# World-Wide Developer Conference

## Genero Browser Client - Customization

Neil J Martin

**JW Marriott Cancùn Spa & Resort**
**September 26-29**, 2017

FOUR Js
The Power of Simplicity

# Intro

It's all about me

**Neil Martin – neilm@4js.com**

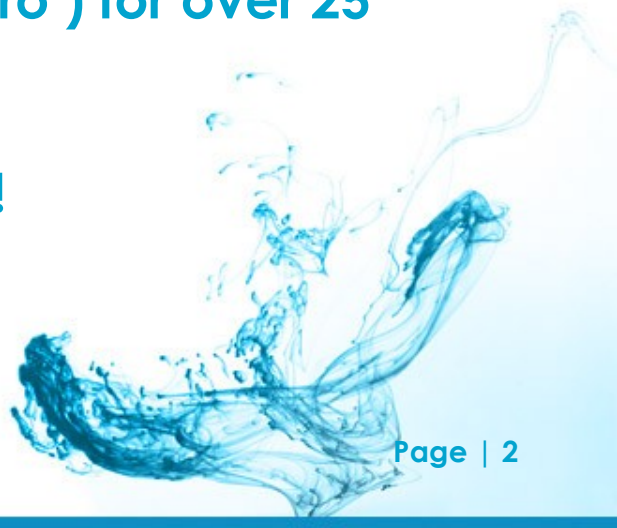
**I work from the UK office.**

**My Roles are support / training / consultancy / development.**

**Currently also working on Genero Cloud.**

**My background is in developing 4gl applications, I have been using 4gl ( InformixRDS / Four Js BDL / Genero ) for over 25 years!**

**I do NOT consider myself a Web Developer!**



# Overview - About the session

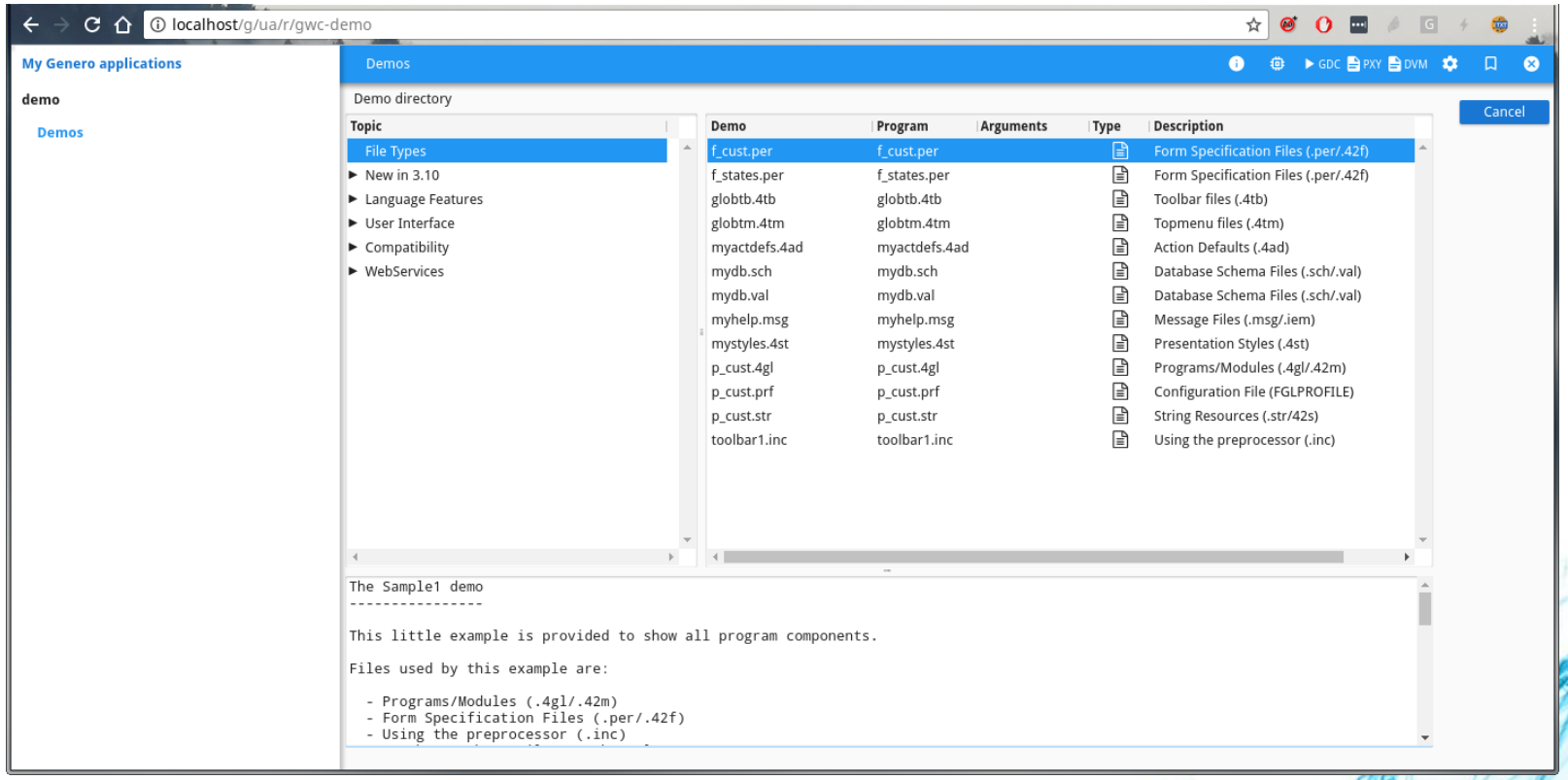
Goals of this session

- Run GBC default with default demos
- Setup the GBC dev environment
- Run the GBC dev build with default demos
- Create custom build folder for the custom GBC files
- Setup GeneroStudio to build GBC
- A test Application
- Colors and Theme
- Headers and Footers
- Packaging and deploy GBC
- More GBC customizations

# GBC Default Test

Basic sanity test

**Before we start, let's make sure the current default GBC is working and we can run the default Genero demos!**



# Setup the GBC Dev Environment

Tools of the trade

**The GBC uses various standard web development tools, these need to be installed in your development environment so you can build the GBC Runtime from the Project sources.**

```
$ sudo yum install git unzip nodejs
$ wget https://nodejs.org/dist/v6.11.3/node-v6.11.3-linux-x64.tar.xz
$ tar xvJf node-v6.11.3-linux-x64.tar.xz
$ export PATH=node-v6.11.3-linux-x64/bin:$PATH
$ sudo npm install -g grunt-cli
```

# Setup the GBC Dev Environment

Our first build

## Once the require packages are installed you need to set up the GBC project folder

```
$ unzip fjs-gbc-1.00.39-build201709041139-
project.zip
$ cd gbc-1.00.39/
$ npm install
$ grunt
```

## When complete you should have the 'default' custom GBC built.

# Setup the GBC Dev Environment

Configure for running

Before you can test the 'default' custom GBC you need to make sure the GAS can find it.
There are a couple of ways to tell the GAS how to find your custom GBC.
One method is a symbolic link in $FGLDIR/web_utilities/gbc

```
$ ln -s $HOME/gbc-
1.00.39/dist/customization/default
$FGLDIR/web_utilities/gbc/mydef
```

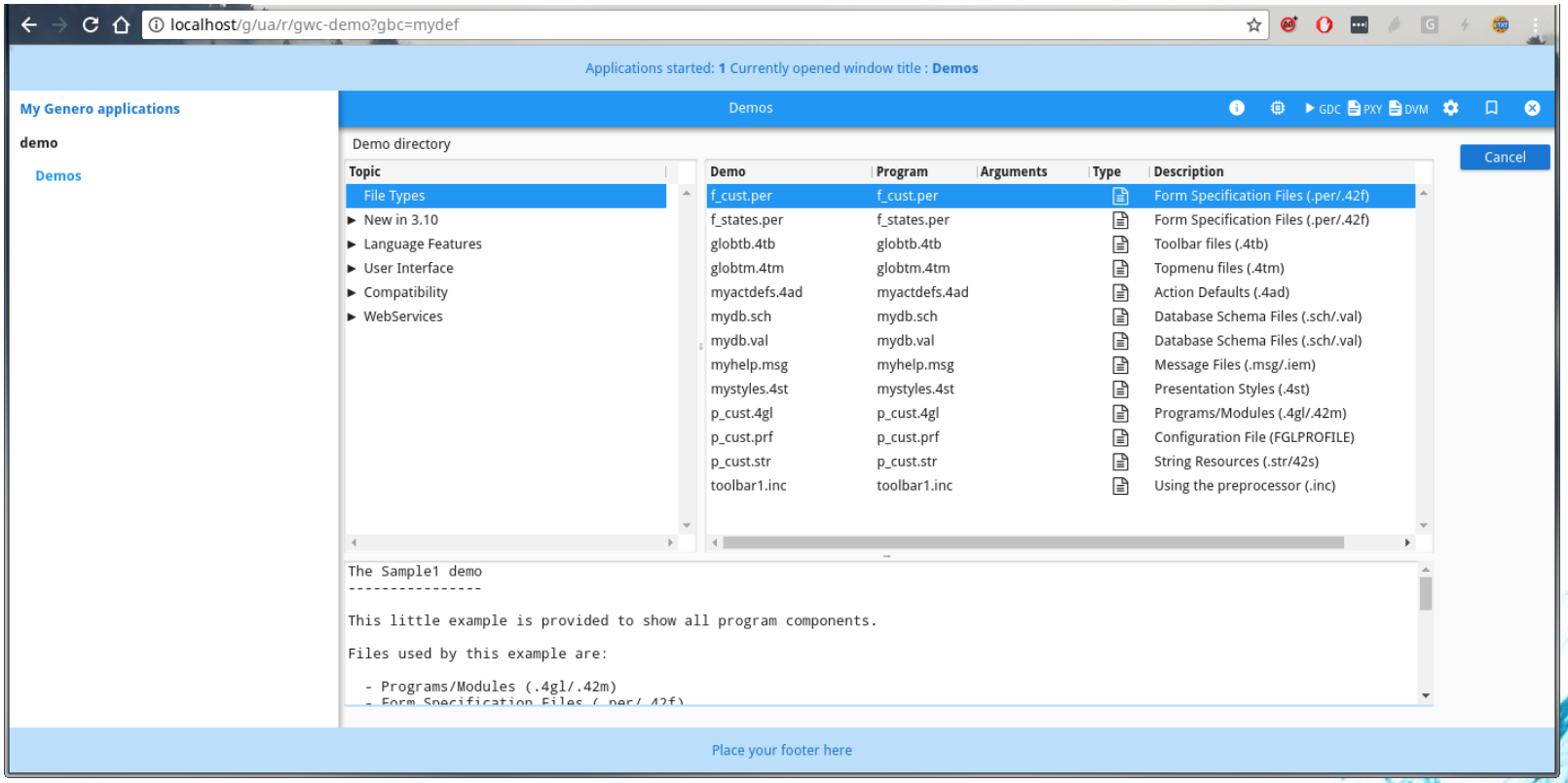
Another method is to modify the as.xcf for the "res.path.gbc.user".

# GBC 'Default' Custom Run

The results of all that effort

## Test the 'default' custom GBC:

**http://<server>/gas/ua/r/gwc-demo?gbc=mydef )**



# First Customization

First a new build

I recommend testing your application with the default Custom GBC build before doing any customizations.

Next, setup your own sources folder like this:

- Create a new folder in your sources location for your custom GBC files. ( so they are part of your Source Code Control )
- Create a link in the GBC customization build folder to this 'new folder'
- Change the custom.json to build that custom folder
- Make a small change to the theme file and do your first build
- Create a link in $FGLDIR/web_utilities/gbc to this new custom build
- Run the new build in your browser.

# First Customization

Do the work

```
$ cd $HOME/gbc-1.00.39/
$ cp -r customization/default $MY_SRC_DIR/mygbc
$ ln -s $MY_SRC_DIR/mygbc/ customization/mygbc
$ vi custom.json
    change:     "customization": "customization/default"
    to this:    "customization": "customization/mygbc"
$ grunt       ( make sure we can build our new folder )
$ ln -s $HOME/gbc-1.00.39/dist/customization/mygbc
   $FGLDIR/web_utilities/gbc/mygbc
$ grunt dev    ( This will automatically build after any changes )

-> new terminal
$ cd $MY_SRC_DIR/mygbc
$ vi theme.scss.json
    do some changes & save them
```

# The 'default' Customization

Anatomy of customization

The 'default' customization folder contains these sub-folders:

- js – JavaScript and template html files.
- locales – Localization files for text messages.
- resources – Images.
- sass – scss files. (syntactically awesome stylesheets!)
- and the 'theme.scss.json' file.

I suggest removing most of the demo files from js and sass to leave only the files relating to the header.

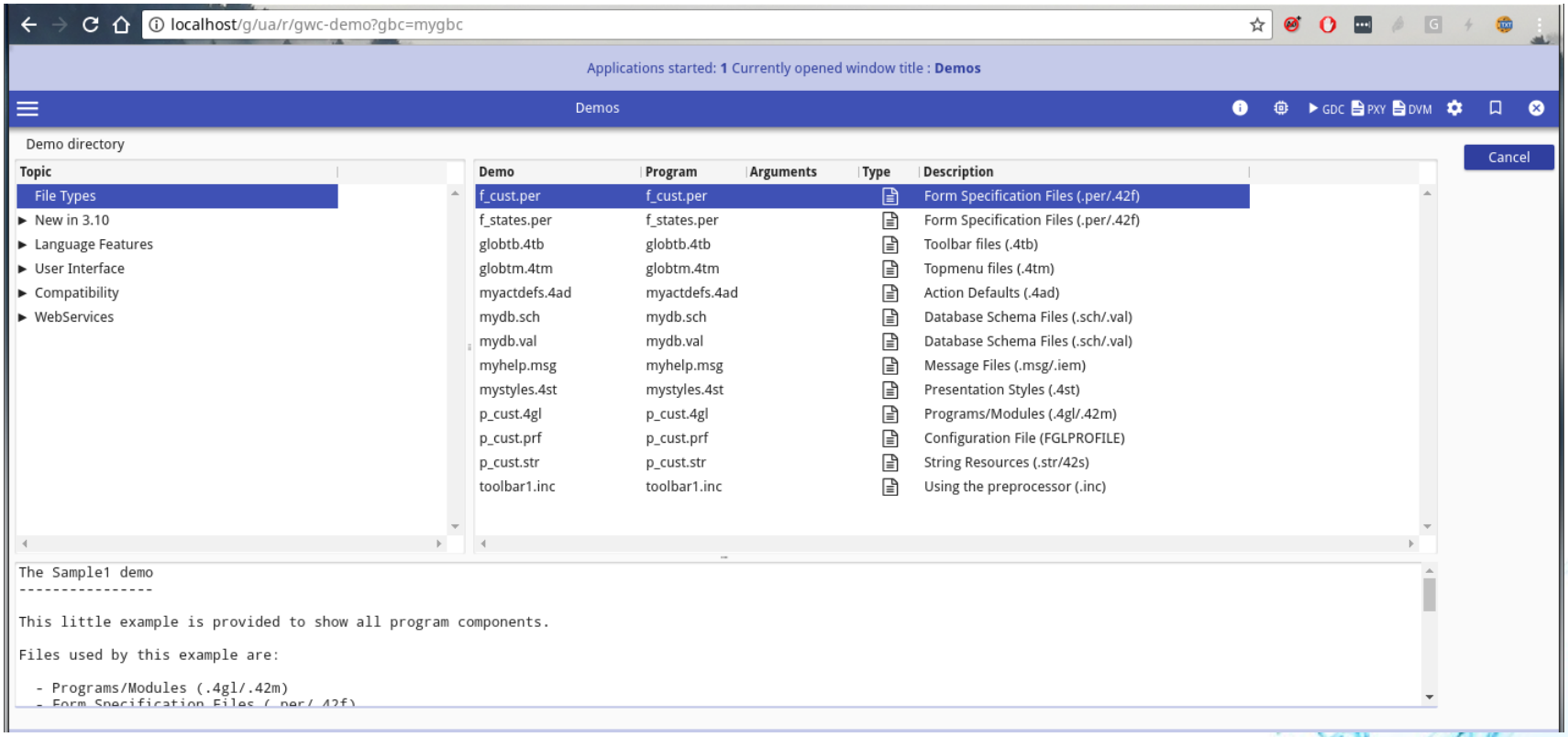
```
js/MyHeaderBarWidget.js
js/MyHeaderBarWidget.tpl.html
js/MyMainContainerWidget.js
js/MyMainContainerWidget.tpl.html
sass/customization.scss
sass/MyMainContainerWidget.scss
```

# GBC 'Default' Custom Run

At last, our own colours!

## Test the custom GBC:

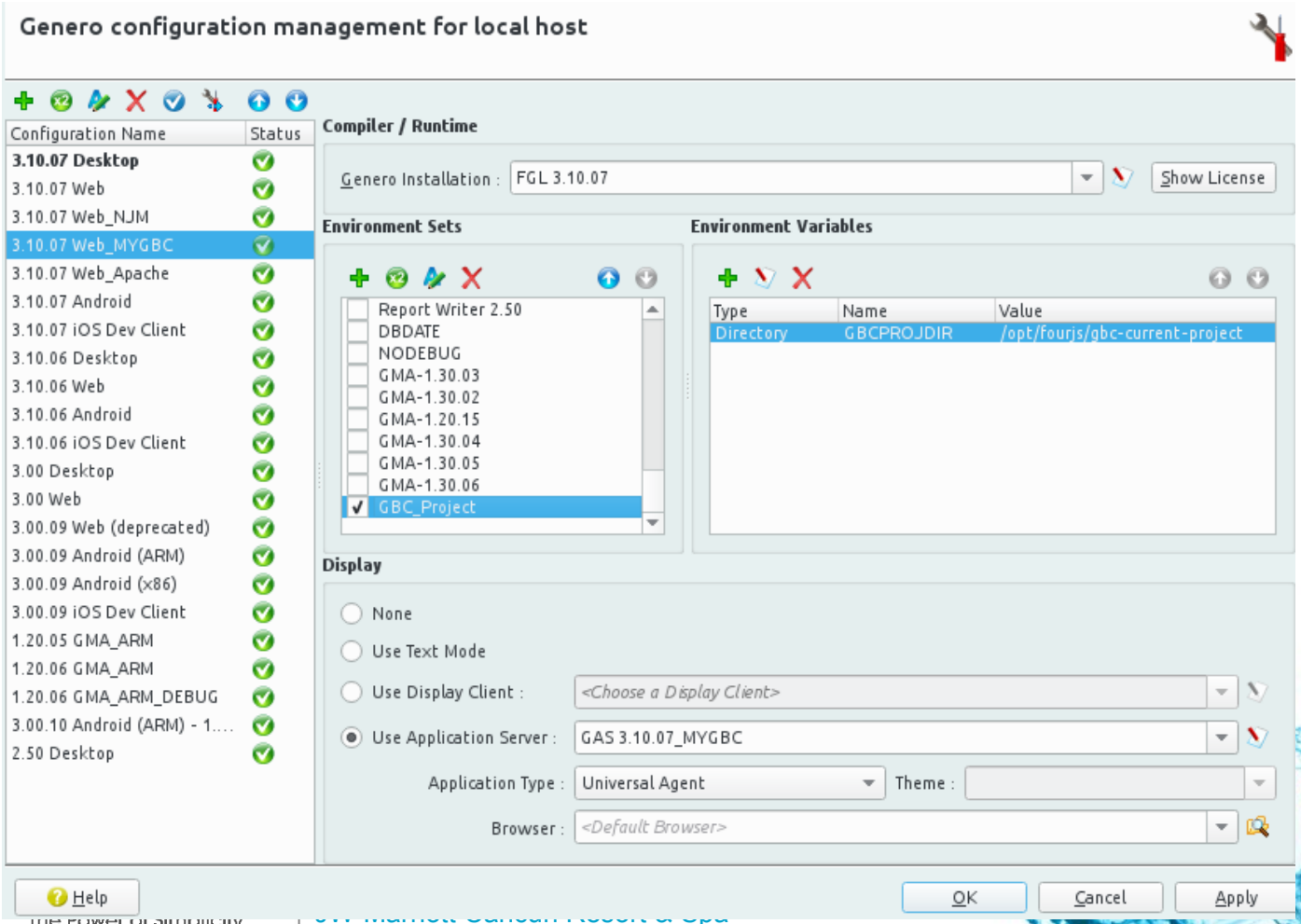
**http://<server>/gas/ua/r/gwc-demo?gbc=mygbc )**



# Genero Studio

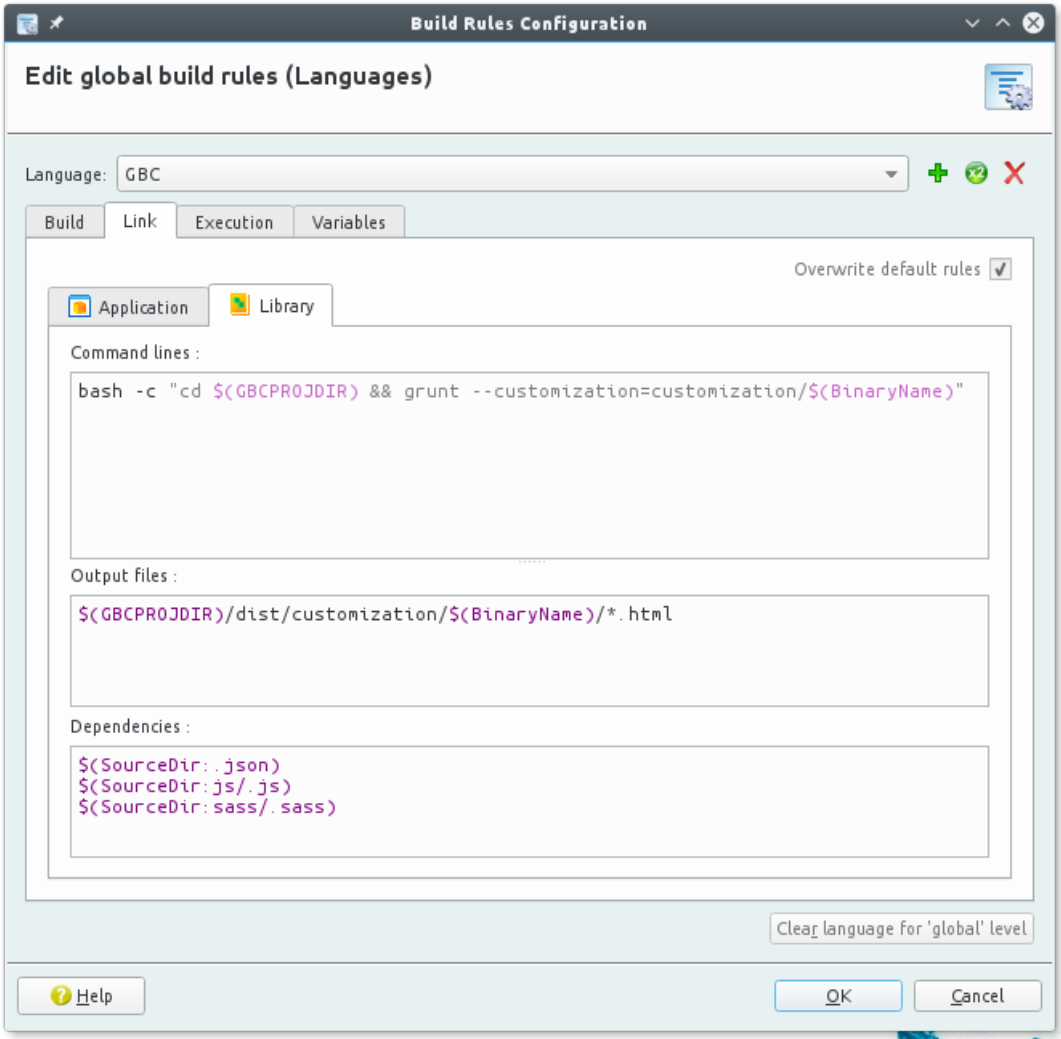
Custom Builds rules

## Genero Studio can also build your custom GBC.

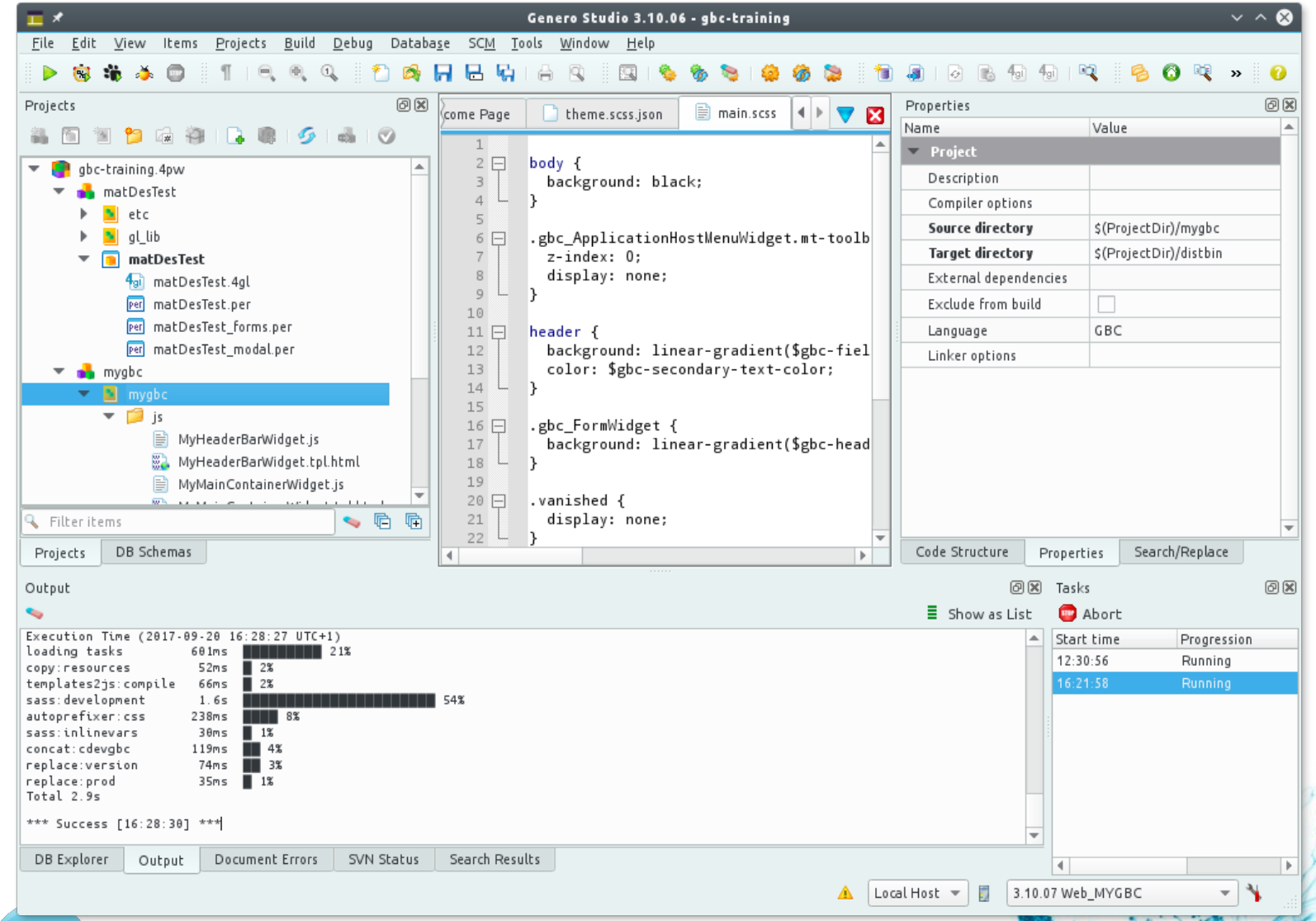


# Genero Studio

## Custom Builds rules

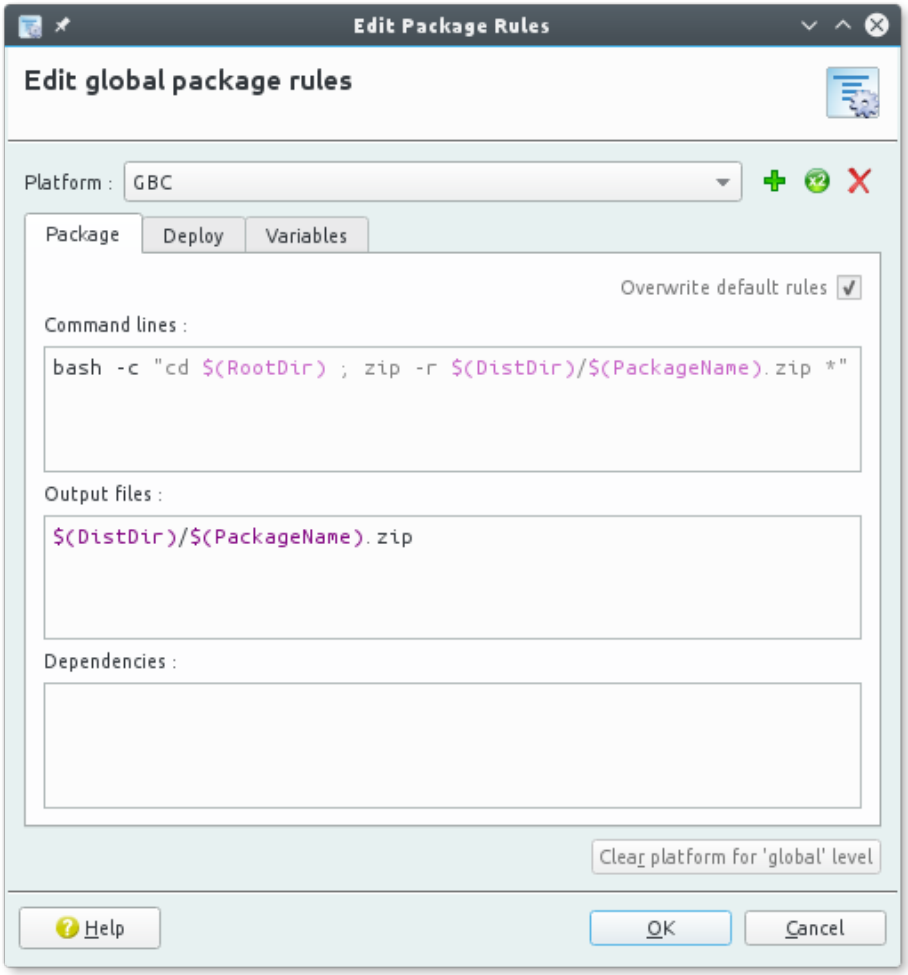


# Genero Studio



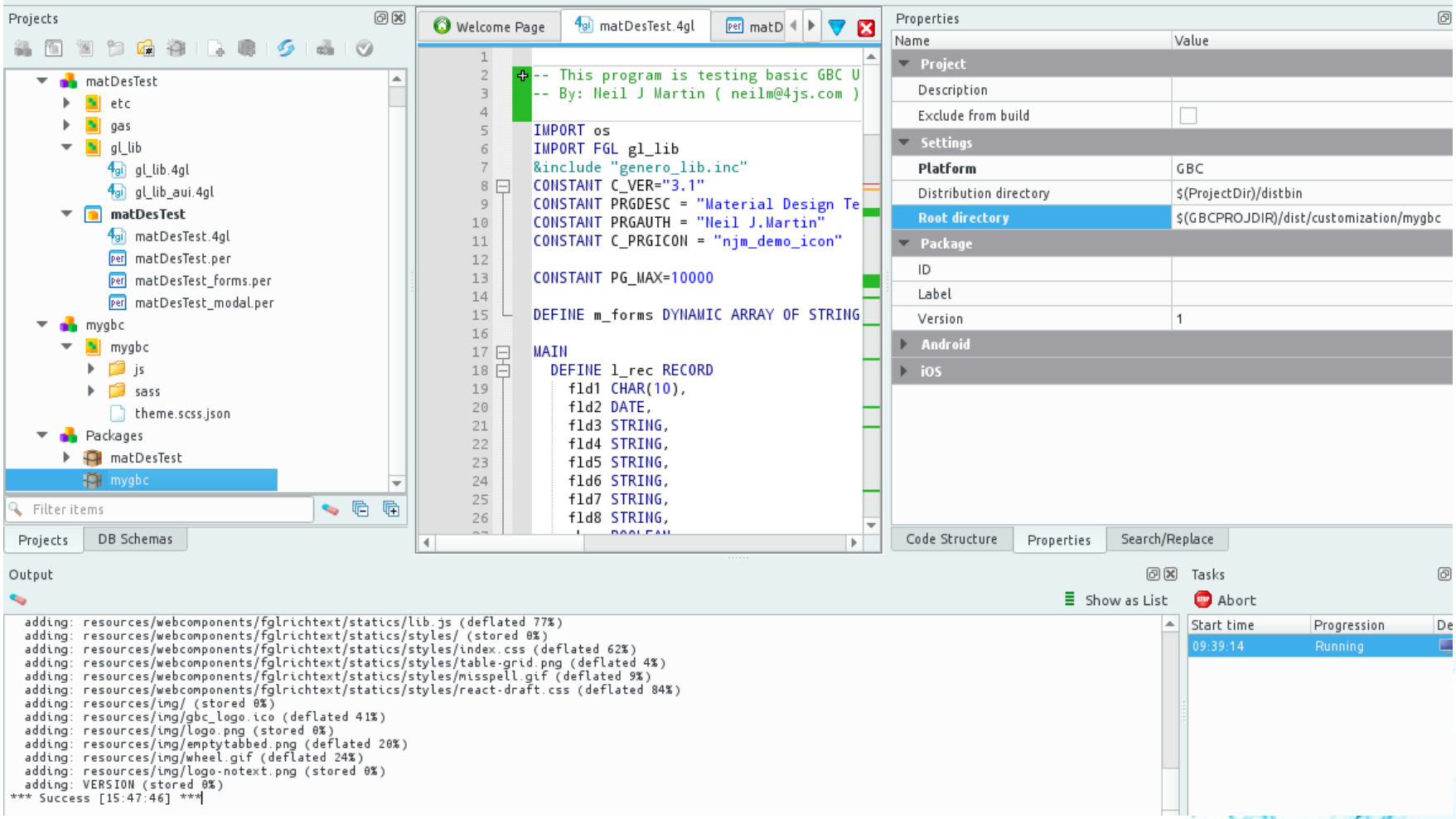
# Genero Studio

## Custom Package rules



# Genero Studio

## Custom Package rules



# Test Application

A testing asset

**A good idea is to write a small stand-alone test program to run using Custom GBC to see all your changes.**

**The 'Test' program should use all the GUI elements you use in a similar way to your main application, but with the advantage that you don't have to navigate through various menus and sub programs to get to see any specific UI features you may use.**

**Another useful feature would be if the program can open and show any of your 42f files.**

# Colors ( Colours )

All the colours of the rainbow

**Coloring the GBC should be done by changing the theme json file. I'd recommend removing most color changes from the .4st file so the GBC can use the theme.**
**NOTE: The .4st colors will override the GBC theme colors.**

**The theme file allows us to change the colors for a 'material design'. See: https://material.io/guidelines/style/color.html#color-color-palette**

# Colors ( Colours )

All the colours of the rainbow

Style – Color

| Deep Purple | |
|---|---|
| 500 | #673AB7 |
| 50 | #EDE7F6 |
| 100 | #D1C4E9 |
| 200 | #B39DDB |
| 300 | #9575CD |
| 400 | #7E57C2 |
| 500 | #673AB7 |
| 600 | #5E35B1 |
| 700 | #512DA8 |
| 800 | #4527A0 |
| 900 | #311B92 |

| Indigo | |
|---|---|
| 500 | #3F51B5 |
| 50 | #E8EAF6 |
| 100 | #C5CAE9 |
| 200 | #9FA8DA |
| 300 | #7986CB |
| 400 | #5C6BC0 |
| 500 | #3F51B5 |
| 600 | #3949AB |
| 700 | #303F9F |
| 800 | #283593 |
| 900 | #1A237E |

| Blue | |
|---|---|
| 500 | #2196F3 |
| 50 | #E3F2FD |
| 100 | #BBDEFB |
| 200 | #90CAF9 |
| 300 | #64B5F6 |
| 400 | #42A5F5 |
| 500 | #2196F3 |
| 600 | #1E88E5 |
| 700 | #1976D2 |
| 800 | #1565C0 |
| 900 | #0D47A1 |

# Colors ( Colours )

All the colours of the rainbow

**The theme file is a JSON file and allows you to use the 'Material Design' color names. After changing this file you must rebuild the GBC using the 'grunt' command.**

```
"gbc-header-color"                      : "$mt-indigo-200",
"gbc-primary-medium-color"              : "$mt-indigo-900",

"gbc-primary-background-color"          : "$mt-indigo-100",
"gbc-primary-color"                     : "$mt-indigo-600",
"gbc-primary-light-color"               : "$mt-indigo-900",

"gbc-secondary-background-color"        : "$mt-indigo-50",
"gbc-secondary-text-color"              : "$mt-white",
"gbc-secondary-color"                   : "$mt-white",

"gbc-field-disabled-background-color"   : "$mt-grey-100",
"gbc-field-disabled-background"         : "rgba(0,0,0,0.44)",
"gbc-field-background-color"            : "$mt-white",
"gbc-disabled-color"                    : "$mt-grey-300",

"gbc-separator-color"                   : "$mt-white",
"gbc-message-color"                     : "$mt-grey-800",
"gbc-error-color"                       : "$mt-red-800",
```

# More Than Colors

All the colours of the rainbow and more

**You can also override the colors and other styles for any element in the GBC by editing or creating a scss file for that element.**
**For example, look at the default MyMainContainerWidget.scss**
**This is controlling the style of the 'header' object.**

```
> header {
  padding: 1em;
  background-color: $gbc-primary-light-color;
  color: $gbc-primary-color;
  text-align: center;
}
```

# Working with SaSS

Getting Sassy

When you create your .scss files you must add them to the customization.scss file so that included in the build.

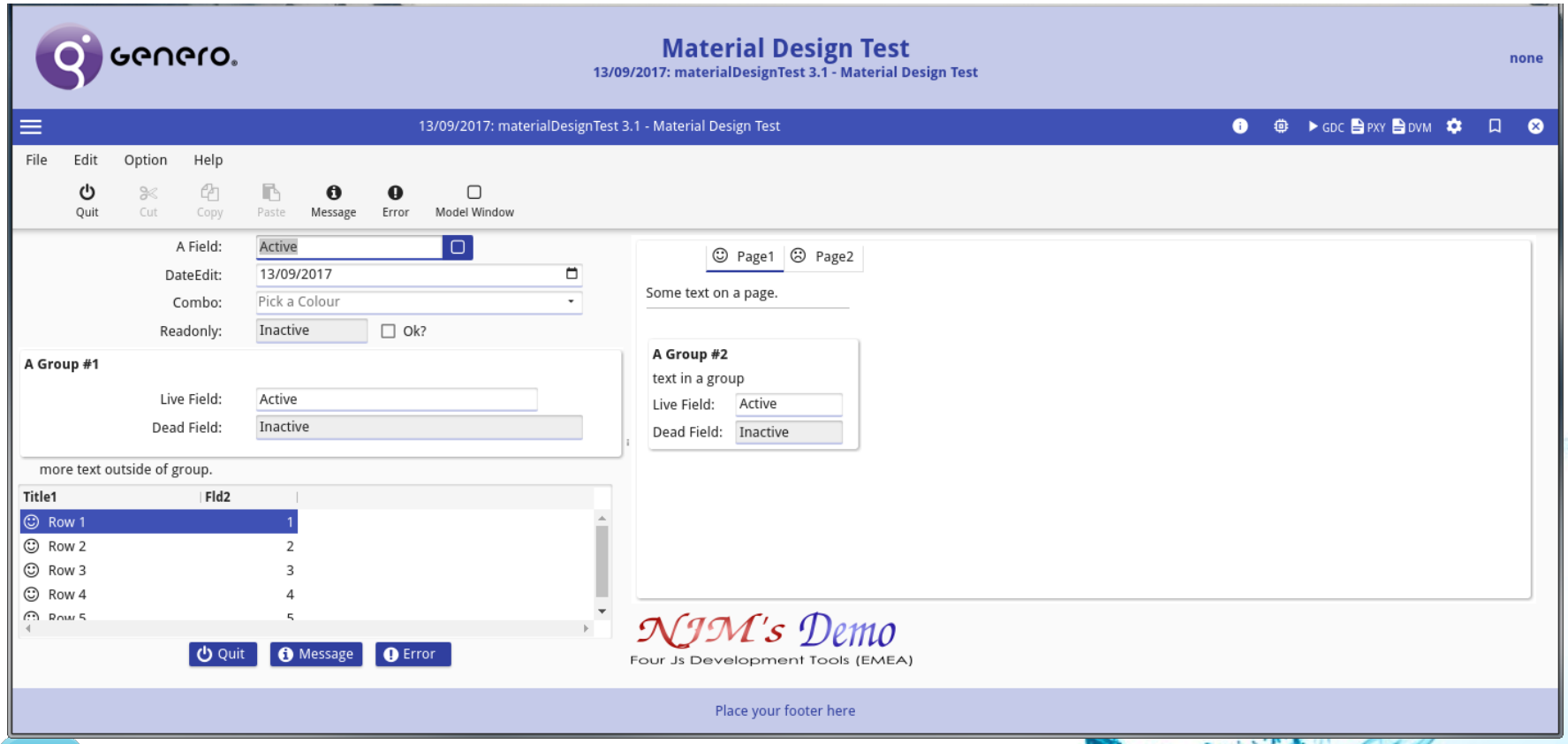
```
@import "MyMainContainerWidget";
@import "MyHeaderBarWidget";
@import "main";
```

Here I've added 'MyHeaderBarWidget' and 'main' to the customization.scss file.

# Headers

Getting a head of ourselves

**Once we are happy with our basic styles, we can work on the heading by adding a title and logo.**



# Headers

Getting a head of ourselves

## My Code adds a the Application Text into the header as well as the form title.

`js/MyHeadingBarWidget.tpl.html:`

```
<div>
  <table width="100%">
    <tr>
      <td id="logocell" width="20%" align="left">
        <img src="./resources/img/logo.png"/>
      </td>
      <td width="60%">
        <h1 class="MyHeaderBarWidget-banner"></h1>
        <b id="MyWinTitle" class="MyHeaderBarWidget-title"></b>
      </td>
      <td width="20%" align="right">
        <b id="dyntext">none</b>
        <span style="display: none">Apps: <b class="MyHeaderBarWidget-counter"></b></span>
      </td>
    </tr>
  </table>
</div>
```

# Headers

## js/MyHeadingBarWidget.js

```
cls.MyHeaderBarWidget = context.oo.Class(cls.WidgetBase, function($super) {
  return {
    __name: "MyHeaderBarWidget",
    _model: null,
    _appsCount: null,

    constructor: function() {
      $super.constructor.call(this);
      this._appsCount = 0;
      this._model = new cls.ModelHelper(this);
      this._model.addNewApplicationListener(this.onNewApplication.bind(this));
      this._model.addCloseApplicationListener(this.onCloseApplication.bind(this));
      this._model.addCurrentWindowChangeListener(this.onCurrentWindowChanged.bind(this));
      elb = this.getElement().querySelector(".MyHeaderBarWidget-banner");
    },

    onNewApplication: function(application) {

    onCloseApplication: function(application) {

    onCurrentWindowChanged: function(windowNode) {
      var elt = this.getElement().querySelector(".MyHeaderBarWidget-title");
      // Set the banner text to the value set by ui.interface.setText()
      elb.textContent = windowNode.getAncestor("UserInterface").attribute('text');
      // Set the header sub title to the window text.
      if (windowNode) {
        elt.textContent = windowNode.attribute('text');
      } else {
        elt.textContent = "<NONE>";
      }
    }
```

# Headers

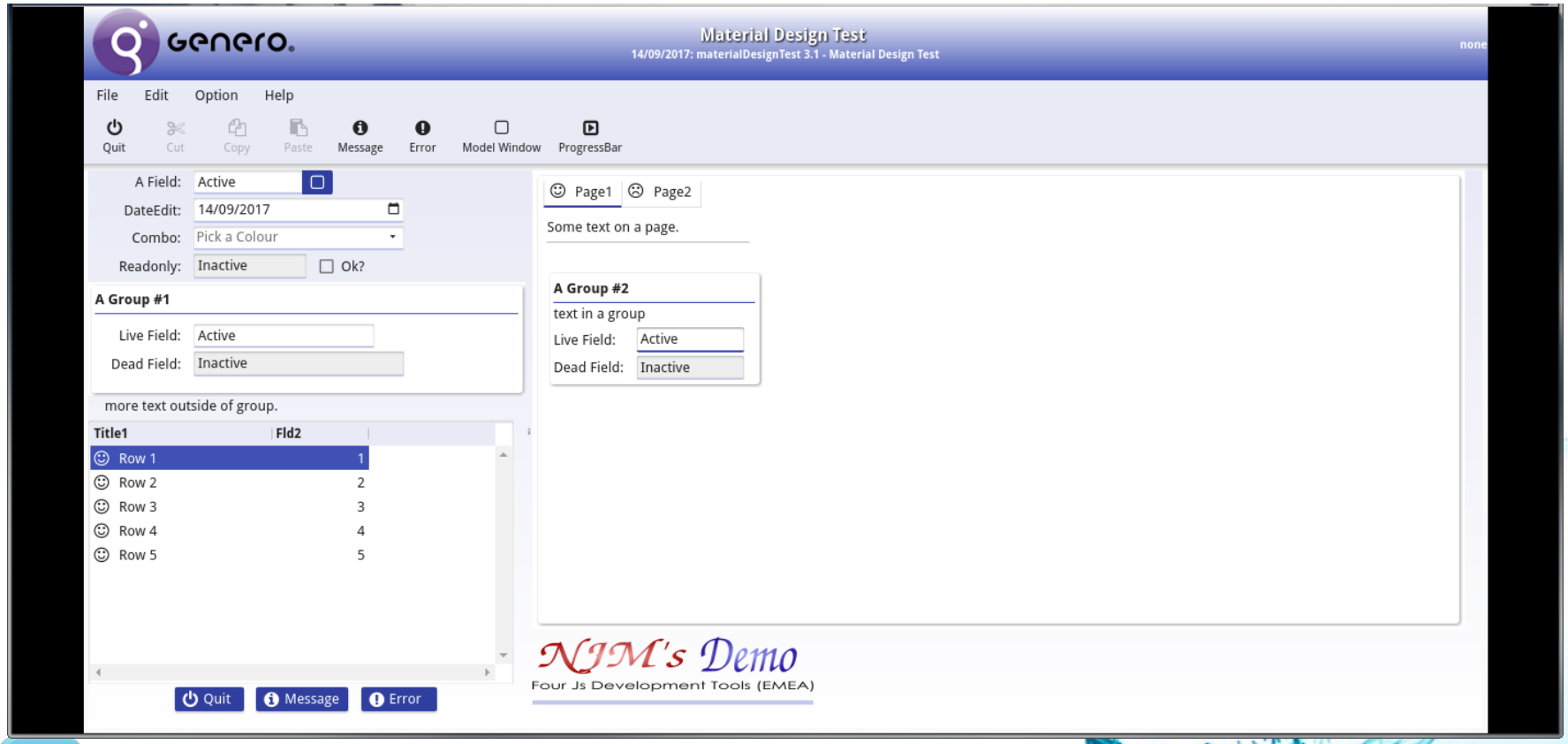
## sass/MyHeadingBarWidget.scss

```
.gbc_MyHeaderBarWidget {
  background: linear-gradient($gbc-field-background-color, $gbc-primary-medium-color);
  color: $gbc-secondary-text-color;

  h1 {
    text-align: center;
    font-size: 1.2em;
    margin: 0px 5px 0px 0px;
    text-shadow: 1px 1px 2px black;
  }

  b {
    font-size: 0.8em;
  }
}

.MyHeaderBarWidget-title {
  text-align: center;
}

.gbc_MyMainContainerWidget > header {
  text-align: center;
}
```

# More Customizations

Closer to our goal

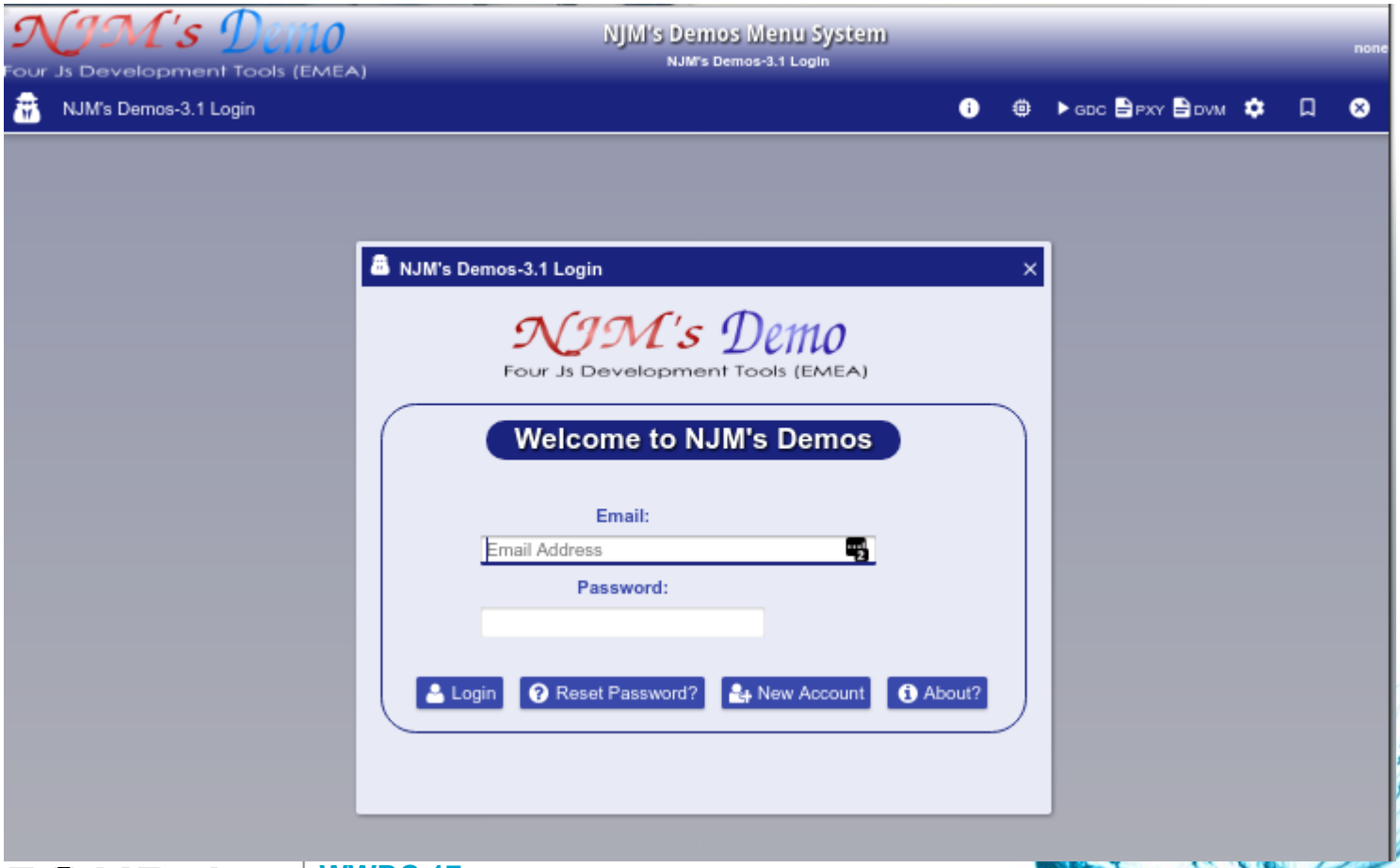
**Here there are few more subtle changed to the styles to limit the size, change group titles, remove some unwanted spaces**



# More Customizations

Multiple Themes: Blues

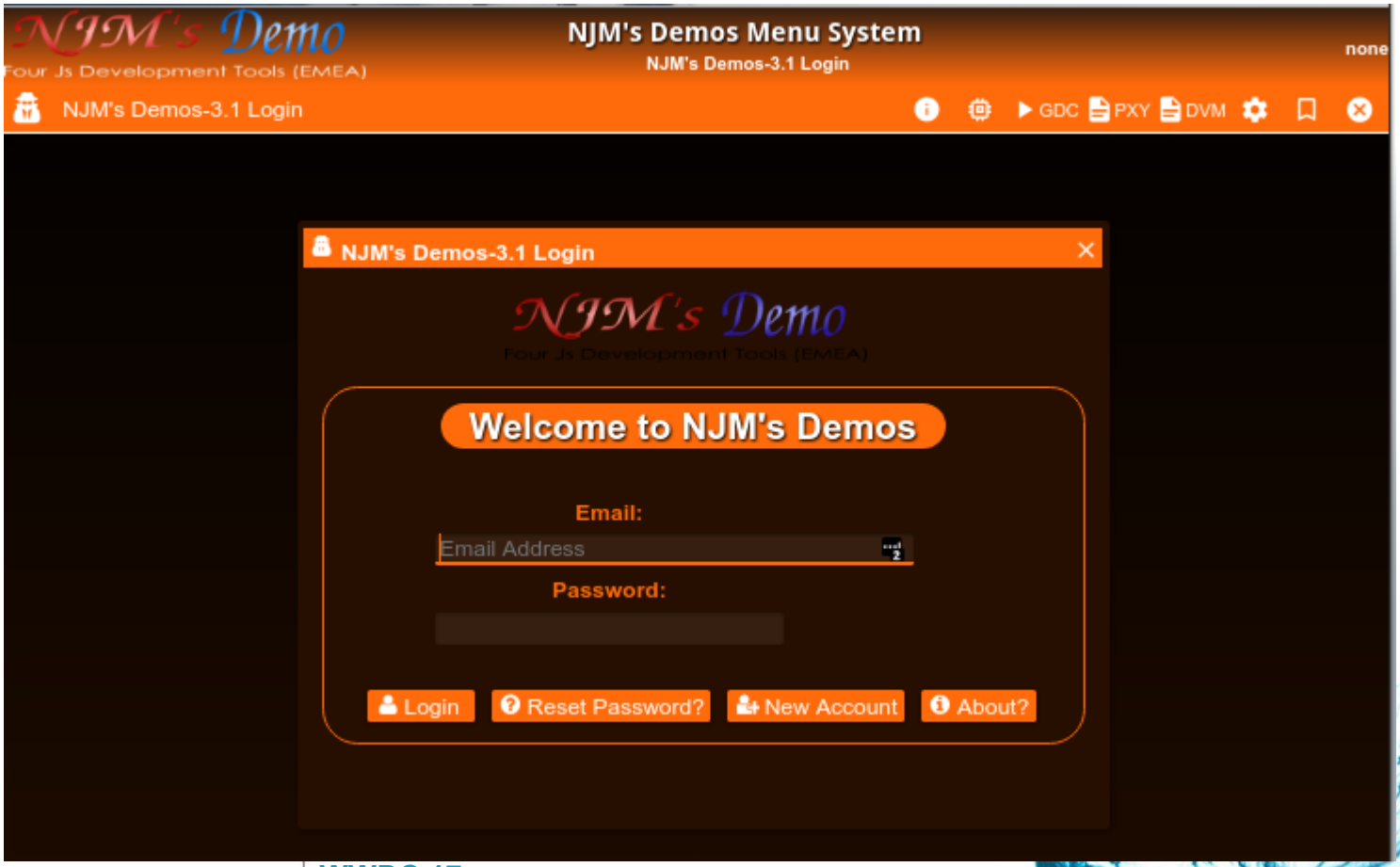
**With basic customizations you can achieve a variety of looks.**



# More Customizations

Multiple Themes: Elite

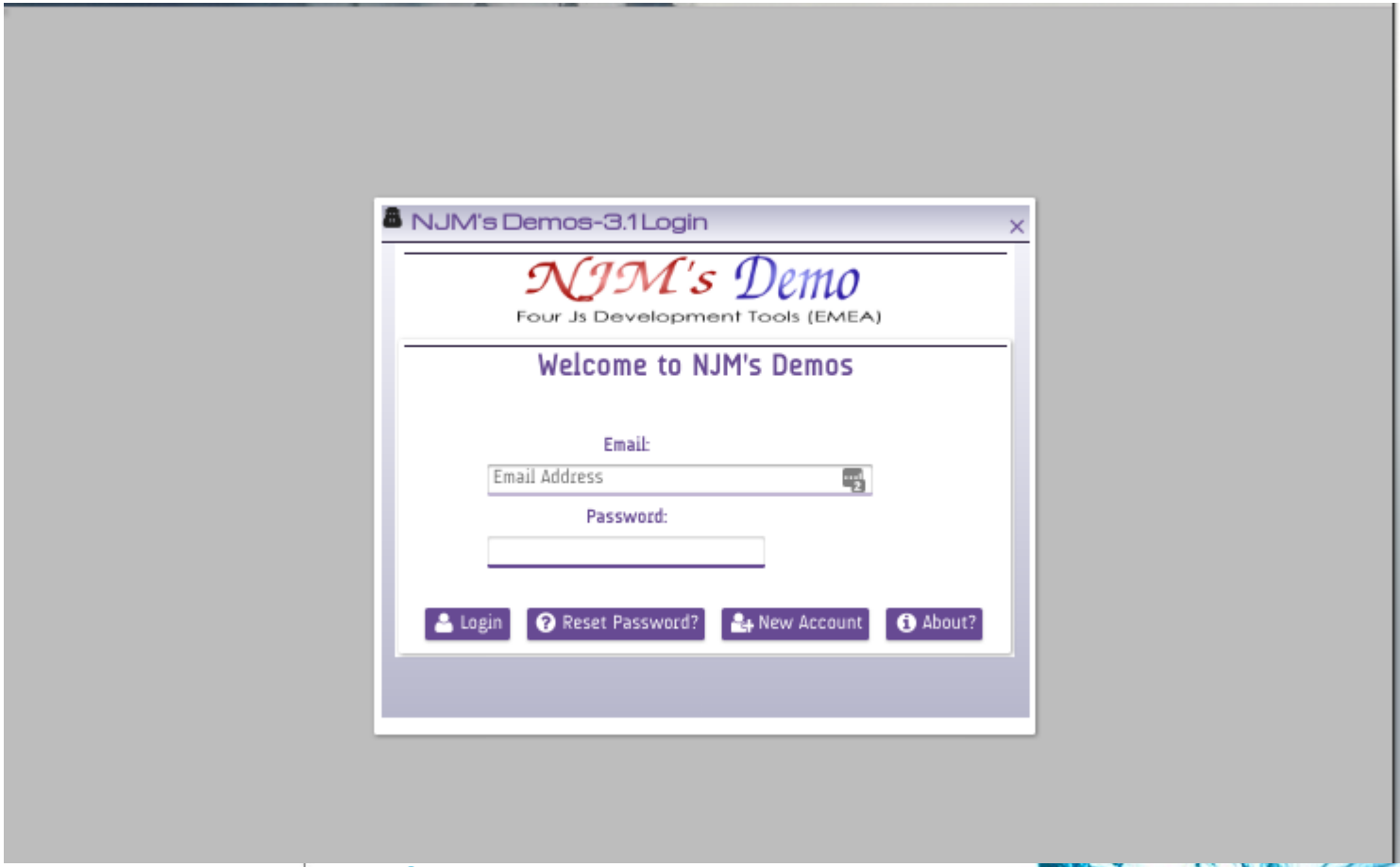
## Only the 'theme' file is different from the previous slide.



# More Customizations

Multiple Themes: Simply Purple

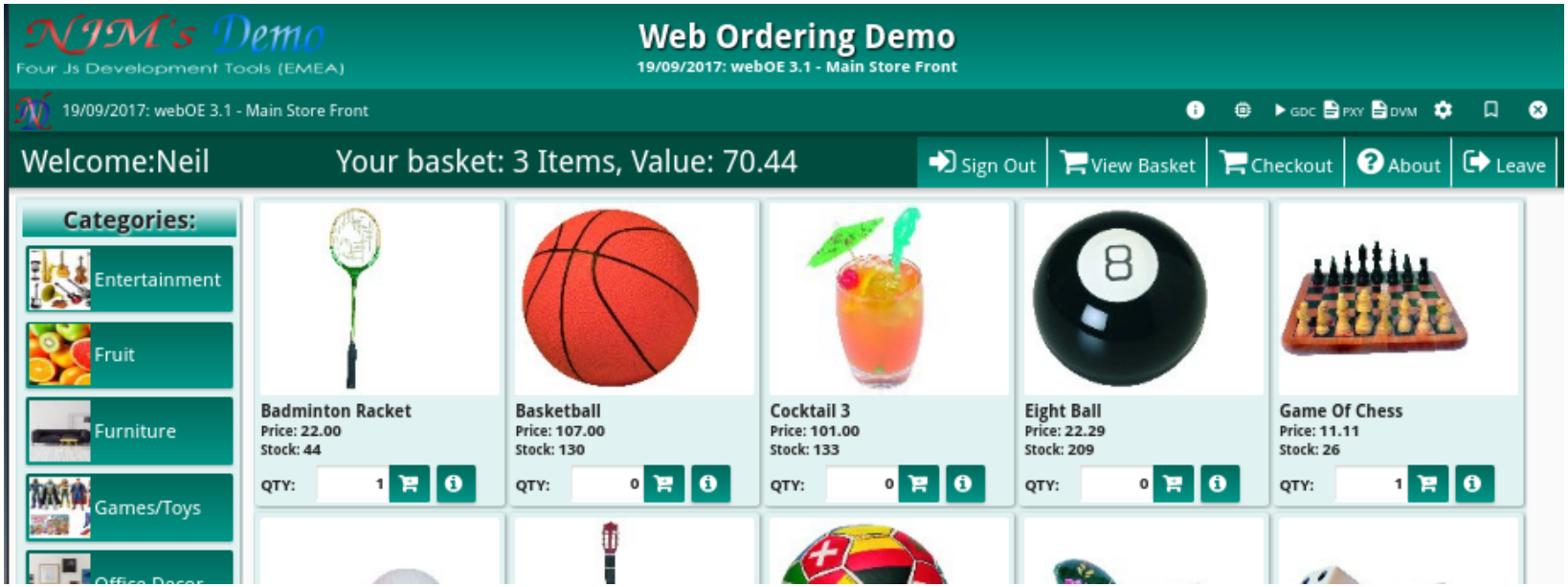
## A very simple plain version.



# More Advanced Customizations

Custom Toolbar

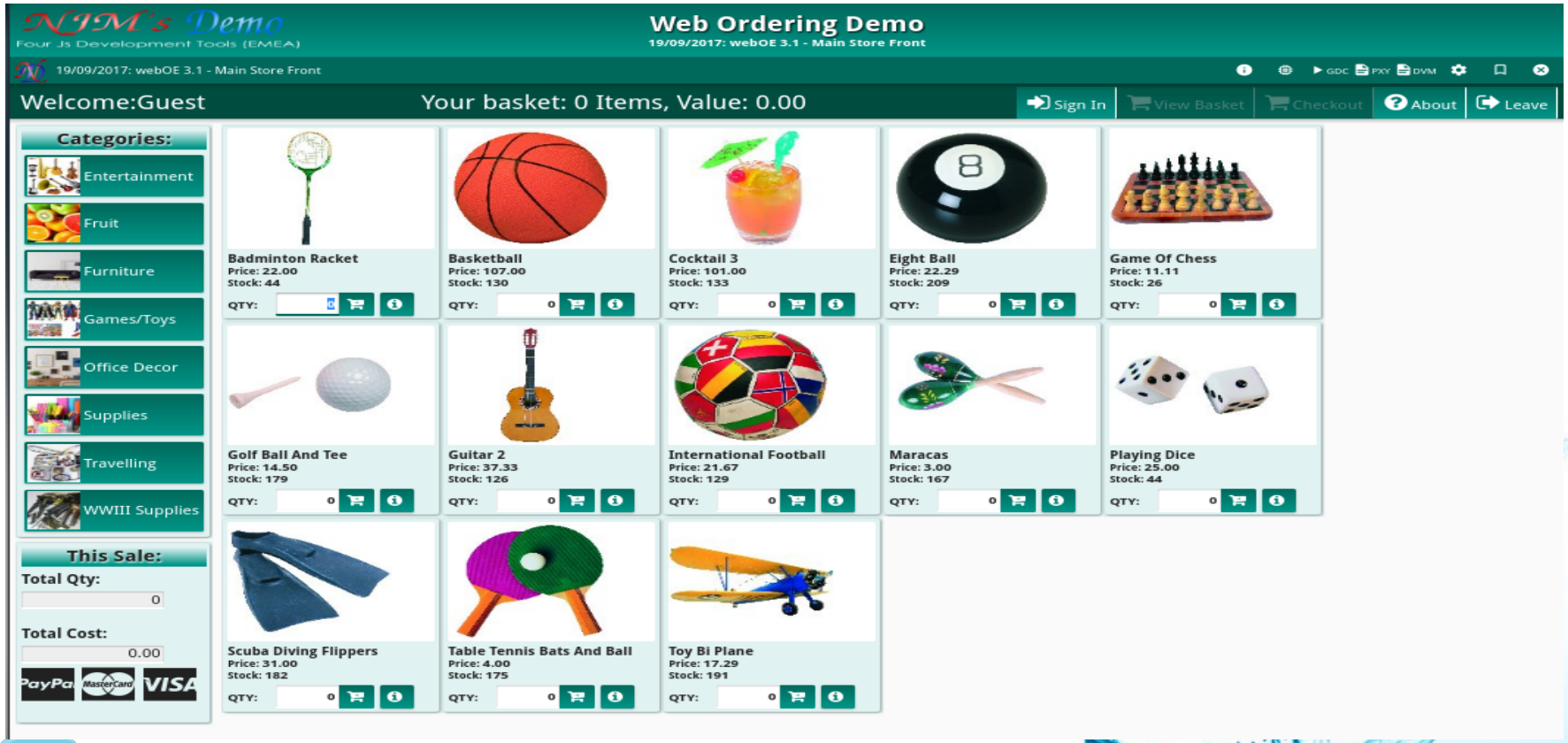
**This screenshot shows a custom 'ToolBar' that included a some fields from the form.**



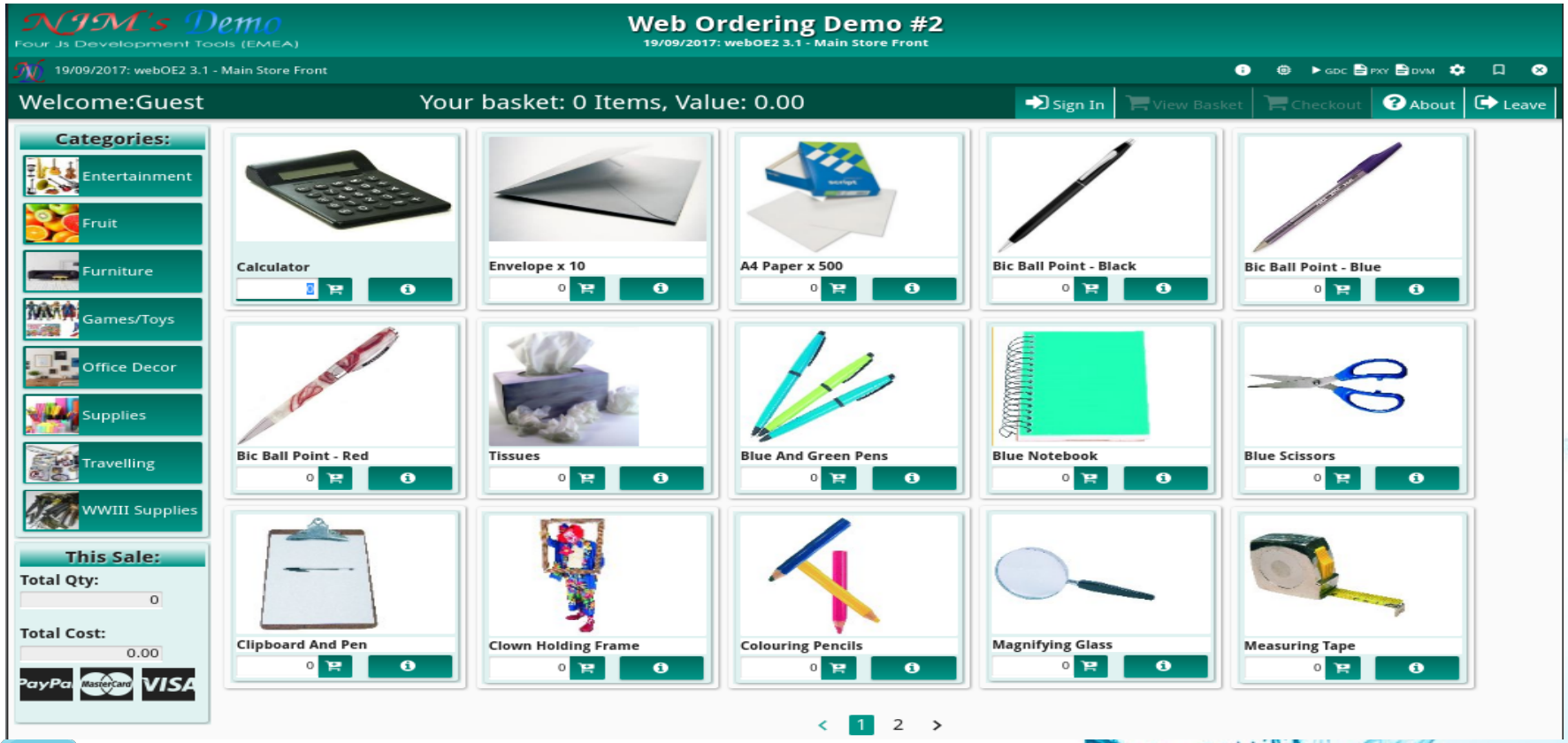
# More Advanced Customizations

Paging products

**There are couple ways to do a responsive design for listing 'items', this example is using JavaScript and a dynamc form layout.**



# More Advanced Customizations

Paging products

## This version is using a 'simple' ScrollGrid