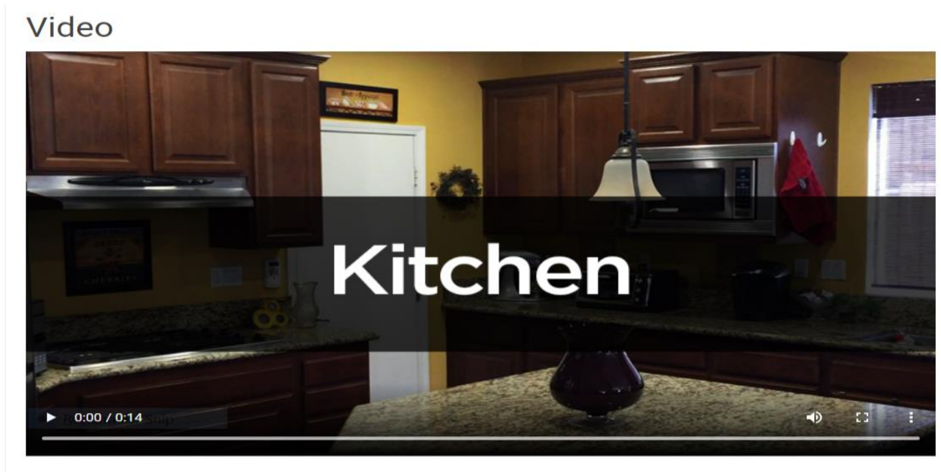
HTML5

Day 2

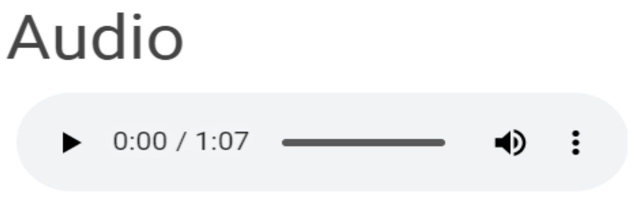
ASSIGNMENTS

1. Create an HTML5 page which plays a video when the page is loaded. The video must loop and must initially display some image as shown in the snapshot below:



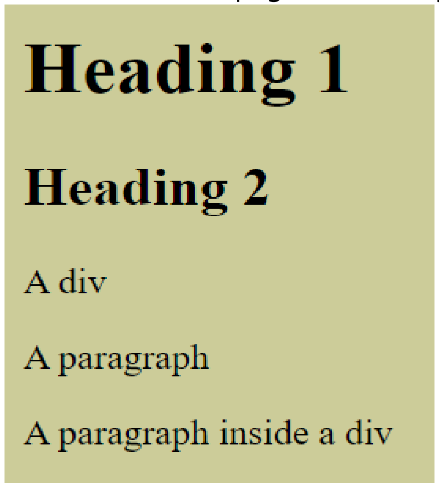
Supported formats must be **mp4, ogy** and **webm.**

2. Create an HTML5 page which plays an audio with the following UI:



Supported formats must be **mp3, ogg** and **wav.**

3. Create an HTML5 page without styles as follows:



Heading 1

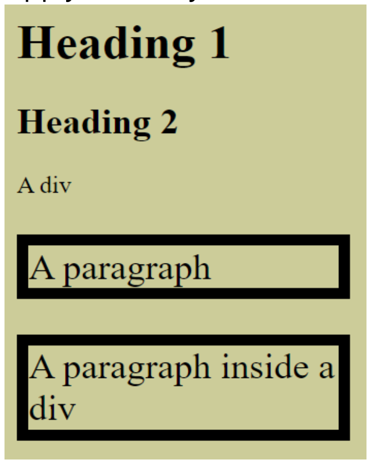
Heading 2

A div

A paragraph

A paragraph inside a div

Apply some styles so that the page now looks as shown below:



Heading 1

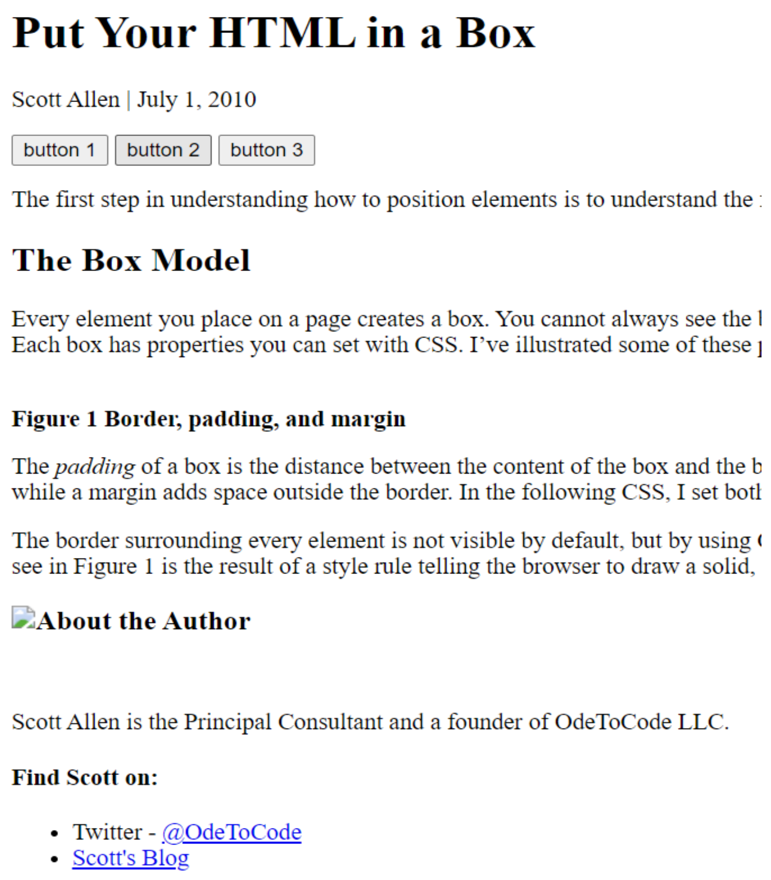
Heading 2

A div

A paragraph

A paragraph inside a div

4. Create an HTML5 page without styles as follows:



Put Your HTML in a Box

Scott Allen | July 1, 2010

button 1 button 2 button 3

The first step in understanding how to position elements is to understand the

The Box Model

Every element you place on a page creates a box. You cannot always see the
Each box has properties you can set with CSS. I've illustrated some of these

Figure 1 Border, padding, and margin

The *padding* of a box is the distance between the content of the box and the
while a margin adds space outside the border. In the following CSS, I set both

The border surrounding every element is not visible by default, but by using
see in Figure 1 is the result of a style rule telling the browser to draw a solid,

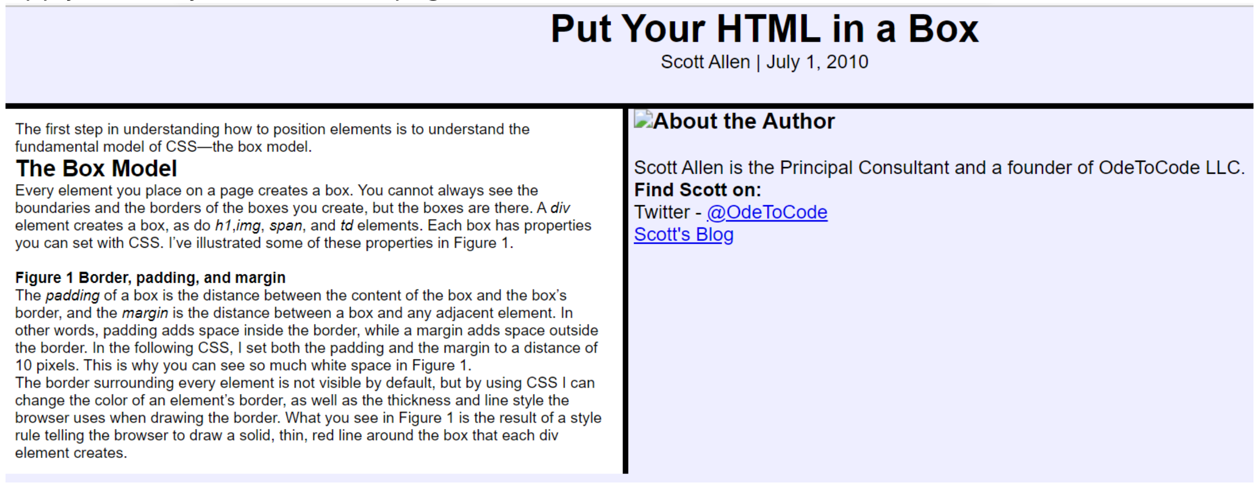
About the Author

Scott Allen is the Principal Consultant and a founder of OdeToCode LLC.

Find Scott on:

- Twitter - @OdeToCode
- Scott's Blog

Apply some styles so that the page now looks as shown below:



Put Your HTML in a Box

Scott Allen | July 1, 2010

The first step in understanding how to position elements is to understand the fundamental model of CSS—the box model.

The Box Model

Every element you place on a page creates a box. You cannot always see the boundaries and the borders of the boxes you create, but the boxes are there. A *div* element creates a box, as do *h1*,*img*, *span*, and *td* elements. Each box has properties you can set with CSS. I've illustrated some of these properties in Figure 1.

Figure 1 Border, padding, and margin

The *padding* of a box is the distance between the content of the box and the box's border, and the *margin* is the distance between a box and any adjacent element. In other words, padding adds space inside the border, while a margin adds space outside the border. In the following CSS, I set both the padding and the margin to a distance of 10 pixels. This is why you can see so much white space in Figure 1.

The border surrounding every element is not visible by default, but by using CSS I can change the color of an element's border, as well as the thickness and line style the browser uses when drawing the border. What you see in Figure 1 is the result of a style rule telling the browser to draw a solid, thin, red line around the box that each div element creates.

About the Author

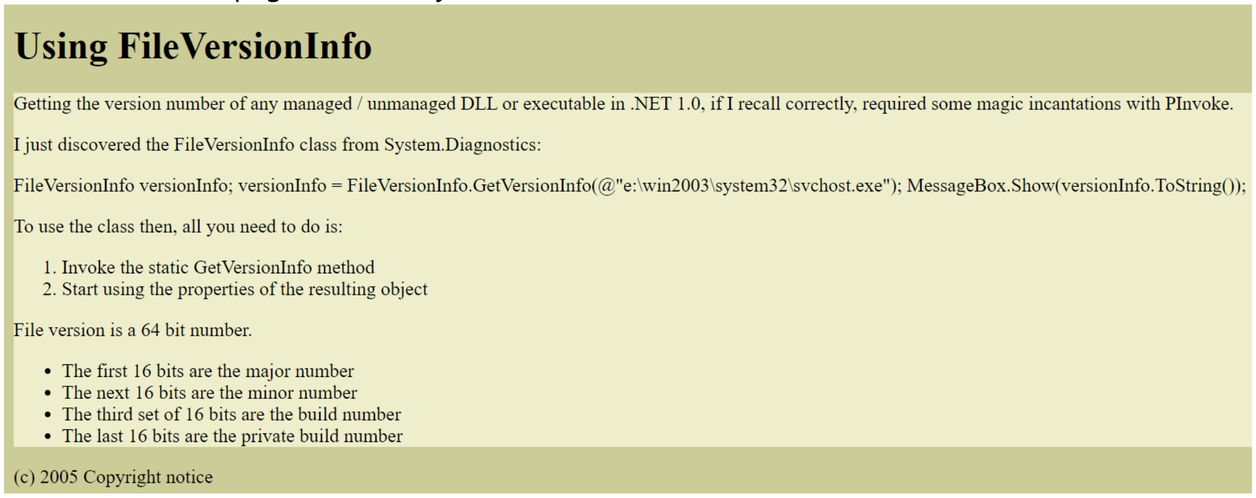
Scott Allen is the Principal Consultant and a founder of OdeToCode LLC.

Find Scott on:

Twitter - @OdeToCode

Scott's Blog

5. Create an HTML5 page without styles as follows:



Using FileVersionInfo

Getting the version number of any managed / unmanaged DLL or executable in .NET 1.0, if I recall correctly, required some magic incantations with PInvoke.

I just discovered the FileVersionInfo class from System.Diagnostics:

FileVersionInfo versionInfo; versionInfo = FileVersionInfo.GetVersionInfo(@"e:\win2003\system32\svchost.exe"); MessageBox.Show(versionInfo.ToString());

To use the class then, all you need to do is:

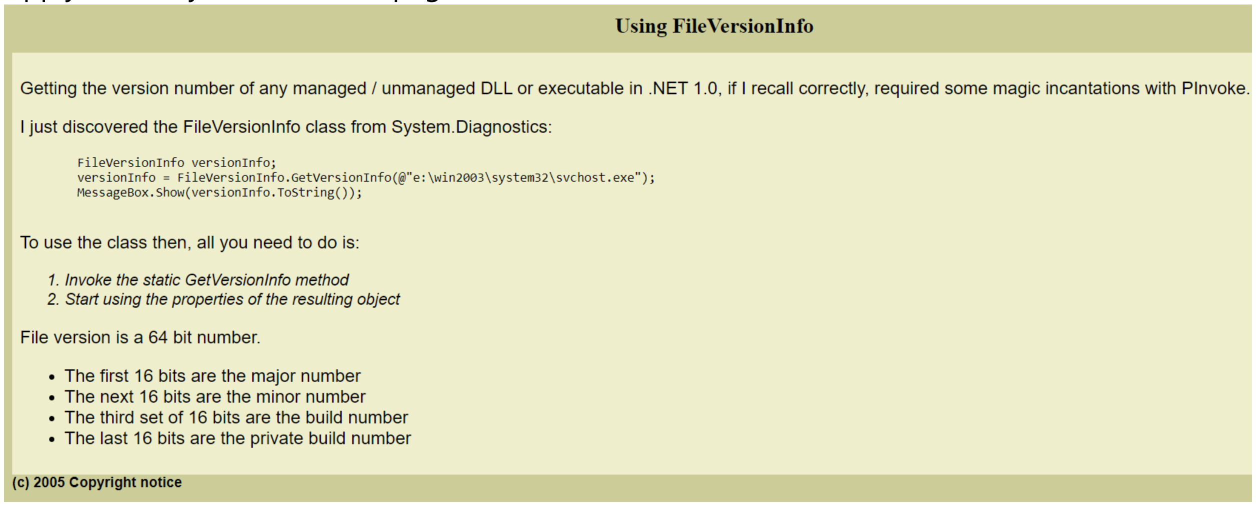
1. Invoke the static GetVersionInfo method
2. Start using the properties of the resulting object

File version is a 64 bit number.

- The first 16 bits are the major number
- The next 16 bits are the minor number
- The third set of 16 bits are the build number
- The last 16 bits are the private build number

(c) 2005 Copyright notice

Apply some styles so that the page now looks as shown below:



Using FileVersionInfo

Getting the version number of any managed / unmanaged DLL or executable in .NET 1.0, if I recall correctly, required some magic incantations with PInvoke.

I just discovered the FileVersionInfo class from System.Diagnostics:

```
FileVersionInfo versionInfo;
versionInfo = FileVersionInfo.GetVersionInfo(@"e:\win2003\system32\svchost.exe");
MessageBox.Show(versionInfo.ToString());
```

To use the class then, all you need to do is:

1. *Invoke the static GetVersionInfo method*
2. *Start using the properties of the resulting object*

File version is a 64 bit number.

- The first 16 bits are the major number
- The next 16 bits are the minor number
- The third set of 16 bits are the build number
- The last 16 bits are the private build number

(c) 2005 Copyright notice