Table of Contents

Contents

Introduction

Hello and thank you for purchasing the Countdown theme. I hope you'll have a pleasant experience customizing and using this theme. If you find yourself stuck, feel free to contact me using themeforest or at themedwarf@gmail.com and I will do my best to help out.

This theme is built using the bootstrap framework, version 3.1.1. If you've worked with bootstrap before, customizing this theme will be a breze for you. If not, please check out their documentation at http://getbootstrap.com/ and familiarize yourself with the framework, it's very powerfull.

The theme also makes use of other cool resources like Jquery, Jquery countdown, FontAwesome, Google Fonts and Animate.css. You can find detailed documentation for each by doing a simple google search (or take a look in the chapter entitled "Resource used" where you will find links for each).

The theme has been using the W3C validators and is HTML5 and CSS3 compliant validated. Also the theme is responsive and mobile friendly thank's to bootstrap.

So please enjoy your new acquisition and good luck with your project.

Resources used

This will be a brief chapther that will contain the detailed list of resources used in this theme alog with links to documentation.

a. Jquery – www.jquery.com
b. Jquery countdown - https://github.com/hilios/jQuery.countdown
c. Bootstrap - http://getbootstrap.com/
d. FontAwesome - http://fortawesome.github.io/Font-Awesome/
e. OpenSans - http://www.google.com/fonts#UsePlace:use/Collection:Open+Sans
f. Animate.css - http://daneden.github.io/animate.css/

General theme structure

The theme is based on bootstrap, so the html structure is basically bootstrap structure. You have access to all bootstrap features like the grid system, UI kit and bootstrap's components.

The demo page is basically a fullwidth container that contains a single row. This row contains two columns: the timer column and the content column.

The timer column, as you might expect, is the leftmost column that holds the countdown timer. The content column is where all of the content is placed (navigation, pages, content, etc).

The theme contains some custom classes, some of which are bilt on top of allready existing bootsrap structure (like the customs forms). It also contains come custom javascript to help with the “magic” (like the page navigation and the timeline component).

I will go into further detail in the following chapter.

Theme components

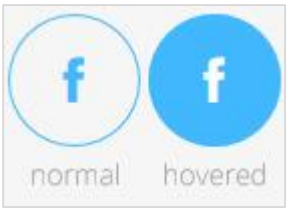


Figure 1 Social Button in both normal and hovered state

The theme come with some custom components. In this chapter I will be detailing each one with pictures and customization tips. But first let’s round them all up:

- Social Buttons
- Countdown Timer
- Custom Forms
- Page Navigation
- Timeline
- Animation (not custom, but will describe how it is used)
- Color Schemes

Social Buttons

The social button has two states: normal and hovered. The transition from one state to the other is done taking advantage of CSS3 transitions. This gives it the smooth and pleasant transition effect.

Html Structure

```
<button class="btn btn-social btn-facebook">
        <i class="fa fa-facebook"></i>
</button>
```

As you can see, is is basically an html button with some custom classes that contains an <i> tag.

The button has some custom classes which are required in order for the styling to work. The “btn” class is from bootstrap and offers some basing styling.

The “btn-social” class belongs to the theme. It holds styling for the social button, like size, shape, border, etc. The final class “btn-facebook” is the class that holds the color information. This is the class you will be replacing when creating your own custom social buttons.

The <i> tag is used by font awesome to insert an icon in that location. The icon displayed is determined by the class given to the <i> tag. For a complete list of supported icons, please review the font awesome cheatsheet at: http://fortawesome.github.io/Font-Awesome/cheatsheet/ .

Customization

Customizing the social button is pretty easy actually. Find the class named .btn-social in the index.css file.

```
.btn-social{
        transition-duration: 0.2s;
        background-color: transparent;
        border: 1px solid #41b8fe;
        color: #41b8fe;
        font-size: 24px;
        width: 66px;
        height: 66px;
        border-radius: 100%;
}
```

The css code is pretty much self explanatory. To modify the border width or color, for example, you need to modify the property named border.

If you need to modify the size of the button, modify the width and height properties, but give them the same value in order to keep the button round. When changing the button size, also consider changing the font size accordingly.

To modify the hover state look for this code in the css:

```
.btn-social:hover{
        transition-duration: 0.2s;
        color: white;
}
```

There isn't much to customize here except the icon color on hover and the duration for the transition effect (from normal to hover).

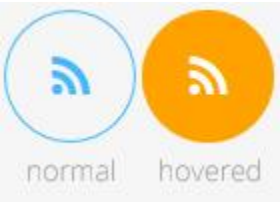
Creating your own custom button

To create you own button you will need to create a css class like the btn-facebook class you see in the html structure example above. This class will basically dictate the color of the icon and the background to which the button will transition when hovered.

So it will look like this:

```
.btn-facebook:hover{
        background-color: #41b8fe;
        border-color: #41b8fe;
}
```

As you can see there is no need to specify a "normal" state because it's allready defined in the btn-social class.



So let's say I want to create an rss feed button that turns orange on hover, like the one on the left.

First create the html structure like so:

```
<button class="btn btn-social btn-rss">
        <i class="fa fa-rss"></i>
</button>
```

Notice that I also added my own custom class "btn-rss" which I will be defining next. Also notice that I have change the icon inside the button by changing the class of the <i> tag to "fa-rss" (from the font awesome cheatsheet).

The only thing remaining is defining the colors for the hover state of my new button. So inside the index.css I create a new class like so.

```
.btn-rss:hover{
        background-color: #ffa200;
        border-color: #ffa200;
}
```

The color is #ffa200. You can choose whatever color you like. The color is defined in hex format. There are color pickers online like http://www.colorpicker.com/ where you can get the hex code for any color you want.

This concludes the social buttons documentation. Happy customizing.

Countdown Timer

Hilios's countdowntimer is an esy to use and customize jquery plugin. In this thme it is aused as follows. First I created an html container with a class of "countdown-timer":

```
<div class="col-sm-3 sidebar">
        <ul class="text-center countdown-timer">
        </ul>
</div>
```

In the index.js file we initialize the plugin on the container along with all of the options:

```
// initialize countdown script
$(".countdown-timer").countdown('2015/04/01', function(event){
        $(this).html(event.strftime(
                '<li>%D</li>'
                + '<li>days</li>'
                + '<li>%H</li>'
```

```
                + '<li>hours</li>'
                + '<li>%M</li>'
                + '<li>min</li>'
                + '<li>%S</li>'
                + '<li>sec</li>'
        ));
});
```

This is the basic structure and usage of the helios's jquery countdown timer.

Customizing

Customizing is easy, when you know what everithing is and does. The html structure is self explanatory. It's just an html container to which you give a class name or an id.

Inside the javascript file (index.js) you need to initialize the timer as shown above. You can change to parameters to make it fit your needs. The first one is the first parameter of the countdown function, namely "2015/04/01". As you have guesed allready this parameter represents the end date of the timer.

The second parameter you might consider editing is the parameter given to the strftime function. This parameter is a string that dictates the html structure of the timer. This structure will be placed inside the container you specified.

As you can see there are placeholders placed inside the html (%D, %H, %M, %S). These are used by the plugin to represent the time in a specific way (%D is for days, %H is for two digit hours, etc).

For a complete list of placeholders (or directives as they are called by the author) take a look in the table below.

Directive	Blank-padded	Description
%Y	%-Y	Years left *
%m	%-m	Monts left *
%w	%-w	Weeks left
%d	%-d	Days left
%D	%-D	Total amount of days left
%H	%-H	Hours left
%M	%-M	Minutes left
%S	%-S	Seconds left

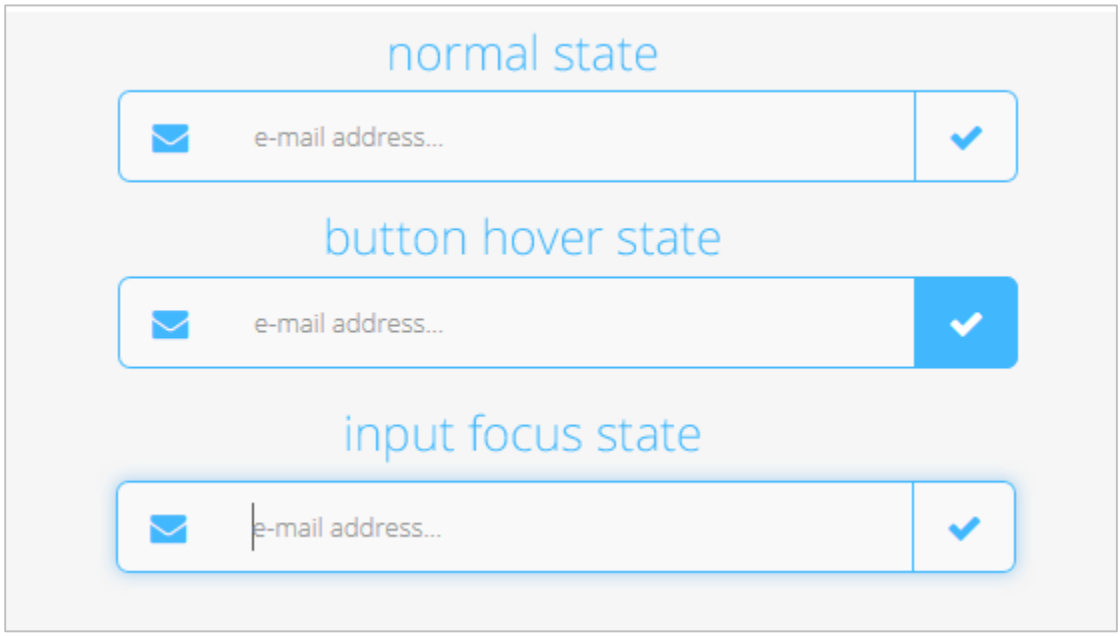
For a complete documentation of Helios's jquery countdown plugin please visit: http://hilios.github.io/jQuery.countdown/documentation.html .

Custom Forms

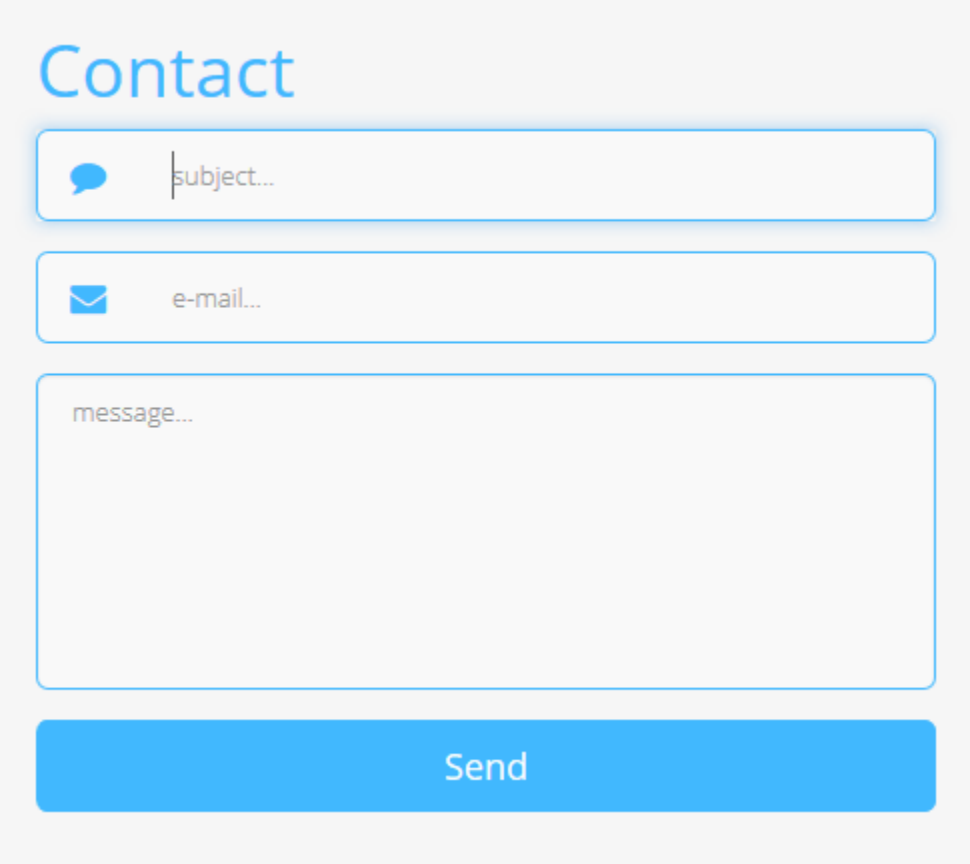
The theme uses custom styles for forms. These styles are represented in the class “newsletter-form”. It uses two aditional helper classes “newsletter-focus” and “newsletter-focus-btn” to obtain the box-sadow effect on the entire input tag. There’s also a small block of jquery code used to trigger the focus effect on the entire input group when the input tag is focused.

The html structure used is the basic bootstrap input group. In the demo there are two kinds of input groups used.

There’s the input with an icon prepended and a button apended:



And there’s the input without the apended button:



The only difference between the two is the use (of lack of) a prepended button. The html structure for the input groups is bootstrap structure, like so:

```
<div class="input-group input-group-lg newsletter-form">
        <span class="input-group-addon"><i class="fa fa-envelope"></i></span>
        <input type="email" class="form-control" placeholder="e-mail address..." required>
        <span class="input-group-btn">
                <button class="btn btn-default" type="submit"><i class="fa fa-check"></i></button>
        </span>
</div>
```

We have the container div with a class of “input-group”. Inside the container there are 3 elements. First a span container with the prepended icon. Then the actual input follows (the input tag). Lastly there’s the span container for the appended button.

If you want an input with both a prepended icon and an appended button, use the html structure above. Else if you want an input with only a prepended icon use the same structure, minus the span containing the button. Either way, make sure you you add the class of “newsletter-form” the the input group.

As mentioned above, the styling for the custom forms are handled by the following css classes: “newsletter-form”, “newsletter-focus” and “newsletter-focus-btn”. The second and third classes are helper classes used in conjuction with javascript to handle the focus effect.

Any customization you might want can be achieved by modifying the classes present in the index.css file. Just look for the "newsletter-form" class (there will be multiple instances of the class, modify what you need).

The javascript code that handles the box shadow on focus is the following:

```
// adds box shadow to form inputs on focus
        $(".newsletter-form input, .newsletter-form textarea").on('focus', function(){
                var newsletter = $(this).parent();
                newsletter.addClass("newsletter-focus");
        });
        $(".newsletter-form input, .newsletter-form textarea").on('focusout', function(){
                var newsletter = $(this).parent();
                newsletter.removeClass("newsletter-focus");
        });
```

It simply adds and removes the helper class at the appropiate time. Not much to customize here.

Page navigation

The page navigation makes use of bootstrap pills with an added class of "centered-pills" to the navigation wrapper that, as you have guessed, centeres the pills. Actual navigation is done by using some javascript code that hides and shows different parts of the page (divs) based on the pill selected.

Navigation HTML structure

As stated above the navigation in based on bootstrap pills with an added class of "centered-pills" to the navigation wrapper.

```
<div class="centered-pills navigation-wrapper">
        <ul class="nav nav-pills">
        <li class="active"><a href="home">Home</a></li>
        <li><a href="timeline">Timeline</a></li>
        <li><a href="contact">Contact</a></li>
</ul>
```

Each list item contains a link tag with an href attribute that holds the id of the div to be shown. That's how the actual navigation works. The id of the div is in the following format "name-page". So if the href attribute value is "timeline", the id of the div will be "timeline-page".

Navigation Javascript

The code that handles the navigation is pretty basic:

```
$(".nav-pills a").on("click", function(event){
                event.preventDefault();
                page = "#"+$(this).attr("href")+"-page";
                item = $(this).parent();
                item.siblings().removeClass("active");
                item.addClass("active");
```

```
            $(".page:visible").addClass("animated bounceOutUp").hide();
            $(page).show().removeClass("animated bounceOutUp").addClass("animated bounceInDown");
    });
```

It listens for a click event on one of the pills. When it happens it hides all divs and shows the div corresponding to the link clicked. The last two lines of code handle the animation events(using Animate.css).

Timeline

The timeline is thought to be used instead of an about page. You can include important events in chronological order. There are two kinds of timeline entries: basic and image. Use them as you see fit. There are examplse of both in the following block of code. This is the html structure for the timeline.

Html structure

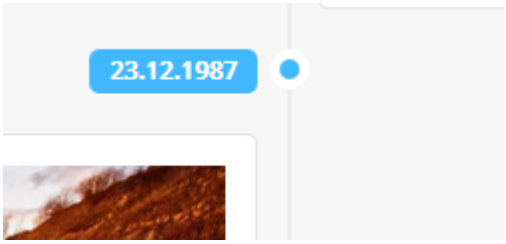
```
<div class="timeline-wrapper top-buffer">
    <div class="row">
        <div class="col-md-6 col-md-offset-6">
            <ul class="time-entry">
                <li><span class="badge">23.12.1987</span></li>
                <li>
                    <h3>Company is founded</h3>
                    <p>
                      Lorem ipsum dolor sit amet, consectetur adipisicing elit. Vero, dignissimos, ipsam
odio aspernatur deserunt iure consectetur veniam iusto in officia officiis ipsum repellendus eligendi
dolorum non qui aliquam tempore obcaecati.
                    </p>
                </li>
            </ul>
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            <ul class="time-entry">
                <li><span class="badge">23.12.1987</span></li>
                <li>
                    <img src="assets/img/img2.jpg" alt="">
                    <div class="hovercontent">
                        <h3>Second project</h3>
                        <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reprehenderit,
dolorem, non velit in eaque nulla cumque itaque.</p>
                        <button class="btn btn-ghost ghost-white">read more</button>
                    </div>
                </li>
            </ul>
        </div>
    </div>
</div>
```

The html structure does make use of bootstrap rows and columns for alignment. The example structure above contains two entries. The first one is the basic text entry and the second one is the image that display's content when the mouse is over it.

The entire timeline is wrapped inside a div with a class of "timeline-wrapper". Entries are basically unordered lists (<ul>) placed inside a div that uses bootrap column classes. Every entry along with the positioning div is placed in a separate row.

Styling the timeline

All of the styles that are aplied to the timeline are placed inside the “timeline.css” file in the assets/css/ folder.
Any aspect of the timeline can be styled by modifying or adding to this file. One element that is present in the timeline but not in the html structure is the little cirle that apears on the line for each entry.



This is added by javascript when the timeline is rendered. The circle is not an image but is made using pure css. That means it is fully customizable. You can modify it by modifying the class “cirlce” in the timeline.css file.

The container that appears when the mouse is over an image entry has a class of “hovercontent”.

The javascript
In principle you don’t need to worry about the javascript part other than initializing the timeline() function like so:

```
$(".timeline-wrapper").timeline();
```

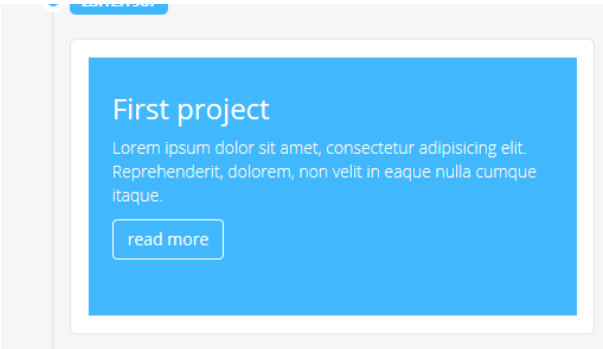
Also don’t forget to reference the timeline.js file in the html page, before the index.js file.

```
<script src="assets/js/timeline.js"></script>
```

If you would like to tweak the code here it requires some knoledge of javascript and jqeury. This is beyond the scope of this documentation.

Ghost buttons

The theme also containts a few classes called “btn-ghost” and “ghost-white”, “ghost-black”, etc. These can be used for buttons within the content that is shown when hovering over an image entry on the timeline. Like in the picture below. You can customise them or create new ones using the same method above.



Animations

Animations are provided by Animation.css (http://daneden.github.io/animate.css/) by daneden.

It's basically a acollection of css classes that animate the elements on screen. By the way the animations are done using CSS3 animations so they are not supported by old browsers.

The theme uses javascript events to trigger the animations by simply adding and removig a class to or from an element to animate it.

```
$("#home-page").show().addClass("animated bounceInDown");
```

The animated class is important as it flags tha this is an animatable object. The next class is the desired animation to be applied to the object. For a complete list of animation classes visit the link provided at the beginning of the chapter.

Since there is really no use in replicating the animate.css documentation here, please visit https://github.com/daneden/animate.css for the complete documentation.

Color Themes

This is probably the easiest customization that you can make to the theme. The theme uses a monotone color scheme, wich makes it really easy on the eyes and easy to customize. Color schemes are changed with the use of a separate css file.

The theme comes with 5 ready made color schemes. These css files are located in the "assets/css/color" folder. You can use any of them by including the corresponding css file after index.css (if you include it before it, it will get overwritten).

If you choose not to include one, the theme will default to the blue color scheme. If you want to make your own color scheme, feel free to do so. Just duplicate any scheme file and modify the colors as desired. For the color scheme, I used the rgba format, as this gives more control over the color. (especifically over transparency).

Final Words

I really hope you enjoy your new acquisition and find it easy to bend to your will. If you face any issues related to the theme, please contact me through themeforests author contact. I will be more than happy to help you, but please expect some delay in response.