

MetroUI **Documentation**

Version 2.1 (2B248d)

Revision 3 - Windows Edition



Table of Contents

Foreword	3
----------	---

Chapter One

Basic Setup

Setting up your first page	4
Setting up MetroUI	5
You've created your first page - What now?	5

Chapter Two

Adding content

Headers	6
Texts	7
Form Elements	7
Progress Controls	9
Lists	9
Flyouts	10

Chapter Three

Navigation

Navigation with Bars	12
Navigation with Menus	13
Navigation using Links	13
Minimal layout for pages	14

Chapter Four

The JavaScript core

Parameters	15
Modals: Notifications, Alert & Confirm	15
Preloaders	16
Manual page navigation	16

Chapter Five

Theming your application

Dark and Light themes	18
Accent Colors	18
Colored Pages	19

Chapter Six

Updates & Compatibility

Compatibilty to older versions	20
Future updates	20

Chapter Seven

Contributing

Contribute to the GitHub project	21
----------------------------------	----

Foreword

Finally! Someone is using MetroUI!

I'm just kidding. Thank you for downloading and using MetroUI! This documentation will show you how to use MetroUI and its functions. It will also show you when to use something and when to not use it. Believe me: It's easy!

As you may have read before MetroUI is the most awesome way to create HTML based applications for Windows 8 and Windows Phone, but with a special focus on being usable on any platform. It took me quite some time and weeks of testing to achieve such a level of precision, but I give you this for free. No donations or anything.

I really love the user interface of both Windows 8 and Windows Phone, but why should I learn another programming language when I can use my existing skill to create something on my own? I have always wanted to create my own Windows application, but C# or Visual Basic were too complicated for me. That's why I created MetroUI. Easy to use, even easier to distribute. No need for struggling with store restrictions. Just distribute it over the internet! For free!

I hope you have as much fun as I have with MetroUI!
Wishing you the best luck creating your application,

Janik Schmidt
Creator of MetroUI

Chapter One

Basic Setup

Setting up your first page

Using MetroUI is super easy. You just need basic knowledge of HTML. Everything that you probably don't know will be explained in this documentation.

Let's start by firing up your favorite code editor. I will be using Coda 2 for references.

First, create a new document and set the syntax mode to HTML (if not done automatically). Then add the Doctype, which, in case of MetroUI, is the HTML5 Doctype:

```
<!DOCTYPE html>
```

Continue with adding the default page containers after the Doctype:

```
<html>
  <head>
  </head>
  <body>
  </body>
</html>
```

The **head** part is where your title and all your scripts, stylesheets and other stuff goes in. The **body** part holds your application visuals. We will need that in Chapter Two: Adding Content.

It's time to give your application a name. Add a **<title></title>** to the **<head>** section and insert a text to name your application.

Next up is adding MetroUI. MetroUI has two files that need to be included: The script file (MetroUI-2.1.js) and the stylesheet (MetroUI.css). It is recommended to store these files in separate folders, just for clarity. Let's imagine your HTML file is in the same folder as your JavaScript folder ("js") and your CSS folder ("css") and that you're using the non minified version of MetroUI. If that's the case, add the following lines after the **<title>** tag:

```
<link rel="stylesheet" type="text/css" href="css/MetroUI.css">
<script type="text/javascript" src="js/MetroUI-2.1.js">
```

In the **<body>** section you need to create at least a page container and a page. Pages have an unique identifier so that when you have multiple pages you can easily navigate through them. Initial pages should always have "index" as the identifier:

```
<div class="pages">
  <div class="page" data-page="index">
    <div class="content"></div>
    <!-- Content will be added in the "content" container -->
  </div>
</div>
```

When you save your page and view it in your browser, you will probably notice that your page is still white. That's because we haven't added any content yet, but we will take care of that in Chapter Two. Before that we should set up the JavaScript core of MetroUI.

Setting up MetroUI

Before we add some content, we should configure the JavaScript core. If you don't need much customization add this right before the closing `</body>` tag:

```
<script>
    var app = new MetroUI();
</script>
```

The `MetroUI()` takes some parameters if you want to customize the setup process. You will learn more about parameters in Chapter Four: The JavaScript Core.

Now we can go straight to adding content to our application. But wait, didn't we miss something?

You've created your first page - What now?

The very last thing we need to check before adding content is: Do you have an idea of what you want to create? What kind of application should it be? Is MetroUI even suitable for this application?

Let me tell you that I have successfully created an application using MetroUI 2.0 some time ago. It was an internet radio player/information page for my school's web radio. It worked pretty well, but there were many things I was unhappy about. That's why I rewrote MetroUI completely, and the 2.1 update was born.

But it's up to you what your application should be capable of. If you have an idea that's great. You should keep in mind that MetroUI is not yet suitable for interactive applications, like calculators or calendars. Support for these types of applications will follow later this year.

TL;DR

If you have an idea of what kind of application you want to create, continue to Chapter Two. If not, get creative and come back to Chapter Two.

Chapter Two

Adding Content

MetroUI supports a vast array of content types: Headers, Texts, Form Elements, Lists and even Context Menus. This chapter will show you how and when to use these types.

Headers

There are three types of headers: Title headers, Section headers and Content headers. Title and Section headers are stored in a `<header>` element right at the beginning of the page.

Title headers are `<h1>` and Section headers are `<h2>`.

```
<div class="page" data-page="index">
  <header>
    <h1>Title</h1>
    <h2>Section Title</h2>
  </header>
  ...
</div>
```

Section titles can be colored with the current accent color using the `colored` class. They can also be used in the Content section.

In the Content section, Content headers are used to differentiate page sections, like various settings.

```
<div class="page" data-page="index">
  ...
  <div class="content">
    <h3>Page section title</h3>
    ...
  </div>
</div>
```

Header Bars

Sometimes, Bars are used instead of simple headers. They are commonly used if you are on a page that hides the menu (more on that later). Header Bars contain a back button and are colored with the current accent color. The Title header is not supported in bars, so the page title must be specified using a Section title.

Back buttons are only required when you're not on the initial page.

```
<div class="page" data-page="not-index">
  <header class="bar">
    <div class="navigate-back"></div>
    <h2>Page Title</h2>
  </header>
  ...
</div>
```

Additionally, colored Section titles are not supported in Header bars. They will simply appear white, just like normal Section titles.

Texts

Inserting texts is very easy. Just add a `<p></p>` tag to the content section, insert some text and you're good to go.

```
<div class="page" data-page="index">
  ...
  <div class="content">
    ...
    <p>Lorem ipsum dolor sit amet...</p>
  </div>
</div>
```

Form Elements

Input Boxes

Input boxes can be created using the `<input>` and the `<textarea>` tag. On desktop only three types are important:

```
<input type="text">
<input type="password">
<textarea></textarea>
```

The first is for text and shows plain input. The second is for passwords and shows your input as bullets. The third is a multiline text field.

There are actually more supported types, but they are more important on mobile devices as they control which keys the virtual keyboard shows.

email, number, search, tel, text, url

Buttons

Buttons can be added using the `<button>` tag. They can be colored by adding the `colored` class and disabled with the `disabled` class. A function callback can be specified with the `onclick=""` attribute or with JavaScript Event Handlers.

```
<button>Click me!</button>
<button class="colored">Click me!</button>
<button onclick="alert('You clicked me')">Click me!</button>
```

Using the `inline` class, you can arrange buttons in a line. This class needs to be applied to every inline button except for the last one.

```
<button class="inline">I'm left</button>
<button class="inline">I'm middle</button>
<button>I'm right</button>

<button>I'm not in that group</button>
```


Checkboxes and Radio Buttons

Checkboxes and Radio Buttons are almost the same. They are stored in their own container which holds an input tag and the element itself. Unlike Switches, Checkboxes should only be used in a form as they temporarily store a setting instead of permanently.

Prechecked Checkboxes and Radio Buttons can be achieved by adding the **checked** attribute to the input tag inside the container.

```
<div class="checkbox">
  <input type="checkbox">
  <div class="checkbox-inner">
    <label>Unchecked</label>
  </div>

<div class="checkbox">
  <input type="checkbox" checked>
  <div class="checkbox-inner">
    <label>Checked</label>
  </div>
```

The same goes for Radio Buttons:

```
<div class="radio">
  <input type="radio">
  <div class="radio-inner">
    <label>Unchecked</label>
  </div>

<div class="radio">
  <input type="radio" checked>
  <div class="radio-inner">
    <label>Checked</label>
  </div>
```

Switches

Switches are checkboxes that store settings for a longer time (or even permanently). They have features that Checkboxes have (**disabled**, **checked**), but they also have Dynamic Labels. These labels change according to their state using predefined Strings ("On" and "Off"), which can also be changed during the initialization process.

```
<div class="switch">
  <input type="checkbox">
  <div class="switch-inner"></div>
  <label>Off</label>
</div>
```

Sliders

Sliders are bars that control values inside a certain range. The user can change this value, either continuously or in steps (10, 20) using the **step** attribute. Sliders have a minimum value of 0 by default, which can be changed with the **min** attribute. If no **max** attribute is given, the slider ranges from 0 to 1. Starting values can be defined with the **value** attribute, which must be between the minimum and the maximum values. Sliders can also be disabled using the **disabled** attribute.

```
<div class="range">
  <input type="range" min="0" max="100" step="0.1" value="50">
</div>
```

Vertical sliders can be created using the `vertical` class:

```
<div class="range vertical">
  <input type="range" min="0" max="100" step="0.1" value="50">
</div>
```

Progress Controls

Quote from MSDN:

A progress control provides feedback to the user that a long-running operation is underway. A *determinate* progress bar shows the percentage of completion of an operation. An *indeterminate* progress bar, or a progress ring, shows that an operation is underway.

Progress Bars

Progress Bars show the progress of a ongoing operation, commonly used with percentages. They have a title, showing what is running this operation, and a label, showing the status of that operation. As iOS supports the `<progress>` tag only since iOS 7.1, MetroUI has its own implementation. This allows us to use custom animations when progress is being made.

```
<div class="progress">
  <progress max="100" value="67"></progress>

  <p class="title">Determinate Progress</p>
  <div class="progress-inner"></div>
  <p class="label">Status related message</p>
</div>
```

Instead of the progress bar itself, MetroUI shows a container ("progress-inner") that receives its value directly from the progress bar. The best part is that when the value in the progress bar changes, the width of the inside container changes too! This happens using so called "Mutation Observers" which detects attribute changes and reports them to the inside container if needed.

Progress Rings

Progress Rings are also known as Indeterminate Progress Controls. They inform the user that something is happening but without any progress.

Progress Rings cannot be used in this build of MetroUI (2B248d). The problem is that the Rings are just GIF images. Once loaded they animate continuously, no matter if they are hidden or not. The solution is to append a random string to the GIF's URL which results in reloading the file every time it is requested. Another possible solution is to use base64 encoded images, but the resulting string is way to long to use within JavaScript.

Lists

Lists contain multiple options the user can select. They are often used in forms, but they can also be used in different places, like settings. Lists are even used within the Theme Selector.

Lists are generated from their `<select>` tag and the options inside. All you have to do is to wrap this selector into a list container.

```
<div class="list">
  <select>
    <option value="test" selected="">Test</option>
    <option value="test2">Test 2</option>
    <option value="test3">Test 3</option>
  </select>
</div>
```

While initializing MetroUI, Lists automatically select the first option that has the **selected** attribute. If there is no selected option, the first entry in the list is selected.

Lists can be, like anything else, disabled using the **disabled** class.

Flyouts

Dropdown Menus

If a button inside an Application bar has multiple options to choose from, a Dropdown Menu is shown to list these options. Dropdown Menus can also be used inside the page.

Using JavaScript, opening a Dropdown Menu can be bound to buttons, links or anything else that has click event handlers. There are two types of elements a Dropdown menu can display: Buttons and separators. These can be added to the JavaScript call. Let's assume we have a button that should call a Dropdown Menu:

```
<button id="dropdown">Open Dropdown</button>
<script type="text/javascript">
  document.getElementById("button").onclick = function() {
    app.context({
      target: this,
      buttons: [{
        text: "Alert Something",
        type: "button",
        href: "javascript:app.alert('Something')"
      },
      {
        type: "separator"
      },
      {
        text: "Open External Page",
        type: "button",
        href: "http://google.de"
      }
    ]
  });
};
</script>
```

Dropdown Menu links can also contain JavaScript functions. This way allows you to use Dropdown Menus for Navigation.

...

```

buttons: [{
    text: "Alert Something",
    type: "button",
    href: "javascript:app.navigation.loadPage('page2')"
},
...

```

More on manual Navigation in Chapter 4.

Flyouts with HTML content

Unlike Dropdown Menus, Flyouts show simple HTML content. Flyouts can contain any of the previously named content types, which makes them like a separate page.

The Flyout method receives, next to the button array, a HTML string. This string can be manually specified or coming from a template. Templates are scripts with **text/template** as a type.

For this example, we use the same button as before, but this time we add a template and a container using the **context-content** class.

```

<button id="dropdown">Open Dropdown</button>

<script type="text/template">
    <div class="context-content">
        <p>All items will be permanently removed from your cart.</p>
        <button>Empty my cart</button>
    </div>
</script>
<script type="text/javascript">
    document.getElementById("button").onclick = function() {
        app.context({
            target: this,
            html: document.getElementById("template").innerHTML
        });
    };
</script>

```

Chapter Three

Navigation

Navigation can be done in multiple ways: Using Bars, Menus or links. This chapter shows you how to navigate through internal and external pages using these types of Navigation.

Navigation with Bars

Bars consist of two segments: The Navigation Bar and the Application bar. The Navigation bar is - as the name might suggest - responsible for navigation. The Application bar holds buttons and other functions that normally are not for, but can be used as navigation.

NOTE: You cannot use Menus and Bars together.

Navigation Bars

Navigation Bars are located at the top of a page and contain links to different pages or page categories. If a page matches the page ID or the category ID of that particular Navigation Bar link, the link gets highlighted.

The Bar container, which holds both Navigation and Application bars, is placed before the Pages container:

```
<div class="bars">
  <div class="navigation-bar">
    <div class="link">
      <p>Link</p>
    </div>
  </div>
</div>
```

Links can be added using a container with the `link` class. If you want to use that container as a link, wrap it inside an `<a>` tag and give it a `href` attribute. More on Links later in this chapter.

Application Bars

Even though Application Bars are actually not responsible for Navigation, they can still be used for this. Normally Application Bars hold several buttons with actions to choose from. These buttons have an icon inside them so the user knows what he is doing right away.

The Icon documentation will follow as they are not completely done yet (There are about 200+ icons for AppBar Buttons).

Application Bars belong into the same container as Navigation Bars. The structure is simple: The first child is always the "More" icon, three little dots that call both NavBars and AppBars.

```
<div class="application-bar minimal">
  <div class="icon-more"></div>
```

Right after the "More" icon, there are two additional containers, this time for buttons. One sits on the left side, the other on the right.

```

<div class="left"></div>
<div class="right"></div>

```

Depending on which site you want your icon to appear, add a button either to the "left" or the "right" container:

```

<div class="left">
  <div class="icon icon-help">
    <label>Button</label>
  </div>
</div>
<div class="right">
  <div class="icon icon-help">
    <label>Button</label>
  </div>
</div>

```

This example will display two buttons in the AppBar, one on the left, the second on the right. Both buttons will have a question mark in them ("icon-help") and are labelled with "Button".

Most buttons will hold a Dropdown Menu. You learned how to create Flyouts and Dropdown Menus in Chapter 2 ("Flyouts").

Navigation using Menus

Menus are located at the left hand side of the page. They contain a title and links to different pages. Even though Menus consist of frames, using multiple menu pages is currently not supported.

NOTE: You cannot use Menus and Bars together.

The structure is similar to Navigation Bars. The only differences are the frames (they will be used in a later build) and the Menu label.

```

<div class="menu">
  <div class="frame" data-frame="index">
    <label>MetroUI</label>
    <a href="#">
      <div class="link selected">
        <p>Playground</p>
      </div>
    </a>
    <a href="page_frames/menu_3.html">
      <div class="link">
        <p>Menu 2</p>
      </div>
    </a>
  </div>
</div>

```

Navigation using Links

Links can be used anywhere, in the Content section, in NavBars or in Menus. When clicked, they automatically load a local page according to their href attribute. If this attribute is "#" (hash), the initial page (in general "index") is loaded and any previously loaded page is discarded.

When you add the **external** class, you can load external pages (like "http://google.com") or run JavaScript functions (like "javascript:alert('test')"). This works with any link in MetroUI.

```
<!-- Simple text link -->
<a href="pages/page2.html">Load second page</a>
<a href="http://google.com" class="external">Load google.com</a>

<!-- Menu/NavBar link -->
<a href="pages/page3.html">
  <div class="link">
    <p>Page 3</p>
  </div>
</a>

<!-- Run JavaScript -->
<a href="javascript:alert('Hello World')" class="external">Hello World!</a>
```

Minimal layout for pages

Local pages that are loaded with Links or otherwise have a minimal layout. This layout is stripped down to the page itself. There is no need for a pages container or **html** and **body** tags. Unfortunately, scripts specified in the new page are not loaded. A solution will be provided in a future build.

The following code is a complete local page:

```
<div class="page external no-bar" data-page="test">
  <header class="">
    <div class="navigate-back"></div>
    <h1>Title 2</h1>
    <h2 class="colored">Header</h2>
  </header>

  <div class="content">
    <p>This is an example page.</p>
  </div>
</div>
```

Chapter Four

The JavaScript Core

Parameters

While initializing MetroUI, several parameters can change the behavior of this process. This table lists all parameters available and what their effect on the initialization process is.

Parameter name	Function	Default value
availableThemesWin	Controls which themes are available to use. (Desktop)	See Chapter 5
alertConfirmButton	Controls the default confirm button title in Alerts and Confirm dialogs	"ok"
alertCancelButton	Controls the default cancel button title in Alerts and Confirm dialogs	"cancel"
modalDefaultTitle	Default header when modals are displayed	"Untitled"
modalDefaultContent	Default content when modals are displayed	"No content"
theming	Enables/disables theming	true
lists	Enables/disables lists	true
switches	Enables/disables switches	true
radios	Enables/disables radios	true
checkbox	Enables/disables checkboxes	true
progress	Enables/disables progress	true
pageLoading	Enables/disables page loading	true
pageTransitions	Controls if pages should transition from one to another Disable for better performance	true
pageTransitionMode	Sets the page transition mode 1 - Animate page as one part 2- Animate header and content separately 3 - Like 2, but animate Titles and Section titles separately	3
notificationTransitions	Controls if Notifications should slide in Disable for better performance	true
barsOnContext	Controls if NavBars and AppBars should appear when the user clicks the right mouse button	true

Modals: Notifications, Alert & Confirm

Notifications

Notifications are generally used by apps to inform that something happened. Notable examples are a received mail or attaching removable media. MetroUI can display notifications in different occasions.

To display a notification, all you have to do is to bind the JavaScript function to an event handler, like clicking on a button. When displaying that notification, you can change the title and the message. You can also specify a callback when dismissing that notification. In a future build you will be able to add icons and use custom colors.

`app.notify(title, message, callback)`

title - String: Specifies the title that should be displayed on the Notification

message - String: Sets the message the notification displays

callback - function: A function that gets triggered when the notification is clicked and dismissed

Callback functions do not get called if the user clicks the Notification close button.

Alert & Confirm

An Alert prevents the user from further interaction with the app. If an Alert is displayed, the user must first dismiss it to continue working. The same applies to Confirm dialogs.

Unlike Alerts, Confirm dialogs have two buttons. They are normally displayed if the user is about to do something potentially harmful, so he must confirm the dialog to continue.

Alerts can contain callbacks while Confirm dialogs must contain callbacks.

`app.alert(title, message, [callback]);`

`app.confirm(title, message, callback);`

title - String: Specifies the title that should be displayed on the Alert or Confirm dialog

message - String: Sets the message the Alert or Confirm dialog displays

callback - function: A function that gets triggered when the user confirms the message (clicking "ok")

Preloaders

Preloaders are often displayed when there is an ongoing process that requires a longer amount of time, like loading pages.

Preloaders have a customizable title and a variable duration. If no duration is set, the application must dismiss the Preloader when the process is done.

`app.preloader(title, [duration]);`

title - String: The title of the Preloader

duration - Float: The duration the Preloader is displayed for - (optional)

The Preloader GIFs have a duration of 3765 ms.

Manual Page Loading

Sometimes simple links can't be used for navigation, like when the user clicks on a notification. This is where Manual Page Loading comes in.

The function accepts a local file as an URL. If the loaded page has a minimal layout, it gets appended to the pages container and the app transitions to the new page.

`app.navigation.loadPage(url);`



`url - String`: The local path of the page that should be loaded

There is also a simple function for navigating back from page to page:

```
app.navigation.back();
```

Dark and Light themes

MetroUI supports different themes for pages. Themes can be applied by adding either "dark" or "light" to the **data-bg** attribute of the application's **body** tag.

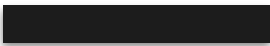

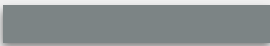




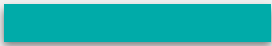



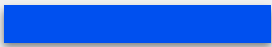






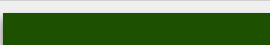



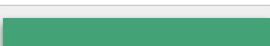

Theme name	Applies to
 dark	Page Backgrounds, Buttons, Switches, Checkboxes, Radio buttons, Sliders, Progress bars, Input fields, Lists
 light	








There is also a List that changes the background for you:

```
<div class="list theme">
  <select onchange="app.design.changeTheme(this)">
    <option value="light" selected="">Light</option>
    <option value="dark">Dark</option>
  </select>
</div>
```

Accent Colors

Several Elements support Accent Colors, such as colored Buttons, Switches or Menus. There are different Accent Colors for Windows and Windows Phone mode. Accent Colors can be applied by adding the theme name to the **data-tint-win** (Windows) and **data-tint-wp** (Windows Phone) attributes of the application's **body** tag.

Theme Name (Windows)	Primary Color	Theme Name (Phone)	Primary Color
dark-gray		lime	
light-gray		green	
dark-red		emerald	
red		teal	
orange		cyan	
yellow		cobalt	
bright-yellow		indigo	
bright-green		violet	
green		pink	
dark-green		magenta	
darker-green		crimson	
dark-lime		red	

dark-teal		orange	
dark-purple			
purple			
darker-pink			
dark-pink			
pink			

Accent colors are applied to the following elements:

Colored Pages, colored Headers, Links, colored Buttons, Switches, Sliders, Progress Bars, Lists, Notifications, Preloaders, Menus

There is an Accent selector, making it easier to change the Accent Color on the go. However, at this state every accent color has to be added manually. This will change in the next Technical Preview.

Colored Pages

Colored Pages can be achieved by adding the **colored** class to the **body** tag. Using Colored Pages is similar to using the Dark Theme. Colored Headers will be rendered in white.

When using Colored Pages, it is not recommended to display Notifications or Bars. Due to similar colors they are hard to distinguish from the background.

Colored Pages only work in Windows Mode. MetroUI also has a switch for toggling Colored pages:

```
<div class="switch colored-pages">
  <input type="checkbox" onchange="app.design.coloredPages(this)">
  <div class="switch-inner"></div>
  <label>Off</label>
</div>
```

It currently works by adding the function for changing Colored Pages to the onchange event. This function checks the state of the Checkbox, so Colored Pages can only be changed from these Switches

Chapter Six

Updates & Compatibility

Compatibilty to older versions

If you have been using MetroUI before (2.0), I have bad news for you. As MetroUI 2.1 is a complete rewrite, it is **not compatible** in any way with previous versions of MetroUI. There are a lot of changes concerning DOM structure, and few things are the same. But the biggest change is the JavaScript part. MetroUI 2.1 does no longer require jQuery. That means that most of the functions names have changed, and other functions require you to strictly keep to the new DOM structure.

You could of course try to use MetroUI 2.1 on an application made with MetroUI 2.0. But I will guarantee you that you will run into errors.

Future Updates

Future Updates for MetroUI will probably not have game breaking changes. They will all be about bugfixes and feature additions. If there is some bigger changes, you will be informed soon enough.

As this build (2B248d) is the first Technical Preview (I know that this term is from Microsoft, but it is what it is), there will be more frequent updates. You can always be informed about new updates via Twitter:

@Sniper_GER - Main Developer - https://twitter.com/Sniper_GER

@DigitalFDev - Development Group - <https://twitter.com/DigitalFDev>

Chapter Seven

Contributing

Contribute to the GitHub project

MetroUI is Open Source. Another reason why it's free. Man, I can't emphasize that enough.

It's free.

No cost at all.

Todo es gratis.

Keine versteckten Kosten.

Gratuit.

Бесплатно.

無料。

That should be enough for now. Well, back to the topic.

Contributing to the GitHub project helps users to get new updates faster, new and improved features will be available sooner. If you have an idea, contact me on Twitter (@Sniper_GER) so we can talk about it.

If you have a quick bug fix, make a pull request on GitHub.

If you experienced an issue, post it to Twitter or GitHub. Or even better: Both.

You can help to shape the future of MetroUI. Wait, is that from Microsoft again? Wasn't intended.

All you have to do is to open the repository on GitHub by clicking on this link:

<https://github.com/SniperGER/MetroUI>

Windows, the Windows logo, Windows Phone, the Windows Phone logo and the Microsoft logo are registered trademarks of Microsoft, Corp.

MetroUI is licensed unter GNU GPLv2.

© 2014-2015 Janik Schmidt. All rights reserved.