

# Introduction to HTML

Max Design - Hands-on CSS

Lesson

# THREE

Max Design - Hands-on CSS

What will we  
**cover** in this  
lesson?

# Lesson 3: Scripts & styles

LANG attribute

<meta> element

<script> element

Adding styles

STYLE attribute

<style> element

<link> element

HTML comments

Open the  
exercise folder

Open the folder called  
“start” and then open the file  
called “**lesson03.htm**” using  
some sort of HTML editor.

Step 1:  
LANG attribute

The LANG attribute can be applied to the <HTML> element to **define the primary language for the entire document.**

A screenshot of a code editor window with a light gray border and a white background. The text '<html lang="en">' is displayed in a monospaced font. The characters are color-coded: '<' is blue, 'html' is purple, 'lang=' is green, and '"en"' is red. The window has a standard macOS-style title bar with a red, yellow, and green button on the right side.

```
<html lang="en">
```



You can also specify a language for **sections of your document** using the LANG attribute.

It could be for a **block of content**... (“fr” defines the block as French content)

```
<div lang="fr">  
  ça m'est égal...  
</div>
```

Or an **inline span of content** for a short phrase (“pt” defines the content as Portuguese).

```
<p>  
  Say <span lang="pt">Olá</span> now...  
</p>
```

All language subtags are all defined in the **Language Subtag Directory**:

[http://www.iana.org/assignments/  
language-subtag-registry/language-  
subtag-registry](http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry)

# Exercise

Let's add a LANG attribute for our entire document.

```
<html>
```

```
<html lang="en">
```

Step 2:

<meta> element

The <meta> element  
**provides metadata** (or  
“information”) about the  
HTML document.



This metadata will **not be displayed in the browser**, but will be used by things like search engines.

The <meta> elements can be used to specify **a range of information** about the document including description, keywords, author, modified date and character encoding.

All HTML documents must include **character encoding**. This information is defined with a CHARSET attribute.

A screenshot of a code editor window with a light gray border and a white background. The text inside is the HTML meta charset attribute: `<meta charset="utf-8">`. The characters are color-coded: the opening tag `<` is blue, `meta` is purple, `charset=` is magenta, `"utf-8"` is green, and the closing tag `>` is blue. The window has a standard macOS-style title bar with a red, yellow, and green button on the right side.

```
<meta charset="utf-8">
```

The character encoding declaration must be within the **first 512 characters** of your document. It should appear before the <title> element.

All HTML documents should include a **description**. This describes the information inside the document to search engines.

```
<meta name="Description" content="Short  
description goes here">
```

# Exercise

Let's add some metadata...

```
<title>Lesson 3</title>
```

```
<meta charset="utf-8">  
<title>Lesson 3</title>  
<meta name="Description" content="Short  
description goes here">
```

Step 3:

`<script>` element

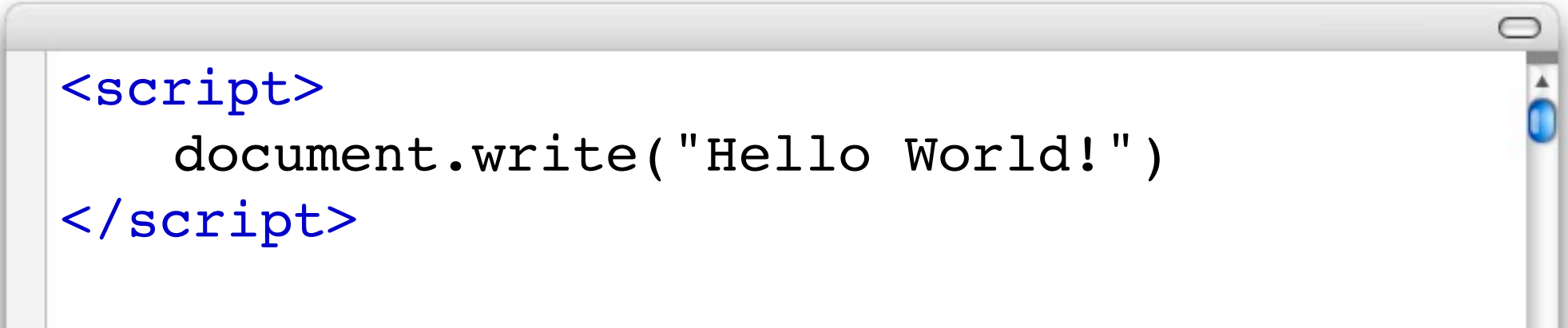


The `<script>` element is used to define a **client-side script**, such as a JavaScript.

A screenshot of a code editor window with a light gray border and a white background. The text `<script></script>` is written in blue monospace font. On the right side of the window, there is a vertical scrollbar with a blue handle and a small upward-pointing arrow at the top.

```
<script></script>
```

The `<script>` element can contain **scripting statements**.

A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The opening and closing script tags are highlighted in blue, while the JavaScript code inside is in black. The code is as follows:

```
<script>  
    document.write("Hello World!")  
</script>
```

```
<script>  
    document.write("Hello World!")  
</script>
```

Or the `<script>` element can be used to **point to** an external script file.

A screenshot of a code editor window with a light gray border and a white background. The window contains a single line of HTML code: `<script src="lesson03.js"></script>`. The text is color-coded: the opening and closing tags are blue, the attribute name 'src' is magenta, and the file path 'lesson03.js' is green. On the right side of the window, there is a vertical scrollbar with a blue handle and a small upward-pointing arrow at the top.

```
<script src="lesson03.js"></script>
```

The `<script>` element can be placed inside the **`<head>` element** or inside the **`<body>` element**.

```
<head>
  <script src="lesson03.js"></script>
</head>
<body>
  <script src="lesson03.js"></script>
</body>
```

To **make the page load faster** scripts can be moved to the bottom of the page - just above the `</body>` tag.

A code editor window with a light gray border and a white background. It contains two lines of HTML code. The first line is `<script src="lesson03.js"></script>` and the second line is `</body>`. The code is color-coded: `<` and `>` are blue, `script` is blue, `src=` is magenta, and `lesson03.js` is green. The window has a standard macOS-style title bar with a red, yellow, and green button on the right.

```
<script src="lesson03.js"></script>  
</body>
```

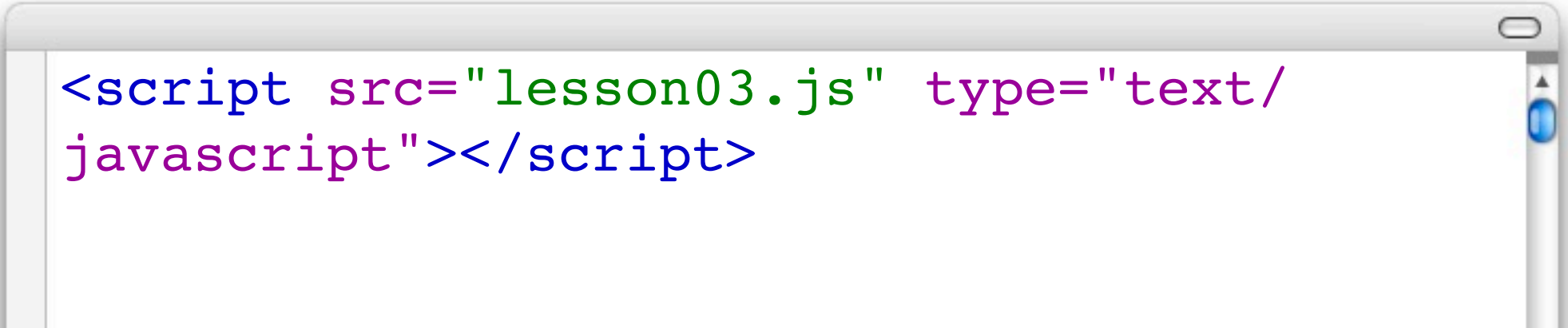
There are a **range of attributes** you can use within the `<script>` element.

The **SRC attribute** specifies the location of the script file.

A screenshot of a code editor window with a light gray border and a white background. The window contains a single line of HTML code: `<script src="lesson03.js"></script>`. The text is color-coded: the opening and closing tags are blue, the attribute name 'src' is purple, and the file path 'lesson03.js' is green. On the right side of the window, there is a vertical scrollbar with a blue handle and a small upward-pointing arrow at the top.

```
<script src="lesson03.js"></script>
```

The **TYPE attribute** specifies the MIME type of the script file. This used to be required in HTML 4.01 but is not in HTML5.

A screenshot of a code editor window with a light gray title bar and a vertical scrollbar on the right. The editor contains a single line of HTML code: `<script src="lesson03.js" type="text/javascript"></script>`. The code is color-coded: the opening and closing tags are blue, the source file name is green, and the MIME type is purple.

```
<script src="lesson03.js" type="text/javascript"></script>
```



# Exercise

Let's add an external script...

```
</body>  
</html>
```

```
<script src="lesson03.js"></script>  
</body>  
</html>
```

**Note:**  
Adding styles

If you look at our exercises in a browser, you will see that the overall page and all of the elements **appear to be “unstyled”**.

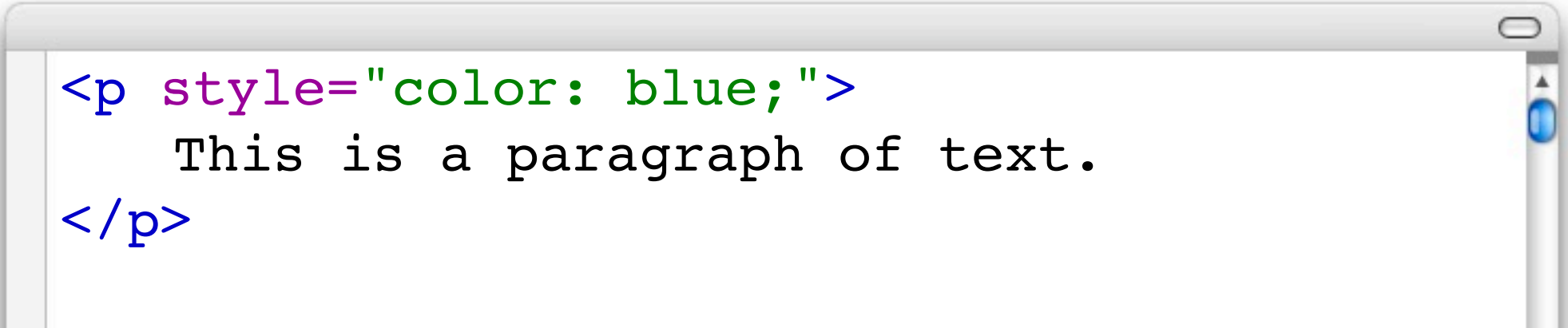
So, how do we change the appearance of the overall page and individual HTML elements? We can use **CSS (Cascading Style Sheets)**.

CSS is a language that **defines the presentation** of web pages and individual elements.

There are **three methods** that we can use to apply CSS to our web documents.

# 1. Inline styles

Apply our CSS directly to an individual HTML element via the STYLE attribute.

A code editor window with a light gray border and a white background. It contains HTML code for a paragraph with blue text. The opening tag is highlighted in purple, the style attribute in green, and the closing tag in blue. The text inside the paragraph is in a monospaced font.

```
<p style="color: blue;">  
    This is a paragraph of text.  
</p>
```



## 2. Header styles

Apply our CSS in the head of the document via the `<style>` element. This allows us to style the entire document.

```
<style>  
  div { color: red; }  
</style>
```

### 3. External style sheets

Apply our CSS in the head of the document via the `<link>` element. This allows us to style numerous documents.

```
<link rel="stylesheet" href="lesson03.css">
```

**Inline and header styles are inefficient.** If you want to make a change, you have to make it to every web document or even multiple instances within a single document.

External style sheets allow us to make a change once and every web document that links to this file **will automatically be updated.**

For this reason, we should **avoid inline and header** styles, and use links to CSS files instead. However, we will cover all three of these techniques here.

# Step 4:

## STYLE attribute

The **STYLE attribute** is used to apply CSS styles to a specific HTML element.

```
<p style="color: blue;">  
    A paragraph with a style.  
</p>
```

**One or more** CSS rules can be added into the STYLE attribute.

```
<p style="color: blue; font-weight: bold; padding: 5px;">
```

This is a paragraph of text.

```
</p>
```



# Exercise

Let's add an inline style...

A code editor window with a light gray title bar and a vertical scrollbar on the right. The text is in a monospaced font.

```
<p>
```

```
    A paragraph with a style.
```

```
</p>
```

A code editor window with a light gray title bar and a vertical scrollbar on the right. The text is in a monospaced font, with the opening tag color-coded.

```
<p style="color: blue;">
```

```
    A paragraph with a style.
```

```
</p>
```

Step 5:

<style> element

The **<style> element** is used to define style information for a single HTML document.

A screenshot of a code editor window with a light gray border and a white background. The text '<style></style>' is written in a blue monospaced font. On the right side of the window, there is a vertical scrollbar with a blue handle and a small upward-pointing arrow at the top.

```
<style></style>
```

Cascading Style Sheet or **CSS rules** are be written inside the `<style>` element.

A code editor window with a light gray title bar and a vertical scrollbar on the right. The code is written in blue text on a white background.

```
<style>
  h2 { color: blue; }
</style>
```

The `<style>` element is placed inside the **`<head>`** element.

```
<head>
  <style>
    p { color: blue; }
  </style>
</head>
```

There are a **range of attributes** you can use within the `<style>` element.

The **TYPE** attribute specifies the MIME type of the CSS rules.

```
<style type="text/css">  
    h2 { color: blue; }  
</style>
```



The **MEDIA attribute** specifies in which devices the CSS rules will be displayed.

```
<style type="text/css" media="screen">  
    h2 { color: blue; }  
</style>
```

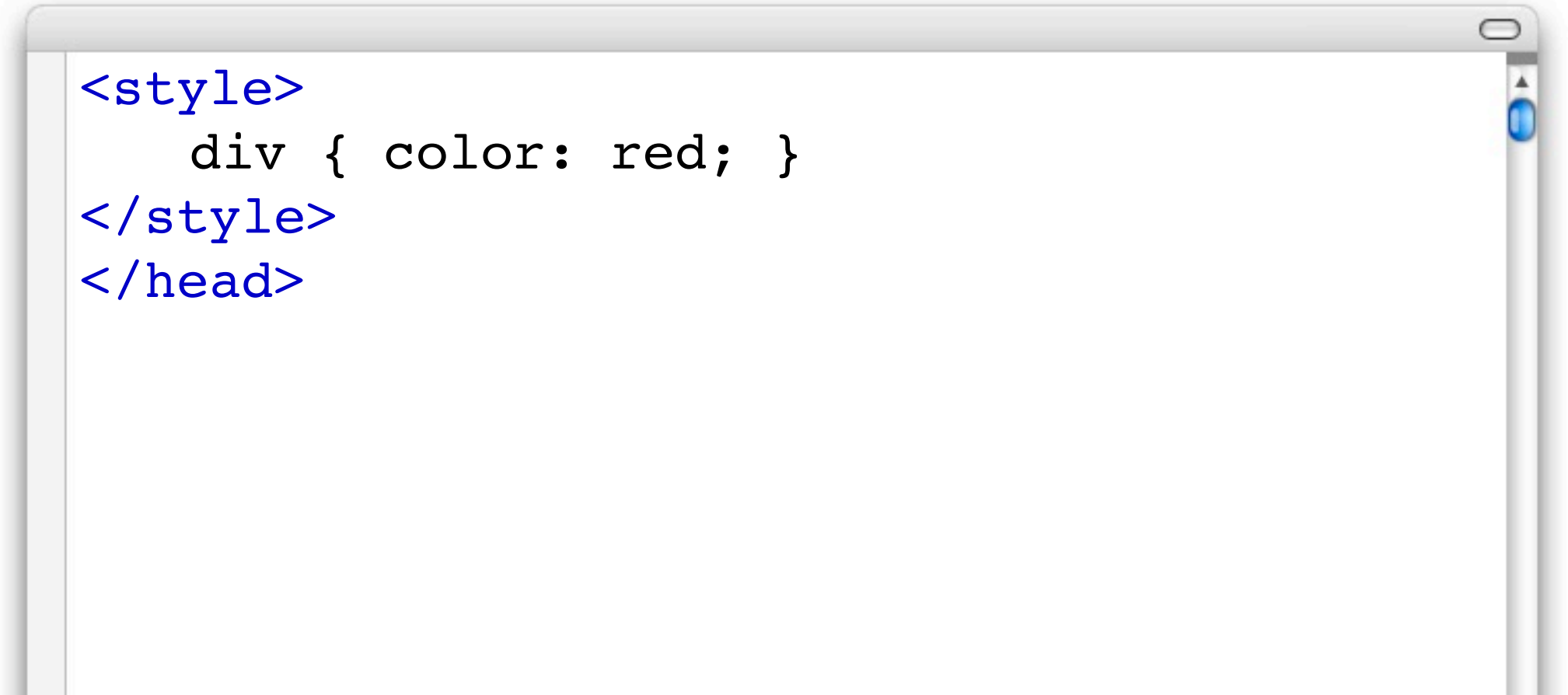
The preferred method is to **link to an external style sheet**, which means you can apply these styles to any number of pages.

# Exercise

Let's add a style element  
and a `<div>...`

A window with a light gray title bar and a vertical scroll bar on the right side. The scroll bar has a blue slider and a small upward-pointing arrow at the top.

```
</head>
```

A window with a light gray title bar and a vertical scroll bar on the right side. The scroll bar has a blue slider and a small upward-pointing arrow at the top.

```
<style>  
  div { color: red; }  
</style>  
</head>
```

Step 6:

<link> element

The `<link>` element allows us to link to **external resources**.

A screenshot of a code editor window with a light gray border and a white background. The text `<link>` is written in a blue monospaced font in the top-left corner. On the right side of the window, there is a vertical scrollbar with a blue handle and a small upward-pointing arrow above it.

```
<link>
```

The `<link>` element is most commonly used to link to **stylesheets**. These stylesheets contain Cascading Style Sheet rules (or CSS rules).

There are a **range of attributes** you can use when linking to stylesheets.

```
<link rel="stylesheet" href="sample.css"  
type="text/css" media="screen">
```



The **REL attribute** (required), specifies the relationship between the current document and the linked resource.

A screenshot of a code editor window with a light gray border and a white background. The text is color-coded: the opening tag is blue, the attribute name is purple, the value is green, and the closing tag is blue. The window has a standard macOS-style title bar with a red, yellow, and green button on the right side.

```
<link rel="stylesheet">
```

In this case, the relationship of the resource is defined as a “**stylesheet**”.



```
<link rel="stylesheet">
```

The **HREF attribute**  
specifies the location of the  
linked resource.

A screenshot of a code editor window with a light gray border and a white background. The window contains a single line of HTML code: <link rel="stylesheet" href="sample.css">. The code is color-coded: the opening tag <link is blue, rel="stylesheet" is green, href="sample.css" is green, and the closing tag > is blue. The window has a standard macOS-style title bar with a red, yellow, and green button on the right side.

```
<link rel="stylesheet" href="sample.css">
```

The **TYPE attribute** specifies the MIME type of the linked resource. This used to be required in HTML 4.01 but is not in HTML5.

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

The **MEDIA attribute** specifies in which devices the linked resource will be displayed.

```
<link rel="stylesheet" href="sample.css"  
type="text/css" media="screen">
```

While there are ten possible media values we could use, the three most common are **“all”** (used by all devices), **“print”** (print devices only) and **“screen”** (screen devices only).

If there is no media type defined, **the default value (media="all") will used.**

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

# Exercise

Let's add a link to a stylesheet...



```
<style>
  div { color: red; }
</style>
</head>
```

```
<link rel="stylesheet" href="lesson03.css">
<style>
  div { color: red; }
</style>
</head>
```

# Step 7:

## HTML comments

You can **add comments into your markup**. These comments are not rendered by browsers.

A screenshot of a web browser window. The address bar is empty. The main content area displays the HTML comment syntax `<!-- HTML comment here -->` in red text. The browser's interface includes a title bar, a scrollbar on the right, and a blue circular button at the bottom right.

```
<!-- HTML comment here -->
```

# Exercise

Let's add a comment...

```
<script src="lesson03.js"></script>  
</body>  
</html>
```

```
<!-- HTML comment here -->  
<script src="lesson03.js"></script>  
</body>  
</html>
```



# Russ Weakley

Max Design

**Site:** [maxdesign.com.au](http://maxdesign.com.au)

**Twitter:** [twitter.com/russmaxdesign](https://twitter.com/russmaxdesign)

**Slideshare:** [slideshare.net/maxdesign](http://slideshare.net/maxdesign)

**Linkedin:** [linkedin.com/in/russweakley](https://linkedin.com/in/russweakley)