7.4 **iframe Element**

You’ve learned how to display an image in the body of a web page and also in the tab area at the top of the browser window. In this section, you learn to display an image in a frame that’s embedded within the browser window. Specifically, you’ll use an iframe (short for inline frame) element to create a *browsing context*. A browsing context is an area within a web page that can display an embedded *web document*, where a web document is something with a URL (i.e., a web page or a stored image).

Instead of loading an entire web page within an iframe’s browsing context, it’s more common to load a stored image into a browsing context. For example, in the Art Exhibit web page in FIGURE 7.4, an iframe element is used to display the large picture in the center. The Art Exhibit web page’s website stores image files for that picture and three other pictures that all have the same dimensions. In addition, the website stores image files for the four smaller pictures at the left of Figure 7.4. Those pictures are smaller versions of the larger pictures. When the user clicks one of the smaller images, the browser grabs the larger version of the clicked image and copies it to the browsing context area. To make sure you understand what’s going on, please find the Art Exhibit web page on the student resources website and try it out for yourself. The smaller images are known as thumbnails. A *thumbnail* is a smallish image that serves as a representative for a larger version of that same image. Thumbnails are often used to help with the organization of a group of images. In conjunction with that effort, they can help users to identify and select standard-size images.

Take a look at the Art Exhibit web page’s source code in FIGURE 7.5A. In particular, note the iframe code, copied here for your convenience:

<iframe class="cell" name="full-size" width="480" height="320" src="../images/houseRenderings/kitchen.jpg"></iframe>

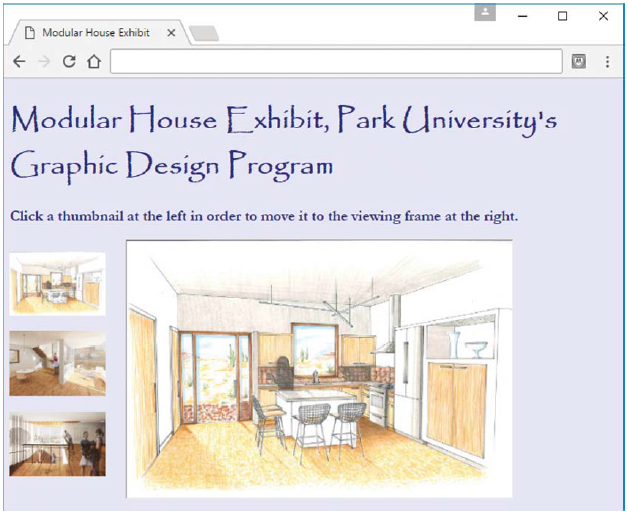
The iframe element’s width and height attributes are self-explanatory. The iframe element’s src attribute specifies the web document that initially appears within the iframe’s browsing context. The src attribute can specify a path to another web page, in which case the browser 

FIGURE 7.4 Art Exhibit web page

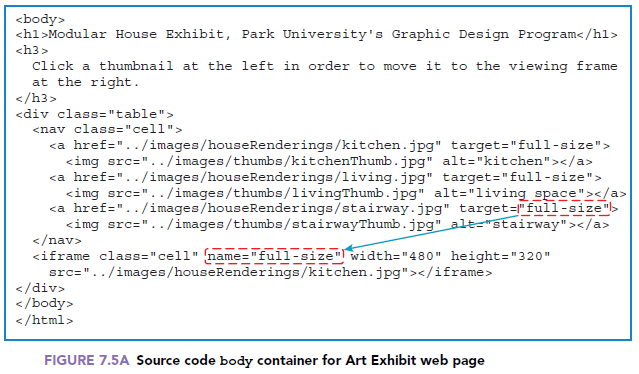
displays the entire specified web page in the browsing context area. But the prior code doesn’t specify a path to another web page. Instead, it specifies an image file, man.jpg. And when a user clicks one of the thumbnails, the image gets replaced by the image specified by the thumbnail.

We’ll get to the thumbnail details soon enough, but first let’s finish going over the iframe’s attributes.

In order to load a new image into the browsing context area, we need to be able to refer to the iframe element, and the name attribute’s value allows us to do that. As you can see in the previous code, the iframe element’s name attribute has a value of full-size. In the Art Exhibit web page, the a elements specify target="full-size" to connect to the iframe element. In the following code (from the Art Exhibit web page), note the target attribute and its full-size value:

<a href="../images/houseRenderings/kitchen.jpg" target="full-size">

<img src="../images/thumbs/kitchenThumb.jpg" alt="kitchen"></a>



As always for an a element, the href attribute indicates what to load when the user clicks the link. In the past, we’ve only loaded another web page, but you can also load an image with the href attribute, and that’s what the prior a element code indicates. If there were no target attribute, the browser would implicitly use the value \_self for the target attribute, which tells the browser to overlay the current web page with the href value. But with target="full-size", the target attribute tells the browser to open the man.jpg image in the full-size browsing context.

Remember that with the a element, it’s the content between the start and end tags that provides the interface for the user to click on. Normally, that content is simply text, so the user clicks text to activate the link. But as you can see in Figure 7.5A, we’re using a thumbnail image instead of text for the user to click on.

Formatting

Before leaving the Art Exhibit web page, there are a few formatting issues that are worth noting. To get the thumbnails to appear vertically at the left of the iframe’s browsing context area, we use a one-row two-cell table. We use a nav container to hold all the thumbnails and form the left cell.

We use the iframe to form the right cell. Go to Figure 7.5A to verify the table structure. You’ll see that the two-cell table is built with the CSS display property and table values. You’ll see that there is no element for the table’s row, which means we’re using an anonymous table-row element.

Peruse FIGURE 7.5B to see all the CSS rules. Here’s the rule that implements the table:

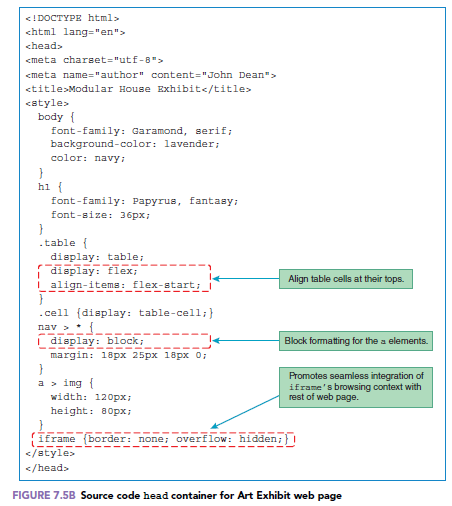
.table {

display: table;

display: flex;

align-items: flex-start;

}



The display: table property-value pair converts a div container to a table, which is pretty straightforward. But the other two property-value pairs took quite a bit of head scratching to come up with. Initially, the top thumbnail’s bottom edge aligned with the large picture’s bottom edge, which made for a lot of dead space above the top thumbnail. To get the top thumbnail’s top to align with the large picture’s top, we introduced the flexible box layout (using display: flex) and the align-items: flex-start property-value pair. The horribly named flex-start value indicates top alignment.

Here’s the rule that causes the nav and iframe elements to become table cells, and it should look familiar:

.cell {display: table-cell;}

And here’s the rule that formats the individual a elements within the nav container:

nav > \* {

display: block;

margin: 18px 25px 18px 0;

}

The rule matches all of the nav container’s child elements, which are a elements. By default, a elements use inline formatting. So without this rule, the browser displays the thumbnail links horizontally. To convert the a elements to block formatting, we use the display: block property-value pair. The margin: 15px 25px 15px 0 property-value pair prevents the pictures from touching each other.

Next is the rule that specifies the thumbnail pictures’ dimensions:

a > img {

width: 120px;

height: 80px;

}

The width and height properties serve as an alternative to including width and height attribute-value pairs for each of the thumbnail img elements. By using a CSS rule, we avoid redundant attribute-value pairs, and avoiding redundancy means the code is easier to maintain.

Typically, browsers display a faint border around the iframe’s browsing context that distinguishes it from the rest the web page’s content. Normally, you’re going to want to avoid that behavior. To get the browsing context to integrate seamlessly with the rest of the Art Exhibit web page, we use the following CSS rule:

iframe {border: none; overflow: hidden;}

The border-none property-value pair makes the iframe’s browsing context border invisible.

Normally, if the iframe’s content is larger than the iframe’s dimensions, the browser displays scrollbars so all of the content can be accessed. The overflow-hidden property-value pair clips the content so no scrollbars display.