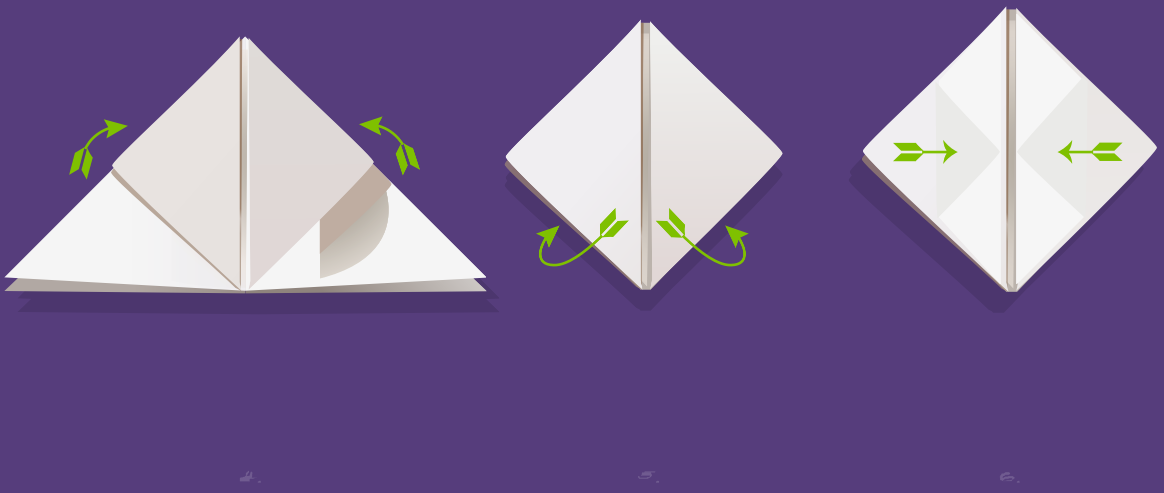
sitepoint

8 PRACTICAL BOOTSTRAP PROJECTS



BUILD RESPONSIVE, MOBILE-FIRST SITES WITH EASE

8 Practical Bootstrap Projects

Copyright © 2018 SitePoint Pty. Ltd.

- **Product Manager:** Simon Mackie
- **English Editor:** Ralph Mason
- **Project Editor:** Maria Antonietta Perna
- **Cover Designer:** Alex Walker

Published by SitePoint Pty. Ltd.

48 Cambridge Street Collingwood
VIC Australia 3066
Web: www.sitepoint.com
Email: books@sitepoint.com

Printed and bound in the United States of America

About SitePoint

SitePoint specializes in publishing fun, practical, and easy-to-understand content for web professionals. Visit http://www.sitepoint.com/ to access our blogs, books, newsletters, articles, and community forums. You'll find a stack of information on JavaScript, PHP, Ruby, mobile development, design, and more.

Table of Contents

Preface

Bootstrap stands as one of the most popular, open-source, front-end frameworks on the Web. Since its official release in 2011, it has undergone several changes, and it’s now one of the most stable and responsive frameworks available. It’s loved by web developers of all levels, as it gives them the capability to build a functional, attractive website design within minutes. A novice developer with just some basic knowledge of HTML and little CSS can easily get started with Bootstrap.

In this book, we'll offer a selection of nine different practical projects that you can follow along with.

Who Should Read This Book?

This book is for all Frontend developers that want to build sites and apps that run faster. You’ll need to be familiar with HTML and CSS and have a reasonable level of understanding of JavaScript in order to follow the discussion.

Conventions Used

You’ll notice that we’ve used certain typographic and layout styles throughout this book to signify different types of information. Look out for the following items.

Code Samples

Code in this book is displayed using a fixed-width font, like so:

```
<h1>A Perfect Summer's Day</h1>
<p>It was a lovely day for a walk in the park.
```

```
The birds were singing and the kids were all back at school.</p>
```

Where existing code is required for context, rather than repeat all of it, ⋮ will be displayed:

```
function animate() {
   ⋮
new_variable = "Hello";
}
```

Some lines of code should be entered on one line, but we've had to wrap them because of page constraints. An ➥ indicates a line break that exists for formatting purposes only, and should be ignored:

```
URL.open("http://www.sitepoint.com/responsive-web-
➥design-real-user-testing/?responsive1");
```

Tips, Notes, and Warnings

Hey, You!

Tips provide helpful little pointers.

Ahem, Excuse Me ...

Notes are useful asides that are related—but not critical—to the topic at hand. Think of them as extra tidbits of information.

Make Sure You Always ...

... pay attention to these important points.

Watch Out!

Warnings highlight any gotchas that are likely to trip you up along the way.

Live Code

This example has a Live Codepen.io Demo you can play with.

Github

This example has a code repository available at Github.com.

Chapter 1

Spicing Up the Bootstrap Carousel with CSS3 Animations

Maria Antonietta Perna

Adding a slider or carousel to showcase content on a website is a common client request for developers. The amount of free and premium carousel plugins available is overwhelming, and a good many of them offer plenty of useful configuration options and dynamic effects. There are times, however, when a lightweight carousel with minimal options is all you need. In this case, if your project uses Bootstrap, the popular open-source, front-end framework, you won't need to look any further than the Bootstrap Carousel component.

In this article, I'm going to show how to add some fun animation effects to the Bootstrap Carousel, while still making sure this handy JavaScript component remains bloat-free and quick to implement.

Introducing Animate.css

As rewarding as crafting my own animation effects can be, I'm going to use a well-known, open-source CSS3 animation library most aptly called Animate.css, by Dan Eden.

This is so that I can focus on the task at hand, rather than on explaining the code for CSS3 animations. However, if you want to delve into that topic, you'll enjoy the CSS3 Animations series here on SitePoint, by Craig Buckler.

Using Animate.css requires two steps:

1. include `animate.min.css` in the `<head>` section of your HTML document
2. add the classes of `animated yourchosenanimation` to the elements you

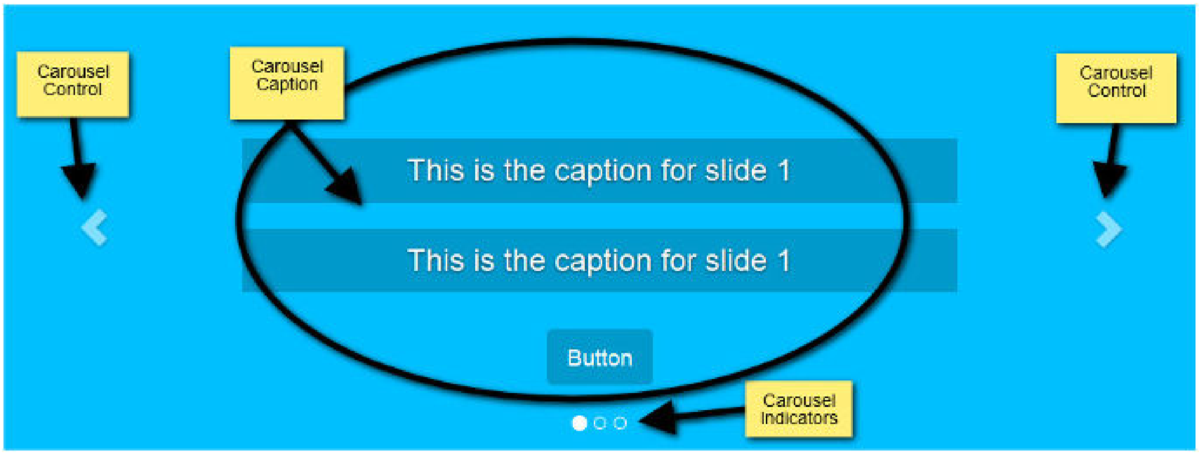
intend to animate on your web page.

In the latter step, you would replace *yourchosenanimation* with the class name corresponding to any of the numerous animations you see on the Animate.css website.

Introducing the Bootstrap Carousel

The Bootstrap Carousel component has three main sections:

- The **Carousel indicators** track the overall number of slides, give users a visual clue of the position the slide currently being viewed occupies, and offer an alternative navigation for the slider.
- The **Carousel item**, located inside a wrapper container with a class of `.carousel-inner`, represents each individual slide. It's inside each item that you place your images. You can also add **captions** to your slides. The nice thing is that you can put pretty much any HTML element inside a container with the class of `carousel-caption` and Bootstrap will take care of the styling and formatting. It's these captions that you're going to animate.
- Finally, the **Carousel controls** are the navigation arrows that enable users to access the next and previous slides.



1-1. Bootstrap Carousel structure

To keep this demo simple, I'm not going to add images to the carousel. The focus is all on how to animate the carousel captions.

Building the HTML Structure

If you're following along, here's what you need to include in your project:

- jQuery
- Bootstrap’s CSS and JavaScript
- Font Awesome, or any other brand of icons, although you're free to replace the icons with simple text
- Animate.css
- A stylesheet and JavaScript document where you’ll add your own code.

Here’s the code for the Bootstrap Carousel:

```
<!-- indicators -->
<div id="carouselExampleIndicators" class="carousel slide">
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active">
    </li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>

  <!-- carousel content -->
  <div class="carousel-inner">

    <!-- first slide -->
    <div class="carousel-item active">
      <div class="carousel-caption d-md-block">
        <h3 data-animation="animated bounceInLeft">
          This is the caption for slide 1
        </h3>
        <h3 data-animation="animated bounceInRight">
          This is the caption for slide 1
        </h3>
        <button class="btn btn-primary btn-lg" data-animation="animated zoomInUp">
        Button</button>
      </div>
    </div>

    <!-- second slide -->
    <div class="carousel-item">
      <!-- second slide content -->
    </div>
```

```
      <!-- third slide -->
      <div class="carousel-item">
        <!-- third slide content -->
      </div>
    </div>

    <!-- controls -->
    <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button"
    data-slide="prev">
      <span class="carousel-control-prev-icon" aria-hidden="true"></span>
      <span class="sr-only">Previous</span>
    </a>
    <a class="carousel-control-next" href="#carouselExampleIndicators" role="button"
    data-slide="next">
      <span class="carousel-control-next-icon" aria-hidden="true"></span>
      <span class="sr-only">Next</span>
    </a>

  </div>
```

The elements inside the carousel caption that you'll be animating have a `data-animation` attribute added to them with the specific animation class name as their respective value.

If you'd like to experiment with other animations from the Animate.css library, feel free to replace the values in the `data-animation` attribute with your chosen animation class names.

Adding CSS to the Carousel

Now, give free rein to your creativity and style the carousel captions according to your taste. The style rules that I'm going to focus on here are those relevant to the smooth working of this demo.

More specifically, you're taking control of the CSS `animation-delay` property, which defines when each animation starts.

```
.carousel-caption h3:first-child {
  animation-delay: 1s;
}

.carousel-caption h3:nth-child(2) {
  animation-delay: 2s;
}

.carousel-caption button {
  animation-delay: 3s;
}
```

The snippet above ensures that the elements start their animation sequentially. There's room for play here. For instance, you can choose to start animating the first two headings at the same time, followed by the button animation. It's up to you, so have fun with it!

Writing the jQuery

Let's start by initializing the carousel. In your custom JavaScript file, add this code snippet:

```
var $myCarousel = $('#carouselExampleIndicators');

// Initialize carousel
$myCarousel.carousel();
```

Now the carousel is in motion. Let's tackle the animation part.

To animate the captions in the first slide, the script has to fire as soon as the page finishes loading in the browser. However, to animate subsequent slides as they come into view, the code will have to fire on the `slide.bs.carousel` event. This means that the same code will be used twice: on page load and on the `slide.bs.carousel` event.

To avoid repetition (following the DRY principle), just wrap your code inside a function and attach it to the appropriate events as required.

Here’s the code:

```
function doAnimations(elems) {
  var animEndEv = 'webkitAnimationEnd animationend';

  elems.each(function () {
    var $this = $(this),
        $animationType = $this.data('animation');

    // Add animate.css classes to
    // the elements to be animated
    // Remove animate.css classes
    // once the animation event has ended
    $this.addClass($animationType).one(animEndEv, function () {
      $this.removeClass($animationType);
    });
  });
}

// Select the elements to be animated
// in the first slide on page load
var $firstAnimatingElems = $myCarousel.find('.carousel-item:first')
  .find('[data-animation ^= "animated"]');

// Apply the animation using the doAnimations()function
doAnimations($firstAnimatingElems);

// Attach the doAnimations() function to the
// carousel's slide.bs.carousel event
$myCarousel.on('slide.bs.carousel', function (e) {
  // Select the elements to be animated inside the active slide
  var $animatingElems = $(e.relatedTarget)
    .find("[data-animation ^= 'animated']");
  doAnimations($animatingElems);
});
```

There’s quite a lot going on in the chunk of code above, so let’s break it down.

Looking into the `doAnimations()` Function

The `doAnimations()` function performs the tasks described below.

It starts by caching a string in a variable containing the name of the **animationend** event. This event fires, you might have guessed, when each animation ends. You need this bit of information because each time the animation ends, you need to remove the Animate.css classes. Failing to do this, the carousel captions will be animated only once, that is, just the first time the carousel shows a particular slide.

```
var animEndEv = 'webkitAnimationEnd animationend';
```

Next, the function loops over each element you want to animate and extracts the value of the `data-animation` attribute. As you recall, this value contains the Animate.css classes that you need to add to the elements inside the carousel in order to animate them.

```
elems.each(function () {
  var $this = $(this),
    $animationType = $this.data('animation');
  // etc...
});
```

Finally, the `doAnimations()` function dynamically adds the Animate.css classes to each element that you want to animate. It also attaches an event listener that fires only once, when the animation ends. After the animation ends, the Animate.css classes that you just added are removed. This ensures that the next time the carousel comes back to the same slide, the animations take place again. (Try removing this bit of code and you'll see the animations happen only once.)

```
$this.addClass($animationType).one(animEndEv, function () {
  $this.removeClass($animationType);
});
```

Animating the First Carousel Caption

As soon as the page loads in the browser, you trigger the animation inside the first slide like so:

```
var $firstAnimatingElems = $myCarousel.find('.carousel-item:first')
  .find("[data-animation ^= 'animated']");
doAnimations($firstAnimatingElems);
```

In this code, you begin by finding the first slide. From there, select the content you want to animate inside the caption by using the values of the `data-animation` attribute starting with *animated*. You then use the piece of data thus obtained as an argument in the `doAnimations()` function and let the function do its job.

Animating the Carousel Captions as They Slide

Animating the carousel captions as each slide becomes visible requires the steps described below.

First, attach an event listener to the `slide.bs.carousel` event. According to the Bootstrap Carousel documentation:

> *This event fires immediately when the slide instance method is invoked.*

```
$myCarousel.on('slide.bs.carousel', function (e) {
  // do stuff...
});
```

Next, select the **active slide** — that is, the slide currently in view — and from there find the elements you intend to animate. The code below uses the

`.relatedTarget` property of the `slide.bs.carousel` event to get hold of the active slide.

```
var $animatingElems = $(e.relatedTarget).find("[data-animation ^= 'animated']");
```

Finally, call the `doAnimations()` function, passing `$animatingElements`, which contains the list of elements to be animated, as an argument.

```
doAnimations($animatingElems);
```

The full demo is shown in the CodePen below.

Live Code

This example has a live Codepen.io demo you can play with.

Conclusion

As many of you probably know, Carousels do have issues that developers need to take into consideration.

With the Bootstrap Carousel component, adding a slider or carousel to a web page is just a matter of entering the appropriate HTML markup.

In this chapter, I've shown how to add some extra pizzazz to the basic Bootstrap Carousel component with a few lines of jQuery and the Animate.css library. However, any other similar CSS library, or coding the CSS3 animations from scratch, will do just as well.

Chapter

2

A Full-screen Bootstrap Carousel with Random Initial Image

George Martsoukos

In this chapter, I'm going to build two simple extensions for the Bootstrap carousel. First, I'll create a full-screen Bootstrap Carousel slideshow, and then I'll show you how to randomize the first slide on page load.

But before digging into those extensions, let's start by creating a carousel based on the default styles.

Building the Carousel

To create the carousel, we'll take advantage of the basic code for the carousel component that Bootstrap provides:

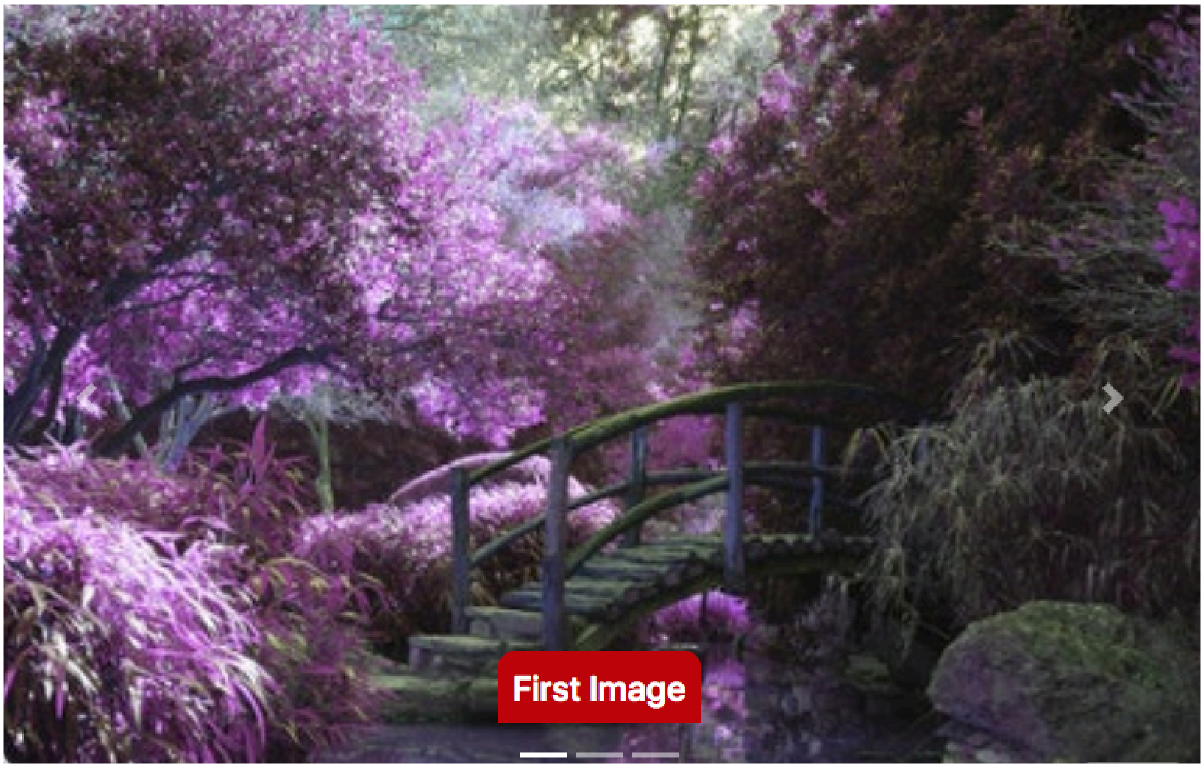
```
<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">

  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active">
    </li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>

  <!-- Wrapper for slides -->
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img class="d-block w-100" src="1.jpg" data-color="lightblue"
      alt="First Image">
      <div class="carousel-caption d-none d-md-block">
        <h5>First Image</h5>
      </div>
    </div>
    <div class="carousel-item">
      <!-- slide content -->
    </div>
    <div class="carousel-item">
      <!-- slide content -->
    </div>
    <!-- more slides -->
  </div>

  <!-- Controls -->
  <a class="carousel-control-prev" href="#carouselExampleIndicators"
  role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators"
  role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>

</div>
```

Notice that each of our images contains the custom `data-color` attribute. Later we'll use its value as a fallback in case the corresponding image fails to load.

The next step is to initialize the carousel via JavaScript and modify the predefined values of the `interval` and `pause` configuration properties. Take note that we choose to set the value of the `pause` property to `false` because we always want the cycling to be active:

```
$('.carousel').carousel({
  interval: 6000,
  pause: "false"
});
```

Having followed those simple steps (and of course imported the required files), we should now be able to build the first version of the carousel. Here's how it looks so far:



2-1. Bootstrap Carousel structure

Live Code

See the Pen Basic Bootstrap Carousel.

Creating Full-screen Bootstrap Carousel Slides

At this point we'll go one step further, converting the existing carousel into a full-screen Bootstrap Carousel slideshow. To implement this updated version we have to add some custom jQuery:

```
var $item = $('.carousel-item');
var $wHeight = $(window).height();

$item.height($wHeight);
$item.addClass('full-screen');
```

```
$('.carousel img').each(function() {
  var $src = $(this).attr('src');
  var $color = $(this).attr('data-color');
  $(this).parent().css({
    'background-image' : 'url(' + $src + ')',
    'background-color' : $color
  });
  $(this).remove();
});

$(window).on('resize', function (){
  $wHeight = $(window).height();
  $item.height($wHeight);
});
```

Next, we add some CSS:

```
.full-screen {
  background-size: cover;
  background-position: center;
  background-repeat: no-repeat;
}
```

In the code above, we do the following:

- Loop through our images and get the values of the `src` and `data-color` attributes.
- Find their direct parent (`.item`) and assign the `full-screen` class along with a few background-related properties to it. Keep in mind that the values of those properties depend on the values of the aforementioned attributes (which are different for each image).
- Set the height of the parent element equal to the viewport height. It's important to mention that we have to recalculate the height of the browser window, when it changes size (using the `resize` event).
- Remove the `img` elements which are not needed since we are relying on the backgrounds.

Although it's not required, let's make one last change. When the page loads, we'll add the `active` class to the first slide using jQuery, rather than having it hard-coded in the HTML:

So instead of this:

```
<!-- Wrapper for slides -->
<div class="carousel-inner">
  <div class="carousel-item active">
```

We do this:

```
$item.eq(0).addClass('active');
```

This allows us to prevent a small toggle that is possible to occur between the two different heights (before and after the changes we made) of the first slide.

Here's the new version of our slideshow:

Live Code

See the Pen Bootstrap Carousel Full Screen.

Initializing a Random Slide

Sometimes we might want to show a random slide when the page loads. To build this new functionality, we should update our code. To begin with, we have to remove the default `active` class from the first slide as well as the first indicator.

Here's the original code:

```
<!-- etc... -->
```

```
<li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>

<!-- etc... -->

<!-- Wrapper for slides -->
<div class="carousel-inner">
  <div class="carousel-item active">

<!-- etc... -->
```

Which now becomes:

```
<!-- etc... -->

<li data-target="#carouselExampleIndicators" data-slide-to="0"></li>

<!-- etc... -->

<!-- Wrapper for slides -->
<div class="carousel-inner">
  <div class="carousel-item">

<!-- etc... -->
```

If you're using jQuery to add the `active` class to the first slide, as I explained earlier, you'd need to remove that code after adding the new random slide code.

We can now add the following code snippet to the existing jQuery:

```
var $numberofSlides = $('.carousel-item').length;
var $currentSlide = Math.floor((Math.random() * $numberofSlides));

$('.carousel-indicators li').each(function(){
  var $slideValue = $(this).attr('data-slide-to');
  if($currentSlide == $slideValue) {
    $(this).addClass('active');
    $item.eq($slideValue).addClass('active');
  } else {
    $(this).removeClass('active');
    $item.eq($slideValue).removeClass('active');
  }
});
```

- We start by getting the total number of slides and use this value to find a random slide.
- We iterate through the carousel indicators and retrieve the value of the `data-slide-to` attribute. If the random number matches the value of this attribute, we assign the `active` class to the corresponding slide and indicator. If that doesn't happen, we remove it from the target elements.

Of course, if you don't want to combine this behavior with the full-page slideshow, feel free to remove the unnecessary code.

You can see this functionality in the following CodePen demo:

Live Code

See the Pen Full-screen Bootstrap Carousel with Random Initial Image.

If you click the *RERUN* button on the CodePen embed, you'll see the initial image change on each load.

Potential Further Customizations?

Beyond the extensions presented in this article, there are many other ways to change the predefined behavior of the carousel component. If you feel adventurous, here are a few other ideas you can examine:

- create additional animation effects (e.g. `fade` or `scale`) when switching between slides

- build an image overlay
- randomize the next and previous slides (use the `slide.bs.carousel` custom event).

Conclusion

In this article, I've shown you how to add more flexibility to your Bootstrap projects by customizing the carousel component. Although not everyone loves carousels useful any more, you might have a client ask for such a feature and maybe these customizations will come in handy.

Chapter

3

Animating Bootstrap Carousels with GSAP's Animation Library

George Martsoukos

In the previous chapter, I covered the process of converting a Bootstrap carousel into a full-screen carousel with a random initial image. In this article, I'll build on that and cover the art of animating Bootstrap carousels, drawing on the assistance of GSAP (GreenSock Animation Platform), a popular JavaScript library.

Live Code

Before going any further, let's look at what we'll be building.

Building the Carousel

Be sure to include Bootstrap and jQuery (Bootstrap's JavaScript components require it) in your page — for example, from a CDN:

```
<!DOCTYPE HTML>
<html>
  <head>
    <meta charset="utf-8">
    <title>Using GSAP for Animating Bootstrap Carousels</title>
    <link rel="stylesheet" type="text/css" href="https://maxcdn.bootstrapcdn.com/b
    ↪ootstrap/4.0.0/css/bootstrap.min.css">
  </head>
  <body>
    ...
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.
    ↪js"></script>
  </body>
</html>
```

The basic structure of our carousel looks like this:

```
<div id="mycarousel" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
```

```
      <li data-target="#mycarousel" data-slide-to="0" class="active"></li>
      <li data-target="#mycarousel" data-slide-to="1"></li>
    </ol>

    <div class="carousel-inner">

      <!-- first slide -->
      <div class="carousel-item first active" id="1">
        <!-- content here -->
      </div>

      <!-- second slide -->
      <div class="carousel-item second" id="2">
        <!-- content here -->
      </div>

    </div><!-- /carousel-inner -->

  </div>
```

As you can see, it contains two slides. The first slide has a class of `first` and an ID of `1`, while the second one has a class of `second` and an ID of `2`.

Regarding their styles:

- we set their height equal to the viewport height
- we give them different background colors.

The associated CSS rules:

```
.item {
  height: 100vh;
}

.first {
  background: #D98F4F; /*orange*/
}
```

```
.second {
  background: #2c9cae; /*lightblue*/
}
```

This should be enough to give us a working carousel.

Building the First Slide

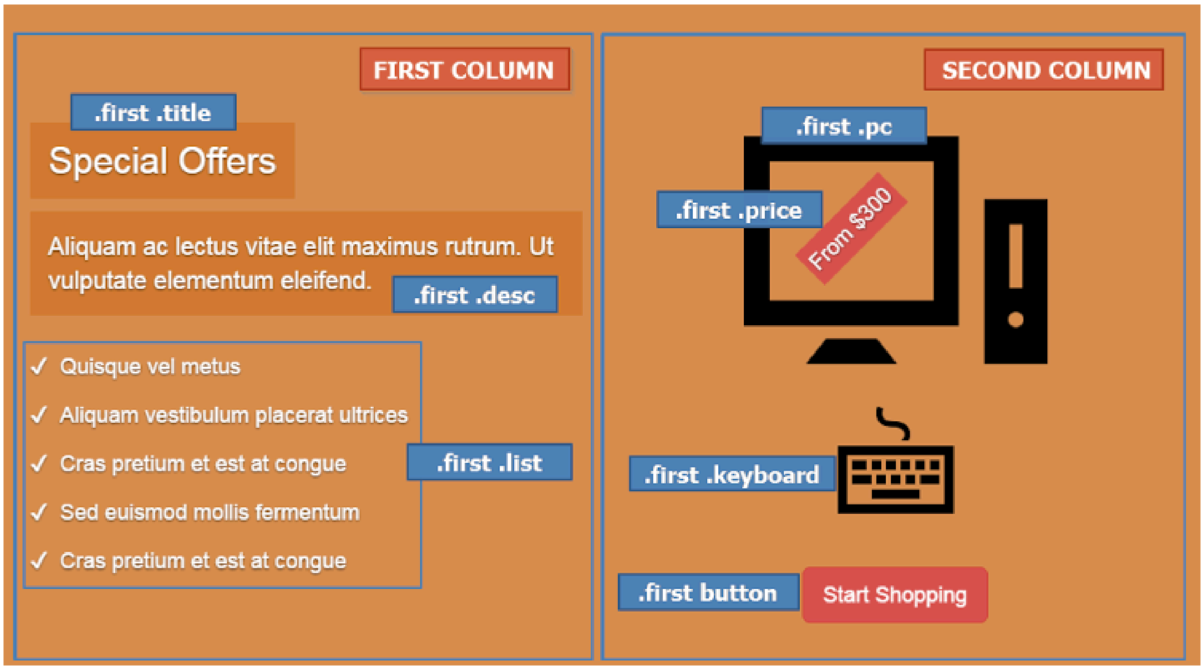
Next, we take advantage of Bootstrap's helper classes (e.g. grid classes) to set up the contents for our slides.

The markup for the first slide is the following:

```
<div class="carousel-item first active" id="1">
  <div class="carousel-caption">
    <div class="container">
      <div class="row justify-content-md-center">

        <div class="col">
          <h2 class="title">
            <!-- content here -->
          </h2>
          <p class="desc">
            <!-- content here -->
          </p>
          <ul class="list">
            <!-- list items here -->
          </ul>
        </div>

        <div class="col">
          <div class="pc-wrapper">
            <img class="pc" src="IMG_PATH" alt="" width="" height="">
            <div class="price">
              <!-- content here -->
            </div><!-- /price -->
          </div><!-- /pc-wrapper -->
          <img class="keyboard" src="IMG_PATH" alt="" width="" height="">
          <button type="button" class="btn btn-danger btn-lg">
            <!-- content here -->
          </button>
        </div>

      </div><!-- /row -->
    </div><!-- /container -->
  </div><!-- /carousel-caption -->
</div><!-- /carousel-item -->
```

If you're following along, be sure to replace `IMG_PATH` with something sensible.

Here's what the result looks like:


FIRST COLUMN
.first .title
Special Offers
Aliquam ac lectus vitae elit maximus rutrum. Ut vulputate elementum eleifend.
.first .desc
✓ Quisque vel metus
✓ Aliquam vestibulum placerat ultrices
✓ Cras pretium et est at congue
✓ Sed euismod mollis fermentum
✓ Cras pretium et est at congue
.first .list
SECOND COLUMN
.first .pc
.first .price
From $300
.first .keyboard
.first button
Start Shopping

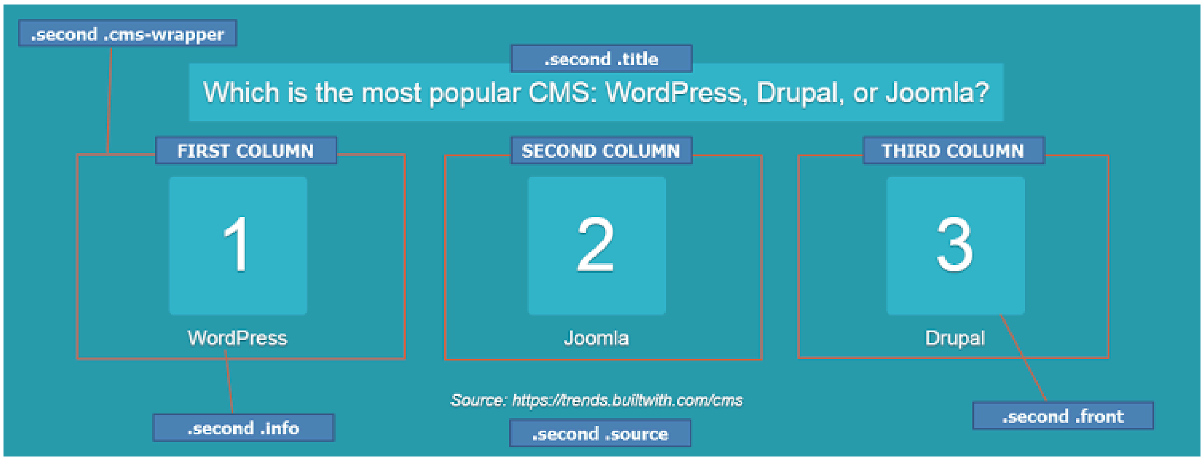
3-1. Animating Bootstrap Carousels: the first slide of the carousel

Building the Second Slide

In the same way, here's the markup for the second slide:

```
<div class="carousel-item second" id="2">
  <div class="carousel-caption">
    <div class="container">

      <h2 class="title">
        <span>
          <!-- content here -->
        </span>
      </h2>

      <div class="row justify-content-md-center">

        <div class="col cms-wrapper">
          <div class="cms">
            <div class="front">
              <!-- content here -->
            </div>
            <div class="back">
              <img class="img-fluid" src="IMG_PATH" alt="">
            </div><!-- /back -->
          </div><!-- /cms -->
          <p class="info">
            <!-- content here -->
          </p>
        </div><!-- /cms-wrapper -->

        <!-- two more columns here -->

      </div><!-- /row -->

      <div class="source">
        <!-- content here -->
      </div><!-- /source -->

    </div><!-- /container -->
  </div><!-- /carousel-caption -->
</div><!-- /carousel-item -->
```

And its visualization:



3-2. Animating Bootstrap Carousels: the second slide of the carousel

Keeping Things Simple

For simplicity, we won't extensively cover the styles for the inner parts of our slides. We'll only refer to the styles that are important for the animations.

Initializing the Carousel

Next we initialize the carousel and disable default autoplay by passing `interval:false` to the configuration object:

```
var $carousel = $("#mycarousel");

$carousel.carousel({
  interval: false
});
```

Adding Keyboard Navigation

By default, Bootstrap carousel isn't compliant with accessibility standards. In our case, though, let's make the carousel a bit more accessible by adding keyboard navigation.

Here's the required code:

```
$(document).keyup(function(e) {
  // right arrow
  if(e.which === 39) {
    $carousel.carousel("next");
  "next"// left arrow
  } else if(e.which === 37) {
    $carousel.carousel("prev");
  }
});
```

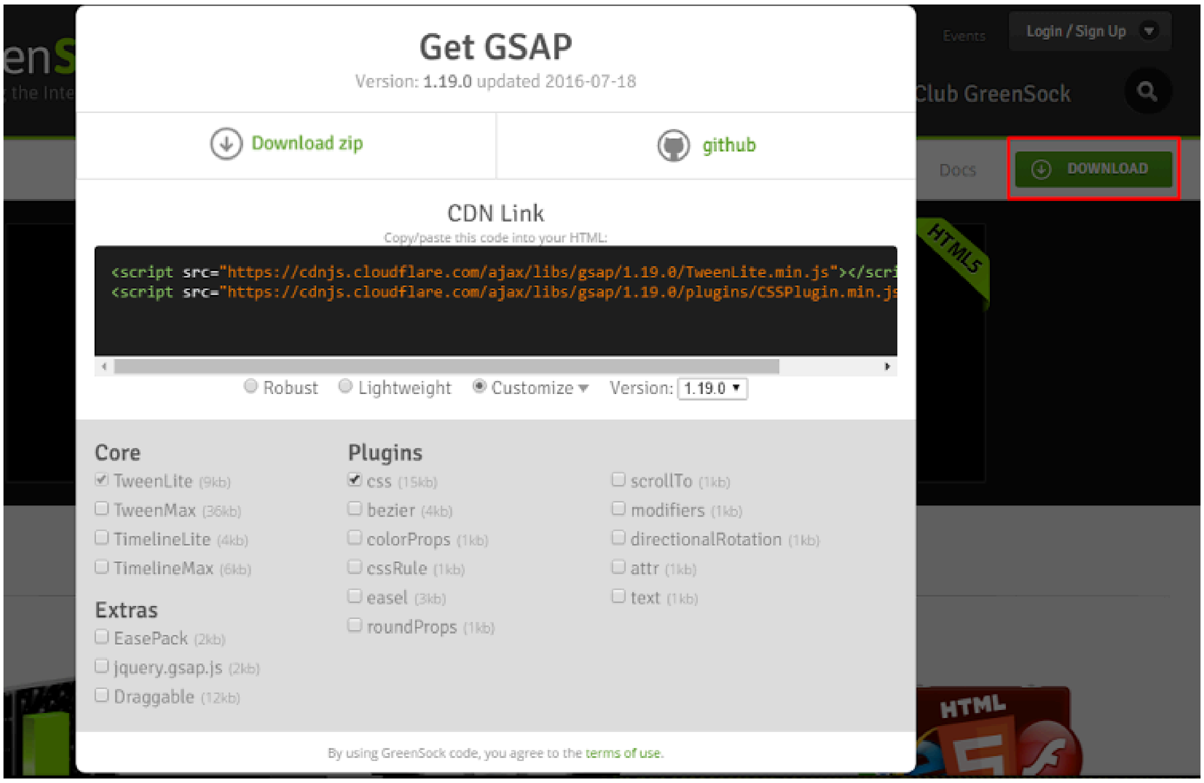
So far, we've built a basic carousel which supports keyboard navigation.

Animating Bootstrap Carousels: First Animations

At this point, let's try to make the carousel more appealing by adding some animations. To achieve this, we'll take advantage of GSAP, one of the most powerful JavaScript animation libraries out there. If you're looking for a thorough introduction to GreenSock, check out GreenSock for Beginners: a Web Animation Tutorial (Part 1).

Getting Started With GSAP

To incorporate GSAP into our projects, we have to visit its site and from there *click the download* button, which appears in the top-right corner of the page. This will open a modal dialog with a link to the project on a CDN.



Get GSAP

Version: 1.19.0 updated 2016-07-18

CDN Link

Copy/paste this code into your HTML:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/1.19.0/TweenLite.min.js"></scri
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/1.19.0/plugins/CSSPlugin.min.js
```

Robust [✓ Customize] Lightweight Version: 1.19.0

Core
- [✓] TweenLite (9kb)
- [ ] TweenMax (36kb)
- [ ] TimelineLite (4kb)
- [ ] TimelineMax (6kb)

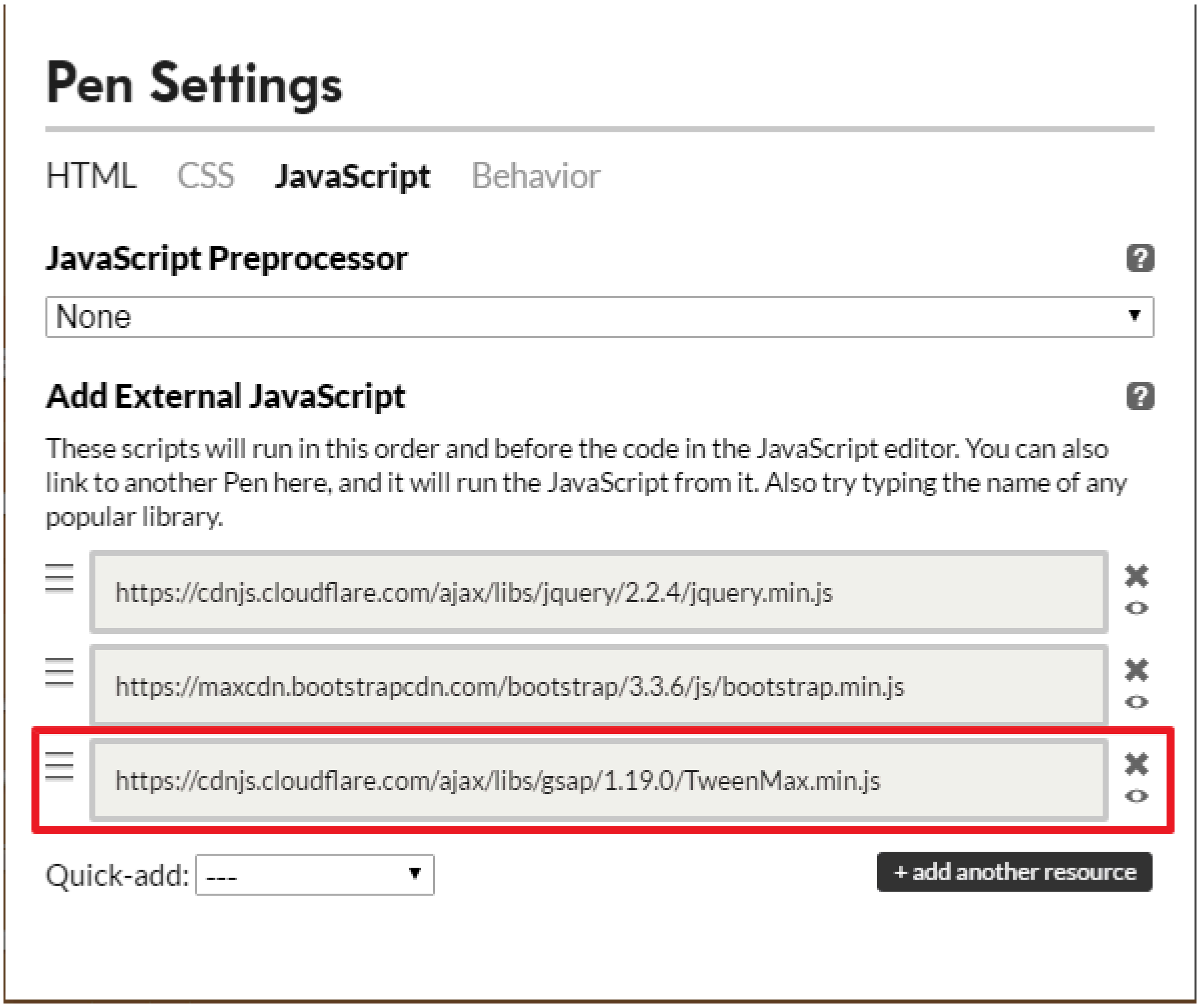
Extras
- [ ] EasePack (2kb)
- [ ] jquery.gsap.js (2kb)
- [ ] Draggable (12kb)

Plugins
- [✓] css (15kb)
- [ ] bezier (4kb)
- [ ] colorProps (1kb)
- [ ] cssRule (1kb)
- [ ] easel (3kb)
- [ ] roundProps (1kb)
- [ ] scrollTo (1kb)
- [ ] modifiers (1kb)
- [ ] directionalRotation (1kb)
- [ ] attr (1kb)
- [ ] text (1kb)

By using GreenSock code, you agree to the terms of use.

3-3. How to download GSAP

If we then select the *Customize* radio button, we can select the parts of the library we want to use. For our project, however, we'll keep things simple and include just the full robust version of it.


Pen Settings

JavaScript Preprocessor

None

Add External JavaScript

These scripts will run in this order and before the code in the JavaScript editor. You can also link to another Pen here, and it will run the JavaScript from it. Also try typing the name of any popular library.

```
https://cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js
https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js
https://cdnjs.cloudflare.com/ajax/libs/gsap/1.19.0/TweenMax.min.js
```

3-4. How to include GSAP into our Codepen demo

Remember that we had to add jQuery to our project because Bootstrap's carousel depends on it. But, keep in mind that GSAP is a pure JavaScript library, and thus it doesn't require it.

Animating Bootstrap Carousels: the First Slide

By default, we want the content of our slides to be hidden:

```
.carousel-caption {
  opacity: 0;
}
```

Only when the page loads do we reveal and animate the contents of the first slide. To do this, we take advantage of TimelineLite, an animation tool that helps us build a sequence of tweens.

So, as soon as all page assets are ready, the `firstTimeline` function is executed:

```
// this variable stores the first timeline
var firstTl;

$(window).on("load", function() {
  firstTl = firstTimeline();
});
```

This function returns a timeline which determines the animations for the first slide:

```
function firstTimeline() {
  var tl = new TimelineLite();

  tl
    .to(".first .carousel-caption", 0.1, {opacity: 1})
    .from(".first .pc", 1, { y: -300, opacity: 0, ease: Bounce.easeOut})
    .from(".first .keyboard", 1, { y: 300, opacity: 0, ease: Bounce.easeOut}, "-=1")
    .staggerFrom(".first .list li", 0.75, {opacity: 0, cycle: {x: [-200, 200]},
    ↪ease: Power2.easeOut}, 0.15, "-=0.5")
    .from(".first .desc", 0.7, {x: 500, opacity: 0, ease: Power4.easeOut}, "-=0.5")
    .from(".first .title", 0.7, {x: -500, opacity: 0, ease: Power2.easeOut},
    ↪"-=0.7")
    .from(".first .price", 0.7, {scale: 0.01, ease: Power4.easeOut})
    .from(".first button", 0.7, { y: -700, autoAlpha: 0, ease: Bounce.easeOut},
    ↪"-=0.3");

  return tl;
}
```

More specifically, inside the function above we do the following:

1. create a TimelineLite
2. add tweens to the timeline using its to, from, and staggerFrom methods
3. return the timeline.

Let's take note of the parameters that we pass to the `to` and `from` methods:

1. The DOM element that we want to animate.
2. The duration of the animation in seconds.
3. An object containing the properties that should be tweened and their respective (starting or end) values. Moreover, this object can also have some other special properties like the `ease` property which defines the easing function.
4. The placement of the tween in the timeline. In other words, when this tween should be executed. For example, we want to run the animations for the `.first .pc` and `.first .keyboard` elements at the same time. To do this, we set the value of the `position` parameter of the `.first .keyboard` element to `"-=1"`. The number "1" inside this value matches the duration of the animation of the `.first .pc` element.

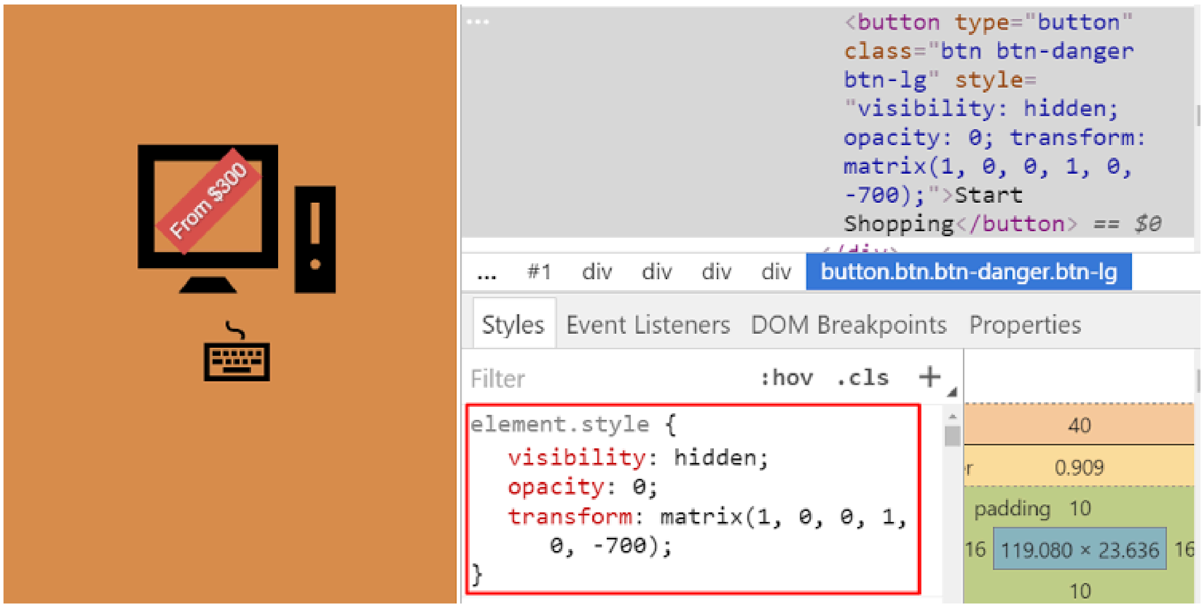
Similar to the `to` and `from` methods, we pass the same parameters to the `staggerFrom` method. The only difference is that we define one additional parameter (the fourth one) which specifies each tween's start time. In our example, this value is set to `0.15`. That said, the target elements won't appear simultaneously but there will be a very small gap between their animations.

Change this value to something big (e.g. 5), to see a clear difference in the effect.

Obviously, the best way to understand how the animations above work is to read the documentation. Plus, use the developer tools of your browser to see what styles GSAP applies to the target elements.

So let's briefly have a closer look at the animations assigned to the button.

Initially the button has `autoAlpha: 0` and `y: -700`. That means it's visually hidden (`opacity: 0`, `visibility: hidden`) and located 700 pixels up from its original position (`transform: matrix(1, 0, 0, 1, 0, -700)`).

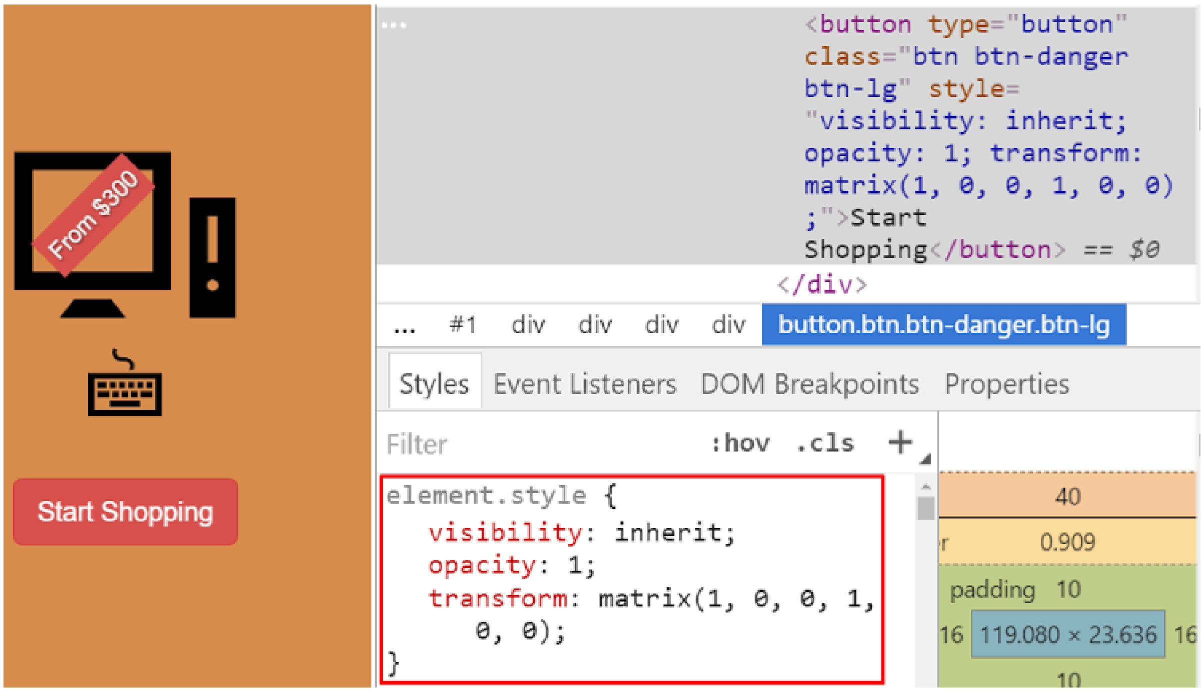


```
<button type="button" class="btn btn-danger btn-lg" style="visibility: hidden; opacity: 0; transform: matrix(1, 0, 0, 1, 0, -700);">Start Shopping</button> == $0
```

```
element.style {
  visibility: hidden;
  opacity: 0;
  transform: matrix(1, 0, 0, 1,
      0, -700);
}
```

3-5. Animating Bootstrap Carousels: which styles initially GSAP applies to the button before animations

Then, when the animation plays, it becomes visible (`opacity: 1`, `visibility: inherit`) and returns back to its default position (`transform: matrix(1, 0, 0, 1, 0, 0)`).


From $300
Start Shopping
#1 div div div div button.btn.btn-danger.btn-lg
Styles Event Listeners DOM Breakpoints Properties
Filter :hov .cls +
40
0.909
padding 10
16 119.080 × 23.636 16
10

```
<button type="button" class="btn btn-danger btn-lg" style="visibility: inherit; opacity: 1; transform: matrix(1, 0, 0, 1, 0, 0);">Start Shopping</button> == $0
</div>
```

```
element.style {
   visibility: inherit;
   opacity: 1;
   transform: matrix(1, 0, 0, 1,
      0, 0);
}
```

3-6. Animating Bootstrap Carousels: which styles finally GSAP applies to the button after animations

Finally, the animation starts 0.3 seconds before the end of the timeline.

Determining Duration

Use TimelineLite's `duration` method to retrieve its duration. The key here is to understand how this value is calculated.

Live Code

Great job so far! The animations for the first slide are ready! You can see them live in this Codepen demo: Using GSAP to Animate Bootstrap Carousels (Part 1).

Using Bootstrap's Carousel Events

As we leave the first slide and move on to the second one (and vice versa), the contents of our slides should be hidden. It should appear as soon as the carousel completes its slide transition. To accomplish this behavior, we take advantage of the following things:

1. The `slide.bs.carousel` and `slid.bs.carousel` events that Bootstrap provides.
2. GSAP's TweenLite animation tool that helps us build a single tween. Note that in the previous section we used a TimelineLite to create a sequence of tweens.

Here's the necessary code:

```
var $carouselCaption = $(".carousel-caption");

$carousel.on("slide.bs.carousel", function() {
  TweenLite.to($carouselCaption, 0.1, {opacity: 0});
});

$carousel.on("slid.bs.carousel", function () {
  TweenLite.to($carouselCaption, 0.1, {opacity: 1});
});
```

As we've discussed earlier, when the page loads, the animations for the first slide run. But when should they run again? Plus, what about the second slide and its animations? To answer all these questions, let's add a few lines of code to the callback of the `slid.bs.carousel` event:

```
// these variables store the timelines
var firstTl, secondTl;

$carousel.on("slid.bs.carousel", function (e) {
  TweenLite.to($carouselCaption, 0.1, {opacity: 1});
  var slideId = e.relatedTarget.id;
  if(slideId === "1") {
    firstTl.restart();
  } else if(slideId === "2") {
    secondTl = secondTimeline();
  }
});
```

The code above does the following:

1. It checks the `id` of our slides.
2. If the `id` is equal to `1`, the first slide has been loaded, and thus we've got to replay the animations for this slide. Remember that we've already created a timeline for these animations (within the `firstTimeline` function), so we can use its `restart` method to play it again.
3. If the `id` is equal to `2`, the second slide has been loaded, and thus the `secondTimeline` function is executed (see next section).

Animating Bootstrap Carousels: the Second Slide

The `secondTimeline` function returns a timeline which determines the animations for the second slide:

```
function secondTimeline() {
  var tl = new TimelineLite({onComplete: allDone});

  tl
    .from(".second .title", 0.5, { y: -400, opacity: 0, ease: Sine.easeInOut})
    .staggerFrom(".second .cms-wrapper", 0.5, {scale: 0, rotation: 180, ease:
        ➥Power2.easeInOut, onComplete: completeFunc}, 1.2);

  return tl;
}
```

More specifically, inside the function above we do the following:

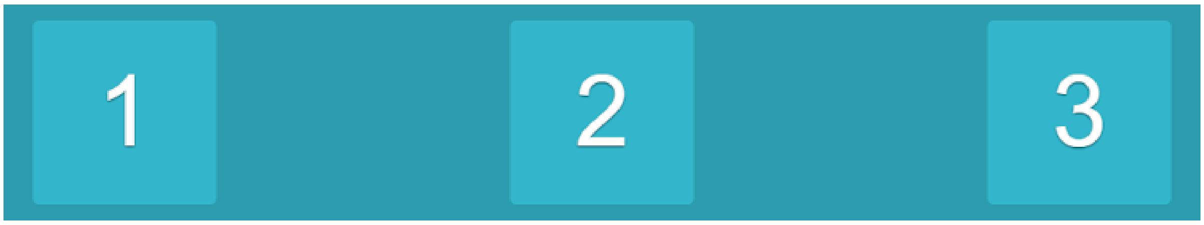
1. create a TimelineLite
2. add tweens to the timeline using its `from` and `staggerFrom` methods
3. return the timeline.

Although this timeline looks like the one we've previously created for the first slide, there are some things here that we haven't seen before.

First, look at the object that we pass to the `staggerFrom` method. This contains a special property called `onComplete`. This property holds a function (i.e. `completeFunc()`) that's fired when the animations for each of the target elements finish.

Let me explain.

By default, only the elements with the class of `.second .front` (i.e. numbers) are visible.



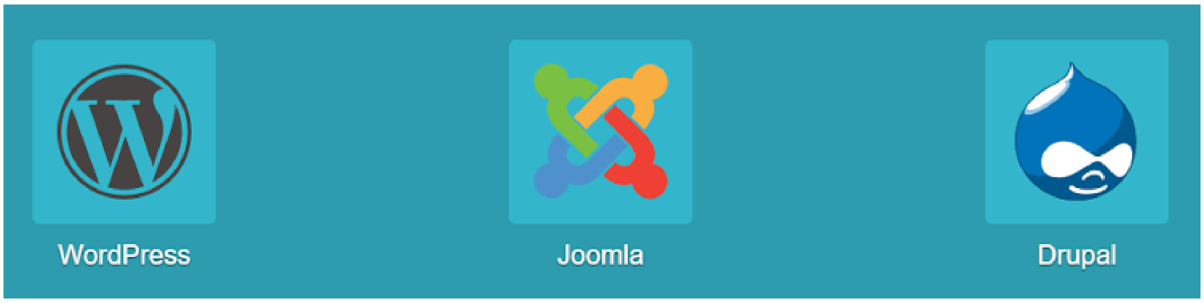
3-7. How the cms-wrapper element of the second slide initially looks like before animations

And the `.second .back` (i.e. CMS logos) and `.second .info` (i.e. CMS names) elements are hidden.

Here are the corresponding CSS rules:

```
.second .back {
  display: none;
  transform: scale(0);
}

.second .info {
  opacity: 0;
  transform: translateY(40px);
}
```

When the `completeFunc` function runs for each of the `.second .cms-wrapper` elements, we set up tweens that affect the visibility of the child elements. Specifically, we hide the `.second .front` element and then show (after 0.3 seconds delay) the `.second .back` and `.second .info` elements.



3-8. What the cms-wrapper element of the second slide looks like after animations

Below you can see the related JS code:

```
function completeFunc() {
  var $this = $(this.target),
      $info = $this.find(".info"),
      $front = $this.find(".front"),
      $back = $this.find(".back");

  TweenLite.to($front, 0.3, {display: "none", scale: 0});
  TweenLite.to($back, 0.3, {display: "block", scale: 1, delay: 0.3});
  TweenLite.to($info, 0.3, {opacity: 1, y: 0, delay: 0.3});
}
```

The second new thing in this timeline is the callback that runs when all animations finish. We specify this by passing a configuration object with an `onComplete` property (whose value is the `allDone` function) to the constructor function.

When the timeline completes, this function is triggered and shows the `.second .source` element which was initially invisible (`opacity: 0`, `visibility: hidden`).

Source: https://trends.builtwith.com/cms

3-9. Animating Bootstrap Carousels: How the source element of the second slide looks like after animations

Here's the `allDone` function:

```
function allDone() {
  TweenLite.to(".second .source", 0.3, {autoAlpha: 1, delay: 1});
}
```

Live Code

At this point, let's examine the current state of our carousel: Using GSAP to Animate Bootstrap Carousels (Part 2).

The animations look good but there's still a small problem with regards to the second slide. To be more specific, the first time we visit the second slide, the animations work as expected. However, in any other case (test it), the result isn't the desired one, as the CMS logos don't animate in.

To fix this issue, we have to restart the corresponding timeline (remember we did the same for the first slide) and clear the tweens (inline styles) defined in the `completeFunc` and `allDone` functions. Lastly, we pause the second timeline when the first one is active. With that in mind, once again we update the callback of the `slid.bs.carousel` event like this:

```
var counter = 0,
    firstTl,
    secondTl;

$carousel.on("slid.bs.carousel", function (e) {
  TweenLite.to($carouselCaption, 0.1, {opacity: 1});
  var slideId = e.relatedTarget.id;

  if(slideId === "1") {
    firstTl.restart();
    secondTl.pause();
  } else if(slideId === "2") {
    if(counter === 0) {
      secondTl = secondTimeline();
    } else {
      TweenLite.set([".second .front", ".second .back", ".second .info",
      ➥".second .source"], {clearProps:"all"});
      secondTl.restart();
    }
    counter++;
  }
 });
```

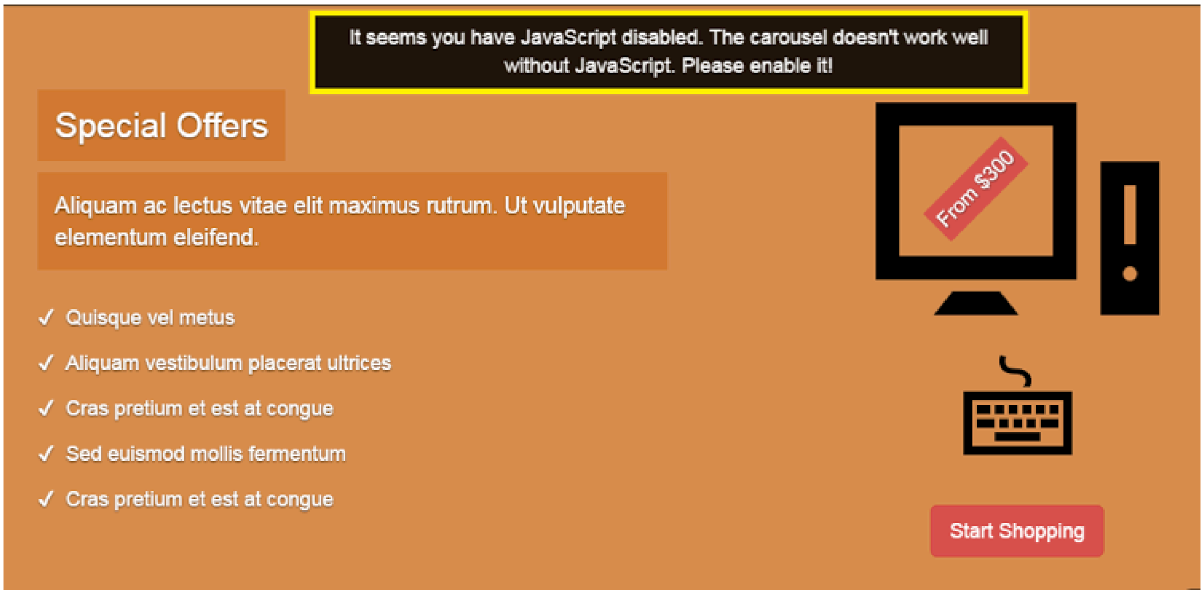
Live Code

So, finally, here's the new version of our carousel: Using GSAP to Animate Bootstrap Carousels (Part 3).

Customizing the Carousel When JavaScript is Disabled

The carousel is almost ready. Before wrapping things up, let's customize its appearance when JavaScript is disabled. In such a scenario, the slides have to appear below each other and a custom message should encourage users to enable JavaScript.

Here's the desired visualization:


It seems you have JavaScript disabled. The carousel doesn't work well without JavaScript. Please enable it!

Special Offers

Aliquam ac lectus vitae elit maximus rutrum. Ut vulputate elementum eleifend.

✓ Quisque vel metus
✓ Aliquam vestibulum placerat ultrices
✓ Cras pretium et est at congue
✓ Sed euismod mollis fermentum
✓ Cras pretium et est at congue

3-10. What the carousel looks like with JavaScript disabled

One way to accomplish this behavior is by using the `<noscript>` tag. This option is great, yet it adds some additional code to our HTML, so let's try something a little bit different.

By default, we add the `no-js` class to the `<html>` tag. As the browser parses the page, if JavaScript is enabled in this browser, the class is removed.

So first, in the HTML we add this code:

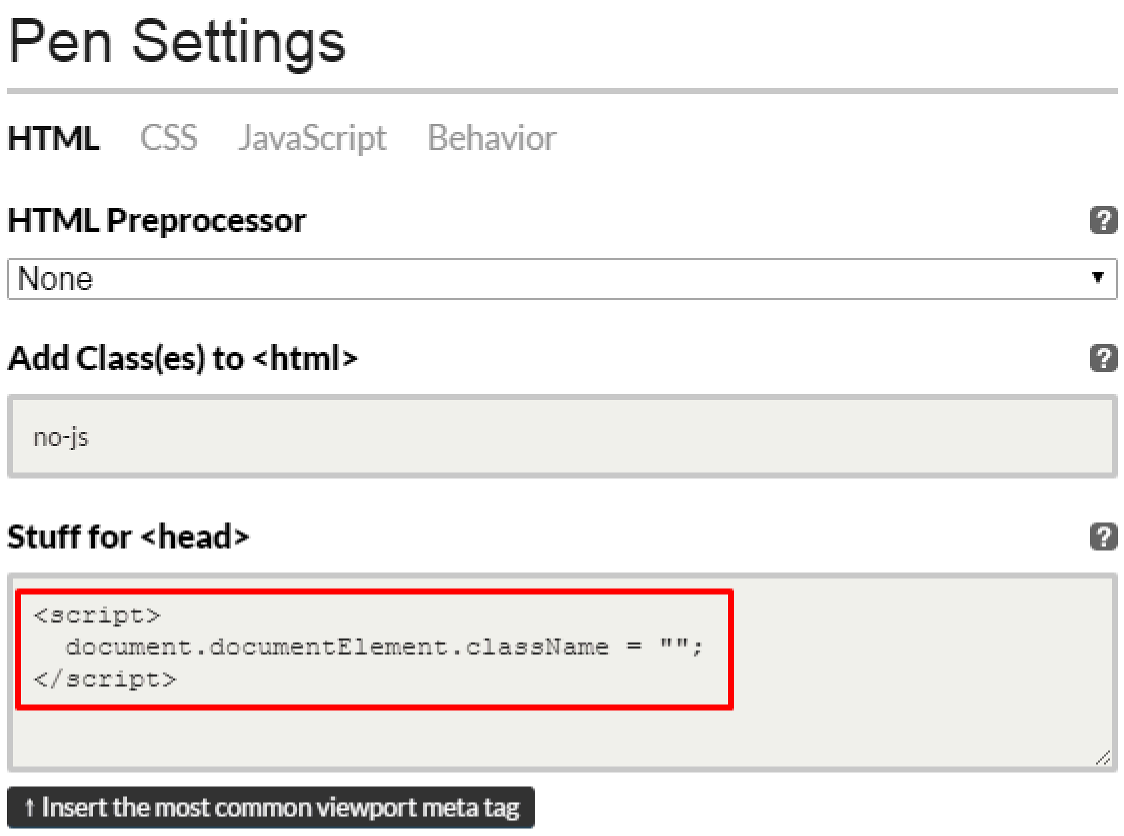
```
<p class="msg">
  It seems you have JavaScript disabled. The carousel doesn’t work well without
  ➥JavaScript. Please enable it!
</p>
```

Then, in the CSS this one (mostly reset rules):

```
.msg {
  display: none;
  position: fixed;
  top: 5px;
  left: 50%;
  transform: translateX(-50%);
  padding: 7px;
  border: 5px solid #fff000;
  text-align: center;
  color: #fff;
  background: rgba(0, 0, 0, .85);
  z-index: 999;
}

.no-js .carousel-inner > .item {
  display: block;
}

.no-js .carousel-caption,
.no-js .second .info,
.no-js .second .source {
  opacity: 1;
}

.no-js .second .info {
  transform: none;
 }

.no-js .second .source {
  visibility: visible;
}

.no-js .carousel-indicators {
  display: none;
}

.no-js .msg {
  display: block;
}
```

Finally, in the `<head>` tag of our page we insert the following code snippet:

```
<script>
  document.documentElement.className = "";
</script>
```



HTML Preprocessor
None

Add Class(es) to <html>
no-js

Stuff for <head>

```
<script>
  document.documentElement.className = "";
</script>
```

3-11. Removing the no-js class from the HTML element

Keep in mind that we place this code within the `<head>` tag because we want it to be executed before the browser starts to paint the elements. This is important in order to prevent DOM flickering.

Here's the final version of our carousel:

Live Code

See the Pen Bootstrap Carousel with GSAP Animations.

Conclusion

In this article, we went through the process of animating Bootstrap carousels with GSAP. You can find all the demos for this article in this Codepen collection. Without doubt, we covered a lot of ground but hopefully you enjoyed what we built and got an idea of GSAP's power! Happy tweening!

Chapter

4

Build a Simple Tumblr Theme with Bootstrap

Ashraff Hathibelagal

If you've ever looked at all the wonderful Tumblr themes and wondered what it takes to build a Tumblr theme from scratch, then this tutorial is for you. In this post, I'm going to show you how to use Bootstrap and Tumblr's special operators to create a theme you can use for your Tumblr blog.

Here's what you need to get started:

- a Tumblr account
- a basic understanding of Bootstrap, the popular front-end framework.

Our Basic HTML and Bootstrap Resources

A Tumblr theme is just an HTML file that uses Tumblr's special operators. We'll start off by creating a new file using our favorite text editor and adding the following boilerplate HTML code to it:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
  </head>
  <body>
  </body>
</html>
```

Next, we'll add the necessary Bootstrap resources. Bootstrap CDN makes it easy to add the files to your Tumblr theme. Just put the following in your theme's `head`:

```
<!-- Bootstrap CSS -->
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
rel="stylesheet">
```

Also add the following before the closing `body` tag:

```
<!-- Bootstrap JS -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js">
</script>
```

Adding Custom Styles

This Tumblr theme is going to have a narrow layout that will be 550px wide. It's also going to have a 50px-tall, fixed-position header. Add the following to the theme's `head` to handle these requirements:

```
<style type="text/css">
  body {
    padding-top: 60px;
  }

  .container {
    width: 550px;
  }
</style>
```

Creating the Tumblr Theme Header

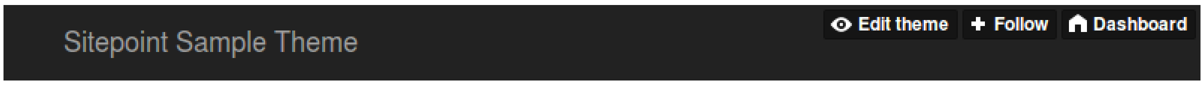
We'll use Bootstrap's `navbar` to create a header for our Tumblr theme. This allows us to use our first Tumblr variable: `{Title}`. The `{Title}` variable refers to the title of the blog. The following code is added to the `body` of the theme:

```
<nav class="navbar fixed-top navbar-dark bg-dark">
  <a class="navbar-brand" href="#">{Title}</a>
</nav>
```

It's a good idea to use the `{Title}` variable in the HTML `<title>` tag as well:

```
<title>{Title}</title>
```

If you were to apply the theme to your Tumblr blog now, it would look like this, with just white space below it:



4-1. Tumblr theme header

Creating a Container to Hold Posts

Next we'll add a `div` that we can use as a container for all Tumblr posts. Add the following code below the `nav` element:

```
<div class="container">
</div>
```

As you might already know, Tumblr lets you add different types of posts to your blog. For the sake of simplicity, we're going to handle the following post types:

- text
- photo
- quote

`{block:Posts}` is a Tumblr **block** that gives us a list of all the posts available on a blog.

Add the following code inside the posts container:

```
{block:Posts}
{/block:Posts}
```

Next, to determine the type of the current post, we're going to use the following blocks:

- `{block:Text}`

- `{block:Photo}`
- `{block:Quote}`

For any post, only one of these will be rendered.

Handling Text Posts

Text posts usually have a title and a body that can be accessed using the variables `{Title}` and `{Body}`, respectively. Add the following code inside the `{block:Posts}` tag.

```
{block:Text}
  <h2>{Title}</h2>
  <p>{Body}</p>
{/block:Text}
```

Handling Photo Posts

Photo posts have a caption and a photo. The caption can be accessed using the `{Caption}` variable. Accessing the photo is a little more complicated. To make it easier for you to use consistent image sizes all over your blog, Tumblr provides you with scaled versions of photos.

For example, to make sure your photo isn't wider than 500px, you could use the variable `{PhotoURL-500}`. Similarly, if you want to make sure the photo isn't wider than 100px, you could use the variable `{PhotoURL-100}`.

For this Tumblr theme, I'm going to use `{PhotoURL-500}`.

We can use a Bootstrap card to show the photo and its caption.

The code to handle photo posts would then look like this:

```
{block:Photo}
  <div class="card">
    <img class="card-img-top" src="{PhotoURL-500}" alt="Card image cap">
    <div class="card-body">
      <h5 class="card-title">{Caption}</h5>
    </div>
  </div>
{/block:Photo}
```

Handling Quote Posts

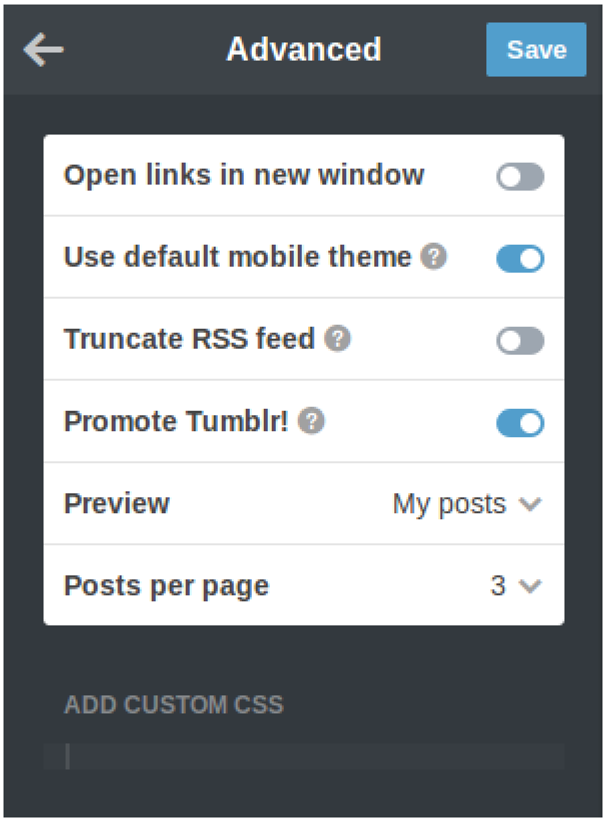
In this type of post, you'll have to use the intuitively named variables `{Quote}` and `{Source}`. Bootstrap applies its own styles to the HTML `<blockquote>` element with the `blockquote` class and the `blockquote-footer` class, so we can use this directly in our theme:

```
{block:Quote}
  <blockquote class="blockquote">
    <p>{Quote}</p>
    <footer class="blockquote-footer">{Source}</footer>
  </blockquote>
{/block:Quote}
```

Handling Pagination

At this point, your Tumblr theme will look complete if you apply it to your blog. However, Tumblr only shows a fixed number of posts at a time.

You can change this number in the **Advanced options** page. As you can see in the screenshot, I've set the **Posts per page** value to 3:



Open links in new window

Use default mobile theme

Truncate RSS feed

Promote Tumblr!

Preview My posts

Posts per page 3

ADD CUSTOM CSS

4-2. Tumblr Advanced options

Currently, if you add too many posts to your blog, the oldest ones will no longer be accessible. Therefore, you have to handle pagination in your theme.

Tumblr has two variables to help with this: `{PreviousPage}` and `{NextPage}`. The values of these variables are links, so you'll have to put them inside `<a>` tags. To make sure these links are shown only when there's a valid previous or next page, we use the `{PreviousPage}` and `{NextPage}` variables inside the `{block:PreviousPage}` and `{block:NextPage}` blocks, respectively.

We can use Bootstrap's button classes to style our pagination links.

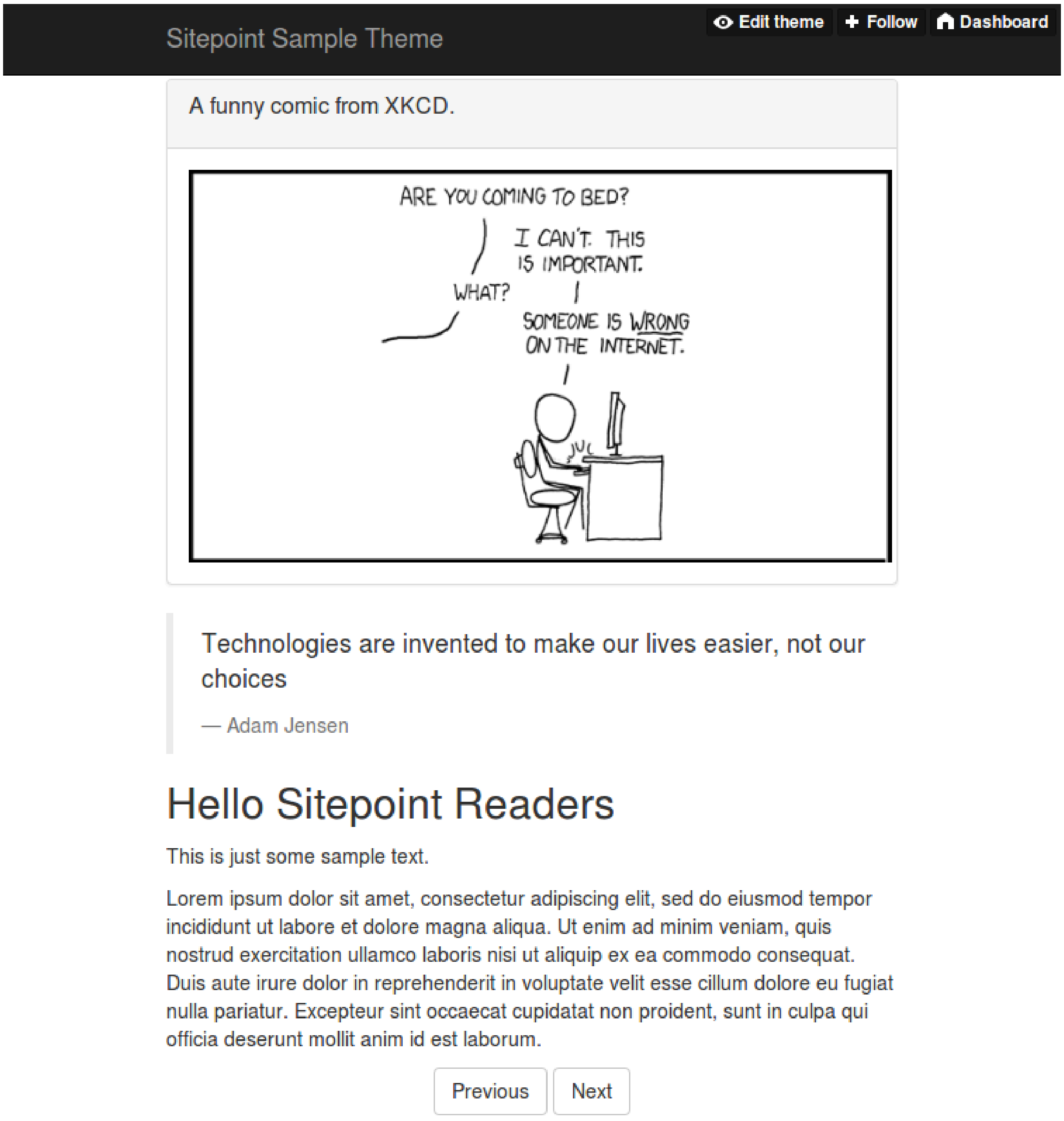
The following code will be added after the `{block:Posts}` block:

```
<p class="text-center">
  {block:PreviousPage}
    <a href="{PreviousPage}" class="btn btn-secondary">Previous</a>
  {/block:PreviousPage}
  {block:NextPage}
    <a href="{NextPage}" class="btn btn-secondary">Next</a>
  {/block:NextPage}
</p>
```

Applying the Theme to Tumblr

Visit your Tumblr dashboard, select a blog, and click *Customize*. In the next screen, click the *Edit HTML* link. You'll be presented with a text editor that will show the HTML contents of your current theme. Replace those contents with the code of the new Tumblr theme we just built. Press the *Update Preview* button first, then the *Save* button to apply the new theme.

Note: since we're using external links for the Bootstrap files, the preview might not look right. However, after pressing the *Save* button, when you visit your blog you should see the Bootstrap styles applied correctly.


Edit theme
Follow
Dashboard
ARE YOU COMING TO BED?
I CAN'T. THIS IS IMPORTANT.
WHAT?
SOMEONE IS WRONG ON THE INTERNET.

Sitepoint Sample Theme

A funny comic from XKCD.

Technologies are invented to make our lives easier, not our choices

— Adam Jensen

Hello Sitepoint Readers

This is just some sample text.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Previous Next

4-3. Tumblr Theme final

Conclusion

We now have a simple but complete Tumblr theme that uses the latest version of Bootstrap. You can always refer to Tumblr's documentation to extend this theme.

Chapter

5

How to Build a Responsive Bootstrap Website

Syed Fazle Rahman

In this article, we're going to build a responsive Bootstrap website from scratch. By the end of this article, you'll be familiar enough with the latest version of this popular CSS framework to be able to build your own variations of it according to your project's needs.

Building responsive websites is a must nowadays. People access websites from all kinds of devices, and Google has made a point of stressing the importance of responsive web design when it comes to assigning ranking in web page results.

Designing responsive websites from scratch can be taxing for beginners, although the latest Flexbox and CSS Grid specifications make achieving great results in this field much easier than it used to be.

However, for those who aren't ready to tackle cutting-edge layout techniques yet, the Bootstrap grid still offers an excellent alternative.

What "Responsive Bootstrap Website" Means

The first thing that comes to mind when we use the word "Responsive Design" is that websites should be compatible with all kinds of devices and screen sizes. There's a constant demand in the industry to make every website responsive for better readability of the online contents in different environments.

With the help of CSS3 and definitely HTML5, this is now a consolidated trend. *But what if you're a developer and not a designer? BONK!*

Well, you don't have to worry any more. Since Bootstrap is a **superhero** in the field of CSS frameworks, you'll be able to tackle responsive web design with its help in no time.

Setting Up

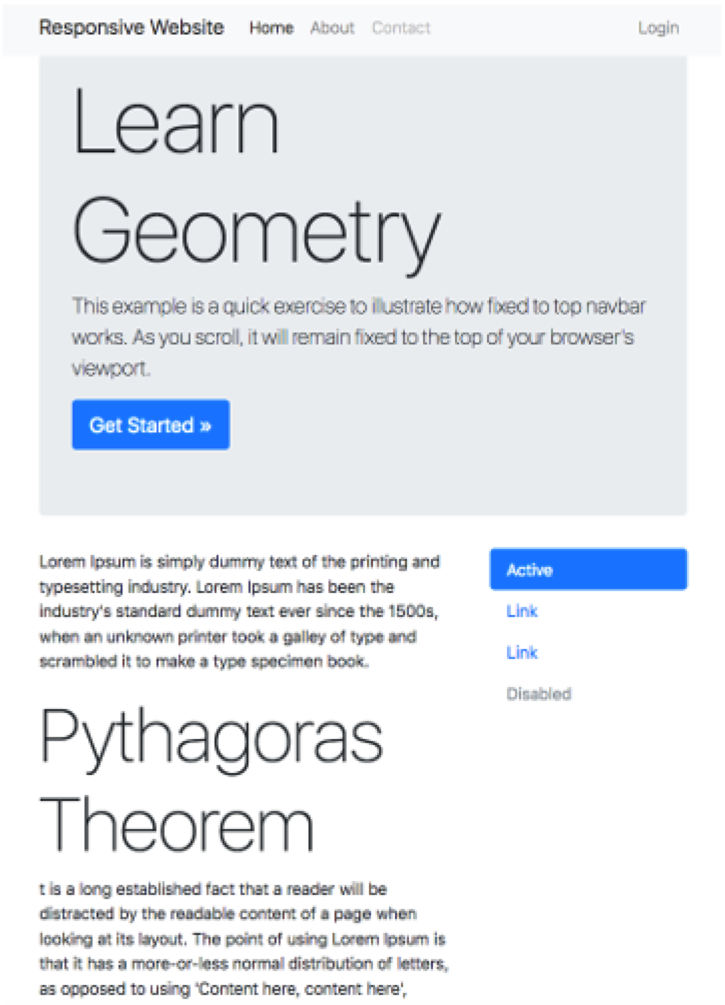
Ensuring you get a responsive Bootstrap website is as simple as placing the correct meta tag inside the head of your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

The above meta tag is quite self-explanatory in nature. We're setting the width of the page to the width of the device and initially scaling it to 1 — its default size.

Apart from this, you're good to go: Bootstrap is responsive by default.

```
<link rel="stylesheet" href="css/bootstrap.css">
<link rel="stylesheet" href="css/bootstrap-responsive.css">
```



This example is a quick exercise to illustrate how fixed to top navbar works. As you scroll, it will remain fixed to the top of your browser's viewport.

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

t is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here',

5-1. Tumblr Theme final

Live Code

Here's a demo of the page we're going to build. See the Pen Building Responsive Websites Using Bootstrap.

Let's Begin Building Our Responsive Bootstrap Website

I've divided the above responsive web page into different categories, and we'll see how to build each one of them in detail:

1. the responsive navigation
2. the marketing area
3. the contents section
4. the right sidebar
5. the footer

The Responsive Navigation

Let's build the navigation bar of the website. It will contain the website's title and some right-aligned menu link items. This is going to be fixed to the top of the website, as you've seen in the demo page. So here's the markup for this:

```
<nav class="navbar fixed-top navbar-expand-md navbar-light bg-light">
```

The `navbar` class is for showing the navigation section. An additional `fixed-top` class makes it stick to the top of the page. The `navbar-light` and `bg-light` classes control the text color and background color of the navigation bar respectively. Pretty clear!

Let’s move ahead and insert some more code into it:

```
<nav class="navbar fixed-top navbar-expand-md navbar-light bg-light">
  <div class="container">

    <!-- more navigation code here -->

  </div>
</nav>
```

The `container` is used to contain everything inside the navigation bar as a wrapper.

Till now, whatever we’ve added is just the basic structure of our navigation bar. Let’s see the real magical stuff that makes the navigation responsive.

```
<nav class="navbar fixed-top navbar-expand-md navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Responsive Website</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
    data-target="#navbarCollapse" aria-controls="navbarCollapse"
    aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <!-- left navigation links -->
      <ul class="navbar-nav mr-auto">

        <!-- active navigation link -->
        <li class="nav-item active">
          <a class="nav-link" href="#">Home
            <span class="sr-only">(current)</span>
          </a>
        </li>

        <!-- regular navigation link -->
        <li class="nav-item">
          <a class="nav-link" href="#">About</a>
        </li>

        <!-- more navigation links -->

      </ul>

      <!-- right navigation link -->
      <ul class="nav navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="#">Login</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

The branding and menu items are self-explanatory. It should be clear that adding the class `navbar-brand` gives the title a clean look and is used for the website's name. The `nav` items are wrapped inside an additional `div` with the classes `collapse navbar-collapse`, which are used to make the menu appear like a stack when viewing in smaller browsers.

Just below the branding, you might be seeing an additional link with the `navbar-toggler` class that wraps a span `navbar-toggler-icon`. This link is visible only on the smaller screens with a list icon. Also see we've used `data-toggle=collapse` that Bootstrap uses to hide/show the menu items in smaller windows. `data-target` is used to identify which menus to hide/show.

The Marketing Area

The marketing area will be created using a `div` with the `jumbotron` class. Inside it, add `h1` with a class of `display-2`, a paragraph, and a link with the `btn btn-primary btn-lg` classes. The `display-2` class makes a common heading stand out. The same goes for the `lead` class on the `<p>` tag. The code should look like below:

```
<div class="jumbotron">
  <h1 class="display-2">Learn Geometry</h1>
  <p class="lead">Marketing message here.</p>
  <a class="btn btn-lg btn-primary" href="#" role="button">Call to Action »</a>
</div>
```

The Content Section

For the content section, we'll be using the new Flexbox-based Bootstrap grid system. We start by sectioning out the page into two columns, a wider and a smaller column. So the initial markup goes like this:

```
<div class="row">
  <!-- wider column -->
  <div class="col-md-8 mb-4">
    <!-- page content -->
  </div>
  <!-- narrower column -->
  <div class="col-md-4 mb-4">
    <!-- sidebar links -->
  </div>
</div>
```

The `col-md-*` class indicates that the two-column layout will only be triggered from medium-sized screens upwards. Smaller screens will get both columns stacked on top of each other. Also, notice the numbers at the end of the `col-md-*` class. Their sum amounts to 12, which is the total number of columns in the Bootstrap grid: 8 + 4 = 12. The `mb-4` class is one of the many utility classes Bootstrap makes available. This particular one is part of the spacing utility classes and is used to create some margin at the bottom of each column.

Bootstrap lets you easily nest columns by adding a further `div` element with the class of `row` inside the containing column. You can't have more than 12 columns overall, but you can certainly have fewer than 12 if you like. Our demo is going to have three equal-length, nested columns. Inside the wider column, add the following code:

```
<!-- nested columns -->
<div class="row">

  <!-- first nested column -->
  <div class="col-md">
    <h3 class="display-4">Title</h3>
    <p>
      Paragraph text.
    </p>
    <a href="#" class="btn btn-primary">Button</a>
  </div>

  <!-- second nested column -->
  <div class="col-md">
    <!-- column content -->
  </div>

  <!-- third nested column -->
  <div class="col-md">
    <!-- column content -->
  </div>
</div>
```

For more details, head over to Bootstrap's Grid docs, which are really user-friendly and packed with awesome information.

The Right Sidebar

The right sidebar will contain a vertical navigation list. Inside the smaller nested column, include the markup for the unordered list:

```
<ul class="nav nav-pills flex-column">
  <li class="nav-item">
    <a class="nav-link active" href="#">Active</a>
  </li>
  <li class="nav-item">
```

```
    <a class="nav-link" href="#">Link</a>
  </li>
  <!-- more links -->
</ul>
```

Bootstrap adds some interest to the simple unordered list with the `pills` class. Adding the `flex-column` class ensures the list displays vertically.

Building the Footer

Our footer is going to be a simple, one-column container with copyright information. But now that you know how the Bootstrap grid works, you can go ahead and make it as complex as you like.

So the code for the footer goes here:

```
<footer class="container mt-4">
  <div class="row">
    <div class="col">
      <p class="text-center">Design by <a href="#">Zetiz Labs</a></p>
    </div>
  </div>
</footer>
```

`text-center` is a handy utility class for text alignment, while `mt-4` is another spacing utility class that creates some margin top for the footer `div`.

Live Code

Have a look again at the final result.

Conclusion

We've come to the end of the article. Congratulations for building your first responsive website. Try resizing the windows or opening the demo page in various other devices, to see the actual responsive nature.

There's actually no end to what you can do with Bootstrap. You can customize it completely to make it look more personal, either by using your custom stylesheet document or by using its Sass variables and Sass maps.

Chapter 6

Bootstrap and WordPress Theme Integration in 8 Easy Steps

Ahmed Bouchefra

This tutorial explains the key steps for using Bootstrap and WordPress together, integrating the latest release of the Bootstrap library with a WordPress theme.

Both Bootstrap and WordPress are highly popular: 3.7 websites on the internet are built with Bootstrap and 29% of the web uses WordPress. Clearly, knowing how to build websites and apps using these two robust and mature, open-source technologies can be a valuable skill for developers in the workplace.

There's quite a lot to cover, so let's dive in!

Why WordPress?

WordPress is open-source software you can use to create a beautiful website, blog or app.

The popularity of WordPress is mostly due to its ease of use and powerful options for appearance, customization, and extensibility (via themes and plugins).

Thanks to WordPress themes, webmasters with little to no coding experience can power their websites with a professional appearance and custom functionality. Users can easily select different themes with a few clicks using the *Appearance* menu in the administration dashboard or copy theme files directly into the *wp-content > themes* folder. Themes can be purchased from dedicated marketplaces, individual developers, or can also be freely installed from the WordPress.org themes directory.

As developers, we can create our own theme, which is what we're going to do in this tutorial. More specifically, we're going to perform the key steps towards building a simple WordPress theme that integrates the latest release of the Bootstrap library.

Why Bootstrap?

Bootstrap is a robust and comprehensive UI library for developing responsive and mobile-first websites and apps. Here are some advantages of using Bootstrap as the styling framework for a WordPress theme.

Advantages of Using Bootstrap and WordPress Themes Together

In my view, there are many advantages of using Bootstrap to style a WordPress theme.

- Bootstrap is a popular, open-source project with extensive development and continuous maintenance, which over time has resulted in fewer bugs.
- It's a cross-browser framework that supports major browsers with a good CSS baseline called Reboot.
- It has an extensive and thorough documentation.
- It deals with reset, grids, typography, utilities, and media queries, thereby freeing up development time.
- It's widely used by developers to style websites, so it's easy to find tutorials, demos, and open-source projects to learn from or extend.
- Bootstrap can be used to quickly create a mobile-first and mobile-optimized WordPress theme without reinventing the wheel.
- There are tons of starter themes made available by the community, such as Understrap, which aim to provide a quick starting point for developers to create WordPress themes with Bootstrap.
- Although it's not created with WordPress in mind, Bootstrap can be easily

integrated with WordPress.

- We can easily customize Bootstrap to meet specific project requirements, once we have enough knowledge of the available classes.
- We can take advantage of hundreds of JavaScript/jQuery plugins already integrated with Bootstrap.
- Starting with Bootstrap 4, plugins use modern ES6.
- With the release of Bootstrap 4, the library now uses Sass instead of Less as the preprocessor of choice, which makes it more widely compatible with a huge number of developer workflows.
- Bootstrap 4 introduces new components such as the card component. Bootstrap cards make it easy to create a modern, card-based layout such as the Masonry-style interface.
- The Bootstrap 4 grid system is built on top of flexbox, which makes the grid even more flexible, developer-friendly and clean.

Are There Any Disadvantages of Using Bootstrap and WordPress Together?

As for the disadvantages, the developer community has raised a few concerns, including the following.

- Bootstrap isn't designed for straightforward integration with WordPress, but that shouldn't be a huge obstacle for most developers.
- If we need to override a lot of predefined Bootstrap styles to meet the design requirements, it might not be worthwhile to use a CSS framework at all.
- It's true that Bootstrap makes it easy to quickly add responsive styling to our

theme. However, we also need to invest time learning about Bootstrap to be able to add our customizations so that our themes look different from the numerous Bootstrap-based websites on the Internet.

- Bootstrap depends on jQuery, so in some situations we might have to deal with problems related to jQuery — such as outdated plugins, or having to include the whole jQuery library, even if our project only needs a small feature like `$.ajax()`.

Prerequisites for Following Along

In this tutorial, I assume you have a development environment with PHP, MySQL and WordPress installed — such as Homestead Improved. This quick tip will help you get up and running with a brand new Homestead Improved Vagrant VM in no time.

You also need to be familiar with WordPress — particularly how to install and activate themes, add WordPress Menus, create posts and pages, etc.

Finally, you need to have some knowledge of how to build a WordPress theme. In fact, this is a tutorial on integrating Bootstrap in a WordPress theme, not a tutorial on how to build a fully functional WordPress theme, which would have a much wider scope than what we have available in this article.

Key Steps to Integrate Bootstrap and WordPress

In this section, we're going to learn about the key steps we need to perform to integrate Bootstrap in a simple WordPress theme project.

First, let's review which files we're going to create.

The Structure of a WordPress Theme

A WordPress theme has a predetermined file structure. Some files are required for the theme to be recognized by WordPress.

The first required file is `style.css`. This CSS file contains styles for the theme. Most importantly, this file also has a special task: it provides meta information about the theme such as the theme name, description, author, and other extra details. The meta information needs to be present in the head of the file in the form of CSS comments.

The other required file is `index.php`, which is the main WordPress theme file, and the last fallback template file WordPress relies on, in case it can't find any other template file to display its content.

There are many optional files, but for our simple Bootstrap-based theme we're only going to add the following files:

- `header.php` and `footer.php`, to lay out the header and footer sections of our WordPress theme respectively, which are displayed on every page of our theme
- `functions.php`, where we're going to write the code for loading our theme's custom stylesheet, Bootstrap styles and scripts, and more.

If you're curious, check out the other templates that you can customize from the WordPress docs.

Let's get down to business!

Step 1: Creating the Theme Folder

First, we head over to our WordPress installation folder and navigate to `wp-content -> themes`. Here, we create a folder for our theme. Let's call it `bs-theme`.

Step 2: Adding style.css

Let's create our first required file, `style.css`, where we're going to put our custom CSS code.

At the very top of this stylesheet document (make sure you leave no blank space at the top), we add the meta information about our theme between CSS comments (remember to replace the `<THEME_URI>`, `<AUTHOR_NAME>`, and `<AUTHOR_URI>` placeholders with values relating to your own project):

```
/*
Theme Name: BS 4 Theme
Theme URI: <THEME_URI>
Description: A Theme for WordPress with Bootstrap for styling.
Author: <AUTHOR_NAME>
Author URI: <AUTHOR_URI>
Version: 1.0
*/
```

Now WordPress can display our theme's info in the admin area.

We can add our custom styles below the meta information comments. For example, let's add a few CSS rules to style the `<body>`:

```
@import url(https://fonts.googleapis.com/css?family=Montserrat:700);

body {
  padding-top: 4.5rem;
  font-family: 'Montserrat','Helvetica Neue',Arial,sans-serif;
  color: #001A33;
}
'Montserrat'
```

Step 3: Creating the Header Section

Let's start by creating `header.php` in the `themes` folder. Next, we add the following content:

```
<!DOCTYPE html>
<!--[if lt IE 7]><html class="no-js lt-ie9 lt-ie8 lt-ie7"> <![endif]-->
<!--[if IE 7]><html class="no-js lt-ie9 lt-ie8"> <![endif]-->
<!--[if IE 8]><html class="no-js lt-ie9"> <![endif]-->
<!--[if gt IE 8]><!--> <html class="no-js"> <!--<![endif]-->
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

    <title><?php wp_title('&laquo;', true, 'right'); ?> <?php bloginfo('name'); ?>
    </title>

    <meta name="description" content="">
    <meta name="author" content="">

    <meta name="viewport" content="width=device-width">
    <link rel="pingback" href="<?php bloginfo('pingback_url'); ?>" />

    <?php wp_head(); ?>
</head>
<body <?php body_class(); ?>>
  <nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
    <a class="navbar-brand" href="#">
      <?php bloginfo('name'); ?>
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
    data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false"
    aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="/">
            Home <span class="sr-only">(current)</span>
            </a>
          </li>
        </ul>
      </div>
    </nav>
```

In the code above, most meta information for the HTML `head` section is added using various WordPress tags, such as `bloginfo('name')` for getting the site's name, `wp_title()` to get the title for the page, and `wp_head()` to fire the wp_head action hook used by WordPress to add links and other functionality to the `head` section.

We've also used different Bootstrap classes to create a responsive navigation bar. However, as it is, the navigation bar is not dynamic. That is, it's not integrated with the WordPress menu functionality. This means that we can't build a WordPress menu in the admin area and see it displayed on the front end of our website.

Giving the static Bootstrap navigation bar the power of WordPress will be our next step.

Step 4: Integrating Bootstrap Navigation with the WordPress Menu

To accomplish our task, we need a WordPress `walker` class, which lets developers traverse tree-like data structures with a view to rendering HTML markup. There are a few Bootstrap navigation `walker` classes available on the Internet. For this tutorial, we'll go with Dominic Businaro's BS4navwalker, which is freely available on GitHub.

We grab `bs4navwalker.php` and save it in the root directory of our theme (which we called `bs-theme` earlier in this tutorial).

Next, we're going to create a `functions.php` file (this name is required by

WordPress) in the theme's root folder, and we'll write this line of code:

```
require_once('bs4navwalker.php');
```

Now we can use the Bootstrap navigation `walker` class in our theme files.

We locate the markup for the `nav` element we just wrote and modify it as follows:

```
<nav class="navbar navbar-expand-md navbar-dark fixed-top bg-dark">
  <a class="navbar-brand" href="#">
    <?php bloginfo('name'); ?>
  </a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
  data-target="#navbarCollapse" aria-controls="navbarCollapse" aria-expanded="false"
  aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <?php
    wp_nav_menu([
      'menu'            => 'primary',
      'theme_location'  => 'menu-1',
      'container'       => 'div',
      'container_id'    => 'navbarCollapse',
      'container_class' => 'collapse navbar-collapse',
      'menu_id'         => false,
      'menu_class'      => 'navbar-nav mr-auto',
      'depth'           => 0,
      'fallback_cb'     => 'bs4navwalker::fallback',
      'walker'          => new bs4navwalker()
    ]);
  ?>
</nav>
```

The code above *assumes we've already created a menu in the WordPress admin area* and have a few pages stored in the WordPress database.

To display our navigation bar, we're using the `wp_nav_menu()` function.

The values for the `menu` and `theme_location` parameters we're passing to `wp_nav_menu()` are taken from the settings of our WordPress menu, which we should have created already in the admin area.

The values for the `container`, `container_id`, and `container_class` parameters are taken from the Bootstrap classes and CSS `id` attribute on the `div` element that wraps the `ul` element containing the list items for our navigation links.

The `menu_class` parameter's value comes from the Bootstrap class on the `ul` element.

The `depth` parameter's value indicates how many hierarchical levels our navigation menu is going to have. We've set this to `0`, which is the default value and stands for *all*.

The `walker` parameter is here very important, and we've set it to a new instance of the `bs4navwalker` class, which is responsible for rendering the Bootstrap navigation markup.

And we're done!

You'll find a detailed explanation with the full list of parameters for the `wp_nav_menu()` function on the WordPress.org documentation page.

Step 5: Creating the Footer Section

The next step is to create the `footer.php` file and add the following content:

```
    <footer>
    </footer>
    < ?php wp_footer(); ?>
  </body>
</html>
```

`wp_footer()` is used by WordPress to dynamically place different markup elements, such as links to stylesheet and JavaScript files, at the bottom of the page.

It's important to note that many plugins use the `wp_head()` and `wp_footer()` hooks to place required elements in the head and footer of the page. Therefore, you'll need to make sure to add both hooks as shown in this tutorial. Doing so will avoid breaking these plugins' functionality when our theme is activated.

Step 6: Adding the index.php Template File

`index.php` is the second required file for every WordPress theme, so let's go ahead and create it. We then add the content below:

```
<?php get_header(); ?>
  <!-- Other Content here -->
<?php get_footer(); ?>
```

Using `get_header()` and `get_footer()` instructs WordPress to include the previously created templates, `header.php` and `footer.php`, inside `index.php`.

Step 7: Adding the WordPress Loop

To show our posts, we'll be using the famous WordPress loop.

Inside `index.php`, between the header and footer tags, let's add the following code:

```
<div>
  <?php if ( have_posts() ) : while ( have_posts() ) :   the_post(); ?>
    <h2>
      <a href="<?php the_permalink() ?>">
        <?php the_title(); ?>
      </a>
    </h2>
    <?php the_content(); ?>
  <?php endwhile; else: ?>
    <p>There no posts to show</p>
  <?php endif; ?>
</div>
```

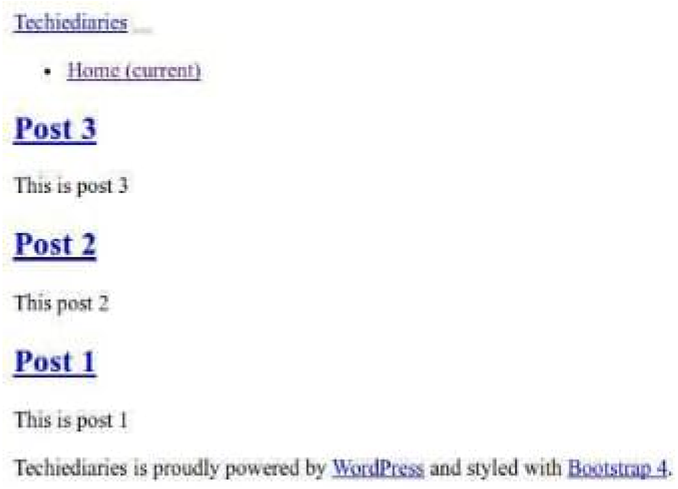
This is what happens above:

- We check if there are any posts by calling `have_posts()`.
- Using a `while` loop, we loop over all the existing posts.
- Finally, we show the title and content for each post. We could also get extra information such as the date when the post is published, the author of the post, comments associated with each post, and so on.

Step 8: Adding Bootstrap

After adding our template files, we now have a good starting theme that we can activate via the *Appearance* menu in the admin panel.

If we preview our theme, we'll get an unstyled site, which looks like the following screenshot:


Techiediaries

- Home (current)

Post 3

This is post 3

Post 2

This post 2

Post 1

This is post 1

Techiediaries is proudly powered by WordPress and styled with Bootstrap 4.

6-1. Preview of our unstyled WordPress theme

To give our theme a Bootstrap look and feel, we need to include the Bootstrap files in our project.

Let's grab the Bootstrap ZIP file from getbootstrap.com and copy the files into our theme. (For organizational purposes, we can add `css` and `js` folders to our project and place the corresponding files inside the appropriate folder.)

Our folder structure should look like this:

```
- bs-theme
  - style.css
  - footer.php
  - functions.php
  -  header.php
  - index.php
  - single.php
  - css
      - bootstrap.min.css
  - js
    - vendor
      - bootstrap.bundle.min.css
```

Next, we'll do the following:

- **Enqueue** Bootstrap stylesheet and JavaScript files
- use WordPress hooks to insert the Bootstrap files into the web page.

Let's start by opening `functions.php` and adding the following code:

```
<?php

function themebs_enqueue_styles() {

  wp_enqueue_style( 'bootstrap', get_template_directory_uri() . '/css/bootstrap.min
  ➥.css' );
  wp_enqueue_style( 'core', get_template_directory_uri() . '/style.css' );

}
add_action( 'wp_enqueue_scripts', 'themebs_enqueue_styles');

function themebs_enqueue_scripts() {
  wp_enqueue_script( 'bootstrap', get_template_directory_uri() . '/js/vendor/boots
  ➥trap.bundle.min.js', array( 'jquery' ) );
}
add_action( 'wp_enqueue_scripts', 'themebs_enqueue_scripts');
```

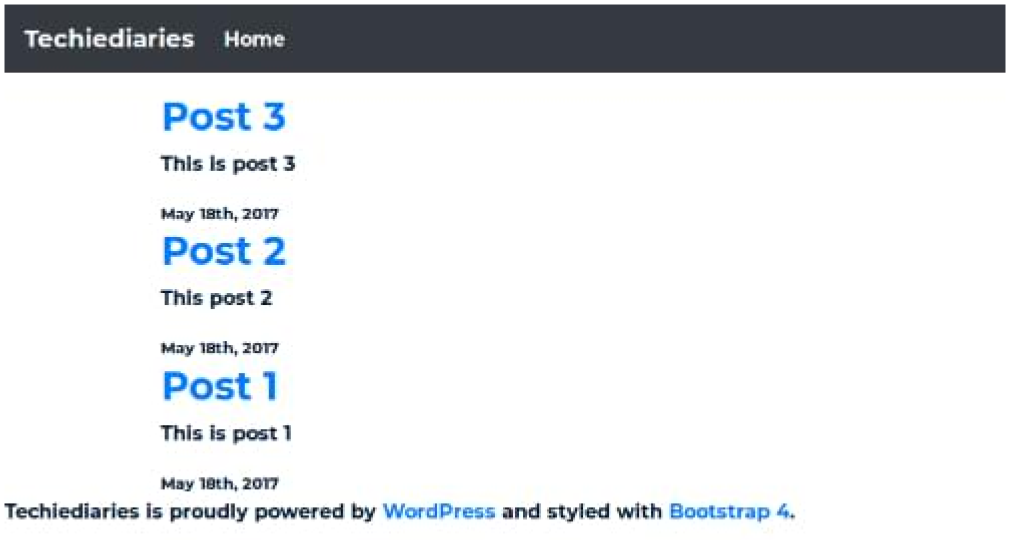
We've used various WordPress methods here, so let's briefly look at each one of them:

- `wp_enqueue_style()` : this method enqueues and loads the stylesheet passed as a parameter. In the code above, we use it to load both the Bootstrap stylesheet and our custom CSS file.
- `wp_enqueue_script()` : this method enqueues and loads a script file. We've used it to load Bootstrap's JavaScript bundle, indicating the jQuery library as a dependency. jQuery comes already bundled with WordPress, so we don't need to load it using a URL. (WordPress knows exactly where to get it.)
- `get_template_directory_uri()` : this method gets the URI of the current activated theme directory.
- `add_action('wp_enqueue_scripts', '...')` : this method hooks our functions into the *wp_enqueue_scripts* action (used when en-queuing styles and scripts

that need to appear on the front-end of the website).

You can find more about including CSS and JavaScript files in the WordPress theme docs.

This is our theme now with its shiny Bootstrap appearance:



Techiediaries Home

Post 3

This is post 3

May 18th, 2017

Post 2

This post 2

May 18th, 2017

Post 1

This is post 1

May 18th, 2017

Techiediaries is proudly powered by WordPress and styled with Bootstrap 4.

6-2. Our Bootstrap-based WordPress theme as it's rendered in the browser

Conclusion

In this tutorial, we've seen how to create a simple WordPress theme that integrates the latest version of Bootstrap.

With this new skill under your developer belt, you're now ready to create your own awesome, Bootstrap-based WordPress theme and share it with the world!

Chapter

7

Integrating Bootstrap with React: a Guide for Developers

Manjunath M

Integrating Bootstrap with React allows React developers to use Bootstrap's state-of-the-art grid system and its various other components.

In this tutorial, we're going to:

- explore tools and techniques for building a user interface with Bootstrap's look and feel for a React-powered web application
- use reactstrap to build a React contact list application.

React is one of the most popular JavaScript technologies for interactive web applications. Its popularity derives from its component-based modular approach, composability and fast re-rendering algorithm.

However, being a view library, React doesn't have any built-in mechanism that can help us create designs that are responsive, sleek and intuitive. That's where a front-end design framework like Bootstrap steps in.

Why You Can't Just Include Bootstrap Components with React

Combining Bootstrap with React isn't as easy as adding `<link rel="stylesheet" />` to an `index.html` file. This is because Bootstrap depends on jQuery for powering certain UI components. jQuery uses a direct DOM manipulation approach that's contradictory to React's declarative approach.

If we need Bootstrap just for the responsive 12 column grid, or the components that don't use jQuery, we can simply resort to the vanilla Bootstrap stylesheet. Otherwise, there are many libraries that take care of bridging the gap between React and Bootstrap. We'll compare both methods so that we'll be in a better position to choose the approach that works for our specific case.

Let's get started!

Setting up a Bootstrap Stylesheet with React

Let's use the Create React App CLI to create a React project, which requires zero configuration to get started.

We install Create React App and run the following command to start a new project and serve the development build:

```
$ create-react-app react-and-bootstrap
$ cd react-and-bootstrap
$ npm start
```

Here's the directory structure created by Create React App:

```
.
├── package.json
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
├── README.md
├── src
│   ├── App.css
│   ├── App.js
│   ├── App.test.js
│   ├── index.css
│   ├── index.js
│   ├── logo.svg
│   └── registerServiceWorker.js
└── yarn.lock
```

Next, let's download the latest version of the Bootstrap library from Bootstrap's official site. The package includes both the compiled and minified versions of the CSS and JavaScript. We'll just copy the CSS and place it inside the `public/` directory. For projects that just need the grid, there is a grid-specific stylesheet included too.

Next, we create a new directory for CSS inside `public/`, paste `bootstrap.min.css` in our newly created `css` directory, and link it from `public/index.html`:

```
<head>
<link rel="stylesheet" href="css/bootstrap.min.css">
</head>
```

Alternatively, we can use a CDN to fetch the minified CSS:

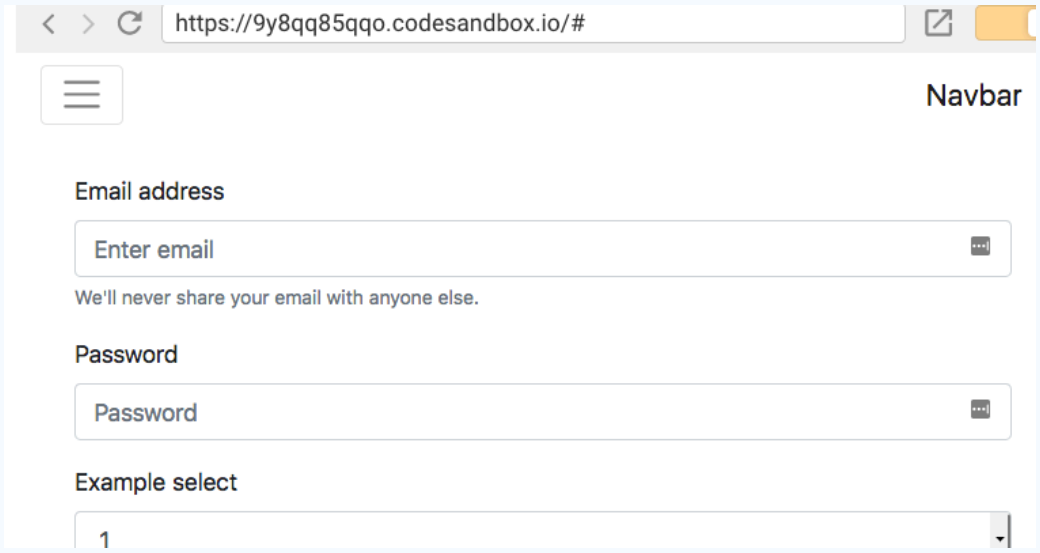
```
<link rel="stylesheet" href= "https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/
    ↪bootstrap.min.css">
```

Let’s have a look at what works and what doesn’t with this setup.

Using Vanilla Bootstrap with React

Once we add the Bootstrap stylesheet to a React project, we can use the regular Bootstrap classes inside JSX code. Let’s head over to the Bootstrap demo site and copy over some random example code, just to check everything is working as it should:

Live Code


https://9y8qq85qqo.codesandbox.io/#

Navbar

Email address

Enter email

We'll never share your email with anyone else.

Password

Password

Example select

1

7-1. Bootstrap demo site, with non-functioning NavBar

Check out the sample site here.

As we can see, the form looks good, but the NavBar doesn't. That's because the NavBar depends on the Collapse jQuery plugin, which we haven't imported. Apart from being unable to toggle navigation links, we can't use features like dropdown menus, closable alerts and modal windows, to name a few.

Wouldn't it be awesome if we could import Bootstrap as React components to make the most out of React? For instance, imagine if we could use a combination of Grid, Row and Column components to organize the page instead of the pesky HTML classes:

```
<!-- Bootstrap with HTML classes -->

<div class="container">
  <div class="row">
    <div class="col-sm">
```

```
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
</div>

<!-- Bootstrap with React Components -->

<Grid>
  <Row>
      <Col sm>
        One of three columns
    </Col>
    <Col sm>
        One of three columns
    </Col>
     <Col sm>
        One of three columns
    </Col>
  </Row>
</Grid>
```

Fortunately, we don't have to implement our own library to serve this purpose, as there are some popular solutions already available. Let's take a look at some of them.

An Overview of Third-party Libraries for React and Bootstrap

There are a few libraries that try to create a React-specific implementation of Bootstrap so that we can use JSX components while working with Bootstrap styles. I've compiled a list of a few popular Bootstrap modules that we can use with our React projects.

React-Bootstrap is by far one of the most popular libraries for adding Bootstrap components into React. However, at the time of writing this tutorial, it targets Bootstrap v3 and not the latest version of Bootstrap. It's in active development and the API might break when a newer version is released.

reactstrap is another library that gives us the ability to use Bootstrap components in React. Unlike React-Bootstrap, reactstrap is built for the latest version of Bootstrap. The module includes components for typography, icons, forms, buttons, tables, layout grids, and navigation. It's also in active development and it's a nice alternative for building React-powered Bootstrap apps.

There are a few other libraries like React-UI, and domain-specific modules like React-bootstrap-table and CoreUI-React Admin Panel on GitHub, which provide extended support for creating awesome UIs using React.

The rest of this tutorial will focus on reactstrap, because it's the only popular module that uses the latest version of Bootstrap.

Setting up the reactstrap Library

We start by installing the reactstrap library using npm:

```
npm install --save reactstrap@next
```

Now we can import the relevant components from the module as follows:

```
import { Container, Row, Col} from 'reactstrap';
```

However, the library won't work as we might expect it to yet. As explained on the reactstrap website, the reason for this is that reactstrap doesn't include Bootstrap CSS, so we need to add it ourselves:

```
npm install --save bootstrap
```

Next, we import Bootstrap CSS in the `src/index.js` file:

```
import 'bootstrap/dist/css/bootstrap.css';
```

Bootstrap Grid Basics

Bootstrap has a responsive, mobile first and fluid grid system that allows up to 12 columns per page. To use the grid, we'll need to import the Container, Row and Col components.

The **Container** accepts a `fluid` property that converts a fixed-width layout into a full-width layout. Essentially, it adds the corresponding `.container-fluid` Bootstrap class to the grid.

As for `<Col>`s, we can configure them to accept props such as `xs`, `md`, `sm` and `lg` that are equivalent to Bootstrap's `col-*` classes — for exmple, `<Col xs="6"> </Col>`

Alternatively, we can pass an object to the props with optional properties like `size`, `order` and `offset`. The `size` property describes the number of columns, whereas `order` lets us order the columns and accepts a value from 1 to 12. The `offset` property moves the columns to the right.

The code below demonstrates some of the Grid features in reactstrap:

Live Code

Grid features in Reactstrap.

Bootstrap Style Components in React

Now that we have reactstrap at our fingertips, there are tons of Bootstrap 4 components that we can use with React. Covering them all is beyond the scope of this tutorial, but let's try a few important components and use them in an actual project — say, a contact list app.

Navigation Bar

reactstrap Navbars are responsive navigation bar components. A Navbar can have subcomponents like NavbarBrand, Nav, NavItem, etc. for better organizing navigation links.

To make the NavBar responsive, we can include a `<NavbarToggler>` inside our `<Navbar>` component and then wrap `<NavItems>` into a `<Collapse>` component.

Take a look at the code below, which shows how we can use the Navbar component and React state to store the toggle data:

```
export default class Example extends React.Component {
  constructor(props) {
    super(props);

    this.toggle = this.toggle.bind(this);
    this.state = {
      isOpen: false
    };
  }
  toggle() {
    this.setState({
      isOpen: !this.state.isOpen
    });
  }
  render() {
    return (
      <div>
```

```
          <Navbar color="faded" light expand="md">
              {/* Brandname */}
                 <NavbarBrand href="/">
                  Demo
              </NavbarBrand>
                 {/* Add toggler to auto-collapse */}
            <NavbarToggler onClick={this.toggle} />
            <Collapse isOpen={this.state.isOpen} navbar>

                {/*Pull left */}
              <Nav className="ml-auto" navbar>
                  <NavItem>
                      <NavLink href="/link/">
                          Left Nav Link
                      </NavLink>
                  </NavItem>
              </Nav>

              {/* Pull right */}
              <Nav className="mr-auto" navbar>
                <UncontrolledDropdown nav inNavbar>
                  <DropdownToggle nav caret>
                    Bob
                  </DropdownToggle>

                  <DropdownMenu >
                    <DropdownItem>
                      Account
                    </DropdownItem>
                    <DropdownItem>
                      Settings
                    </DropdownItem>
                    <DropdownItem divider />
                    <DropdownItem>
                      Logout
                    </DropdownItem>
                  </DropdownMenu>
                </UncontrolledDropdown>
              </Nav>
            </Collapse>
          </Navbar>
```

```
        </div>
      );
    }
  }
```

Modal Window

The reactstrap Modal component creates a Bootstrap Modal with a header, a body, and a footer.

A modal component accepts a few props and callbacks to make the window interactive and closable.

The `isOpen` prop determines whether the modal should be visible. The `toggle` callback can be used to toggle the value of `isOpen` in the controlling component.

There are additional props for adding transition effects. The callbacks available include `onEnter`, `onExit`, `onOpened`, and `onClosed`:

```
{/*For the modal to open, this.state.show should become true which is usually
        ↪triggered by an onClick event */}
{/* toggleModal would set state of show to false onClose*/}

<Modal isOpen={this.state.show} toggle={this.toggleModal} >

    <ModalHeader toggle={this.toggle}>
        Modal title
    </ModalHeader>

    <ModalBody>
        Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
        ↪tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
        ↪veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
        ↪commodo consequat.
    </ModalBody>
    <ModalFooter>
        <Button color="primary" onClick={this.toggle}>Do Something</Button>{' '}
        ' '<Button color="secondary" onClick={this.toggle}>Cancel</Button>
     </ModalFooter>
</Modal>
```

Forms

A reactstrap `<Form>` can be either *inline* or *horizontal*.

An `Input` component renders an `<input>` element. We can wrap multiple `Input` components into a `FormGroup` for state validation, proper spacing, and to access other FormGroup features.

It's always a good idea to set a label using `<Label>`. There's a lot more to forms, and you should check out the Bootstrap documentation on Forms.

Here's the code for our reactstrap form:

```
<Form>
  <FormGroup row>
     <Label for="exampleEmail" sm={2}>Email</Label>
     <Col sm={10}>
         <Input type="email" name="email" id="exampleEmail"
         placeholder="with a placeholder" />
     </Col>
  </FormGroup>

  <FormGroup row>
     <Label for="examplePassword" sm={2}>Password</Label>
     <Col sm={10}>
        <Input type="password" name="password" id="examplePassword"
        placeholder="password placeholder" />
     </Col>
  </FormGroup>
```

```
    <FormGroup row>
       <Label for="exampleSelect" sm={2}>Select</Label>
       <Col sm={10}>
           <Input type="select" name="select" id="exampleSelect" />
       </Col>
    </FormGroup>

    <FormGroup row>
       <Label for="exampleSelectMulti" sm={2}>Select Multiple</Label>
       <Col sm={10}>
         <Input type="select" name="selectMulti" id="exampleSelectMulti" multiple />
       </Col>
     </FormGroup>

   <FormGroup row>
     <Label for="exampleText" sm={2}>Text Area</Label>
     <Col sm={10}>
       <Input type="textarea" name="text" id="exampleText" />
     </Col>
   </FormGroup>
 </Form>
```

ListGroup

The reactstrap ListGroup makes it easier to style and control list items in React. The `ListGroup` wraps `ListGroupItems`, which can be made interactive using the `onClick` callback.

Here's what the code looks like:

```
<ListGroup>
  <ListGroupItem>Item 1</ListGroupItem>
  <ListGroupItem>Item 2</ListGroupItem>
  <ListGroupItem>...</ListGroupItem>
</ListGroup>;
```

Buttons

Buttons are an integral part of any design framework. For buttons, there's a reactstrap `<Button>` component. Apart from the `active` and `disabled` properties, we can use `color` and `size` to set the button's style (primary, success, etc.) and size ('lg', 'sm', etc.):

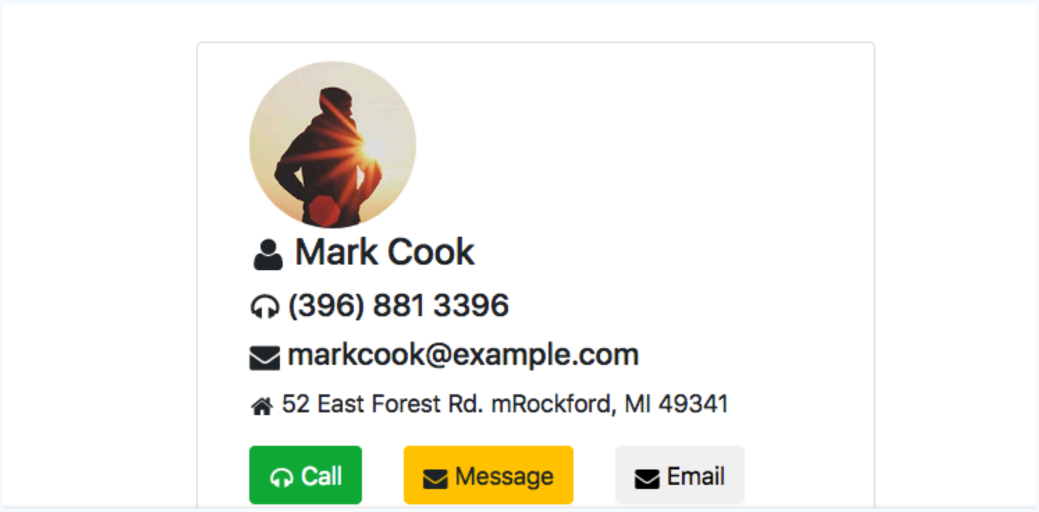
```
<div>
   <Button color="primary">primary</Button>{' '}

   <Button color="secondary">secondary</Button>{' '}

   <Button color="success">success</Button>{' '}

   <Button color="info">info</Button>{' '}

   <Button color="warning">warning</Button>{' '}

   <Button color="danger">danger</Button>{' '}

   <Button color="link">link</Button>
</div>
```

Demo: A Contact List Application UI with reactstrap

Here's a sample demo built using the components discussed in this tutorial.

Check it out and experiment with it to familiarize yourself with reactstrap:

Live Code



7-2. Reactstrap demo site

Check out the sample site here.

Summary

We've now covered everything you need to know to make the best of Bootstrap with React.

There's no shortage of libraries for integrating Bootstrap with React, and we've looked at some of the most popular options. We also explored a popular library called reactstrap. If you're ever in doubt, don't forget to check the documentation page to know more about the available components.

Chapter

8

Integrating Bootstrap with Vue.js using Bootstrap-Vue

Zeeshan Chawdhary

In this article, we'll explore how to integrate Bootstrap with Vue.js using Bootstrap-Vue.

React and Vue.js are two leading, modern JavaScript frameworks for front-end development. While React has a steep learning curve, a complex build process (if you're coming from a jQuery world), all you need to do to start using Vue.js is a simple import script:

```
<script src="https://unpkg.com/vue@2.5.13/dist/vue.js"></script>
```

Bootstrap has become a popular HTML/CSS framework for building mobile responsive websites. However, it relies mostly on jQuery for its core features as well as its extensive list of components — such as alerts, and modals. So we'll explore how to use Bootstrap with Vue.js, thereby removing the need for jQuery.

Introducing Bootstrap

Originally created in late 2011 by Mark Otto and Jacob Thornton at Twitter Inc., Bootstrap soon found popularity outside Twitter after being open-sourced. It continued to grow as the fastest front-end framework for web developers worldwide.

Today, Bootstrap has become the de-facto standard for starting a new website project, with its CSS and JS architecture providing mobile responsive and common UI components, along with support for most modern browsers.

Connecting Bootstrap with Vue.js

As we mentioned earlier, using with Bootstrap with Vue.js is slightly tricky, because of Bootstrap's dynamic components' heavy dependency on jQuery. However, there are at least a couple of good projects (very early builds, so don't expect everything to be tried and tested) that are in the process of bridging that gap:

- Bootstrap-Vue
- VueStrap

We'll explore the first option here, Bootstrap-Vue, since it's the most updated and popular project.

Bootstrap-Vue

Bootstrap-Vue not only supports the Bootstrap components and grid system, but also includes support for Vue.js Directives, which gives us the full feature-set from the Vue.js ecosystem.

One of the cool features of Bootstrap-Vue is that it has an online Playground. This playground is hot-reloaded and also lets us export our code to JSFiddle. Isn't that cool!

I believe a good documentation and developer ecosystem is necessary for the success of any software project and Bootstrap-Vue ticks all the checkboxes.

Getting Started with Bootstrap-Vue Using the Command Line

If you've been following modern web development trends, you'd definitely know about npm and installing libraries with it. Bootstrap-Vue can be installed with npm through the following command:

```
npm i bootstrap-vue
```

Bootstrap-Vue also provides two vue-cli templates that can scaffold our projects without too much hassle:

- Webpack Simple: quick scaffold for a small application.
- Webpack: for larger production capable projects.

First, we install the vue-cli by:

```
npm i -g vue-cli
```

Then, we initialize our project — let's call it *getting-started* — using the webpack-simple template by typing the following in the terminal:

```
$ vue init bootstrap-vue/webpack-simple getting-started

$ cd getting-started
$ npm install
$ npm run dev
```

Let's look at this code line by line:

- The first line starts with `vue init` creates a new directory called `getting-started`, where a new Bootstrap-Vue project is initialized.
- With `cd getting-started`, we access the new project's directory.
- `npm install` takes care of installing all the project's dependencies.
- Finally, with `npm run dev`, the app is compiled and launched in the browser.

Your local environment should now contain the following files and folders:

```
├── index.html
├── node_modules
├── README.md
├── package.json
├── src
│   ├── App.vue
│   ├── assets
│       ├── logo.png
│   ├── main.js
└── webpack.config.js
```

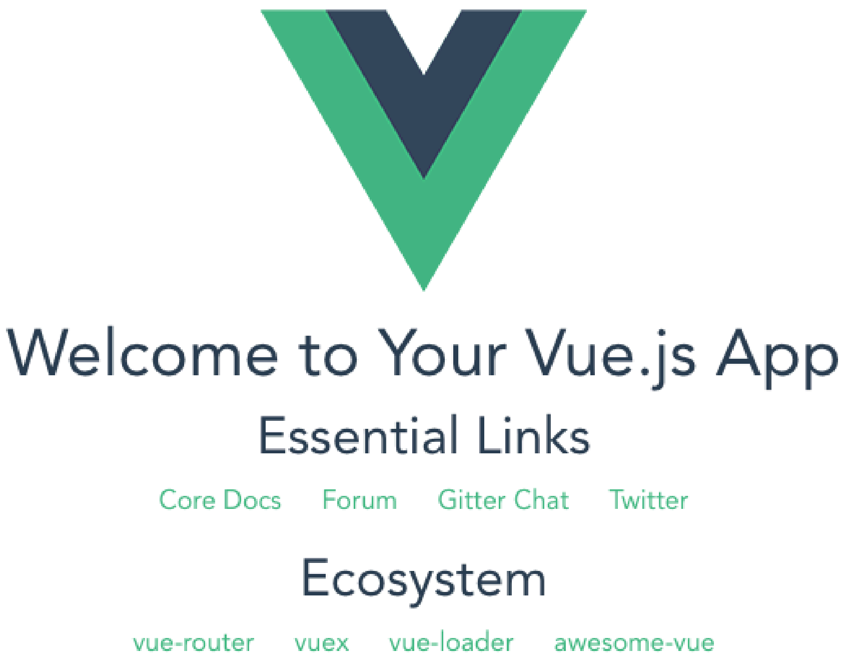
Here, `App.vue` and `main.js` are the primary files of interest. If we fire up our text editor and open `main.js`, we'll see the following code, which imports the

Bootstrap stylesheet and Bootstrap-Vue:

```
import Vue from 'vue'
import BootstrapVue from "bootstrap-vue"
import App from './App.vue'
import "bootstrap/dist/css/bootstrap.min.css"
import "bootstrap-vue/dist/bootstrap-vue.css"

Vue.use(BootstrapVue)

new Vue({
  el: '#app',
  render: h => h(App)
})
```

At the same time, the `App.vue` document loads up the front-end code.

After running the `npm run dev` command, the project's `index.html` page should render a page like the one below:



8-1. Vue.js splash page

Importing Bootstrap-Vue with a `script` Tag in the Document `<head>` Section

While we saw the npm way of installing and working with Bootstrap-Vue, let's also look at the simpler option: including Bootstrap-Vue using a `script` tag in the `<head>` section of our HTML document:

```
<!-- Add Bootstrap, Bootstrap-Vue CSS, Vue,
Babel, and Bootstrap-Vue JS to the <head> section -->
<link type="text/css" rel="stylesheet" href="//unpkg.com/bootstrap/dist/css/boot
↪strap.min.css"/>
<link type="text/css" rel="stylesheet" href="//unpkg.com/bootstrap-vue@latest/dist/
```

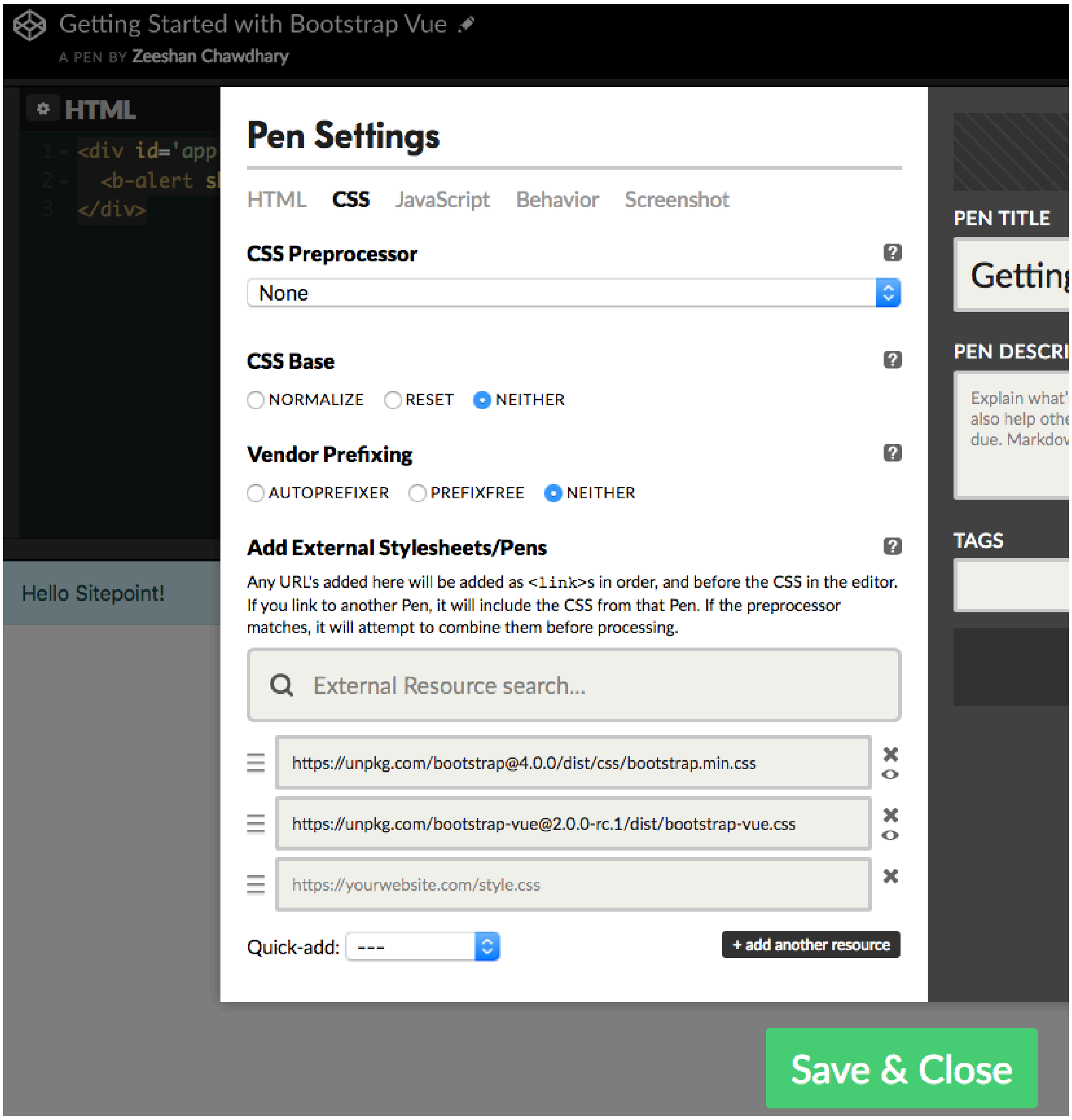
```
↪bootstrap-vue.css"/>

<script src="https://unpkg.com/vue@2.5.13/dist/vue.js"></script>
<!-- Add this after vue.js -->
<script src="//unpkg.com/babel-polyfill@latest/dist/polyfill.min.js"></script>
<script src="//unpkg.com/bootstrap-vue@latest/dist/bootstrap-vue.js"></script>
```

Now we can work with Vue.js, Bootstrap, and Vue-Bootstrap on our local machine.

Working With Bootstrap-Vue Components

For the demos in this article, we'll use CodePen. To set it up, let's create our Pen, click on the settings icon, and import the following CSS in the CSS tab:

```
https://unpkg.com/bootstrap@4.0.0/dist/css/bootstrap.min.css

https://unpkg.com/bootstrap-vue@2.0.0-rc.1/dist/bootstrap-vue.css
```


Getting Started with Bootstrap Vue

A PEN BY Zeeshan Chawdhary

HTML

```
<div id='app
  <b-alert s
</div>
```

Hello Sitepoint!

Pen Settings

HTML CSS JavaScript Behavior Screenshot

CSS Preprocessor

None

CSS Base

NORMALIZE RESET [✓ NEITHER]

Vendor Prefixing

AUTOPREFIXER PREFIXFREE [✓ NEITHER]

Add External Stylesheets/Pens

Any URL's added here will be added as `<link>`s in order, and before the CSS in the editor. If you link to another Pen, it will include the CSS from that Pen. If the preprocessor matches, it will attempt to combine them before processing.

External Resource search...

https://unpkg.com/bootstrap@4.0.0/dist/css/bootstrap.min.css

https://unpkg.com/bootstrap-vue@2.0.0-rc.1/dist/bootstrap-vue.css

https://yourwebsite.com/style.css

Quick-add: ---

+ add another resource

PEN TITLE

Gettin

PEN DESCRI

Explain what'
also help othe
due. Markdov

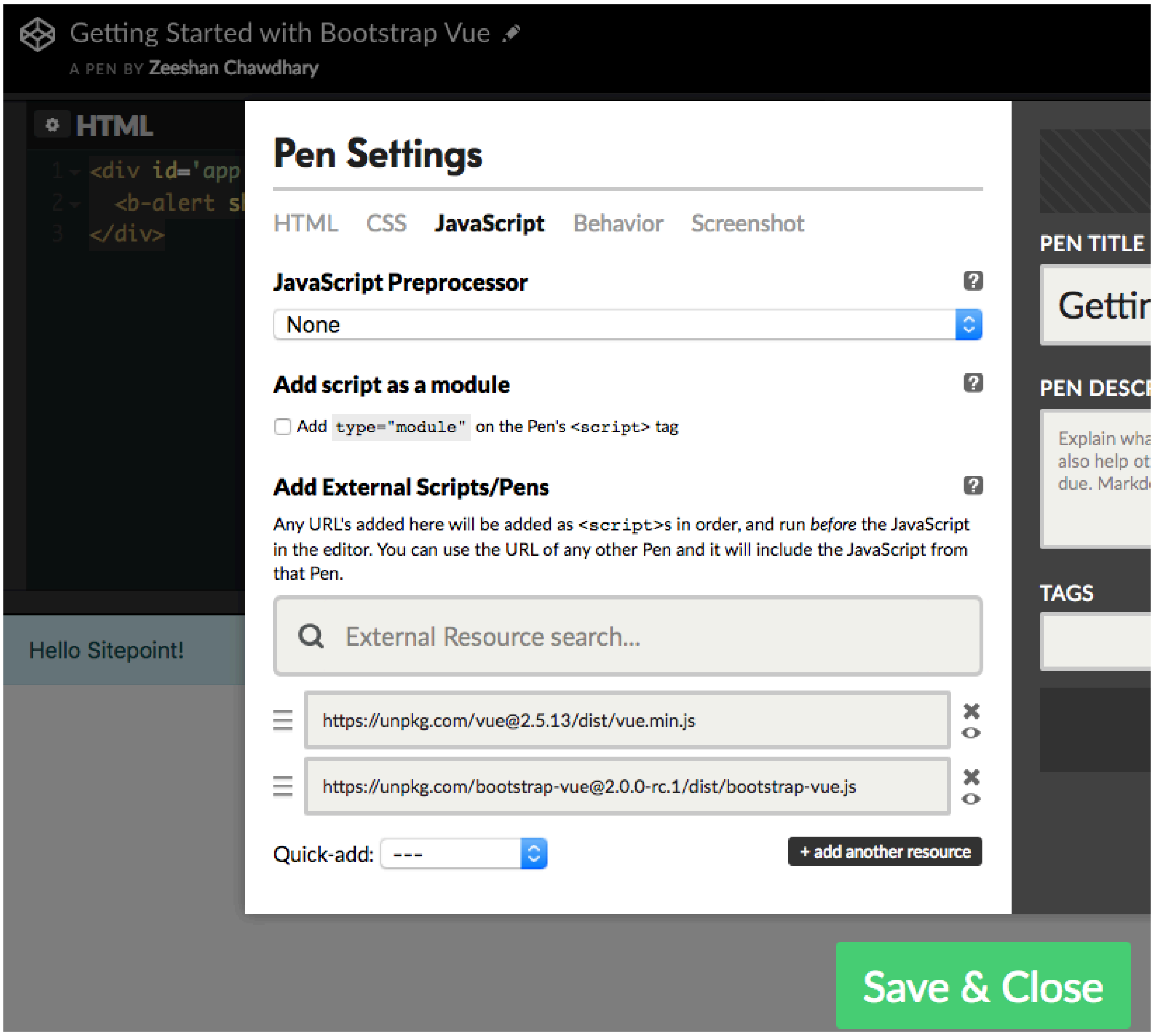
TAGS

Save & Close

8-2. CodePen CSS settings

In the Javascript tab, let’s import the Vue and Bootstrap Vue libraries:

```
https://unpkg.com/vue@2.5.13/dist/vue.min.js

https://unpkg.com/bootstrap-vue@2.0.0-rc.1/dist/bootstrap-vue.js
```


Getting Started with Bootstrap Vue
A PEN BY Zeeshan Chawdhary
HTML
Hello Sitepoint!
PEN TITLE
PEN DESC
TAGS
Save & Close

```
<div id='app
  <b-alert s
</div>
```

Pen Settings

HTML CSS **JavaScript** Behavior Screenshot

JavaScript Preprocessor: None

Add script as a module

Add `type="module"` on the Pen's `<script>` tag

Add External Scripts/Pens

Any URL's added here will be added as `<script>`s in order, and run *before* the JavaScript in the editor. You can use the URL of any other Pen and it will include the JavaScript from that Pen.

External Resource search...

https://unpkg.com/vue@2.5.13/dist/vue.min.js

https://unpkg.com/bootstrap-vue@2.0.0-rc.1/dist/bootstrap-vue.js

Quick-add: ---

+ add another resource

8-3. CodePen JavaScript settings

Finally, let's click the *Save & Close* button. Now we're ready to start playing with Bootstrap-Vue components.

Building a Bootstrap-Vue Alert Component

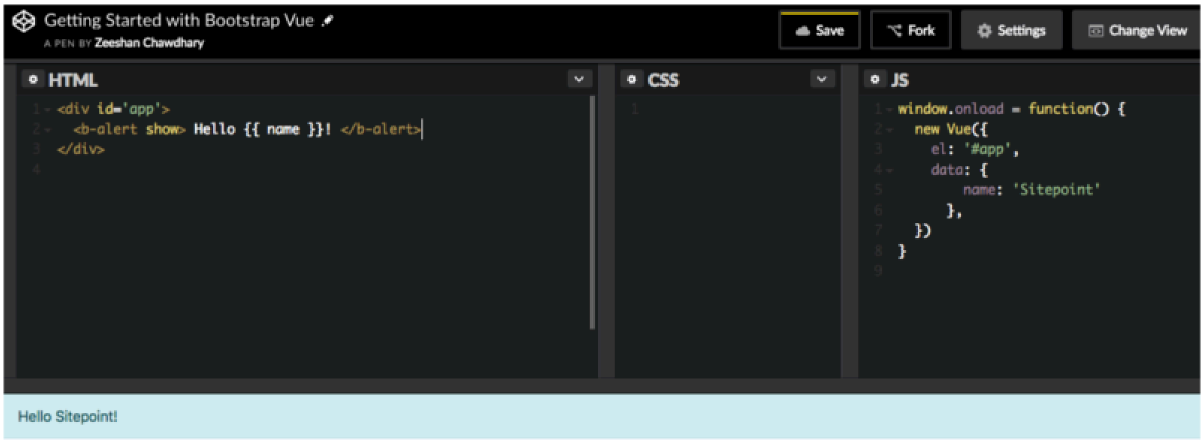
To create a Bootstrap-Vue alert component, we're going to add the following to our CodePen HTML area:

```
<div id='app'>
  <b-alert show> Hello {{ name }}! </b-alert>
</div>
```

Next, we add the Vue instance to the JS area:

```
window.onload = function() {
  new Vue({
    el: '#app',
    data: {
      name: 'Sitepoint'
    },
  })
}
```

As a result, we should see the “Hello Sitepoint!” alert in the output view area:


Getting Started with Bootstrap Vue
A PEN BY Zeeshan Chawdhary
Save
Fork
Settings
Change View
HTML
CSS
JS

```
<div id='app'>
  <b-alert show> Hello {{ name }}! </b-alert>
</div>
```

```
window.onload = function() {
  new Vue({
    el: '#app',
    data: {
        name: 'Sitepoint'
    },
  })
}
```

Hello Sitepoint!

8-4. Output of Bootstrap-Vue alert component

Live Code

See the Pen Getting Started with Bootstrap Vue.

The above code displays a simple Bootstrap alert component using Vue.js and Bootstrap-Vue. Next, we’re going to change some of the parameters for this alert box to make it more interesting. For example, to make the alert box dismissible, let's add the keyword `dismissible` next to the `show` directive:

```
<b-alert show dismissible> Hello {{ name }}! </b-alert>
```

Now the alert displays a dismiss icon button, which, when clicked, removes the alert from the page. Try it out for yourself!

Refer to the official documentation for Bootstrap-Vue alerts for more configurable props.

Building a Dynamic Bootstrap-Vue Listview Component

So now that we have got a good idea of how to use Bootstrap-Vue, let's build a list component. This is perhaps the most common piece of UI you'll find in web and mobile apps.

Bootstrap-Vue provides two components that together help us to build a list: `<b-list-group>` and `<b-list-group-item>`. We can think of the former as the HTML equivalent of a `<ul>` or `<ol>` tag, while the latter renders a `<li>` element.

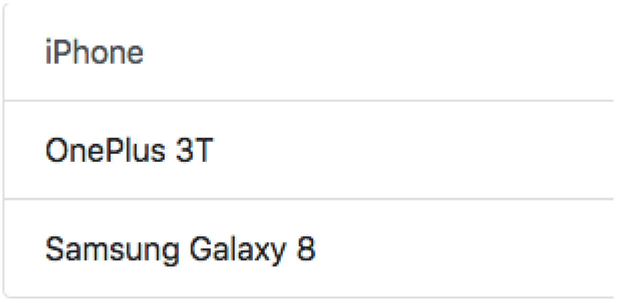
We start by building a static list of some smartphones:

```
<div id='app'>
  <b-list-group>
    <b-list-group-item href="http://apple.com">iPhone</b-list-group-item>
    <b-list-group-item>OnePlus 3T</b-list-group-item>
    <b-list-group-item>Samsung Galaxy 8</b-list-group-item>
  </b-list-group>
</div>
```

Now, we add our Vue instance in the JavaScript panel:

```
window.onload = function() {
  new Vue({
       el: '#app'
  })
}
```

And here's our simple list:


iPhone

OnePlus 3T

Samsung Galaxy 8

8-5. Static Bootstrap-Vue list

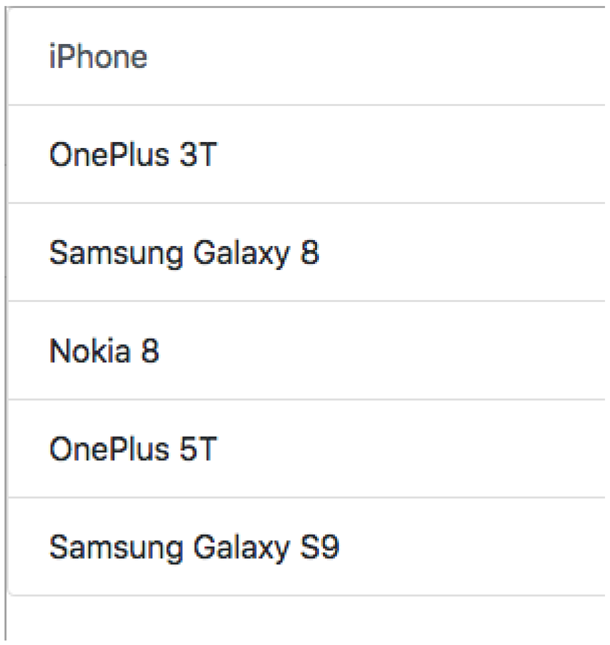
However, this is nowhere close to being a dynamic list. Let's add the Vue.js v-for directive inside the list component's markup to render a few list items from an array:

```
<b-list-group-item v-for="item in items">
  {{ item.message }}
</b-list-group-item>
```

Next, let's add the `items` array to the Vue instance:

```
new Vue({
  el: '#app',
  data: {
    items: [
      { message: 'Nokia 8' },
      { message: 'OnePlus 5T' },
      { message: 'Samsung Galaxy S9' }
    ]
  }
})
```

And here's our smartphone data displayed in the Bootstrap-Vue list component:



iPhone
OnePlus 3T
Samsung Galaxy 8
Nokia 8
OnePlus 5T
Samsung Galaxy S9

8-6. Dynamic Bootstrap-Vue list

Have a play with the live demo:

Live Code

See the Pen ListView with Bootstrap Vue.

As a challenge for you, try making this list even more dynamic by adding an Ajax call to pull in content from an API or from an RSS.

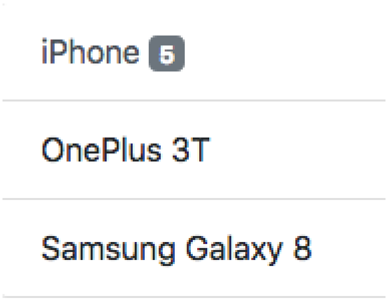
Listening to Events on Bootstrap badges

Bootstrap has a badge component which is useful for keeping count of items or labeling them. For example, in the list example above, we could add a badge to the iPhone list item indicating the number of variants (5 versions of iPhone).

With Bootstrap-Vue, we can use the `<b-badge>` component as a child of the `<b-list-group-item>` element to indicate the number of the various iPhone types as follows:

```
<b-list-group-item href="http://apple.com">iPhone <b-badge>5</b-badge>
</b-list-group-item>
```

This should render the list looking like this:



8-7. Bootstrap-Vue list with badge

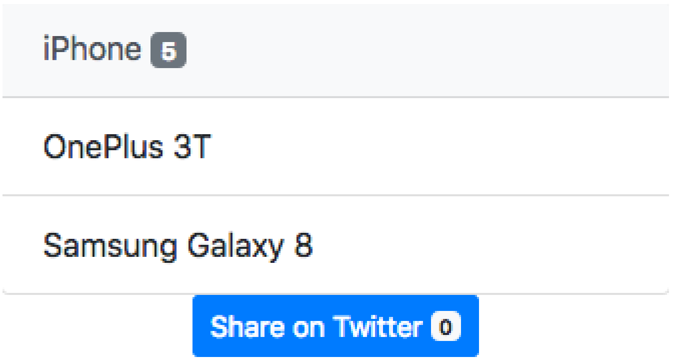
Now, let's add a *share this* button on our page to keep track of the page's number of shares. To accomplish this, we can use the `<b-button>` component to create a Bootstrap button and within the button we'll nest the `<b-badge>` child element:

```
<div class="text-center">
  <b-button variant="primary" size="sm">
    Share on Twitter <b-badge variant="light">{{ share_count }}</b-badge>
  </b-button>
</div>
```

We modify our JavaScript code by adding a `share_count` variable, which will keep track of the number of times our page is shared:

```
new Vue({
  el: '#app',
  data: {
    share_count:0
  }
})
```

This should give us the following output:



8-8. Bootstrap-Vue share button badge

Note, the button is still not dynamic — that is, it won't increment the share counter when we click on the *Share on Twitter* button, as we haven't added an event listener to the button yet. We'll use the Vue.js v-on directive to listen to the button click event:

```
<b-button variant="primary" size="sm" v-on:click="share_count += 1">
```

This makes the `share_count` variable increment whenever we click on the button, the code for the badge need not change, since it's already bound to the `share_count` variable. Therefore, whenever the button is clicked, the `share_count` variable is incremented and so is the badge.

That is the beauty of Vue.js data binding!

Live Code

See the Pen Listening to Events on Badges.

Conclusion

In this tutorial, we've seen how to use Bootstrap-Vue to add Bootstrap-like components to Vue.js applications.

Now, it's over to you: build the next awesome Bootstrap-Vue web page and share it with the world!