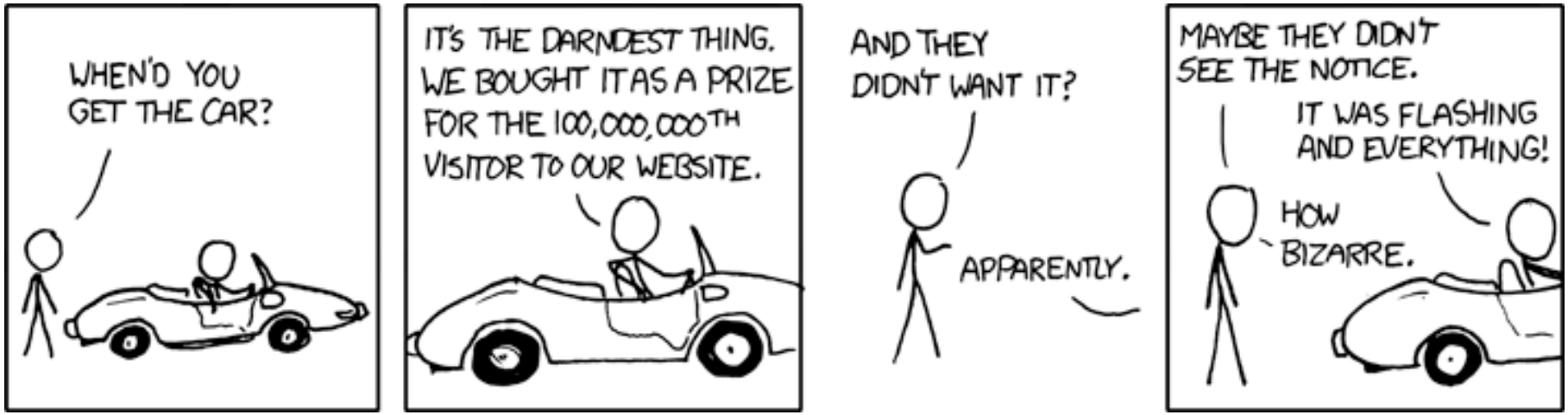

WHEN'D YOU GET THE CAR?
IT'S THE DARNDEST THING. WE BOUGHT IT AS A PRIZE FOR THE 100,000,000TH VISITOR TO OUR WEBSITE.
AND THEY DIDN'T WANT IT?
APPARENTLY.
MAYBE THEY DIDN'T SEE THE NOTICE.
IT WAS FLASHING AND EVERYTHING!
HOW BIZARRE.

INFX 1606
INTRODUCTION TO
WEBSITE CREATION

DALHOUSIE UNIVERSITY
FACULTY OF
COMPUTER SCIENCE

CSS – “Cascading”
and
HTML (Cont’d)

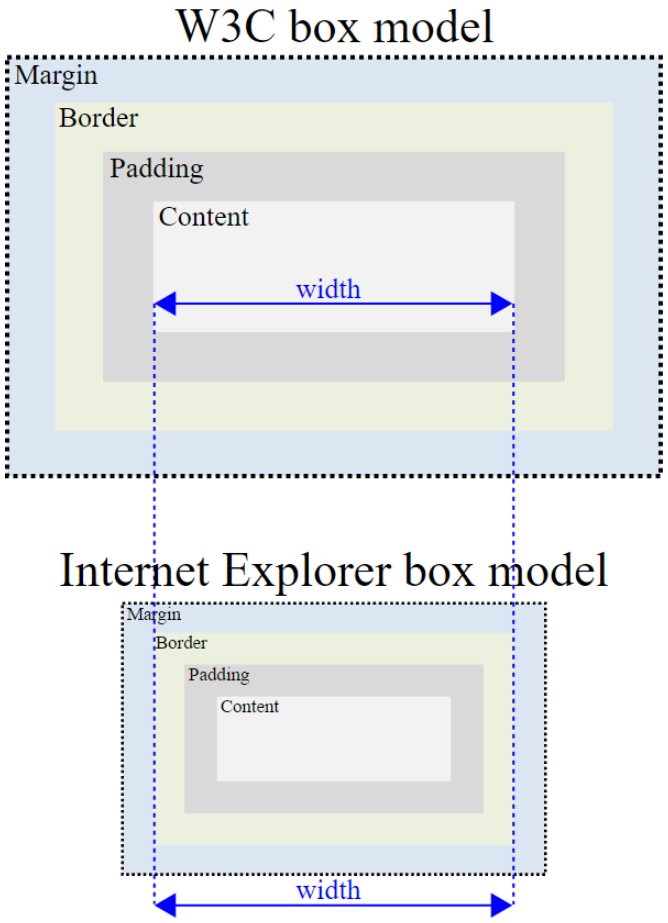
Raghav V. Sampangi

Instructor
Faculty of Computer Science, Dalhousie University

raghav@cs.dal.ca

The Box Model (Recap)

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!

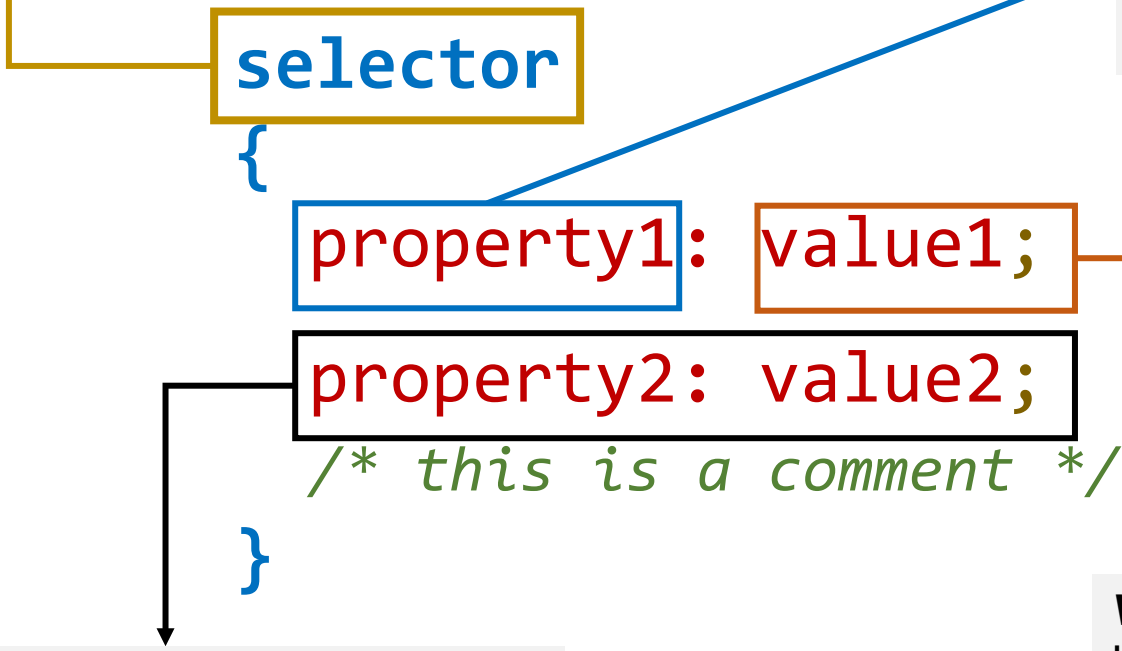


More info: **https://en.wikipedia.org/wiki/Internet_Explorer_box_model_bug**

Syntax of CSS Rules (Recap)

Selector.
Indicates the element to which the style rule applies.

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



Value.
Indicates the settings for the chosen property.

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.

Three ways to include styles on a web page (TOP HAT)

Limitations of using style attribute

Limitation of using style element

Limitation of using external style.css

External and Internal Styling on a Single Web Page

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.

Both styles imported from <link> and applied using <style> will be overridden by the style attribute of an element.

What is “cascading” about CSS?

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

cbmw.org

- When the page loads,
  - Objects in the DOM are first loaded, i.e. the page loads without any styles;
  - Styles are applied in the sequence they appear on the page, i.e.
    - Style definitions in the <link> and <style> are resolved first.
    - Style definitions present in the element’s style attribute are resolved next.
  - This is a priority-based style rule application scheme.

- There is another (more important) scheme of priorities –
  i.e. depending on who defines the style rules.

What is “cascading” about CSS?

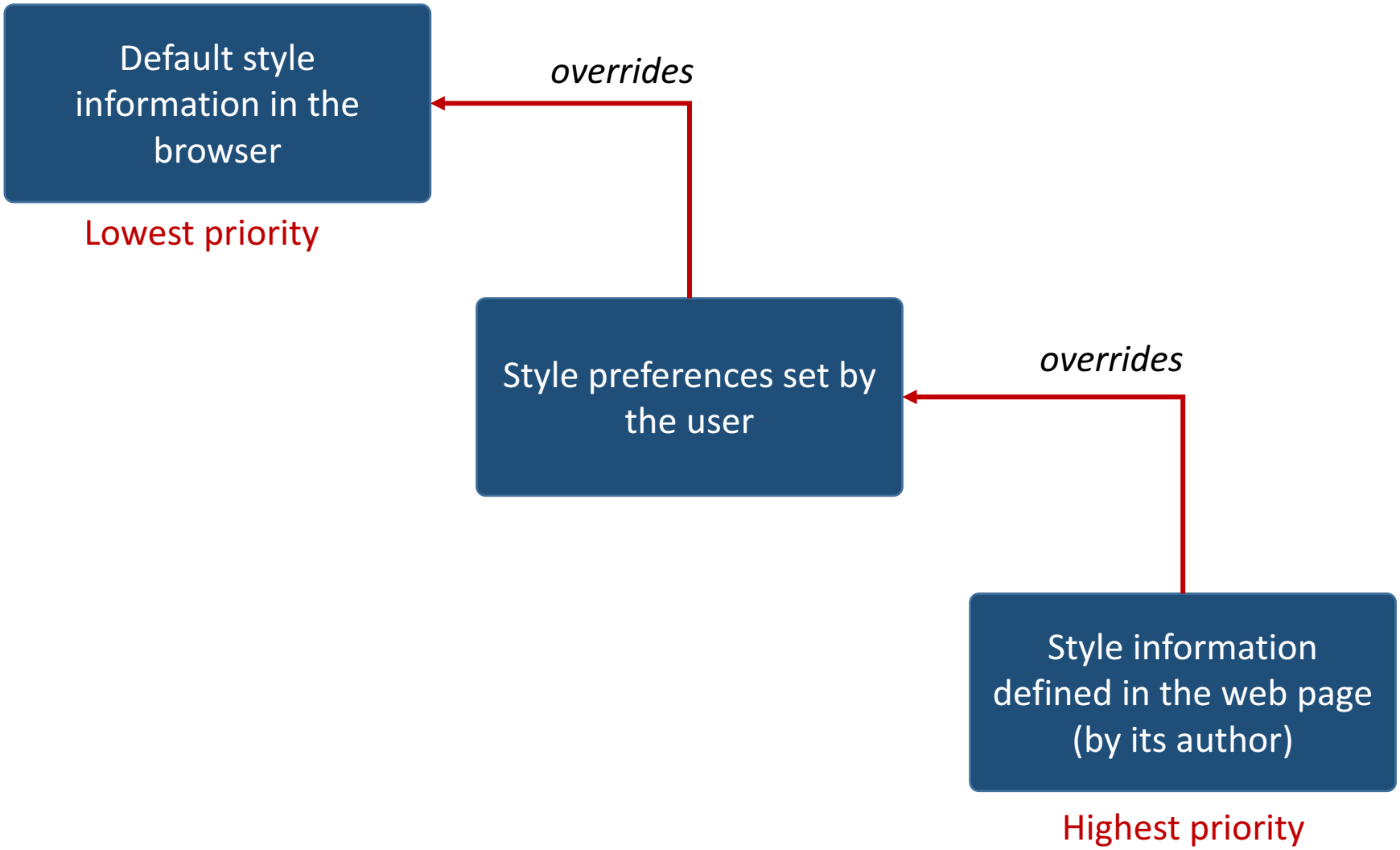
The style information about a web page forms a “cascade”, of sorts…

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…



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

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

The inherit style definition 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.

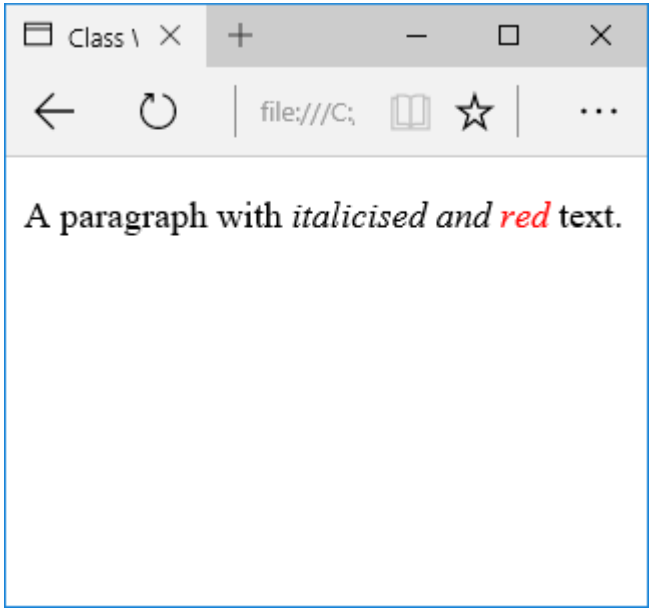
This means that for that style property, *the value will be inherited from its parent element*.

Inheritance in CSS

The inherit style definition 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

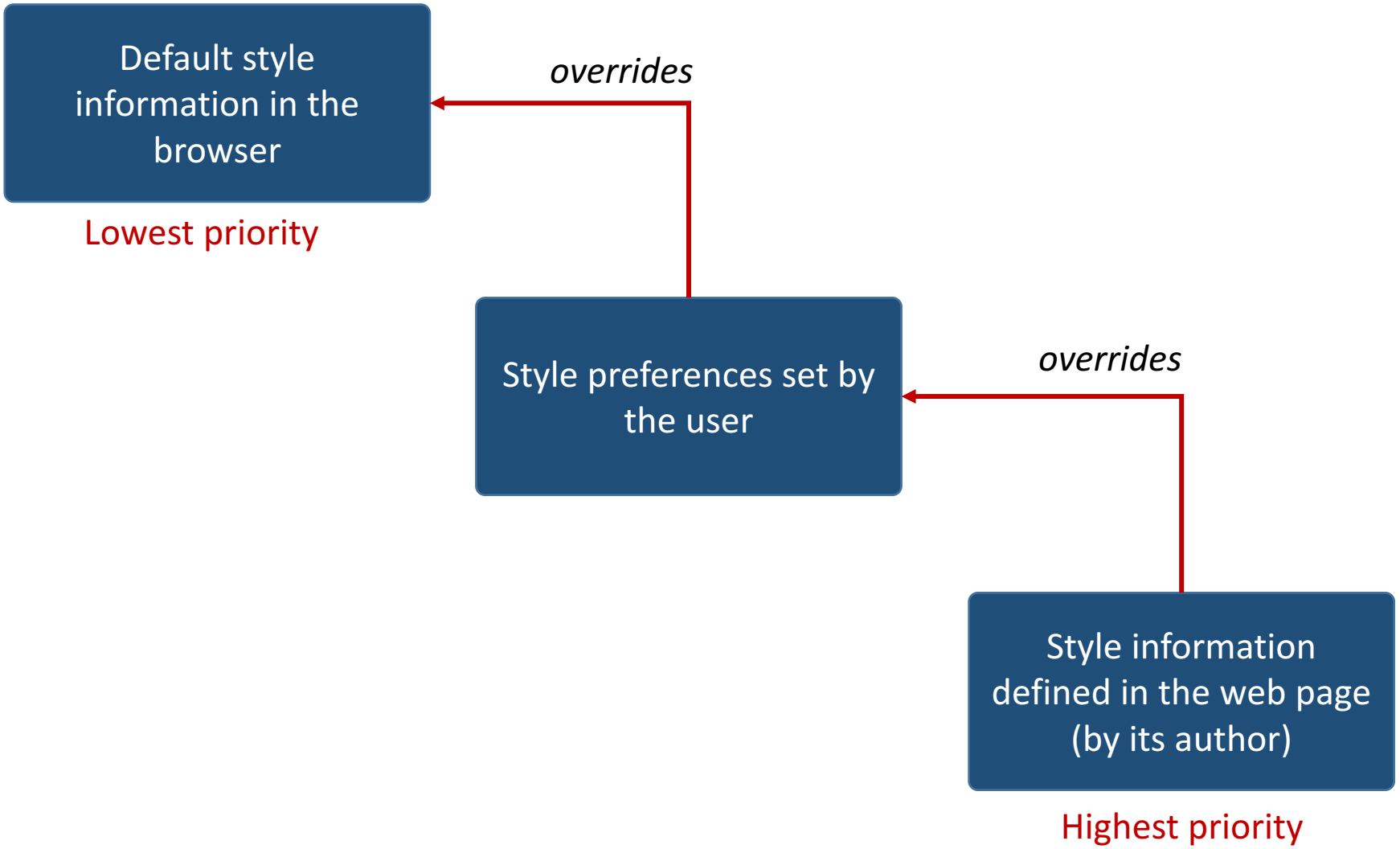
!important rules

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

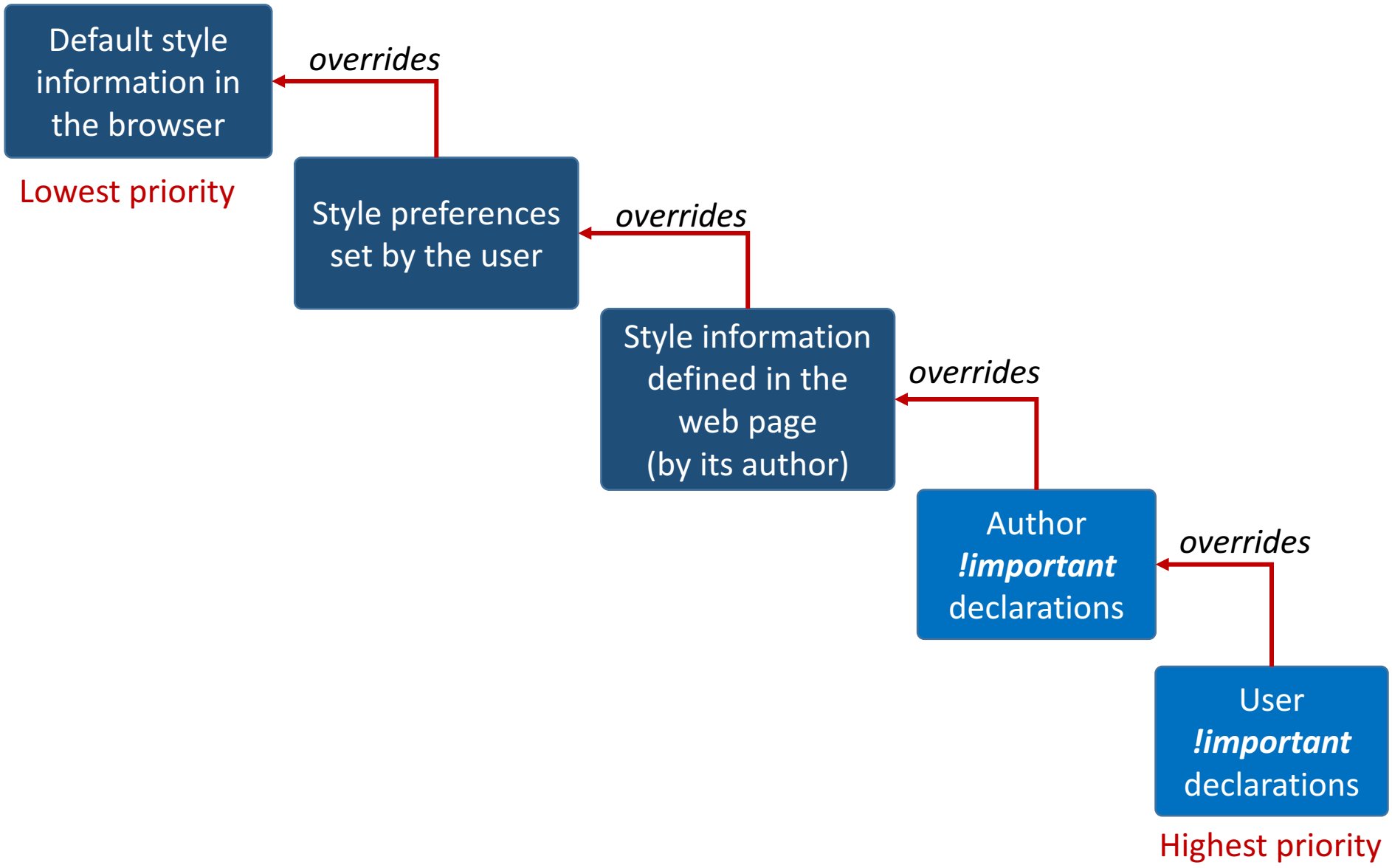
This is because some users may prefer text with a larger font, or may have a preference for colour, or something else.

This means that we have to update our cascade…

The “initial” cascade



Updating the “cascade” rules



Outline

HTML Elements

- Text
- Lists
- Links

Semantic Markup: Recap

Semantic markup is mainly used to tell browsers and web agents that the content has special meaning

By default, some of these markup may have specific styling

However, we have to use CSS to apply styles on the content to make it appear the way we want, i.e. to customize its presentation

So, structural markup only indicates content, semantic markup identifies the meaning that is depicted by the content, and styles help make the content "presentable"

Discussion & demo:
Styling semantic markup

Note

```
<!DOCTYPE html>
<html lang="en">
<head><title>Title goes here</title></head>
<body>
<section><h1>Main heading</h1>
<p>A paragraph.</p></section>
</body>
</html>
```

versus

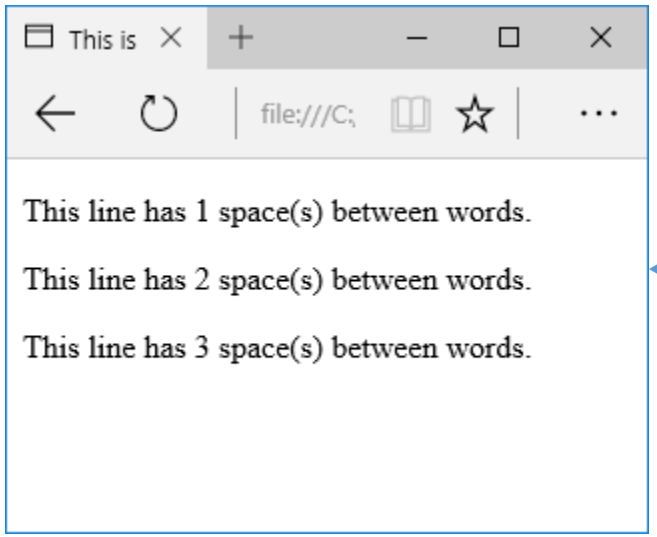
```
<!DOCTYPE html>
<html lang="en">
   <head>
      <title>Title goes here</title>
   </head>
   <body>
      <section>
         <h1>Main heading</h1>
         <p>A paragraph.</p>
      </section>
   </body>
</html>
```

Remember to indent your code.
Just that it will become so much easier to read, understand, and identify errors earlier in website development!

White “space” in markup

White space helps differentiate between various elements on a page

```
<p>This line has 1 space(s) between words.</p>
<p>This  line  has  2  space(s)  between  words.</p>
<p>This   line   has   3   space(s)   between   words.</p>
```



A browser “collapses” multiple white spaces when it displays the web page content → ***white space collapsing***

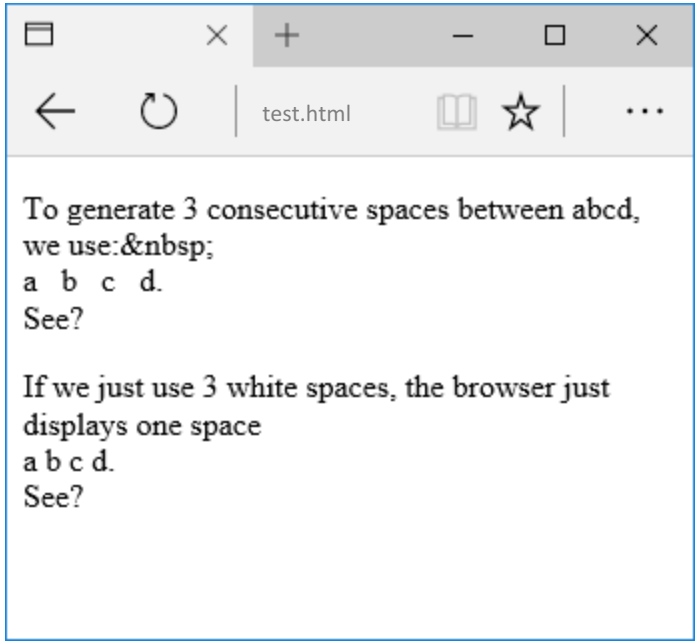
White “space” in markup

White space helps differentiate between various elements on a page

A browser “collapses” multiple white spaces when it displays the web page content → ***white space collapsing***

But, what if I want to use more than one space?

You can generate multiple spaces if you really want to.
You will have to use an ***HTML entity*** → ** **



HTML Entities

Tags include < and > symbols

What if you wanted to use less-than or greater-than symbols in your content?

You have to explicitly tell the browser that the symbols are not tags, but a part of the content

For this purpose, we use ***HTML entities***

Syntax:

&entity_name; OR **&#entity_number;**

HTML Entities

Some examples of HTML entities:

Entity name	Entity number	Description
		Non-breaking space ( )
&lt;	&#60;	Less-than (<)
&gt;	&#62;	Greater-than (>)
&amp;	&#38;	Ampersand (&)
&copy;	&#169;	Copyright (©)
&reg;	&#174;	Registered trademark (®)

HTML Lists

HTML Elements: Lists

List. An ordered way to display content or a set of items in content

Three main types of lists in HTML:

Unordered lists. Use symbols (e.g. bullet points) to indicate order.

Ordered lists. Use characters or numbers to indicate order.

Description or definition lists. Used to indicate descriptions or definitions of terms.

You must not put these lists inside paragraphs, i.e.
They exist by themselves, as independent content elements.

HTML Elements: Lists

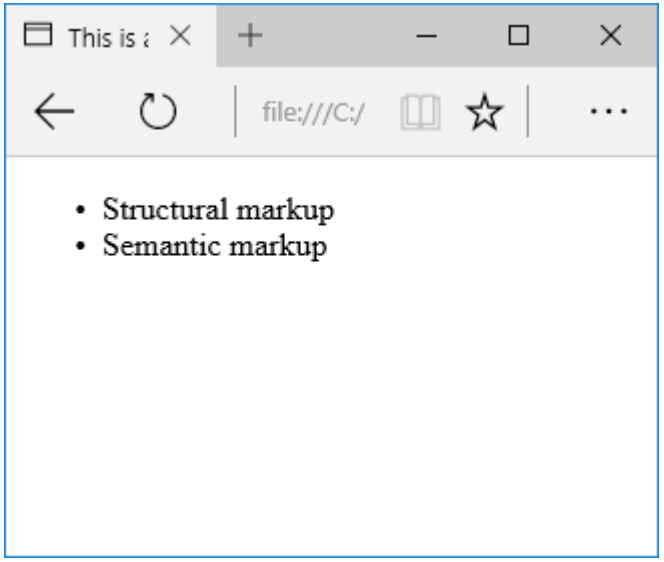
Unordered lists. Use symbols (e.g. bullet points) to indicate order.

Unordered lists are identified by **
**

List items are identified by **
**

Example:

```
<ul>
    <li>Structural markup</li>
    <li>Semantic markup</li>
</ul>
```



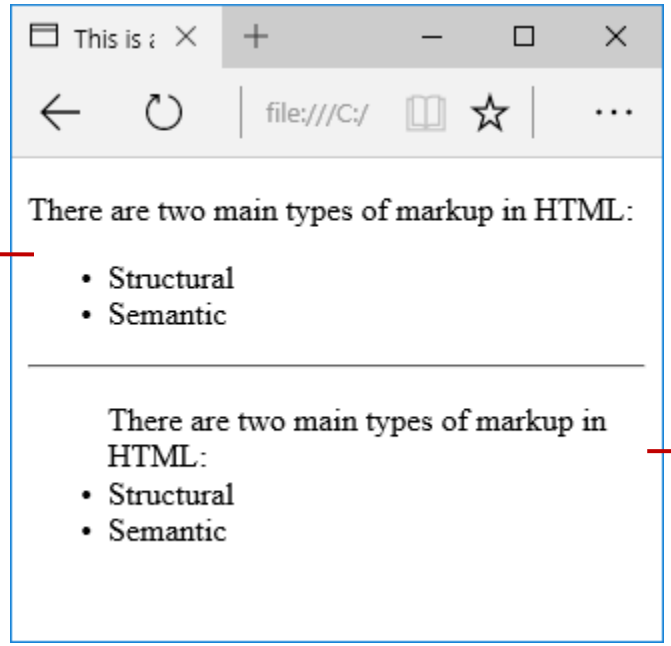
HTML Elements: Lists

Unordered lists. Use symbols (e.g. bullet points) to indicate order.

If you wish to have an association between some text and the items in the list, you must include a ***paragraph element before the list***, rather than including the text when the list begins.

The browser will render the markup in the second case, but it is wrong!!

```
<p>There are two main
types of markup in
HTML:</p>
<ul>
<li>Structural</li>
<li>Semantic</li>
</ul>
```



**Text not allowed between <ul> and **

```
<ul>There are two
main types of markup
in HTML:
<li>Structural</li>
<li>Semantic</li>
</ul>
```

HTML Elements: Lists

Unordered lists. Use symbols (e.g. bullet points) to indicate order.

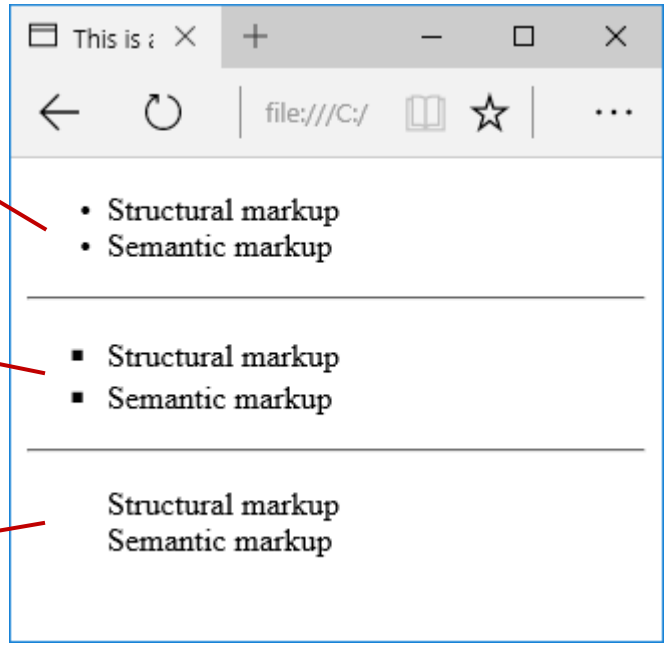
There is a default set of symbols used to render unordered lists. However, if you wish to change the symbol, you have to use the "style" attribute of the lists (*we will learn more about styles later in the course*)

Example:

```
<ul>
    <li>Structural markup</li>
    <li>Semantic markup</li>
</ul>
```

```
<ul style="list-style-type:square">
    <li>Structural markup</li>
    <li>Semantic markup</li>
</ul>
```

```
<ul style="list-style-type:none">
    <li>Structural markup</li>
    <li>Semantic markup</li>
</ul>
```



HTML Elements: Lists

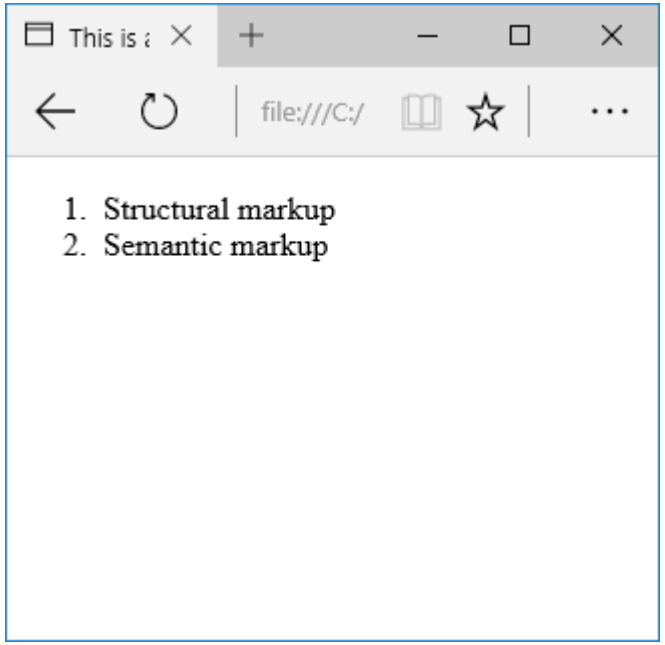
Ordered lists. Use characters or numbers to indicate order.

Ordered lists are identified by **
**

List items are identified by **
**

Example:

```
<ol>
    <li>Structural markup</li>
    <li>Semantic markup</li>
</ol>
```



HTML Elements: Lists

Ordered lists. Use characters or numbers to indicate order.

There is a default set of characters used to render ordered lists. However, if you wish to change the characters, you have to use the ***type*** attribute of ordered lists

Type	Description
“1”	Numbers (default)
“A”	Uppercase letters
“a”	Lowercase letters
“I”	Uppercase Roman numbers
“i”	Lowercase Roman numbers

HTML Elements: Lists

Ordered lists: Examples of using the “type” attribute:

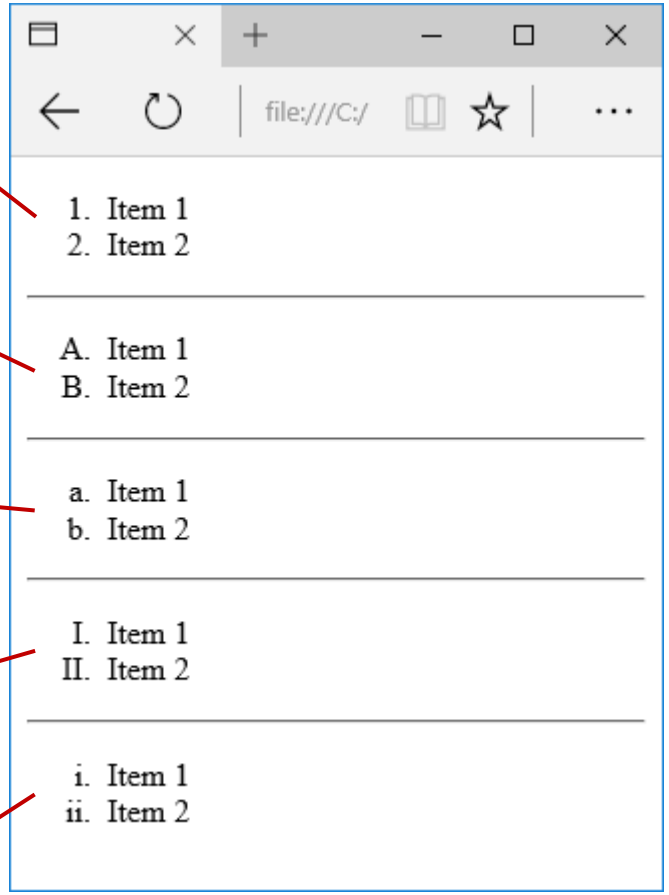
```
<ol type="1">
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```

```
<ol type="A">
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```

```
<ol type="a">
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```

```
<ol type="I">
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```

```
<ol type="i">
    <li>Item 1</li>
    <li>Item 2</li>
</ol>
```



HTML Elements: Lists

Description or definition lists. Used to indicate descriptions or definitions of terms.

Let's say that you are defining some key words / terms used in an article. Description / definition lists are very useful in such contexts.

Some other elements you can use for this purpose:

<p>
Defines a paragraph

<ol>
Defines an ordered list

<ul>
Defines an unordered list

HTML Elements: Lists

Description or definition lists. Used to indicate descriptions or definitions of terms.

Elements of a description list:

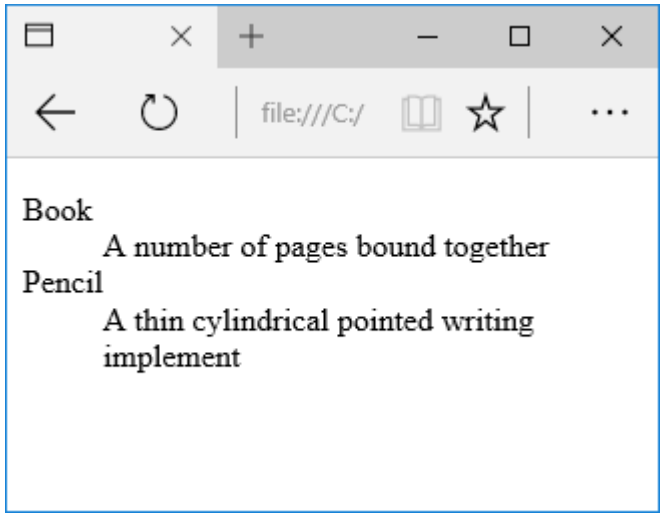
<dl></dl> defines a description list element

<dt></dt> defines a description term (or the name)

<dd></dd> defines the description of the term

Example:

```
<dl>
    <dt>Book</dt>
    <dd>A number of pages bound together</dd>
    <dt>Pencil</dt>
    <dd>A thin cylindrical pointed writing implement</dd>
</dl>
```

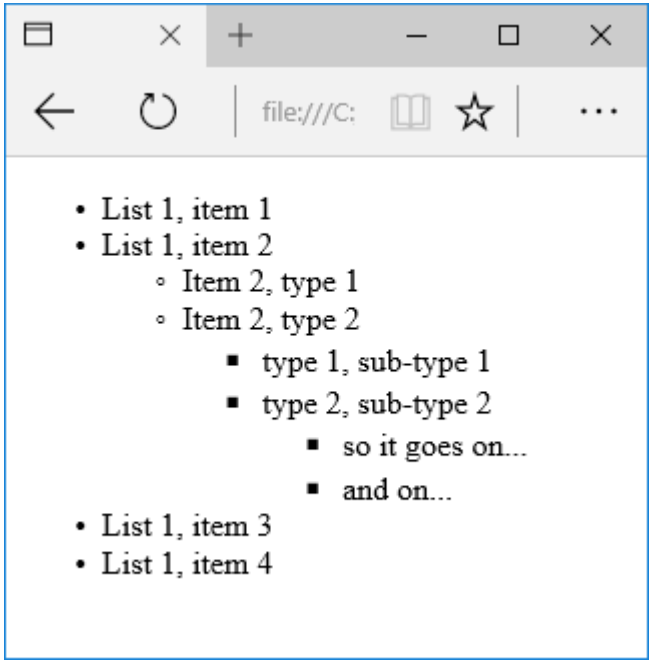


HTML Elements: Lists

Nested lists. You may also include lists within lists.

Example:

```
<ul>
  <li>List 1, item 1</li>
  <li>List 1, item 2
    <ul>
      <li>Item 2, type 1</li>
      <li>Item 2, type 2
        <ul>
          <li>type 1, sub-type 1</li>
          <li>type 2, sub-type 2
            <ul>
              <li>so it goes on...</li>
              <li>and on...</li>
            </ul>
          </li>
        </ul>
      </li>
    </ul>
  </li>
  <li>List 1, item 3</li>
  <li>List 1, item 4 </li>
</ul>
```



HTML Elements: Lists

Nested lists. You may also include lists within lists.

This is wrong, but will be rendered by the browser.

Why is this wrong?

```
<ul>
   <li>List 1, item 1</li>
   <li>List 1, item 2</li>
   <ul>
      <li>Item 2, type 1</li>
      <li>Item 2, type 2</li>
   </ul>
   <li>List 1, item 3</li>
   <li>List 1, item 4 </li>
</ul>
```

← This is wrong!

This is the correct way to create nested lists.

```
<ul>
   <li>List 1, item 1</li>
   <li>List 1, item 2
      <ul>
         <li>Item 2, type 1</li>
         <li>Item 2, type 2</li>
      </ul>
→  </li>
   <li>List 1, item 3</li>
   <li>List 1, item 4 </li>
</ul>
```

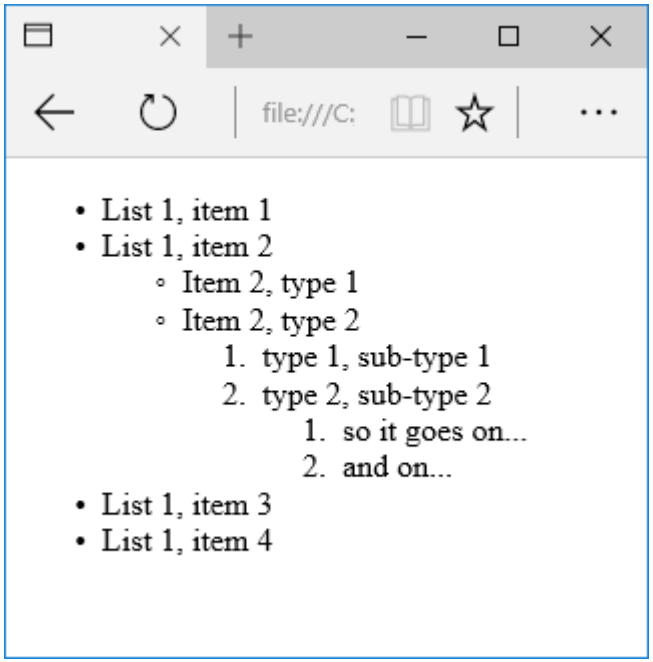
 elements are **not valid child elements of another **;
they are valid child elements of **<li>**

HTML Elements: Lists

Nested lists. You may also include lists within lists.

Example (different types of lists, having default styling):

```
<ul>
  <li>List 1, item 1</li>
  <li>List 1, item 2
    <ul>
      <li>Item 2, type 1</li>
      <li>Item 2, type 2
        <ol>
          <li>type 1, sub-type 1</li>
          <li>type 2, sub-type 2
            <ol>
              <li>so it goes on...</li>
              <li>and on...</li>
            </ol>
          </li>
        </ol>
      </li>
    </ul>
  </li>
  <li>List 1, item 3</li>
  <li>List 1, item 4</li>
</ul>
```

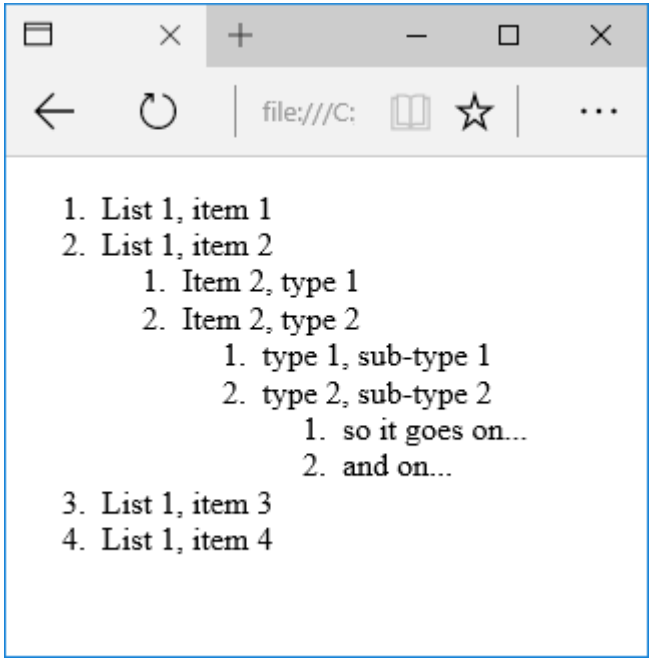


HTML Elements: Lists

Nested lists. You may also include lists within lists.

Example (ordered lists, with default “type” attributes):

```
<ol>
  <li>List 1, item 1</li>
  <li>List 1, item 2
    <ol>
      <li>Item 2, type 1</li>
      <li>Item 2, type 2
        <ol>
          <li>type 1, sub-type 1</li>
          <li>type 2, sub-type 2
            <ol>
              <li>so it goes on...</li>
              <li>and on...</li>
            </ol>
          </li>
        </ol>
      </li>
    </ol>
  </li>
  <li>List 1, item 3</li>
  <li>List 1, item 4</li>
</ol>
```

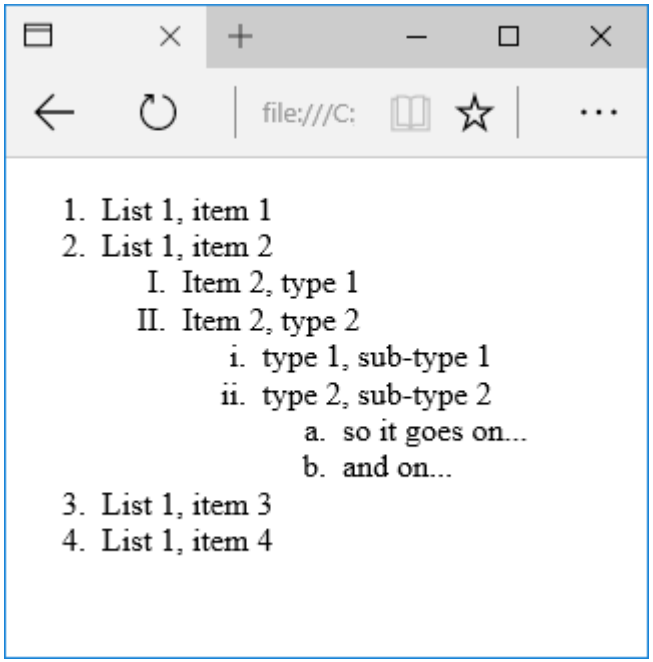


HTML Elements: Lists

Nested lists. You may also include lists within lists.

Example (ordered lists, with different “type” attributes):

```
<ol>
  <li>List 1, item 1</li>
  <li>List 1, item 2
    <ol type="I">
      <li>Item 2, type 1</li>
      <li>Item 2, type 2
        <ol type="i">
          <li>type 1, sub-type 1</li>
          <li>type 2, sub-type 2
            <ol type="a">
              <li>so it goes on...</li>
              <li>and on...</li>
            </ol>
          </li>
        </ol>
      </li>
    </ol>
  </li>
  <li>List 1, item 3</li>
  <li>List 1, item 4</li>
</ol>
```



HTML Elements: Lists

Summary.

Browsers indent list items by default

Three types of lists – unordered, ordered and description/definition

If you wish to change the appearance of [the bullet or type of numbers used in] the lists, you can use CSS to set such properties.

HTML Links

HTML Elements: Links

HyperLinks are one of the essential elements of the web.

Why are they important?

> Because the web was founded based on the idea of managing information by embedding links to other sources of information within a page.

HyperLinks, or simply *links*, represent a connection between two resources on the web.
Note that one of these resources *must* be the current document (or, whatever document you are creating).

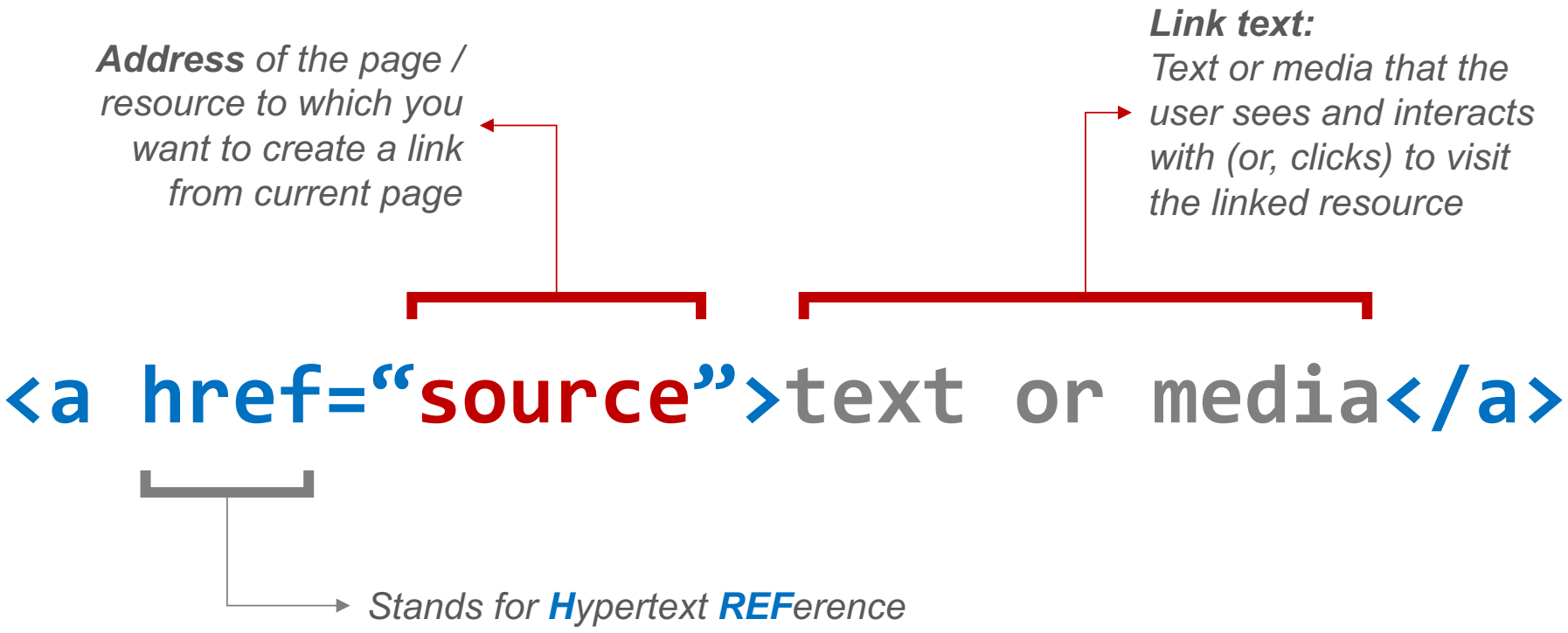
HTML Elements: Links

Links help in connecting:

- One website to another
- One page on a website to another page on the same website
- One part of a page to another part of the same page
- A website to an e-mail address [it starts the e-mail program and addresses the e-mail to the receiver]

HTML Elements: Links

Structure of HTML Links:



Example:

```
<a href="https://www.dal.ca">Dalhousie website</a>
```

HTML Elements: Links

The **href** attribute.

HREF = Hypertext REFerence.

If you want to link to any resource, you *must* use this attribute in the anchor <a> tag.

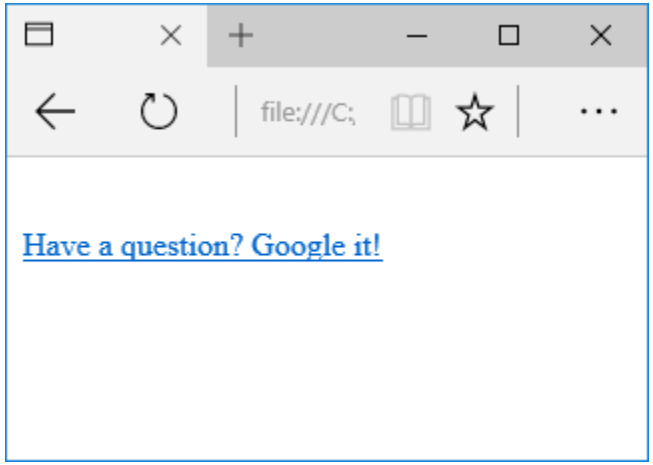
You must include the [complete] URL if you are linking to an external resource (we will discuss these with some examples on the next slide).

HTML Elements: Links

The **href** attribute: Examples.

[1] Linking to another (external) website:

```
<a href="https://www.google.com">Have a question? Google it!</a>
```

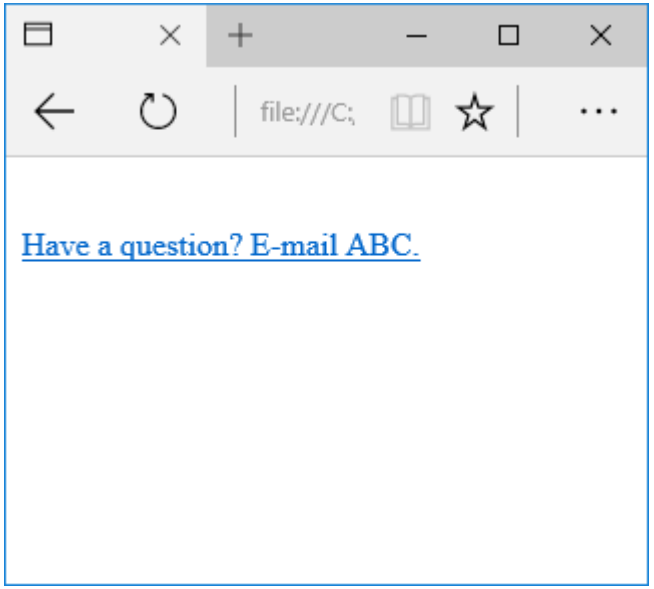


HTML Elements: Links

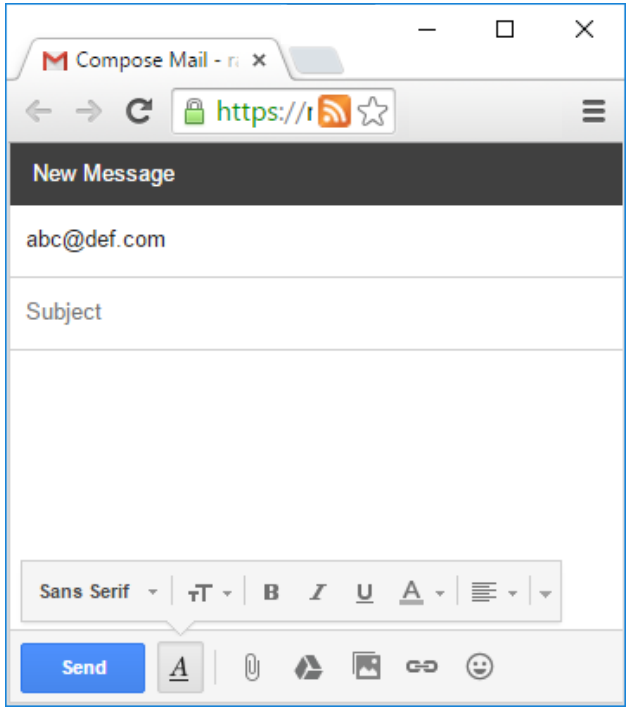
The **href** attribute: Examples.

[2] Linking to an e-mail address:

```
<a href="mailto:abc@def.com">Have a question? E-mail ABC.</a>
```



On clicking the link…



HTML Elements: Links

The [deprecated] **name** attribute: *A digression*.

In earlier versions of HTML, there used to be an attribute called ***name***.

It was mainly used to specify a “name” to an anchor;
Often helped create “bookmarks” within an HTML document.

Example:
Let’s say you had a section called “Recent news”, and it was on the Home (or, front) page of a website. You wanted to create a link from the top of the page to “Recent news”.
Earlier (a long long time ago), you would do the following:

In the paragraph element:

```
<p>Our <a href=“#rnews”>recent
news</a> section presents news.</p>
```

In the Recent news element:

```
<h3>
   <a name=“rnews”>Recent news</a>
</h3>
```

HTML Elements: Links

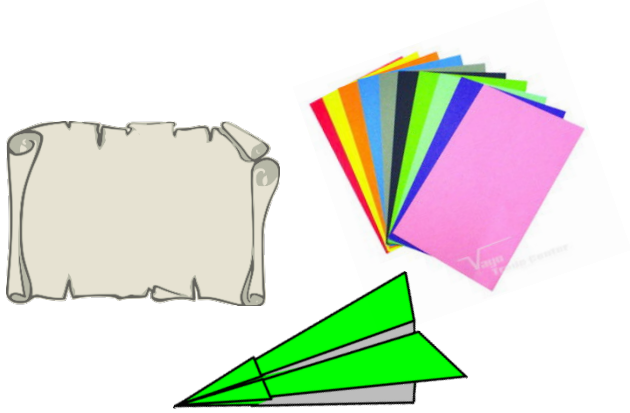
How do you create a bookmark on the same page now?

We use the ***id*** attribute.

The ***id*** attribute helps in identifying an HTML element uniquely, and helps us manipulate it as an “object”.

For now, don’t worry about what an object means. Just know that in this context, each element can be thought of as being similar to a *piece of paper*.

Why paper analogy?
Because paper can be folded (origami), painted on, cut into shapes, and so many other things!



Similarly, you can change the appearance of an object and its functionality, in many different ways.

HTML Elements: Links

How do you create a bookmark on the same page now?

We use the ***id*** attribute.

The ***id*** attribute helps in identifying an HTML element uniquely, and helps us manipulate it as an “object”.

We change the markup from our earlier example as follows:

In the paragraph element:

```
<p>Our <a href=“#rnews”>recent
news</a> section presents news.</p>
```

No changes here.

In the Recent news element:

```
<h3>
   <a id=“rnews”>Recent news</a>
</h3>
```

*We change **name** to **id**.*

We will discuss more about **id**’s in future classes.

HTML Elements: Links

The **href** attribute: Examples.

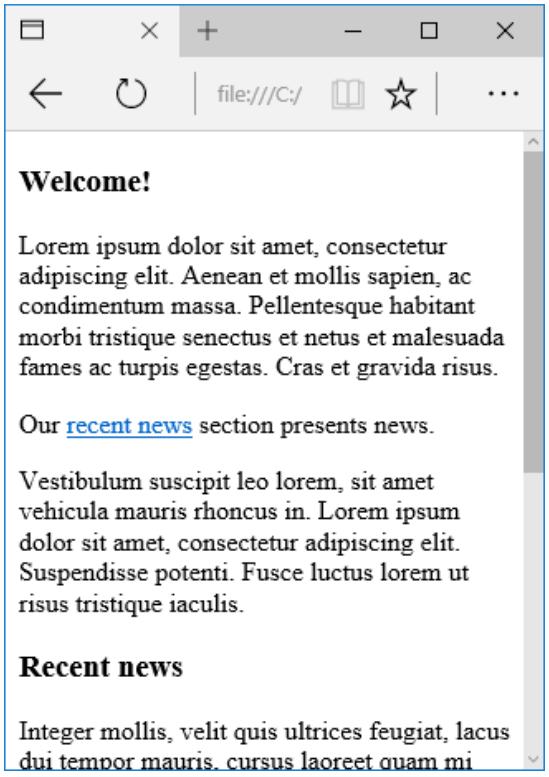
[3] Using the **id** attribute to create links within the same document / page:

In the paragraph element:

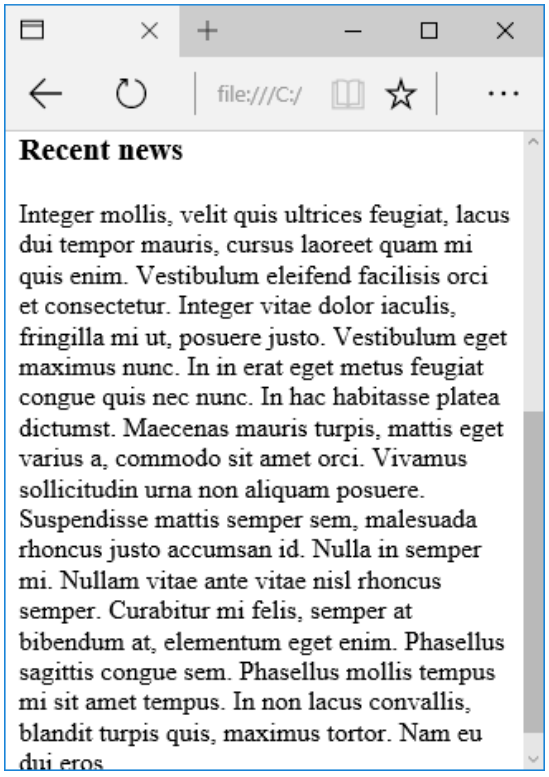
```
<p>Our <a href="#rnews">recent
news</a> section presents news.</p>
```

In the Recent news element:

```
<h3>
   <a id="rnews">Recent news</a>
</h3>
```



On clicking the link…



HTML Elements: Links

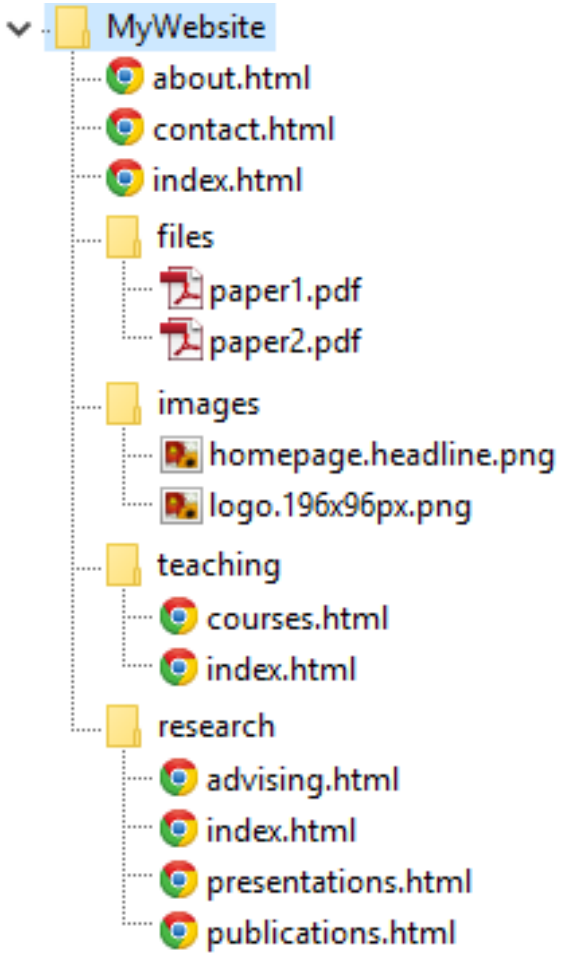
Directory Structure.

Directories or folders help in organizing files better.

You can organize your files (e.g. PDF), images, and include categories for other web pages.

In this example,

- "MyWebsite" is the ***parent*** directory; also called ***root folder***
- files, images, teaching, research are the ***child*** directories of MyWebsite
- If any of these child directories contained another directory, it would be the ***grandchild*** directory of MyWebsite



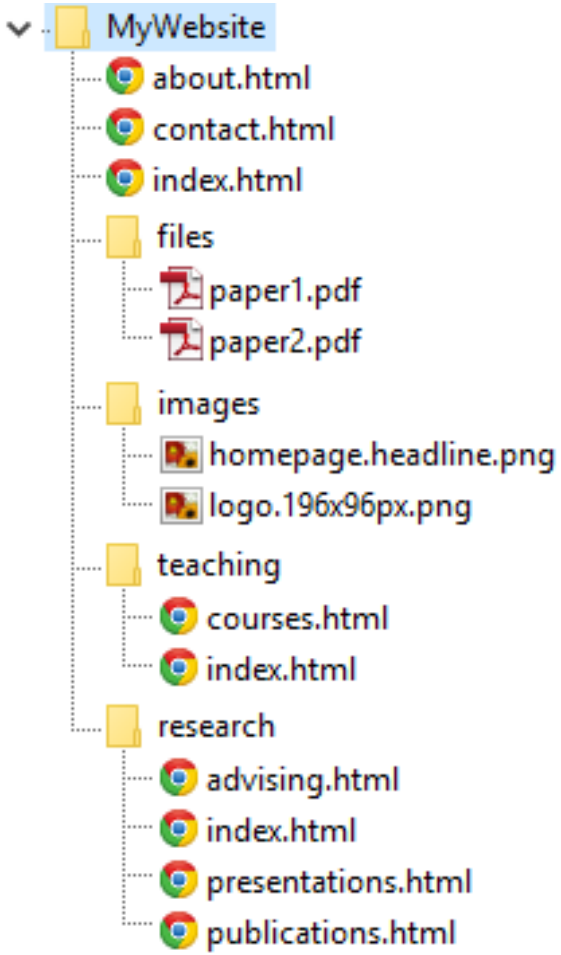
HTML Elements: Links

Directory Structure.

Why is this important?

You will need to know these to link between various pages, and to use resources (e.g. images, files) in your websites.

Also helps you understand how web page / website access actually works.



HTML Elements: Links

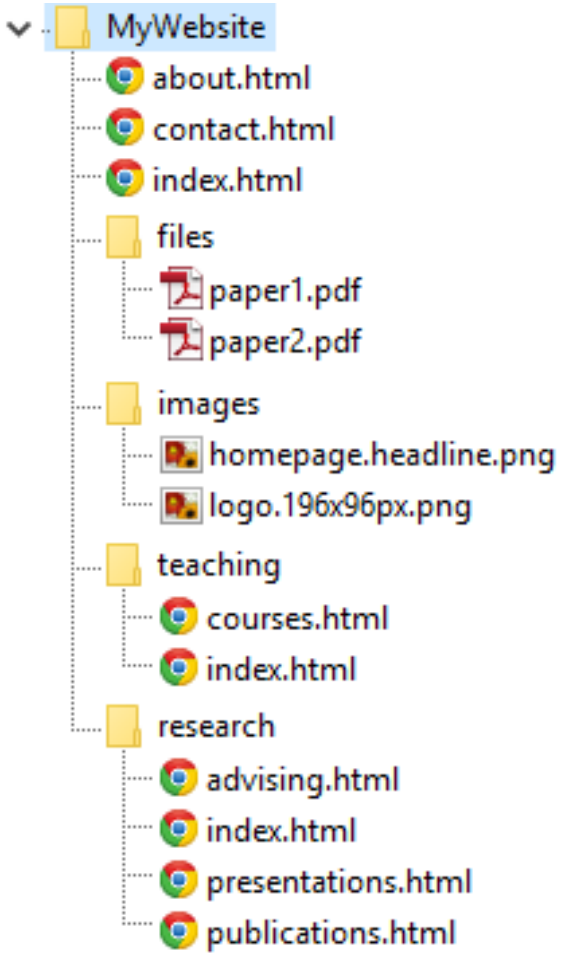
Directory Structure: the homepage (or, front page) of a website

When you visit any website, you don't usually type in the name of the page you want to visit.

In this example, the website name is "mywebsite.com", and you would just enter *www.mywebsite.com* in the browser address bar.

By default, websites are configured to direct you to the ***index.html*** page of any folder.

When you visit a website, you are actually visiting the **root** folder, and therefore, you are directed to the index page in the root folder.



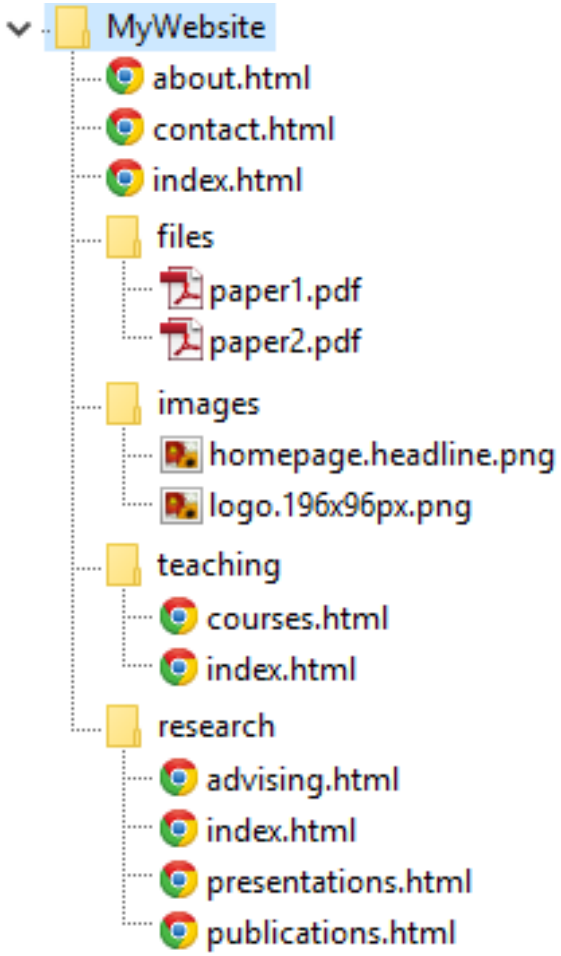
HTML Elements: Links

Directory Structure: the homepage (or, front page) of a website

The page, index.html in the root folder, is called the website's *homepage* or *front page*. (because that's the first page you see when you visit the website).

Similarly, if you visit:
www.mywebsite.com/research

you will be directed to
www.mywebsite.com/research/index.html



HTML Elements: Links

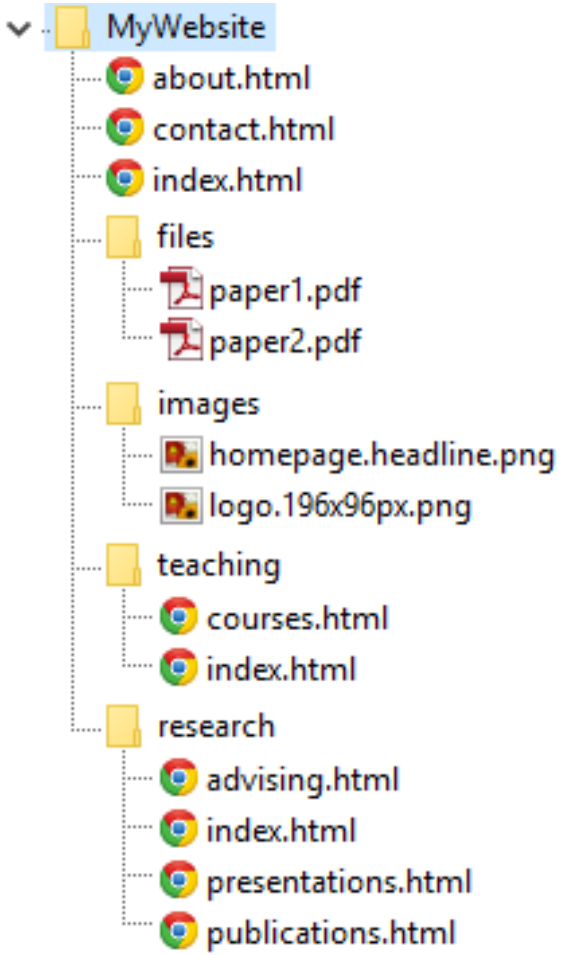
Directory Structure: URLs and relative paths

The full URL of the index.html file in the root folder of this website will be:
http://www.mywebsite.com/index.html

If we wanted to visit courses.html under the “teaching” folder, then, the URL will be:
http://www.mywebsite.com/teaching/courses.html

But, do we need to use the full path from anywhere within the website?

No, we don’t! (we can, but we do not need to)
We can use ***relative paths***.



HTML Elements: Links

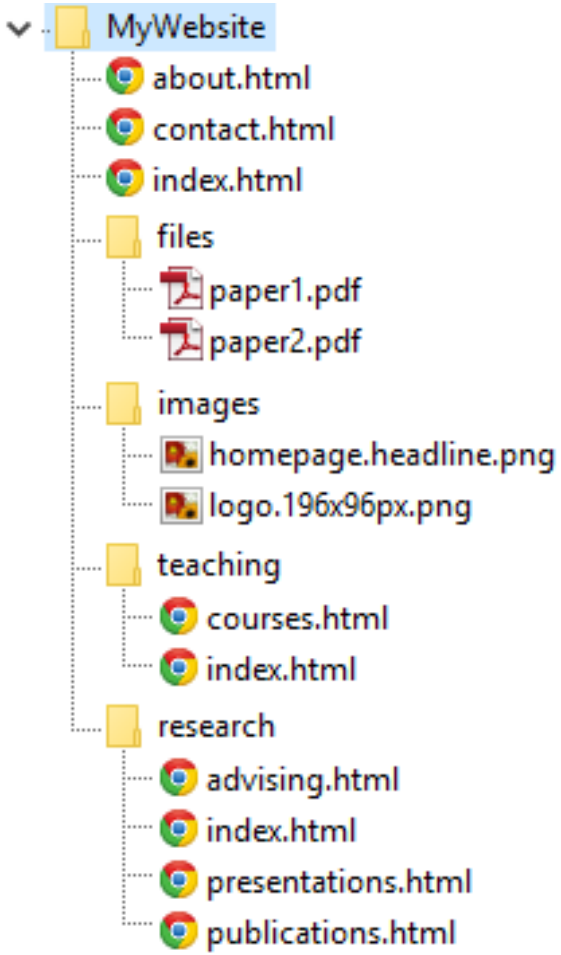
Directory Structure: URLs and relative paths

Let's say that you are on the website homepage:
http://www.mywebsite.com/index.html

To visit courses.html under the "teaching" folder, the path we can use is (navigating to child folder):
href="teaching/courses.html"

If you wanted to visit about.html in the root folder, the path we can use is (within the same folder):
href="about.html"

This is the ***relative path*** between various files – you are telling the browser the location of a file, relative to the current path
(or, *relative to which page you are on at the moment*)



HTML Elements: Links

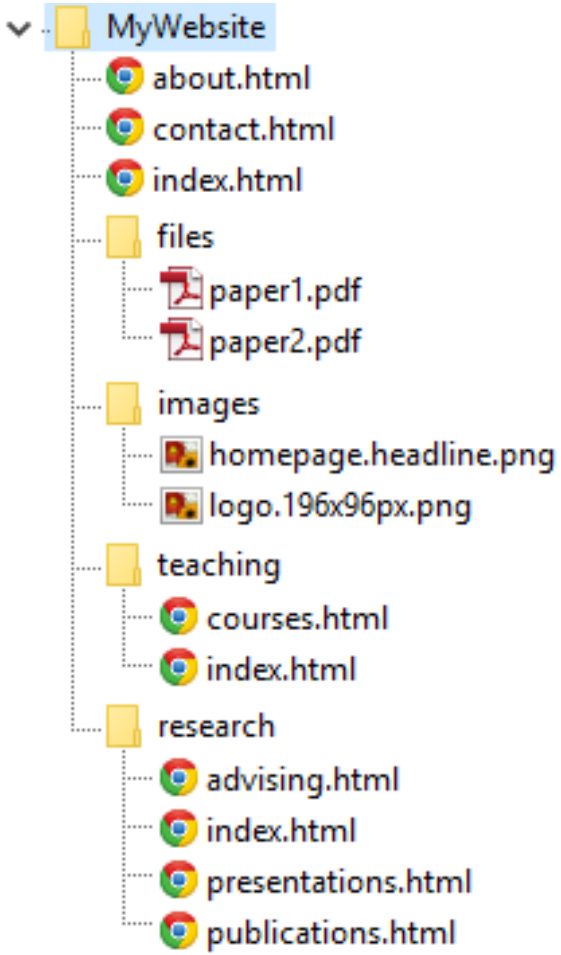
Directory Structure: URLs and relative paths

Let's say that you are on the advising page, under research:
http://www.mywebsite.com/research/advising.html

To visit contact.html under the root folder, the path we can use is (navigating to parent folder):
href="../contact.html"

If you wanted to visit courses.html in the teaching folder, the path we can use is (navigating to another child folder):
href="../teaching/contact.html"

The two dots ".." tell your browser that anything you type in after the dots are inside the parent folder (establishing relationship).

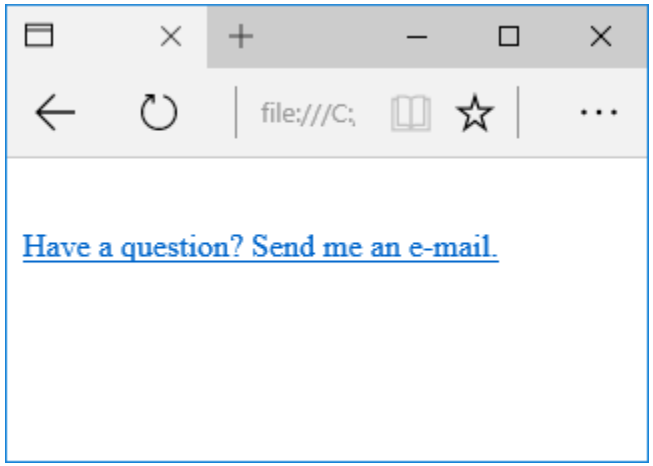


HTML Elements: Links

The **href** attribute: Examples.

[4] Linking to another web page on the same site **(Relative URLs)**:

```
<a href="contact.html">Have a question? Send me an e-mail.</a>
```



HTML Elements: Links

The **target** attribute.

Sometimes, you might consider opening links in a new tab or new window (typically considered when opening external links).

For example, if we were linking to anotherwebsite.com from mywebsite.com, we could consider letting the user view the other website, ***while retaining our website in the current tab/window.***

For this, you use the **target** attribute of the anchor <a> element.

HTML Elements: Links

The **target** attribute.

Values you can assign to the target attribute:

target	Description
_blank	Opens the linked document in a new tab or window
_self	Opens the linked document in the same frame (this is the default behaviour)
_parent	(Used with a frame*) Opens the linked document in the parent frame
_top	(Used with a frame*) Opens the linked document in the full body of the browser window
framename	Opens the linked document in a named frame*

**Frames are not preferred or used now.*

Key Ideas

HTML Elements

- Text: structural and semantic markup

- Lists: unordered, ordered and description/definition lists

- Links: directory structure, relative paths / URLs,
  opening pages in same/different browser windows