

Introduction to HTML

Max Design - Hands-on CSS

What will we
cover in this
course?

This course is for those who have **never done HTML markup before** and for those who **want to learn HTML markup the right way.**

This course will give you a **clear understanding** of how HTML markup works. We will cover all of the basic elements and attributes, and how they should be used.

With each lesson, we will cover some concepts in slideshow form, then do a series of **hands-on coding exercises**.

By the time you finish this course, you should be able to **write HTML markup with confidence!**

Lesson 1: Basic markup

What is HTML?

The doctype

HTML elements

<html> element

<head> element

<title> element

<body> element

Heading elements

HTML whitespace

<p> element

 element

Block vs inline elements

<div> element

 element

HTML attributes

ID attribute

CLASS attribute

Lesson 2: Content markup

 element

 element

<i> element

<a> element

File paths

 element

 element

 element

<dl> element

<abbr> element

<hr> element

<q> element

<blockquote> element

<address> element

<small> element

<sub> element

<sup> element

Lesson 3: Scripts & styles

LANG attribute

<meta> element

<script> element

Adding styles

STYLE attribute

<style> element

<link> element

HTML comments

Lesson 4: Table markup

What are tables?

<table> element

<tr> element

<td> element

<th> element

SUMMARY attribute

<caption> element

<thead> element

<tfoot> element

<tbody> element

<col> element

<colgroup> element

What is needed?

Lesson 5: Form markup

What are forms?

<form> element

<fieldset> element

<legend> element

<label> element

<input> elements

input type="text"

input type="email"

input type="url"

input type="tel"

input type="password"

input type="file"

input type="checkbox"

input type="radio"

input type="hidden"

input type="reset"

input type="image"

input type="submit"

input type="button"

<button> element

<select> & <option>

<optgroup> element

<textarea> element

Lesson 6: Valid markup

What is valid markup?

Missing doctype

Missing `` ALT attributes

ID referenced more than once

Incorrectly nested elements

Block-level elements inside inline elements

Forgetting to close elements

Escaping special characters

Exercise: finding the problems

What do I need?

The files:

Each lesson includes a “start” and “finished” folder with all the relevant files inside. We will use the “start” files for each exercise. The finished files are provided for your reference.

The tools:

Text editor: BBedit, Coda, Dreamweaver, Notepad++, Sublime text etc

Browser: Chrome, Safari, Firefox, Opera, Internet Explorer 9+, etc

Lesson ONE

Max Design - Hands-on CSS

What is HTML?

HTML (HyperText Markup Language) is a markup language that is used to **create web documents.**



```
<h1>Hello world!</p>
```

HTML markup uses **HTML elements and attributes** to describe the contents and structure of the web page.

Element

Attribute



The diagram illustrates the components of an HTML tag. Two black boxes at the top, labeled 'Element' and 'Attribute', have arrows pointing down to a code snippet. The 'Element' arrow points to the opening and closing tags '<p' and '>/p>'. The 'Attribute' arrow points to the 'class="intro"' part of the tag. The code snippet is displayed in a light gray box with a shadow, mimicking a code editor window.

```
<p class="intro">Some text</p>
```

Over the years, there have been **various versions of HTML released** by the World Wide Web Consortium (W3C).

HTML

| | |
|-----------|---------------|
| HTML 2.0 | November 1995 |
| HTML 3.2 | January 1997 |
| HTML 4.0 | December 1997 |
| HTML 4.01 | December 1999 |

XHTML

| | |
|-----------|--------------|
| XHTML 1.0 | January 2000 |
| XHTML 1.1 | May 2001 |

HTML5

| | |
|-------|--------------|
| HTML5 | January 2008 |
|-------|--------------|

Open the
exercise folder

Open the folder called
“start” and then open the file
called “**lesson01.htm**” using
some sort of HTML editor.

Step 1: the doctype

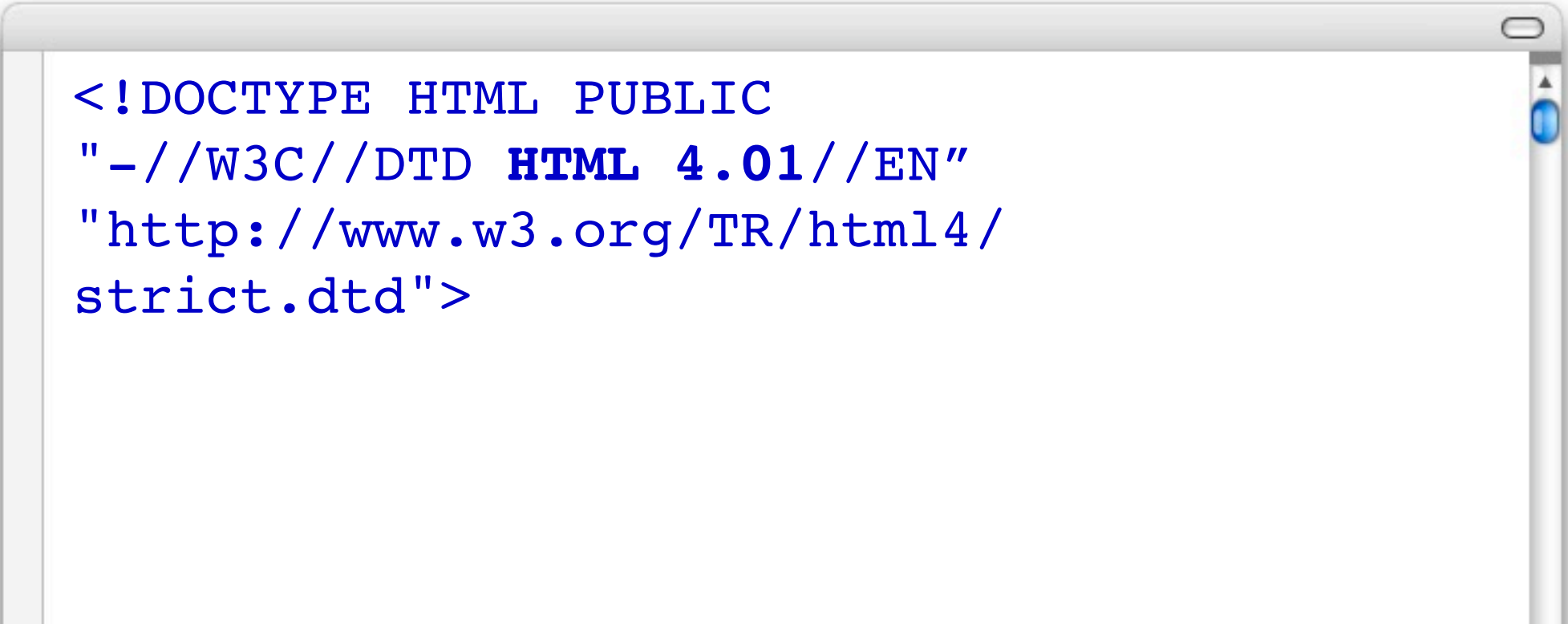
Every HTML document
should start with a
document type declaration
- or a **doctype**, as it is more
commonly known.

The doctype should appear as the **very first line** in your HTML markup.

The doctype is not an element, it is an instruction to the browser about **which version** of HTML you are using for your web page.

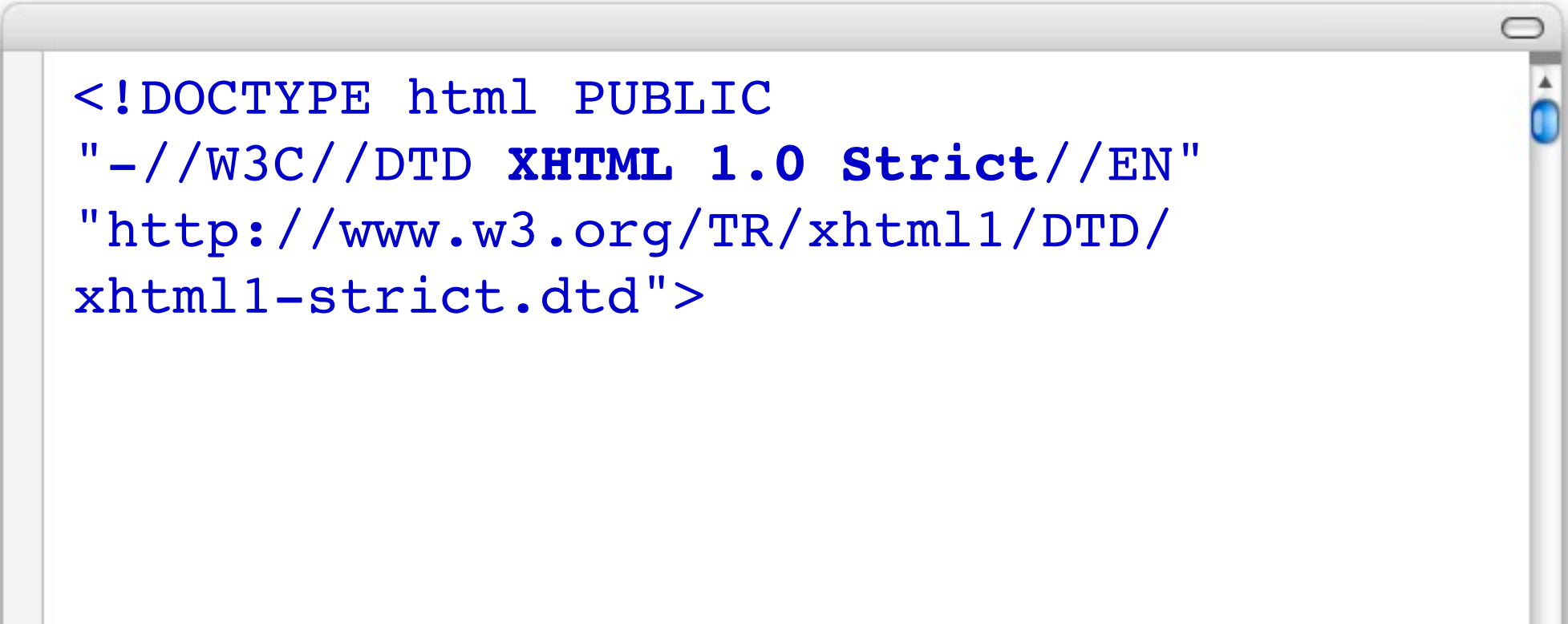
There are many different
Doctypes, but the **three**
main Doctypes are:

HTML 4.01



```
<!DOCTYPE HTML PUBLIC  
"-//W3C//DTD HTML 4.01//EN"  
"http://www.w3.org/TR/html4/  
strict.dtd">
```

XHTML 1.0



```
<!DOCTYPE html PUBLIC  
"-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/  
xhtml1-strict.dtd">
```

HTML5



```
<!DOCTYPE html>
```

As you can see, the **HTML5 doctype** is the simplest and easiest to use. We will use the HTML5 doctype and syntax for all of our exercises.

Exercise

Let's add a doctype now...

A window with a light gray title bar and a vertical scrollbar on the right side. The scrollbar has a blue slider. The main area of the window is white and contains a single line of blue text at the top left.

```
<!DOCTYPE html>
```

Note:

HTML elements

HTML elements are the **building blocks of HTML markup**. They help us define the content and structure of web pages.

HTML elements generally consist of a **start tag** (or opening tag) and an **end tag** (or closing tag).

A diagram illustrating the structure of an HTML element. It features a light gray rectangular frame with a thin border, resembling a window or a code editor. Inside the frame, the text "<p>Some content here</p>" is displayed in a monospaced font. The opening tag "<p>" is highlighted in blue, and the closing tag "</p>" is also highlighted in blue. Below the opening tag, there is a black rectangular box containing the white text "Start tag". A black arrow points upwards from this box to the opening tag. Similarly, below the closing tag, there is a black rectangular box containing the white text "End tag". A black arrow points upwards from this box to the closing tag. The overall layout is clean and focused on the HTML syntax.

```
<p>Some content here</p>
```

Start tag

End tag

Omitting the closing tag can **cause issues**, so you should always close your elements!

A screenshot of a web browser window. The address bar is empty. The main content area displays the text "<div>Some content here". The opening tag "<div>" is highlighted in blue, while the rest of the text is in black. The browser's scrollbar is visible on the right side.

```
<div>Some content here
```

Void elements

There are times when HTML elements have only one tag. **Void elements** have no content and no end tag.

A diagram of a web browser window. Inside the window, the text "
" is displayed in blue. A black arrow points upwards from a black box containing the text "No end tag" to the "
" text.

No end tag

Void elements are **written differently** depending on whether you are writing HTML 4.01, XHTML or HTML5.

For example, the `
` element is written with a trailing slash in **XHTML 1.0**



`
`

Trailing slash

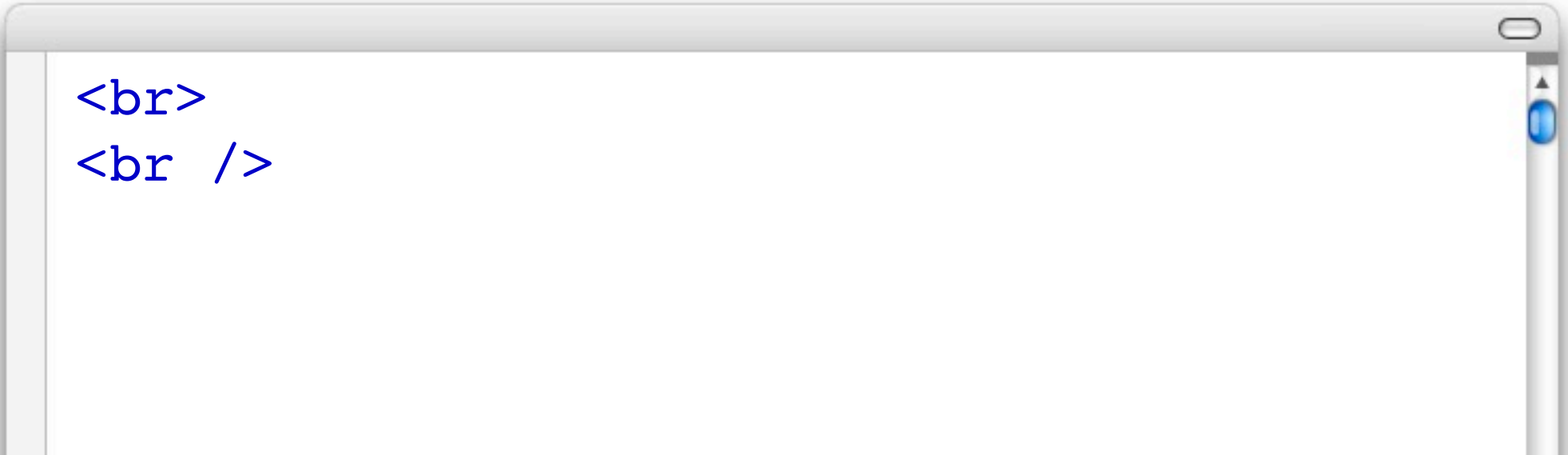
But it is written without a trailing slash in **HTML 4.01**



`
`

No trailing slash

With HTML5, you can use
**whichever of these
options** you prefer.

A browser window with a grey title bar and a vertical scrollbar on the right. The content area is white and displays two lines of blue text: "
" on the first line and "
" on the second line.

```
<br>  
<br />
```

Step 2:

<html> element

The `<html>` element is used to tell **the browser that this is an HTML document.**

This element wraps around all other elements in the HTML document.

Exercise

Add the `<html>` element...

A window with a light gray border and a white background. In the top right corner, there is a small gray circle with a white dot inside, resembling a close button. On the right side, there is a vertical scrollbar with a blue slider and a small upward-pointing arrow at the top.

```
<!DOCTYPE html>
```

```
<html>
```

```
</html>
```


Step 3:

<head> element

The <head> element is a container for a range of elements that are **not displayed in the browser.**

Exercise

Add the `<head>` element...



```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

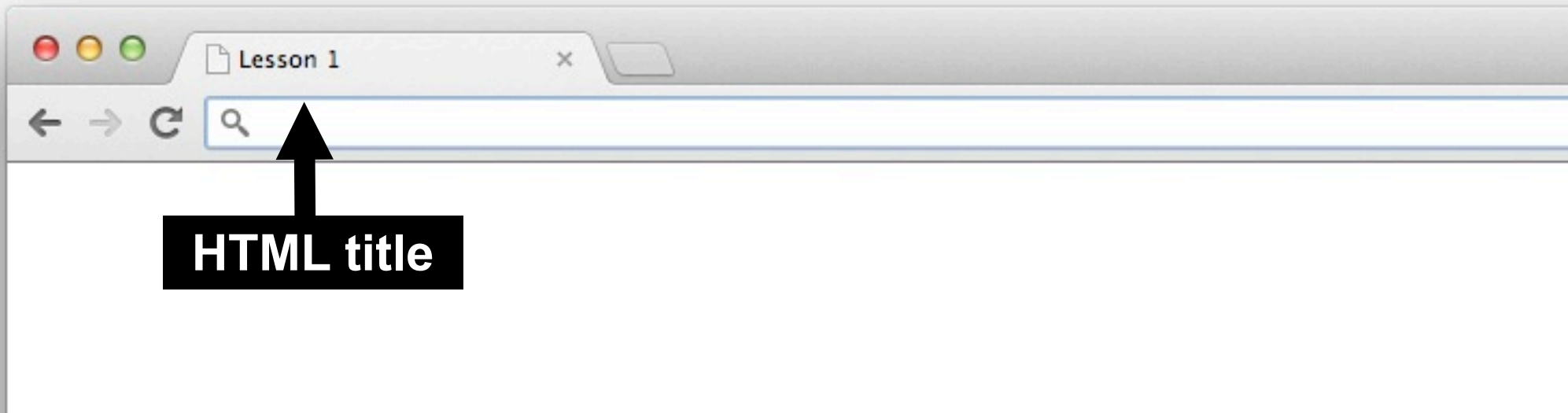
```
  </head>
```

```
</html>
```

Step 4:

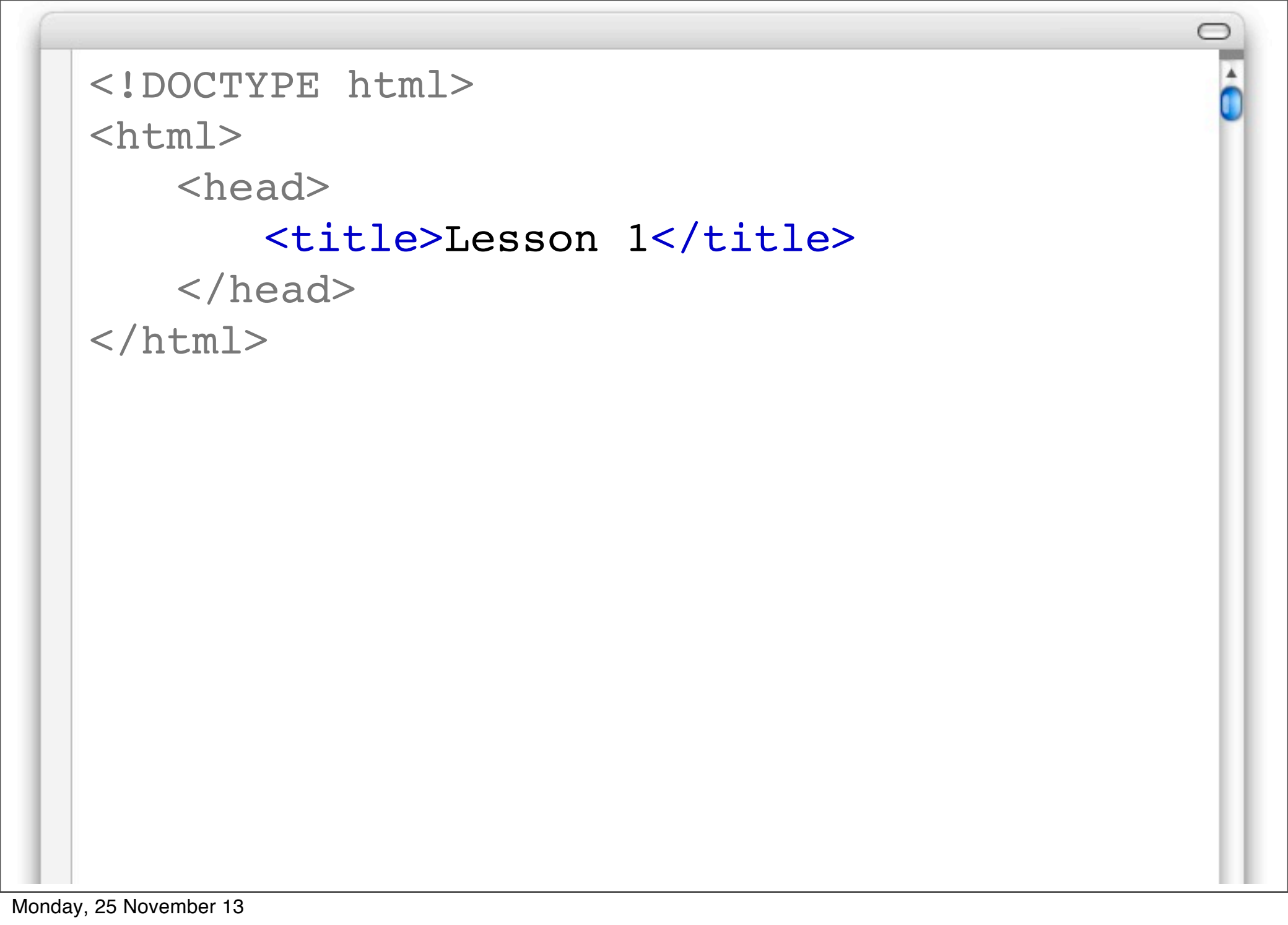
<title> element

The `<title>` element is not displayed on the page, but will be seen in the **browser title strip** (or browser tab).



Exercise

Add the `<title>` element...



```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
</html>
```

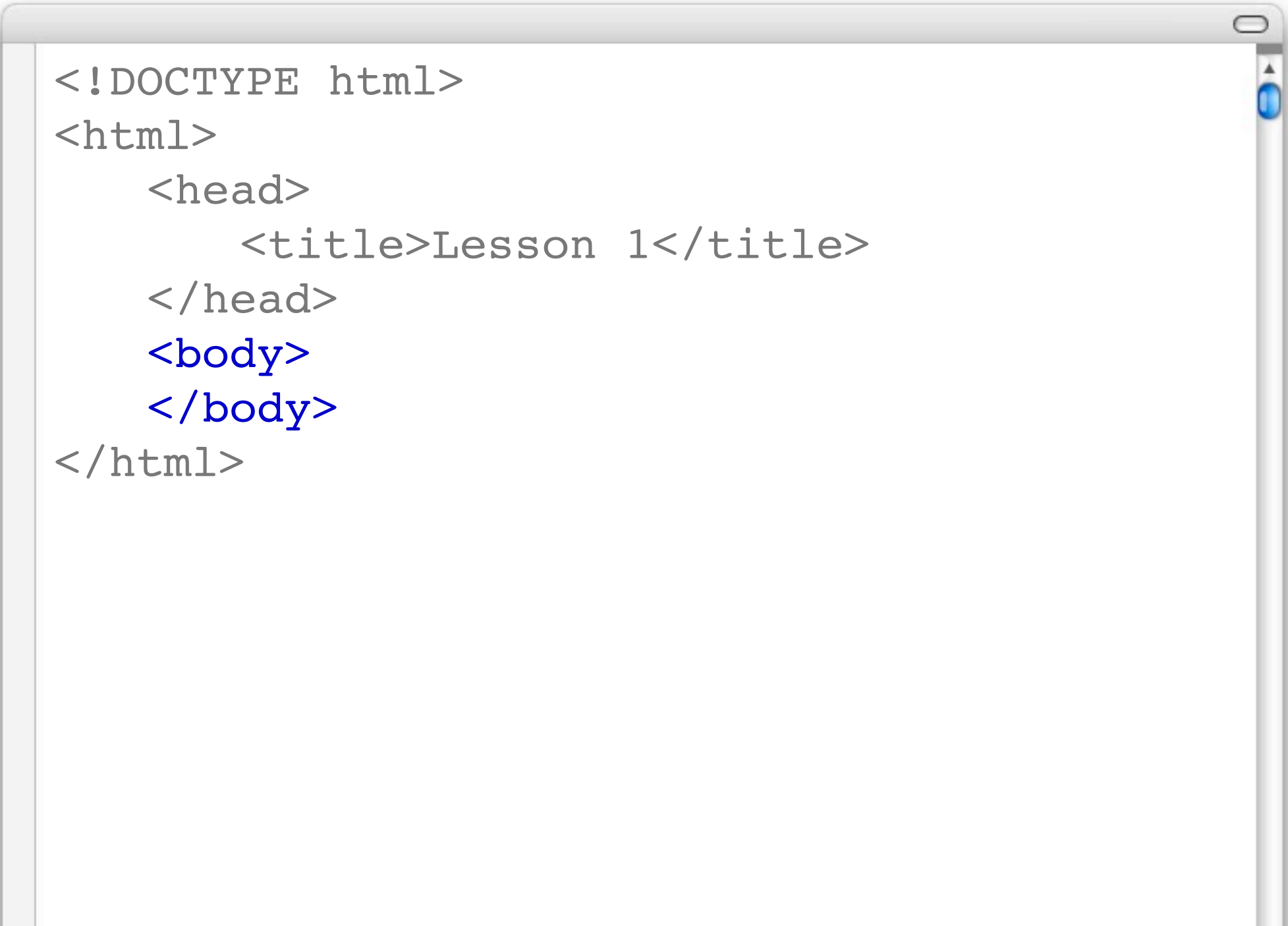

Step 5:

<body> element

The `<body>` element
**contains all the contents of
an HTML document**, such
as headings, paragraphs,
hyperlinks, images, tables,
lists etc.

Exercise

Add the `<body>` element...

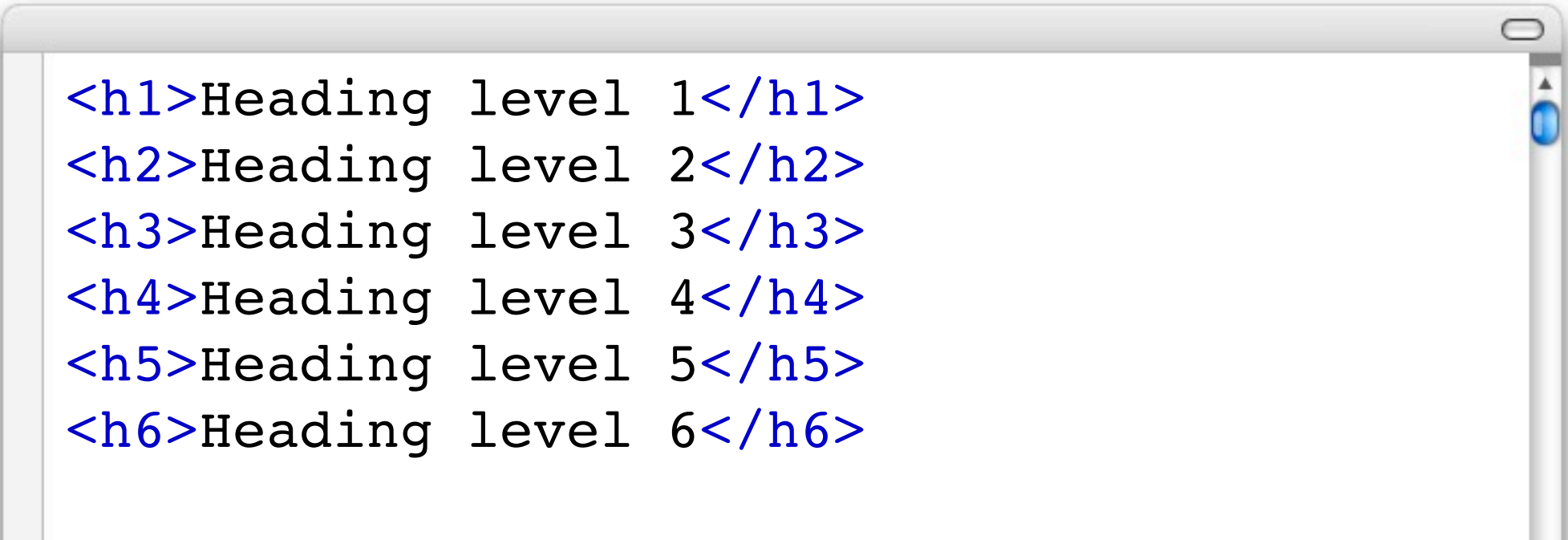


```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
  </body>
</html>
```

Step 6: heading elements

To add headings and subheadings to our document, we can use one of the **heading elements**.

There are **six different heading levels:**

A code editor window with a light gray border and a white background. It contains six lines of HTML code, each representing a heading level from 1 to 6. The code is written in a blue monospaced font. The window has a standard macOS-style title bar with a red, yellow, and green button on the right side.

```
<h1>Heading level 1</h1>  
<h2>Heading level 2</h2>  
<h3>Heading level 3</h3>  
<h4>Heading level 4</h4>  
<h5>Heading level 5</h5>  
<h6>Heading level 6</h6>
```

These headings are **ranked in importance**. The `<h1>` element is the highest rank, the `<h6>` element has the lowest rank.

Ideally, heading levels should be set out **in a hierarchical order** - in the same way you would use if you were writing an essay.

This means that you should start with an `<h1>` for the main site name or page heading, and **avoid jumping over heading levels.**

Exercise

Add an `<h1>` and `<h2>` element...

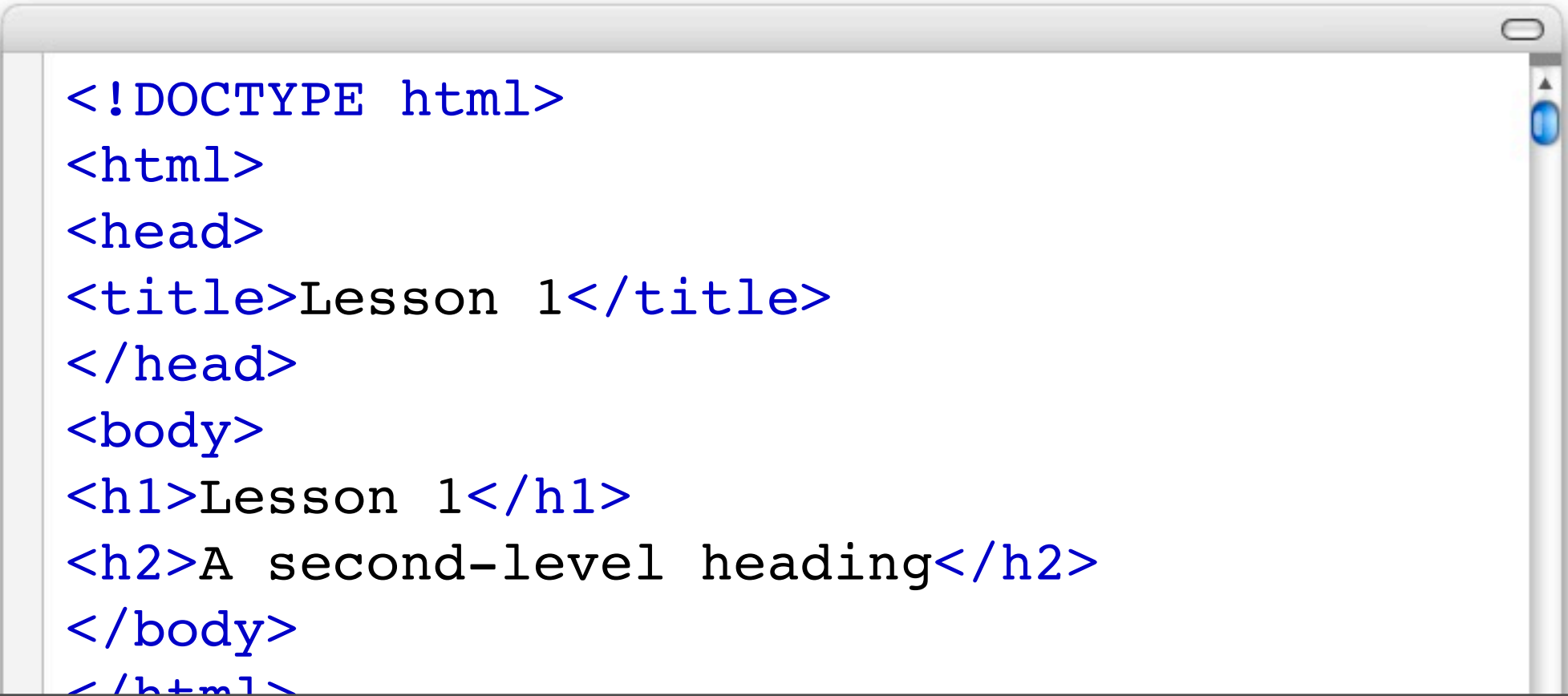
```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <h1>Lesson 1</h1>
    <h2>A second-level heading</h2>
  </body>
</html>
```

Note:
whitespace in
markup

Whitespace inside your markup is **ignored by browsers** - apart from character spaces inside content.

So, it doesn't really matter
how our markup is **visually
formatted**.

We could have our markup
without any indents...

A window with a grey title bar and a vertical scrollbar on the right. The window contains a code editor with blue text on a white background. The code is HTML markup for a document titled 'Lesson 1'. It includes a DOCTYPE declaration, a head section with a title, and a body section with two headings: 'Lesson 1' and 'A second-level heading'. The code is written without any indentation for the opening and closing tags.

```
<!DOCTYPE html>
<html>
<head>
<title>Lesson 1</title>
</head>
<body>
<h1>Lesson 1</h1>
<h2>A second-level heading</h2>
</body>
</html>
```


Or **with lots in indents...**

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <h1>
      Lesson 1
    </h1>
    <h2>
```

It could all be written as **one long string** if we wanted!

```
<!DOCTYPE html><html><head><title>Lesson  
1</title></head><body><h1>Lesson 1</  
h1><h2>A second-level heading</h2></body>  
</html>
```

Step 7:

<p> element

To add a paragraph of text to our page, we can use a `<p>` or **paragraph element**.

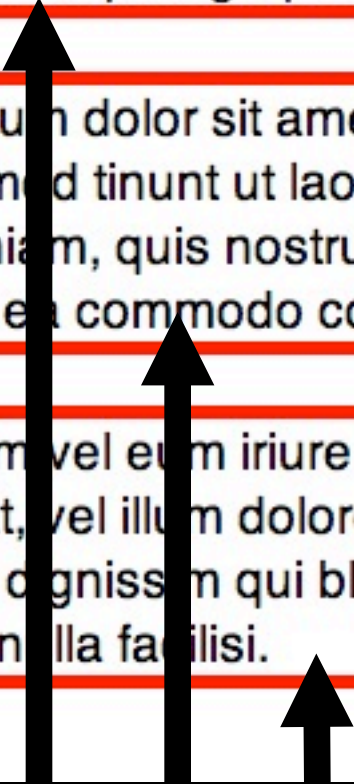
By default the paragraph element will generate whitespace **above and below** the paragraph.

This is a heading level 1

This is a short paragraph of text.

Lorem ipsum dolor sit amet consectetur adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.



**Space above and below
each paragraph**

Exercise

Add a `<p>` element...

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <h1>Lesson 1</h1>
    <h2>A second-level heading</h2>
    <p>A paragraph of text.</p>
  </body>
</html>
```


Step 8:

 element

If you want some text to appear on a new line, you can use the `
` element. **This will create a line break.**

Some people use multiple `
` elements to create fake paragraphs. **This is not a good practice.** If there needs to be whitespace between blocks of text, it is better to use `<p>` elements.

<p>

A paragraph of text

A second line of text

</p>

Bad practice

<p>

A paragraph of text

</p>

<p>

A second paragraph of text

</p>

Good practice

Exercise

Add a `
` element...

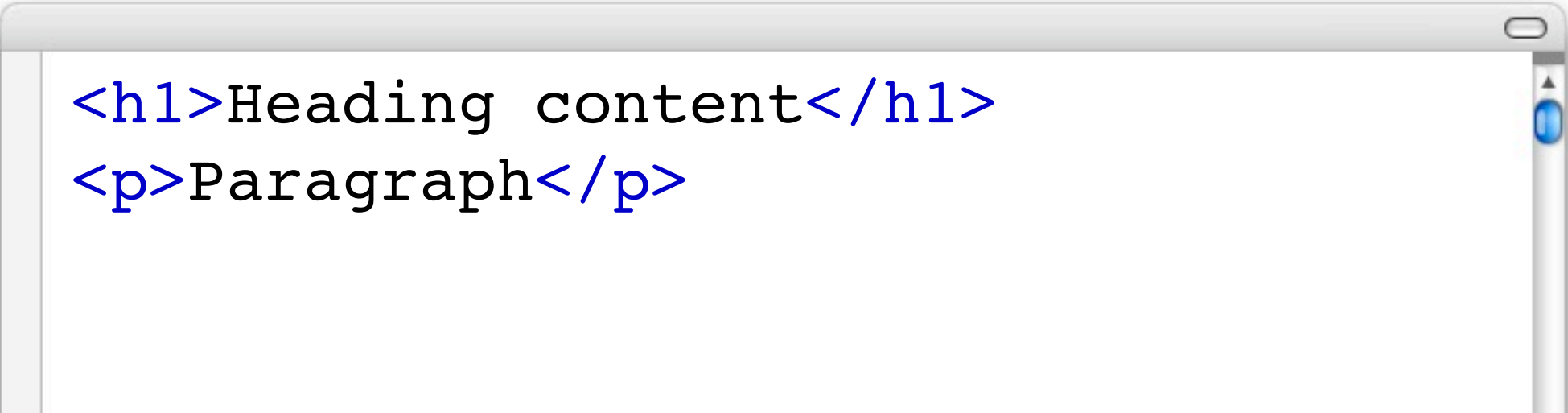
```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <h1>Lesson 1</h1>
    <h2>A second-level heading</h2>
    <p>A paragraph of text.</p>
    <p>
      123 Bent Street<br>
      South Pole
    </p>
  </body>
</html>
```

Note:

Block vs inline elements

HTML elements are often defined as either **block level** or **inline elements**. This determines how they are displayed in a browser.

Block level elements are elements that are formatted **visually as blocks** with whitespace above and below.



```
<h1>Heading content</h1>  
<p>Paragraph</p>
```

This is a heading level 1

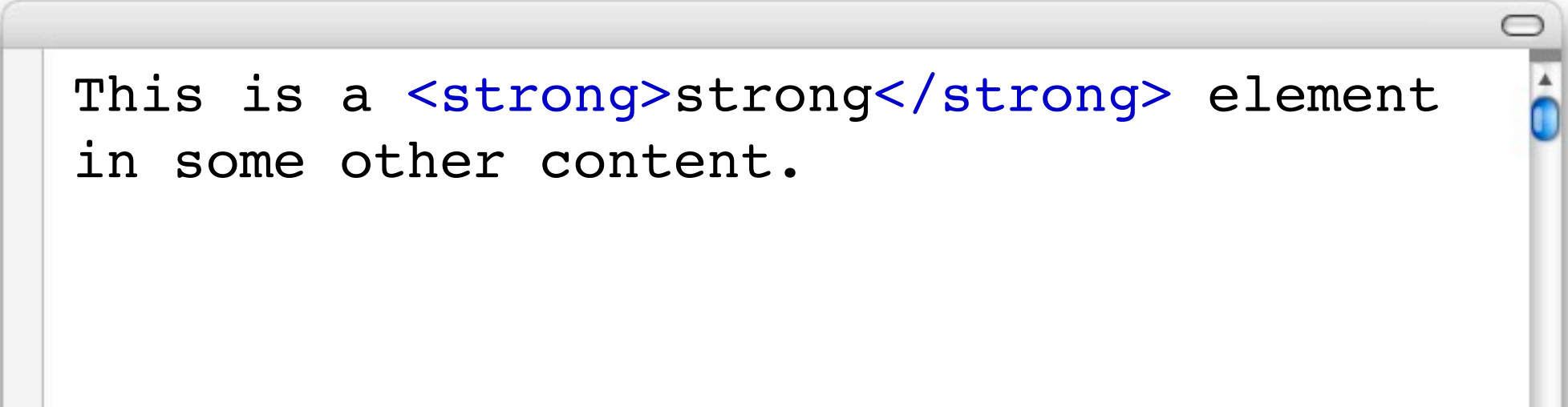
This is a short paragraph of text.

Lorem ipsum dolor sit amet consectetur adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volut. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.

**Block level elements
(display as blocks)**

Inline elements are elements that do not form new blocks of content; the content is **distributed in lines.**



This is a `strong` element in some other content.

This is a heading level 1

This is a short paragraph of text.

Lorem ipsum **dolor** sit amet consectetur adipiscing elit sed diam nonummy nibh euismod tincidunt ut laoreet *dolore magna aliquam* erat volut. Ut wisi enim ad minim veniam, quis nostrud exercitation **"ullamcorper suscipit lobortis"** nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luctatum zzril delenit augue dui dolore te feugait nulla facilisi.

**Inline elements
(displayed within lines)**

Step 9:

<div> element

Elements and semantics

Most HTML elements **have a specific purpose**. This is sometimes referred to as their “semantic meaning”.

For example, when you wrap a `<p>` element around some content, this content would be **defined as “paragraph content”**.

This often changes **how the content is displayed in a browser**. As we saw before, a default paragraph will be displayed with whitespace above and below.

The `<div>` and ``

The **<div>** and the **** are “generic” elements - they have no specific meaning. These elements are designed to be used as containers, to wrap around other elements.

The `<div>` is a **block-level** container and the `` is an **inline** container.

The `<div>`

We can use the `<div>` element to **wrap around other elements** and create “chunks” of content. We can then treat these “chunks” of content differently if needed.

For example, we might want to wrap a container around a series of paragraphs and give them a **different appearance**.


This is a heading level 1

Lorem ipsum dolor sit amet consectetur adipi scing elit sed diam nonummy nibh euismod tinunt ut laoreet dolore magna aliquam erat volut.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat.

Vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue dui dolore te feugait nulla facilisi.



**A chunk of content
styled differently**

Exercise

Add a `<div>` element around all the HTML elements inside the `<body>` element.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <div>
      <h1>Lesson 1</h1>
      <h2>A second-level heading</h2>
      <p>A paragraph of text.</p>
      <p>
        123 Bent Street<br>
        South Pole
      </p>
    </div>
  </body>
```

Step 10:

 element

The `` element
represents an **inline generic
container**.

The span is used to **wrap around fragments of inline content**. Like the `<div>` element, we can then treat these fragments differently if needed.

Exercise

Add a `` element...

<p>

Some text with a span.

</p>

Note:

HTML attributes

HTML attributes are used to assign **additional properties** to HTML elements.



```
<p class="intro">Some text</p>
```

Attributes generally consist of an **attribute and a value.**

Attribute

Value




The diagram consists of two black rectangular boxes, one labeled 'Attribute' and one labeled 'Value', positioned above a simulated code editor window. A black arrow points from the 'Attribute' box to the 'class=' part of the HTML code, and another black arrow points from the 'Value' box to the 'intro' part of the HTML code.

```
<p class="intro">Some text</p>
```

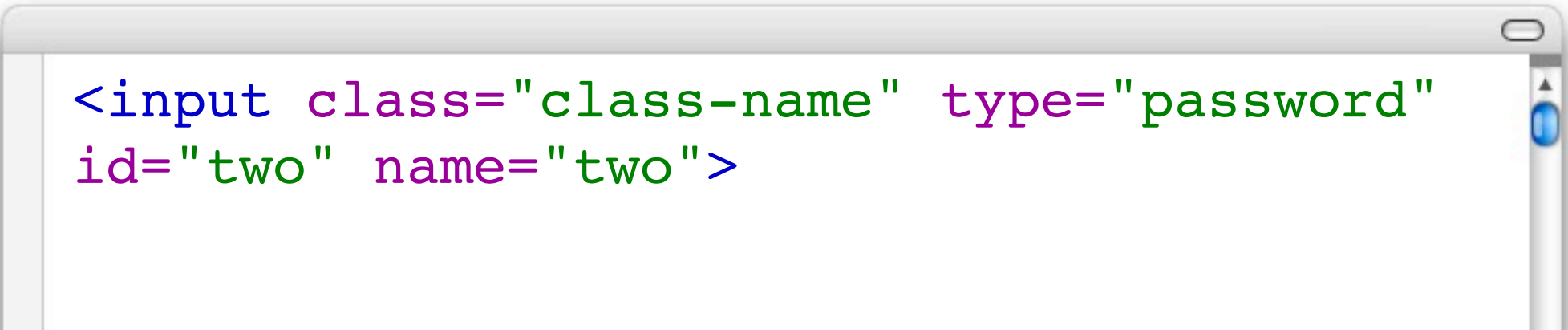
Attributes must **appear in the start tag** (or within the tag in the case of void elements)

Start tag



```
<p class="intro">Some text</p>  
<hr class="border">
```

HTML elements can have
more than one attribute
assigned to them.

A screenshot of a code editor window with a light gray title bar and a vertical scrollbar on the right. The editor contains two lines of HTML code. The first line is <input class="class-name" type="password" and the second line is id="two" name="two">. The code is color-coded: the opening tag <input is blue, the attribute names class, type, id, and name are purple, and the values "class-name", "password", "two", and "two" are green. The closing tag > is also green.

```
<input class="class-name" type="password"  
id="two" name="two">
```

Attributes must be
separated by spaces and
can appear in any order.



```

```

Attribute values can be placed inside **double quotes**.



```
<p class="intro">Some text</p>
```

Attribute values can also be placed inside **single quotes**.

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. The opening tag <p is blue, the attribute class= is purple, the value 'intro' is green, and the closing tag >Some text</p> is blue. The text 'Some text' is in black.

```
<p class='intro'>Some text</p>
```

Attribute values can even be written **without quotes** - as long as the value contains no spaces or special characters.

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: `<p class=intro>Some text</p>`. The code is color-coded: the opening and closing tags are blue, the attribute name 'class' is purple, and the value 'intro' is green. The text 'Some text' is black.

```
<p class=intro>Some text</p>
```


Boolean attributes

In HTML 4.01, some attributes had to be **written out in full** - even though the attribute and value were the same.

A screenshot of a code editor window with a light gray border and a blue scrollbar on the right. The text inside is an HTML option tag: `<option selected="selected">Opt1</option>`. The tags are blue, the attribute name is purple, the value is green, and the text 'Opt1' is black.

```
<option selected="selected">Opt1</option>
```

In HTML5, these attributes can be shortened. They are referred to as “**boolean attributes**”.

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: `<option selected>Option 1</option>`. The word `selected` is highlighted in purple, while the other parts of the code are in blue.

```
<option selected>Option 1</option>
```

If a boolean attribute is present it is considered to be true. If not present, it is **considered to be false.**



```
<option selected>Option 1</option>
```

Here is a list of **most of the
boolean attributes**
available today.

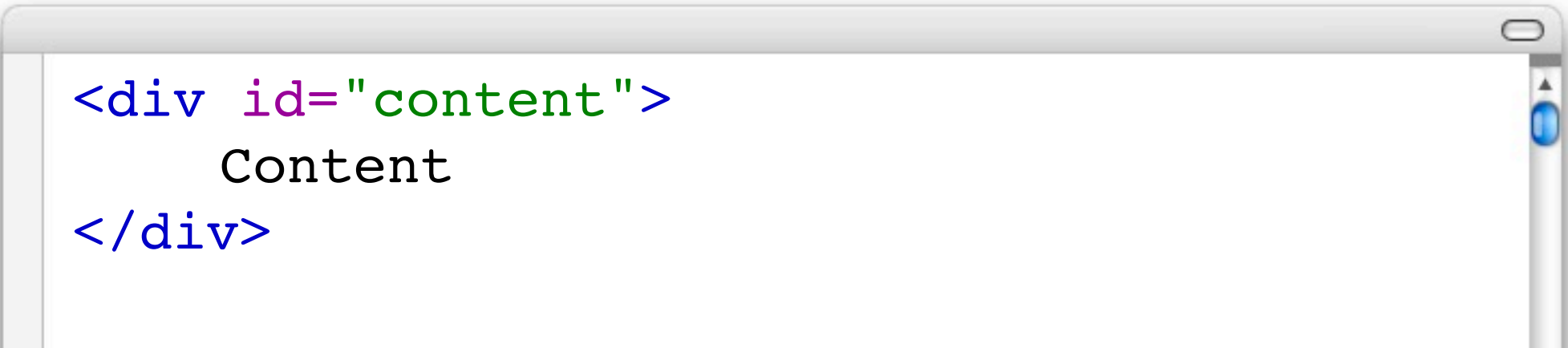
```
<input type="text" autofocus>
<input type="checkbox" checked>
<object id="object" declare>
<script src="demo.js" defer></script>
<input type="text" autofocus>
<input type="button" disabled>

<input type="email" multiple>
<area shape="rect" nohref>
<frame src="frame_a.htm" noresize>
<input type="text" readonly>
<input type="email" required>
<option value="a" selected>A</option>
```

Step 11:

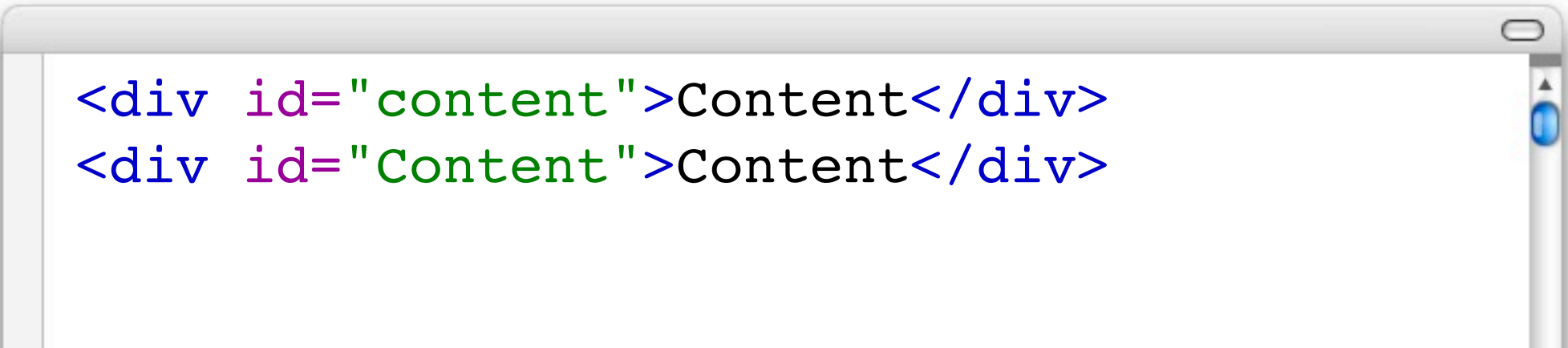
ID attribute

The **ID attribute** assigns a name to an element.

A code editor window with a light gray border and a white background. It contains three lines of HTML code. The first line is an opening tag for a div element with an id attribute set to 'content'. The second line is the word 'Content' centered within the div. The third line is the closing tag for the div element. The code is color-coded: the opening and closing tags are blue, the id attribute and its value are green, and the text 'Content' is black.

```
<div id="content">  
    Content  
</div>
```

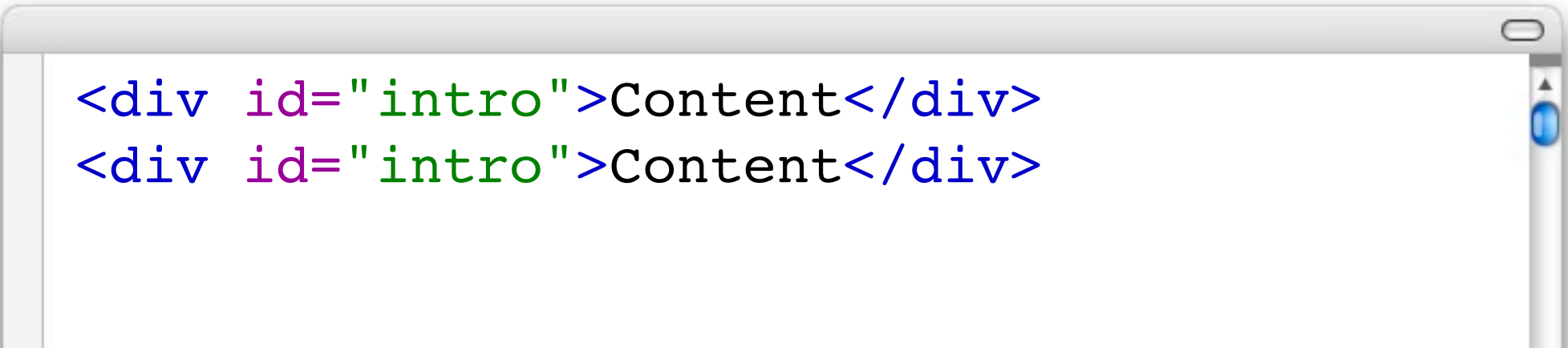

ID attributes are **case sensitive**. The two ID attributes below would be treated as different values.



```
<div id="content">Content</div>  
<div id="Content">Content</div>
```

ID attributes must be
unique within a document.

You cannot have two ID
attributes with the same
value.



```
<div id="intro">Content</div>  
<div id="intro">Content</div>
```

Exercise

Add an ID attribute...

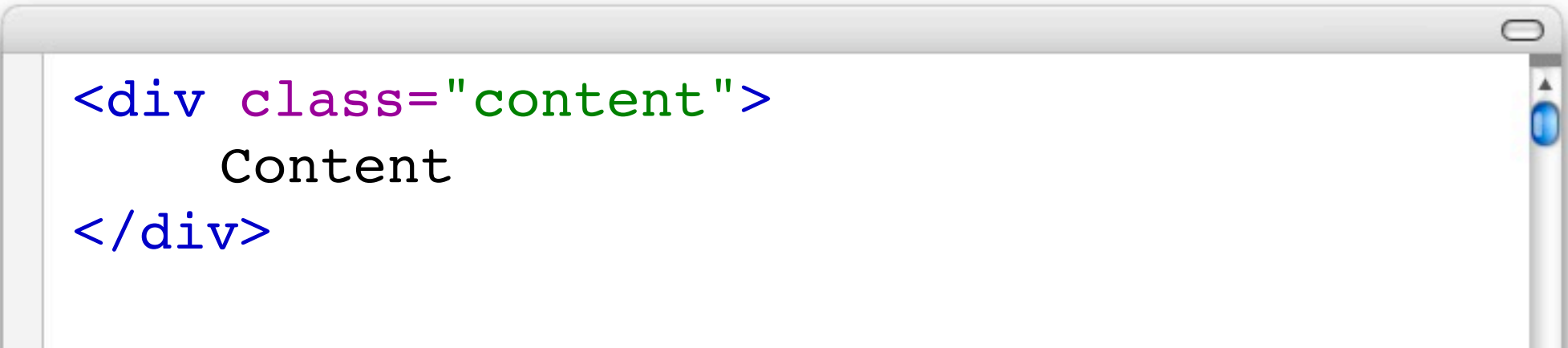
```
<!DOCTYPE html>
<html>
  <head>
    <title>Lesson 1</title>
  </head>
  <body>
    <div id="container">
      <h1>Lesson 1</h1>
      <h2>A second-level heading</h2>
      <p>A paragraph of text.</p>
      <p>
        123 Bent Street<br>
        South Pole
      </p>
      <p>
        Some text with a <span>span</

```

Step 12:

CLASS attribute

The **class attribute** assigns a class name or set of class names to an element.

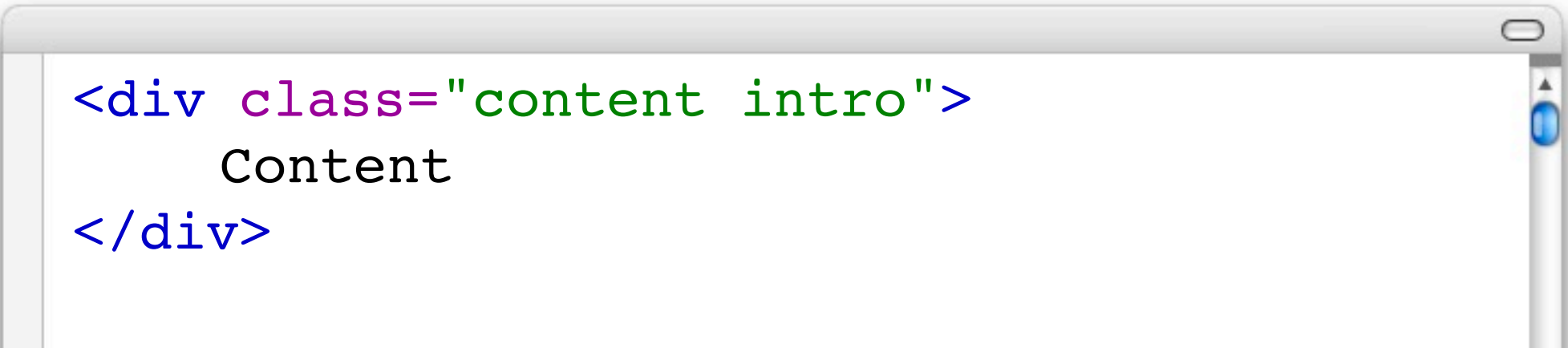
A code editor window with a light gray border and a white background. It contains three lines of HTML code. The first line is an opening tag with a class attribute. The second line is the text content. The third line is the closing tag. The code is color-coded: opening and closing tags are blue, the class attribute is purple, the class value is green, and the text content is black.

```
<div class="content">  
    Content  
</div>
```

Any number of elements
may be given the same
class name.

```
<div class="content">  
  <p class="content">Paragraph text</p>  
</div>
```

You can assign **more than one class to an element**.
Multiple class names must be separated by whitespace.

A code editor window with a light gray title bar and a vertical scrollbar on the right. The code is displayed in a monospaced font with syntax highlighting: the opening tag is blue, the attribute name is purple, the value is green, and the closing tag is blue.

```
<div class="content intro">  
    Content  
</div>
```


Exercise

Add a CLASS attribute...

<p>

Some text with a <span
class="highlight">span.

</p>



Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley