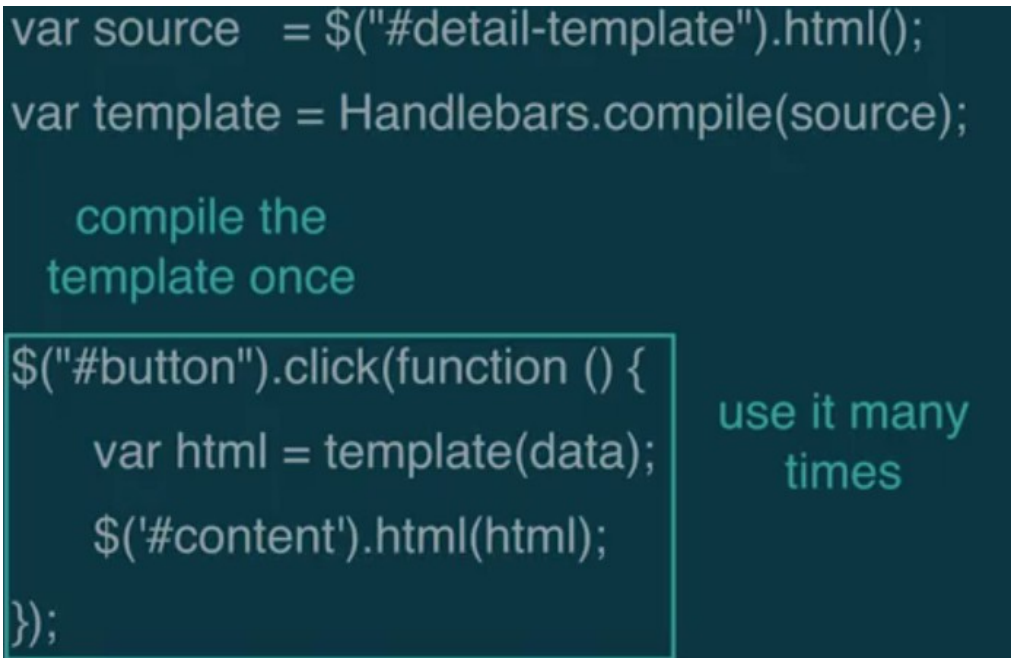
2.3.1.5: Displaying different data with the same template

We've learned about one of the most important concepts in web programming, which is the object and the separation of content and structure. And we've learned a really nice way of doing that with templates. And you're really now ready to start making some interesting dynamic websites. And over the rest of this video, and next weeks lectures, we're gonna look at a lot of ways in which we can do that. Now I wanna start with a couple of simple things that we can do already using these templates. Now, what I've told you is that we can separate contents and structure, and that allows us to use the same HTML, or in this case the same template, but put different types of data in.

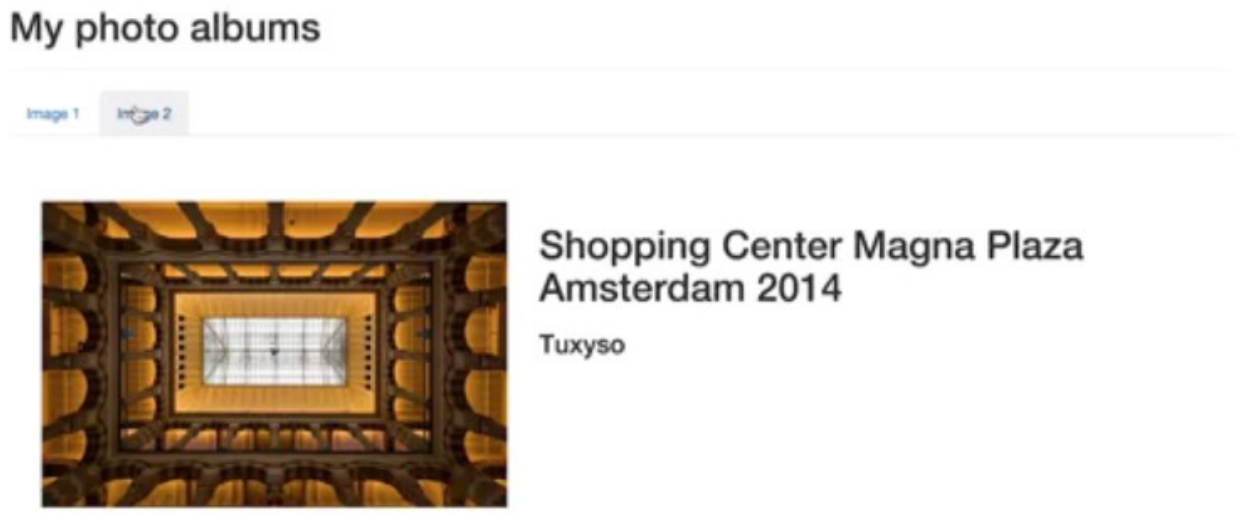
So, if we look at, here we've got an object, and via that object in the template we can create an HTML. But we can easily have another object with some other data and some other variables, and create some different HTML because the image is different, the title is different, and the author is different. And with this simple thing we can start generating a lot more web pages than we initially could just by typing them in, and it all becomes a lot easier.

So before we look at the complete example, I might want to focus in on the code we looked in last video, I mean, which first took the template, compiled it into some JavaScript and then applied the data to that template, and put it into the dom. Well, there's actually two bits to that.



So, the first bit up here is getting the template and compiling it so that we've got the template code. Now, we only ever need to actually do that once. And then once we've got that, we can use it many, many times in many different types of data. So, the only bit that we need to change when we have new data is the bit where we put the data in the template, get the HTML out, and put that into the dom.

So, we can put that part into a click function, as we saw in the first MOOC, and to get that working into actively so whenever we click on a button, we can load in some new data into the template and create some new HTML.



So, this is a webpage that's of the kind you might want to create. I've got an nav bar with two images. I can select image one. And image two. And up to now, in previous weeks we might have done them by having links to separate web pages, I'm gonna show you how to do that with a template.

Okay. So, this is the actual code for navigation that we've seen in our webpage, and we, it's basically a list. And we've got a couple of list items, and each one's got an ID. One is for the first image, the other is for the second image.

```
<div class="page-header">
  <h1>My photo albums </h1>
</div>

<!-- this is a simple nav bar for selecting between images -->
<ul class="nav nav-tabs">
      <li role=""  id="image1btn">
        <a href="#">Image 1</a>
        </li>
      <li role="" id="image2btn">
        <a href="#">Image 2</a>
        </li>
    </ul>
<br/><br/>
```

Now, in previous examples, these were links, and they were linking directly to another page. But I've kept them as links because they keep some of those functionalities by having linked them on the page.

- The href here is actually not a page, it's just this hash symbol. What does this hash symbol mean? Well, it just means a link back to the same web page. And it's basically saying, you know, this kind of a link, I want to sort of make it to look like a link, but actually it's not going to do anything, and the real functionality of this link is going to happen via jQuery.

I'm going to put that later down in a script. And this is a very common way of doing things in these kind of very interactive websites we want to create. Have things that are basically links, but don't link to other web pages. They actually trigger jQuery clicks that do stuff.

As before, we've got a content div that is empty, the only difference from the last example is I've made it more Bootstrappy. I've made the container fluid with a main role so it's gonna be a nice looking Bootstrap page instead of the very simple one's we've looked at so far.

```
<!-- the content of the web page starts off empty
  because we will fill it later from the template -->
<div id="content"  class="container-fluid"  role="main">
</div>
```

We've got the template, again, is pretty much exactly the same thing, except we're using Bootstrap now. So, we've got our fluid rows and we've got the columns. All of this is gonna be laid out using Bootstrap's nice grid structure. And, in particular, we're gonna have, separate out on one side, in one column we're gonna have the image tag itself.

```
<script id="detail-template" type="text/x-handlebars-template">
  <br>
  <div class="row-fluid">
    <div class="col-sm-5">

      <img  style="width:100%" src="{{src}}"/>
    </div>
              <div class="col-sm-7">
    <h1>{{title}}</h1>
                <h3 class="author">
                  {{author}}
                </h3>
    </div>

    </div>
  </div>
</script>
```

So the image in the other column, we're gonna have the metadata, the title and author. And down here is where it gets really interesting. We've got, this is where we've got two bits of data. So we've got two variables, one for the Earth image, as we can see here, and another variable for the image of the ceiling of this lovely shopping center in Amsterdam. As we've seen before, we've done exactly, we've compiled our template, pretty much exactly same as we saw in the last video. And this is another bit where it gets different.

We've got jQuery click functions that are being called, and these correspond to, this ID is the ID of image one in the navigation menu, and image two in the navigation menu. So, each one is getting separate click, and whenever we load one of those, where whenever one is clicked, we do this code which is loading one of our data variables into our template to create some HTML, and then putting that directly onto our contents, into our content div. So, this exactly the same code but we can do it differently for each of the images, and in particular, the one thing that's changing is this, the name of the data variable we're using. So, we're using data one in one context, and data two in the other. These two click functions actually look really, really similar, and really, we wouldn't want to have to write out separate functions for each one of these images.

```
$("#image1btn").click(function () {
  var html    = detail_template(data1);

  $('#content').html(html);
});


$("#image2btn").click(function () {
  var html    = detail_template(data2);

  $('#content').html(html);
});
```

And we'll see, in the next video, and the following MOOC of this specialization we don't really need to write separate functions. But, I thought for simplicity at the moment, we'll just separate them out so it's very clear what's going on. And that's the very starting point of how you can use templates to create a number of different HTML pages all from the same basic structure.