Introduction to Web Technology

CSS

Joseph Tonien
School of Computing and Information Technology
University of Wollongong

CSS

Objectives:

- understand the need of separation between the content and the style of your website
- learn 3 different ways to define web page styles
- use CSS language to define styles for your web pages

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**

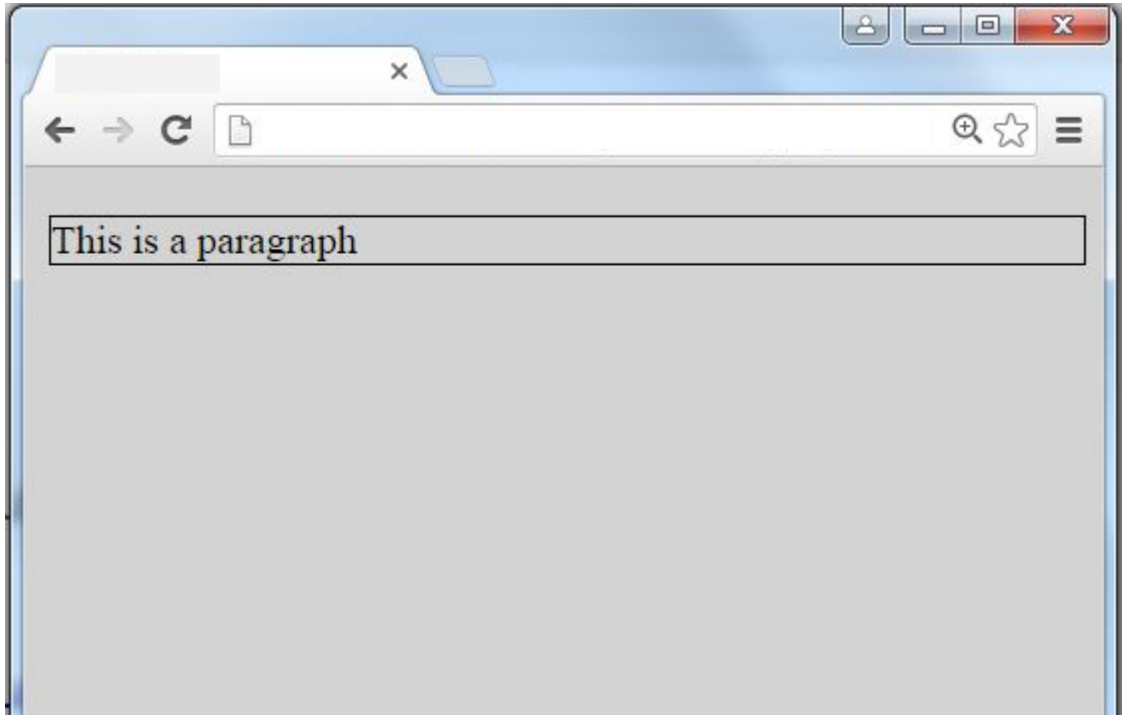
Inline CSS

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

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

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

Inline CSS

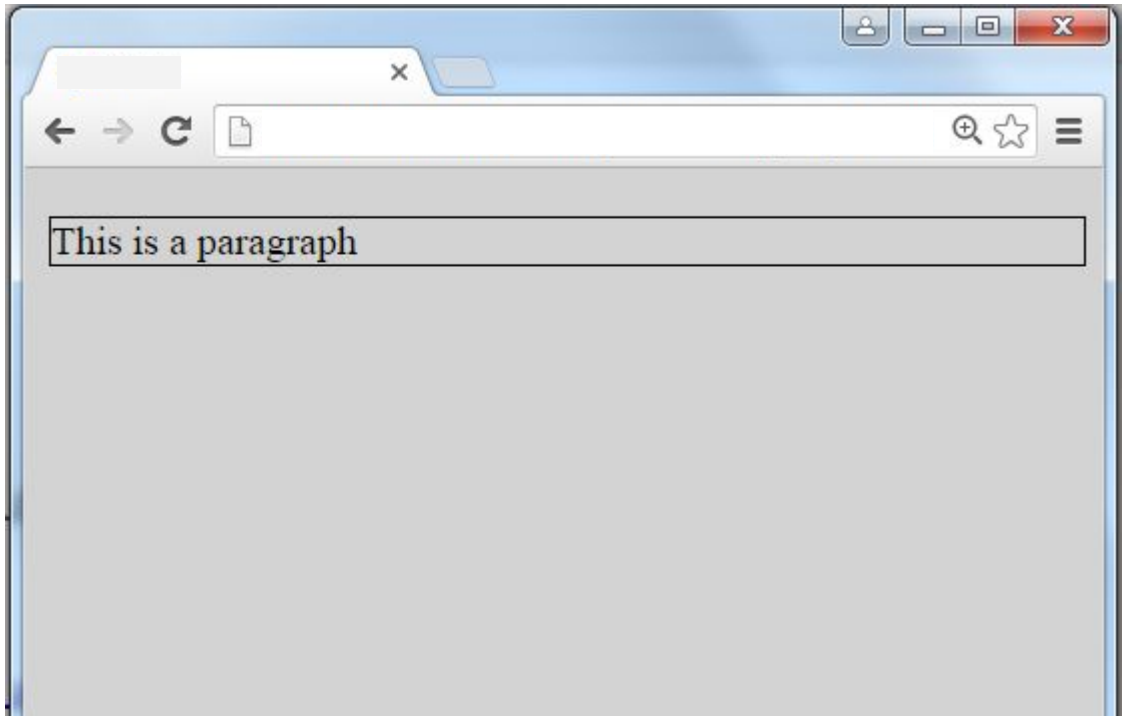
```
<p style="border:1px solid black;">

This is a paragraph with border

</p>
```



Inline CSS

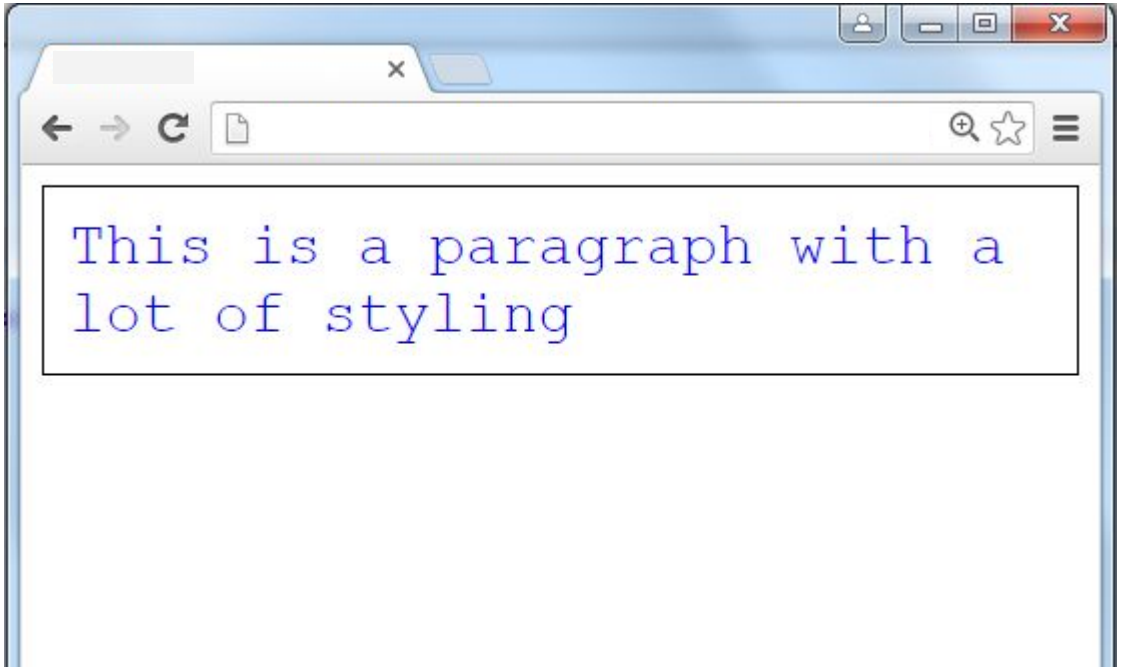
this is called a CSS **property**

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

This is a paragraph with border

</p>
```



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>
```



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

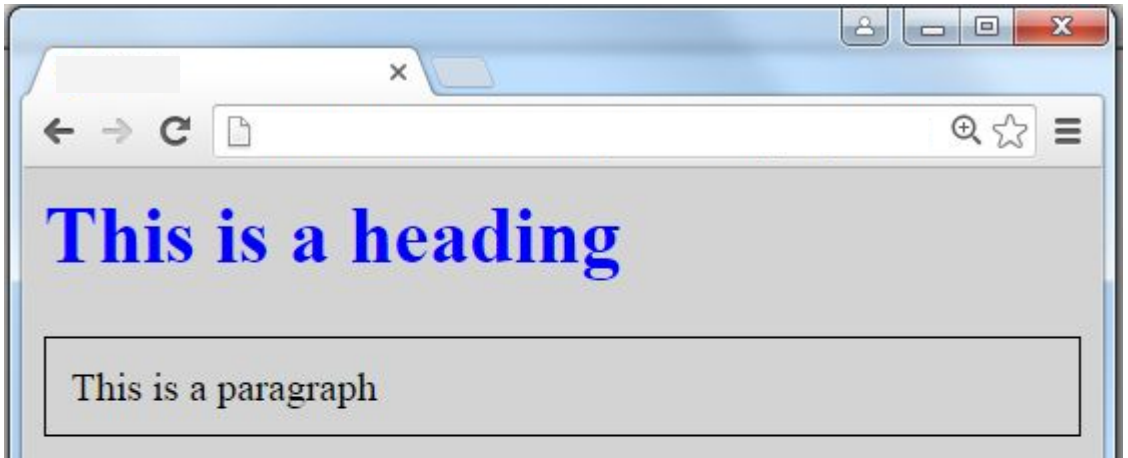
CSS supports 140 standard color names.

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>
```

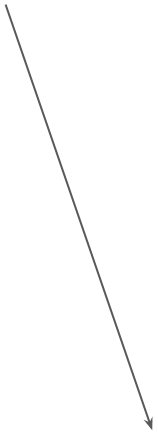
Document CSS

```
<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>
```



External CSS

```
<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>
```



mystyle.css

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

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.

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

Simple selector

This is called a simple selector

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

We can also have this simple selector.

In this case, all `<h1>` and `<h2>` elements will be applied with this style.

```
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>
```

All <p> elements of class userInfo will be applied with this style.


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

All <h1> and <h2> elements of class userInfo will be applied with this style.


```
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>
```

All elements of `class eticket` will be applied with this style.


```
.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>
```

The element with `id` `userHeading` will be applied with this style.


```
#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>
```

Contextual Selector

Apply this style to every **descendant** `F` of `E`

```
E F {
  property:value
  …
}
```

Apply this style to every **child** `F` of `E`

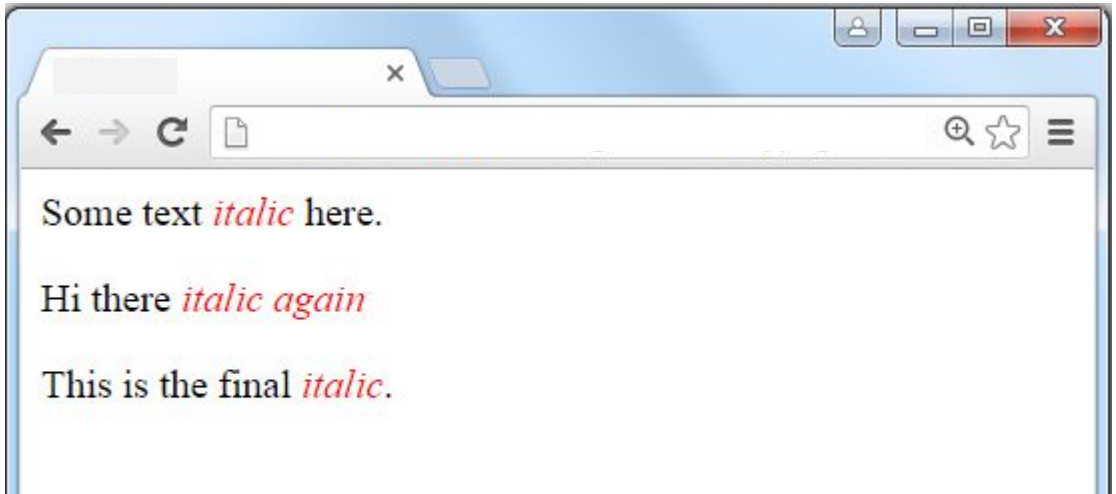
```
E > F {
  property:value
  …
}
```

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;
}
```

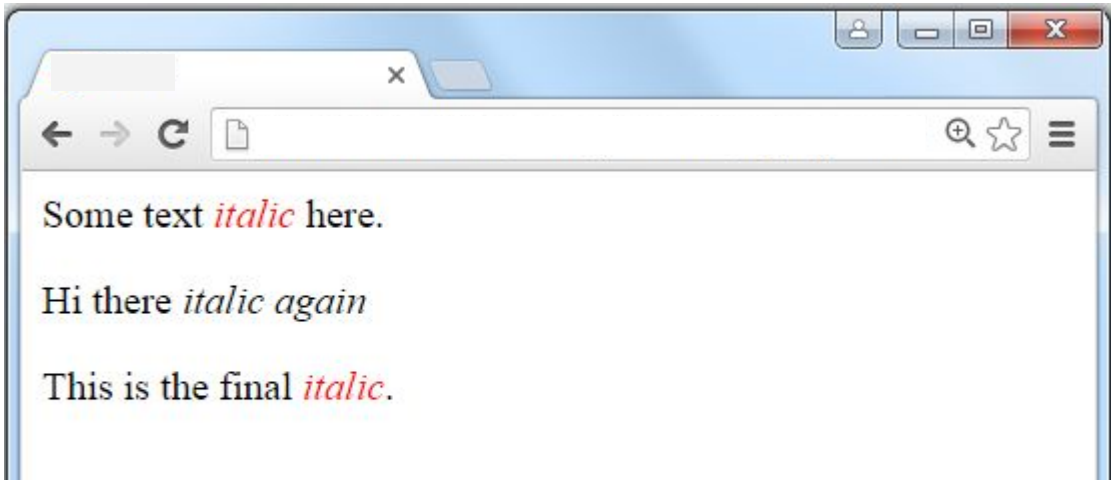


Contextual Selector

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;
}
```

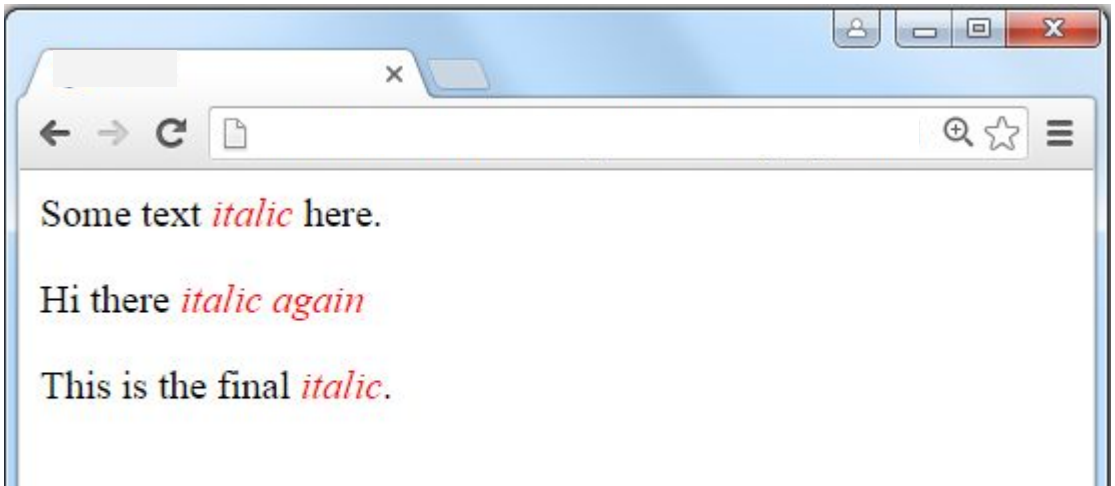


Contextual Selector

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;
}
```

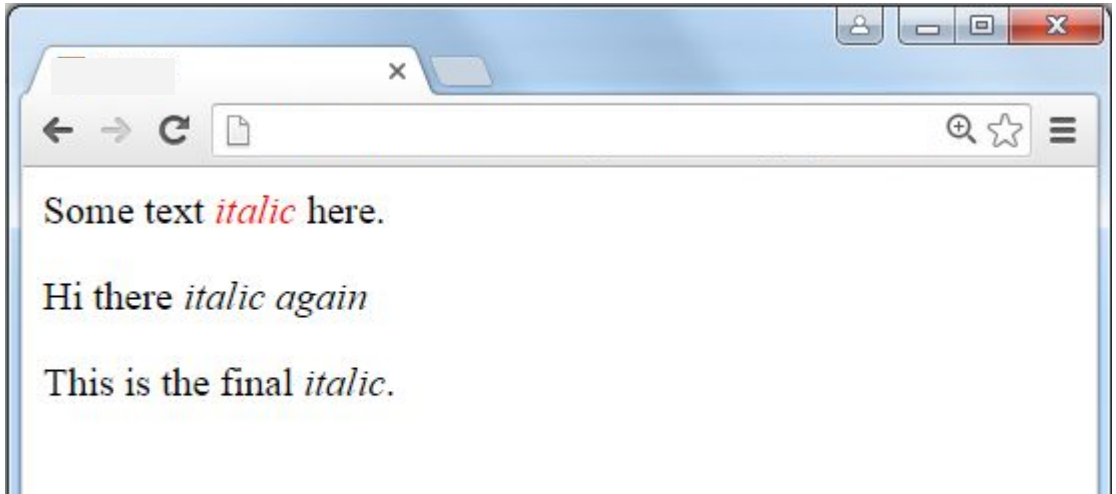


Contextual Selector

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;
}
```

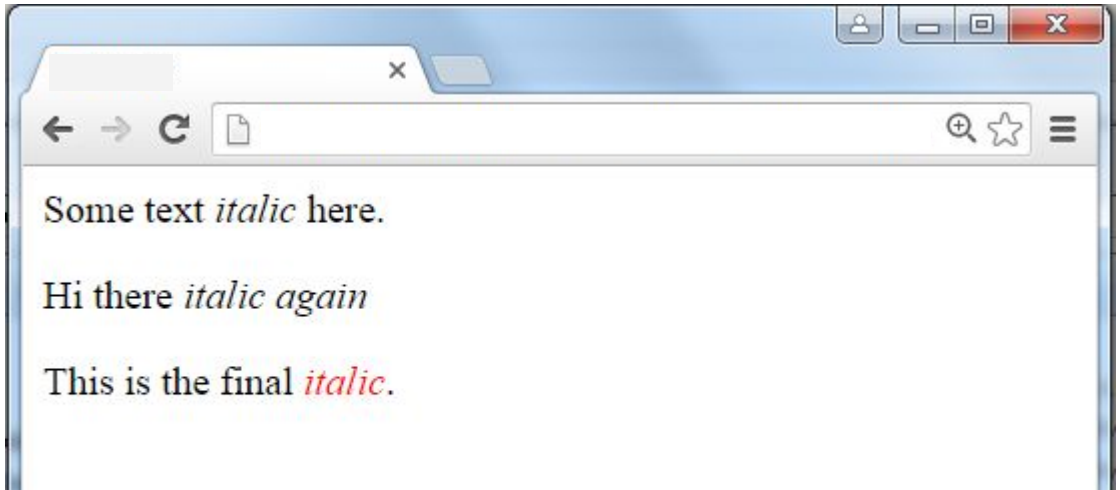


Contextual Selector

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;
}
```

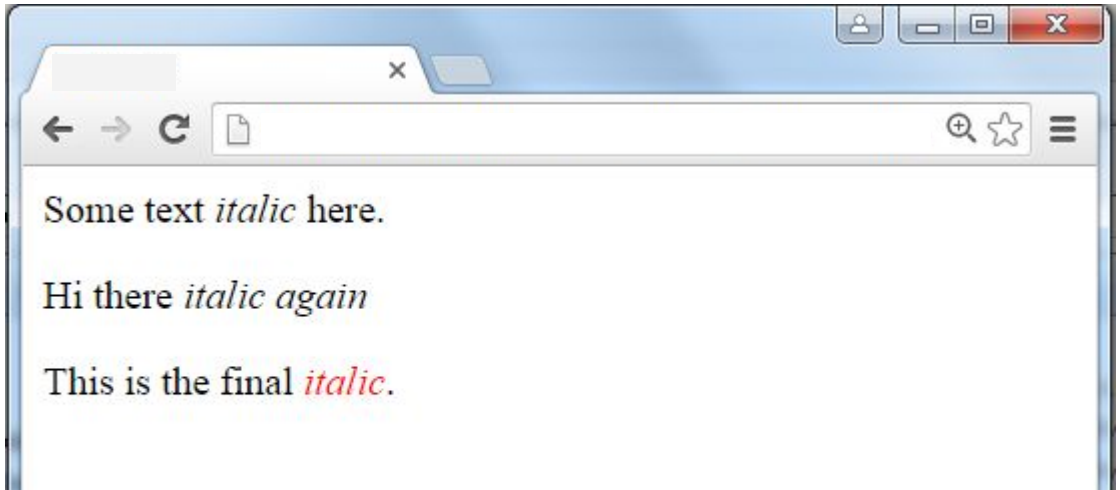


Contextual Selector

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;
}
```



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>
```

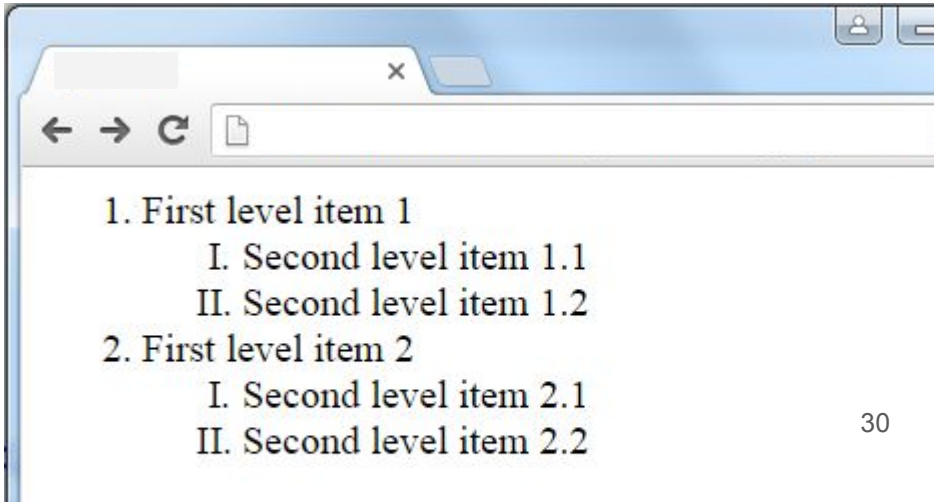
Any time the mouse cursor is position over the `h1` element then the style will be applied.

List properties

```
<ol>
  <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>
```

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

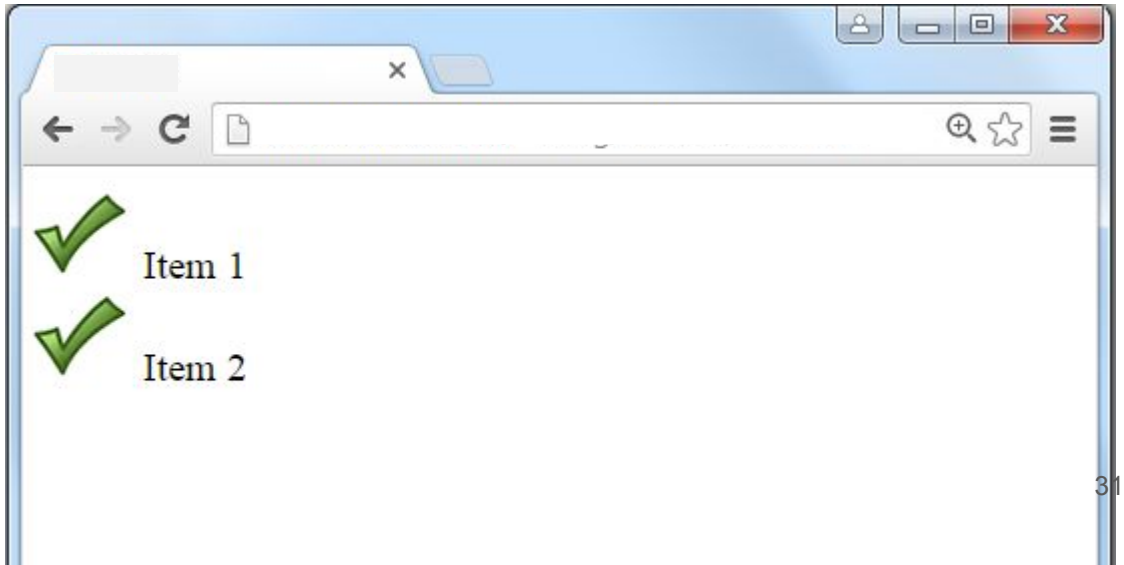
```
ol {
  list-style-type:decimal;
}

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



List properties

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

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

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

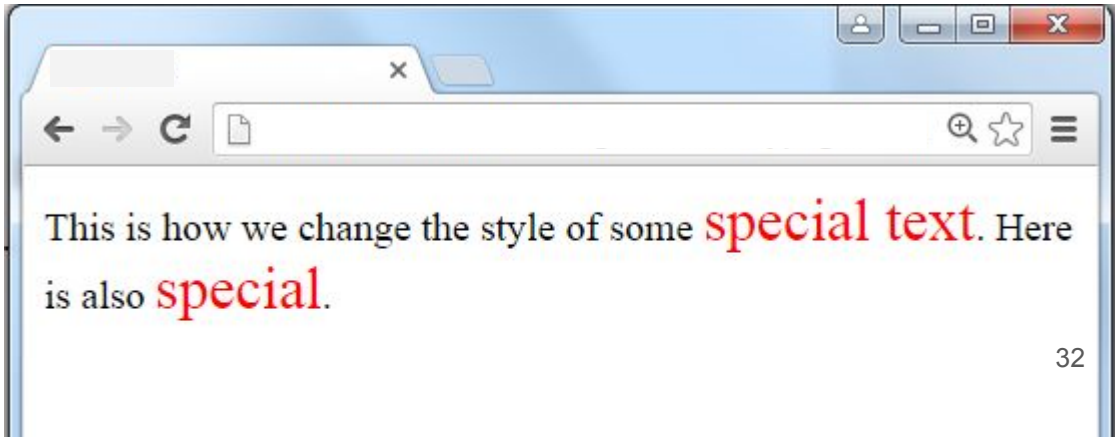


span

Sometimes it is useful to have a word or phrase in a line appear in a different style, we use … 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%;
}
```

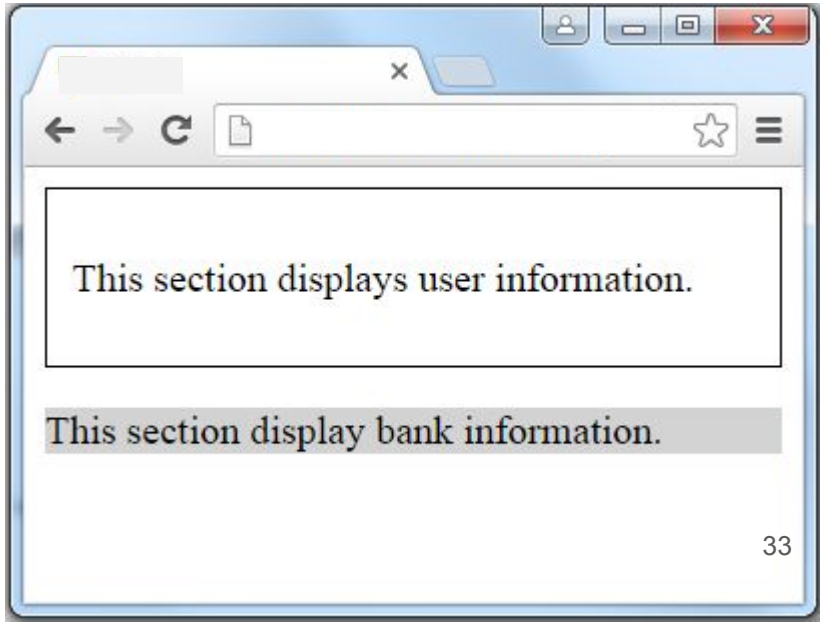


div

Sometimes we want to have different style at different section of the webpage, we use … for this purpose.

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

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

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

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



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 */
```

References

- http://www.w3schools.com/css
- https://en.wikipedia.org/wiki/Cascading_Style_Sheets
- https://developer.mozilla.org/en-US/docs/Web/CSS/Reference