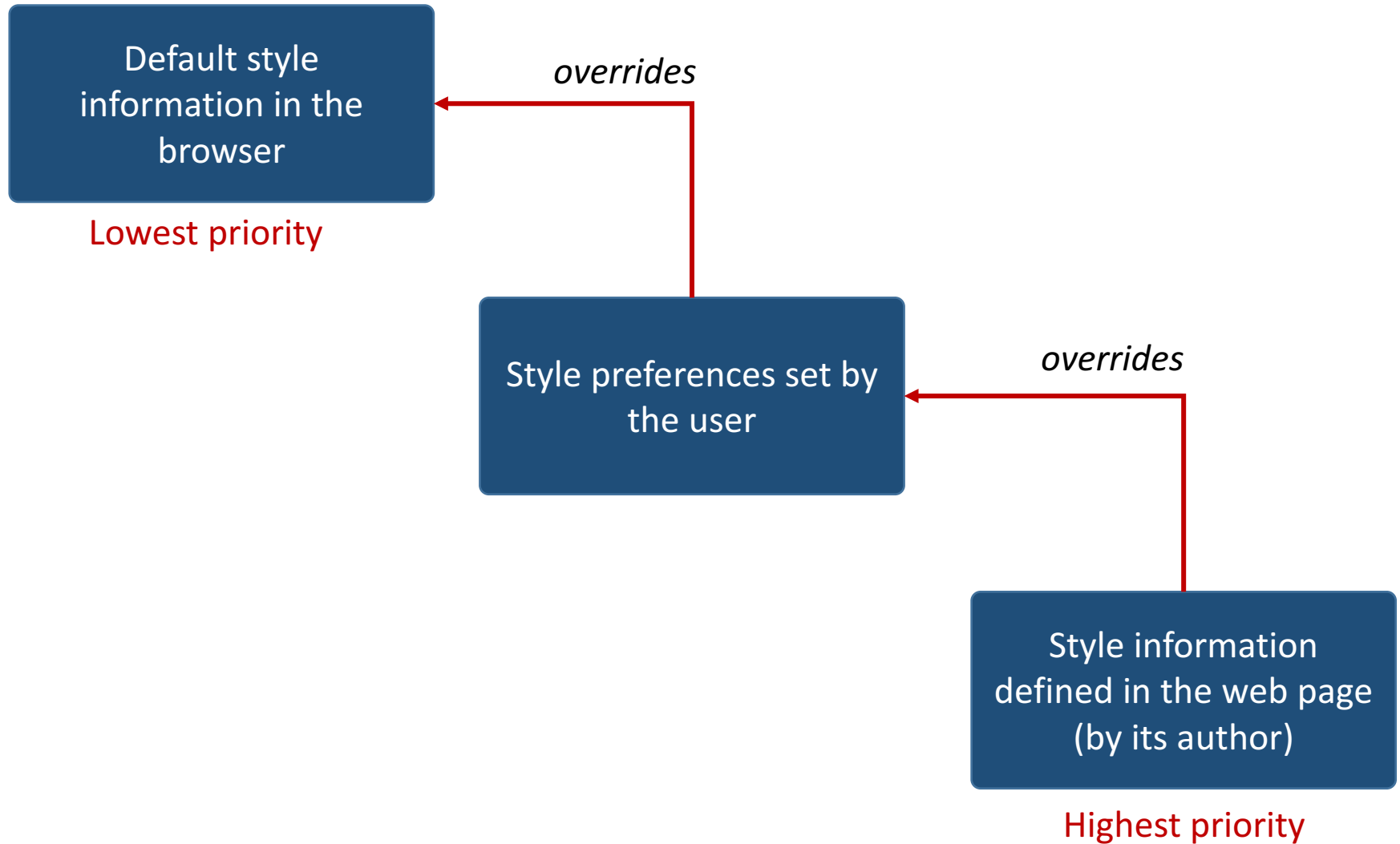
!important rules

In this case, however, the user set values which are marked as ***!important*** have priority over author set values.

This is because some users may prefer text with a larger font, or may have a preference for colour, or something else.

This means that we have to update our cascade...

The “initial” cascade



Updating the “cascade” rules

