Hello World

SUGGEST EDITS

This HelloWorld example serves two purposes

- it introduces the basics of developing with SCADE
- it provides a quick way of checking if the installation was successful and all settings are configured correctly

We will develop a HelloWorld example in three iterations

1. SuperSimple Version displaying HelloWorld and compiling to iOS/Android
2. Extended Version that captures an OnClick event and writes a HelloWorld message out to the console
3. Extended Version demoing model view binding that binds a variable to a control property

Learning outcomes

You will learn about

- project structure
- compile and running apps in Android / iOS
- event handing
- basic binding
- fixing common issues

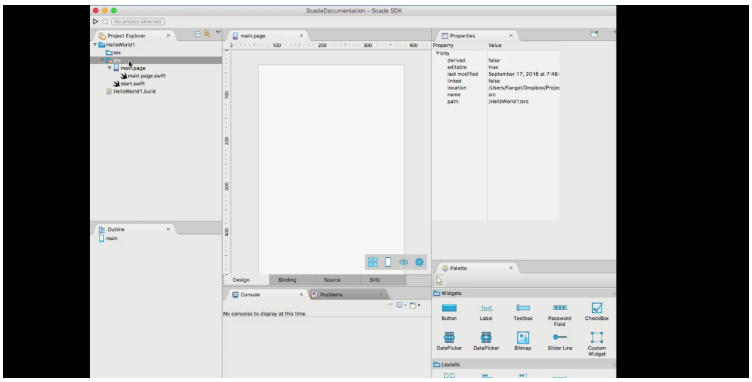
Create project

1. In the IDE, choose `File | New | SCADE Project`
2. Enter the name of the project, `HelloWorld1`
3. Expand the HelloWorld1 folder. You will see the following structure
    - **products** folder that contains the binaries (Android, iOS binaries once you ran your first compilation)
    - **src** folder that contains all the source code.
        - One artifact are pages, which contain UI and source code and are marked with a **.page** file extension
    - **res** folder that contains all the resources
    - **start.swift** contains the code executed upon startup of the application
    - **<ProjectName>. build** contains the configuration of the project

    The directory hierachy starts after HelloWorld1, i.e. referencing files in code is done using /res/image1.bnp for example

Create your first Hello World page

1. Open the main.page file by doubleclicking it.
2. Drag a label into the page, set Hello World as the text
3. Center it in the middle of the screen
4. Don't forget to press **Control + s** to save the file



00:00 -00:23

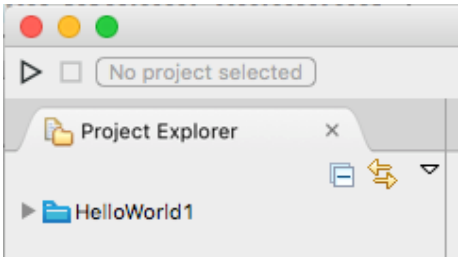
Run the project

This minimum project can be started and run. The running options are

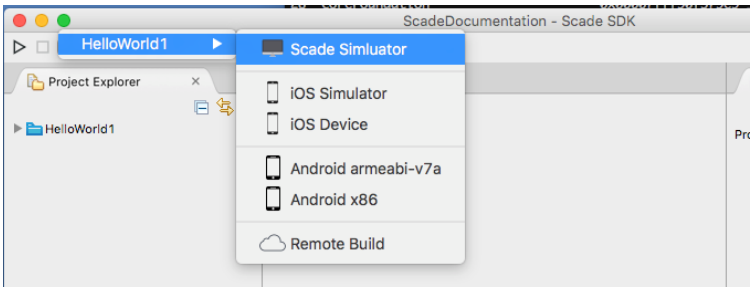
- the powerful, superfast SCADE OSX runtime (aka **SCADE Simulator**)
- the original Android Simulator
- the original iOS Simulator
- as an Android app
- as an iOS App

Let's start with the SCADE simulator. Its very productive and easy to use for fast development purposes.

1. Click on the large rectangle button, the project launch selector button

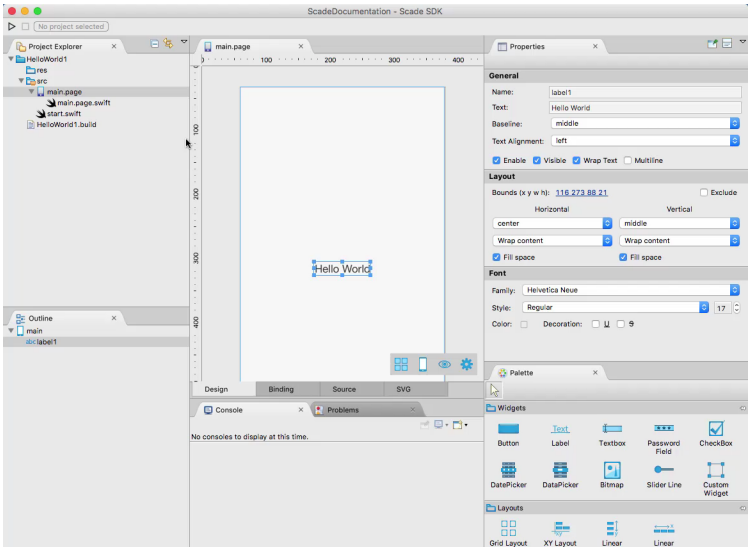


2. Select the Scade Simulator as the runtime for launching



3. Press the **Play** button to run the app

4. The app should launch successfully with a HelloWorld page being displayed.

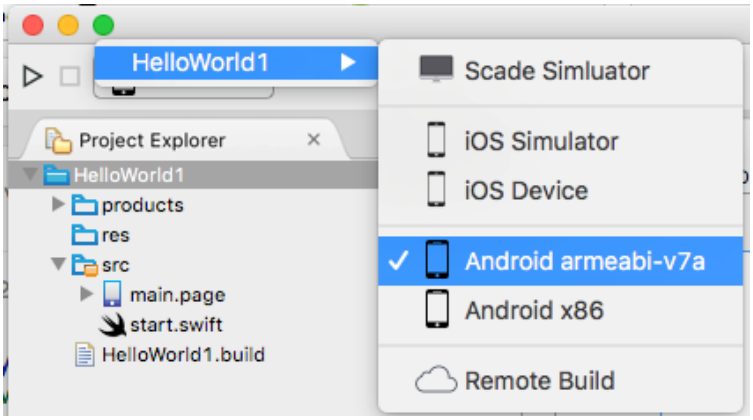


00:00 -00:13

Compile to Android and iOS

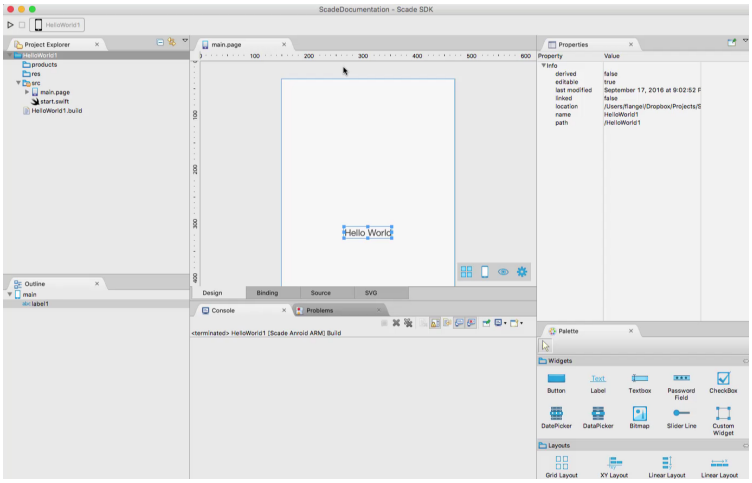
So know that we have a minimal app, lets compile them to iOS and Android

1. Select Android ARM7 as the target



2. Press the **Play** button to compile the app to Android

   - The first time you compile, it takes > 60 seconds to finish compilation
   - After the first time, compilation speed is much faster as only the changed binaries are recreated



00:00 -00:21

Adding an OnClick event to app

> **Changes**
>
> The video might not reflect the latest versions. Here are two things that changed
>
> - you need to write **self.page!** instead of

> **self.page.**

As a next step, lets printout "Hello World" when we click on a button. A very simple enhancement.

1. Let's add a button. Click and drag the button on the bottom of the screen and drop it there
2. Center it by using **Layout Horizontal Center** and **Layout Horizontal Fill space**
3. Now, lets add an event
   3.1 Go to the underlying source code located in the **main.page.swift** file
   3.2 Within the load method, create a local variable button1 that holds the reference to the button

Swift

```
let button1 = self.page!.getWidgetByName("button1") 
```

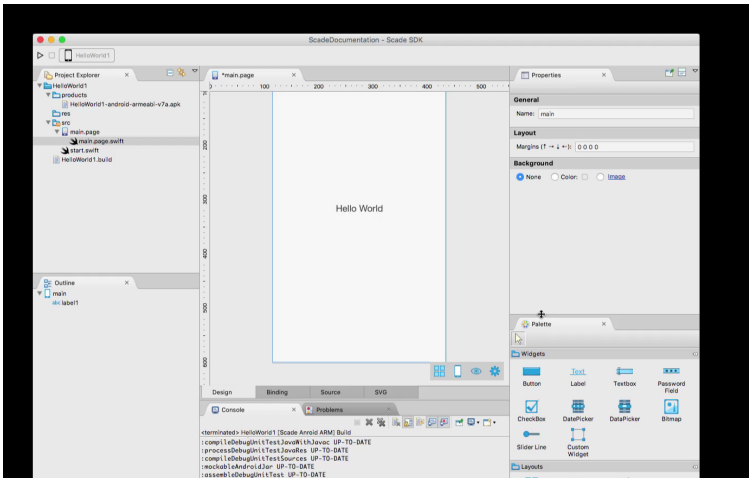
- **self.page!** is the (self) reference to the current page you just modified in the page editor
- **getWidgetByName** is a method that returns the reference to a widget by name.
- Then, we cast it to the respective class for strongly typed use of variables

Now that we have a reference to the button, lets add the event handler

Swift

```
button1.onClick.append(SCDWidgetsEventHandler{ _ in 
```

- Append to the **OnClick** array to add event handlers
- The **SCDWidgetsEventHandler** is our event handler for all general click events. It takes a closure as input, where we execute the print statement



00:00 -01:16

Using model binding to set label

We support model view binding capabilities that allow you to link your model to your view control. In this step, we provide a simple example by introducing a variable and binding the value of the variable to the label control

1. Add a variable labelText in the main class

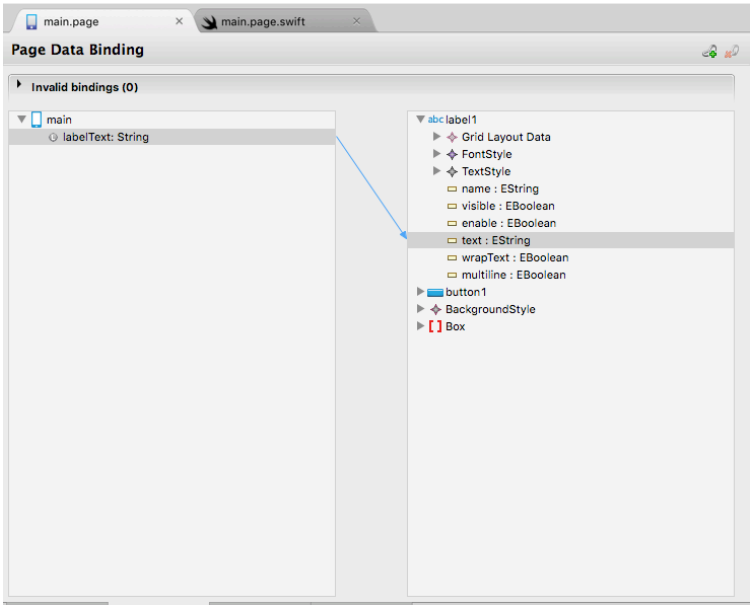
Swift

```
// use of the dynamic keyword is mandatory
dynamic var labelText : String = "Hello World"
```

2. Bind the variable to the labels

   2.1 go to the binding tab of the page editor

   2.2 link labelText to the label called label1

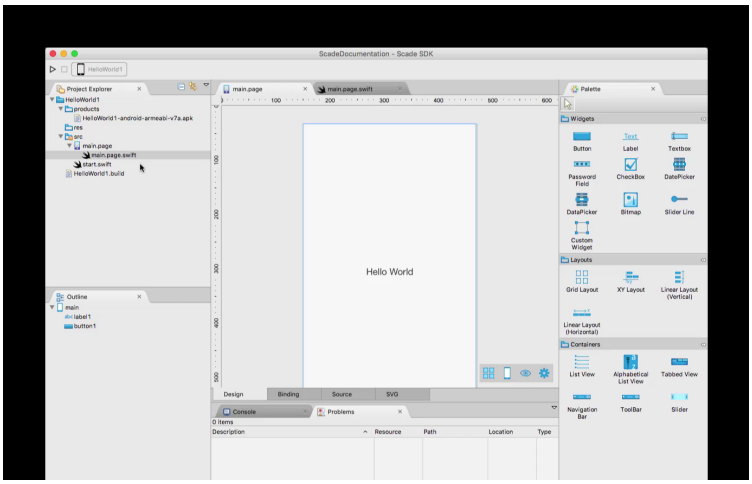


3. Change the value of the variable in the closure

- lets move the logic into a function sayHello() (for clarity purposes)
- set the variable within that function

Swift

```
import ScadeKit

   class MainPageAdapter: SCDLatticePageAdapter {

        // very important. Always use dynamic keywor
        dynamic var labelText : String = "Hello Worl

        // page adapter initialization
        override func load(path: String) {
                super.load(path)

                // Typecast to SCDWidgetsButton
                let button1 = self.page!.getWidgetByI
                button1.OnClick.append(SCDWidgetsEve
        }

        func sayhello() {
                print("Hello World")
                self.labelText = "Hello SCADE develo
        }
   }
```



00:00 -01:21

Common gotchas and troubleshooting

1. **You forgot to save** the code/page and the changes did not show up. Use Control + s to save. Autosaving mode also ask you if you want to save if you didn't.

2. **Binding doesnt work**. You forgot to use **dynamic** keyword for variables you want to bind to.

3. **Binding doesnt work because you changed variables name or type** and the binding is not valid anymore. Delete the invalid bindings in the binding tab. Invalid bindings are shown in the panel.

Next reads

- More information to handling pages Pages
- More information on layouting Layout Manager
- Try a larger app eReader App