FlexSlider

jQuery Slider Plugin

Basic FlexSlider

In this project, you will use a jQuery plugin to create a simple, responsive web page with an image slider.

Markup

The markup is very basic. Just a header, then a div for the slider, then a main element with three columns, and a footer.

The page is set to be responsive with a mobile first approach.

```
<body>
    <div id="page">

        <header>
            <h1>Kittens for Sale!</h1>
        </header>

        <div>
            <ul>
                <li><img src="slides/kitten01.jpg" alt="kitten"></li>
            </ul>
        </div>

        <main>
            <article class="column">
                <h1>Kittens are Fun!</h1> ...
            </article>

            <article class="column">
                <h1>Kittens are Cute!</h1>...
            </article>

            <article class="column">
                <h1>Kittens are Soft!</h1>...
            </article>
        </main>

        <footer>
            <p>Sponsored by KITTENS ROCK &copy;2020</p>
        </footer>

    </div>
</body>
```

Media Queries

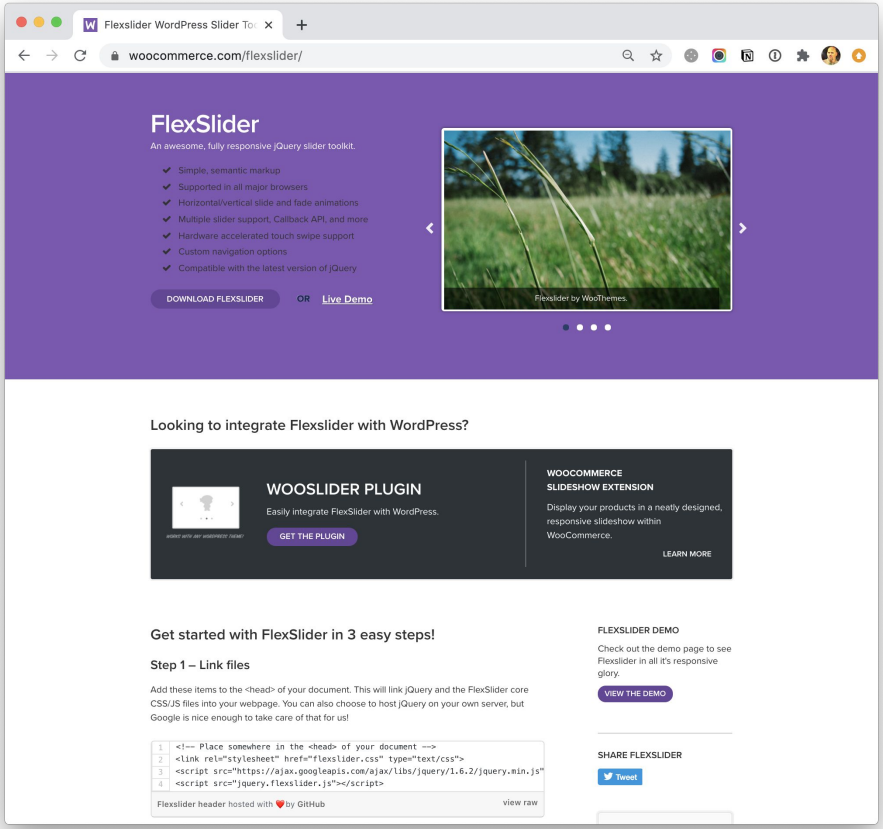
The media queries take care of the layout, if the screen width is wider than 600 pixels. This provides a very basic layout, so you can focus on building the flexslider.

```
/**************** Media Queries ****************/

@media only screen and (min-width: 600px) {

    main { display: flex; }

    .column { flex: 1; }

}

@media only screen and (min-width: 1000px) {

    #page {
        max-width: 1000px;
        margin: auto;
    }

    header h1 {font-size: 120px;}
}
```

FlexSlider Website

Go to the [FlexSlider website] and either click the Download FlexSlider button, or use the files I provided with this lesson. They are the same files.

This website provides basic instructions on how to plug in and use flexslider, so you will be referring to it.

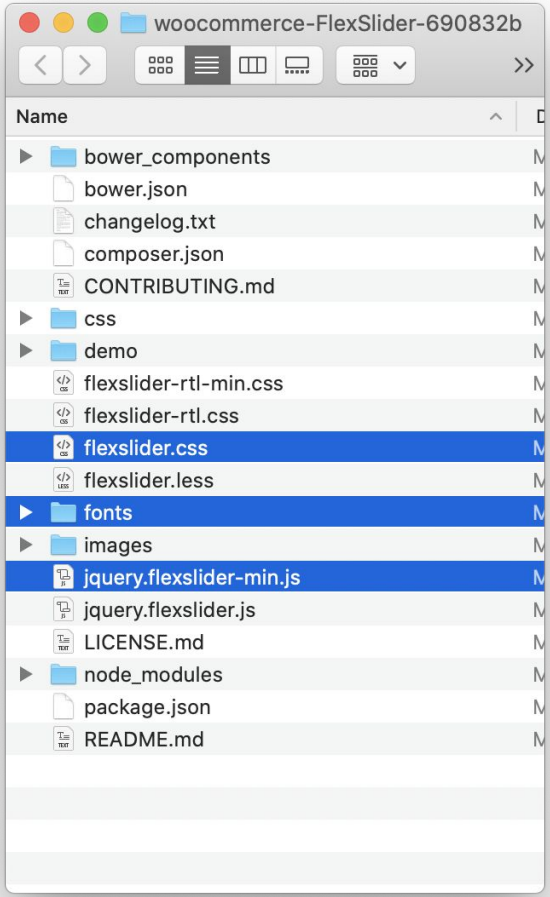


Files Needed

When you unzip the FlexSlider files, you get a whole bunch of stuff. The only things you need are the files highlighted in the picture on the left:

1. flexslider.css
2. fonts
3. jQuery.flexslider-min.js

The rest of the files are either for the demo, or for different install tools.

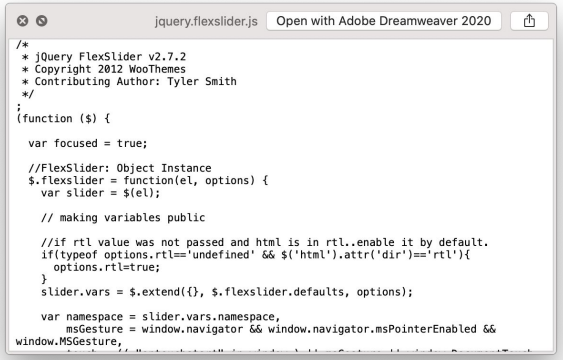


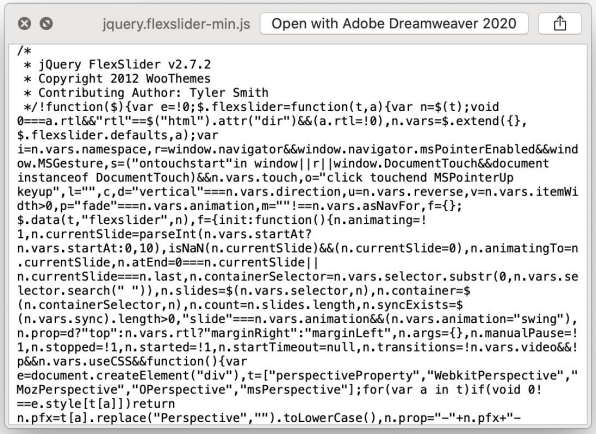
About the Minified Version

A quick note about the minified versus non-minified versions:

The minified version is the same file with all the whitespace removed and a few other things done to make the file smaller and faster to download. Use that version, if it is available for any plugin you want to use.

The non-minified version is available, if you want to see the actual JavaScript or maybe even edit it. You will probably never want to do that.





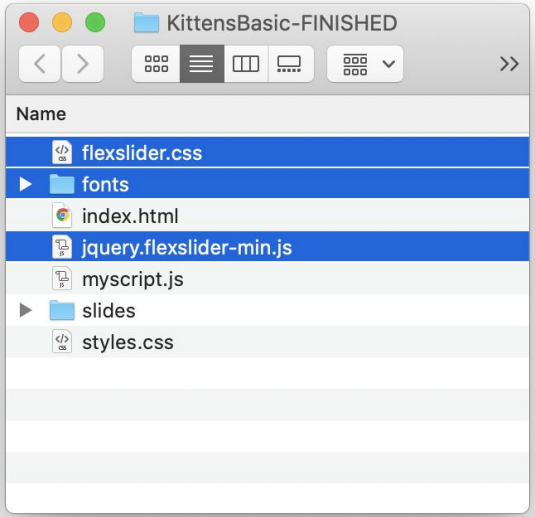
Add the Files to Your Project

Just drop these three files in the folder for your project.

The JavaScript file includes all the plugin functions you need to make it work.

The CSS file includes the styling for the slider to function.

The fonts folder, in this case, includes the next and previous arrows that show up on the left and right sides of the slider.

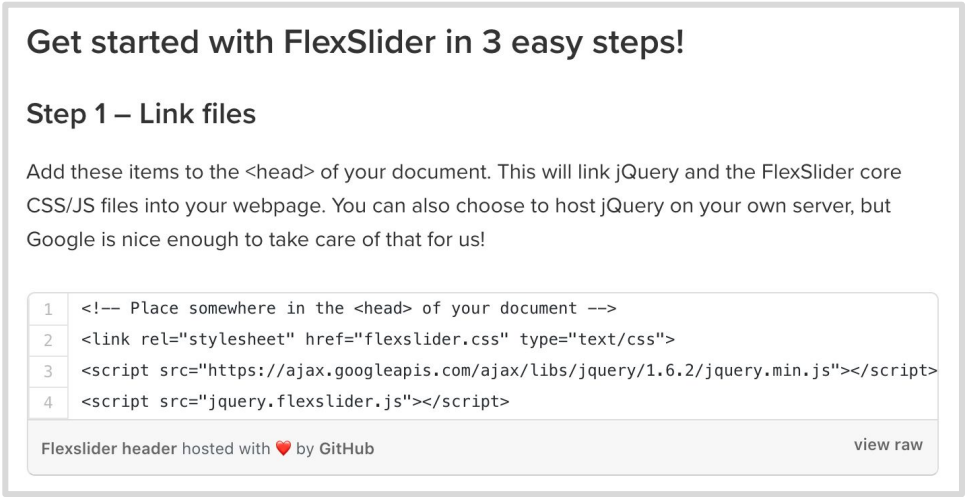


Following Directions

On the Flexslider website. Look at the directions on how to get it to work. Different plugins will give you different directions, however, there are usually some similarities.

Given what you know already about JavaScript and the way web pages are constructed, you can follow the directions exactly, or you can make your own choices about how to organize the code.

Let’s do it together so you can see what I mean.



Get started with FlexSlider in 3 easy steps!

Step 1 – Link files

Add these items to the <head> of your document. This will link jQuery and the FlexSlider core CSS/JS files into your webpage. You can also choose to host jQuery on your own server, but Google is nice enough to take care of that for us!

```
1  <!-- Place somewhere in the <head> of your document -->
2  <link rel="stylesheet" href="flexslider.css" type="text/css">
3  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.6.2/jquery.min.js"></script>
4  <script src="jquery.flexslider.js"></script>
```

Flexslider header hosted with ❤ by **GitHub** view raw

Your index.html File

```
<!-- Flexslider CSS -->


<!-- My CSS -->
<link href="styles.css" rel="stylesheet">

<!--Load jQuery...-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js" defer></script>

<!--Load jQuery Flexslider...-->

<!--My Script file-->
```

I have marked on the file where I want these scripts to go. Notice the directions on the flexslider website have an older version of jQuery.

I have a newer version already loaded here. Also notice my CSS file loads after the flexslider one. That is important because you want you styles to be able to override the flexslider styles.

Link to the Files

Link to the files in your folder like this...

I have chosen to put the JavaScript files in the head of the page, with the defer property set, but you could put them at the bottom of the page, before the closing body tag.

It is most important that the link to jQuery comes BEFORE the link to the flexslider functions. jQuery has to load before you can start using functions in the plugin file.

```
<!-- Flexslider CSS -->
<link href="flexslider.css" rel="stylesheet">

<!-- My CSS -->
<link href="styles.css" rel="stylesheet">

<!--Load jQuery...-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.

<!--Load jQuery Flexslider...-->
<script src="jquery.flexslider-min.js" defer></script>

<!--My Script file-->
<script src="myscript.js" defer></script>
```

Step Two Directions

Step two says to add markup. Notice that the script requires a div with a class of “flexslider” on it, and an unordered list with a class of “slides” on it.

Check what is in our file and make it match.

Step 2 – Add markup

The FlexSlider markup is simple and straightforward. First, start with a single containing element, in this example. Then, create a <ul class=”slides”>. It is important to use this class because the slider targets that class specifically. Put your images and anything else you desire into each <li> and you are ready to rock.

```
<!-- Place somewhere in the <body> of your page -->
<div class="flexslider">
  <ul class="slides">
    <li>
      <img src="slide1.jpg" />
    </li>
    <li>
      <img src="slide2.jpg" />
    </li>
    <li>
      <img src="slide3.jpg" />
    </li>
  </ul>
</div>
```

Flexslider Markup hosted with ❤ by GitHub view raw

Add Markup

```
<div class="flexslider">
    <ul class="slides">
        <li><img src="slides/kitten01.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten02.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten03.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten04.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten05.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten06.jpg" alt="kitten photo"></li>
        <li><img src="slides/kitten07.jpg" alt="kitten photo"></li>
    </ul>
</div>
```

Making the markup match is not difficult. There are seven slides in the slides folder. Just add them all and make sure the correct classes are on the div and unordered list.

Step Three Directions

Step three says to hook up the slider by adding this code in the head, after the links to the CSS file, jQuery and the plugin file.

I am going to put the window load function on my own linked script file instead. You already linked that file in the head of the page.

```
$(window).load(function() {
    $('.flexslider').flexslider();
});
```

Step 3 – Hook up the slider

Lastly, add the following lines of Javascript into the <head> of your document, below the links from Step 1. The $(window).load() is required to ensure the content of the page is loaded before the plugin initializes.

```
<!-- Place in the <head>, after the three links -->
<script type="text/javascript" charset="utf-8">
  $(window).load(function() {
    $('.flexslider').flexslider();
  });
</script>
```

Basic Flexslider Usage hosted with ❤ by GitHub view raw

Ready to Test!

The window load function you have seen once before. It just makes sure that all the assets in the browser window have loaded before the script runs.

```
$(window).load(function() {
    $('.flexslider').flexslider();
});
```

Quick check that we have all 5 things...

1. Markup? CHECK!
2. jQuery? CHECK!
3. Plugin? CHECK!
4. CSS File? CHECK!
5. Initialize the Script? CHECK!

Try it out!

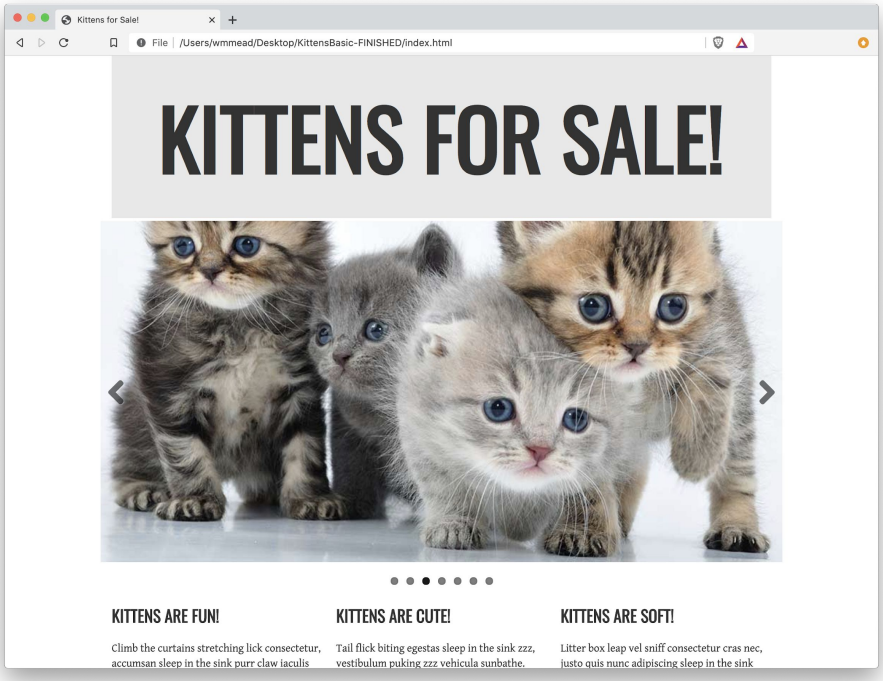
It Should Be Working

It should be working, but you can see there are a few styling issues to fix. It is sticking out on the sides, and there is a weird border on the top.

Also, I want to change the color of the dots at the bottom of the interface.

Some poking around in the inspector reveals what is going on and some simple rules in the CSS will fix it.

You will add these rules to YOUR stylesheet. Not the FlexSlider one.



FlexSlider Rules

Add these two rules, before the media query rules at the bottom of the stylesheet.

You could adjust rules in the flexslider.css file, but if you ever update that script with new files, you will lose those changes.

It is better to make sure your stylesheet loads last and overrides styles where necessary.

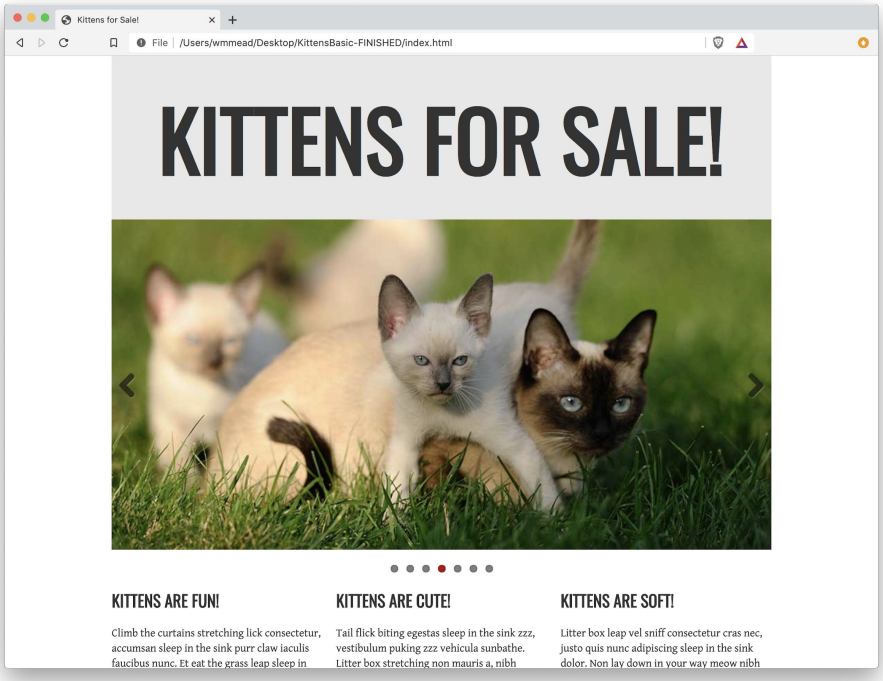
```
/*********** Flexslider Rules ***************/

.flexslider {
    margin: 0 20px;
    border:none;
    border-radius:0;
}

.flex-control-paging li a.flex-active {
    background:#B40205;
}
```

Looking Good!

This is looking good. It is functioning and everything, but what about that long fourth section of the flexslider website with all the options?

It turns out, you can use different options to affect the way the flexslider works.

These get passed in as an object. We haven't officially learned about object syntax yet, but we saw it a bit in the animate() function, so just go with it for now.



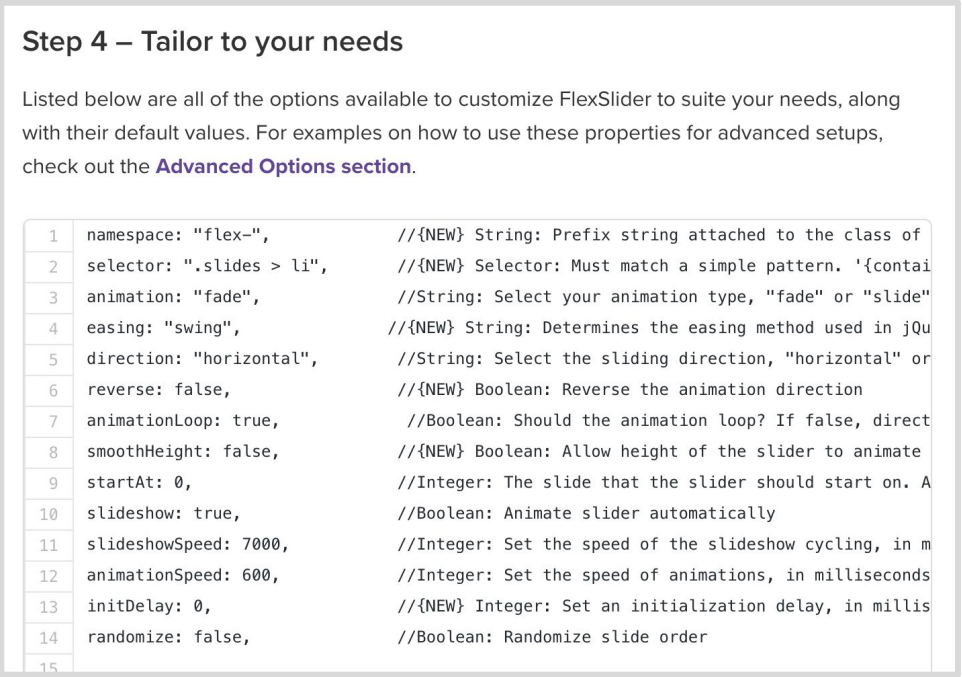
Additional Controls

The flexslider comes with all sorts of additional controls. The image on the right shows just a few of them.

If you poke around in there you can see what some of the different options are.

But how do you use them?

See the next slide for the answer!



Step 4 – Tailor to your needs

Listed below are all of the options available to customize FlexSlider to suite your needs, along with their default values. For examples on how to use these properties for advanced setups, check out the **Advanced Options section**.

```
namespace: "flex-",             //{NEW} String: Prefix string attached to the class of
selector: ".slides > li",       //{NEW} Selector: Must match a simple pattern. '{contai
animation: "fade",              //String: Select your animation type, "fade" or "slide"
easing: "swing",               //{NEW} String: Determines the easing method used in jQu
direction: "horizontal",        //String: Select the sliding direction, "horizontal" or
reverse: false,                 //{NEW} Boolean: Reverse the animation direction
animationLoop: true,             //Boolean: Should the animation loop? If false, direct
smoothHeight: false,            //{NEW} Boolean: Allow height of the slider to animate
startAt: 0,                     //Integer: The slide that the slider should start on. A
slideshow: true,                //Boolean: Animate slider automatically
slideshowSpeed: 7000,           //Integer: Set the speed of the slideshow cycling, in m
animationSpeed: 600,            //Integer: Set the speed of animations, in milliseconds
initDelay: 0,                   //{NEW} Integer: Set an initialization delay, in millis
randomize: false,               //Boolean: Randomize slide order
```

FlexSlider with Options

The options get passed in as an object, which starts and ends with curly braces.

Inside the object, you have key / value pairs separated by a colon, and a comma between each pair.

The key does not get quotes, but the value does, unless it's a number (like 2000) or a boolean (like true), because JavaScript knows what those things are.

Notice there is no comma after the last key / value pair.

```
$(window).load(function() {
    $('.flexslider').flexslider(

        {
            animation: "slide",
            slideshowSpeed: 2000,
            direction: "vertical",
            reverse: true,
            pauseOnHover: true
        }

    );
});
```

Advanced FlexSlider

Now that you have the basic one working, you can really pimp it out with some more sophisticated options.

Before you do that, save what you have and make a copy of the folder, and change the name of the folder to **KittensAdvanced**

Open the advanced one in your code editor and work from there!

Update the Markup

For this fancier version, the slides will have a background image, and some text on top, along with an animated call to action button.

Update the markup by using the markup on the right. You don't need to type it, I will add the markup in a snippet on the assignment page.

You can copy from there and replace the markup for the old flexslider.

Each list item will have a div, and that div will have a background image.

```
<div class="flexslider">
    <ul class="slides">
        <li>
            <div class="slide1">
                <h2>Kittens are good for you!</h2>
                <a href="#" class="cta"><span>These Kittens Available Now!</span></a>
            </div>
        </li>
        <li>
            <div class="slide2">
                <h2>Kittens will make you happy!</h2>
                <a href="#" class="cta"><span>Love This Kitten Today!</span></a>
            </div>
        </li>
        <li>
            <div class="slide3">
                <h2>Your life will never be boring if you have a kitten!</h2>
                <a href="#" class="cta"><span>This Kitten is Available Now!</span></a>
            </div>
        </li>
        <li>
            <div class="slide4">
                <h2>So Cute!</h2>
                <a href="#" class="cta light"><span>Awe, Look At That Face!</span></a>
            </div>
        </li>
        <li>
            <div class="slide5">
                <h2>Get a new kitten every week!</h2>
                <a href="#" class="cta"><span&ldquo;Please Take Us Home&rdquo;</span></a>
            </div>
        </li>
    </ul>
</div>
```

Additional Styling

Add these rules to the flexslider section of the stylesheet. You can copy them from the assignment page in Canvas as well.

The first set of rules sets the background image for each slide.

The next two rules set the layout and positioning of content within each slide.

```
.slide1 { background: url(slides/kitten01.jpg) top left no-repeat; }
.slide2 { background: url(slides/kitten02.jpg) top left no-repeat; }
.slide3 { background: url(slides/kitten06.jpg) top left no-repeat; }
.slide4 { background: url(slides/kitten04.jpg) top left no-repeat; }
.slide5 { background: url(slides/kitten05.jpg) top left no-repeat; }

.slides li div {
    width: 100%;
    position: relative;
    padding-bottom: 50%;
    background-size: cover;
}

.slides li div h2 {
    position: absolute;
    top: 20px;
    left: 20px;
    color: #B40205;
}
```

Styling the CTA

Next, add the rule for the call to action button that will animate down.

There is one additional rule for one slide that needs light text on it, instead of dark text on it.

```
.cta {
    display: flex;
    align-items: center;
    height: 25vw;
    width: 25vw;

    position: absolute;
    right: 20px;
    bottom: 100%;

    background: rgba(123,157,179,.3);
    border-radius: 50%;
    border: 5px solid #084D64;

    text-align: center;
    padding: 20px;
    text-decoration: none;
    font-family: 'Oswald', sans-serif;
    font-size: 4vw;
    color: #03425B;
}

.light { color: #EDEDED; }
```

One More Rule

Put this css rule inside the 600px wide media query. It will change the size of the call to action button for tablet sized browser windows.

```
.cta {
    height: 20vw;
    width: 20vw;
    font-size: 3vw;
}
```

Put the bottom one in the 1000 pixel wide media query. It will control the size of the call to action in the desktop sized browser windows.

```
.cta {
    height: 150px;
    width: 150px;
    font-size: 24px;
}
```

You can type these because they are short.

Replace the Options

Replace the object inside the flexslider with this one instead. You can copy and paste it from the snippets on the canvas assignment page.

Notice it is using the bouncing easing, so you will need to add the easing plugin as well. That is shown on the next slide.

```
{
    animation: "slide",
    slideshowSpeed: 5000,
    pauseOnHover: true,
    before: function(){ $(".cta").css("bottom", "100%"); },
    start: function(){ $(".cta").animate({ bottom: "5%" }, 3000, "easeOutElastic"); },
    after: function(){ $(".cta").animate({ bottom: "5%" }, 3000, "easeOutElastic"); }
}
```

Add the Easing Plugin

At the top of the index.html file, add the easing plugin. It MUST go after the link to jQuery, and BEFORE your script.

You can copy and paste the link from the snippets.

```
<!--Load jQuery...-->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.4/jquery.min.js" defer></script>

<!--Load jQuery Easing...-->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-easing/1.3/jquery.easing.min.js" defer>
</script>

<!--Load jQuery Flexslider...-->
<script src="jquery.flexslider-min.js" defer></script>
```

Should be Working!

A few more details about how these options work:

Before runs before the slideshow starts.
Start runs when it starts.
After runs after each slide animation.

You want the call to action to slide down, so place it out of view, above the slider, then animate it down and into place.

Start will make the animation work for the first slide.

After will make it work for the other slides.

Before is necessary to set the elements back to the top and into position so that when the slideshow wraps around, it can animate down again.

```
before: function(){ $(".cta").css("bottom", "100%"); },
start: function(){ $(".cta").animate({ bottom: "5%" }, 3000, "easeOutElastic"); },
after: function(){ $(".cta").animate({ bottom: "5%" }, 3000, "easeOutElastic"); }
```

Summary

This is an involved project, but I hope the end result is satisfying. Plugging in a pre-made script can be a little challenging, but not nearly as much work as writing all the code yourself.

Once you have worked with a few plugins like this, a whole world opens up where you can take advantage of scripts other people have written and use them in your own projects.