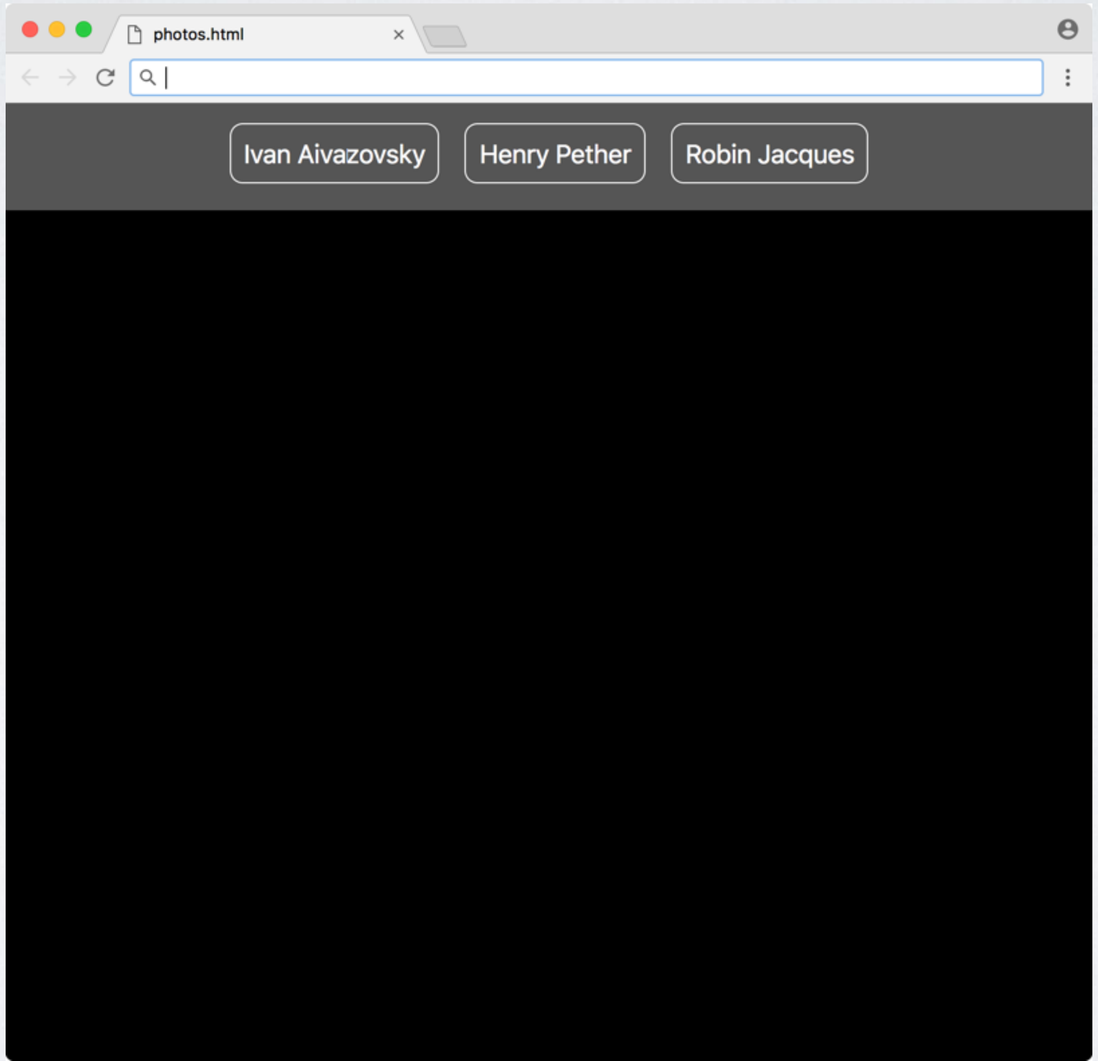
ASSIGNMENT 2

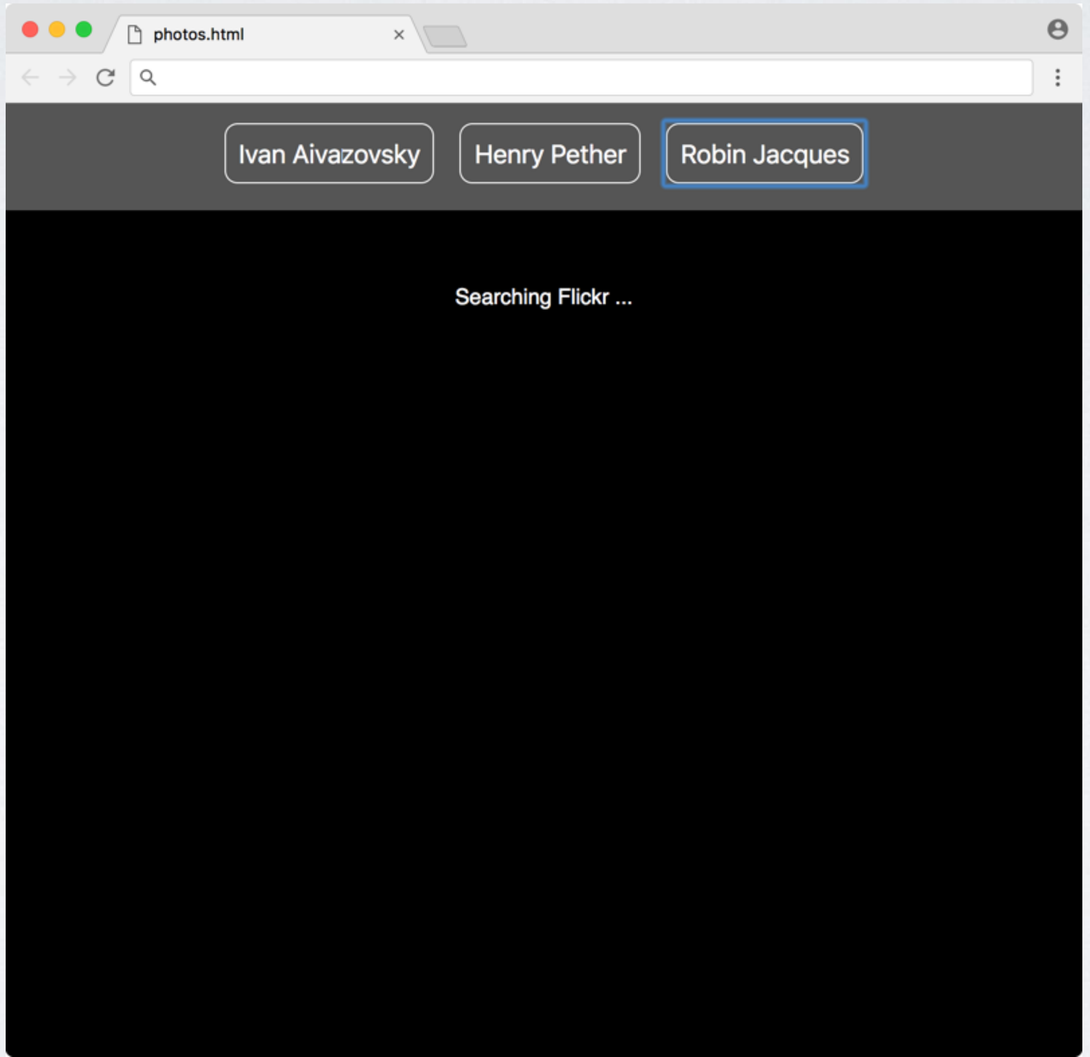
Interactive Client-side Web Development
2018

You must write the HTML, CSS and JavaScript to implement the following web page.

You must create a page with three buttons/links corresponding to search terms.



When you click on a button your program should search Flickr (via their API using JSON-P) with those keywords. The user should be informed that the search has started.

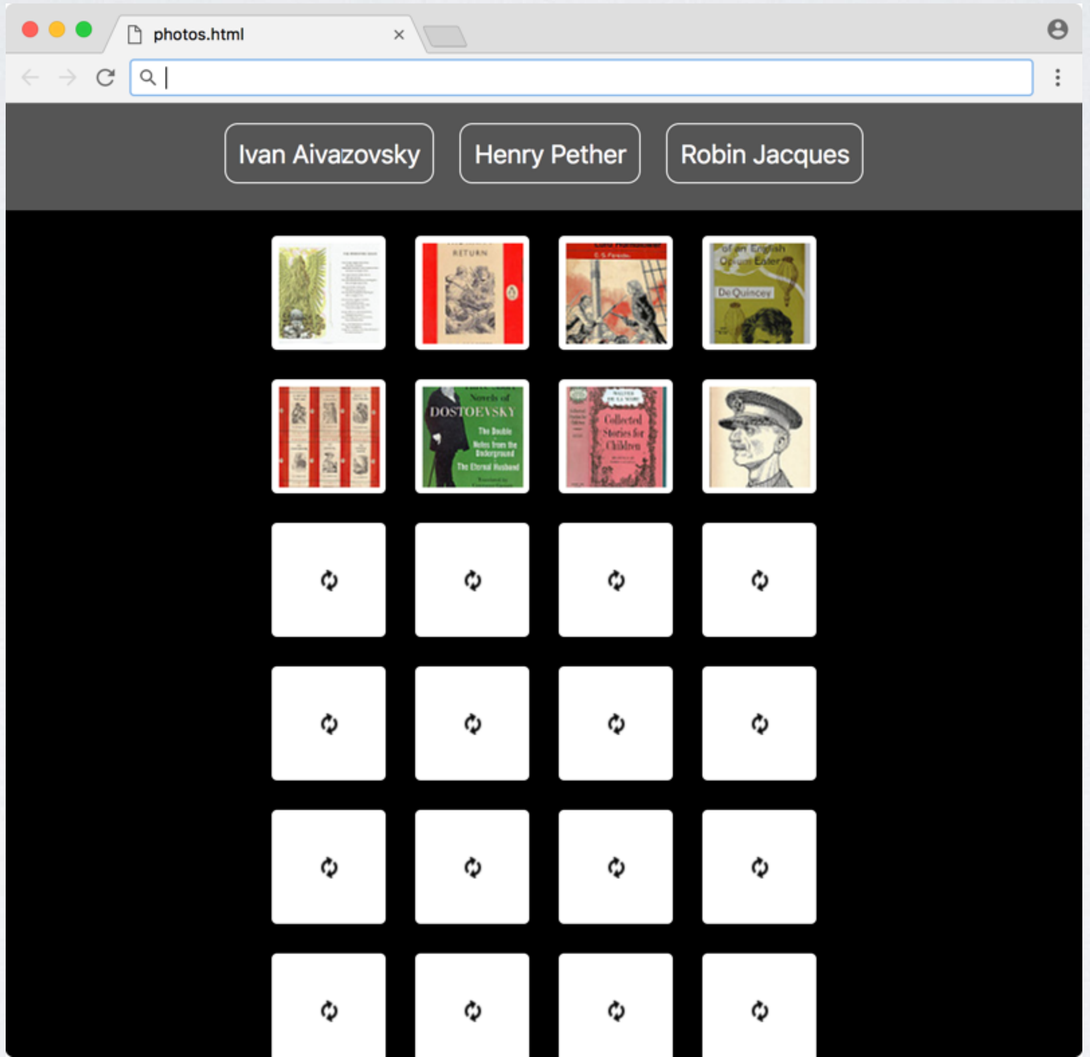


When the data comes back from Flickr you should create the HTML elements necessary to display the incoming images via **DOM Scripting**.

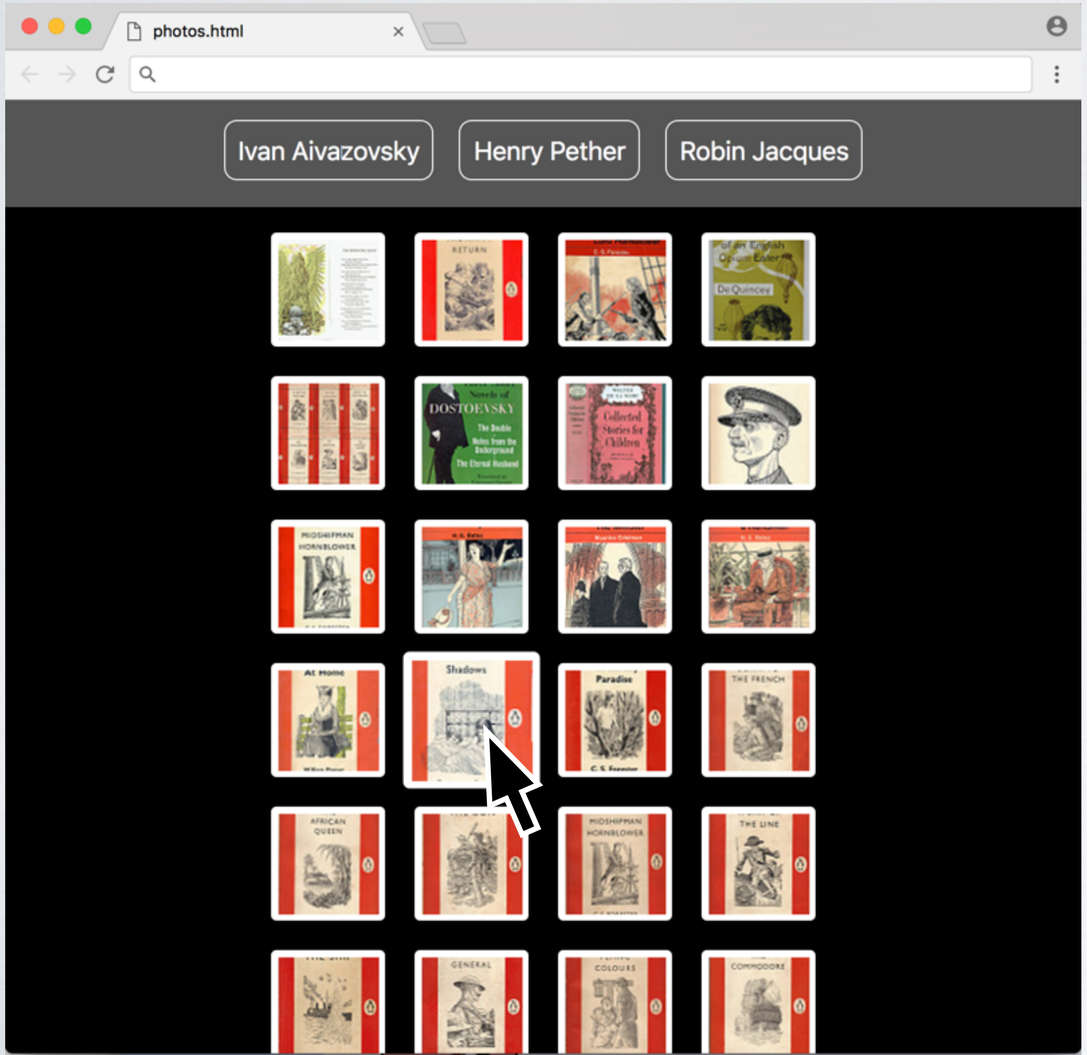
I.e. loop through the photo data that came back creating & <img> elements and adding them to the page.

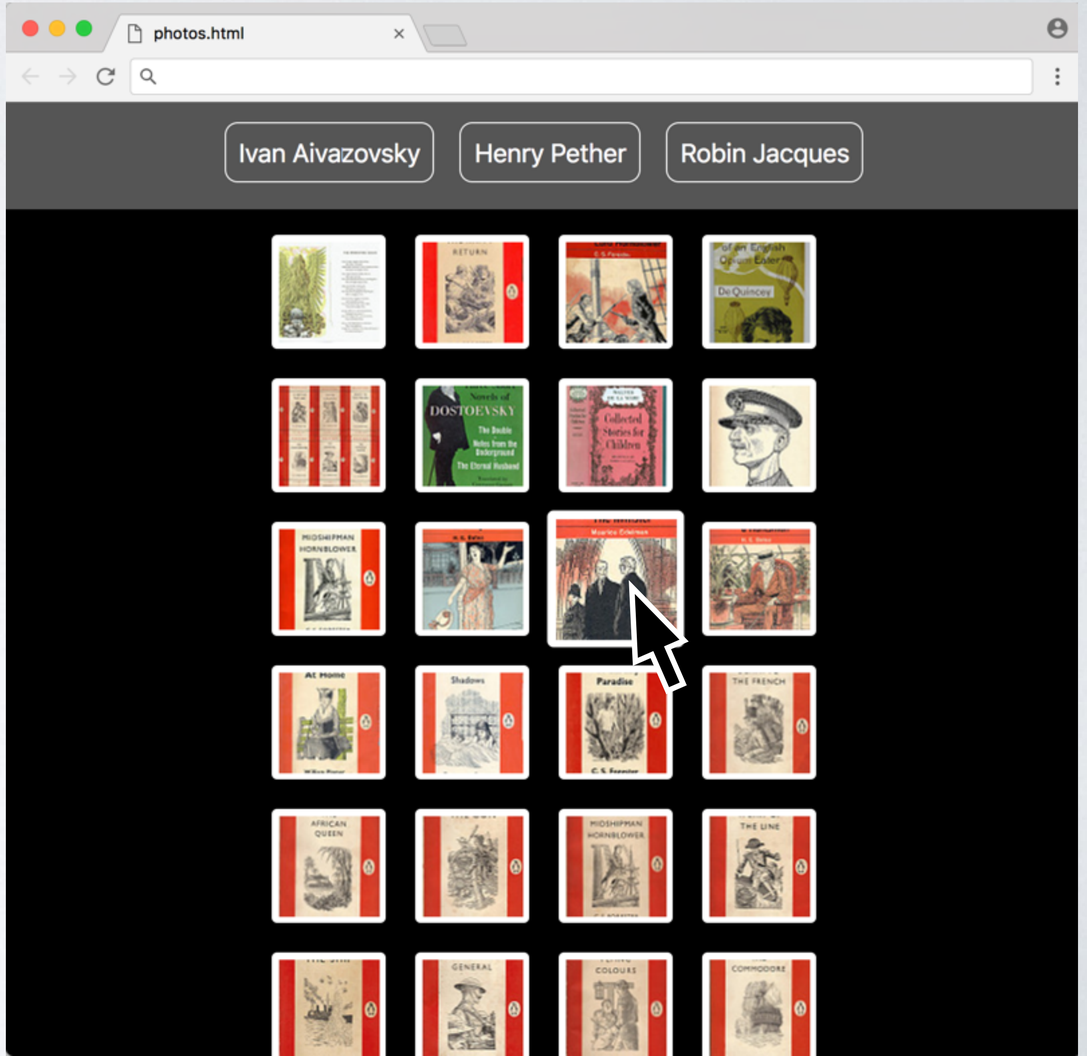
Initially you should show the thumbnail sized images.

However, you should show a loader gif until the image has fully downloaded.

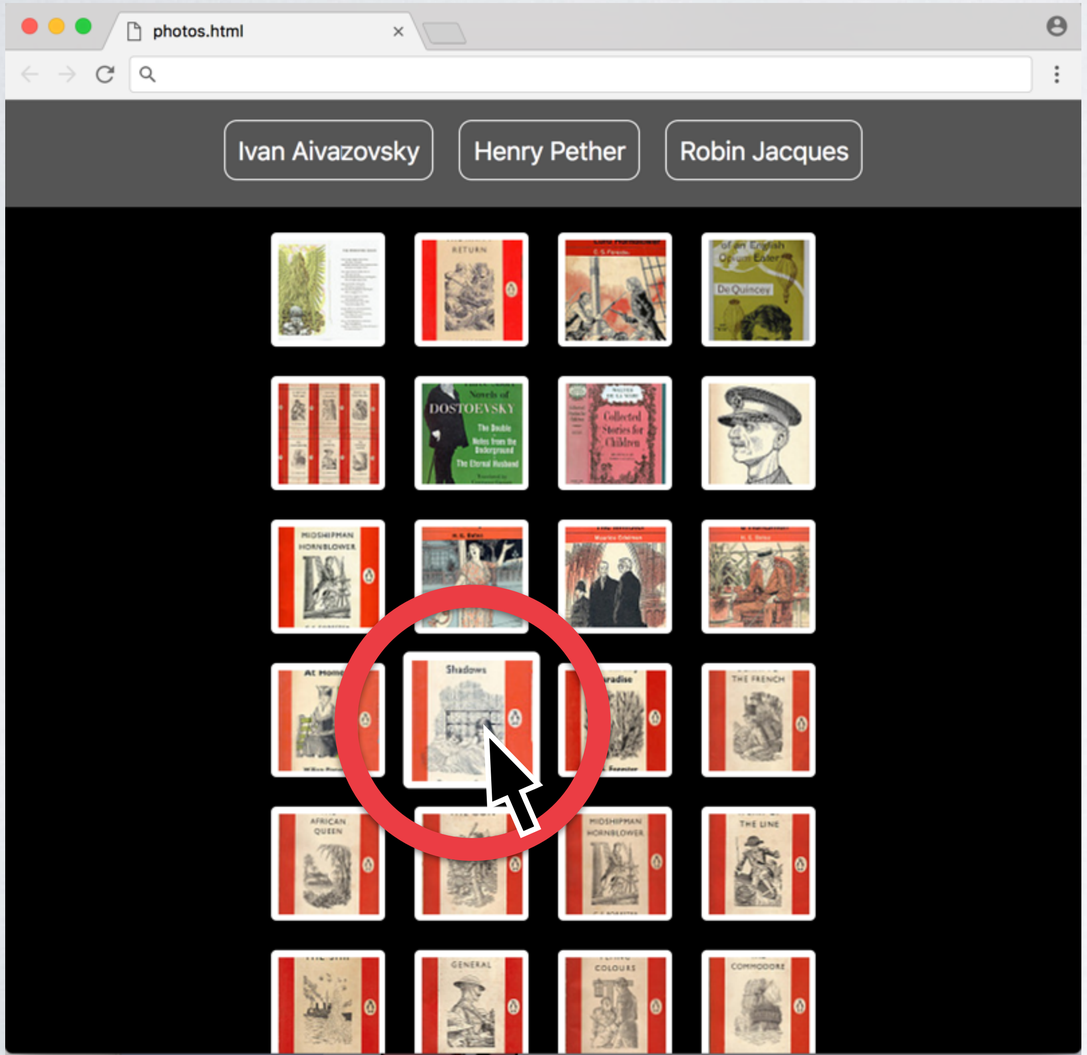


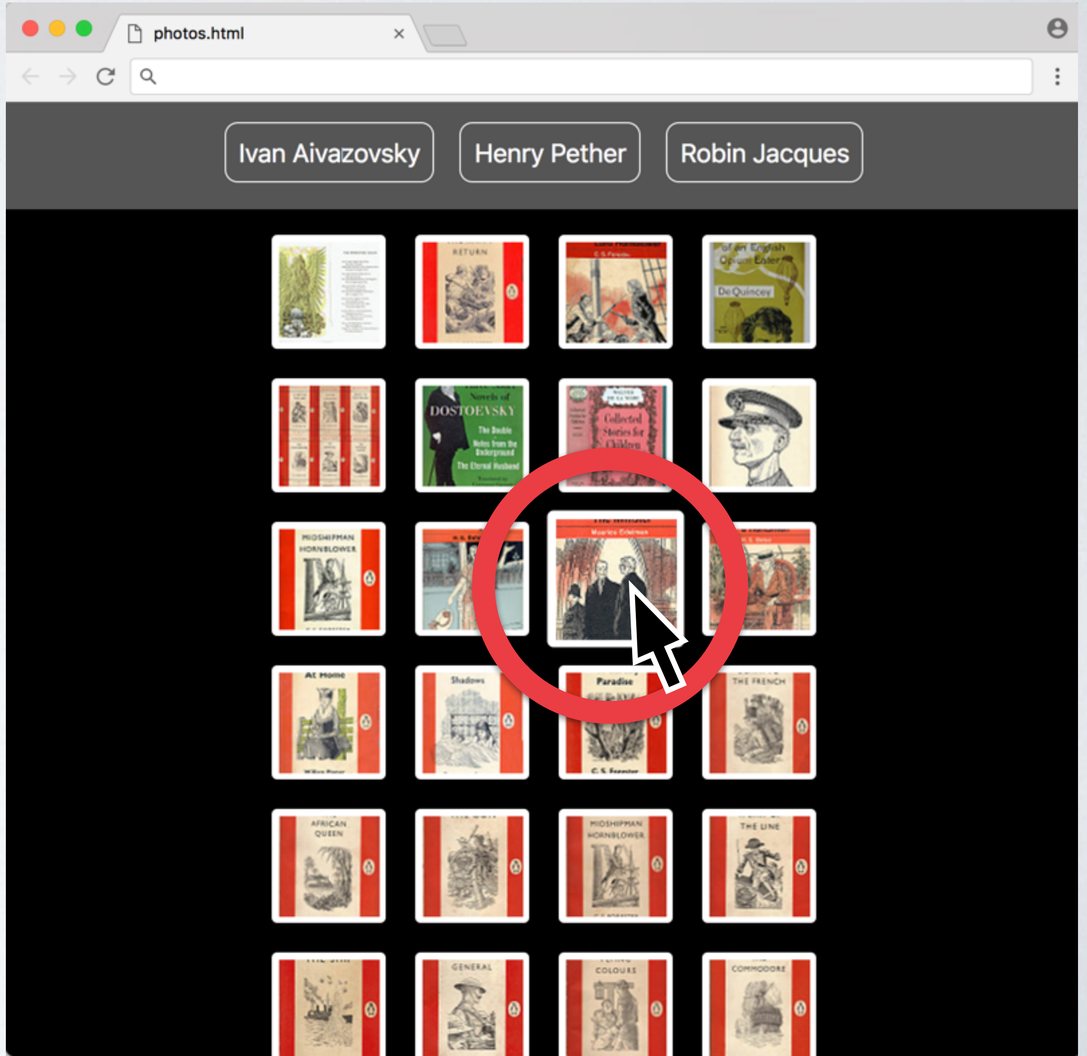
Hovering over a thumbnail should enlarge it slightly (and that should be animated). All this can be done via CSS.



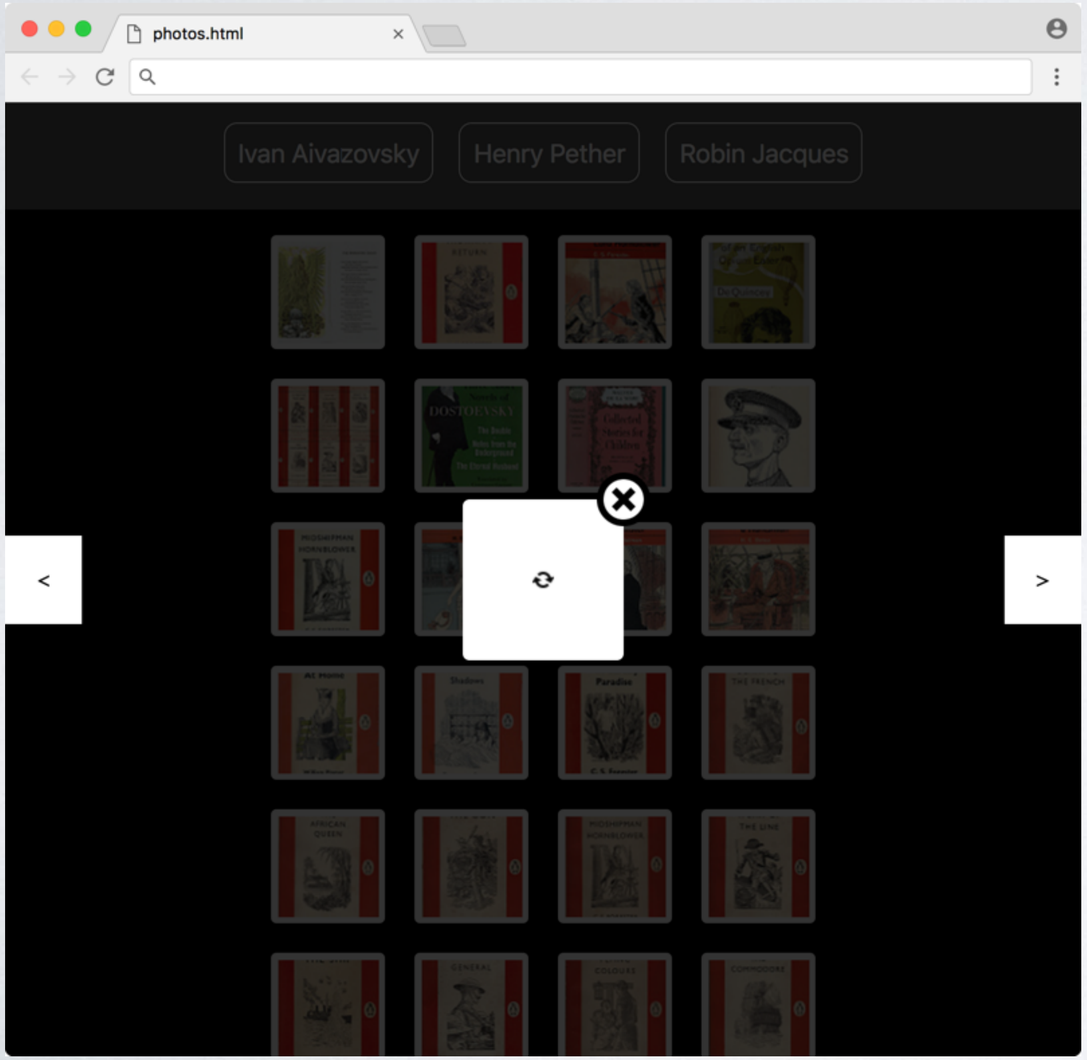


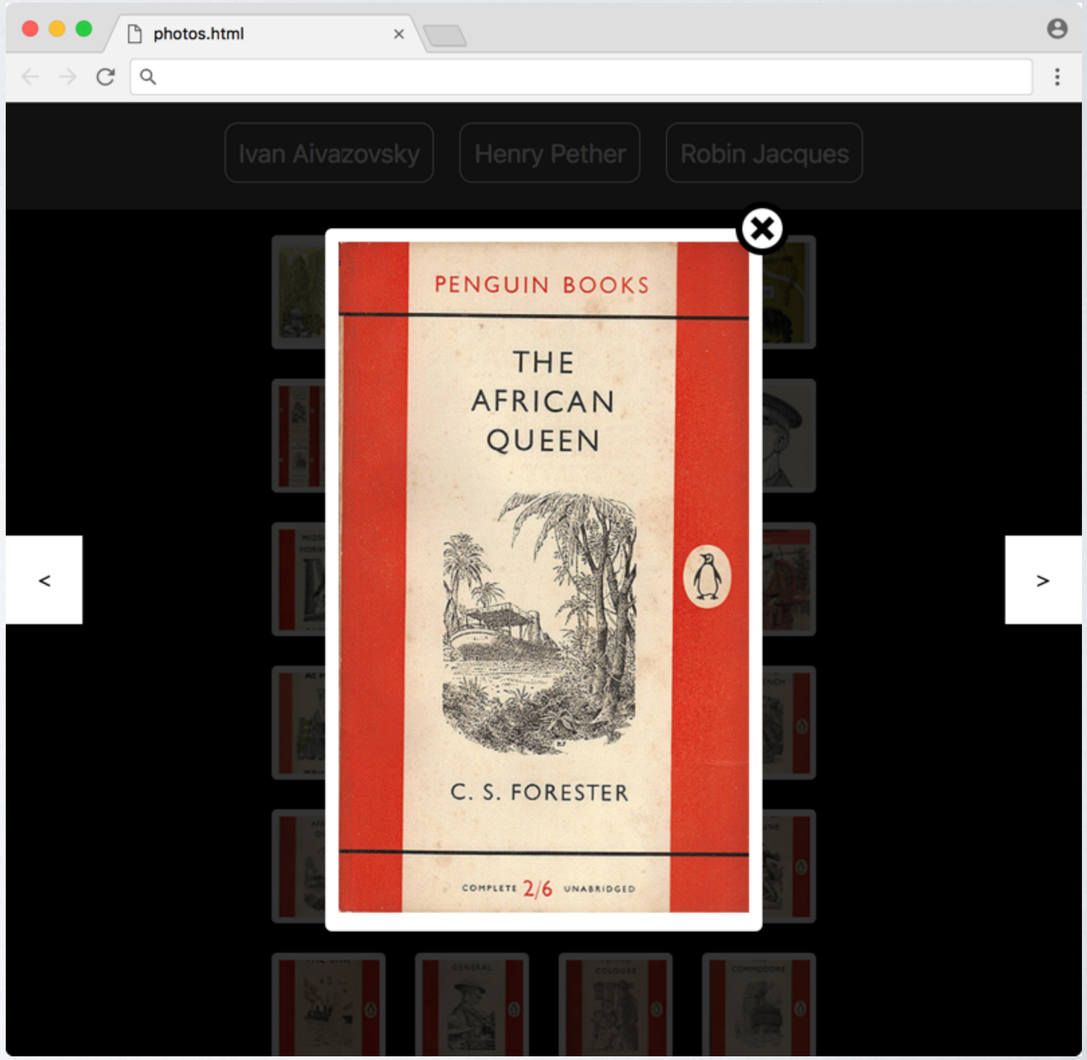
Hovering over a thumbnail should enlarge it slightly (and that should be animated). All this can be done via CSS.

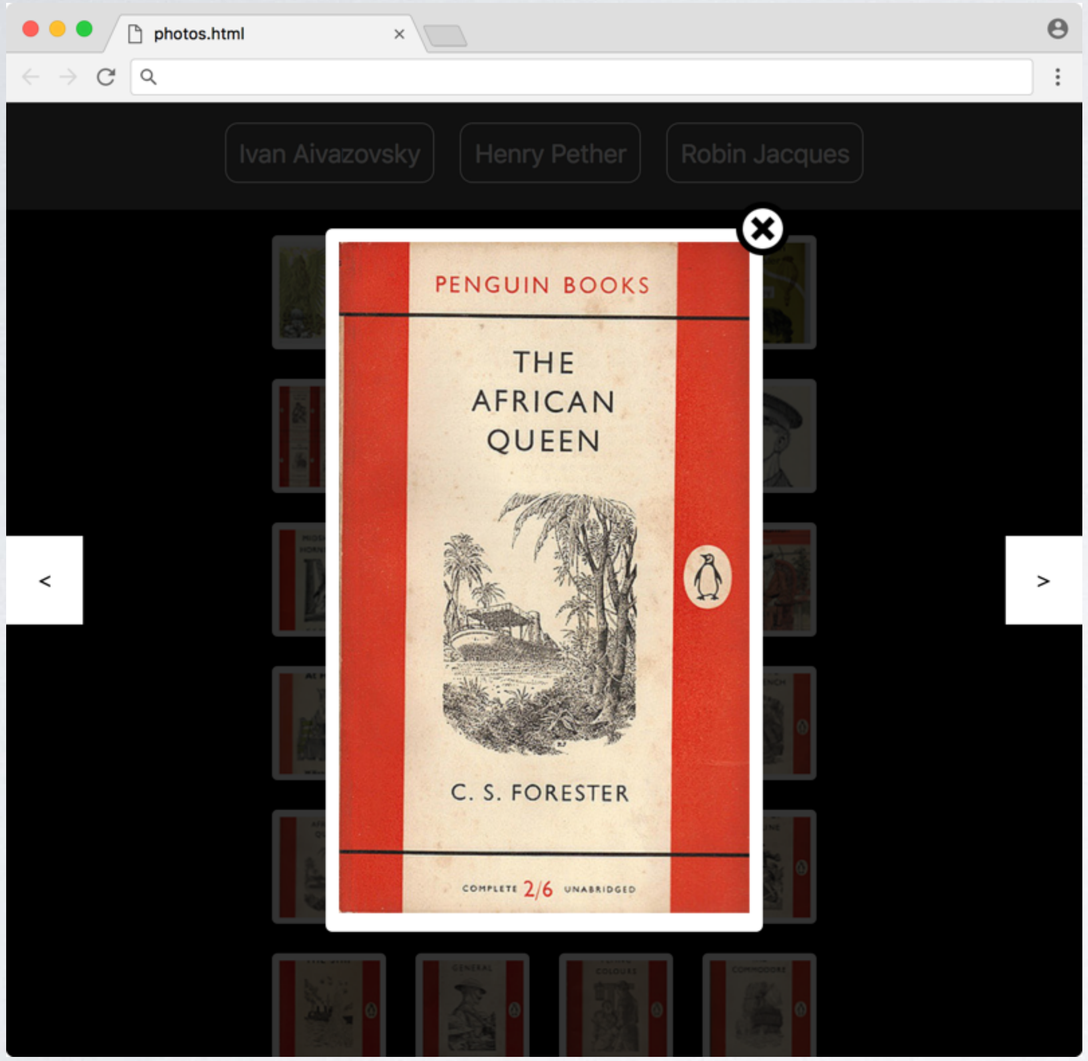




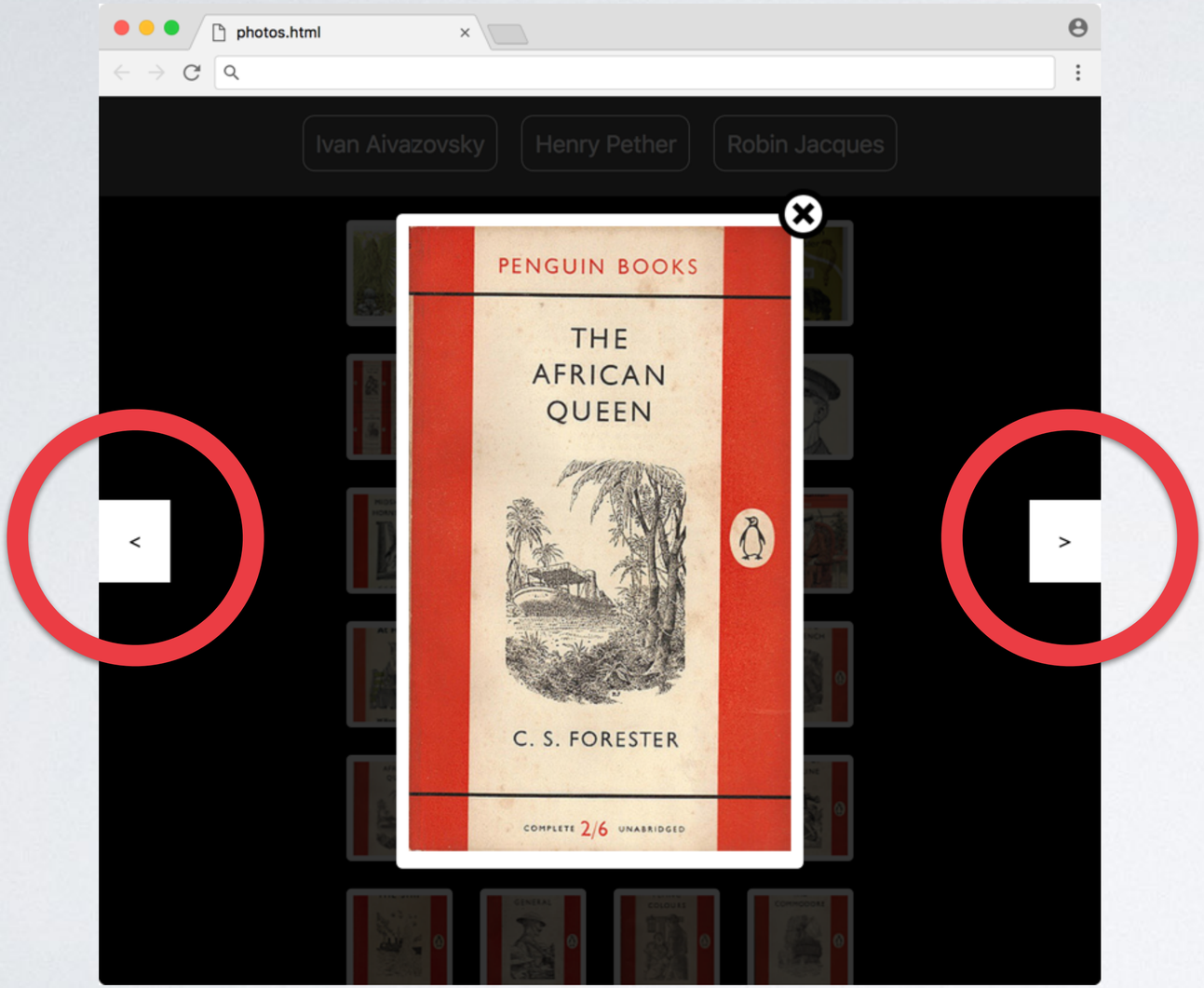
Clicking on a thumbnail should show the full size version. As with the thumbnails you should show a loader gif until the full image has downloaded.





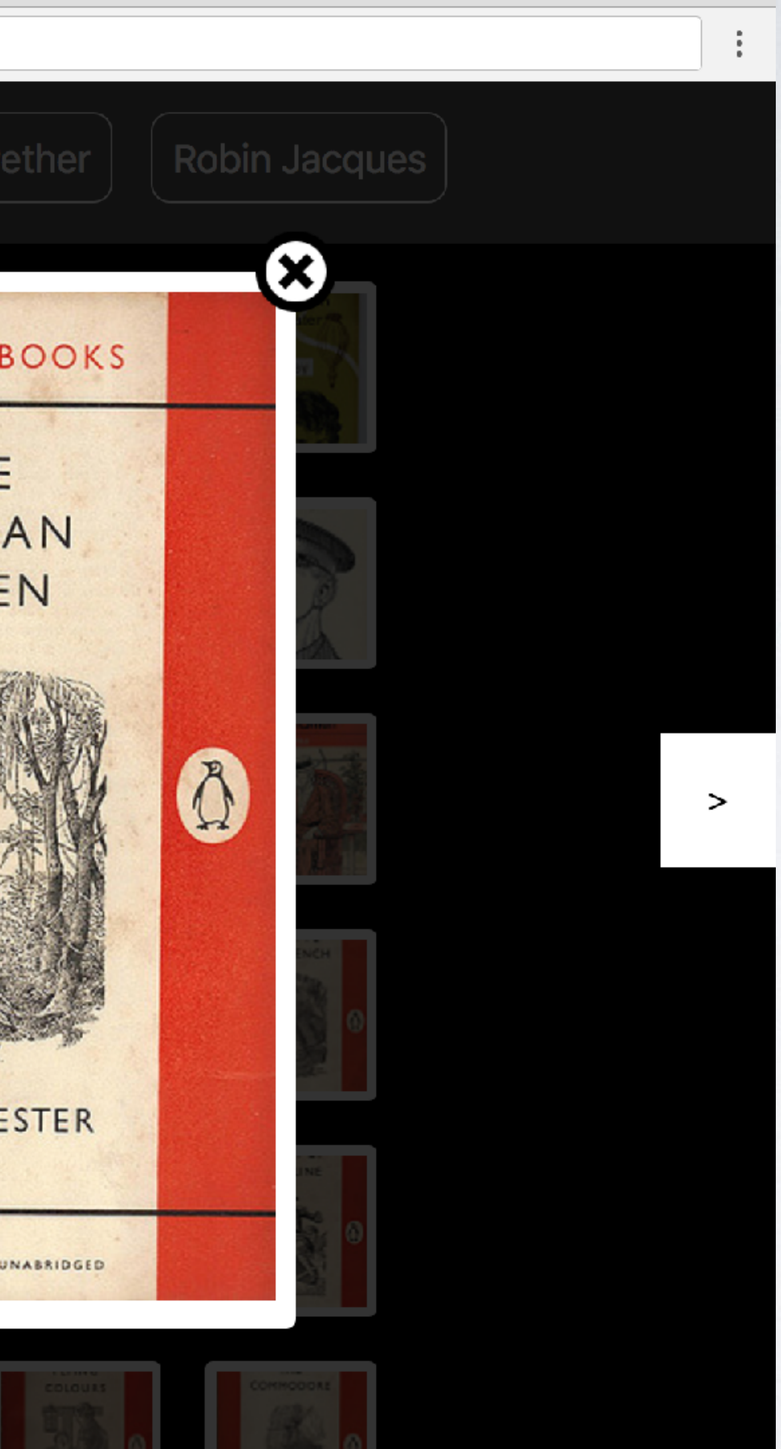


When showing an image you should show buttons/links to show the **next** and/or **previous** image without going back to the thumbnails.

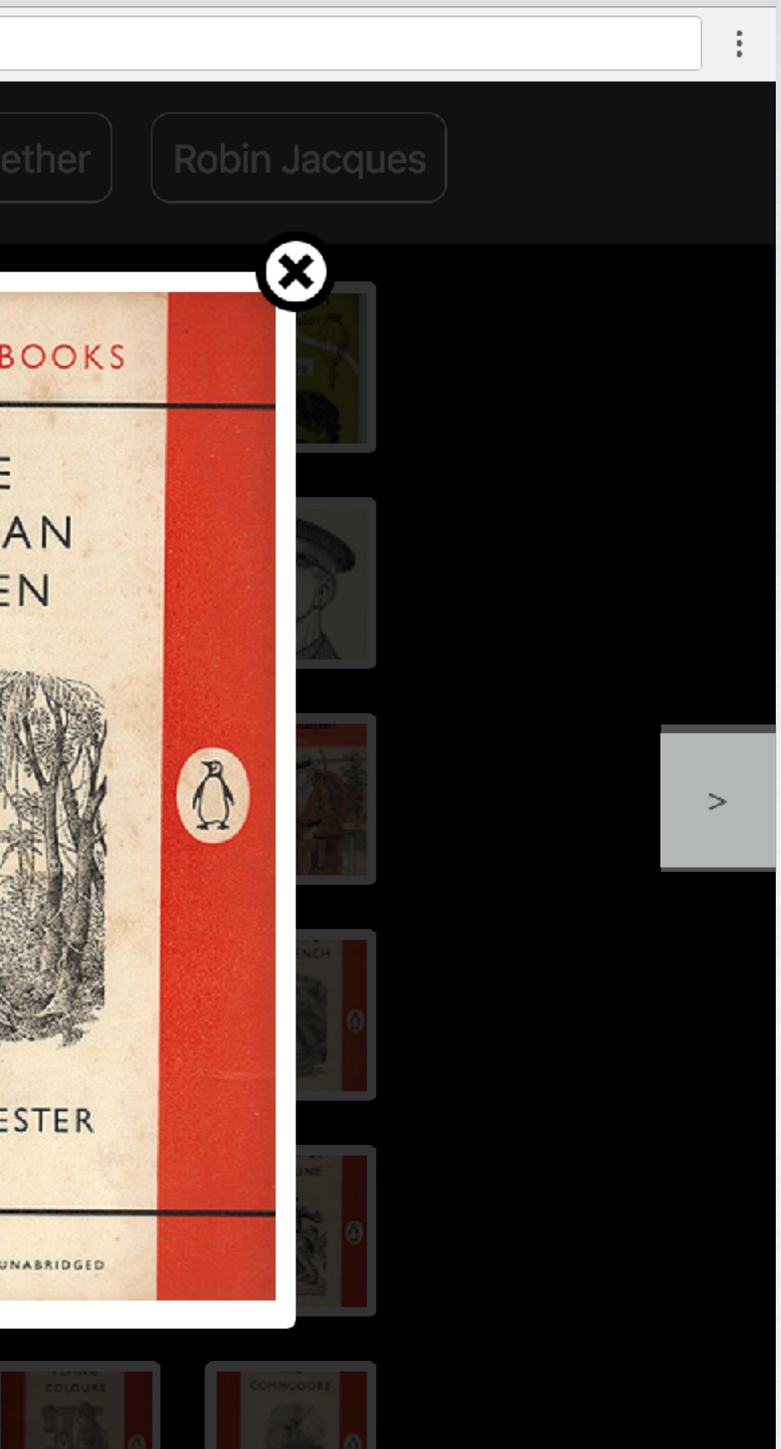


When showing an image you should show buttons/links to show the **next** and/or **previous** image without going back to the thumbnails.

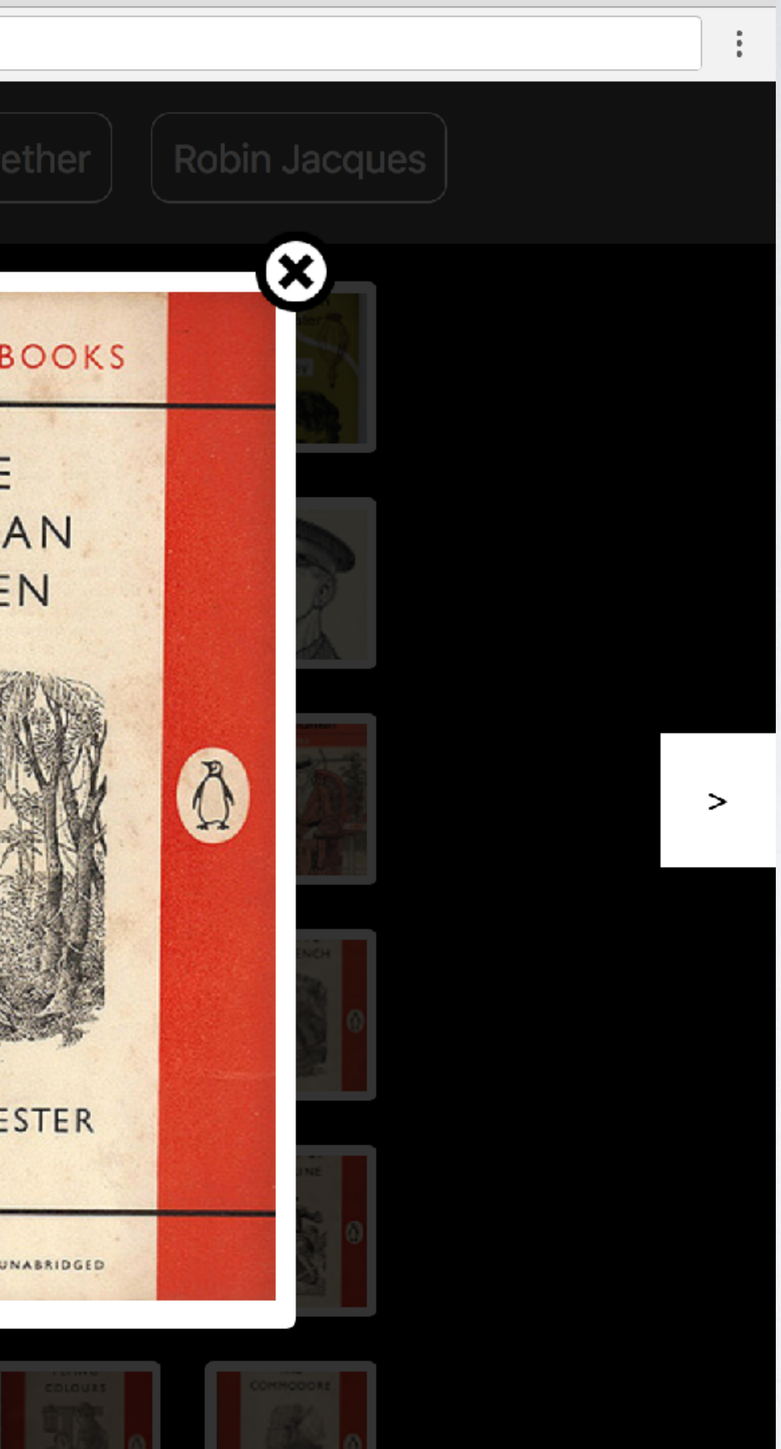
Pressing the **right arrow** key should also show the **next photo**, whereas pressing the **left arrow** key should show the **previous photo**.



When the **next/previous** buttons (or the corresponding keys) are pressed, you should **briefly change the colour** of the button.

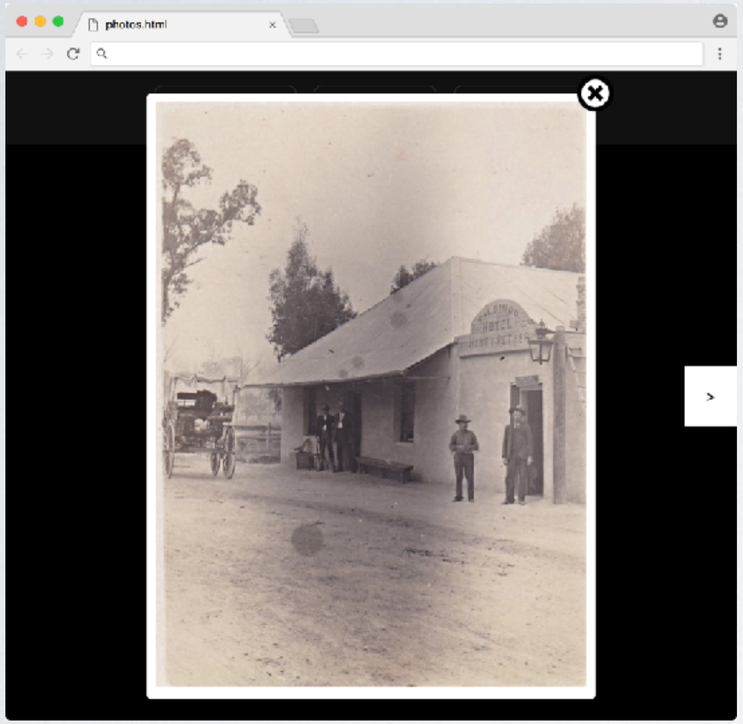


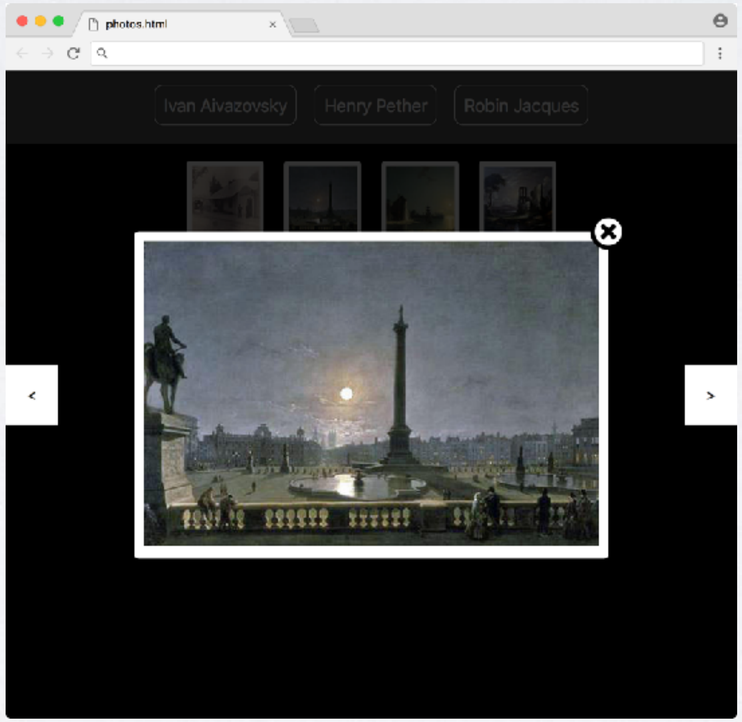
When the **next/previous** buttons (or the corresponding keys) are pressed, you should **briefly change the colour** of the button.

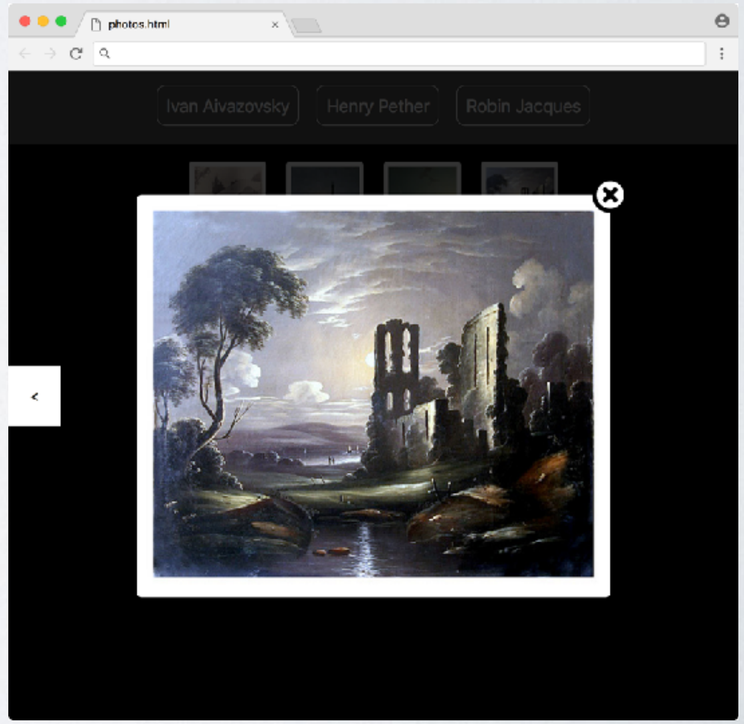


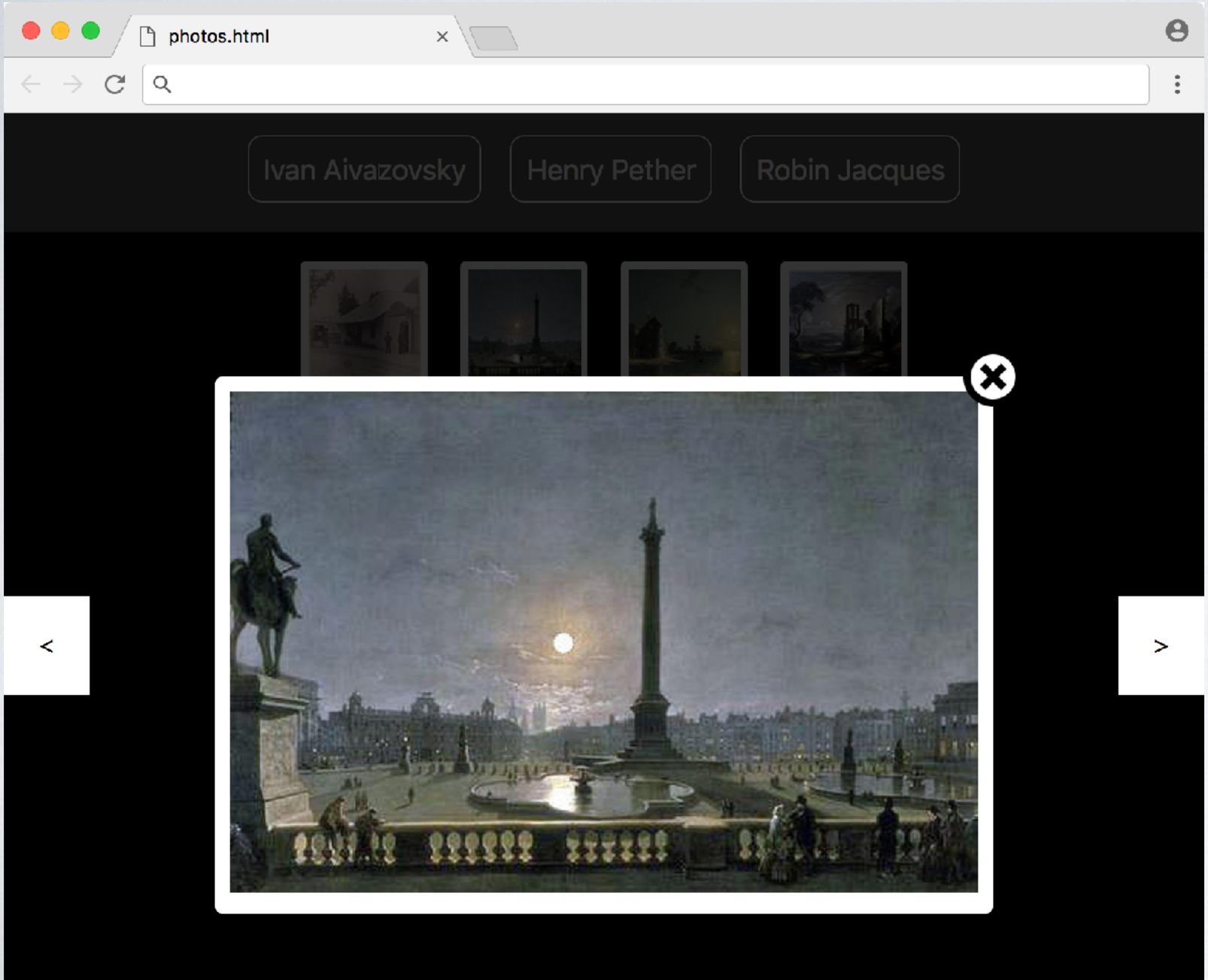
When the **next/previous** buttons (or the corresponding keys) are pressed, you should **briefly change the colour** of the button.

If showing the first image in a collection then you should hide the **previous** button. Similarly when showing the last image you should hide the **next** button.







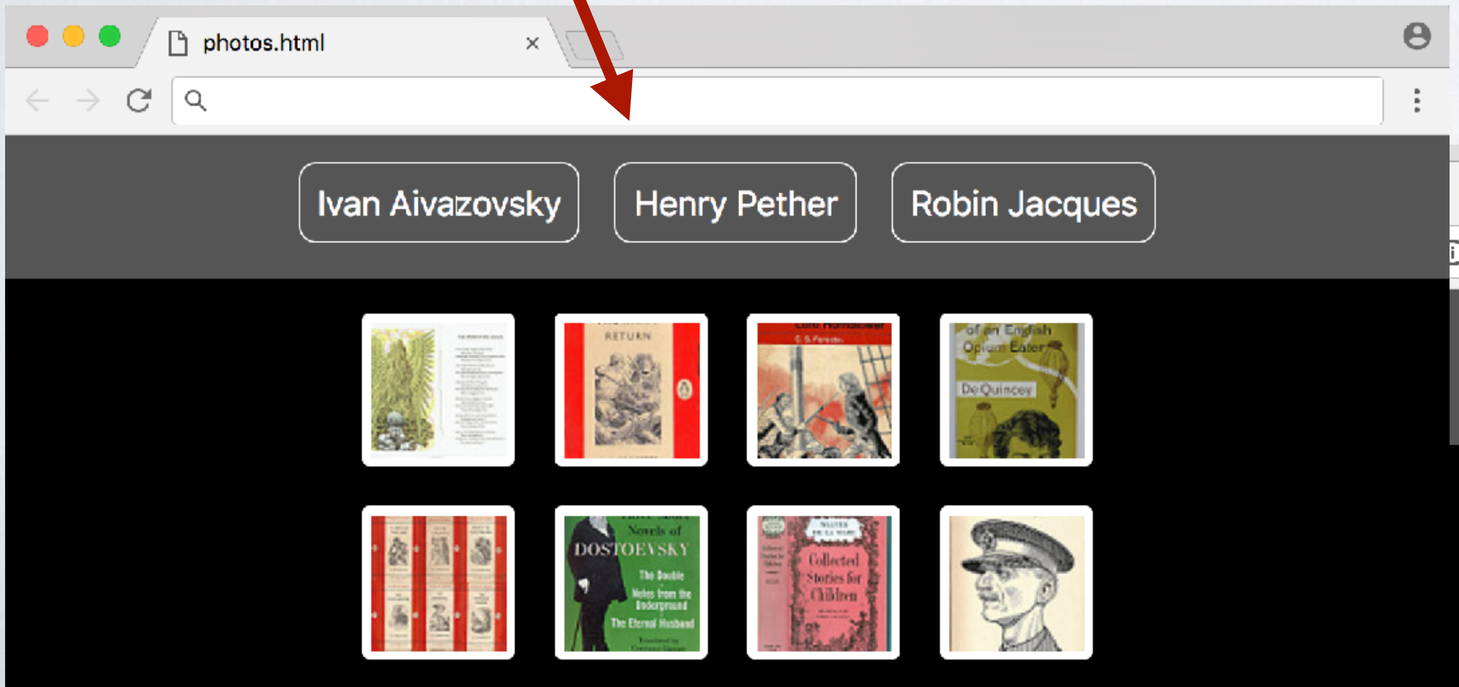


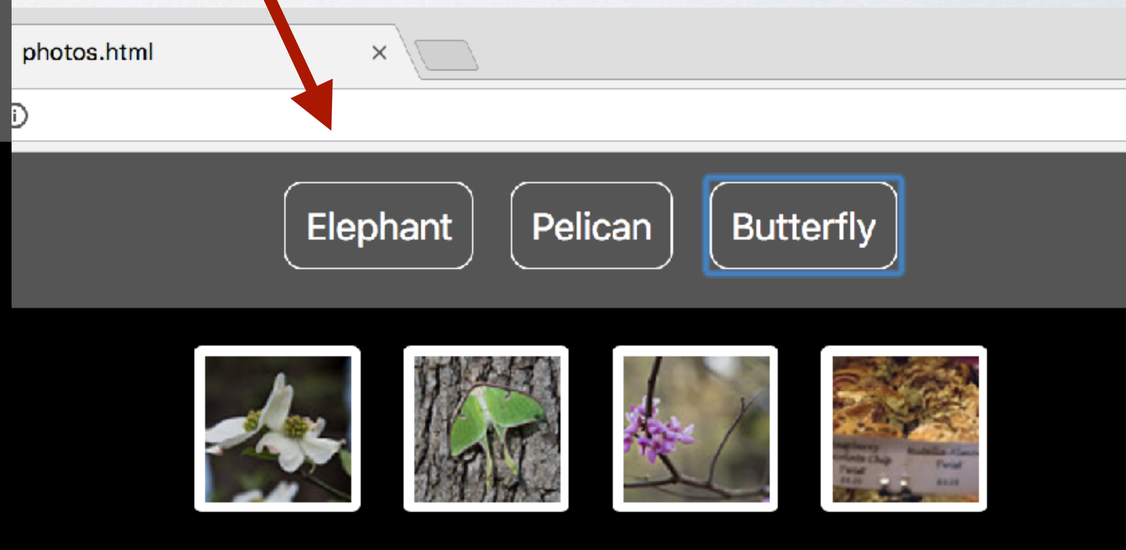
Clicking on the close box or pressing any key other than the left and right arrows should dismiss/ hide the large image.

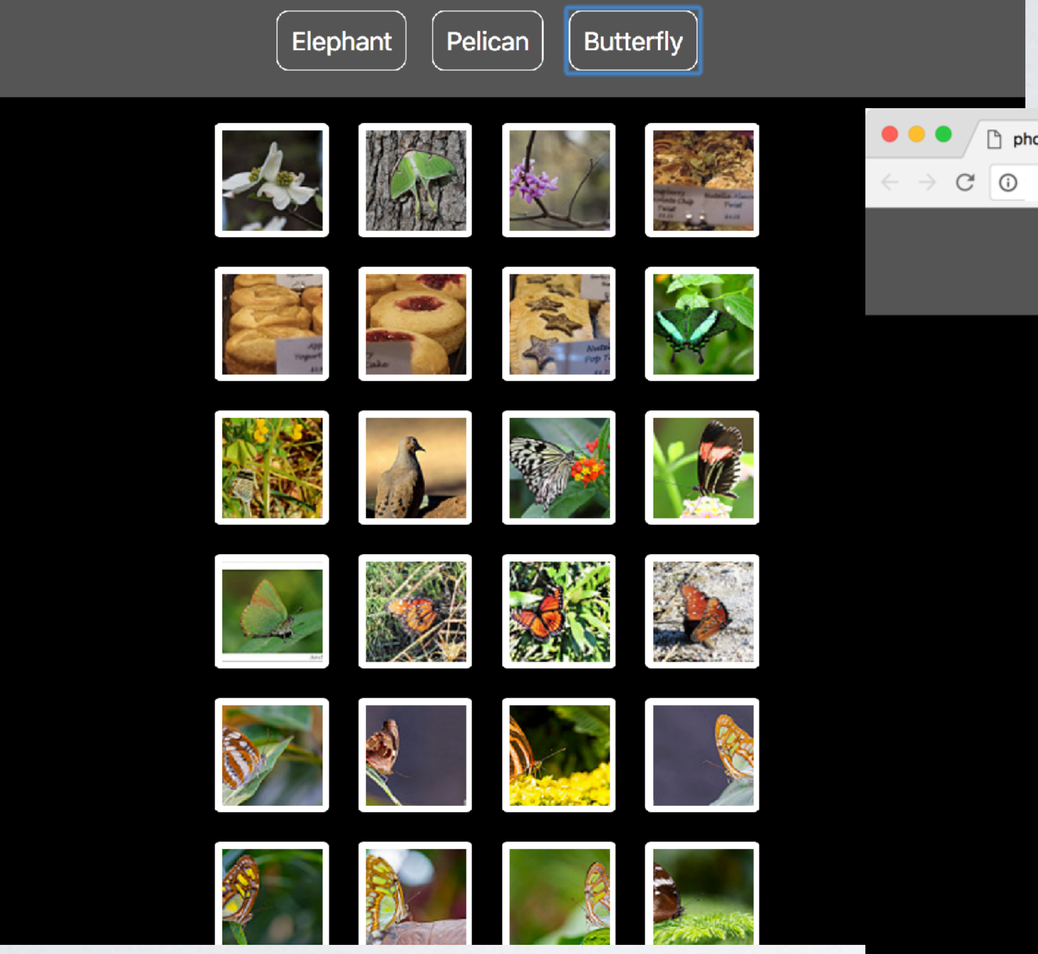
The search terms should be stored in an array, and the search buttons should be automatically generated from that array.

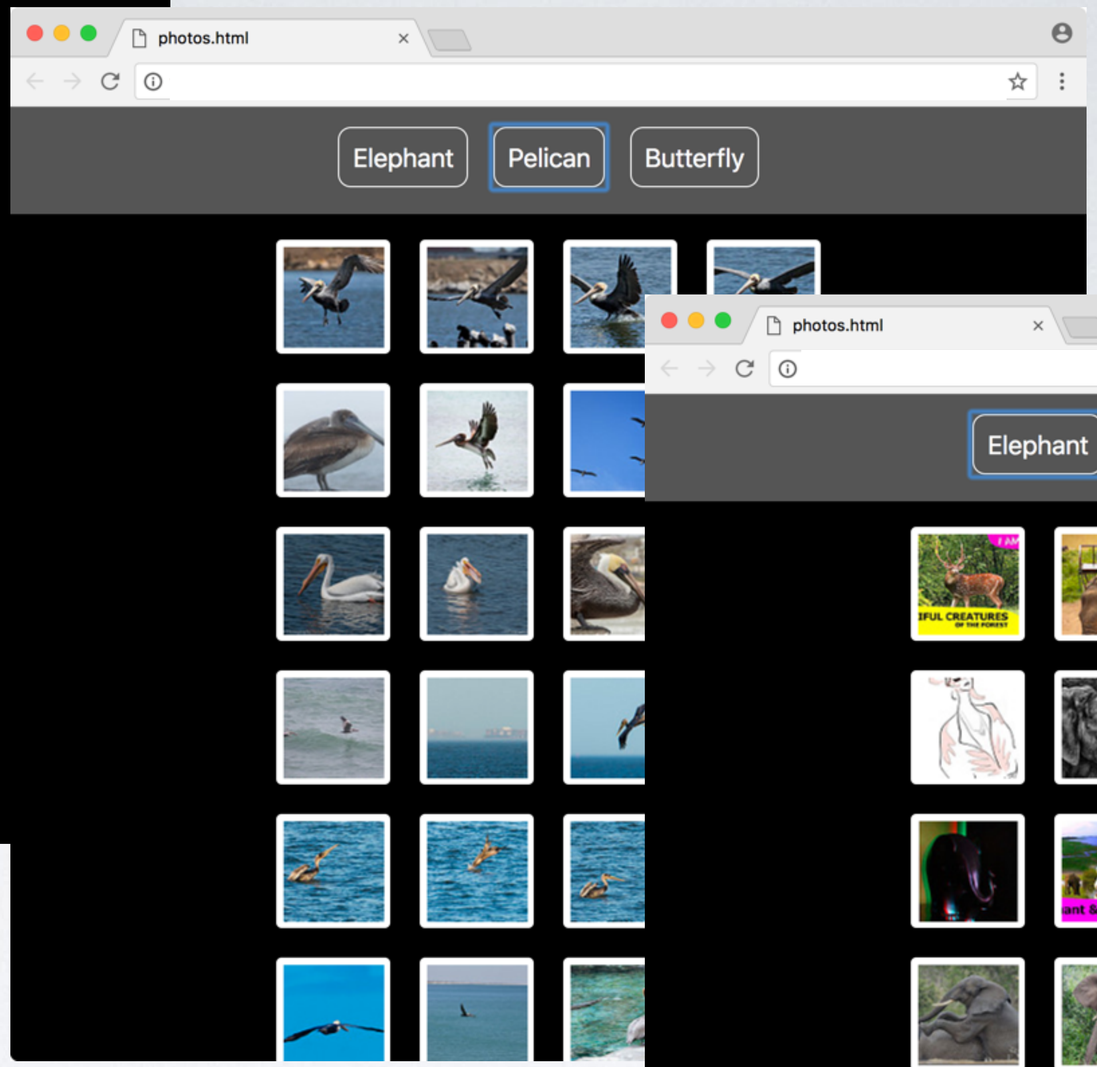
var searchTerms = ["Ivan Aivazovsky", "Henry Pether", "Robin Jacques"];

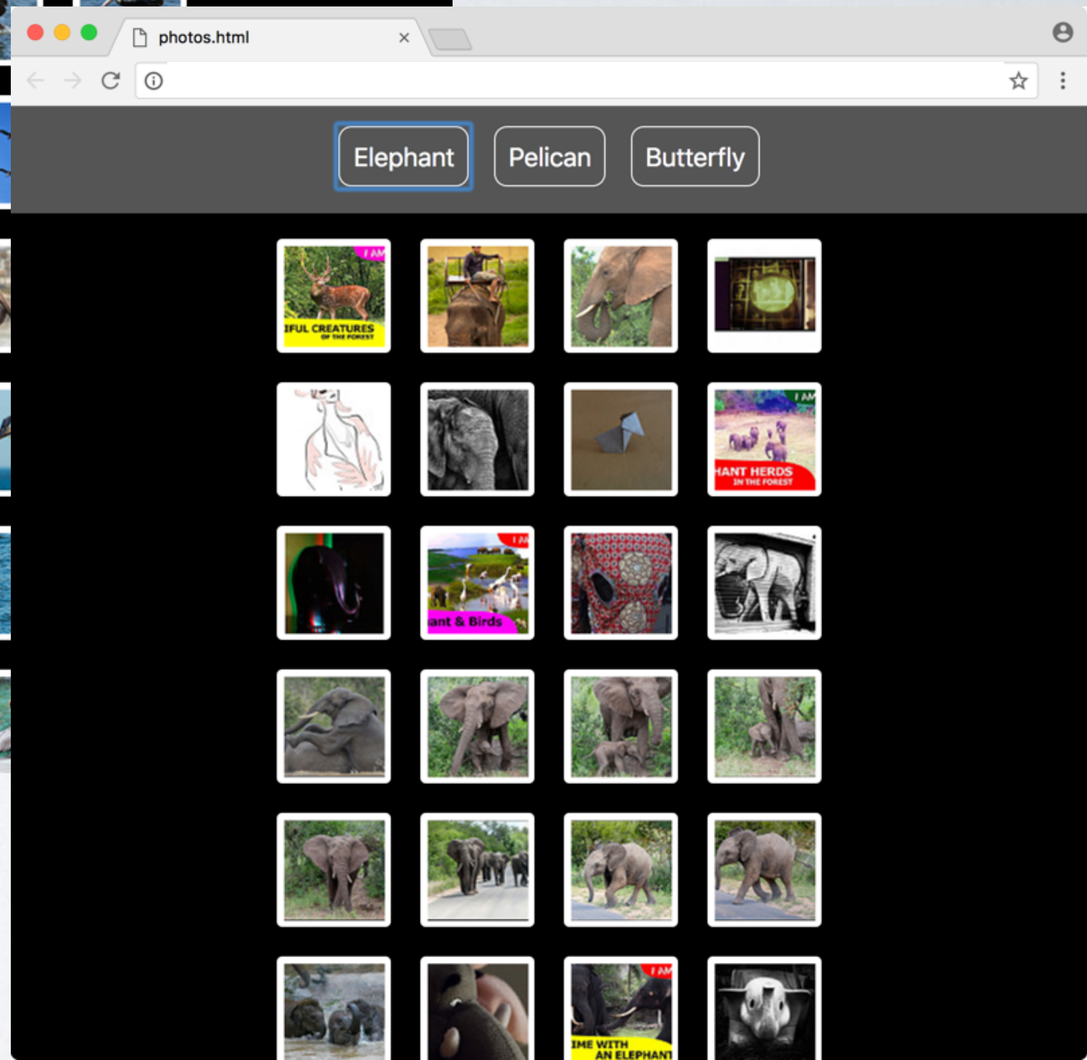
var searchTerms = ["Elephant", "Pelican", "Butterfly"];






Elephant
Pelican
Butterfly


photos.html
Elephant
Pelican
Butterfly


photos.html
Elephant
Pelican
Butterfly

When you start a search you should store the search term in localStorage.

Whenever you load the page you should check localStorage for a search term, and if present, you should start that search automatically.

You should search for a maximum of a hundred images.

Technical Tips

Loader Gifs

Loader gifs can be found here:

https://loading.io

or

http://www.ajaxload.info

Waiting for Images to Load

You can attach **onload** events to **Image** objects.

Code to display final image goes here.

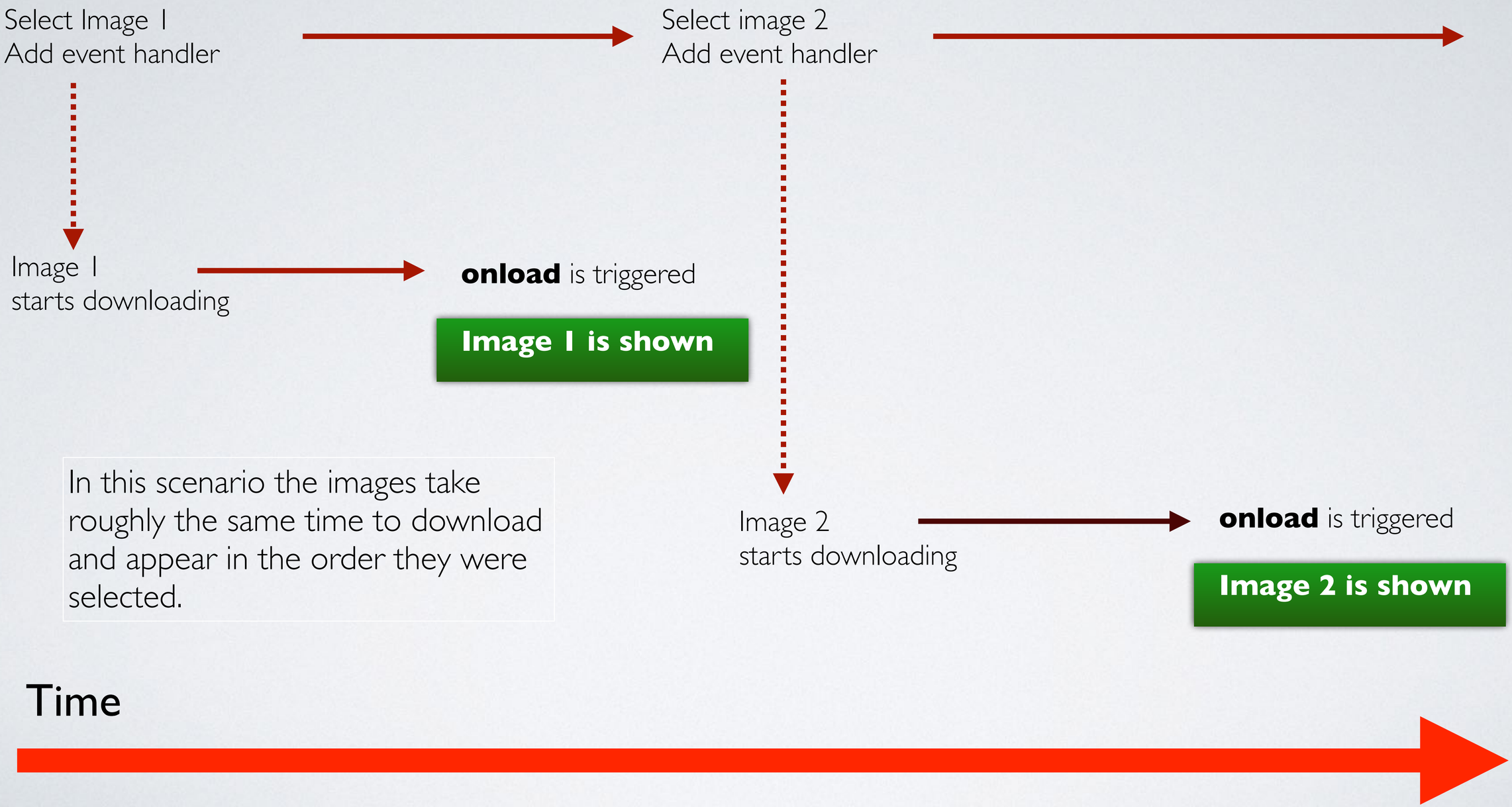
```
var imgObj = new Image();

imgObj.onload = function() { ......};

imgObj.src = url;
```

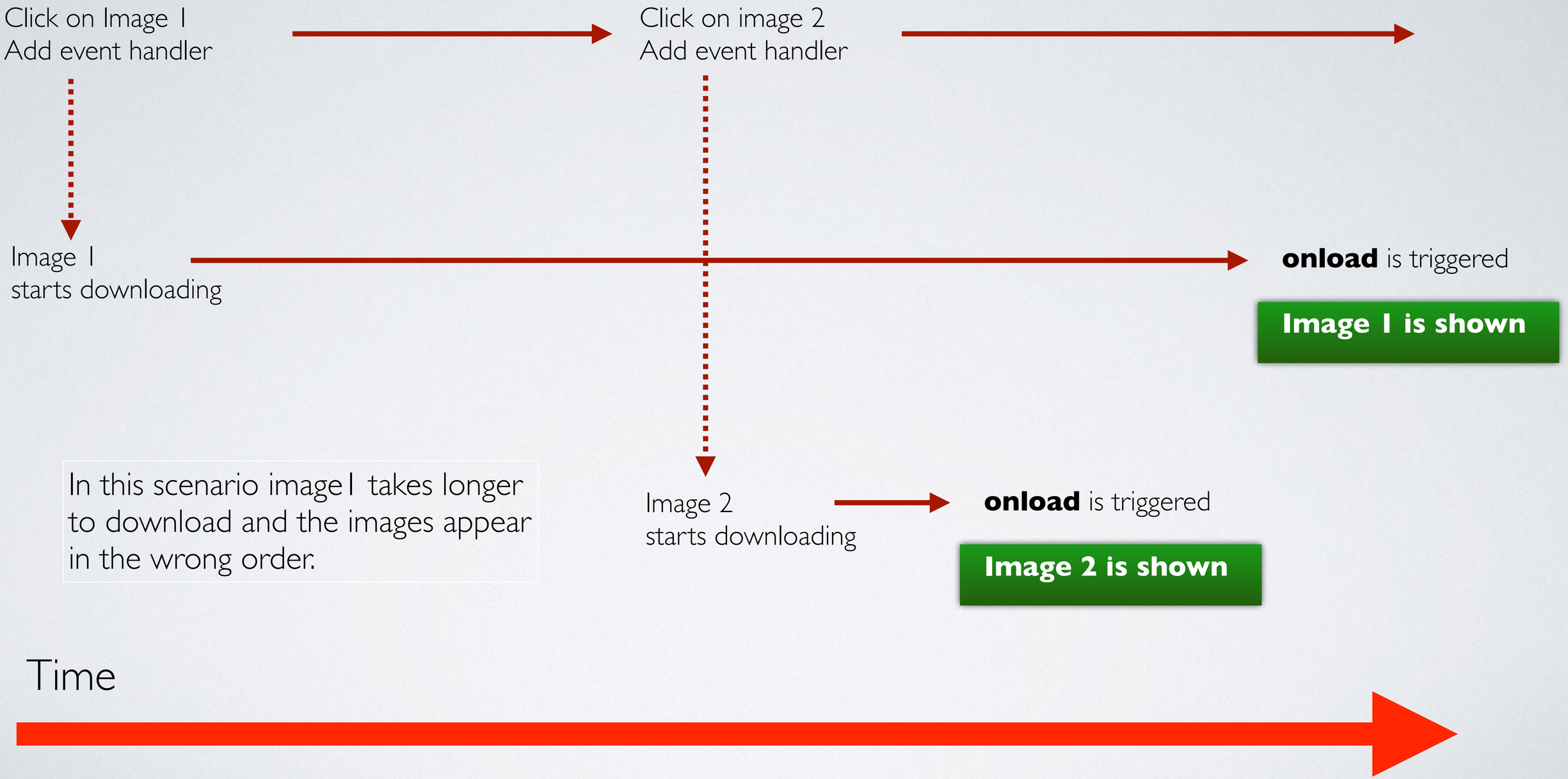
URL of image goes here.

Problems With onload Events

When showing a large image you start to download the larger version of the image. You add an event handler to only show the image once it has fully downloaded.

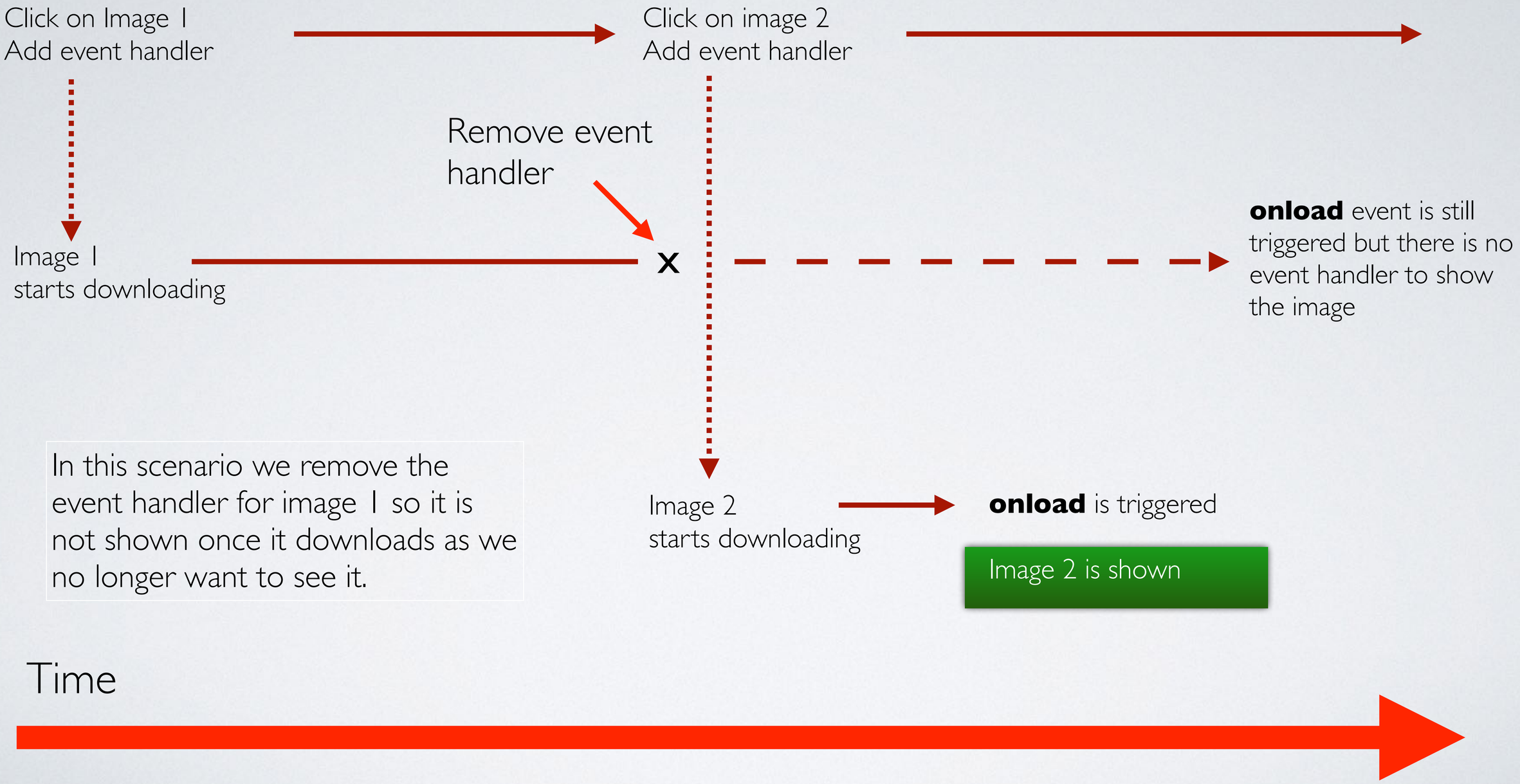
However, if you are rapidly swapping between images (using the **next** and **previous** buttons) you need to remember that **onload** events will not necessarily be triggered in the order you added them and the images may appear in a seemingly random order.


Select Image 1
Add event handler
Select image 2
Add event handler
Image 1
starts downloading
onload is triggered
Image 1 is shown
In this scenario the images take roughly the same time to download and appear in the order they were selected.
Image 2
starts downloading
onload is triggered
Image 2 is shown
Time


Click on Image 1
Add event handler
Click on image 2
Add event handler
Image 1
starts downloading
onload is triggered
Image 1 is shown
In this scenario image1 takes longer to download and the images appear in the wrong order.
Image 2
starts downloading
onload is triggered
Image 2 is shown
Time

A solution is to remove the **onload** event handler for the previous image when we select a new image to view.

Then we add an **onload** event handler only to the newly selected image. This way the most recently selected image is the only one that will be shown.


Click on Image 1
Add event handler
Click on image 2
Add event handler
Remove event handler
Image 1 starts downloading
X
onload event is still triggered but there is no event handler to show the image
In this scenario we remove the event handler for image 1 so it is not shown once it downloads as we no longer want to see it.
Image 2 starts downloading
onload is triggered
Image 2 is shown
Time

How you remove the eventHandler will depend on how you added it.

E.g.

If you added it as:

```
imageObject.onclick = function() { .... };
```

Then you can remove it with:

```
imageObject.onclick = null;
```

Key Presses

The **which** property of the event object passed to the **keyup** event handler contains the code of the key that was pressed.

```
window.addEventListener("keyup", keyboard );

function keyboard(e)
{
   if (e.which == 37)
    { ..... } // code for left arrow

    else if (e.which == 39)
    { .... } // code for right arrow

    else
    { .....}
}
```

Showing the Next/Previous Image

One solution to showing the next or previous image is to navigate the DOM.

The URLs of the two images (large and small) are the same except for the last letter in the file name.

E.g.

Thumbnail:

http://farm5.static.flickr.com/4243/34886428006_c24751507d_**s**.jpg

Large image:

http://farm5.static.flickr.com/4243/34886428006_c24751507d_**z**.jpg

The URLs of the two images (large and small) are the same except for the last letter in the file name.

E.g.

Thumbnail:

http://farm5.static.flickr.com/4243/34886428006_c24751507d_**s**.jpg

Large image:

http://farm5.static.flickr.com/4243/34886428006_c24751507d_**z**.jpg

You can store the large version of the URL in the **dataset** of the thumbnail's image tag (or its container).

When you click on a thumbnail you can extract the URL of the large version from the DOM object (via datasets).

To find the next image you can find the next or previous sibling in the images' container.

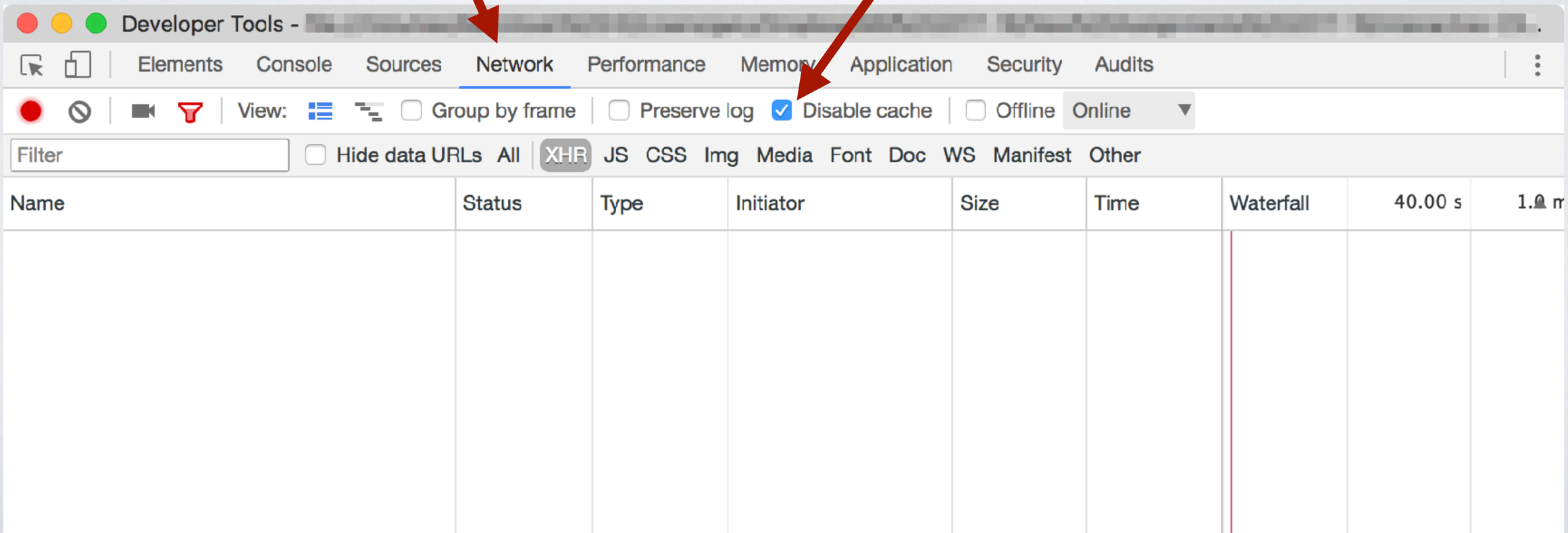
Scope

When creating elements in a loop, and adding event listeners, pay special attention to scope issues (we will cover this in class).

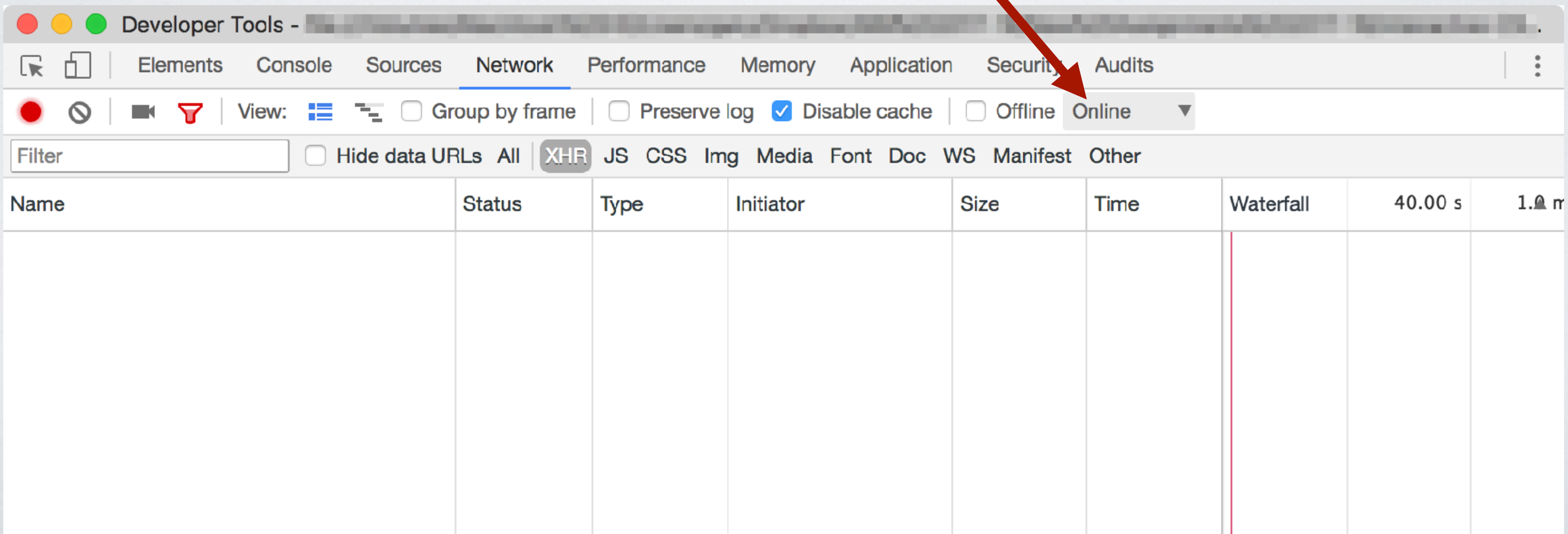
Testing

If you pick the **Network** tab in Chrome ...

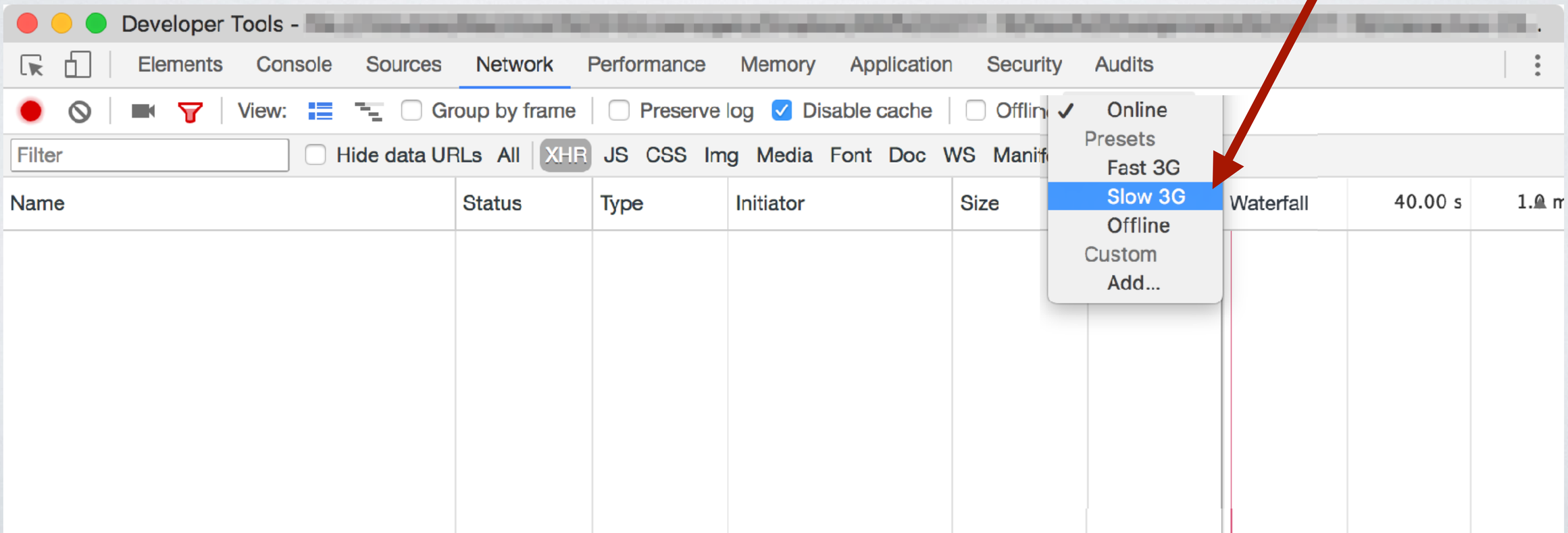
... you can disable the cache so that the browser will always use the network to access the images.



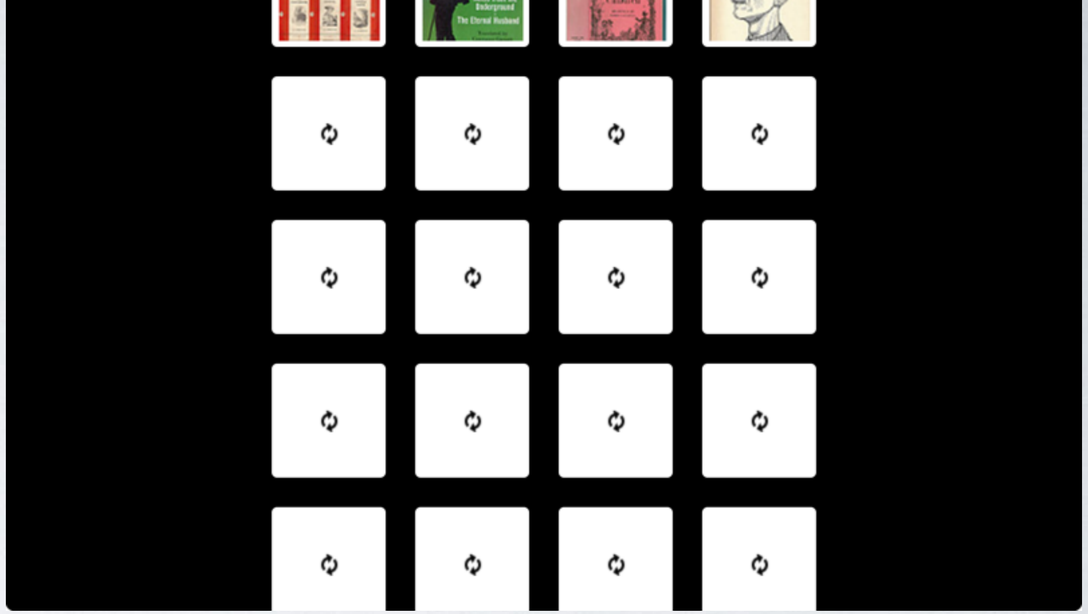
You can also have the browser simulate slower network speeds.

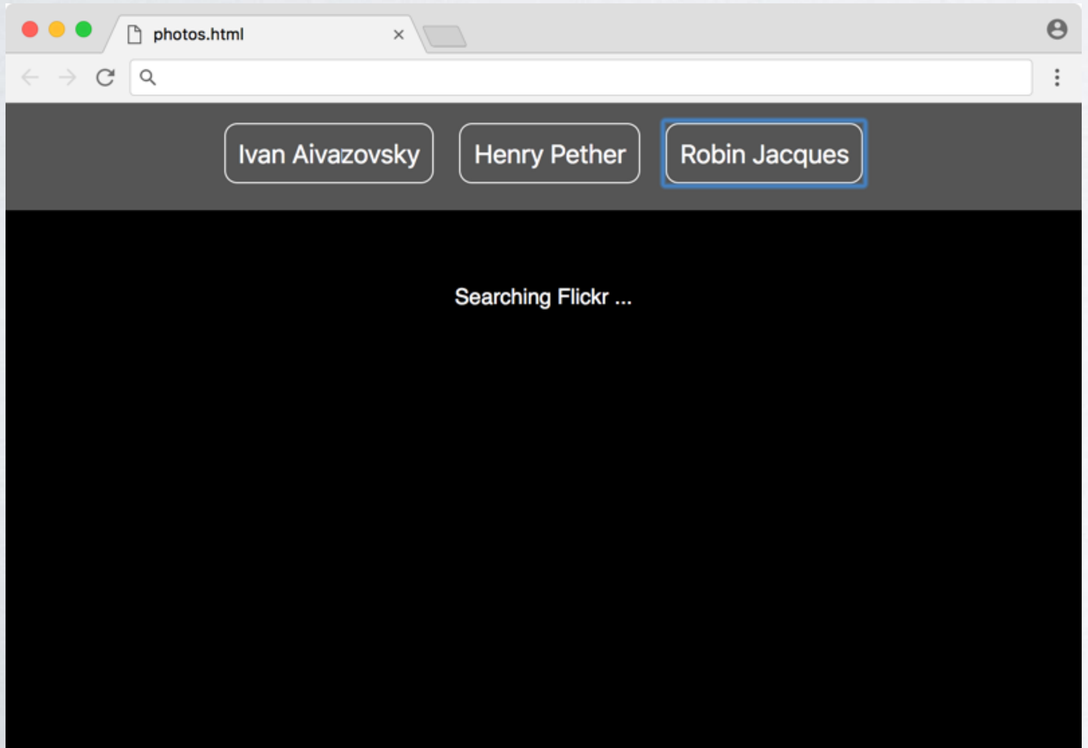


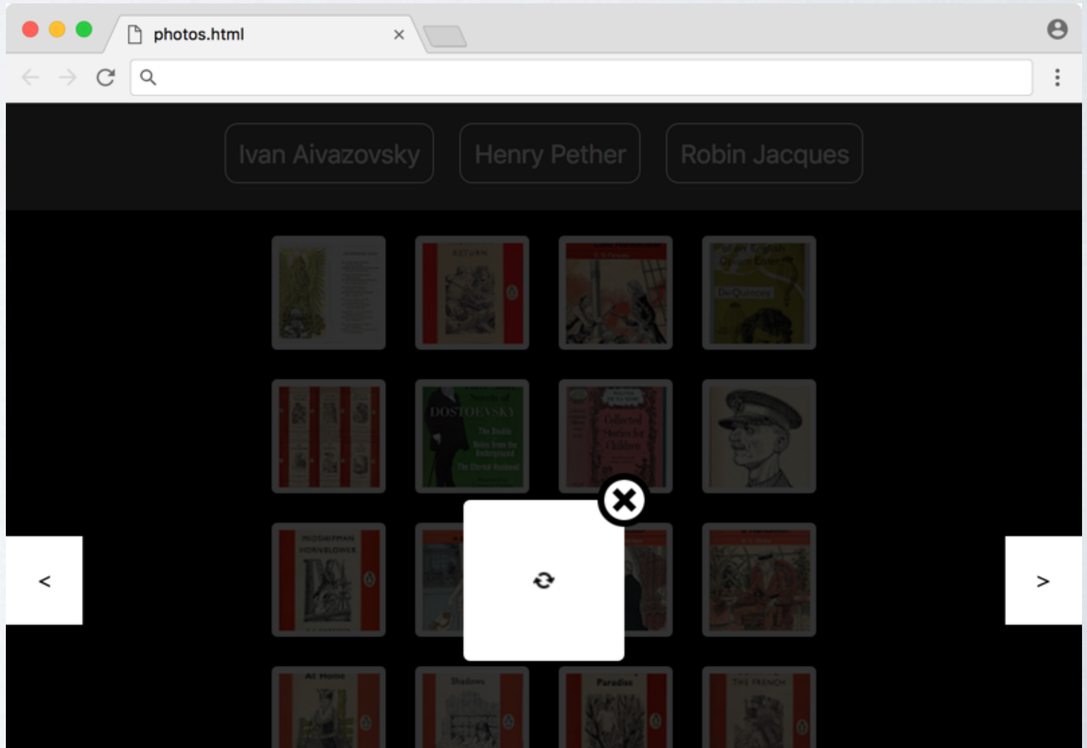
You can select "slow 3G" as a slower speed.



This means that it will take so long to download images and make network requests that you will be able to see your loader GIFs and test that that part of your code works.







One of the provided screencasts was made in this mode so you can see the loader gifs and **onload** events.

Submission

Submit your code on Blackboard

See separate document on assignment regulations

See included screencasts