Contents

How to Use This Book

The examples in this book demonstrate HTML, CSS, and JavaScript features that are supported by leading web browsers, and screenshots illustrate the actual results produced by the listed code examples. Colorization conventions are used to clarify the code listed in the steps...

HTML tags and punctuation are **Blue**, attribute values are **Orange**, and literal text is **Black**:

```
<p class="frame">HTML, CSS & JavaScript in easy steps</p>
```

CSS selectors, properties, and punctuation are **Blue**, attributes are **Orange**, specified values are **Red**:

```
p.frame { color : White ; background : Green ; }
```

JavaScript keywords and punctuation are **Blue**, specified names are **Red**, and literal values are **Black:**

```
let greeting = 'Hello World!' ;
```

All comments are colored **green**:

```
<!-- HTML Comments -->  /* CSS & JS Comments */
```

Additionally, in order to identify each source code file described in the steps, a file icon and file name appears in the margin alongside the steps:

The source code of HTML documents used in the book's examples is not listed in full to avoid unnecessary repetition – only the relevant code is listed for each example. You can download a single ZIP archive file containing all the example files by following these easy steps:

1. Browse to **www.ineasysteps.com** then navigate to Free Resources and choose the Downloads section
2. Next, find HTML, CSS, & JavaScript in easy steps in the list, then click on the hyperlink entitled All Code Examples to download the ZIP archive file
3. Now, extract the archive contents to any convenient location on your computer

If you don't achieve the result illustrated in any example, simply compare your code to that in the original example files you have downloaded to discover where you went wrong.

1 Get Started in HTML

This chapter is an introduction to the exciting world of HTML. It demonstrates how to create a valid HTML document and how to include style rules, script code, and linked resources.

Meet HTML

Historically, the desire to have text printed in specific formats meant that original manuscripts were "marked up" with annotation to indicate to the book printer how the author would like sections of text laid out. This annotation had to be concise and needed to be easily understood both by the printer and the author. A series of commonly-recognized abbreviations therefore formed the basis of a standard markup language.

HyperText Markup Language (HTML) is a modern standard markup language that uses common abbreviations called "tags" to indicate to the web browser how the author would like to have sections of a web page laid out. It was first devised in 1989 by British physicist Tim Berners-Lee at CERN in Switzerland (the European organization for nuclear research) to share all computer-stored information between the CERN physicists. Berners-Lee created a text browser to transfer information over the internet using hypertext to provide point-and-click navigation. In May 1990 this system was named the World Wide Web and was enhanced in 1993 when college student Marc Andreessen added an image tag. Now that HTML could display both text and images, the World Wide Web quickly became hugely popular.

You can find the HTML Living Standard specification, and other related specifications, online at **whatwg.org**

As various web browsers were developed, their makers began to add individual proprietary tags – effectively creating their own versions of HTML! The World Wide Web Consortium (W3C) standards organization recognized the danger that HTML could become fragmented, so they created a standard specification to which all web browsers should adhere. This successfully encouraged the browser makers to support the standard tags. The final W3C standard specification of HTML5 is now continued by the Web Hypertext Application Technology Working Group (WHATWG) as the "HTML Living Standard".

The World Wide Web comprises a series of large-capacity computers, known as "web servers", which are connected to the internet via telephone lines and satellites. The web servers each use the HyperText Transfer Protocol (HTTP) as a common communication standard to allow any computer connected to any web server to access files across the web.

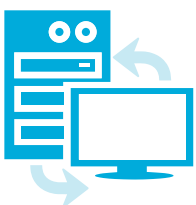
HTML web pages are merely plain text files that have been saved with a ".htm" or ".html" file extension, such as **index.html**

In order to access an HTML file across the internet, its web address must be entered into the address field of the web browser. The web address is formally known as its "Uniform Resource Locator" (URL), and typically has three parts:

- **Protocol** – any URL using the HTTP protocol begins by specifying the protocol as **http://** or secure **https://**
- **Domain** – the host name of the computer from which the file can be downloaded. For instance: **www.example.com**
- **Path** – the file name prefixed by any parent directory names where applicable. For instance: **/htdocs/index.html**

A URL describing the location of a file by protocol, domain, and path is stating its full "absolute address". Files resident within the same domain can be referenced more simply by their "relative address", which means that files located in the same directory can be referenced just by their file name. Additionally, a relative address can reference a file in its parent directory by prefixing its name with "../". For instance, a file named "higher.html" in the parent directory can be referenced as **../higher.html**

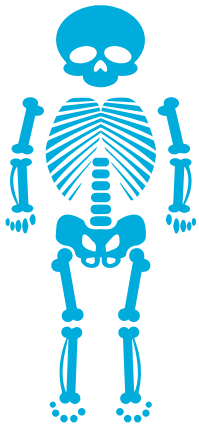
How do web servers work?



When you enter a URL into the browser address field, the browser first examines the protocol. Where the protocol is specified as HTTP, or assumed to be HTTP if unspecified, the browser recognizes that a file is being sought from a web server. It then contacts a Domain Name Server (DNS) to look up the numerical Internet Protocol (IP) address of the specified domain name. Next, a connection is established with the web server at that IP address to request the file at the specified path. When the file is successfully located, it is copied back to the browser, otherwise the web server sends an error code, such as "404 – Page Not Found".

A successful response sends HTTP headers to the web browser, describing the nature of the response, along with a copy of the requested file. The HTTP headers are not normally visible but can be examined using various development tools, such as the F12 Developer Tools feature in the Google Chrome web browser.

Understand Structure



The skeletal structure of an HTML document has three parts:

- **Document type declaration** – declaring precisely which version of HTML is used to mark up the document.
- **Head section** – providing descriptive data about the document itself, such as the document's title and the character set used.
- **Body section** – containing the content that is to appear when the document gets loaded into a web browser.

Document type declaration

The document type declaration must appear at the start of the first line of every HTML document to ensure the web browser will "render" (display) the document in "Standards Mode" – following the HTML specifications. The document type declaration tag for all HTML documents looks like this:

```
<!DOCTYPE HTML>
```

It is important to note that HTML is not a case-sensitive language – so the document type declaration tag, and all other tags, may alternatively be written in any combination of uppercase and lowercase characters. For example, the following are all valid:

```
<!DOCTYPE html>
```

```
<!Doctype Html>
```

```
<!doctype html>
```

The choice of capitalization is yours, but it is recommended you adhere consistently to whichever style you choose. The document type declaration tag capitalization style favored throughout this book uses all uppercase to emphasize its prominence as the very first tag on each page – but all other tags are in all lowercase.

Those familiar with earlier versions of HTML may be surprised at the simplicity of the HTML document type declaration. In fact, the document type declaration in earlier versions was not actually part of the HTML language – so required lengthy references to schema documents. By contrast, the modern HTML document type declaration is an intrinsic part of HTML itself.

The document type declaration in earlier versions of HTML was part of the Standard Generalized Markup Language (SGML) from which HTML is derived.

The entire document head section and body section can be enclosed within a pair of **<html> </html>** tags to contain the rest of the document. The HTML specification actually states that these are optional, but it is logical to provide a single "root" element. Most HTML tags are used in pairs like this to act as "containers" with the syntax ***< tagname > data </ tagname >***

An HTML "element" is any matching pair of opening and closing tags, or any single tag not requiring a closing tag – as described in the HTML element tags list on the inside front cover of this book.

Head section

The document's head section begins with an HTML opening **<head>** tag and ends with a corresponding closing **</head>** tag. Data describing the document can be added later between these two tags to complete the HTML document's head section.

Body section

The document's body section begins with an HTML opening **<body>** tag and ends with a corresponding closing **</body>** tag. Data content to appear in the browser can be added later between these two tags to complete the HTML document's body section.

Code comments

Comments can be added at any point within both the head and body sections between a pair of **<!--** and **-->** tags. Anything that appears between the comment tags is ignored by the browser.

Fundamental structure

So, the markup tags that create the fundamental structure of every HTML document look like this:

```
<!DOCTYPE HTML>
<html>
  <head>
    <!-- Data describing the document to be added here. -->
  </head>

  <body>
    <!-- Content to appear in the browser to be added here. -->
  </body>

</html>
```

The "invisible" characters that represent tabs, newlines, carriage returns, and spaces are collectively known as "whitespace". They may optionally be used to inset the tags for clarity.

Create Documents

The fundamental HTML document structure described on page 14, can be used to create a simple HTML document in any plain text editor – such as Windows' Notepad application. In order to create a valid "barebones" HTML document, information must first be added defining the document's primary written language, its character encoding format, and its title.

The document's primary language is defined by assigning a standard language code to a **lang** "attribute" within the opening **<html>** root tag. For the English language the code is **en**, so the complete opening root element looks like this: **<html lang="en">**

The document's character encoding format is defined by assigning a standard character-set code to a **charset** attribute within a **<meta>** tag placed in the document's head section. The recommended encoding is the popular 8-bit Unicode Transformation Format for which the code is **UTF-8**, so the complete element looks like this: **<meta charset="UTF-8">**

Finally, the document's title is defined by text between a pair of **<title> </title>** tags placed in the document's head section.

Follow these steps to create a valid barebones HTML document:

1. Launch your favorite plain text editor then start a new document with the HTML document type declaration
```
<!DOCTYPE HTML>
```

2. Below the document type declaration, add a root element that defines the document's primary language as English
```
<html lang="en">
  <!-- Head and Body sections to be added here. -->
</html>
```

3. Within the root element, insert a document head section
```
<head>
  <!-- Descriptive information to be added here. -->
</head>
```

4. Within the head section, insert an element defining the document's encoding character set
```
<meta charset="UTF-8">
```

HTML documents should not be created in word processors such as MS Word, as those apps include additional information in their file formats.

hello.html

The **<meta>** tag is a single tag – it does not have a matching closing tag. See the element tags list on the inside front cover of this book to find other single tags.

5 Next, within the head section, insert an element defining the document's title

```
<title>Get Started in HTML</title>
```

6 After the head section, insert a document body section

```
<body>
  <!-- Actual document content to be added here. -->
</body>
```

7 Within the body section, insert a size-one large heading

```
<h1>Hello World!</h1>
```

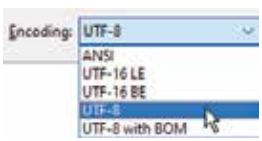
hello.html - Notepad

File Edit Format View Help

```
<!DOCTYPE HTML>

<html lang="en">

<head>
<meta charset="UTF-8">
<title>Get Started in HTML</title>
</head>

<body>
<h1>Hello World!</h1>
</body>

</html>
```

The quotation marks around an attribute value are usually optional but are required for multiple values. For consistency, attribute values in the examples throughout this book are all surrounded by quotation marks.

8 Set the file's encoding to the UTF-8 format, then save the document as "hello.html"



9 Now, open the HTML document in a modern web browser to see the title displayed on the title bar or tab, and the document content displayed as a large heading

Get Started in HTML

hello.html

Hello World!

You will discover more about headings on pages 34-35.

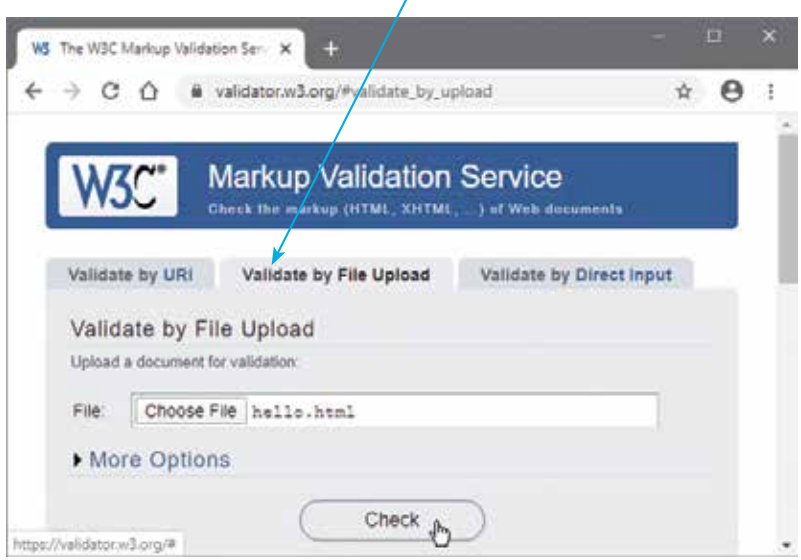
Validate Documents

Just as text documents may contain spelling and grammar errors, HTML documents may contain various errors that prevent them from conforming to the specification rules. In order to verify that an HTML document does indeed conform to the rules of its specified document type declaration, it can be tested by a validator tool. Only HTML documents that pass the validation test successfully are sure to be valid documents.

Web browsers make no attempt at validation so it is well worth verifying every HTML document with a validator tool before it is published, even when the content looks fine in your web browser. When the browser encounters HTML errors it will make a guess at what is intended – but different browsers can make different interpretations so may display the document incorrectly. Conversely, valid HTML documents should always appear correctly in any standards-compliant browser.

The World Wide Web Consortium (W3C) provides a free online validator tool at **validator.w3.org** that you can use to check the syntax of your web documents:

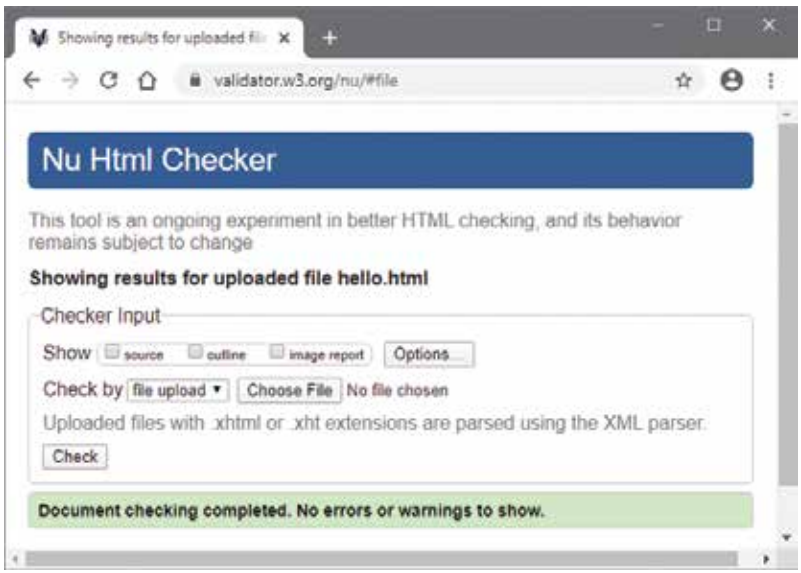
1. With an internet connection, open your web browser and navigate to the W3C Validator Tool at **validator.w3.org** then click on the **Validate by File Upload** tab



Other tabs in the validator allow you to enter the web address of an HTML document located on a web server to "Validate by URI" or copy and paste all code from a document to "Validate by Direct Input".

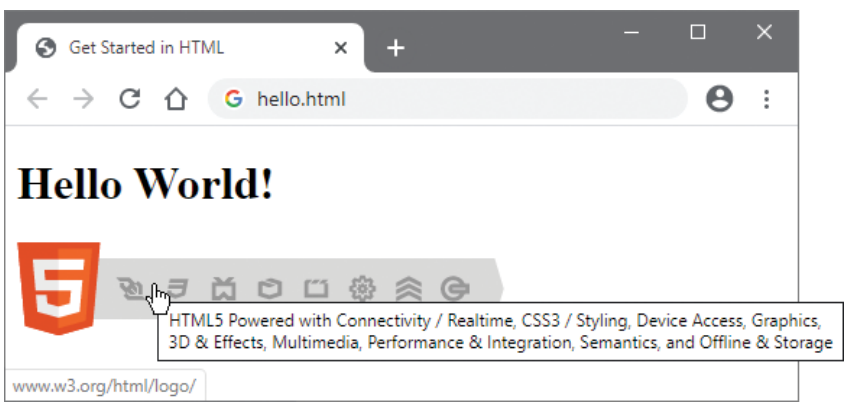
2. Click the **Browse** button then navigate to the HTML document you wish to validate – once selected, its local path appears in the validator's "File" field

3. Next, click the validator's **Check** button to upload a copy of the HTML document and run the validation test – the results will then be displayed



The validator automatically detects the document's character set and HTML version.

If validation fails, the errors are listed so you may easily correct them. When validation succeeds, you may choose to include a suitable logo at the end of the document to prove validation:



The validation logo can be customized to describe the technology classes used by the web page. Discover the logo Badge Builder online at **w3.org/html/logo** where you can generate the code to paste into your HTML document and so display a suitable logo.

Bestow Titles

The specifications require every HTML document to have a title, but its importance is often overlooked. The document title should be carefully considered, however, as it is used extensively:

- **Bookmarks** – save the document title to link back to its URL.
- **Title Bar** – a web browser window may display the title.
- **Navigation Tab** – a web browser tab may display the title.
- **History** – saves the document title to link back to its URL.
- **Search Engines** – read the document title and typically display it in search results to link back to its URL.

Document titles should ideally be short and meaningful – each tab on a modern tabbed browser may display only 10 characters.

Document titles throughout a website should follow a consistent naming convention and capitalize all major words. One popular naming convention provides a personal or company name and brief page description separated by a hyphen. For example, “Amazon - C Programming in easy steps”. An alternative puts the description first, so it remains visible when the title is truncated. For example, “C Programming in easy steps - Amazon”.

The specifications do not define a naming scheme for document titles but do encourage authors to consider accessibility issues in all aspects of their web page designs.

Document titles and document content may contain special characters that are known in HTML as “entities”. Each entity reference begins with an ampersand and ends with a semicolon. For example, the entity **<** (less than) creates a “<” character and the entity **>** (greater than) creates a “>” character. These are often needed to avoid confusion with the angled brackets that surround each HTML tag. Other frequently used entities include ** ** (a single non-breaking space), **•** (bullet point), **©** (©), **®** (®), **™** (™), and **"** (quotation mark). These are best avoided in document titles, however, as the vocal narrator used by visually impaired viewers may read each entity character as a word.

You can find a chart of all character entities at dev.w3.org/html5/html-author/charref

1. Start a new HTML document with a type declaration

```
<!DOCTYPE HTML>
```

2. Add a root element containing head and body sections

```
<html lang="en">

<head>
<!-- Data describing the document to be added here. -->
</head>

<body>
<!-- Content to appear in the browser to be added here. -->
</body>

</html>
```

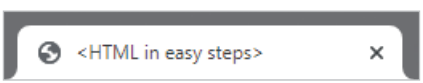
3. Within the head section, insert a meta element specifying the character set and an empty title element

```
<meta charset="UTF-8">
<title> </title>
```

4. Within the title element insert a title including entities

```
&lt;HTML in easy steps&gt;
```

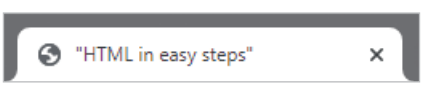
5. Save the document then open it in your web browser



6. Start a vocal narrator to hear that the title may be read out as "Less-than-HTML-in-easy-steps-greater-than"

7. Edit the document title to make it more user-friendly

```
&quot;HTML in easy steps&quot;
```



8. Save the document once more then open it in your web browser to hear the narrator now read the document title as "HTML in easy steps"

title.html

The character set can be defined in uppercase, as shown here, or in lowercase as "utf-8".

In Windows 10, press **WinKey** + **Ctrl** + **Enter** to launch the narrator, then click the tab to hear the title. Title text that is not visible on the tab will still be read by the narrator. Windows 10 ignores angled brackets in a title, but they are read literally by the narrator in earlier versions of Windows.

Supply Metadata

Meta information is simply data that describes other data. In the context of HTML, document metadata describes the document itself – rather than the document's contents.

HTML metadata is defined in the head section of the HTML document using the **<meta>** tag. The **<meta>** tag is an "empty" tag that needs no matching closing tag to create an HTML element – it is only used to specify information with its tag attributes. Previous examples have used this tag to specify the document's character-set. Further **<meta>** tags can be added to describe other aspects of the document.

Given the number of handheld devices that may view a web page, it is useful to optimize the page for smaller screens by including this **<meta>** tag in all your HTML documents' head sections:

```
<meta  name="viewport"
       content="width=device-width, initial-scale=1">
```

This will ensure your document will fill the device screen width and sets the initial zoom level so the content is not zoomed.

Setting the **width** to the **device-width** typically sets the **initial-scale** to **1** automatically, but it doesn't hurt to set it explicitly as meta data.

A **<meta>** tag can also assign a document HTTP header property to an **http-equiv** attribute and can specify that property's value to a **content** attribute. You can assign the HTTP "refresh" property to an **http-equiv** attribute to reload the page after a number of seconds specified to its **content** attribute – for example, to reload the page after five seconds, like this:

```
<meta http-equiv="refresh" content= "5">
```

This technique is often used on websites to dynamically update news or status items, as it does not depend on JavaScript support.

Another popular use redirects the browser to a new web page after a specified number of seconds, like this:

```
<meta http-equiv="refresh" content= "5 ; url='new-page.html' ">
```

In this case, the **<meta>** tag's **content** attribute specifies both the number of seconds to delay and the new URL to load.

1 Create a barebones HTML document

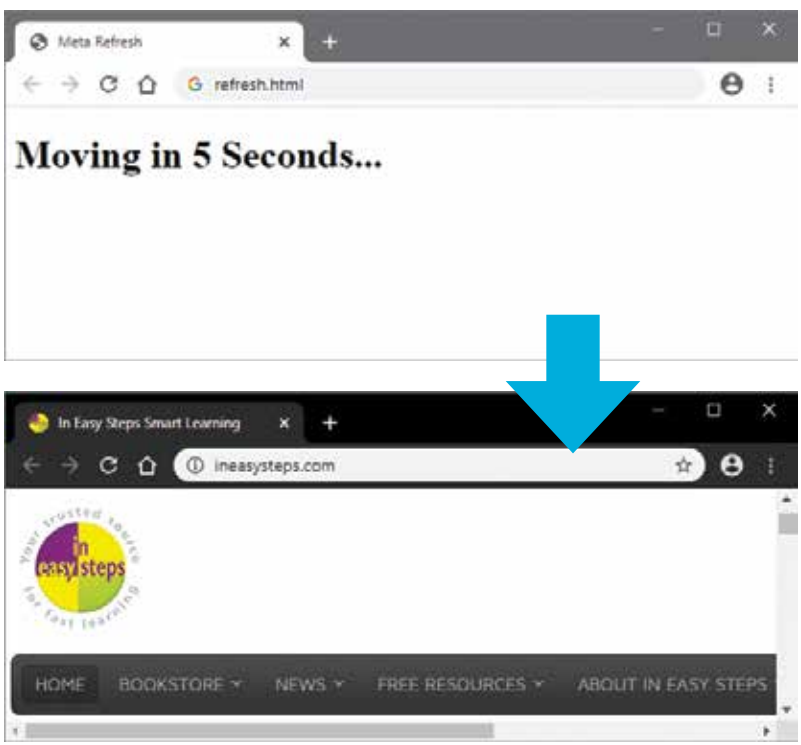
```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<!-- More metadata to be inserted here. -->
<title>Meta Refresh</title>
</head>
<body>
<h1>Moving in 5 Seconds...</h1>
</body>
</html>
```

refresh.html

2 Insert two more elements of metadata

```
<meta  name="viewport"
       content="width=device-width, initial-scale=1">
<meta http-equiv="refresh"
       content= "5 ; url='https://ineasysteps.com' ">
```

3 Save the document then open it in your web browser and wait five seconds to see the browser redirect



When you only specify the domain to the **url** attribute, as in this case, the browser will automatically load the **index.html** page at that domain location.

Describe Contents

In addition to specifying the document's character-set and expiry date, **<meta>** tags can be used to provide information that may be used by search engines. This offers no guarantee of high ranking, however, as search engines also use other page information for that purpose – especially the document title. Typically, a Search Engine Results Page (SERP) will show the meta description in search results below the page title.

Search Engine Optimization (SEO) is highly prized to ensure a web page will appear at the top of a SERP to increase traffic to a website. Unfortunately, there is no sure-fire technique to achieve this as the search engines constantly change the algorithm by which pages are ranked. It is, however, useful to provide metadata that describes the page content.

All search engines find pages to add to their index – even if the page has never been submitted to them.

Descriptive **<meta>** tags have a **name** attribute that is assigned a "description" value, and a **content** attribute that is assigned a description of the page contents.

The description should be between 50-160 characters long, as lengthy descriptions may be truncated. The description should include keywords relative to the text content. For example, a search for "italian ceramics" could return all web pages with "italian" and "ceramics" in their description.

Always include the three most important keywords in the description.

The description serves as advertising copy so a readable, compelling description using important keywords will encourage visits to the page from a SERP. You should not repeat keywords in the description, but do try to use the plural form for keywords – to match searches made with both the single and plural form of that word. Additionally, you should not include double quotation marks in the description as Google may truncate the description at a double quotation mark.

If a website contains pages of identical or very similar content, you can specify which page is to be indexed by including a "canonical link" in your HTML code to indicate the preferred source. This uses a **<link>** tag containing a **rel** (relationship) attribute to specify a "canonical" value, and an **href** (hypertext reference) attribute to specify the URL address of the preferred page.

keywords.html

1 Create a barebones HTML document

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<!-- More metadata to be inserted here. -->
<title>Tuscan Home Decor</title>
</head>
<body> <h1>Beautiful Tuscan Ceramics</h1> </body>
</html>
```

2 Insert a metadata description of the web page

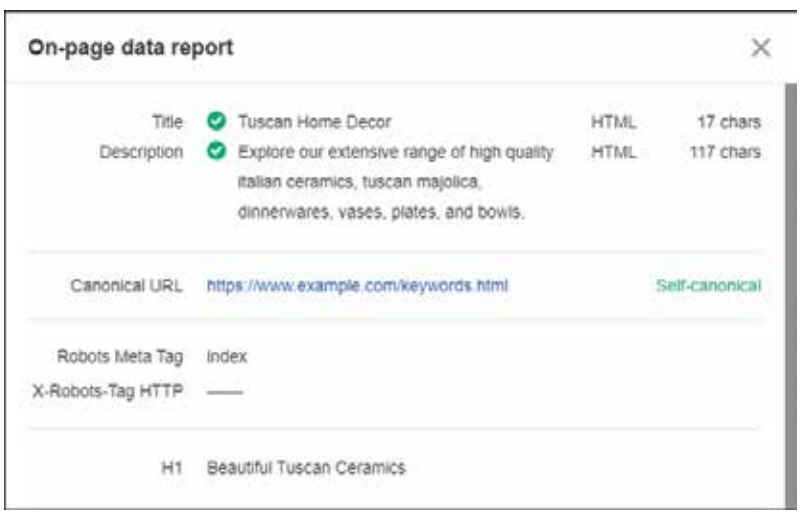
```
<meta  name="description" content="Explore our
extensive range of high quality italian ceramics including
tuscan majolica, dinnerwares, vases, plates, and bowls">
```

3 Next, in the head section, add an element to specify that this page is the preferred page for indexing purposes

```
<link rel="canonical"
        href="https://www.example.com/keywords.html" >
```

4 Save the document then visit the Chrome Web Store at **chrome.google.com/webstore/category/extensions** and search for "seo" to add a search engine analysis extension

5 Open the HTML document in the Google Chrome web browser then use the analysis tool to see the meta data



There are a number of free meta tag generators available online – enter "free meta tag generator" into a search engine.

Add Styles

Cascading Style Sheets (CSS) rules can be incorporated within HTML documents to control the presentational aspects of each element on the page. The use of style sheets has replaced all features of HTML that formerly related to presentation. For example, the **<font>** tag has become obsolete, as font family, weight, style, and size are now specified by a style sheet rule.

Style sheets embedded with **<style> </style>** tags can be added within the head section of an HTML document to enclose rules governing how the content will appear. For example, a simple style sheet containing rules to determine the appearance of all size-one headings could look like this:

```
<style>

h1 { color : red ; background : yellow ; }

</style>
```

When multiple rules select the same property of an element for styling, the rule read last by the browser will generally be applied, but in-line rules take precedence over embedded rules and external rules. Embedded rules take precedence over external rules.

This is acceptable and will validate but, in line with the aim of HTML to separate content from presentation, style sheets may be contained within a separate file. The great advantage of placing style sheets in separate files is that they can be applied to multiple HTML documents – thus making website maintenance much easier. Editing a shared style sheet instantly affects each HTML document that shares that file.

An external style sheet is incorporated within an HTML document by adding a **<link>** tag in the document's head section. This must contain a **rel** (relationship) attribute assigned a "stylesheet" value, and the URL of the style sheet must be assigned to its **href** (hypertext reference) attribute – for example, add an adjacent style sheet file named "style.css", like this:

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

You can also specify style rules "in-line" to a style attribute of presentational HTML tags, like this:

```
<h1 style="color:red">
```

In-line style rules are useful in some circumstances but can make page maintenance more difficult.

1 Create a barebones HTML document

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Style Sheet Example</title>
</head>
<body>
<h1>Styled Heading</h1>
</body>
</html>
```

style.html

2 Next, in the head section, add an embedded style sheet

```
<style>
h1 { color : Red ; background : Yellow ; }
</style>
```

3 Now, in the head section, insert a link to an adjacent external style sheet file

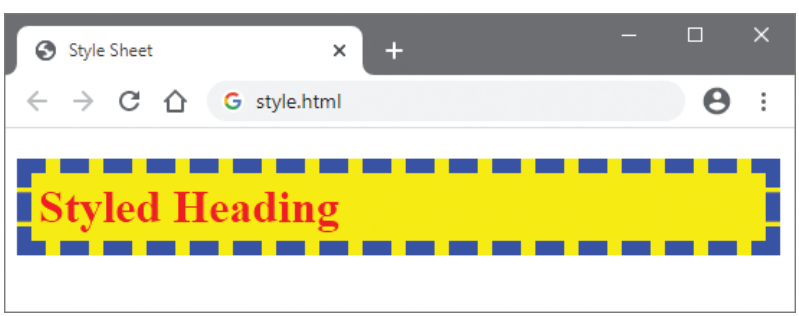
```
<link rel="stylesheet" href="style.css">
```

style.css

4 Save the HTML document then open a new text editor window and precisely copy this style sheet

```
h1
{
        border : 10px dashed Blue ;
        padding : 5px ;
        width : 500px ;
}
```

5 Save the Cascading Style Sheets file in the same directory as the HTML document, then open the web page in your browser to see the style rules applied



Some HTML elements, such as ** ** and **** (see page 46), exist purely for styling. CSS is a separate topic but many examples in this book include embedded CSS style sheet rules to provide standalone example files that demonstrate the use of HTML elements. Some examples include unlisted CSS rules to illustrate the size and position of HTML elements and their content in screenshots.

Include Scripts

Scripts can be incorporated within HTML documents to interact with the user and to provide dynamic effects. This ability has become increasingly important with the development of pages in which sections of the page can be dynamically updated. Previously, the browser would typically request an entire new page from the web server, which was less efficient and more cumbersome.

JavaScript code enclosed by **<script> </script>** tags can be embedded within an HTML document. These are best placed in the body section of the document, just before the **</body>** closing tag, so the browser can process the content of the document before reading the script.

Remember that the **<script>** tag always needs to have a matching closing tag.

In line with the aim of HTML to separate content from presentation, scripts may also be contained in a separate file. In this case, the URL address of the script file must be assigned to a **src** attribute within the **<script>** tag. The **</script>** closing tag is also required. These, too, can be placed at the end of the body section of the HTML document, as the browser will treat the external script as if it was embedded there – for example, to add an adjacent external script file named "script.js", like this:

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

You can also specify script "in-line" to on-event attributes of HTML tags. For example, to recognize a mouse click event:

```
<h1 onclick="alert('Clicked!')">
```

In-line script is useful in some circumstances but can make page maintenance more difficult. Alternative fallback content can be provided in the document's body section between **<noscript> </noscript>** tags, which will only be displayed when script functionality is absent or disabled.

script.html

1 Create a barebones HTML document

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>JavaScript Example</title>
</head>
<body>
</body>
</html>
```

2 In the body section, insert a fallback message and heading

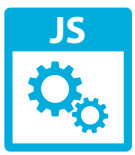
```
<noscript>JavaScript Is Not Enabled!</noscript>
<h1 onclick="this.innerText='Mouse Clicked!';
        this.style.color='Red'">Active Heading</h1>
```

3 At the end of the body section, add an embedded script and nominate an external script

```
<script>
document.getElementsByTagName('h1')[0].onmouseover =
function ( ) {
this.innerText= 'Mouse Is Over' ; this.style.color='Blue' }
</script>
<script src="script.js"></script>
```

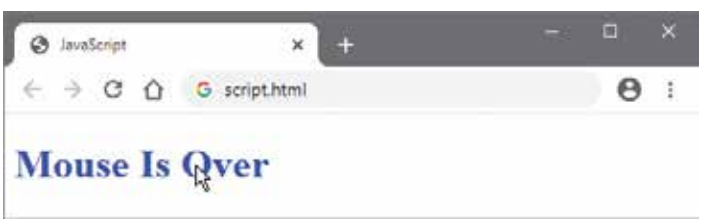
4 Save the HTML document then open a new text editor window and create the external script

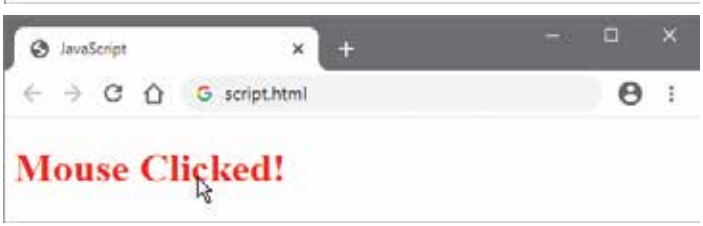
```
document.getElementsByTagName('h1')[0].onmouseout =
function ( ) {
this.innerText= 'Active Heading' ; this.style.color = 'Black' }
```



script.js

5 Save the JavaScript file as "script.js" in the same directory as the HTML document, then open the web page in your browser and click on the heading





Some HTML elements, such as **<template>** and **<slot>** (see page 120), exist purely for scripting. JavaScript is a separate topic but many examples in this book include embedded JavaScript code to provide standalone example files that demonstrate the use of HTML elements.

Link Resources

The **<link>** tag that was used in an earlier example to incorporate a style sheet in an HTML document can also be used to incorporate other resources into a document.

This tag may only appear in the head section of a document, but the head section can contain many **<link>** tags. Each **<link>** tag must contain **rel** and **href** attributes, stating the relationship and location of the link resource. It may also include a **type** attribute where appropriate, to hint at the MIME type of the link resource.

Permitted rel (relationship) values				
alternate	author	bookmark	help	icon
license	next	nofollow	noreferrer	prev
search	stylesheet	tag	shortcut icon	

MIME (Multipart Internet Mail Extension) types describe file types – such as **text/html** for HTML files. You can find the list of official MIME types at https://www.iana.org/assignments/media-types/media-types.xhtml

Many of the link types above are intended to help search engines locate resources associated with that HTML document, and the **<link>** tag may also include a **title** attribute to further describe the resource – for example, a version of the page in another language:

```
<link rel="alternate" type="text/html" href="esp.html"
        title="Esta página en Español - This page in Spanish" >
```

In this case, the location of the resource is specified using a relative address that, by default, the browser will seek in the directory in which the HTML document is located. The browser can, however, be made to seek a relative address in a different directory by inserting a **<base>** tag at the start of the document's head section. Its **href** attribute can then specify the absolute directory address – for example, to specify a separate "resources" directory, like this:

```
<base href= "http://localhost/resources/">
```

When using a **<base>** element it must be placed in the head section before any **<link>** elements.

It is popular to link an icon resource to display in the web browser. This is named exactly as "favicon.ico" and can be placed in the same directory as the HTML document, or in a directory specified by the **<base>** tag. All browsers recognize any other resources in the directory specified by the **<base>** tag.

1. Create a new HTML document that includes metadata, a linked resource, and areas for style rules and script code

```
<!DOCTYPE HTML>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta  name="viewport"
       content="width=device-width, initial-scale=1">
<link rel="shortcut icon" href="favicon.ico">
<title>Document Title</title>
<style>

</style>
</head>
<body>


<script>

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

This template is the basic HTML document that is used in all ensuing examples to create a new HTML document – only the title changes to suit each example.

2. At the beginning of the head section, insert an element to specify a base "resources" directory

```
<base href="http://localhost/resources/">
```

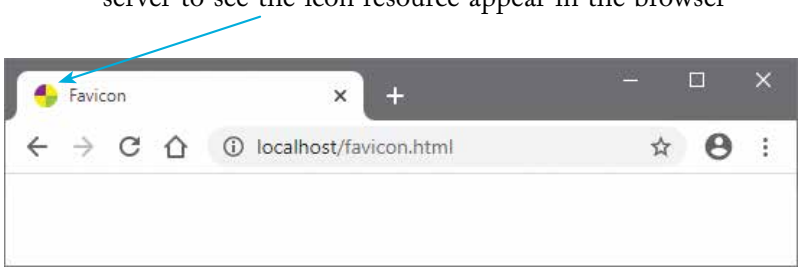
favicon.html

3. Change the document title to "Favicon", then save the HTML document

4. Open an icon editor and create an icon sized 32 x 32 pixels and save your icon alongside the HTML document, or in the "resources" directory, named as "favicon.ico"

favicon.ico 32px x 32px

5. Open the HTML document in your web browser via a web server to see the icon resource appear in the browser



You can force your browser to refresh the favicon by assigning **favicon.ico?v=2** to the link's **href** attribute.

Summary

- The Web Hypertext Application Technology Working Group (WHATWG) oversees the HTML Living Standard.
- HyperText Transfer Protocol (HTTP) is the common communication standard used by web servers.
- A Uniform Resource Locator (URL) is an absolute web address comprising protocol, domain, and path components.
- A relative address can reference an adjacent file by its name, and may use the ../ syntax to reference a parent directory.
- Web servers send response headers back to the requesting computer and a copy of the requested file, or an error code.
- Each HTML document should have a document type declaration, a head section, and a body section.
- Information about the document itself is contained in the head section, and content is contained in the body section.
- The document's written language is specified to a **lang** attribute in the opening **<html>** root element.
- The document's character-set encoding is specified to a **charset** attribute in a **<meta>** tag within the head section.
- The document's title is specified between **<title>** **</title>** tags within the head section.
- The free online W3C validator tool can be used to verify that an HTML document is free of errors.
- Metadata describes the document, and a content description can be used by search engines to index the web page.
- The **<style>** **</style>** tags can be used to embed style sheets within an HTML document.
- The **<script>** **</script>** tags can be used to include internal and external JavaScript code in an HTML document.
- The **<link>** tag can be used to embed external style sheets and other resources within an HTML document.