Web Applications

Name: _____ ( ) Class: _____ Date: _____

Lesson 3: Cascading Style Sheets

Instructional Objectives:

By the end of this task, you should be able to:

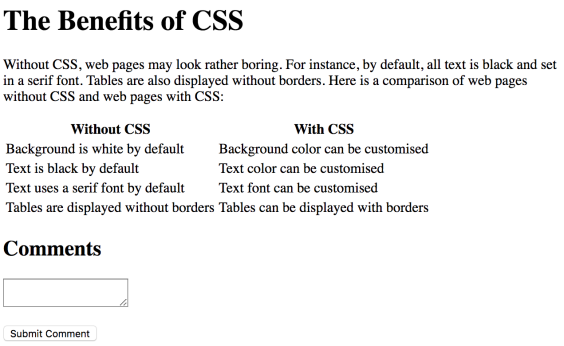
- Explain how CSS only describes the presentation of a web page and not its structure or contents
- Use the `<link>` tag to associate a web page to a CSS style sheet
- Identify the rules, selectors, declarations and properties that are present in a given CSS style sheet
- Use an element selector to limit effect of a CSS rule to a particular element type
- Use an `id` attribute in HTML and an `id` selector in CSS to limit effect of a CSS rule to a single element
- State that it is an error for a web page to have more than one HTML element with the same `id` attribute
- Use a `class` attribute in HTML and a `class` selector in CSS to limit effect of a CSS rule to elements with the specified class
- Use a descendent selector in CSS to limit the effect of a CSS rule to elements that are nested in a particular order
- Distinguish between elements that have a block appearance by default and elements that have an inline appearance by default
- Identify and distinguish between the margin, border and padding of elements that have a block appearance
- Use the `<div>` tag to represent structural elements that should have a block appearance by default and are not represented by other HTML tags

- Use the `<span>` tag to represent structural elements that should have an inline appearance by default and are not represented by other HTML tags
- Use the `display`, `background`, `color`, `height`, `width`, `border`, `border-bottom`, `border-left`, `border-right`, `border-top`, `margin`, `margin-bottom`, `margin-left`, `margin-right`, `margin-top`, `padding`, `padding-bottom`, `padding-left`, `padding-right`, `padding-top`, `font-family`, `font-size`, `font-style`, `font-weight`, `text-align` and `text-decoration` properties to control the presentation of a web page without producing any syntax errors

---

Part 1: Why Another Language?

So far, the web pages we have created look rather plain:



The Benefits of CSS

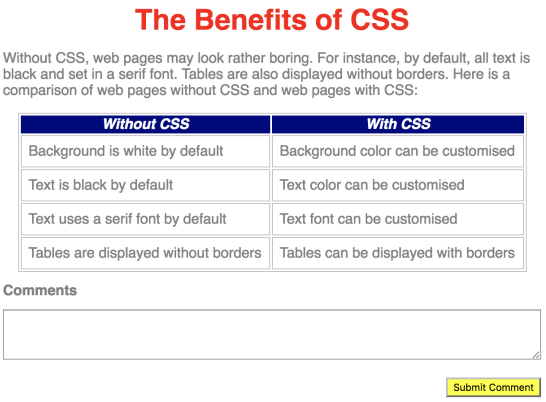
Without CSS, web pages may look rather boring. For instance, by default, all text is black and set in a serif font. Tables are also displayed without borders. Here is a comparison of web pages without CSS and web pages with CSS:

Without CSS	With CSS
Background is white by default	Background color can be customised
Text is black by default	Text color can be customised
Text uses a serif font by default	Text font can be customised
Tables are displayed without borders	Tables can be displayed with borders

Comments

Submit Comment

However, with the help of CSS, we can improve the appearance of such web pages greatly with almost no change to the HTML:



The Benefits of CSS

Without CSS, web pages may look rather boring. For instance, by default, all text is black and set in a serif font. Tables are also displayed without borders. Here is a comparison of web pages without CSS and web pages with CSS:

Without CSS	With CSS
Background is white by default	Background color can be customised
Text is black by default	Text color can be customised
Text uses a serif font by default	Text font can be customised
Tables are displayed without borders	Tables can be displayed with borders

Comments

Submit Comment

You may wonder why we need CSS to control a web page's appearance. The answer lies in a computer science principle called **separation of concerns**, where a program is divided into distinct sections such that each section only deals with one aspect of the final product and has minimal knowledge of the other parts.

In general, a web page's structure (how the content is organized) and presentation (how the content is displayed) are largely independent of each other and therefore should be handled separately. In particular, we define the web page's structure using HTML in one file and its presentation using CSS in a separate file. Separating these two components lets us lets us modify the web page's structure without affecting its presentation and vice versa. Thus, as an example, a person can be in charge of the content of the website, while another person can design how the website is presented without worrying over the content of the website. It also lets us specialise the HTML and CSS languages so that each language is more suited for its purpose.

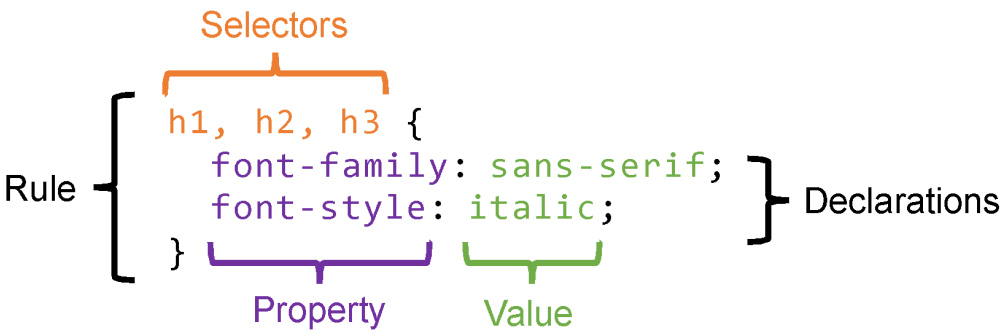
To demonstrate, open Google Chrome and visit `www.csszengarden.com`. Click on any of the available designs to see how changing CSS style sheets can dramatically affect the appearance of a web page without modifying its HTML. This is possible because the web page's structure and presentation are controlled separately. (You may view the source code by pressing Ctrl-U to check that the HTML is exactly the same for each design.)

Part 2: Anatomy of a CSS Style Sheet

Open Google Chrome, visit `www.example.com` and press Ctrl-U. Notice that the first part of the HTML document includes some CSS inside a `<style>` tag:

For simplicity, this web page has chosen to include its style sheet inside the HTML file using a `<style>` tag. However, in general we write our style sheet in a separate `.css` file and link it to a HTML file using a `<link>` tag. We will learn how to do this in the next part of this practical task.

Focusing on the CSS between the `<style>` and `</style>` tags, we see that the style sheet is made of multiple **rules**. Each rule starts with one or more **selectors** separated by commas, followed by curly braces surrounding a number of **declarations**. These terms are illustrated for a single rule as follows:



Each declaration is made of two parts: a **property** name and one or more values separated by spaces. Multiple declarations in a rule are separated by semicolons.

1 Circle all selectors and underline all properties in the following style sheet:

```
body {
    margin: 0;
```

```
    padding: 0;
}

h1, h2, #danger {
    border: 1px solid red;
}
```

2 For each code snippet below, place a tick in the corresponding box if the snippet is valid CSS. Note that `background` and `color` are both valid CSS property names.

Code Snippet	Valid?
`body (` `  background: white;` `)`	
`body {` `  background: white` `  color: black` `}`	
`body {` `  background: white;` `  color: black;` `}`	√
`{ color: black; }`	
`body { color: black; }`	√

Part 3: Your Own Style Sheet

Now that we know about the basics of CSS, it is time to create our own style sheet. Like HTML documents, CSS style sheets are just text files with a `.css` extension. You may wish to use Notepad++ to edit CSS files as it offers syntax highlighting. Open a text editor, create a new file and save it in your user directory as `example.css`. Then enter the following style sheet:

```
h1 { color: red; font-family: sans-serif; }
p { color: blue; font-style: italic; }
```

To use a stylesheet, we need an accompanying document. Create a new file and save it as `example.html` in the same directory as `example.css`. Then enter the following HTML document:

```
<!doctype html>
<html>
  <head>
    <title>CSS Example</title>
    <link rel="stylesheet" href="example.css">
  </head>
  <body>
    <h1>CSS Example</h1>
    <p>This is an example of using CSS to style a web page.</p>
  </body>
</html>
```

Notice that we have added a `<link>` tag under the head element. The `rel` attribute (short for "relationship") has a value of "`stylesheet`", which indicates that the `href` attribute (short for "hyperlink") contains the URL of a CSS style sheet. This is how we normally associate HTML documents with style sheets.

Now, double-click `example.html` in Windows Explorer to open it in a web browser. You should see something similar to the following:

This is an example of using CSS to style a web page.

Congratulations! You have just used CSS to control the presentation of a web page.

3 Use a text editor to create a HTML document and CSS style sheet using the following directory structure and contents. Open `index.html` and add a `<link>` tag so that the web page is styled using `colours.css`. Use relative paths so the style sheet is always linked correctly even if the `colourful` directory and its contents are moved to another location.

colours.css

```
body { background: cyan; }

h1 { color: blue; }

p {
    background: yellow;
    color: red;
}
```

index.html

```
<!doctype html>
<html>
      <head>
            <title>Colourful</title>
            <link rel="stylesheet" href="styles/colours.css">
      </head>
      <body>
            <h1>Welcome</h1>
            <p>This is a colourful document!</p>
      </body>
</html>
```

Part 4: Types of Selectors

The elements which are affected by each CSS rule are determined by the selectors at the start of that rule. Each selector picks out (i.e., selects) a set of elements from the HTML document to be affected by the rule's declarations.

We shall organize selectors into four groups: element selectors, id selectors, class selectors and descendent selectors.

Element Selectors

An element selector picks out *all* elements of a particular type from the HTML document. To use an element selector, we simply enter the name of a tag or element type (e.g., `body`, `h1`, `table`, etc.). For instance, the second rule in `example.css` uses an element selector for `p` elements:

```
p { color: blue; font-style: italic; }
```

Based on this rule, all `p` elements on the web page will appear as blue italic text.

Sometimes, however, we do not want to select *all* the elements of a particular type. In such cases, we can fine-tune our selection by making use of two special attributes that are valid for all HTML tags: `id` and `class`.

id Selectors

An id selector picks out the unique element that has a particular value for its `id` attribute. To use an id selector, we enter a hex symbol (#) followed immediately by the desired element's `id` attribute value. Since `id` attributes on a web page cannot be repeated, an `id` selector will always pick out exactly *one* element if it exists.

For instance, suppose we have the following HTML and CSS files in the same folder:

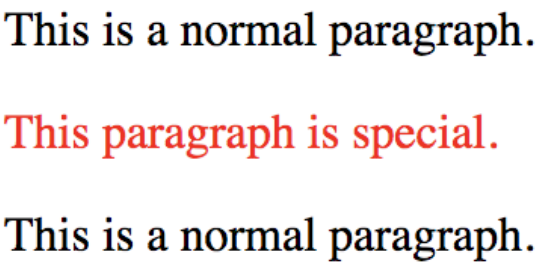
id-example.html

```
<!doctype html>
<html>
  <head>
    <title>ID Selectors Example</title>
    <link rel="stylesheet" href="id-example.css">
  </head>
  <body>
    <p>This is a normal paragraph.</p>
    <p id="special">This paragraph is special.</p>
    <p>This is a normal paragraph.</p>
  </body>
</html>
```

id-example.css

```
#special { color: red; }
```

If we open `id-example.html`, we see that only the second `p` element with an `id` of `special` is formatted as red.



class Selectors

A class selector picks out all elements that are associated with a particular class. To use a class selector, we enter a period (.) followed immediately by class name we wish to reference.

For instance, suppose we have the following HTML and CSS files in the same folder:

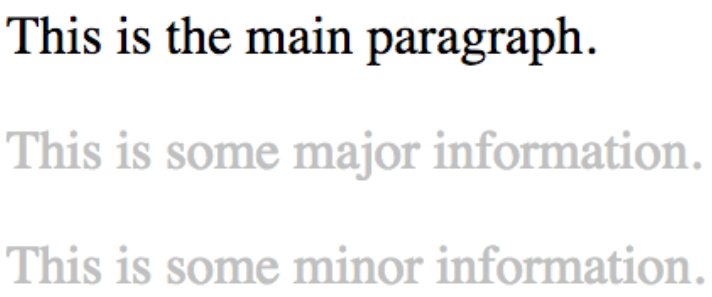
class-example.html

```
<!doctype html>
<html>
  <head>
    <title>Class Selectors Example</title>
    <link rel="stylesheet" href="class-example.css">
  </head>
  <body>
    <p>This is the main paragraph.</p>
    <p class="info major">This is some major information.</p>
    <p class="info minor">This is some minor information.</p>
  </body>
</html>
```

class-example.css

```
.info { color: silver; }
```

If we open `class-example.html`, we see that only the second and third `p` elements with a `class` of `info` are formatted as silver. These `p` elements also have additional classes (i.e., "`major`" and "`minor`" respectively), but these have no effect as they do not match any of the selectors used in the style sheet.



To summarise the use of `id` and `class` attributes:

- The `id` attribute is used to *uniquely* identify an element in a HTML document. If set, its value must be unique within the document and cannot contain any whitespace. A web page cannot have more than one element with the same `id` attribute.

- The `class` attribute, on the other hand, is used to associate an element with one or more classes (i.e., categories). If set, its value must be a space-separated list of class names. Unlike `id` attributes, multiple elements on a web page can have the same value for their `class` attributes.

Descendent Selectors

Sometimes, element selectors, id selectors and class selectors are not sufficient and it is necessary to select an element only if the element has a parent element that matches another selector. This can be achieved using the descendent selector. To use a descendent selector, separate any two selectors using a space: the corresponding rule will only be applied for elements that match the selector on the right *and* have a parent element that matches the selector on the left.

For instance, suppose we have the following HTML and CSS files in the same folder:

descendent-example.html

```
<!doctype html>
<html>
  <head>
    <title>Descendent Selectors Example</title>
    <link rel="stylesheet" href="descendent-example.css">
  </head>
  <body>
    <h1>Heading with <i>Italics</i></h1>
    <p>This paragraph has <i>italics</i>.</p>
    <i>Bare italics</i>
</body>
```

```
</html>
```

descendent-example.css

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

If we open `descendent-example.html`, we see that only the `p` element has its italics portion formatted as red. The other `i` elements remain unformatted as they do not match the specific requirements of the selector, which requires the `i` element to be a descendent of a `p` element.

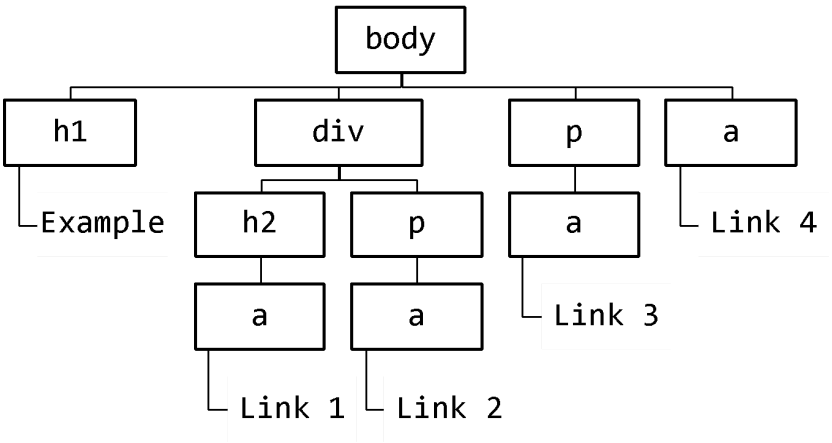
To further illustrate how the different selector types work, consider the following HTML document. In this example, we introduce a new `<div>` tag that is used as a generic element for grouping other elements. We will learn more about the `<div>` tag later in this practical task.

multiple-example.html

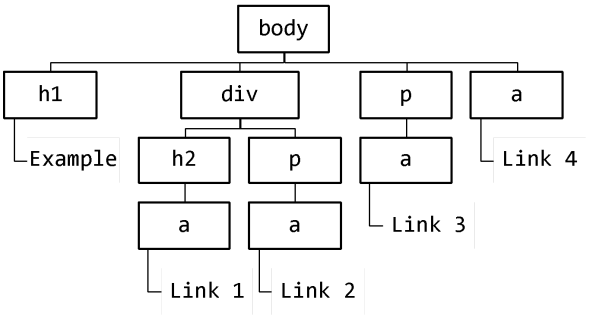
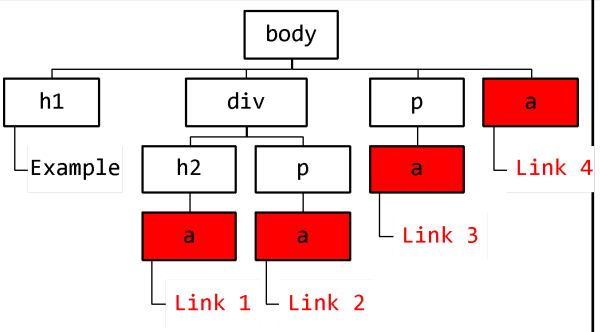
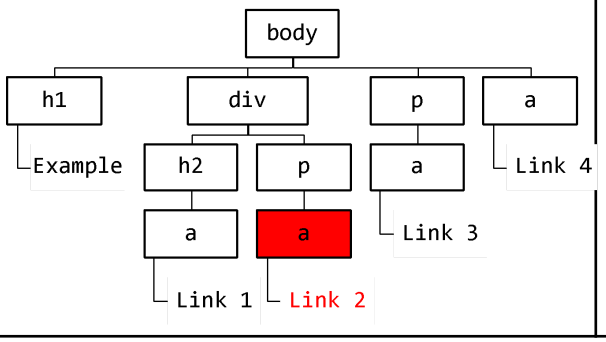
```
<!doctype html>
<html>
      <head><title>Example</title></head>
      <body>
            <h1>Example</h1>
            <div>
                  <h2><a id="link1" href="link1.html">
                        Link 1
                  </a></h2>
                  <p><a id="link2" href="link2.html">
                        Link 2
                  </a></p>
            </div>
            <p><a class="simple" href="link3.html">Link 3</a></p>
            <a class="simple" href="link4.html">Link 4</a>
      </body>
</html>
```

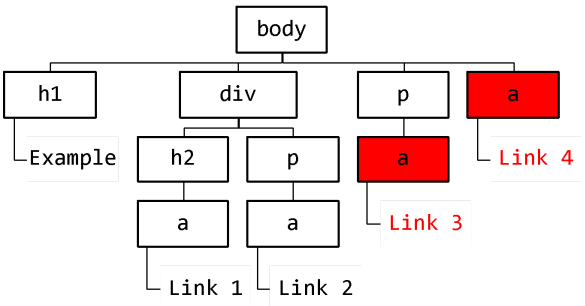
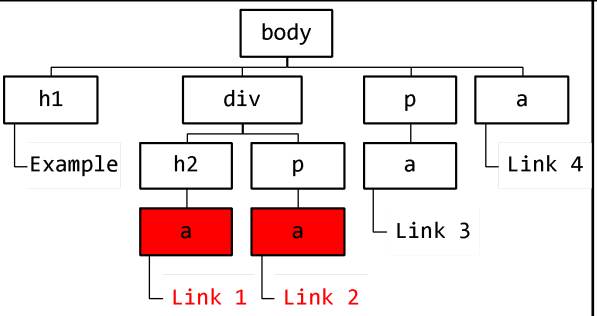
Web Applications

This HTML describes the following tree structure of elements:



Based on this HTML document, the following table shows which elements from the tree are matched for different CSS selectors:

CSS	Matched Elements	Result
No style sheet		**Example** Link 1 Link 2 Link 3 Link 4
`a { background: red; }`		**Example** Link 1 Link 2 Link 3 Link 4
`#link2 { background: red; }`		**Example** Link 1 Link 2 Link 3 Link 4

`.simple { background: red; }`		**Example** **Link 1** Link 2 Link 3 Link 4
`div a { background: red; }`		**Example** **Link 1** Link 2 Link 3 Link 4

Note that an element may be selected for more than one rule in the style sheet. If these rules have conflicting declarations, the rule with a selector that is most specific to the element has priority. The CSS language defines an algorithm for calculating the **specificity** of a selector. However, we can avoid worrying about specificity by crafting our style sheets to avoid rule conflicts in the first place.

Now that we know how CSS determines which elements are affected by each rule, let us focus on the various properties that can be set by rule declarations. We shall organize the properties into three groups: common properties, box model properties and typography properties.

4 Use a text editor to create the following HTML document. Without changing the HTML, add a CSS file such that only the text "RED" is coloured in red:

```
<!doctype html>
<html>
	<head>
		<title>Selectors Quiz</title>
		<link rel="stylesheet" href="selectors.css">
	</head>
	<body>
		BLACK
		<h1>RED</h1>
		<p>BLACK <b>BLACK</b> <i>BLACK</i></p>
		<p id="test1">BLACK <b>RED</b></p>
		<p class="test2">BLACK <i>RED</i></p>
	</body>
```

```
</html>
```

Part 5: Common Properties

A number of properties can be used with *any* element. For instance, in many of the CSS examples so far, we have been using the `background` and `color` properties to set the background and text color of elements respectively.

The following are some of the color names that can be used with the `background` and `color` properties:

red	orange	yellow	green	blue	purple
black	gray	silver	white	transparent	
				(no color)	

If a desired color does not match any of the above names, we can also specify a color in terms of its red, green and blue components. Each component is expressed as an integer between 0 to 255 (inclusive) and the color is written as `rgb(R, G, B)`, where `R` is the red component integer, `G` is the green component integer and `B` is the blue component integer. For example, the following CSS sets the page background to a shade of pale yellow:

```
body { background: rgb(255, 255, 128); }
```

The same color can be expressed as three hexadecimal numbers of 2 digits each (including a leading zero if needed). The color can thus be written as `#RRGGBB`, where `RR` is the red component in hexadecimal, `GG` is the green component in hexadecimal and `BB` is the blue component in hexadecimal. For example, the shade of pale yellow shown previously can also be written as:

```
body { background: #ffff80; }
```

Notice that `ff` is the integer `255` in hexadecimal, while `80` is the integer `128` in hexadecimal. Also note that hexadecimal digits are not case sensitive.

For convenience, if each of the three hexadecimal numbers is made of repeated digits (e.g. `00`, `11`, `22`, ..., `FF`), then the color can be shortened to `#RGB`, where `R` is the repeated hexadecimal digit for red, `G` is the repeated hexadecimal digit for green and `B` is the repeated hexadecimal digit for `blue`. For example, while `#FFFF80` cannot be shortened, the color `#00FFCC` can be shortened to `$0FC`.

Besides colors, we can also use a repeating image as the background by using `url(PATH)`, where `PATH` is the path and filename of the background image. For instance, the following CSS uses `pattern.png` (which is located in the same folder with the style sheet) as a repeating background image for the page.

```
body { background: url(pattern.png); }
```

The following HTML and pattern.png image demonstrates this:

HTML	pattern.png	Result
`<body>` `<p>Hello, World</p>` `</body>`		

Finally, all elements have a `display` property that can be used to hide the element by setting its value to `none`. The `display` property can also be set to either `block` or `inline`. These settings will be explained in the next part of this task.

Part 6: Box Model Properties

Block and Inline Appearance

You may have noticed that some HTML tags such as `<h1>` and `<p>` always start on a new line and force the following element to also start on a new line. On the other hand, tags such as `<b>` and `<i>` do not:

HTML	Result
`<h1>h1</h1>` `<p>p</p>` `<b>b</b>` `<i>i</i>`	**h1** p **b** *i*

This is because the `<h1>` and `<p>` tags have a block appearance by default while `<b>` and `<i>` have an inline appearance by default. However, we can use the display

property to override these defaults. For instance, the following CSS changes all the elements to use a block appearance instead:

CSS	Result
`h1, p, b, i { display: block; }`	**h1** p **b** *i*

Similarly, the following CSS changes all the elements to use an inline appearance:

CSS	Result
`h1, p, b, i { display: inline; }`	**h1** p **b** *i*

5 Experiment with the HTML tags you have learnt previously. For each tag in the table below, circle "Block" if it creates a block element by default and circle "Inline" if it creates an inline element by default:

Tag	Default Appearance
<h1>	Block / Inline
<h2>	Block / Inline
<h3>	Block / Inline
<p>	Block / Inline
<b>	Block / Inline
<i>	Block / Inline

Tag	Default Appearance
<a>	Block / Inline
<img>	Block / Inline
<table>	Block / Inline
<form>	Block / Inline
<input>	Block / Inline
<textarea>	Block / Inline

Web Applications

* and Tags*

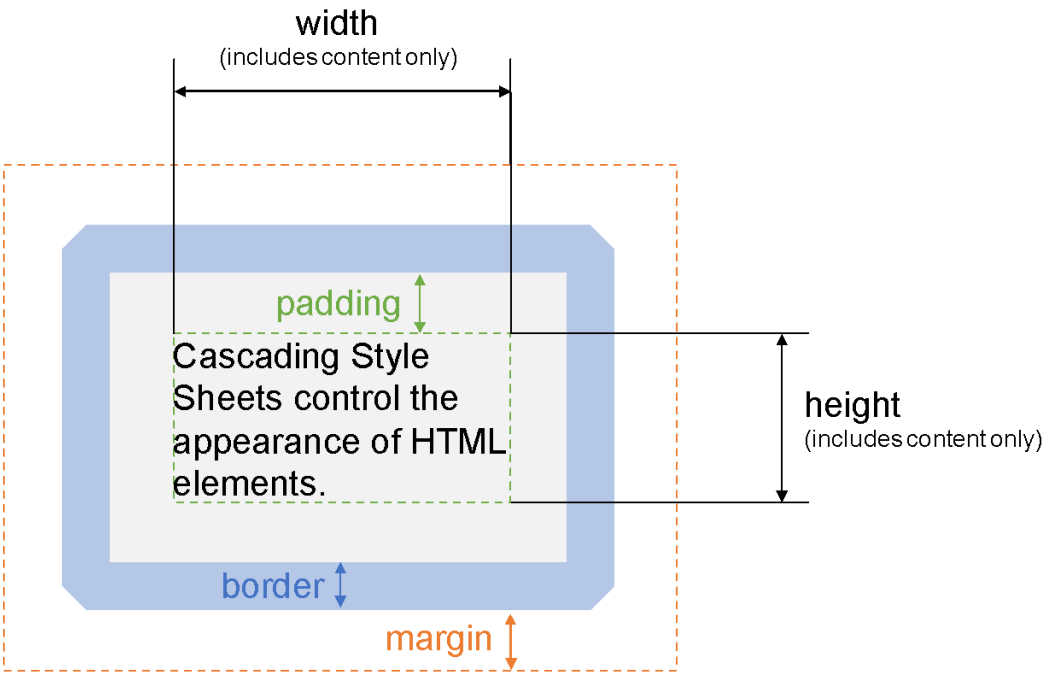
Sometimes, we may need a generic block or inline element to represent some structural feature of a document that the built-in HTML tags do not provide. The `<div>` and `<span>` tags can be used to create block and inline elements respectively for this purpose. These tags are generally used in combination with the `id` and `class` attributes to help clarify their purpose.

For instance, the following HTML document on Python syntax uses `<div>` and `<span>` tags so that Python's rules and keywords are marked out and styled:

HTML	CSS	Result
`<h1>Functions</h1>` `<div class="rule">` `Python functions` `are defined using` `the <span` `class="keyword">` `def</span> keyword.` `</div>`	`.rule {` `  background: silver;` `}` `.keyword {` `  color: red;` `}`	**Functions** Python functions are defined using the def keyword.

Margin, Border, Padding, Width and Height

Besides background and text colour, block elements have a number of additional properties that can be specified. In particular, the `margin`, `border`, `padding`, `width` and `height` properties determine how the box surrounding the element is sized and displayed. This **box model** is illustrated below:



The various box model properties are summarized in the following table:

Property Name	Description
`border`	Specifies the thickness of the optionally-coloured border around the element
`margin`	Specifies the thickness of the transparent space surrounding the border
`padding`	Specifies the thickness of the space between the content and the border that is filled with the element's background colour or pattern
`width`	Specifies the element content's width, regardless of the surrounding margin, border and padding
`height`	Specifies the element content's height, regardless of the surrounding margin, border and padding

When setting a box's width and height or the thickness of its margin, border and padding, we must specify a unit of measurement. One common unit of measurement is pixels, represented by the `px` suffix. For instance, to set the padding of `p` elements to 10 pixels, we would use the following CSS:

CSS	Result
`p { background: silver; padding: 10px; }`	This is an example of a paragraph with additional padding written using the <p> tag.

To specify that a border should be drawn with a solid colour, we use the value of a thickness, followed by a space, the word "solid", another space and finally the colour we wish to use for the border. For instance, the following CSS gives p elements a 5 pixel thick red border:

CSS	Result
`p { border: 5px solid red; }`	This is an example of a paragraph with a 5 pixel thick red border written using the <p> tag.

By default, the margin, border and padding properties control the appearance for all four sides of the element's box. However, we can append `-bottom`, `-left`, `-top` or `-right` to any of these properties so that we control the appearance for only one side of the box. For instance, the following table shows how the `border-bottom`, `border-left`, `border-top` and `border-right` properties can be used to draw a line on only one side of a p element:

CSS	Result
`p { border-bottom: 1px solid gray; }`	This is an example of a paragraph.
`p { border-left: 1px solid gray; }`	This is an example of a paragraph.
`p { border-top: 1px solid gray; }`	This is an example of a paragraph.
`p { border-right: 1px solid gray; }`	This is an example of a paragraph.

Finally, if a box's width is less than the document's width, we can center the box horizontally by setting its `margin-left` and `margin-right` properties to `auto`:

CSS	Result
`p { background: silver; margin-left: auto; margin-right: auto; width: 100px; }`	This is an example of a paragraph that is forced to be only 100 pixels wide and centered.

6 For each HTML document below, create a CSS file so that the completed web page appears as closely to the required appearance as possible.

<table>
<tr><th>HTML</th><th>Required Appearance</th></tr>
<tr><td>

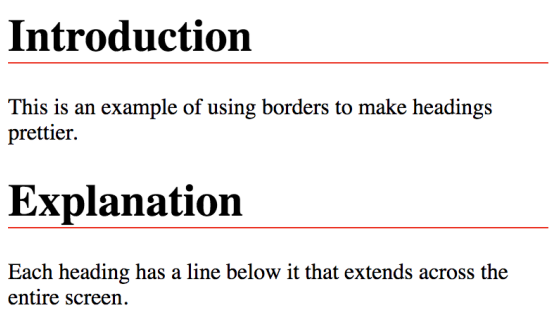
```
<h1>Introduction</h1>

<p>This is an example of
using borders to make
headings prettier.</p>

<h1>Explanation</h1>

<p>Each heading has a line
below it that extends across
the entire screen.<p>
```

</td><td>



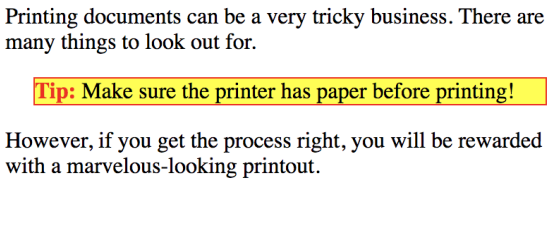
</td></tr>
<tr><td>

```
<p>Printing documents can be
a very tricky business.
There are many things to
look out for.</p>

<p class="tip"><b>Tip:</b>
Make sure the printer has
paper before printing!</p>

<p>However, if you get the
process right, you will be
rewarded with a
marvelous-looking
printout.</p>
```

</td><td>

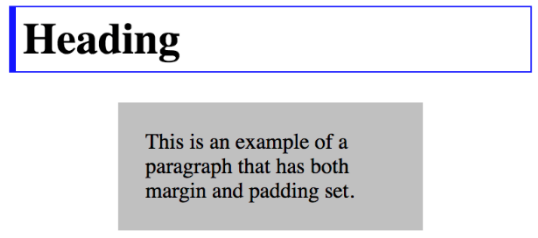


</td></tr>
<tr><td>

```
<h1>Heading</h1>

<p>This is an example of a
paragraph that has both
margin and padding set.</p>
```

</td><td>



</td></tr>
</table>

Part 7: Typography Properties

Web Applications

Besides controlling the size and spacing of box elements, you can also specify how text in each element is displayed using the `font-family`, `font-size`, `font-style`, `font-weight`, `text-align` and `text-decoration` properties.

The `font-family` property specifies which typeface is used to display the text. Its value is most often a comma-separated list of font names, ending with either `serif` or `sans-serif`. Note that if a font name has spaces, it must be enclosed in quotes (e.g., "Times New Roman"). The browser will use the first font in the list that is installed. However, if none of the named fonts are installed, it will fall back to using a generic `serif` or `sans-serif` font instead.

A serif font such as "Times New Roman" has lines extending from the ends of each letter stroke. Such fonts are traditionally used for long pieces of printed text. A sans-serif font such as "Arial", however, does not have these additional lines. The following examples illustrate how to set a serif or sans-serif font using CSS:

CSS	Result
`p { font-family: serif; }`	This is an example of a paragraph.
`p { font-family: sans-serif; }`	This is an example of a paragraph.

The `font-size` property can be used to specify text size in pixels:

CSS	Result
`p { font-size: 24px; }`	This is an example of a paragraph.

The `font-style` property specifies whether an italic font is used. The most common values for this property are `normal` and `italic`. Similarly, the `font-weight` property specifies whether a bold font is used. The most common values for this property are `normal` and `bold`:

CSS	Result
`p { font-style: italic; }`	*This is an example of a paragraph.*
`p { font-weight: bold; }`	**This is an example of a paragraph.**
`p {` `  font-style: italic;` `  font-weight: bold;` `}`	***This is an example of a paragraph.***

The `text-align` property specifies how the text is aligned in its box. The most common values for this property are `left`, `center`, `right` and `justify`:

CSS	Result
`p { text-align: left; }`	This is an example of a paragraph with enough content to see how text-align works.
`p { text-align: center; }`	This is an example of a paragraph with enough content to see how text-align works.
`p { text-align: right; }`	This is an example of a paragraph with enough content to see how text-align works.
`p { text-align: justify; }`	This is an example of a paragraph with enough content to see how text-align works.

Finally, the `text-decoration` property specifies whether additional elements of the font are displayed. The most common values of this property are `none`, `underline` and `line-through`. The `line-through` value causes text to appear cancelled or struck through using a horizontal line:

CSS	Result
`p { text-decoration: underline; }`	This is an example of a paragraph.
`p { text-decoration: line-through; }`	~~This is an example of a paragraph.~~

7 For each HTML document below, create a CSS file so that the completed web page appears as closely to the required appearance as possible. However, there is no need to match the required appearance exactly, so if a font size is needed, use an approximate value.

HTML	Required Appearance
`<h1>Reminder</h1>` `<p>This is an <span class="expanded">important`	***Reminder*** This is an important reminder to try new & different ideas.

Web Applications

`</span> reminder to try new & different ideas.</p>`	
`<h1>Heading 1</h1>` `<h2>Heading 2</h2>` `<p>Paragraph</p>`	**Heading 1** *Heading 2* Paragraph
`<p>This is a <a href="/">link</a> to the root of your web site.</p>`	This is a link to the root of your web site.