CSS3 Techniques

Overview

In this lab, you'll implement a Photo Viewer web page. The web page will display a series of photo thumbnails along the bottom of a panel. When the user clicks a thumbnail, the image will be displayed as a full-size photo in the panel.

The HTML web content and the JavaScript functionality is already complete. Your task will be to define appropriate CSS rules to make the page content look good.

Roadmap

There are 6 exercises in this lab, of which the last exercise is "if time permits". Here is a brief summary of the tasks you will perform in each exercise; more detailed instructions follow later:

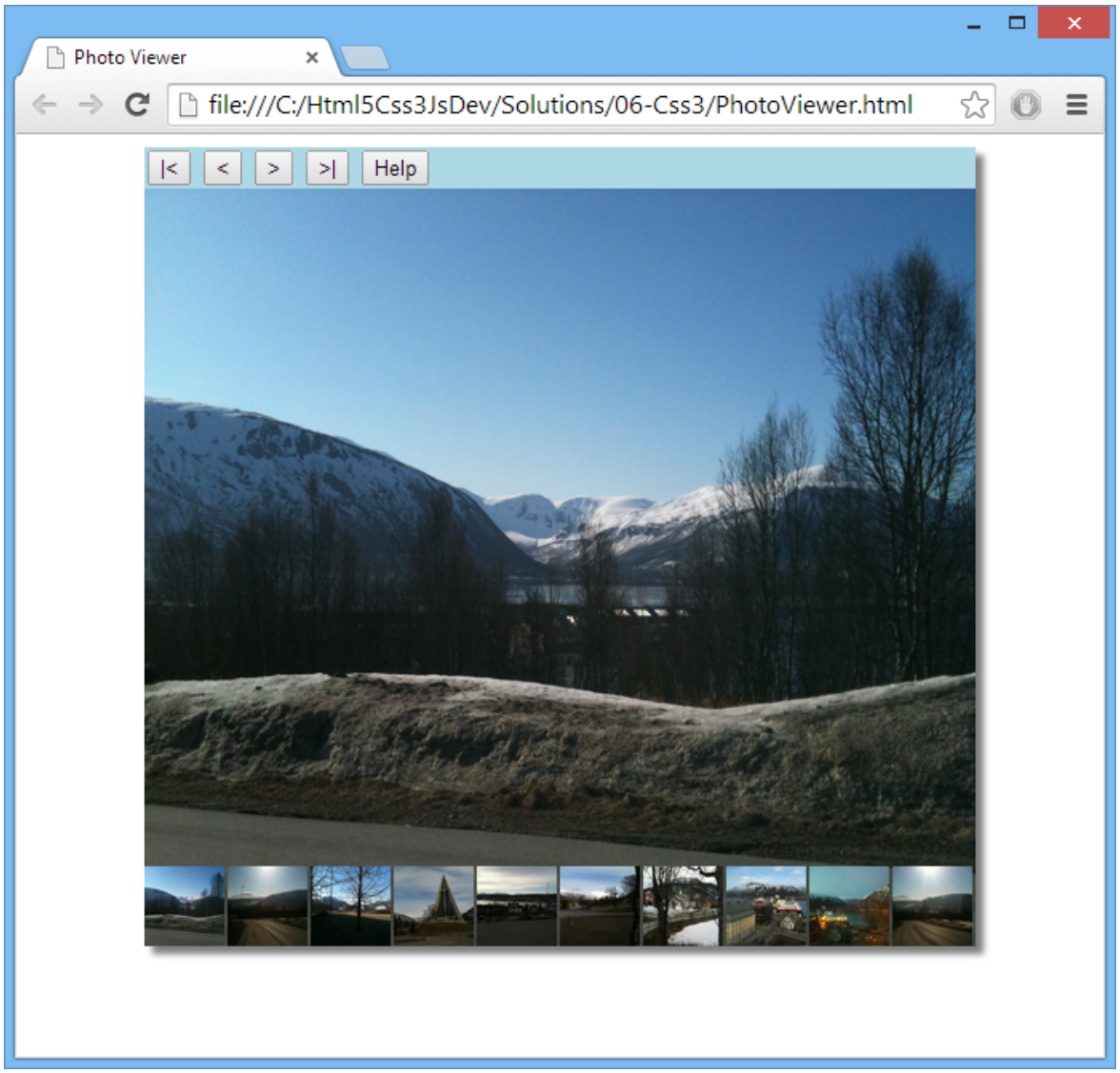
1. Defining box shadows
2. Defining styles for the main container panel
3. Defining styles for the thumbnails panel
4. Defining styles for the help panel
5. Defining styles for child elements in the help panel
6. Additional suggestions (if time permits)

Familiarization

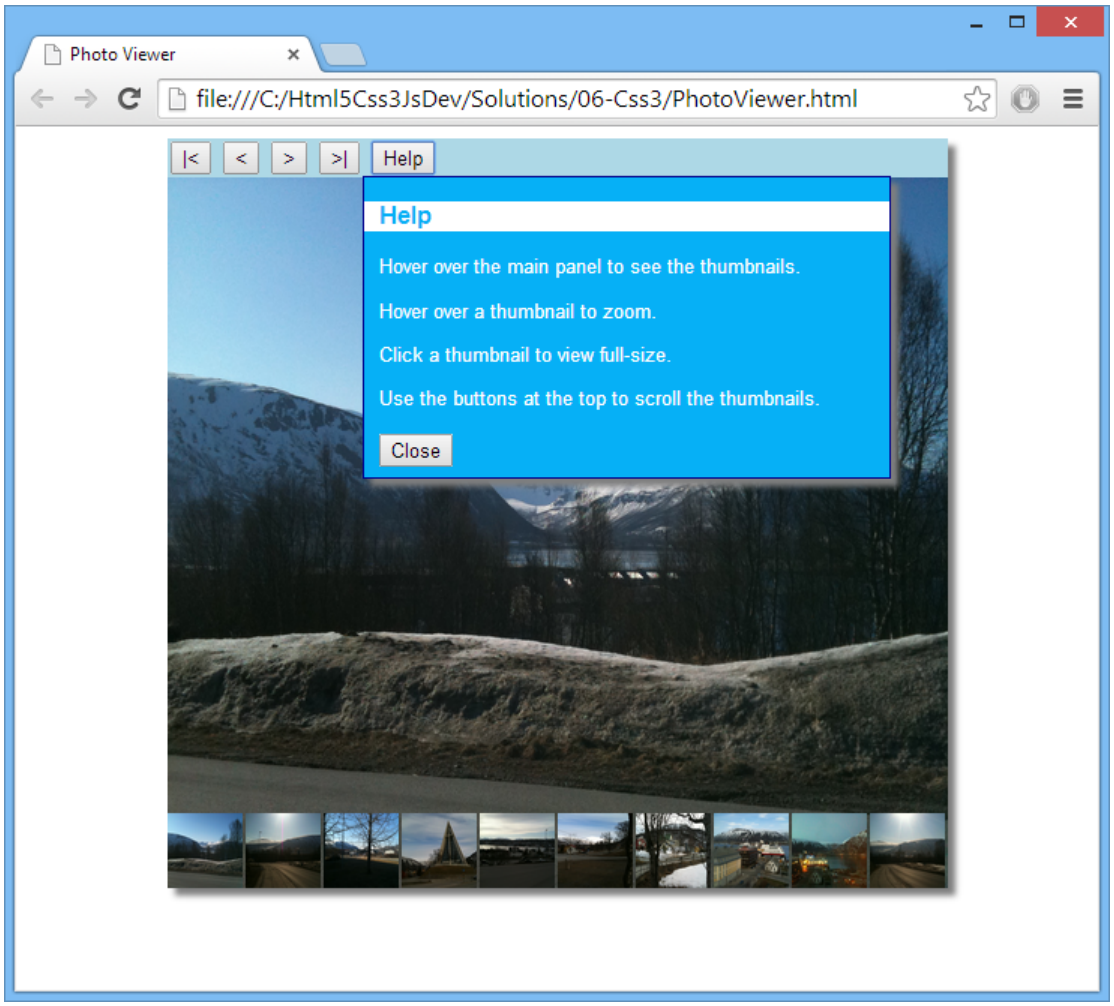
Open a browser window and browse to the *Solution* web page:

```
..\solution\PhotoViewer.html
```

The web page appears as follows initially:



Click the *Help* button at the top of the image viewer panel. A help panel expands smoothly into view, and explains how to use the application:



Try out the features of the Photo Viewer application, as described by the help panel. If you're wondering, the photos are of Tromsø and Bøde in the Norwegian Arctic Circle.

When you're happy with how the solution application works, close the browser and go to the *Start* folder:

```
\start
```

The folder contains an HTML page, a JavaScript file, and a CSS style sheet. We explain each of these files on the next page…

`PhotoViewer.html` (complete)

- The `<body>` tag has an `init()` handler, which will be called when the page has loaded. This function is defined in `PhotoViewerScript.js`, with all the other JavaScript code.
- The `<div>` element named `containerPanel` is a wrapper for all the content in the photo viewer web page. You'll define styles for this element, plus various other elements, in `PhotoViewerStylesheet.css` during this lab.
- The `<div>` element named `buttonsPanel` contains the buttons along the top of the photo viewer, so the user can scroll through the thumbnails panel and also display help info. Each of these buttons triggers a JavaScript handler function, which you'll see shortly.
- The `<div>` element named `helpPanel` contains the popup help info. The style sheet sets the `display:none` style for this element, so it's not displayed initially.
- The `<span>` element named `viewerPanel` displays the main image in the photo viewer, when the user clicks a thumbnail in the thumbnails panel.
- The `<ul>` element named `thumbnailsPanel` displays the thumbnails images as an unordered list (we've defined some careful styling to get the visual effect we want).

`PhotoViewerScript.js` (almost complete)

- The `init()` function loads 20 photos from the `photos` folder, and displays them as thumbnail images in the thumbnails panel.

  The function also creates a global object named `dimensions`, to hold the width of the thumbnails panel and the container panel. Note that the thumbnails panel is much wider than the container panel – i.e. the thumbnails are laid out in a straight line that shoots off the end of the container panel.

  The function then sets up various event handlers, using various jQuery techniques. The comments in the code explain the purpose of each event handler. There are also some TODO comments, which you'll tackle at the end of the lab (if time permits).
- `displayLargeImage()` and `fadeInLargeImage()` are helper functions to display an image with the specified URL in the viewer panel.
- The next 4 functions, `moveThumbnailsPanelFirst()`, `moveThumbnailsPanelRight()`, `moveThumbnailsPanelLeft()` and `moveThumbnailsPanelLast()`, scroll the thumbnails panel horizontally. These functions call on a helper function, `moveThumbnailsPanel()`, to actually do the work.
- `showHelpPanel()` and `hideHelpPanel()` show and hide the help panel, respectively.

`PhotoViewerStylesheet.css` (partial)

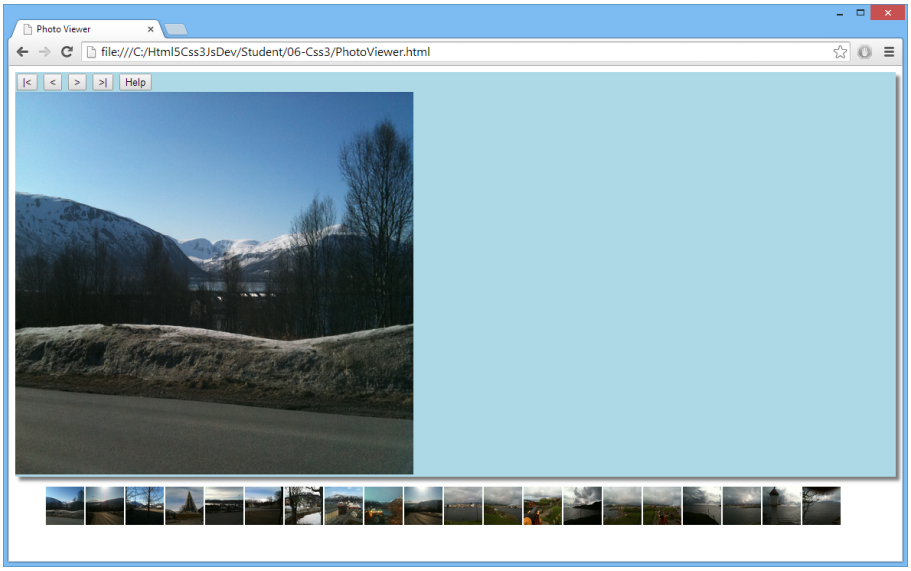
- The first four CSS rules define some basic styles for the web page, and are complete. The remaining CSS rules define the appearance of the panels themselves; you'll complete these rules in this lab.

Exercise 1: Defining box shadows

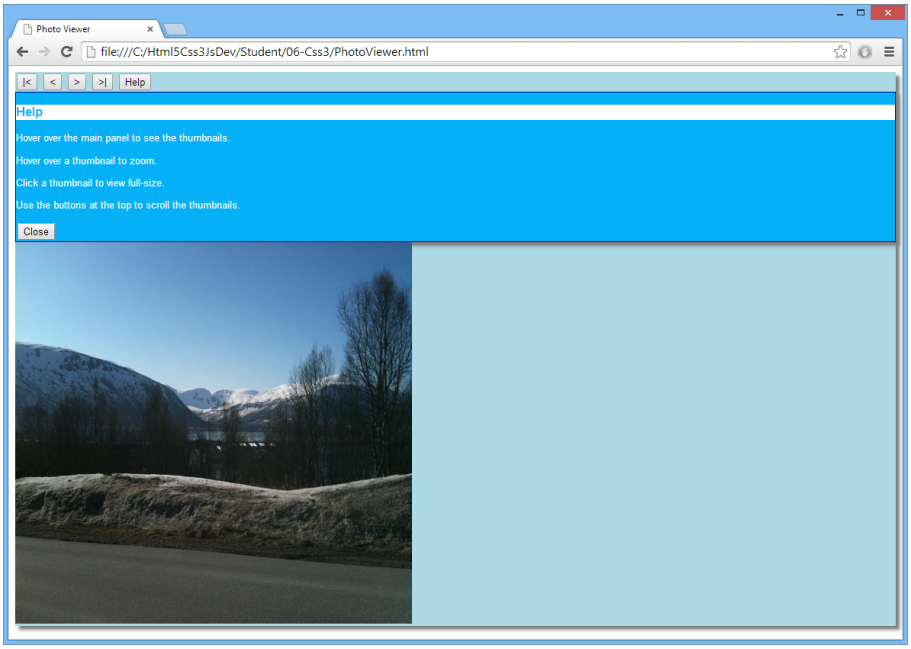
In `PhotoViewerStylesheet.css`, where indicated by the TODO 1 comment, define a CSS rule to create a box shadow for the container panel and help panel. Then open a browser window and browse to the *Start* web page:

```
\start\PhotoViewer.html
```

The web page should appear as follows initially – note the box shadow around the main container panel (don't worry about the rest of the page layout – you'll fix this later):



Click the *Help* button, and verify the help panel has a box shadow too (don't worry about page layout here either):

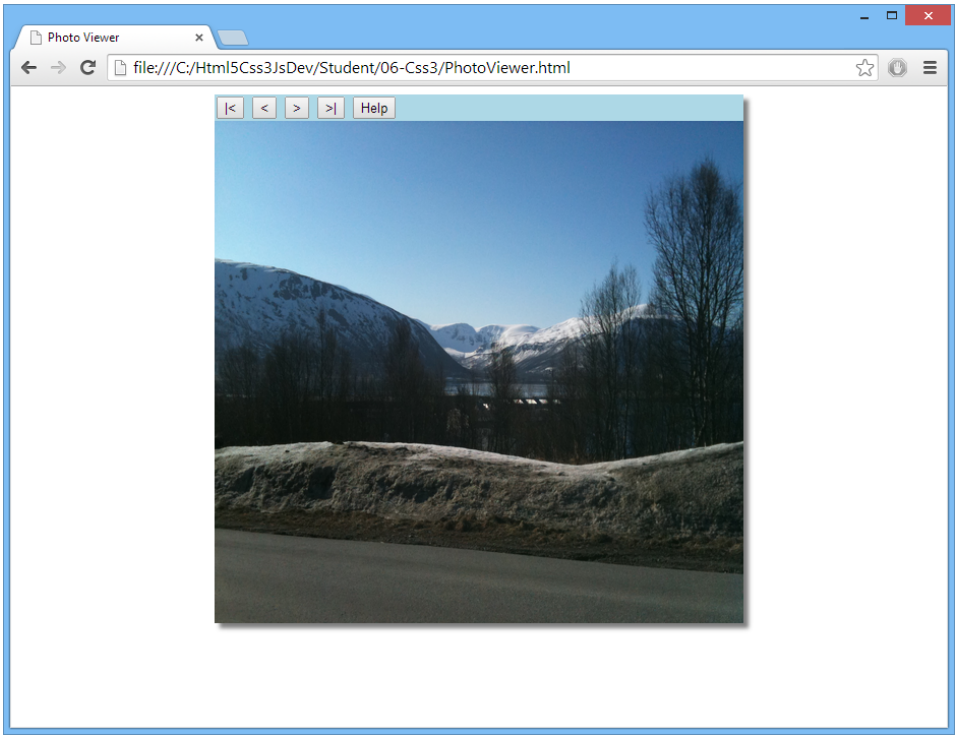


Exercise 2: Defining styles for the main container panel

In this exercise, you'll define styles to control the size and location of the main container panel. In `PhotoViewerStylesheet.css`, locate the TODO 2 comment. The CSS rule already defines some simple styles for `#containerPanel`. Add the following styles:

- Set the `width` and `height` styles to `520px`
  This is the width required to fit in 10 thumbnails horizontally at the bottom of panel (each thumbnail is 50px wide, plus 2px margin).
- Set the `margin-left` and `margin-right` styles to `auto`
  This is how you centralize an element in a web page – it causes the browser to choose the size of the margin on the left and right of the container panel, so that it appears centrally-aligned in the web page.
- Set the `position` style to `relative`
  An element with relative positioning enables you to absolutely position child elements inside it. This is very useful – it'll allow you to absolutely position the thumbnails panel inside the container panel (you'll do this in Exercise 3).
- Set the `overflow` style to `hidden`
  This means any child content that overflows the limits of the container panel will be hidden. This is very useful – it ensures the portion of the thumbnails panel that overflows off the end of the container panel won't be visible (see Exercise 3).

Go back to the browser window and refresh the view of the *Start* web page. It should now appear as follows:



Exercise 3: Defining styles for the thumbnails panel

In this exercise, you'll define styles to position the thumbnails panel at the bottom of the main container panel. In the style sheet, locate the TODO 3 comment and add the following styles:

- Set the `width` style to `1240px`
  This is 20 x 52, i.e. wide enough for 20 thumbnail images at 52px per thumbnail. We've hard-coded the assumption that there will be 20 thumbnails, for simplicity. In a real application, you wouldn't know how many thumbnails to expect, so you'd have to compute this style programmatically in JavaScript code.
- Set the `height` style to `50px`
  Each thumbnail has a height of 50px (as per the CSS rule for `.thumbnailImage`), so it makes sense for the thumbnails panel to have the same height.
- Set the `position` style to `absolute`
  This allows you to define the absolute position of the thumbnails panel, relative to its first positioned (not `static`) ancestor element, i.e. relative to the main container panel.
- Set the `bottom` style to `0px`
  For absolutely positioned elements, the `bottom` property sets the bottom edge of the element to a unit above/below the bottom edge of its containing element. Setting `bottom` to `0px` aligns the thumbnails panel along the bottom edge of the container panel.
- Set the `margin` and `padding` styles to `0px`
  This ensures there's no unwanted margin or padding on any edge of the thumbnails panel.

In the browser, refresh the *Start* web page. The thumbnails panel should appear correctly:

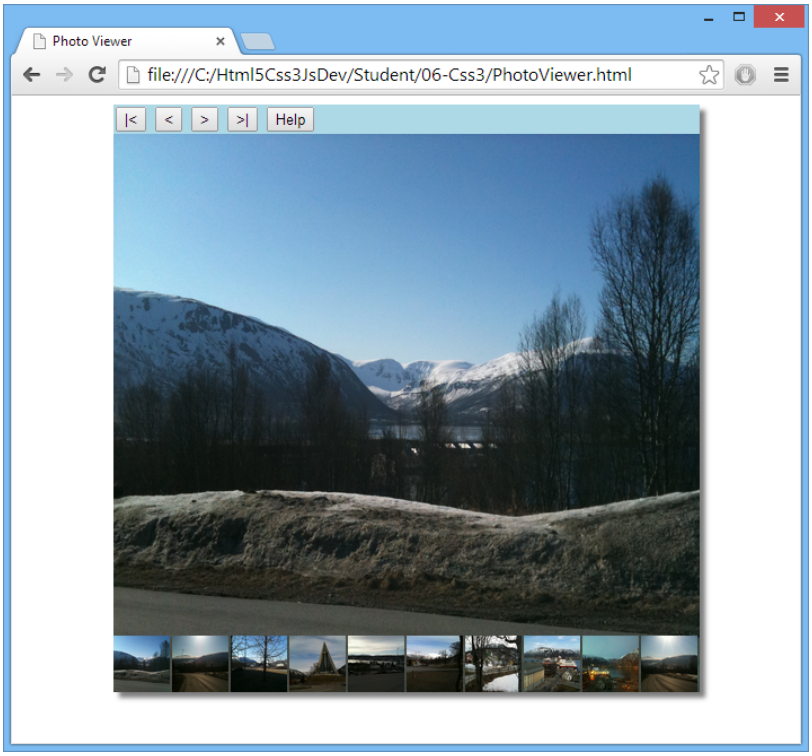


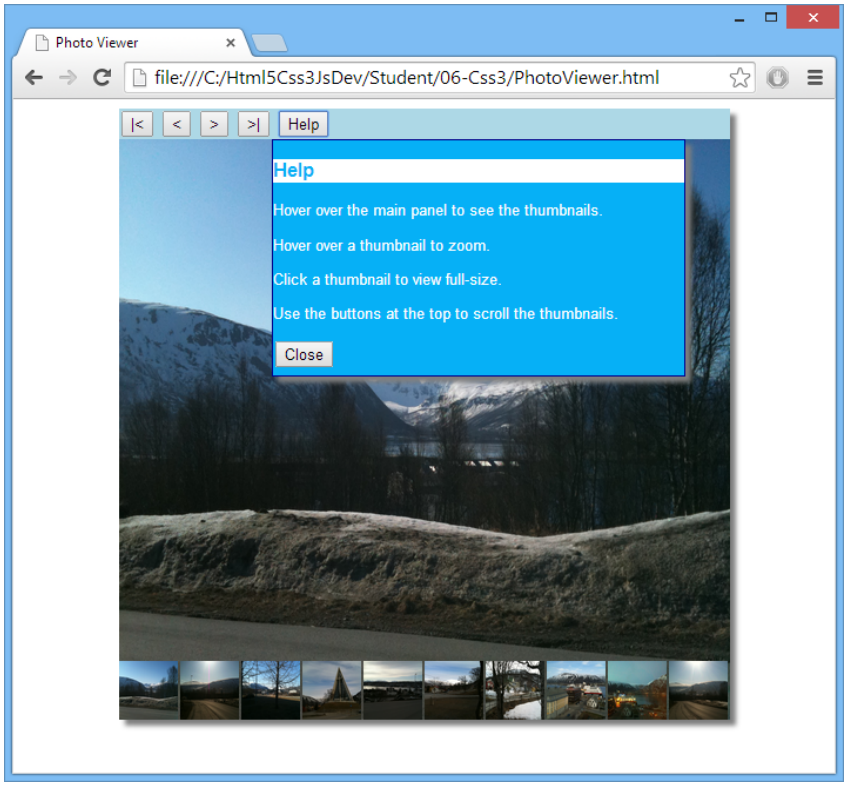
Photo Viewer
file:///C:/Html5Css3JsDev/Student/06-Css3/PhotoViewer.html
|<
<
>
>|
Help

Exercise 4: Defining styles for the help panel

In this exercise, you'll define styles to position the help panel within the main container panel. In the style sheet, locate the TODO 4 comment and add the following styles:

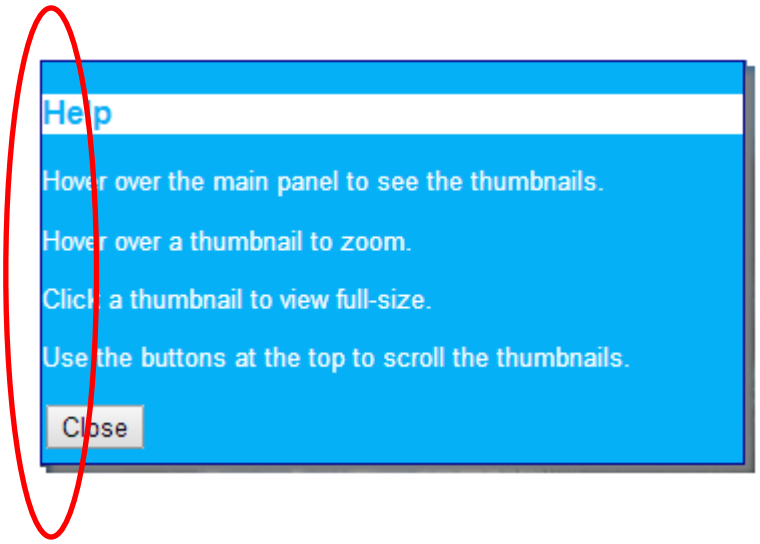
- Set the `width` style to `350px`, and set the `height` style to `200px`
  We chose these values empirically, based on what we thought looked good!
- Set the `position` style to `absolute`
  As per the thumbnails panel in the previous exercise, this allows you to define the absolute position of the help panel within the container panel
- Set the `top` style to `5%`, and set the `left` style to `25%`
  We chose these values empirically, to define a suitable absolute position for the help panel within the container panel.

In the browser, refresh the *Student* web page and click the *Help* button. The help panel should appear with a suitable position and size within the container panel:



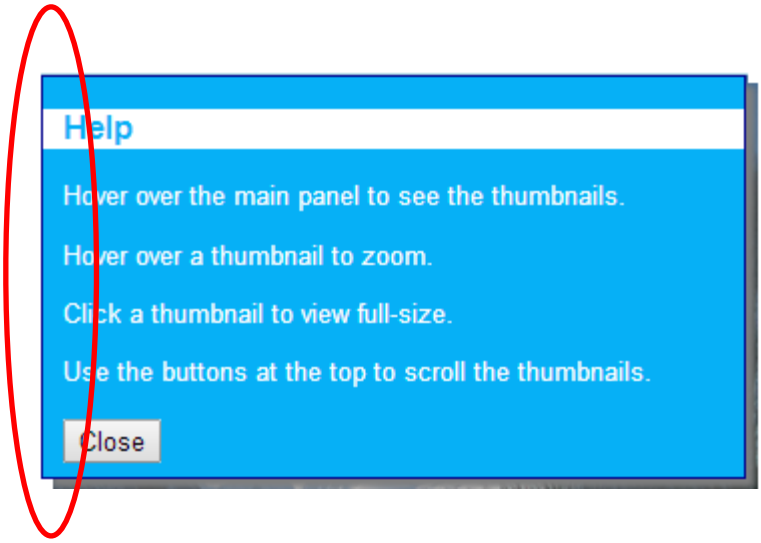
Exercise 5: Defining styles for child elements in the help panel

In the previous exercise, you might have noticed that the help panel contents were squished-up against the left-hand-side edge of the panel, which isn't ideal:



You'll improve the appearance of the help panel in this exercise. In the style sheet, locate the TODO 5 comment, and define a CSS rule that targets `<p>`, `<span>`, and `<button>` elements anywhere in the `#helpPanel`, so that they have a 10px margin.

In the browser, refresh the *Start* web page and click the *Help* button. The help panel contents should now have a margin as follows:



Exercise 6 (If time permits): Additional suggestions

Fine-tune the JavaScript code in `PhotoViewerScript.js`, to define subtle animation effects as indicated by the following TODO comments in the code (TODO 6a through to TODO 6d):

a) When the mouse enters the main container panel, the thumbnails panel currently appears immediately, which is a bit abrupt. Modify the code so that it slides the thumbnails panel into view gradually, rather than showing it immediately.

b) When the mouse leaves the main container panel, the thumbnails panel currently disappears immediately. Modify the code so that it slides the thumbnails panel out of view gradually.

c) When the user clicks the *Help* button, the help panel currently appears immediately. Modify the code so that it shows the help panel gradually. When you test this, you'll notice that the text in the help panel is word-wrapped initially, while the help panel is growing in size, which isn't ideal. To prevent word-wrapping in the help panel, set its `white-space` style to `nowrap`.

d) When the user closes the help panel, it currently disappears immediately. Modify the code so that it hides the help panel gradually.