CSIT884:
Web Development

CSS

School of Computing and Information Technology
University of Wollongong

Cascading Style Sheets

CSS provides a separation between the HTML document **content** and document **presentation** (style).

3 ways to add styling to HTML elements:

- **Inline**

  using a `style` attribute in HTML elements

- **Document**

  using `<style>` element in the HTML `<head>` section

- **External**

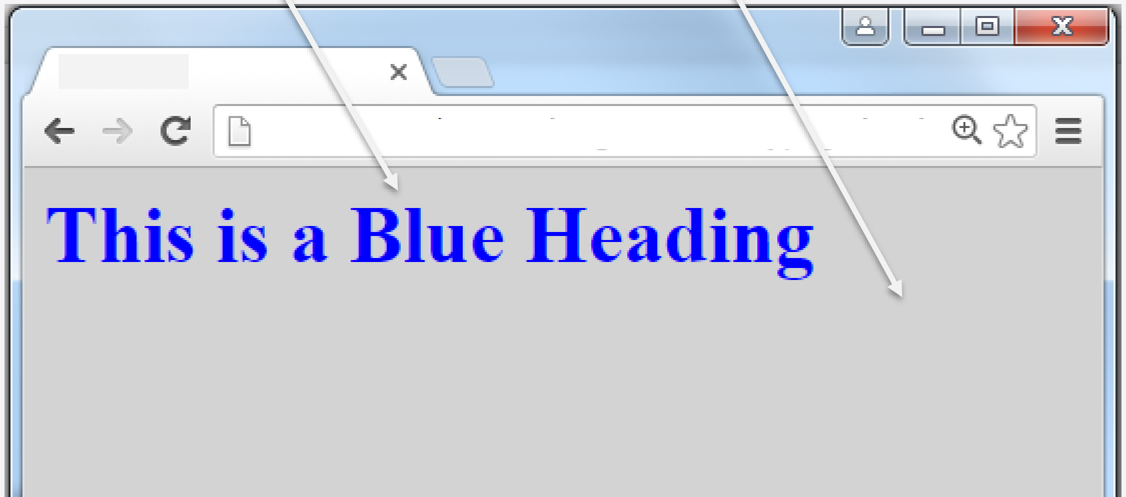
  using external **CSS files**

Inline CSS

By using a **`style`** attribute in HTML elements

```
<body style="background-color:lightgrey;">
```

```
<h1 style="color:blue;">This is a Blue Heading</h1>
```



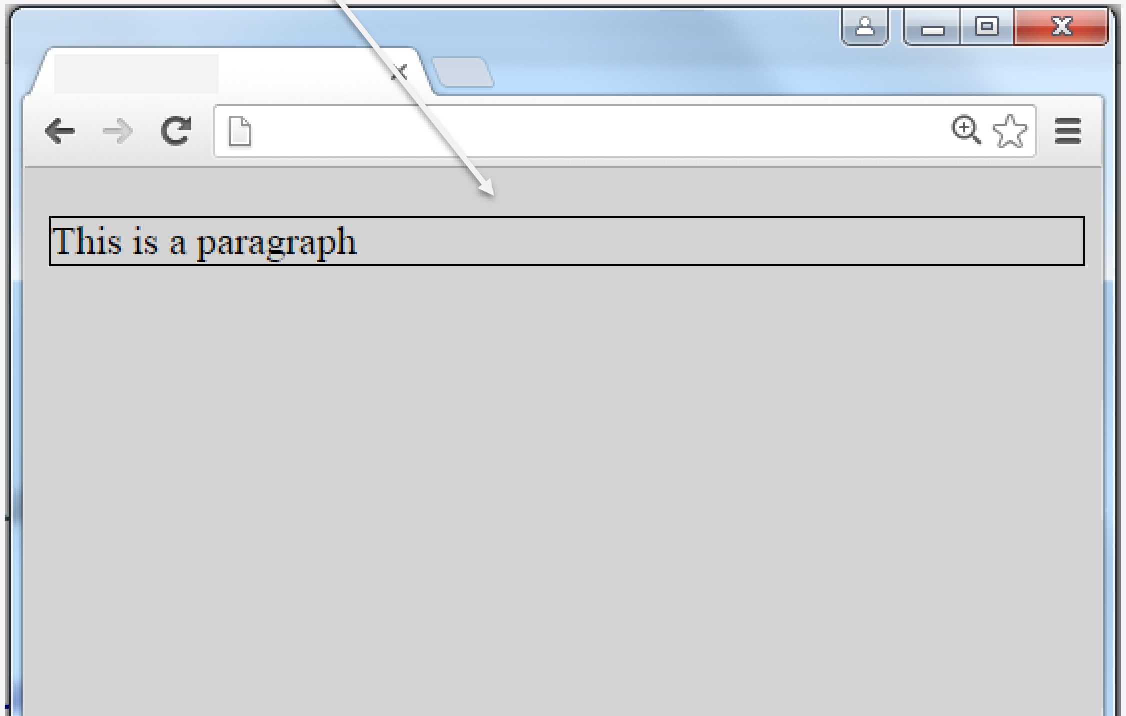
This is a Blue Heading

Inline CSS

```
<p style="border:1px solid black;">

This is a paragraph with border

</p>
```



This is a paragraph

Inline CSS

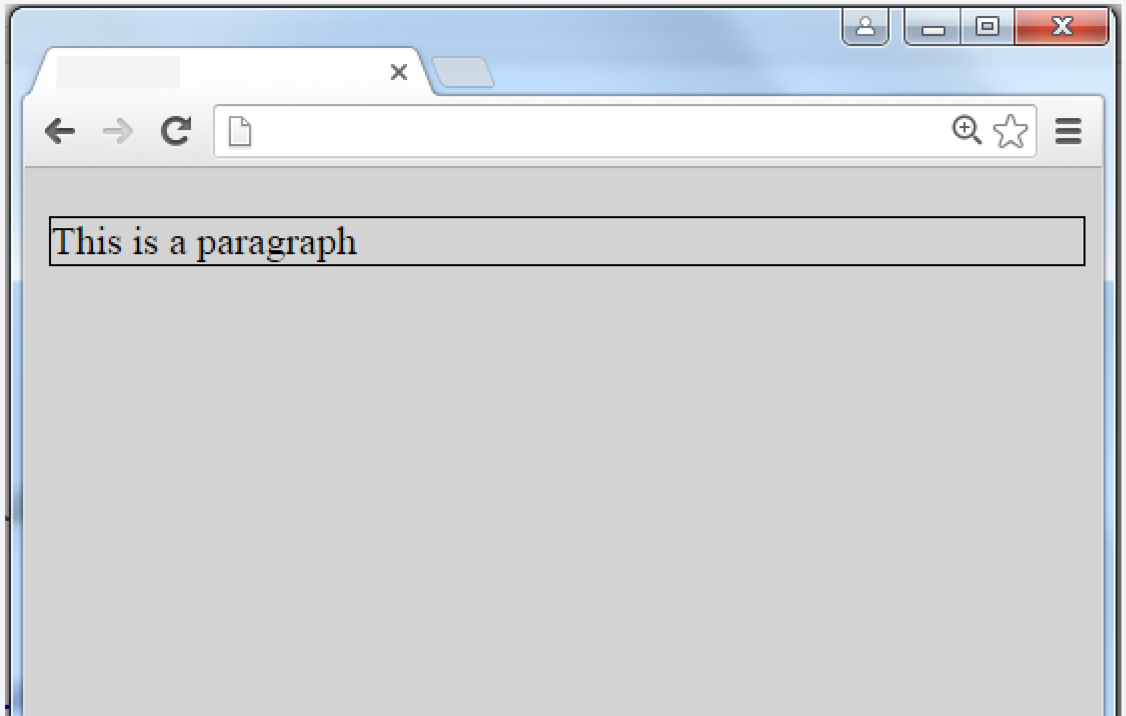
this is called a CSS **property**

```
<p style="border:1px solid black;">
```

CSS property-value pair

```
This is a paragraph with border

</p>
```



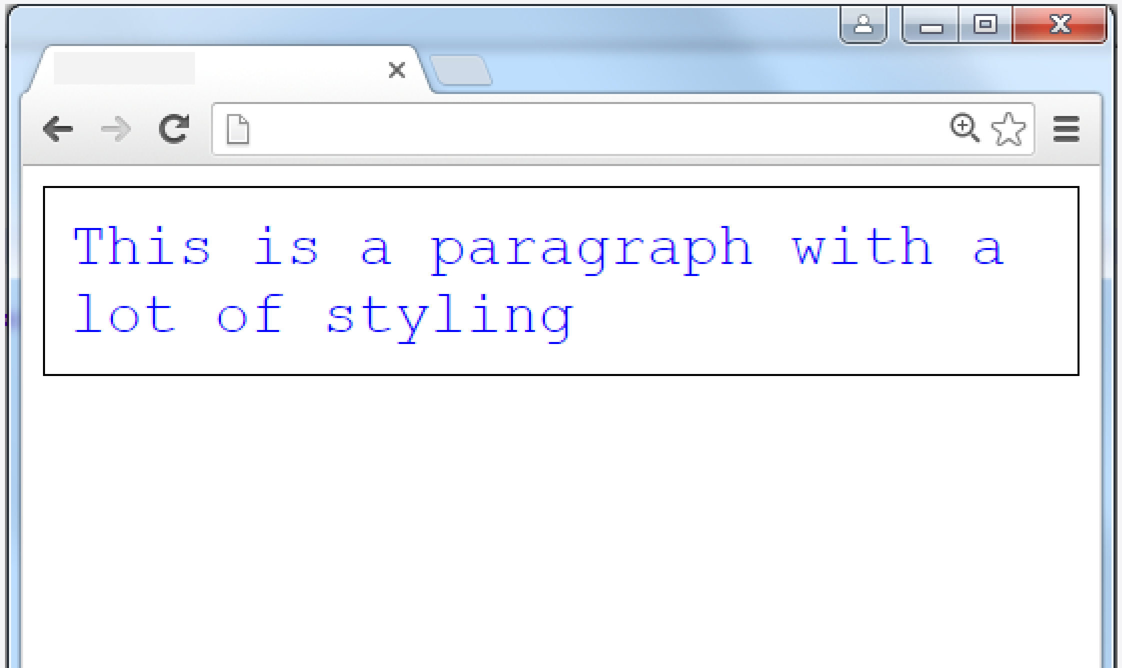
This is a paragraph

Inline CSS

```
<p style="border:1px solid black; padding:10px; color:blue;
font-family:courier; font-size:150%;">

This is a paragraph with a lot of styling

</p>
```



This is a paragraph with a lot of styling

Inline CSS

```
<p style="border:1px solid black; padding:10px; color:blue;
font-family:courier; font-size:150%;">

This is a paragraph with a lot of styling

</p>
```

- A CSS style is specified with the following format

  `property:value`

- We can specify more than one CSS property, separated by a semicolon (`;`)

  `style="border:1px solid black; padding:10px; color:blue; font-family:courier; font-size:150%;"`

- A CSS property may have many values separated by space

  `border:1px solid black`

Color

CSS supports 140 standard color names.

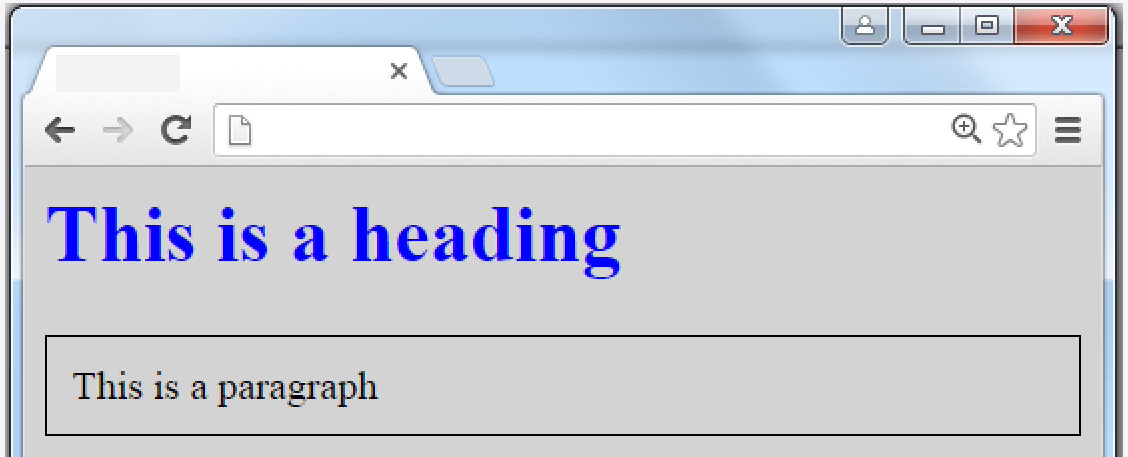
Color can also be specified by hex code.

```
<h1 style="color:lightgrey;">This is a Light Grey Heading</h1>

<h1 style="color:#D3D3D3;">This is a Light Grey Heading</h1>
```

Document CSS (Internal or Embedded)

```
<html>
<head>
<title>W3</title>
<style>
body {background-color:lightgrey;}
h1 {color:blue;}
p {border:1px solid black; padding:10px;}
</style>
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph</p>
</body>
</html>
```



This is a heading

This is a paragraph

External CSS

```html
<html>
<head>
<title>W3</title>
<link rel="stylesheet" href="path/to/mystyle.css">
</head>
<body>
<h1>This is a heading</h1>
<p>This is a paragraph</p>
</body>
</html>
```

or

```html
<style>
@import url("path/to/mystyle.css");
</style>
```

mystyle.css

```css
body {background-color:lightgrey;}
h1 {color:blue;}
p {border:1px solid black; padding:10px;}
```

CSS convention

This is a valid CSS →

mystyle.css

```
body {background-color:lightgrey;}
h1 {color:blue;}
p {border:1px solid black; padding:10px;}
```

But for better clarity, we should use the following convention:

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

h1 {
  color:blue;
}

p {
  border:1px solid black;
  padding:10px;
}
```

each property on a separate line

Levels of CSS

- Inline CSS has precedence over document CSS
- Document CSS has precedence over external CSS
- Suppose an external CSS specifies a value for a particular property of a HTML element, then that value can be overridden by a document CSS, which in turn, can be overridden by an inline CSS.

Simple selector

This is called a simple selector

```
p {
  border:1px solid black;
  padding:10px;
}
```

declarations

We can also have grouped selectors

The style will be applied to all **<h1>** and **<h2>** elements

```
h1, h2 {
  border:1px solid black;
  color:lightgrey;
}
```

Class selector

```
<h1 class="userInfo">This is a heading 1</h1>
<p class="userInfo">This is a paragraph 1</p>
<h2 class="userInfo">This is a heading 2</h2>
<p class="userInfo">This is a paragraph 2</p>

<h1 class="eticket">This is a heading</h1>
<p class="eticket">This is a paragraph</p>
<h2 class="eticket">This is a heading</h2>
```

This style will be applied to all `<p>` elements of `class` `userInfo` →

```
p.userInfo {
  border:1px solid black;
  padding:10px;
}
```

This style will be applied to all `<h1>` and `<h2>` elements of `class userInfo` →

```
h1.userInfo, h2.userInfo {
  color:blue;
}
```

Class selector

```
<h1 class="userInfo">This is a heading 1</h1>
<p class="userInfo">This is a paragraph 1</p>
<h2 class="userInfo">This is a heading 2</h2>
<p class="userInfo">This is a paragraph 2</p>

<h1 class="eticket">This is a heading</h1>
<p class="eticket">This is a paragraph</p>
<h2 class="eticket">This is a heading</h2>
```

This style will be applied to
all elements of `class eticket` →

```
.eticket {
  color:green;
}
```

Id selector

```
<h1 id="userHeading">This is a heading 1</h1>
<p id="userDetails">This is a paragraph 1</p>

<h2 id="bankHeading">This is a heading 2</h2>
<p id="bankDetails">This is a paragraph 2</p>
```

This style will be applied to the element with `id userHeading` →

```
#userHeading {
  color:blue;
}
```

Note that each HTML element should have a unique id

Descendant-Ancestor

An element F is a *descendant* of element E if it appears in the content of E.
In this case, E is called an ancestor of F.

```
<E>
   …
   <F>
   …
</E>
```

```
<E>
  <E2>
     …
     <F>
     …
  </E2>
</E>
```

```
<E>
  <E2>
    <E3>
       …
       <F>
       …
    </E3>
  </E2>
</E>
```

Descendant-Ancestor

What are the descendants of this element `div` ?

Example:

```
<div>
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div>
    This is the final <i>italic</i>.
  </div>
</div>
```

Child-Parent

An element F is a *child* of element E if it is nested directly in the content of E.
In this case, E is called a parent of F.

```
<E>
   …
   <F>
   …
</E>
```

Of course, if F is a child of E then F is also a descendant of E.

Child-Parent

What are the children of this element `div` ?

Example:

```
<div>
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div>
    This is the final <i>italic</i>.
  </div>
</div>
```

Child-Parent

What are the children of this element `div` ?

Example:

```
<div>
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div>
    This is the final <i>italic</i>.
  </div>
</div>
```

Contextual Selector

Descendant selector
Apply this style to every **descendant** F of E. . . . (1)

```
E F {
  property:value
  …
}
```

Child selector
Apply this style to every **child** F of E. . . . (2)

```
E > F {
  property:value
  …
}
```

Next-sibling selector
Apply this style to every element F that shares same parent and comes directly after G. . . . (3)

```
G + F {
  property:value
  …
}
```

Subsequent-sibling selector
Apply this style to every element F that shares same parent as G and comes after it. . . . (4)

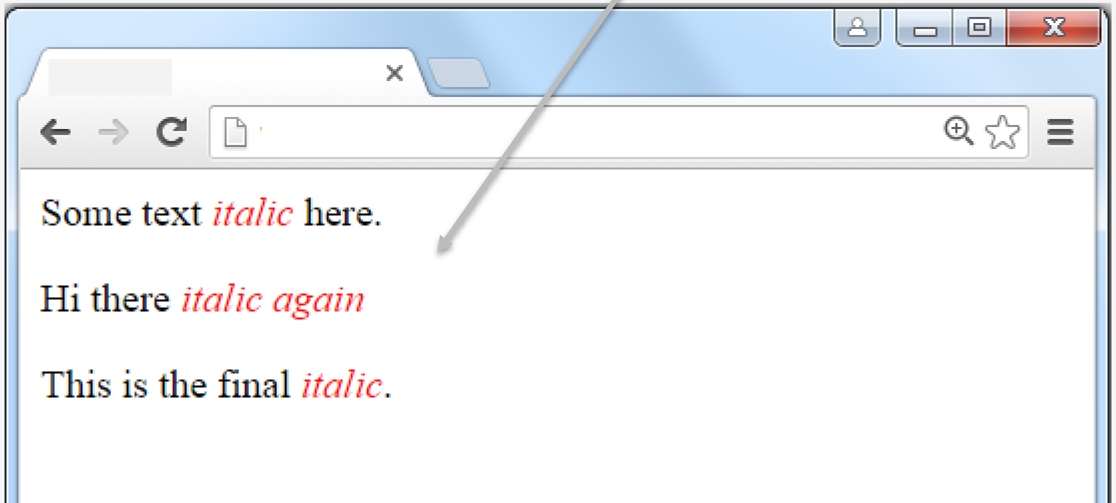
```
G ~ F {
  property:value
  …
}
```

Contextual Selector

Example:

```
<div>
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div>
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div i {
  color:red;
}
```



Some text *italic* here.

Hi there *italic again*

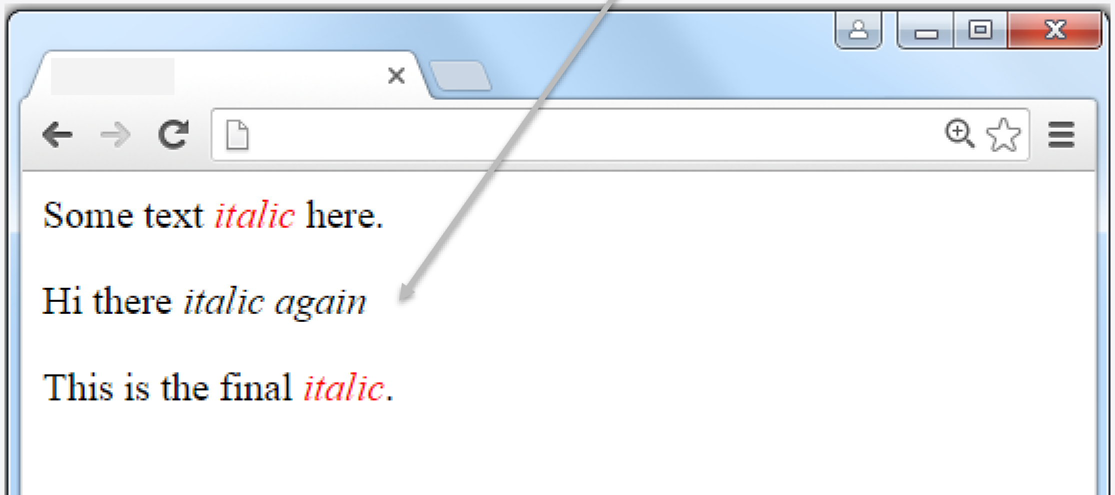
This is the final *italic*.

Contextual Selector

Example:

```
<div>
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div>
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div > i {
  color:red;
}
```



Some text *italic* here.

Hi there *italic again*

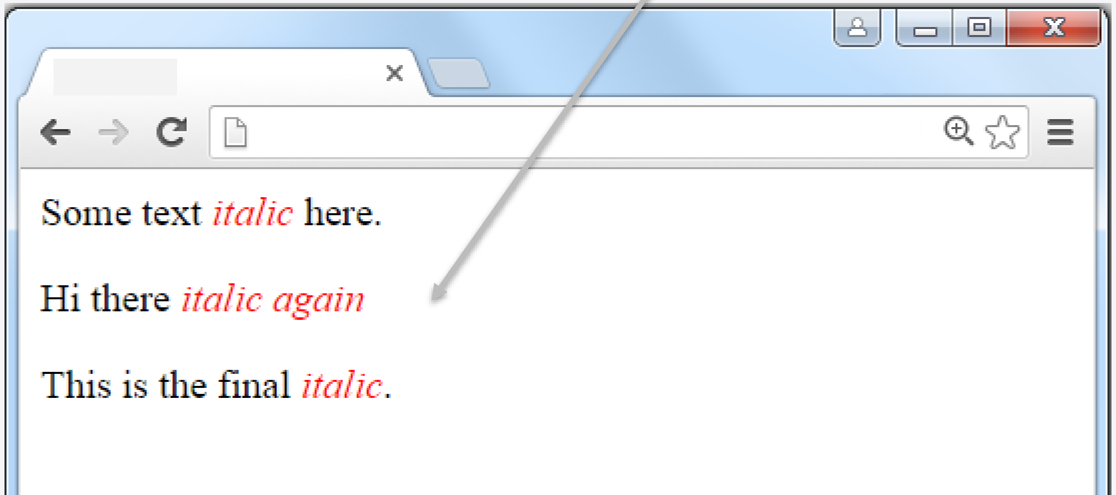
This is the final *italic*.

Contextual Selector

Example:

```
<div class="userInfo">
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div class="bankInfo">
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div.userInfo i {
  color:red;
}
```



Some text *italic* here.

Hi there *italic again*

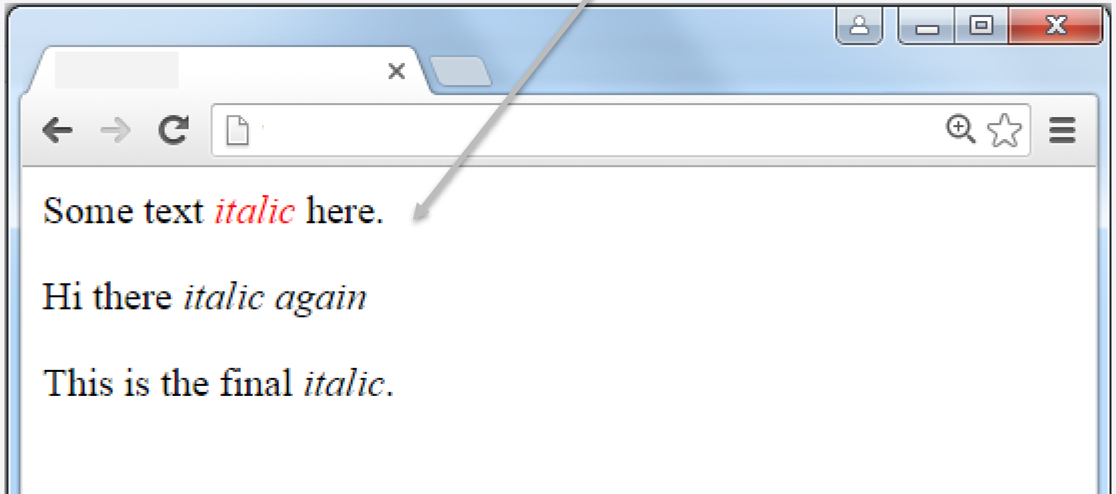
This is the final *italic*.

Contextual Selector

Example:

```
<div class="userInfo">
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div class="bankInfo">
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div.userInfo > i {
  color:red;
}
```



Some text *italic* here.

Hi there *italic again*

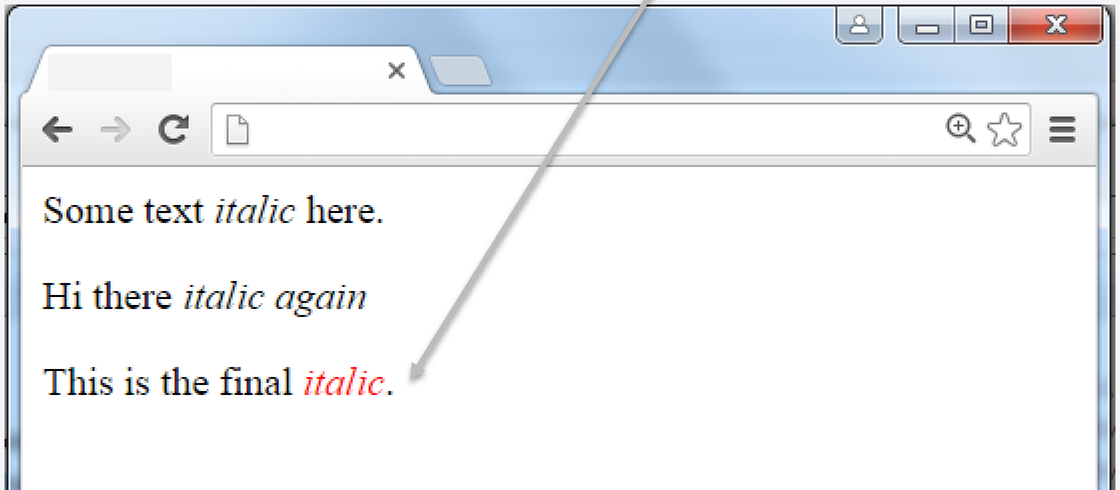
This is the final *italic*.

Contextual Selector

Example:

```
<div class="userInfo">
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div class="bankInfo">
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div.bankInfo i {
  color:red;
}
```


Some text *italic* here.

Hi there *italic again*

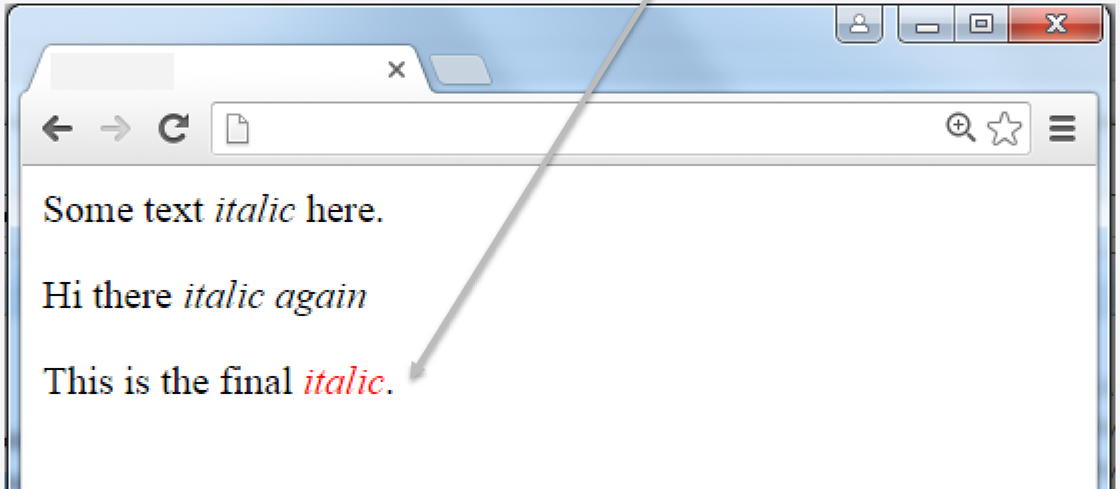
This is the final *italic*.

Contextual Selector

Example:

```
<div class="userInfo">
Some text <i>italic</i> here.
  <p>
    Hi there <i>italic again</i>
  </p>
  <div class="bankInfo">
    This is the final <i>italic</i>.
  </div>
</div>
```

```
div.bankInfo > i {
  color:red;
}
```


Some text *italic* here.

Hi there *italic again*

This is the final *italic*.

Pseudo class selector

```
<a href="http://www.uow.edu.au">UOW</a>
```

The `link` pseudo class is used to style a link that has not been selected.

The `visited` pseudo class is used to style a link that previously has been selected.

```
a:link {
  color:red;
}

a:visited {
  color:green;
}
```

```
h1:hover {
  color:blue;
}
```

```
<h1>A heading</h1>
```

Any time the mouse cursor is position over the `h1` element then the style will be applied.

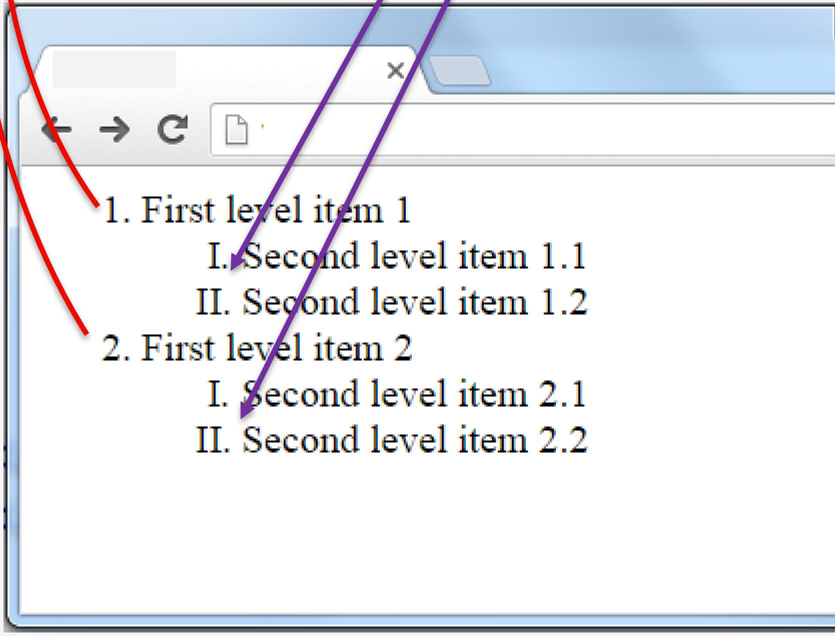
List properties

```
ol {
  list-style-type:decimal;
}

ol ol {
  list-style-type:upper-roman;
}
```

```
<ol>
  <li>First level item 1
    <ol>
      <li>Second level item 1.1</li>
      <li>Second level item 1.2</li>
    </ol>
  </li>

  <li>First level item 2
    <ol>
      <li>Second level item 2.1</li>
      <li>Second level item 2.2</li>
    </ol>
  </li>
</ol>
```



```
1. First level item 1
      I. Second level item 1.1
     II. Second level item 1.2
2. First level item 2
      I. Second level item 2.1
     II. Second level item 2.2
```

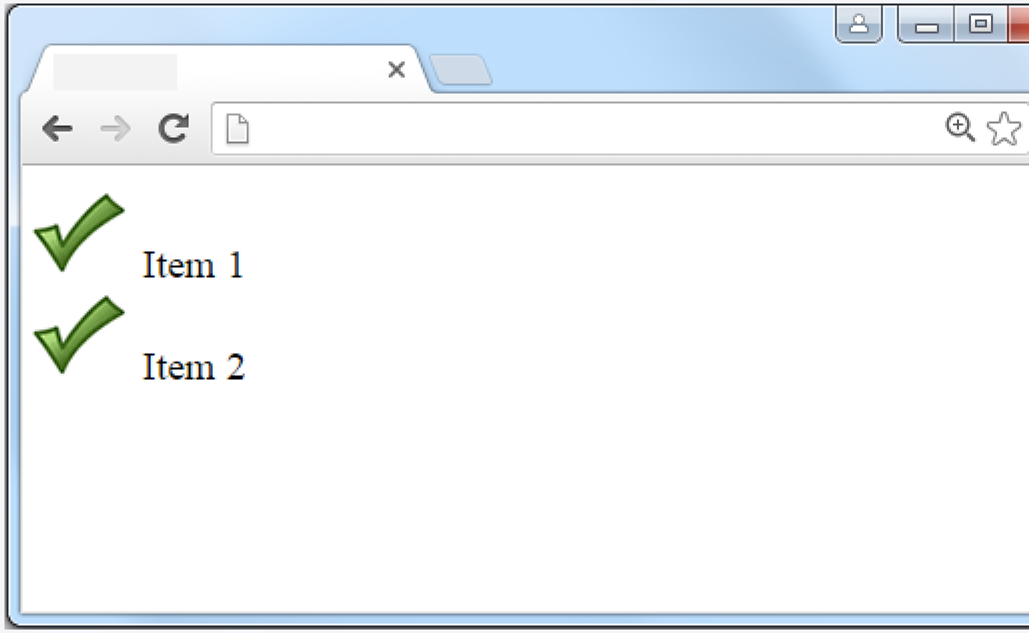
other values: `decimal-leading-zero, lower-alpha, lower-latin, lower-greek, disc, square, circle`

List properties

```
ol {
  list-style-image:url(path/to/imagefile);
}
```

```
<ol>
  <li>Item 1</li>

  <li>Item 2</li>
</ol>
```



Item 1

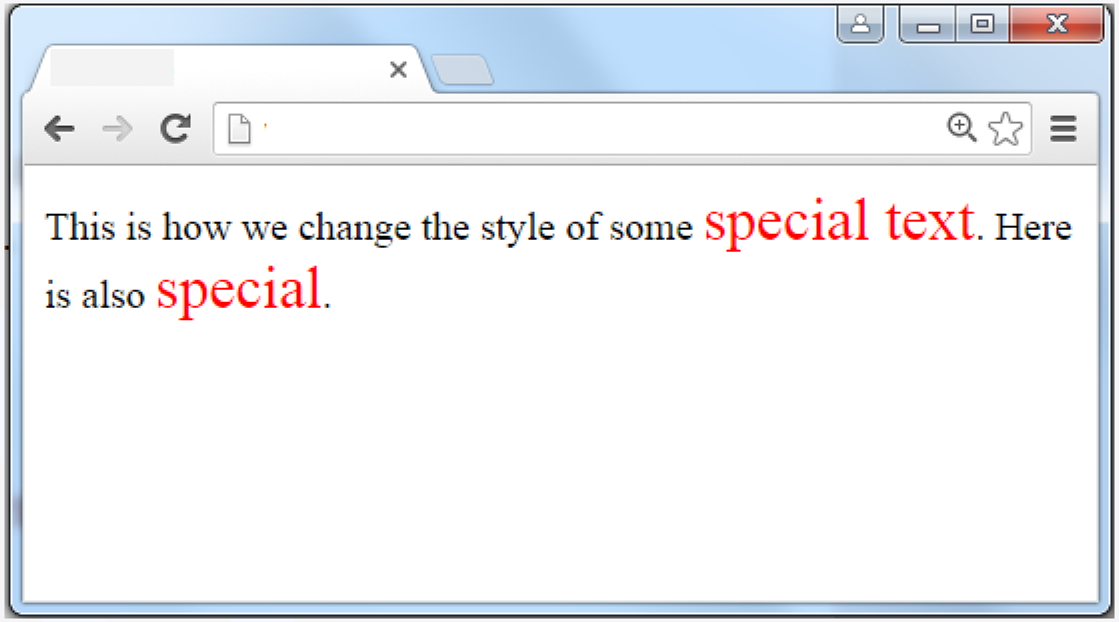
Item 2

span

Sometimes it is useful to have a word or phrase in a line to appear in a different style, we use `<span>… </span>` for this purpose.

```
This is how we change the style of some
<span class="specialText">special text</span>.
Here is also
<span class="specialText">special</span>.
```

```
span.specialText {
  color:red;
  font-family:Ariel;
  font-size:150%;
}
```



This is how we change the style of some special text. Here is also special.

div

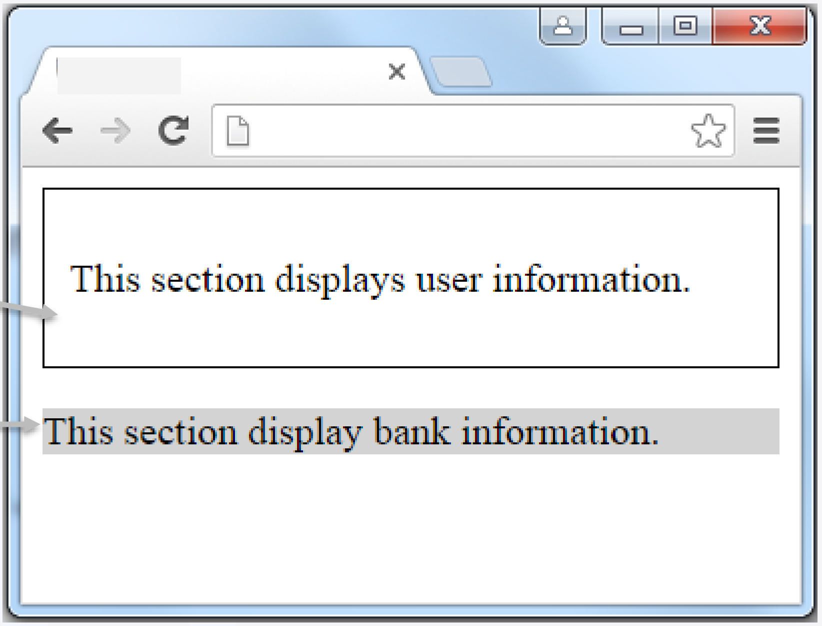
Sometimes we want to have different style at different section of the webpage, we use `<div>… </div>` for this purpose.

```html
<div class="userInfo">
<p>This section displays user information.</p>
</div>

<div class="bankInfo">
<p>This section display bank information.</p>
</div>
```

```css
div.userInfo {
  border:1px solid black;
  padding:10px;
}

div.bankInfo {
  background-color:lightgrey;
}
```



This section displays user information.

This section display bank information.

Comments in CSS

A comment starts with /* and ends with */

Comments can span over multiple lines.

```
p {
      border:1px solid black;

      /* This is a single-line comment */

      color:blue;
}

/* This is

a multi-line
comment */
```