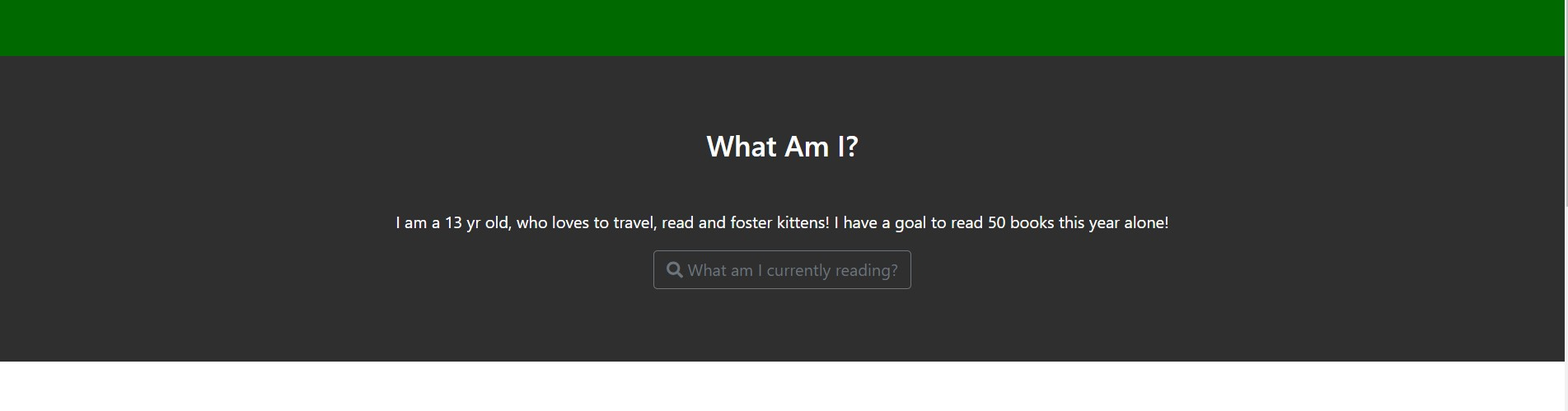
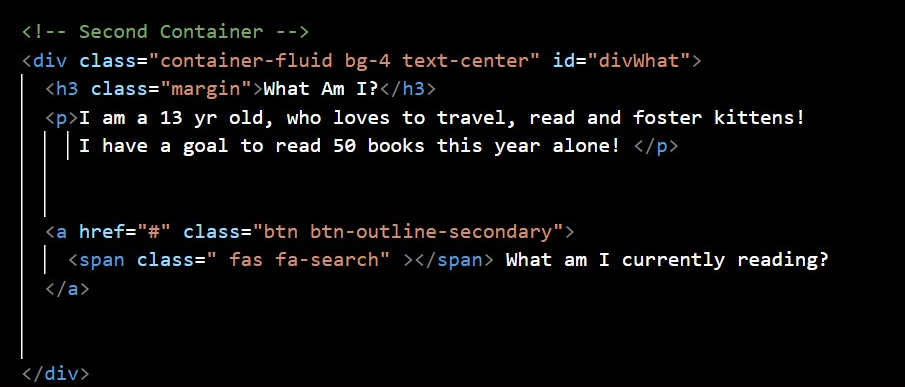
**Lesson Plan 3: Bootstrap Grids and Modals**

We continue building our project from where we left off.  


**<!--Second Container –>**

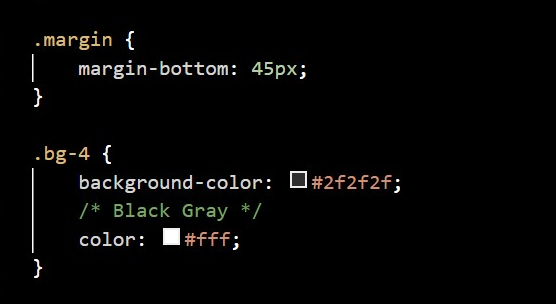
The second container is very straightforward.

We will start by creating it first then add a modal to the “What am I currently reading” button.



The main div for this has a **container-fluid** class. The **text-center** ensures all the text in this div is aligned to the horizontal center**. bg-4** class is defined in our **style.css** and adds the grey colour to the div.  
  
The **id=”divWhat”** is for the navigation at the top.

The heading is added with an **<h3>** which will automatically be center aligned, however, the class **margin** is defined in our **style.css** and adds a bottom margin of 45px for better display.

****

Finally, a single line description is added via a <p> tag.  
  
For the button with the search icon🔍 we are using an anchor tag. However, adding the class **btn**, adds the necessary outline and width to it. (explore : **btn-outline-secondary**)

We use the Font awesome icons fa-search that adds the search icon to our button.  
The syntax to use any font-awesome character is

**<span class=”fas fa-\*”></span>** where **\*** is replaced with the available character list.

Before we move on to adding our modal, we will need to know about the Bootstrap Grids.

**Bootstrap Grids.**

This lesson will cover the most important aspects of Bootstrap - The Bootstrap Grid System,

**Part 1: Bootstrap Grids**

As we have discussed before, the power of Bootstrap framework lies in its ability to create Responsive designs.

In very simple terms, whenever we create a web page, we tell the browser

1. Which element we want to display (for eg: text, image, button etc),
2. Where on the screen we want to display it
3. How much size should the element occupy.

We have often specified the size of the elements in px or in %.

When we specify the size in px, irrespective of the size of the screen available, the element always takes up the specified size. This disrupts the aesthetic and utility of the design and might make the website very difficult to view.

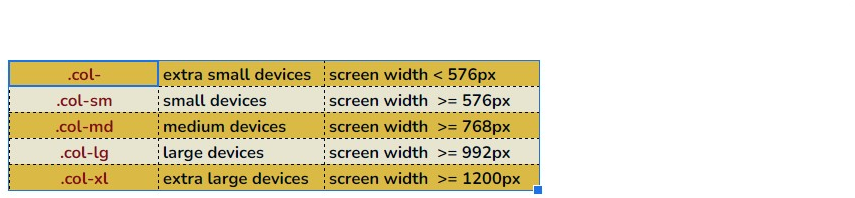
For ex: if we want a big image header, we provide a size of 500px. It might look perfect on a large screen but when opened on a smaller screen, it will distort all the other elements (since it will still take up 500px space) and make viewing very difficult if not impossible.

Using % is the first step to making it responsive. We can specify that an image should take up 50% of the screen. So if the available screen size is smaller, the image size also decreases proportionately.

The Bootstrap framework works on this model.

**The 12 column rule**

In Bootstrap we segregate the display devices in 5 classes on the basis of their size(in pixel)



The screen, irrespective of its size, is made up of 12 equal responsive columns.

(Teachers open the Bootstrap Grid project to show the output. Resize the screen to show how the width of the columns are always equal however they decrease proportionally as the screen size reduces.)

If the available screen size falls below 576px, the columns will automatically stack one below the other. This is very useful when viewing on mobiles.

How to code?

Create the main wrapper div with container or container-fluid class.

<div class=”container-fluid”>

</div>

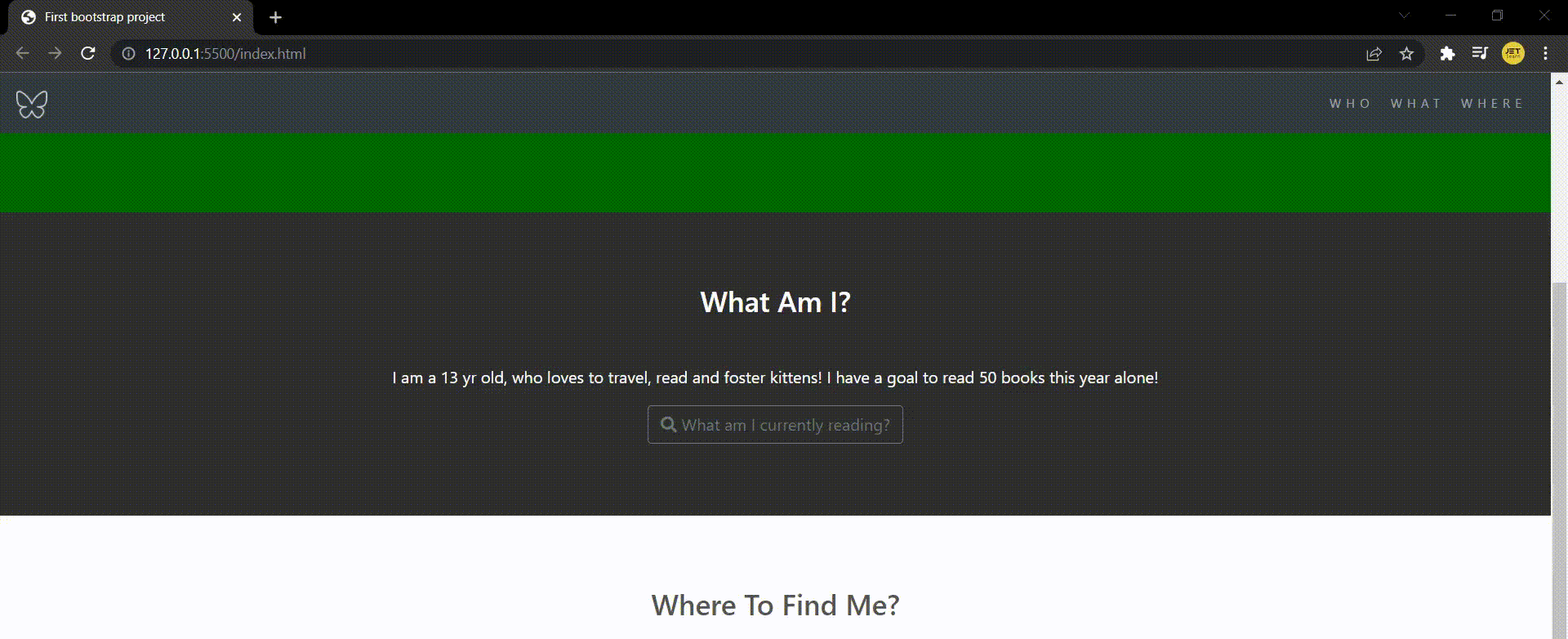
And

We have seen previously with our container-fluid and container classes, how the size of the div increases or decreases depending on the screen size.

**Part 2: Modals**

**What is a Modal?**

Project showcase: Open the Project and click on the ‘What am I currently reading?’ button



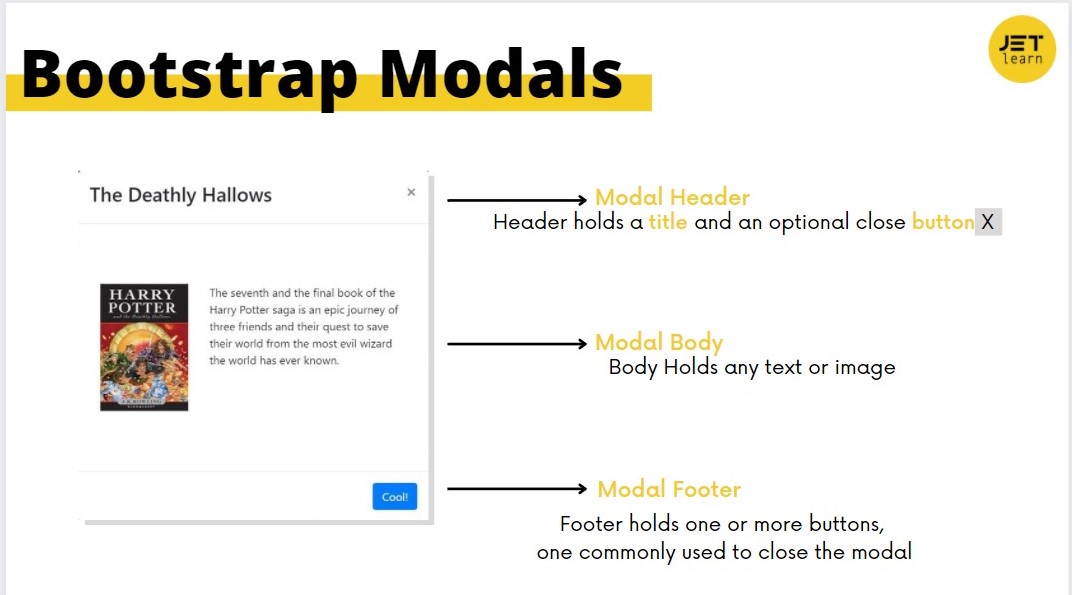
**Bootstrap Modals** are these popups which appear on your page when you click on some button(or text/image). These pop ups might give us some information or allow us to take some actions. For example, we could also add a link to Amazon here along with the book description.

As you can see, once the modal pop up, it also has the focus on it such that your main page seems to darken or fade.

These are pretty easy to create with Bootstrap once you understand the structure.

Bootstrap gives us all the classes we need to create and stylise our modals.   
  
💡One more important point to remember is when using Modals in your websites, make sure you have included the Bootstrap Javascript files in your project.

The content of the Modal typically has 3 parts.



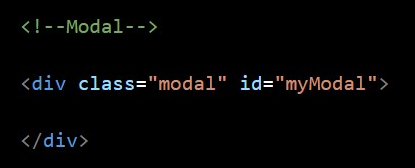
💡 Modals by default are not visible on the screen unless an element on the screen triggers it to show itself.

There are 2 steps to creating a Modal.  
  
1. Create a Modal.

2. Add codes to the element (text/button/image) that on clicking will open the modal

1. **Create a modal.**
2. The entire modal code is wrapped inside a main <div> with a class=”**modal”** like so.

**.modal** Identifies the content as a modal and brings focus to it



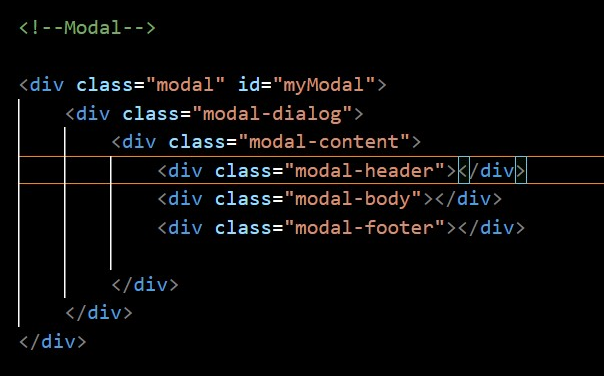
1. We also added an **id** to this div to ensure that our button(text/image) is able to identify and open this modal.
2. We create another div inside our modal div with class=”**modal-dialog**”

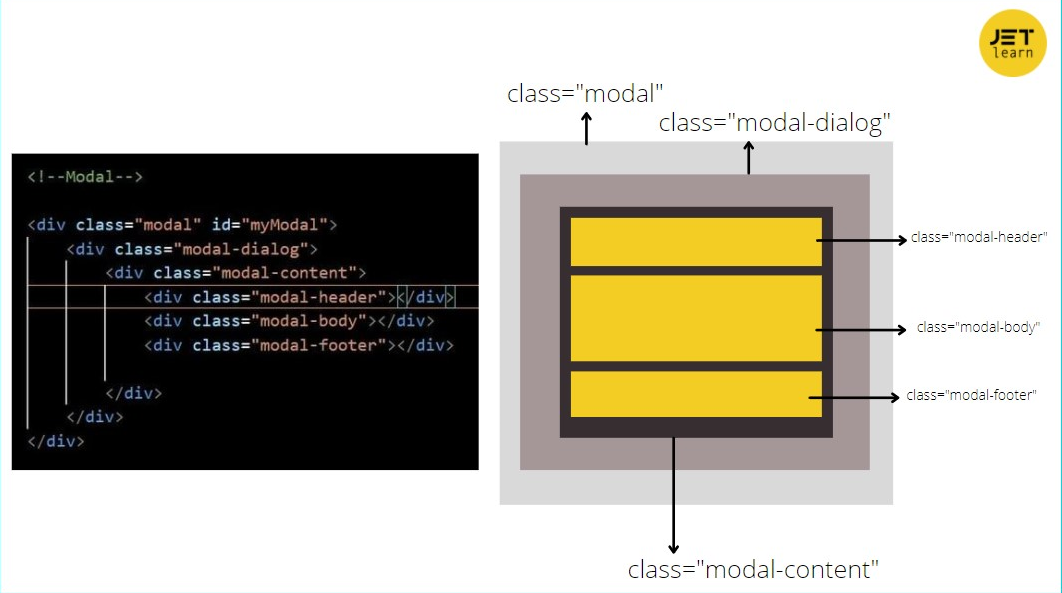


Now we are ready to add all the content of our modal, hence we will add another div with class=**”modal-content”**

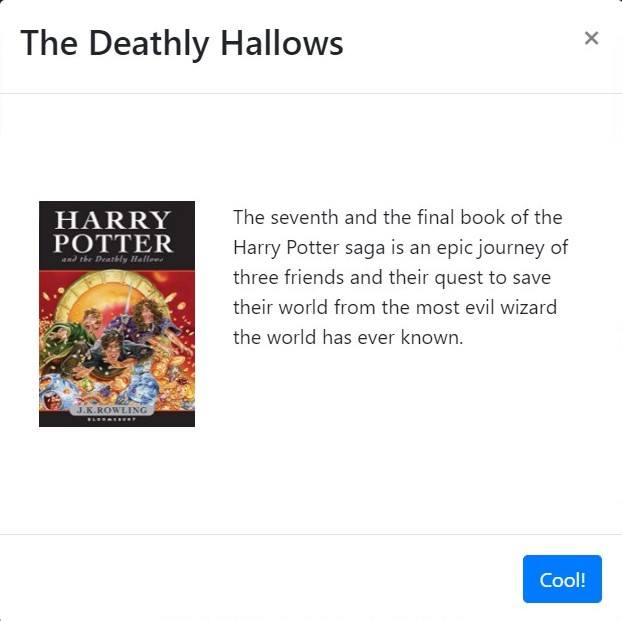


**modal-content** has 3 parts we discussed in the beginning, **a header, a body and a footer**.



So the basic structure looks like this:  
  


And we are ready to add the contents.  
  
**The modal-header.**  
We will add a heading and a close button to the heading.



For the heading, I am using a **<h3>** tag. You can choose any tag or style of your choice.  
The close button you see on the right, will close the modal. It is easy to create and position it with the bootstrap **close** class

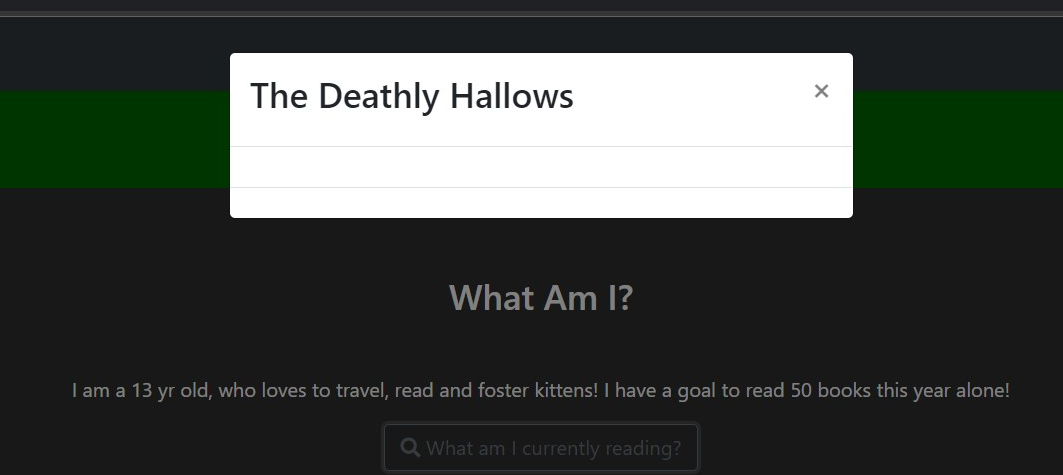
**<button type=”button” class=”close” > </button>**

.close Styles a close icon. This is often used for alerts and modals. Often used together with the **&times;** symbol to create the actual icon (a better looking "x"). **It floats right by default**



To ensure this close icon can actually close our modal, we add a **data-dismiss=”modal”** attribute to the button

We haven’t linked it to our***‘What am I currently reading”***button, but if we had, this would be the result.



**The modal-body.**

| .modal-body | Defines the style for the body of the modal. Add any HTML markup here (p, img, etc) |
| --- | --- |

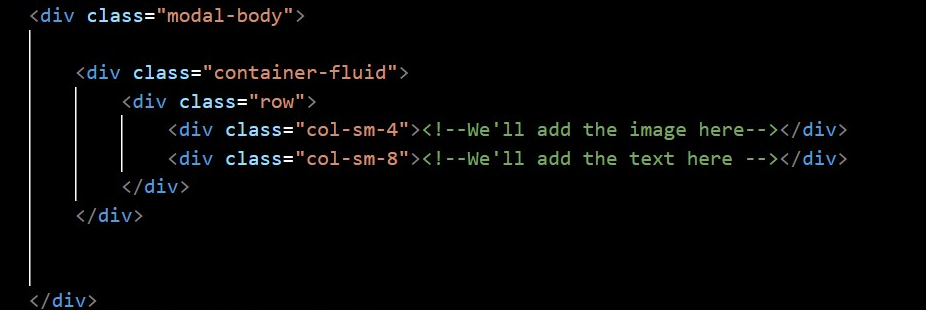
The modal-body is used to display the main content that needs to be displayed on our modal.

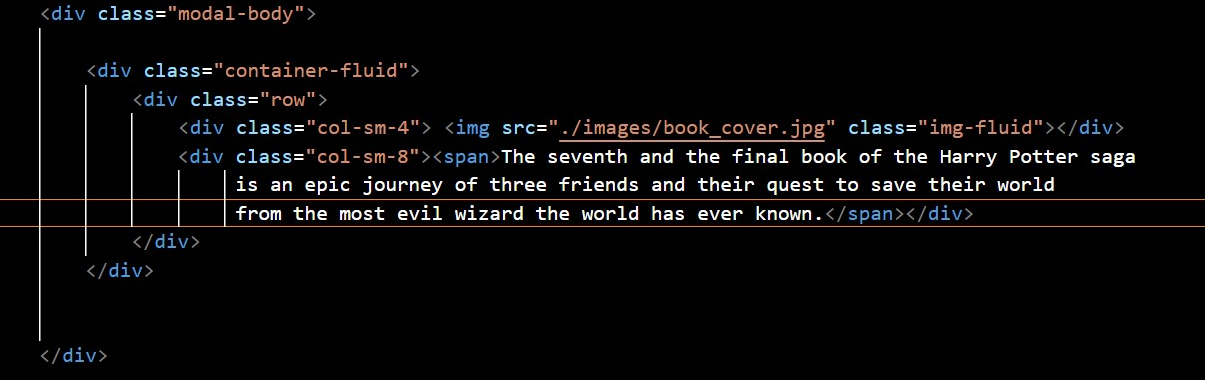
The display is completely up to the user’s choice, we’ll see how we can display 2 contents, an image and some associated text, side by side.  
  
For this we will use the Bootstrap grids.   
We know, Bootstrap divides the available width of the screen in 12 equal columns.   
We will use 4 columns for the image and the remaining 8 for the text.

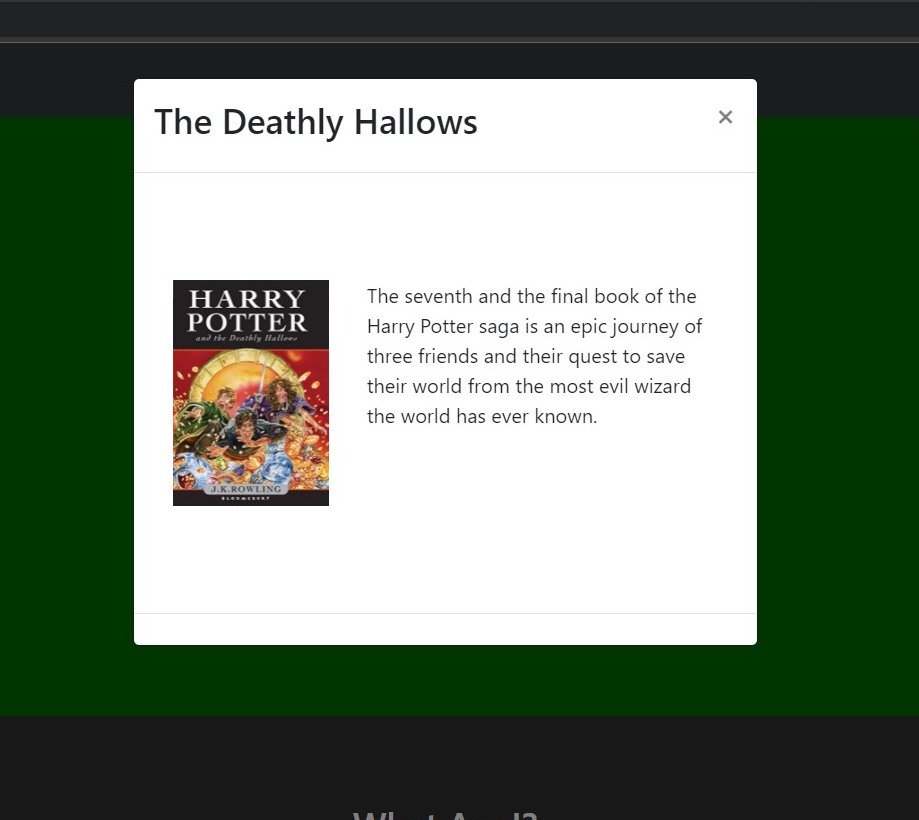
So we create a **container-fluid div** which will utilise the entire space available.

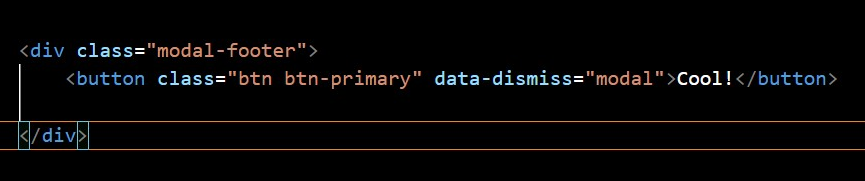
The **<div class=”row”> </div>** will create a row to hold our 12 columns.

**<div class=”col-sm-4”> </div>** space will be our holder for the image.

**<div class=”col-sm-8”></div>** will be the holder for text. And since both these are wrapped inside the **div class=”row”** they will appear side by side.   
  


All we need to do is add the book image (downloaded and saved in the images folder) and the text.  


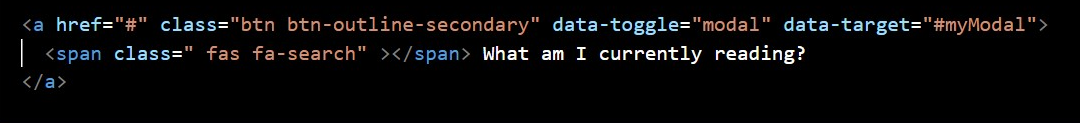
**Output**  


**The modal-footer.**Our Footer will have one button. And we will use the button element along with the bootstrap **btn** class to create it. Give it a colour of your choice, using **btn-primary** or **btn-success** (or any other colour class of your choice.  
  


Of course, just like the close icon in the header, this button should also disable the modal, achieved by adding the **data-dismiss =”modal”** attribute.

Our Modal is all ready.  
  
**II. Add the Modal to it’s trigger element**

To link the element that will display our modal we’ll follow some simple steps.   
I will be adding it to our ***‘What am I currently reading”*** *button.*

**

We already have the anchor tag with the **class=”btn”**. Adding the two attributes to the same will allow it to toggle our modal.

***data-toggle=”modal”***

And the id of the modal it needs to open/close.  
***data-target = “#myModal”***

💡**Note:** the id of the modal needs to be preceded with the ‘#’ in the data-target attribute

Our modal is all ready.   
Added tip: to add a smooth fade effect to the modal when it appears, add class fade to the main wrapper div

HW: Try creating more modals and applying them to other elements on your web page