1. **Google Maps API**

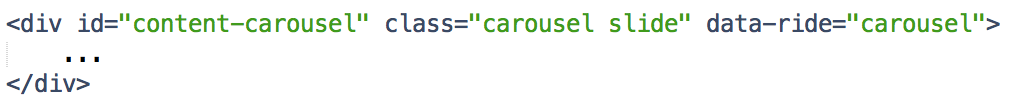
To help you along the way with the Google API here is an example that produces a map centered on the Sky buildings where you are sat now!

<!DOCTYPE html>  
<html>  
<head>  
<script src="http://maps.googleapis.com/maps/api/js"></script>  
<script>  
function initialize() {  
  var mapProp = {  
    center:new google.maps.LatLng(51.487240, -0.327445),  
    zoom:5,  
    mapTypeId:google.maps.MapTypeId.ROADMAP  
  };  
  var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);  
}  
google.maps.event.addDomListener(window, 'load', initialize);  
</script>  
</head>  
  
<body>  
<div id="googleMap" style="width:500px;height:380px;"></div>  
</body>  
  
</html>

# Bootstrap Carousel

**<div> wrapper**

Wrap your Carousel in *<div>* tags, and ensure that the *class* is specified as ‘*carousel slide’* and that *data-ride****­*** parameter indicate that the <*div>* is a ‘*carousel*’. Also provide an *ID* that will be used to reference the carousel.



**Indicators**

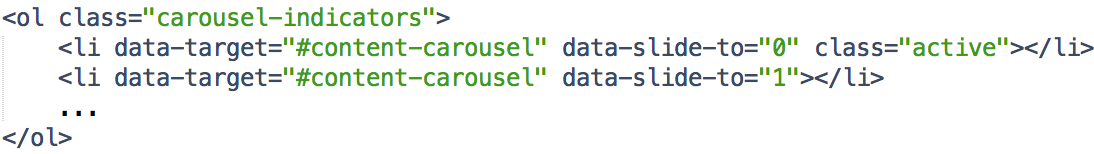
****

Create the indicators on the carousel. These are the icons displayed at the base of the carousel showing which slide the carousel is currently on.

The *data-slide-to* parameter will indicate the slide that should be shown if the indicator is clicked.

Ensure that the class ‘*carousel-indicators’* is specified for the *<ol>.*

Ensure that the *data-target* of the list items points to the *ID* of the carousel. Finally, set the active icon by setting the class as *active* on the list item to be active.

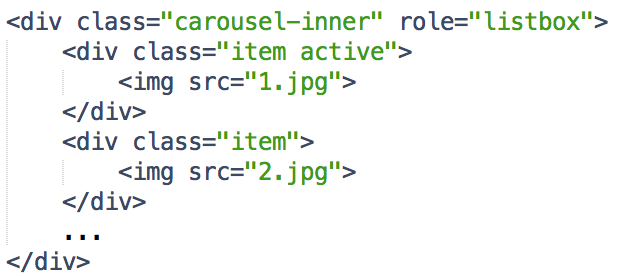


**Carousel Content**

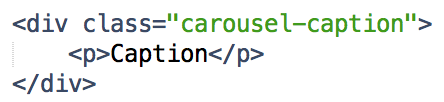
The content of the carousel, displaying the assets chosen.

Ensure that the <*div>* has a class of ‘*carousel-inner’* and it has a *role* of *‘listbox’.*

Inside the <*div>*, create items to indicate a slide on the carousel, with each having a class of *‘item’.* Set the active class with the ‘*active’* class name.



You can also add a caption to each item by including a caption <*div>* inside of it. You can set the content however you wish (it does not have to be a *<p>* as shown below).



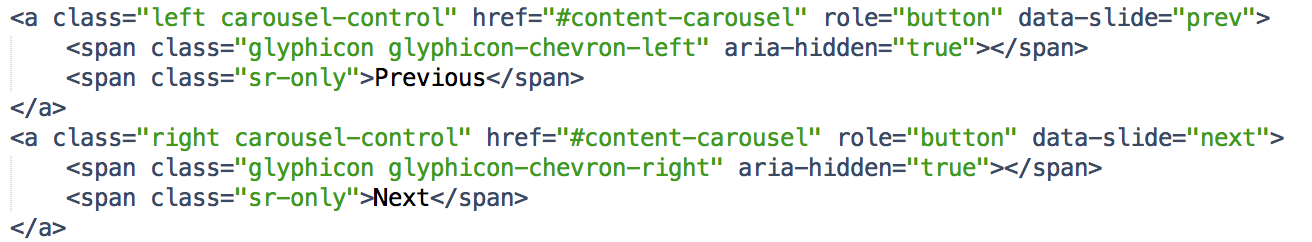
**Carousel Controls**

****

These are the icons you see on the side of the carousel that can be clicked to navigate slides.

*<a>* tags are used to specify the controls. They should have a class of *‘left|right carousel-control’*. Their role should specify that they are a *‘button’*, and the action of this button is indicated in the *‘date-slide’* parameter, in which *‘prev’* links to the previous slide, and *‘next’* links to the next slide. Finally the *href* for the *<a>* tag should refer to the *ID* of the carousel.

Inside the *<a>* tags should be two spans. The first refers to the icon that is displayed (using Bootstrap, the inbuilt icons are called glyphicons). The second span is optional, and is only required for screen readers.

****

1. **Twitter API**

If you are struggling with the API, the easiest way to create a Twitter for Websites widget is to use Twitter’s configuration tools at https://twitter.com/settings/widgets/new. To do this you will need to make an account for Twitter. If you follow the instructions on Twitter’s widget page you will get something like the following:

<a class="twitter-timeline" href="https://twitter.com/SkyUK" data-widget-id="648182594014703616">Tweets by @SkyUK</a>

<script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id)){js=d.createElement(s);js.id=id;js.src=p+"://platform.twitter.com/widgets.js";fjs.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-wjs");</script>

1. **Sky Content Server**

An alternative to the http get request suggested earlier is to use jQuery. An example is to use jQuery’s “.get” to load data from the server.

$.get( "http://10.88.58.229:8080/sports", function( data ) {

console.log( data );

});

Note. Make sure that jQuery is included in the <head> of your page!