WOOX

by Crumina templates

1.Template description 3
2.Working with GULP 4
3.CSS and JS files 11
3.1 SCSS and CSS files overview 11
3.2 CSS files loaded in template 12
3.3 JS files loaded in template 13
3.4 RTL version 14
4. Components and their usage 14
4.1 Balls animation 14
4.2 Video section 15
4.3 Popup-windows 16
4.4 "Load more" button 16
4.5 Google maps 17
4.6 SVG-icons 17
4.7 Sticky header 17

1. Template description

What a great product – WOOX! Its creative, clean, flexible and attractive HTML template. The WOOX was developed using the most popular streaming build system called Gulp. It made WOOX responsive, fast and easy to use and edit. SCSS/CSS styles in the template are based on SASS technology, that let us make the template well-structured and user-friendly. Due to SASS, users are able to change main template colors only in only one click by changing styles for global variables. (..\sass\theme-styles_variables.scss).

Another valuable advantage of this template is an accurate code structure with a table of contents, comments and block structure throughout the entire template. All necessary information about it and even more is located in this documentation, that comes bundled with the template. All this makes WOOX easy customizable, clean, with clear options and simple ways to change them.

The WOOX is completely responsive along with aesthetic and clean design and looks great on any extension and device. The template includes 40+ HTML pages, where you can find ready solutions for the entire page or separate components suitable for your project. The template comes with SVG icons, a huge variety of sliders, thematic widgets, galleries and many other things - this is a short list of the features you really get with this template. Our preview demo will tell you everything about WOOX and the solution will come along by itself!

If you have any questions about using our template you can always ask our fast and skilled support service. We are always ready to help you☺

2. Working with GULP

Gulp - is a tool for automating front-end development. It will help you automate routine tasks and speed up your work.

The main tasks that Gulp will help you to solve:

- Minimize and concatenate Javascript and CSS files;
- HTML minimization;
- Using CSS preprocessors - Sass, Less, Stylus and others;
- Images optimization;
- Automatic installation of vendor prefixes;
- Using template engines;
- Code validation and testing.

Useful links:

Gulp official site -gulpjs.com
Gulp at Github https://github.com/gulpjs/gulp
Gulp plugins catalog http://gulpjs.com/plugins/
Gulp documentation
https://github.com/gulpjs/gulp/blob/master/docs/README.md#articles

So, we give you a brief instruction on how to get started with the templates assembler - GULP.

Step1. Node.js installation

In order to install Node, visit nodejs.org site, click the green "Install" button. When the download is complete, start the application and install Node.

Step 2. NPM installation

At the command line, please go to the main folder of your project: cd/WOOX

Note: the path to the project can be individual, as it depending on the location on the PC.

If you created the project from scratch, you would have to run the command: npm init

It creates a package.json file that contains information about the project (project description and dependencies). But, since WOOX already contains this file, this step should be skipped.

Step 3.Gulp.js installation

Installation of Gulp is done with the command: npm install -g gulp

Here are some details:

-npm- is the application by which we install the package (plugin).
-install- we «tell» the command line to install the package.
-g- is a kind of flag indicating that we want to install the package globally.
-gulp- is the name of the package we want to install.

After we installed Gulp.js, let's make sure that everything is correct and enter the old command by analogy with Node and npm: gulp -v

The next command is to install Gulp locally in the project folder: npm install --save-dev gulp.

The only difference is that we specify in our "flag" that we record dependencies in our package.json.

Step 4.Gulp.js setup and launch

To make Gulp work, you need to explain to him what it should do. There are several methods: task, start, and path to files. All tasks, their settings and configurations are recorded in ../WOOX/gulpfile.js. file.

Let's analyze our file:

```
var gulp = require('gulp'),
   plumber = require('gulp-plumber'), // generates an error message
   prefixer = require('gulp-autoprefixer'), // automatically prefixes to css properties
   svgmin = require('gulp-svgmin'), // for minimizing svg-files
   sass = require('gulp-sass'), // for compiling scss-files to css
   browserSync = require('browser-sync'), // for online synchronization with the browser
   imagemin = require('gulp-imagemin'), // for minimizing images-files
   cache = require('gulp-cache'), // connecting the cache library
   htmlhint = require("gulp-htmlhint"), // for HTML-validation
   runSequence = require('run-sequence'); // for sequential execution of Gulp-tasks
```

With the help of these commands, we enable Gulp and plugins that perform our tasks.

It's important to note, that each plugin needs to be installed by analogy with Gulp installation before you can use it. Once they are installed, you can begin to set them up in the gulpfile.jsfile.

We won't review all the Gulp plugins and their possibilities since there are great many of them. But we will review the plugins used in WOOX in detail:

1) `plumber = require('gulp-plumber'), // generates an error message`

Installation: install--save-devgulp-plumber
Description:This plugin makes development easier so, that when there occurs an error, it outputs information about it in the console and doesn't interrupt implementation of tasks flow.
Documentation:https://www.npmjs.com/package/gulp-plumber

2) `prefixer = require('gulp-autoprefixer'),  // automatically prefixes to css  properties`

Installation:npm install—save-dev gulp-autoprefixer
Description:The plugin automatically complements CSS styles with vendor prefixes (-webkit, -ms, -o, -moz etc). These prefixes ensure support of CSS3 properties by browsers.

This plugin is used in our 'sass' task:

```
/*=========== Compile SCSS ==============*/
gulp.task('sass', function() {
   gulp.src('src/sass/*.scss')
      .pipe(plumber())
      .pipe(sass())
      .pipe(prefixer())
      .pipe(gulp.dest('./src/css'))
      .pipe(gulp.dest('./src/css'))
        .pipe(browserSync.reload({
            stream: true
        }))
});
```

Documentation-https://www.npmjs.com/package/gulp-autoprefixer

3) `svgmin = require('gulp-svgmin'), // for minimizing svg-files`

Installation: npm install --save-dev gulp-svgmin
Description:The plugin minimizes SVG files with help of Gulp. It is used in our 'svg-min' task:

```
gulp.task('svg-min', function () {
   gulp.src('src/svg-icons/*.svg')
      .pipe(svgmin({
         plugins: [{
            removeDoctype: true
         }, {
            removeComments: true
         }, {
            cleanupNumericValues: {
               floatPrecision: 2
            }
         }, {
            convertColors: {
               names2hex: true,
               rgb2hex: true
            }
         }]
      }))
      .pipe(gulp.dest('src/svg-icons'));
});
```

Documentation -https://www.npmjs.com/package/gulp-svgmin

4) `sass=require('gulp-sass'), // for compiling scss-files to css`

Installation: npm install --save-dev gulp-sass
Description: The plugin serves for compilation SCSS files into CSS. It is used in 'sass' task:

```
...
.pipe(sass())
…
```

The result is recorded in: .pipe(gulp.dest('./src/css'))
Documentation - https://www.npmjs.com/package/gulp-sass

5) `browserSync = require('browser-sync'), // for online synchronization with the browser`

Installation: npm install --save-dev browser-sync
Description:BrowserSync is one of the most useful plugins, that a developer would like to have. It actually gives you a possibility to start a server, where you can run your applications.

```
...
.pipe(browserSync.reload({
    stream: true
```

```
}))
…
```

And now imagine for a second that you have Gulp running, which automatically compiles, for example, scss, and in the same task uses browserSync. That is, after compiling scss, the browser automatically reboots the page and pulls up all compiled changes to css accordingly. Convenient, isn't it?

Documentation: https://www.npmjs.com/package/browser-sync

6) `imagemin = require('gulp-imagemin'), // for minimizing images-files`

Installation: npm install --save-dev gulp-imagemin
Description: This plugin is intended for working with graphics, it optimizes images and records the result in the specified path. It is used in 'images' task:

```
/*=========== Minimization IMAGE ==============*/
gulp.task('images', function(){
    gulp.src('src/img/*')
        .pipe(cache(imagemin({
            interlaced: true
        })))
        .pipe(gulp.dest('src/img'));
});
```

Documentation - https://www.npmjs.com/package/gulp-imagemin

7) `cache = require('gulp-cache'), // connecting the cache library`

Installation: npm install --save-dev gulp-cache
Description: The plugins using for data caching. In particular, it is used in the task 'images'.

A large number of pictures are processed much longer, so it would be quite good to add a cache in order to make images to cache much faster - this way we save our time.
Documentation - https://www.npmjs.com/package/gulp-cache

8) `htmlhint = require("gulp-htmlhint"), // for HTML-validation`

Installation: npm install --save-dev gulp-htmlhint
Description: This is a simple HTML validator that we used in the 'html-valid' task:

```
/*============= HTML-validator ==============*/
gulp.task('html-valid', function() {
   gulp.src("app/*.html")
      .pipe(htmlhint());
});
```

9) `runSequence = require('run-sequence'); // for sequential execution of Gulp-tasks`

Installation: npm install --save-dev run-sequence
Description: By default, when you run Gulp, the tasks in it are executed asynchronously, which can sometimes lead to very undesirable and not logical actions ... For example, starting a command to build a project "gulp build", Gulp may start to clear the folder app + to compile the same css, to concatenate js, etc., which will lead to unpredictable results. To avoid such situations we use the run-sequence plugin, which helps to establish dependencies on the tasks performance.

```
/*============= Join tasks ==============*/
gulp.task('default', function(callback) {
   runSequence([ 'sass', 'browserSync', 'watch', 'images'],
      callback
   )
});
gulp.task('build', function(done) {
   runSequence('sass', 'html-valid', 'svg-min', 'images',
'compress', done);
});
```

Documentation - https://www.npmjs.com/package/run-sequence

And now a few words about the launch of tasks and combining them. To run any task, you need to write on the command line: gulp 'task name' and run it.

We also have 2 sets of join tasks:

```
/*============= Join tasks ==============*/
gulp.task('default', function(callback) {
   runSequence([ 'sass', 'browserSync', 'watch', 'images'],
      callback
   )
});
gulp.task('build', function(done) {
   runSequence('sass', 'html-valid', 'svg-min', 'images',
'compress', done);
});
```

'default' - is used in the main development including ['sass', 'browserSync', 'watch']
'build' - is a full assembly of the project, that includes tasks ('sass', 'html-valid', 'svg-min', 'compress', 'images')

Official Gulp documentation:
https://gulpjs.com/
https://gulpjs.org/

3. CSS and JS files

3.1 SCSS and CSS files overview

The template stylesheet is based on the SASS language. All .scss files are located in the sass folder:

sass/theme-styles folder includes the main stylesheet, global files with styles for the whole template, namely:

- _content-block.scss – styles for main content blocks;
- _footer.scss - contains styles for footer and for all components, required with it.
- _global.scss - contains the styles for the main web components, such as "body", "a", "p", "table", "ol", "ul", "form", "input" идр.;
- -grid.scss - contains the modular grid styles (Bootstrap) of our project;
- _header.scss — stylesheet for the header, animation styles for sticky header and all elements bundled with it.
- _helper.scss – all auxiliary classes and their styles, that were used for building a project;
- _magnific-popup.scss – styles for pop-up windows of pictures and videos;
- _normalize.scss - a configurable file, by means of which browsers display all the elements more consistently according to modern standards;
- _overlay-menu.scss — styles for all varieties of pop-up menu.
- _page404.scss — styles for 404 page;
- _preloader.scss — style for preloader;
- _tabs.scss — tabs styles;
- _typography.scss — contains the styles responsible for the typography.
- _variables.scss - includes all global variables and their styles, main colors used in the template, font size, font families etc.

"..sass/blocks/.." and "..saas/widgets/.." folders include a stylesheet for blocks and widgets, used in the template.

"..saas/plugins/.." folder includes plugins styles, using in the template.

For your convenience, all stylesheet files are divided blockwise. There you can edit required styles and responsive settings for the particular block. Detailed folders structure is represented in saas/__table-of-content.scss file.

In the root of sass/.. folder you will find the following files:

blocks.scss, plugins.scss, theme-styles.scss, widgets.scss - it provides paths for proper compilation of the *.scss files into a appropriate files –

/css/blocks.css; ../css/theme-styles.css, ../css/widgets.css, ../css/plugins.css.

Font-awesome.scss file includes styles for Font-Awesome v5.0.6.

rtl.scss file contains styles for RTL version

3.2 CSS files loaded in template

CSS files, that load in <head> section of page in the current template are the following:

```
<!-- Theme Styles CSS -->
<link rel="stylesheet" type="text/css" href="css/plugins.css">
<link rel="stylesheet" type="text/css" href="css/theme-
styles.css">
<link rel="stylesheet" type="text/css" href="css/blocks.css">
<link rel="stylesheet" type="text/css" href="css/widgets.css">
<link rel="stylesheet" type="text/css" href="css/font-
awesome.css">
<link
href="https://fonts.googleapis.com/css?family=Playfair+Display:400
,400i,700i,900," rel="stylesheet">
```

- plugins.css – styles for all theme plugins
- theme-styles.css - main site's stylesheet;
- blocks.css – styles for blocks and template components;
- widgets.css — styles for widgets;
- font-awesome.css — styles for the icon font “Font Awesome»;
- https://fonts.googleapis.com/css?family=Playfair+Display:400,400i,700i,900 — connected external font file

3.3 JS files loaded in template

Javascript files, that load right before the closing <body> tag of a page in the current template are following:

```
<script src="js/method-assign.js"></script>
<!-- jQuery first, then Other JS. -->
<script src="js/jquery-3.3.1.min.js"></script>
<script src="js/js-plugins/ajax-pagination.js"></script>
<script src="js/js-plugins/bootstrap.js"></script>
<script src="js/js-plugins/bootstrap-datepicker.js"></script>
<script src="js/js-plugins/Headroom.js"></script>
<script src="js/js-plugins/imagesLoaded.js"></script>
<script src="js/js-plugins/isotope.pkgd.min.js"></script>
<script src="js/js-
plugins/jquery.datetimepicker.full.js"></script>
<script src="js/js-plugins/jquery.magnific-popup.js"></script>
<script src="js/js-plugins/jquery.matchHeight.js"></script>
<script src="js/js-plugins/jquery-circle-progress.js"></script>
<script src="js/js-plugins/jquery-countTo.js"></script>
<script src="js/js-plugins/material.min.js"></script>
<script src="js/js-plugins/parallax.js"></script>
<script src="js/js-plugins/particles.js"></script>
<script src="js/js-plugins/perfect-scrollbar.js"></script>
<script src="js/js-plugins/popper.min.js"></script>
<script src="js/js-plugins/select2.js"></script>
<script src="js/js-plugins/svgxuse.js"></script>
<script src="js/js-plugins/swiper.min.js"></script>
<script src="js/js-plugins/waypoints.js"></script>
<script src="js/js-plugins/map-shortcode.js"></script>
<script src="js/js-plugins/ion.rangeSlider.js"></script>
<!-- FontAwesome 5.x.x JS -->
<script defer src="fonts/fontawesome-all.js"></script>
<script src="js/main.js"></script>
```

- method-assign.js – a script that supports “assign” method in Internet Explorer.
- jquery-3.3.1.min.js –latest version of jQuery library connected
- ajax-pagination.js — load more scripts
- bootstrap.js – scripts for framework elements work
- bootstrap-datepicker.js — calendar script
- Headroom.js –script for sticky header
- imagesLoaded.js – script for correct isotope work
- isotope.pkgd.min.js — isotope sorting scripts
- jquery.datetimepicker.full.js – calendar script

• jquery.magnific-popup.js – script for opening photo and video in popup windows

• jquery.matchHeight.js - script for setting the same height to elements in one row

• jquery-circle-progress.js – script for building round progress bar

• jquery-countTo.js – script for counters

• material.min.js –script for animation and building form elements

• select2.full.js — script for selectors

• parallax.js –script for footer animations

• particles.js – script for circle animation

• perfect-scrollbar.js – script for custom scrollbar

• popper.min.js – script for tooltip

• select2.js – script for different popup list

• svgxuse.js - a simple polyfill that fetches external SVGs referenced in use elements when the browser itself fails to do so

• swiper.jquery.min.js — sliders

• waypoints.js - Waypoints is a library that makes it easy to execute a function whenever you scroll to an element

• map-shortcode.js – script for Google Map

• ion.rangeSlider.js – script for building Range Sliders

• fontawesome-all.js — Font-awesome script

• main.js - main custom javascript file of a template.

3.4 RTL version

One of the greatest features of the Utouch template is RTL version. In order to activate it, you need to find and uncomment the following code in the HTML markup:

```
...
<!--Styles for RTL-->
<!--<link rel="stylesheet" type="text/css" href="css/rtl.css">-->
...
```

to this look:

```
...
<!--Styles for RTL-->
<link rel="stylesheet" type="text/css" href="css/rtl.css">
```

4. Components and their usage

4.1 Balls animation

4.1.1 "Soaring balls" animation

This animation was implemented by using the plugin particals.js

GitHub -https://github.com/VincentGarreau/particles.js

In order to add this animation to any section, you need:

- for the section add the following classes and 1 data attribute:

```
<section data-settings="particles-1" class="crumina-flying-balls particles-
js">..</section>
```

• data-settings="particles-1" - data-attribute specifies file name(here is our example ..\src\js\particles-1.json), which is responsible for the settings parameters for animation. (here you can set the number of elements, shape, color, etc.)

In order to get more detailed information, you can read the official documentation, moreover, you can generate online any animation and get ready json with settings: https://vincentgarreau.com/particles.js/

• «crumina-flying-balls particles-js" — classes, that are responsible for plugin initializing and styles

Note: Since the plugin generates the canvas for drawing elements, we strongly do not recommend placing this animation on too "high" sections (> screen heights), because this may affect the loading of the GPU.

In order to deactivate it, you need to delete classes stated above.

4.1.2 Balls animation - "background-parallax" in footer section

This animation was implemented using the plugin parallax.js

GitHub -https://github.com/wagerfield/parallax

In order to add this animation to any section, you need:

-for the section add the class:"crumina-parallax-background". Then, add the following code to the class:

```
<section class="crumina-parallax-background">
<div class="scene">
  <div data-depth="0.2" class="layer first-layer"></div>
  <div data-depth="0.6" class="layer second-layer"></div>
  <div data-depth="1" class="layer third-layer"></div>

</div>
</section>
```

class = "scene" - the parent class responsible for animation initializing

class = "layer" -layers which animated between each other during moving a mouse. Each layer has own picture used as background.
Data-depth = "1" - the data-attribute responsible for the degree of layer displacement relative to each other

In order to deactivate it, you need to delete layers markup and classes, stated above.

4.2 Video section

Blocks which contain video content have a main parent class «crumina-our-video». In order to run the video, you need to use a button, e.g:

<svg class="woox-icon icon-play-button-5"><use xlink:href="svg-icons/sprites/icons.svg#icon-play-button-5"></use></svg>

class js-popup-iframe – is obligatory to perform popup action.
Attribute href="" -we specify either any URL or path to some content, that will be opened in the popup window.

4.3 Popup-windows

The standard HTML markup of popup window has a structure:

```
<div class="... has-popup">
<a href="#" class="btn btn--round btn--secondary btn--transparent btn--hover-decoration js-open-popup">..</a>
<div class='window-popup'>
   <div class='content'>
<a class='js-open-popup popup-close' href='#'>
  ..
</a>
...
   </div>
</div>
</div>
```

“has-popup” – it is the main parent block, which responsible for window location. It can be any independent block, the main condition is the presence of the class "has-popup"

Following is the button for opening the pop-up window. In general, it can be any element, not necessarily a link, most importantly it must have the class "js-open-popup", which is responsible for opening the window.

.. - the main parent of popup-window block
 - wrap for window content
<a class='js-open-popup popup-close' href='#'> - button for closing the window

You can easily modify the content of any pop-up window and add them to any place as well.

4.4 "Load more" button

In order to add content to a page, by clicking the **“load more”** button you need to use the library:

```
<script src="js/js-plugins/ajax-pagination.js"></script>
```

The button has the following look:

```
<a href="#" class="btn btn--medium btn--secondary btn--transparent" id="load-more-button" data-load-link="portfolios-to-load.html" data-container="portfolio-grid">load more works</a>
```

The button code includes the following data-attributes:

• id="load-more-button" — mandatory id for initialization

• data-load-link="portfolios-to-load.html" — here you specify the path to the HTML page, that contains required content to load;

• data-container="portfolio-grid" — data attribute, indicating container`s id, and place, where the content will be added.

4.5 Google maps

In order to add a map to the page, just add the following markup:

```
<!-- Google map -->

<div class="google-map" data-map-style="”>
    <div class="map-canvas"></div></div>
```

data-map-style – data attribute, by which you can record styles for maps, the styles format you can find in markup as well.

4.6 SVG-icons

All the svg-icons used in the template are located in the .. \ src \ svg-icons \ .. folder.

There is a file icons.svg in the .. \ src \ svg-icons \ sprites \ folder. This file is an svg-sprite, generated via http://app.fontastic.me. It contains a set of svg-icons, each of which has its own unique id. To call and place any icon from the sprite, you should use the html markup (for example):

<svg class="woox-icon icon-close"><use xlink:href="svg-icons/sprites/icons.svg#icon-close"></use></svg>

- class="woox-icon" - mandatory class, which is used for all theme icons
- href="svg-icons/sprites/icons.svg#icon-close — this is a "link" to the unique id of the necessary icon from our sprite.

4.7 Sticky header

The template comes with sticky header functionality, that makes pages navigation together with the menu much more convenient. Sticky header is available in 4 animation variations - Slide, Swing, Flip, Bounce. You can choose one you like most. For this find the following code in **js/main.js file**:

```
CRUMINA.fixedHeader = function () {

  // grab an element
  $header.headroom(
    {
      "offset": 10,
      "tolerance": 5,
      "classes": {
        "initial": "animated",
        "pinned": "slideDown",
        "unpinned": "slideUp"
      }
    }
  );
};
```

... and replace the default animation effect with the following:

for Slide - "pinned": "slideDown",
"unpinned": "slideUp"

for Swing - "pinned": "swingInX",
"unpinned": "swingOutX"

for Flip - "pinned": "flipInX",
"unpinned": "flipOutX"

for Bounce - "pinned": "bounceInDown",
"unpinned": "bounceOutUp"

A detailed documentation on sticky header can be found by the following links:

http://wicky.nillia.ms/headroom.js/

https://github.com/WickyNilliams/headroom.js

In order to deactivate sticky header function, you just need to delete the line id="site-header" from HTML markup, namely from:

<header class="header" id="site-header">...</header>

To bring it to this look:

<header class="header">...</header>.