15
The Mono County Site

As I was writing this book, I decided it should be a text book; something you can take to the beach and read, without a computer anywhere near, and learn what web development is all about. That is why the examples were kept intentionally short and with only code snippets in print. The complete code for only a few chapters, which you can use to test during those hopefully rare moments when you are at a computer, are available online at Packt Publishing's website.

However, the book would not be complete without a full example of a website or application where we apply most, if not all, the things we have learned. This is what this chapter does, and as it only makes sense to try it out right away on a computer, it is only available online.

Our sample web application

You may have deduced, from the examples that I used and the names of the people and places that appear in them, that I am also a photographer and a big fan of the Eastern Sierra portion of California. I even went as far as to write a book about it.

At first, I was thinking of doing a photo gallery as our full example, but instead, I opted for what could be the website of the *Chamber of Commerce* of my favorite county in California—*Mono County*. Using a Chamber of Commerce website as an example has the benefit of being more than a photo gallery, so there will be more opportunities to use some of the elements we learned about in this book. As for choosing Mono County...

Mono County

Mono County, California, is located between the east entrance of Yosemite National Park and the border with Nevada. The main road that goes through it is the extremely beautiful US 395, which travels in a north-south direction. The town of Bishop is south of the county; Lake Tahoe and Carson City, NV are to the north.

It is a paradise for photographers, hikers, and flyfishers. It is part of the Eastern Sierra: the area on the east side of the Sierra Nevada mountain range. The county seat is the small town of Bridgeport, CA.

Its major attractions, besides the *High Sierra* portion of Yosemite National Park, are the very unique Mono Lake, Mammoth Lakes (for skiing in winter and mountain biking in summer), and Bodie, which is a mining town from the Gold Rush days, turned into a state park. Its true appeal is the great view you can have from US 395, and all the wonderful lakes you can hike to at elevations exceeding 10,000 ft. My personal favorite part of Mono County is the wonderful town of June Lake.

However, this is not a tourist guide; this is a book about web development; so without further delay, we need to start working on our website. I just wanted to give you an idea about what kind of things and towns the website will need to talk about.

Progressive enhancement – the basic site (monocountybasic.html)

Before we start our *Mobile first, responsive design* killer website, we need to create an old school static one, so people visiting our site using old browsers can actually see useful information rather than a blank screen. We explained this in an earlier chapter and now we will give you a real example.

For each of the main attractions of Mono County, we will create a page with some information and a photograph. One photograph is enough, as we want to encourage people to look at our site using a modern browser. On the home page, we will have a menu with links to these pages, and photographs that will also act as links to those same pages. We can safely assume that the only people who will be looking at this will be using a decent size computer screen, so we can specify that a width of. 950px wide is a good choice.

The east entrance of Yosemite National Park is at 9,943 feet and is called Tioga Pass. The entire area around it is beautiful for hiking, both inside and outside the park. Its best kept secret is *Saddlebag Lake*, which is a reservoir with 20 lakes on the other side of it, that you can reach by water taxi in the summer time. So, we will use *Tioga Pass* as one of our menu items. The others will be *June Lake*, *Mammoth*, *Bodie*, *Mono Lake*, and *Lee Vining*. The town of Lee Vining is at the junction of the road coming out of Yosemite and US 395, and is visited by thousands of tourists every year. Most of them may not remember the name of the town.

We will create a very basic CSS file, where we will specify a background color and some simple, yet cute, borders for our photographs. As far as fonts go, we need to make sure we include in the **font-family** list fonts that are very common in older computers running Windows, such as `Verdana` or `Arial`.

As we are going to eventually have JavaScript files, CSS files, and other kinds of files, it is best to create subfolders in our project directory to contain those. So, these could be our basic site components:

- `basic.css` (in a folder called `styles`)
- `monocountybasic.html`
- `tioga.html`
- `junelake.html`
- `monolake.html`
- `bodie.html`
- `mammoth.html`
- `leevining.html`

We also include an `index.html` file to glue all the examples together. With our progressive enhancement approach and the future use of **Foundation** in mind, we put our Mono County description in `<div>` with the class `row`, our six photos in `<div>` with the id `basicbody`, and add an empty `<div>` with the id `varicontent`. In our basic CSS file, we make sure the latter is not visible. Why? Once we start filling it up, we do not want people who are forced to see the basic version of our site to see any of the HTML we are going to put inside it.

Our basic menu is a `<ul>` styled in `basic.css`, but we wrap an HTML `<nav>` and `<section>` tag around it with the appropriate Foundation classes for an on-canvas top-menu.

Testing enhance.js (monocountyenhancetest.html)

Next, we will add the **enhance.js** testsuite to the code that we described in *Chapter 12*, *Mobile first, Responsive Design with Progressive Enhancement*. To make sure that everything works, we load a different CSS file, using a lighter background color, and load the jQuery library when the tests are successful. We use the copy of jQuery that comes with Foundation, so we put that one in `foundation5/js/vendor/jquery.js`. For enhance.js, we use a separate folder.

To check if jQuery works, we add a small JavaScript file that actually changes the color of the header text. Now we know enhance.js works.

Starting to build the home page with Foundation (monocountyfoundation.html)

Now that we have set up our basic site in the spirit of progressive enhancement, it is time to produce the skeleton of our Foundation based site. You should have downloaded the framework itself by now, as we have already used its copy of jQuery. We now add the relevant files to load in the `enhance()` function.

You will notice that, because of the inclusion the Foundation CSS file, our Mono County description is in a different font and size. And because we wrapped a `<div>` with the class `row` around it, our site is already partially responsive.

The same is the case with the menu. It already has the Foundation look and feel, but when we click on a menu item, we still land on a different, static page. However, this required a little bit of extra work. In the basic stylesheet, we gave our horizontal menu some styling, using the class `horizontalmenu`. We use JavaScript to remove that class in this intermediate version of our site.

This might trigger our first case of **Flash of unstyled content** (**FOUC**). The menu shows up as being way too big and long and then reappears as a smooth, Foundation style menu. We can avoid this by taking advantage of our knowledge that enhanced.js adds a class `enhanced` to the `html` element and enhance our `basic.css` file with the lines:

```
html.enhanced #mainmenu {
visibility:hidden;
}
```

Of course, in the JavaScript file where we remove the class `horizontalmenu`, we will have to show it again, like this:

```
$('#mainmenu').show();
```

The only other thing we need inside it at this point is to hide the `#basicbody <div>` as we are about to start putting content in the `#varicontent <div>`. We also include jQuery code to make the, albeit still empty, `#varicontent visible`.

Mobile first – determining what is important

Mobile devices with HTML5 capable browsers support cool features that are not available on computer screens. Simple and common features include touching, sliding, and so on; others that are more typical for these devices are GPS capabilities, audio, video, and automatically dialing a phone number if the device is a phone. This is intended to be a bonus chapter, not an entirely new book, so we will not cover adding support for these features.

What we want to be focused on here is *what* information the mobile visitor is interested in and *how* we are going to give them access to it. Monthly costs of using smartphones have come down and bandwidths have gone up but that does not mean that we need to make the service providers of our visitors rich (and our visitors poor) by forcing downloads of high resolution photographs and then having the browser shrink it to size.

If you use headers and footers on small screens as well, the eye-catching info on the homepage may not be visible on a GSM without scrolling down first.

So, I recommend, and Foundation makes this easy for us to do, not to show the following on a small screen:

- headers
- footers
- slideshows

On the other hand, we want to use the cool off-canvas capabilities to have a menu show up as if it were coming from right next (it is actually left) to where the phone is, with the main information and contact information first on the list.

Put down your thoughts on a piece of paper or on the smart drive that is inside your brain, as you will need it later, not now.

The slideshow (monocountyowl.html)

Many modern websites start with a moving slideshow of pictures or other graphical elements that are almost as wide as the screen, except on extremely wide ones. There, the use of `margin:auto` is in order. For the actual size of the images, we recommend 1280px by 480px. It is recommended not to use photographs with a standard aspect ratio of 4:3 or 6:4, because then visitors will have to always scroll down before they see some of the information part on your home page.

Foundation includes and supports **Orbit**, a JavaScript portion of the framework, to create sliders. This is what attracted me to Foundation to begin with. However, Orbit is now *deprecated* (by the way, I find this to be one of the strangest words in the English language). Depreciated may be a better choice.

At the time of writing this bonus chapter, the Foundation folks recommended two alternatives: **Slick** and **Owl**. I decided to go with Owl. You can find them at `kenwheeler.github.io` and `owlgraphic.com`, and make your own choice, depending on your taste.

So, what are we using the slider for? We are using it to draw attention by using photographs and adding some animation. With so much beauty present in Mono County, the only challenge is probably to not include too many. I decided to go with showing one photograph at a time.

The portion of our page that contains the slider will be a `<div>` that is inside our `#varicontent <div>`, as we will also want some kind of headline or blog information in the section between the header and the footer. I went for six photographs in total on Mono County subjects. I have also added, as a caption, a short version of the text that was used in the static files, so if people want to know what it is and nothing else, no further navigation is needed. To accommodate Owl, we will need some additional CSS and JavaScript, so we are adding `myowl.css` and `myowl.js`. The latter contains the options we want to use for Owl and the code to start up the slider. The CSS file we can use to style or caption.

All of this will eventually move into the real files we will use for the final version `monocountyhfooter.less`, `monocountyhfooter.css` and a few JavaScript files. What we are doing here, constantly changing filenames, is not the recommended way of doing your development; it is just a didactical approach so you can follow along with what the various steps are.

Adding less.js (monocountyless.html)

As we are about to add our own styling from now on, it would be smart to start using **less.js** from this point on. This will allow us to better organize our CSS, making it more manageable to change, and overall, having to write, you guessed it, *less* CSS.

So, we are modifying our code and are preparing ourselves for more (using *less*), by downloading the less.js JavaScript file and creating a `monocounty.less` file.

For this example, so you can continue to follow this step by step, we will create a `monocountyless.less` file in a folder called `less`. The only code we put in there is to create a variable for what will be our color for `h1` elements—to make it extremely obvious that it works, we picked red for this example and added the simple CSS to define the color of all our `h1` elements, the `less` way:

```
@h1color: red;
h1{
color: @h1color;
}
```

Note that when you use the `less` files, the `rel` attribute has to be `stylesheet/less`. Note also that enhance.js supports specifying the attributes in the following format:

```
 { href:'less/monocountyless.less', rel:'stylesheet/less',
type:'text/css' }
```

Using less should be for our development phase. Once we go to production, you should convert your final `less` file into a `css` file, to avoid the overhead for the people visiting your site.

Adding the blog section (monocountyblog.html)

Next, we probably want some kind of *blog* section underneath the slider. This is where the grid portion of Foundation will come in really handy. This content will typically come out of a database, but here we just add some blocks of text so we can test the responsiveness of our site. We want them to stack horizontally on wide screens and vertically on narrow screens, and Foundation will do that for us.

The blog section mimics events that go on in Mono County. A main event is always the opening of Tioga Pass, another one, the opening of the fishing season. This is always the key part of a site that *needs to be updated as often as possible*. The challenge there is usually that there needs to be an easy way for people who are *not* web developers to be able to update/change/add content. We will give you a few hints as to how to do that, at the end of the chapter.

We add a simple note for every one of our six Mono County topics.

The header and the footer (monocountyhfooter.html)

Headers and footers are the apparent easy portions of a page, and also the ones that usually never change. However, in responsive design, they offer challenges of their own; headers often contain logos and header text. As a consequence, you use images and text with large font sizes. This is the area where the smart use of media queries can become extremely important. We may also want to simplify what the header/footer contains once the viewport becomes smaller. As suggested before, we can also leave them out completely on the smallest ones.

Footers typically contain additional contact information and useful links to other sites, and usually have a copyright notice at the bottom. So, I added some links to sites of our various popular Mono County destinations, giving it an id `popular`. For the footer itself, I only included a Copyright line.

For the header, I just included a placeholder with a different background color. Both of them are `<div>` elements with the id `#header` and `#footer`. After we create them, we have to make sure we change our `basic.css`, so that nothing shows up on the basic view.

The on-canvas menu: refurbishing the menu (monocountynewmenu.html)

We promised a responsive design site and also a single application site where we always remain on the same page, so now is the moment to refurbish our standard menu, so that we actually replace a part of the page when a user clicks on a menu item, rather than send them to one of the many static html pages.

Of course, a real killer site will have additional menu items and submenus. Our exercise is to simply show you how to transform the existing basic menu into a modern one, and nothing else. We will accomplish this task by using JavaScript (jQuery).

We basically add an event handler that does what needs to be done when someone clicks on a menu item. The first thing we need is to make sure what happened until now no longer happens—going to another static html file. This is where the JavaScript function, `preventdefault()`, comes in handy.

For the event handler to work, we simply add a class, `spa`, to the `<a>` tags in the menu. I picked `spa` because what we are doing here is the first step to turning our site into a `Single Page Application`. In the JavaScript file that comes with it, we fetch the name of the original link and compose the name of the html file that needs to be loaded.

So, when a user clicks on a menu item, we use the jQuery/Ajax `load()` function to load in html from that file and replace the content of the `#varicontent <div>` by that code. A convenient way to do this is to use the names of the original static html files but to go fetch them in a different directory, in our case, `foundation5/content`.

In real life, things will be a bit more complex and we will typically `pass` a string to a `.php` file using `.post()`, and extract what we need from a database.

Adding the history fix (monocountyhist.html)

The Single Page Application approach has repercussions on what happens when a visitor presses the **BACK** button in the browser. We spent an entire chapter on this topic in the book, as it is that important. Now that we have changed partial content of our page, each time a user clicks on a menu item, we need to make sure that part works as well when they want to go back to what they think of as the previous page.

Fortunately, this is not difficult at all. Now may be the right time to go read the chapter on this topic again. We just created JavaScript code to stuff `#varicontent` with the appropriate content; we now simply add some more JavaScript to do the right thing on a `popstate` event and do a `pushstate` each time we load html that corresponds with a menu item. When a `popstate` event occurs, due to the visitor pushing the **BACK** key, we recompose the name of the HTML file, based on what we pushed earlier, and load that file to replace what is inside `#varicontent`, making people think they landed on what they believe is the first page.

As a mathematician, I admit that there is a fine line between being smart and being lazy. I did not include code to restore the initial home page once a visitor typed **BACK,** so much so that the history stack is empty. Some browsers, such as Safari, issue `popstate` on startup, and adding such code would only create more delay and more FOUC. Of course, when a visitor wants to see the **HOME** page, all they need to do is to hit the **HOME** menu item. *Simple comme bonjour*, excuse my French.

The off-canvas menu (monocountyoffcanvas.html)

We kept our Mobile first promise by jotting down on paper, first, what we wanted people to see when they visited our site on their small mobiles. Now it is time to implement it. The way Foundation is structured, makes it easier to add the code at a later stage, as it wraps around the code of the main menu.

The *off-canvas* menu feature allows us to put a real menu behind the three line icon you see on these devices when there is not enough screen real estate left to place a real menu, for example, when you use a phone or turn a small tablet from landscape to portrait.

Once you click on that menu, it will appear as if it comes from the left of your phone. To users of the Facebook app, this will look very familiar.

When people arrive at an airport and use their cell phone to look up the site of the hotel where they booked a room, they are not doing that to see what the rooms look like—they did that at home—but rather because they want to know the phone number or address. With smartphones, they may even want to use the GPS features of the device to get there.

It would not be unwise to have additional menu items that on larger screens would appear in the footer section. That way, for instance, a cell phone user is only two clicks away from the websites of some of the motels in Mono County.

So, the off-canvas menu is organized in a similar way to the main menu, but we added a few things that on larger screens appear in the *practical* section, so you can quickly, from your phone, visit exciting sites with useful information.

Putting it all together (monocounty.html)

There may be a thing or two we need to add or clean up before we have our final version, so this is the file that contains it. I have provided comments where needed.

One thing would be to convert your `less` file back to a CSS file. Check `lesscss.org` on how to do that. It is very easy; you use a less compiler, `lessc`, that you need to install using `npm`. Another thing is to remove the enhance.js `toggle`, as this can only confuse your visitors.

It is part of the enhance.js `API`. Simply set the `appendToggleLink` property to `false`.

The next steps

We supplied an example of a website, taking you through the steps you need to follow to go from a basic site that works everywhere and is potentially boring, to one that works on mobile phones and tablets and is totally responsive, by using what you learned from this book and the Foundation framework.

However, to dynamically create portions of our page when a user navigates, we still used static html files. This was intentional, so you could download the entire site (do not go off and print and frame my pictures), check out the code, and see how it works, without the need to choose a database, set up and propagate tables into a database, and so on.

But that would be a logical next step. Rather than loading html code from static files, you want to consider storing the content information on the server in a database. Take your pick; we presented you with two databases, MySQL and MongoDB, as well as good structured formats for data, such as XML and JSON.

The potential other next step is to create a program for the people that you made the site for; to allow them to—in an easy way—add and change content for that site. You are not going to do this by teaching them about databases and forcing them to read this book first. Instead, you could write a special program just for those people.

A program like that is often referred to as a **Content Management System** (**CMS**). Some popular commercial ones are Wordpress, Joomla!, and Drupal, but nothing prevents you from writing your own, which will be tailored to the needs of your customer, so that it only does what it needs to do.

Summary

We have walked you through the basic steps of building a *Mobile first, responsive design* website. We did not include a database, we did not use any PHP code; all we wanted to do was to give you an idea of what the workflow or stepping stones are, so that you do not leave users of old gear in the dark, and make sure your site looks cool when looked at with HTML5-capable browsers on modern devices.

The sky is the limit, moving forward. As we cannot fit the sky into this chapter, we will leave it up to you to be creative; look into and use the cool features Foundation (or Bootstrap) offers. There was neither time nor space to give you an example that demonstrates all of them.

I hope you, as a beginner or experienced developer, learned a thing or two from this book, and more importantly, enjoyed reading it. I enjoyed writing it. There were words of wisdom and words of wit, as I believe that reading up on a serious topic should be fun too.

Enjoy the World Wide Web and add content in the coolest possible way, but remember, it has to be mobile first, and responsive.