Creating Mobile Apps with Go

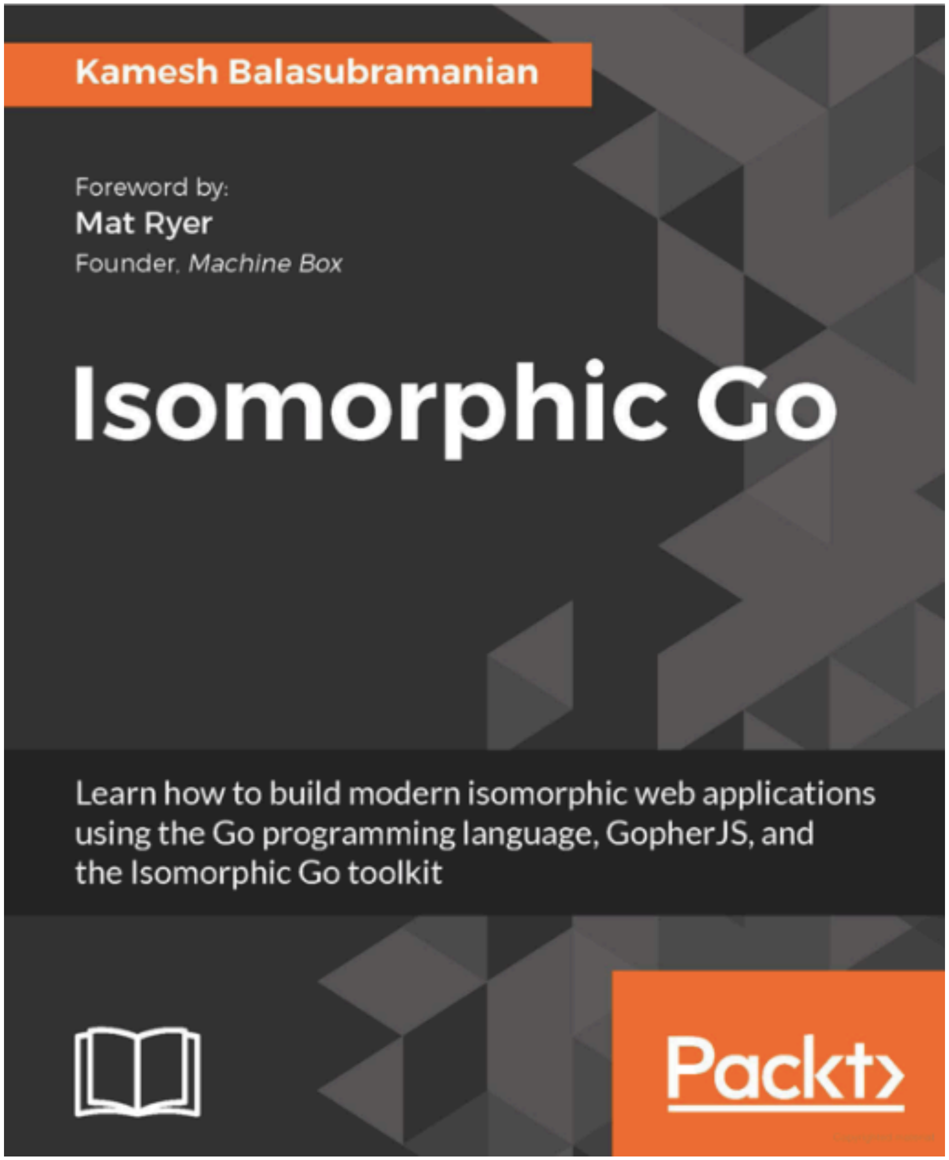
for iOS and Android

Kamesh Balasubramanian
Founder/CEO, Wirecog, LLC

About Wirecog

- We're a Go shop
- We created the Isomorphic Go project
- We have experience providing Go consulting for 3 Fortune 100 companies

Isomorphic Go


Kamesh Balasubramanian

Foreword by:
Mat Ryer
Founder, *Machine Box*

Isomorphic Go

Learn how to build modern isomorphic web applications using the Go programming language, GopherJS, and the Isomorphic Go toolkit

Packt>

Introduction

Why use Go for mobile development?

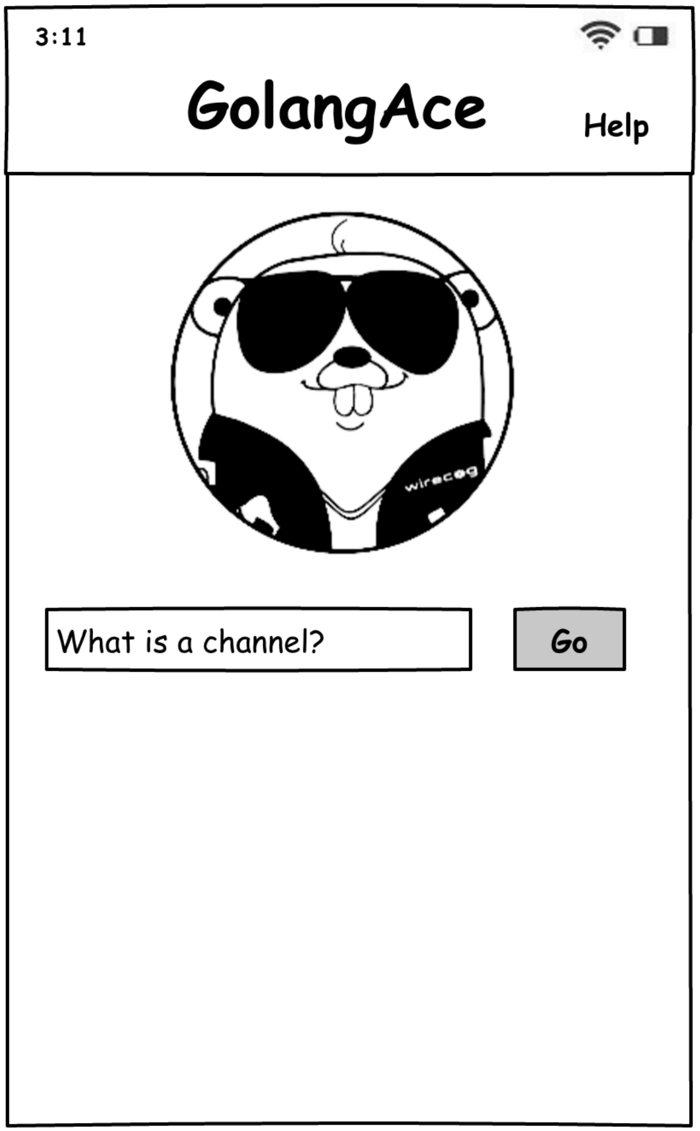
- Robust standard library (networking, string manipulation, math)
- Goroutines
- Compiles to machine code / Transpiles to JavaScript
- Can be used for UI programming!
- Cross-platform (reuse code)

Design

The GolangAce App

- A virtual assistant to answer common Go questions
- Provides trivia, standard package info, code examples
- 85 to 90% of the code is cross platform
- It uses Go for both business logic and UI
- It feels like a native app

Question Screen



What is a channel?

Setup

Ingredients

- Go based REST API
- gomobile (https://godoc.org/golang.org/x/mobile/cmd/gomobile) (business logic)
- gopherjs (https://github.com/gopherjs/gopherjs) (UI)
- Xcode (Mac) / Android Studio (Linux)
- Device(s) to test with

gomobile

1. Get gomobile

```
go get -u golang.org/x/mobile/cmd/gomobile
```

2. Initialize it

```
gomobile init
```

See gomobile documentation (https://godoc.org/golang.org/x/mobile/cmd/gomobile)

gopherjs

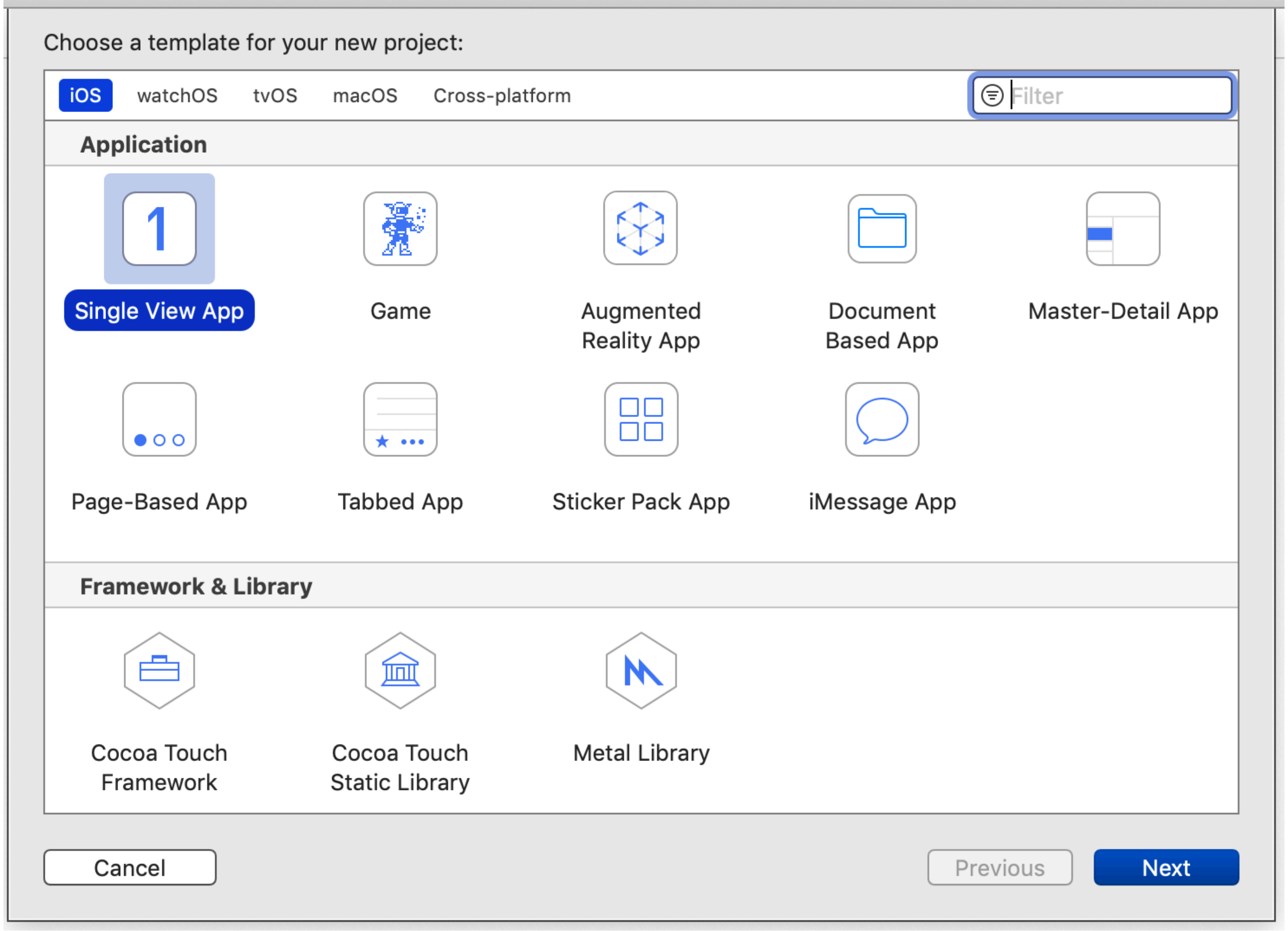
1. Get gopherjs

```
go get -u github.com/gopherjs/gopherjs
```

See gopherjs getting started guide (https://github.com/gopherjs/gopherjs#getting-started)

XCode (for iOS Apps)

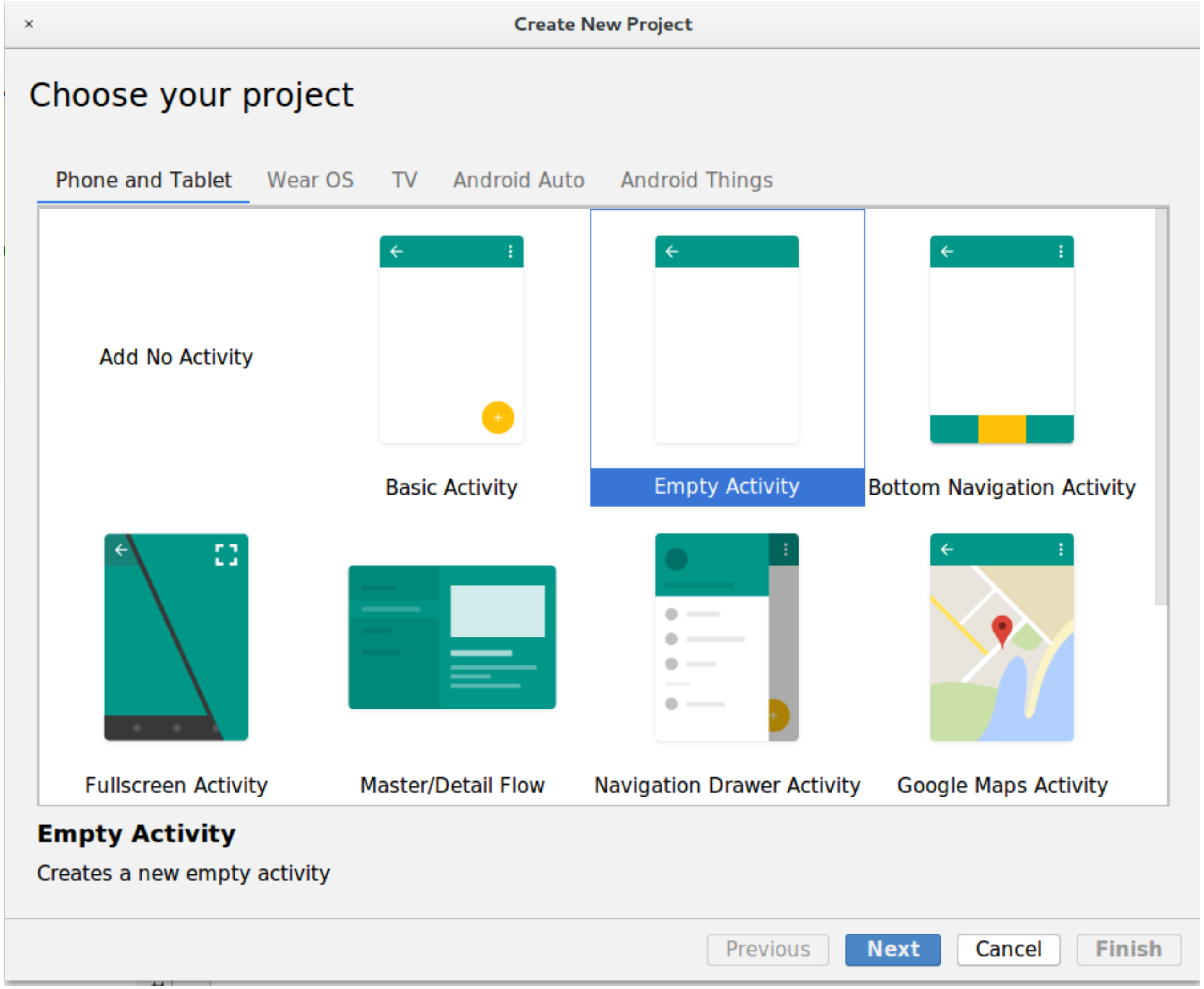
Create a new "Single View App" template for iOS and choose Objective-C



Choose a template for your new project:

Android Studio (for Android apps)

Create a new "Empty Activity" template and choose Java



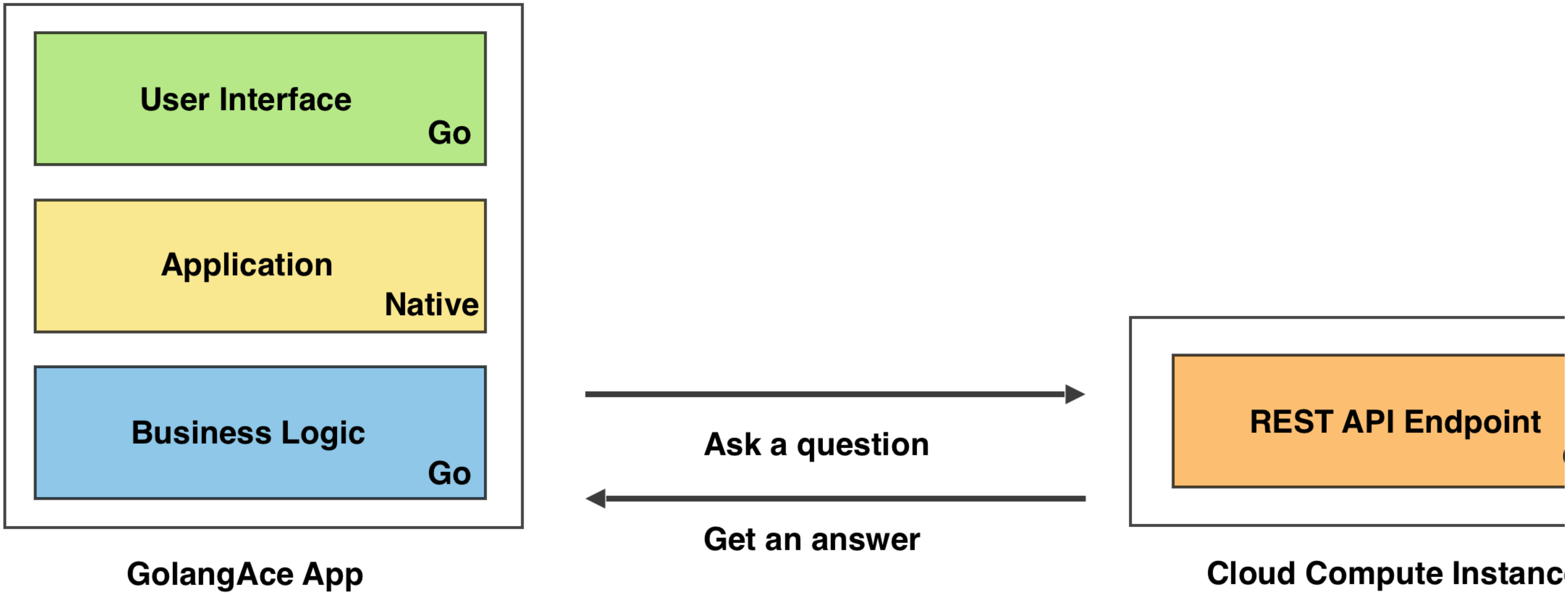
Create New Project

Choose your project

Empty Activity

Creates a new empty activity

Architecture



User Interface

Go

Application

Native

Business Logic

Go

GolangAce App

Ask a question

Get an answer

REST API Endpoint

Cloud Compute Instanc

Building Mobile-Friendly Go Code

Business Logic - iOS Framework / Android Archive

iOS (Ace.framework)

```
gomobile bind -o Ace.framework -target ios .
```

Android (ace.aar)

```
gomobile bind -o ace.aar -target android .
```

See gobind documentation (https://godoc.org/golang.org/x/mobile/cmd/gobind) and type restrictions (https://godoc.org/golang.org/x/mobile/cmd/gobind#hdr-Type_restrictions)

UI - Go transpiled to JavaScript

Cross-Platform (ace.min.js)

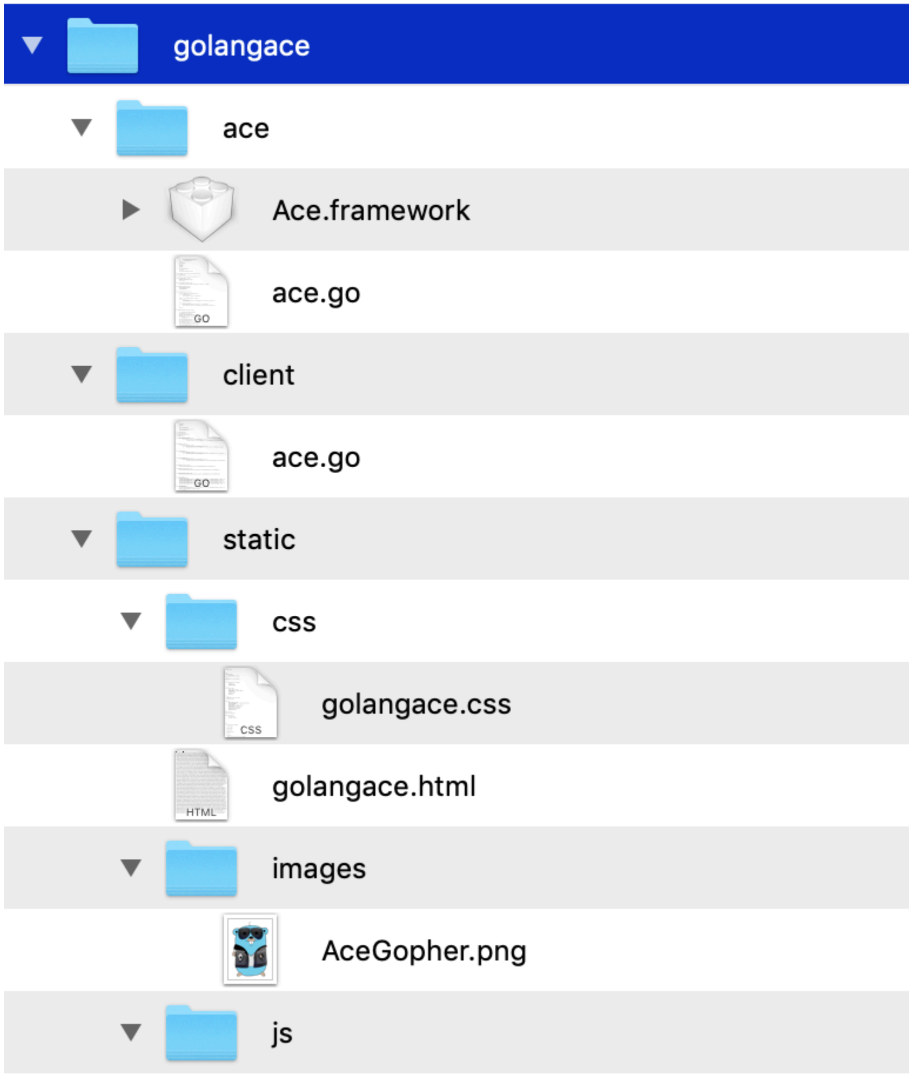
```
gopherjs build -m -o $GOPATH/src/golangace/static/js/ace.min.js
```

See gopherjs usage guide (https://github.com/gopherjs/gopherjs#installation-and-usage)

Code Organization

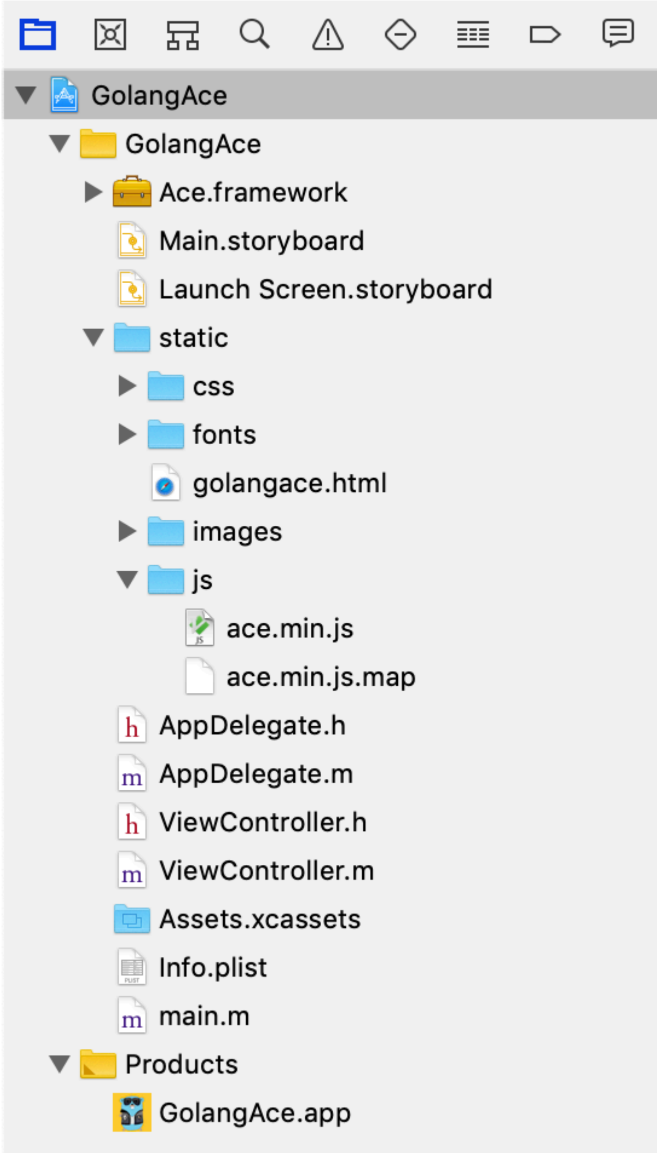
GolangAce Go Project

The `golangace` folder is inside `$GOPATH/src`



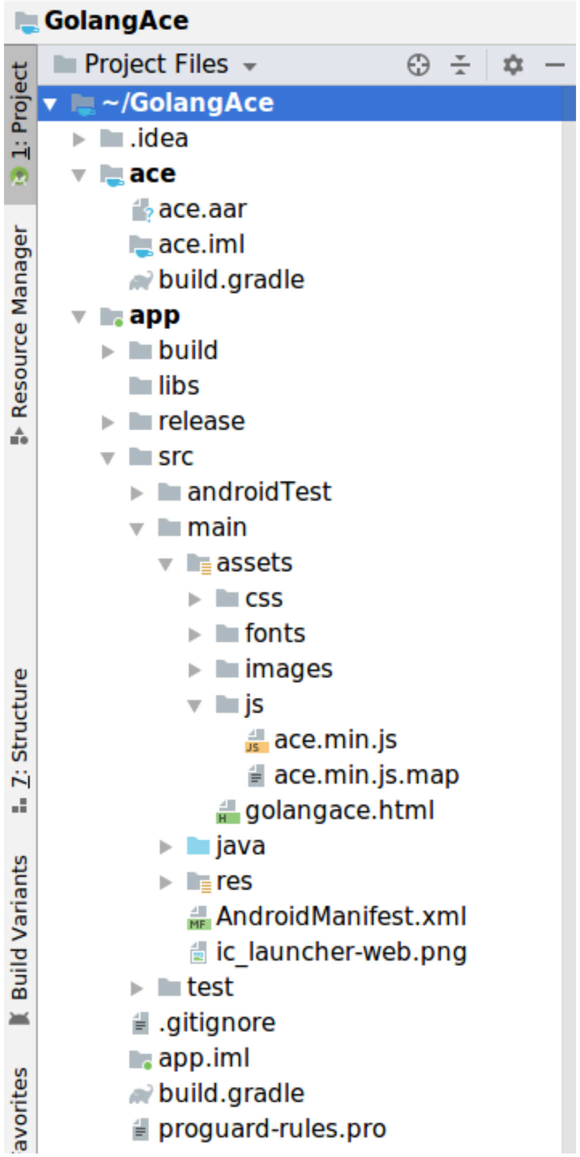
```
golangace
    ace
        Ace.framework
        ace.go
    client
        ace.go
    static
        css
            golangace.css
        golangace.html
        images
            AceGopher.png
        js
```

Xcode



```
▼ GolangAce
  ▼ GolangAce
    ▶ Ace.framework
      Main.storyboard
      Launch Screen.storyboard
    ▼ static
      ▶ css
      ▶ fonts
        golangace.html
      ▶ images
      ▼ js
          ace.min.js
          ace.min.js.map
      AppDelegate.h
      AppDelegate.m
      ViewController.h
      ViewController.m
      Assets.xcassets
      Info.plist
      main.m
  ▼ Products
      GolangAce.app
```

Android Studio



```
▼ ~/GolangAce
  ▶ .idea
  ▼ ace
      ace.aar
      ace.iml
      build.gradle
  ▼ app
    ▶ build
      libs
    ▶ release
    ▼ src
      ▶ androidTest
      ▼ main
        ▼ assets
          ▶ css
          ▶ fonts
          ▶ images
          ▼ js
              ace.min.js
              ace.min.js.map
            golangace.html
        ▶ java
        ▶ res
          AndroidManifest.xml
          ic_launcher-web.png
      ▶ test
      .gitignore
      app.iml
      build.gradle
      proguard-rules.pro
```

Setting Up the Web View

iOS: Objective-C

Prepping the View Controller

Implement the `WKScriptMessageHandler` and `WKNavigationDelegate` interfaces

```
@interface ViewController () <WKScriptMessageHandler, WKNavigationDelegate>
```

Define properties for the `WKWebView` and the `WKWebViewConfiguration`

```
@property (nonatomic,strong) WKWebView* webView;
@property (nonatomic, strong) WKWebViewConfiguration * webConfig;
```

See docs for WKScriptMessageHandler (https://developer.apple.com/documentation/webkit/wkscriptmessagehandler), WKNavigationDelegate (https://developer.apple.com/documentation/webkit/wknavigationdelegate), WKWebView (https://developer.apple.com/documentation/webkit/wkwebview), WKWebViewConfiguration (https://developer.apple.com/documentation/webkit/wkwebviewconfiguration)

Initialize the Web View Configuration

Creates a new `WKWebViewConfiguration` object with a `userController` that can handle the `askQuestion` event notification

```
-(WKWebViewConfiguration*) webConfig {
    if (!_webConfig) {
        _webConfig = [[WKWebViewConfiguration alloc]init];
        WKUserContentController* userController = [[WKUserContentController alloc]init
        [userController addScriptMessageHandler:self name:@"askQuestion"];
        _webConfig.userContentController = userController;
        }
        return _webConfig;
}
```

See docs for WKUserContentController (https://developer.apple.com/documentation/webkit/wkusercontentcontroller)

Initialize the Web View

The `webView` gets initialized in the `viewDidAppear` method

```
-(void)viewDidAppear:(BOOL)animated {
    [super viewDidAppear:animated];

    self.webView = [[WKWebView alloc]initWithFrame:self.view.frame configuration:self.webConfi
    self.webView.navigationDelegate = self;
    self.webView.frame = CGRectMake(self.view.frame.origin.x,self.view.frame.origin.y,
            self.view.frame.size.width, self.view.frame.size.height);
    [self.navigationController setNavigationBarHidden:YES];
    [self loadHtml];
}
```

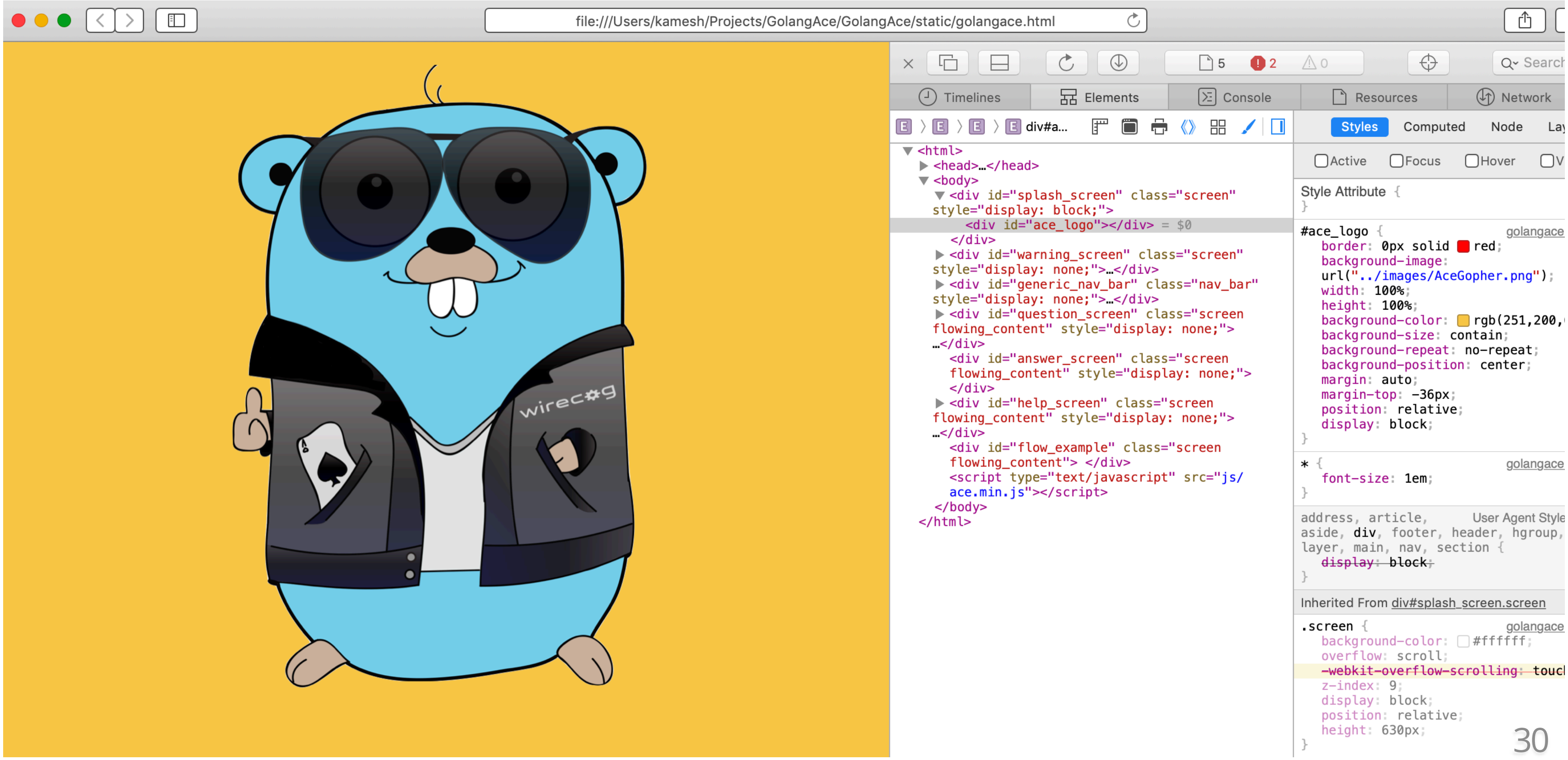
Load and Render the UI

The webView object loads and renders the `golangace.html` file.

```
-(void)loadHtml {

    NSString* htmlPath = [[NSBundle bundleForClass:[self class]] pathForResource:@"/static/gol
    if (htmlPath) {
        [self.webView loadRequest:[NSURLRequest requestWithURL:[NSURL fileURLWithPath:htmlPath
    }else {
        [self showAlertWithMessage:@"Could not load HTML file!"];
    }
}
```

The Rendered UI



```
<html>
  <head>…</head>
  <body>
    <div id="splash_screen" class="screen"
  style="display: block;">
        <div id="ace_logo"></div> = $0
      </div>
    <div id="warning_screen" class="screen"
  style="display: none;">…</div>
    <div id="generic_nav_bar" class="nav_bar"
  style="display: none;">…</div>
    <div id="question_screen" class="screen
  flowing_content" style="display: none;">
  …</div>
      <div id="answer_screen" class="screen
      flowing_content" style="display: none;">
      </div>
    <div id="help_screen" class="screen
  flowing_content" style="display: none;">
  …</div>
      <div id="flow_example" class="screen
      flowing_content"> </div>
      <script type="text/javascript" src="js/
      ace.min.js"></script>
    </body>
  </html>
```

```
Style Attribute {
}

#ace_logo {                               golangace
   border: 0px solid 🟥 red;
   background-image:
   url("../images/AceGopher.png");
   width: 100%;
   height: 100%;
   background-color: 🟨 rgb(251,200,
   background-size: contain;
   background-repeat: no-repeat;
   background-position: center;
   margin: auto;
   margin-top: -36px;
   position: relative;
   display: block;
}

* {                                       golangace
   font-size: 1em;
}

address, article,            User Agent Style
aside, div, footer, header, hgroup,
layer, main, nav, section {
   ~~display: block;~~
}
```

Inherited From div#splash_screen.screen

```
.screen {                                 golangace
   background-color: □ #ffffff;
   overflow: scroll;
   ~~-webkit-overflow-scrolling: touc~~
   z-index: 9;
   display: block;
   position: relative;
   height: 630px;
}
```

Android: Java

Create a new Web View

```
public class MainActivity extends AppCompatActivity  implements OnInitListener {

    private WebView webView;

        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            webView = (WebView) findViewById(R.id.activity_main_webview);
            webView.setWebViewClient(new WebViewClient());
            WebSettings webSettings = webView.getSettings();
            webSettings.setJavaScriptEnabled(true);
            webView.loadUrl("file:///android_asset/golangace.html");
            webView.addJavascriptInterface(this, "Android");
        }

    }
```

See docs for WebView (https://developer.android.com/reference/android/webkit/WebView)

From Question to Answer

User submits a question (UI layer)

Display the loader, ask the question, and clear the question field.

```
func handleSearchButtonClickEvent(event dom.Event) {
    questionField := document.GetElementByID("question_field").(*dom.HTMLInputElement)
    question := questionField.Value

    displayLoader()

    timer := time.NewTimer(108 * time.Millisecond)
    go func() {
        <-timer.C
        askQuestion(question)
        questionField.Value = ""
    }
}
```

Send question over to the native layer

Send the askQuestion event notification from the UI layer to the userContentController (the native layer)

Dispatch the question using the `postMessage` method of the `window.webkit.messageHandlers.askQuestion` object.

```
func askQuestion(message string) {
    if js.Global.Get("webkit") != nil {
        if js.Global.Get("webkit").Get("messageHandlers") != nil {
            js.Global.Get("webkit").Get("messageHandlers").Get("askQuestion").Call("postMessag
                message)
        }
    }
}
```

iOS: Send the question and get the answer

```
- (void)userContentController:(WKUserContentController *)userContentController
     didReceiveScriptMessage:(WKScriptMessage *)message {
    if ([message.name isEqualToString:@"askQuestion"]) {
        id messageBody = message.body;
        if ([messageBody isKindOfClass:[NSString class]]) {
            NSString* argFromFunc = messageBody;
            NSString* answer = AceAskQuestion(argFromFunc);
            NSString* callJS = [NSString stringWithFormat:@"displayAnswer('%s')",
                [answer UTF8String]];
            [self.webView evaluateJavaScript:callJS completionHandler:nil];
        }
    }
}
```

Note: Once the answer is retrieved, the `displayAnswer` func will be called in the UI layer

Android: Send the question and get the answer

```
@JavascriptInterface
public void askQuestion(String question) {
    try {
        theQuestion = question;
        theAnswer = Ace.askQuestion(theQuestion);
        webView.post(
                new Runnable() {
                    @Override
                    public void run() {
                        webView.loadUrl("javascript:displayAnswer('" + theAnswer + "')");
                    }
                });
    }
    catch (Exception e) {}
}
```

Call the REST Endpoint

```
func AskQuestion(question string) string {
    q = optimizeQuestion(question)

    var data bytes.Buffer
    var r io.Reader = &data
    enc := gob.NewEncoder(&data)
    enc.Encode(q)
    req, err := http.NewRequest("POST", serviceURL+"/askOptimizedQuestion", r)
    req.Header.Set("Content-Type", "application/octet-stream")
    if err != nil {
        log.Println("Failed to set up request with the following error:", err)
        return ""
    }
    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil {
        return ""
    }
    defer resp.Body.Close()
    body, _ := ioutil.ReadAll(resp.Body)
    return string(body)
}
```

Display the Answer

```
func displayAnswer(answer string) {
    hideLoader()
    if answer == "" {
        checkIfServiceIsReachable()
        return
    }
    displayAnswerScreen(answer)
}
```

Demo

- GolangAce on the App Store (https://apps.apple.com/us/app/golangace/id1444825728)
- GolangAce on the Play Store (https://play.google.com/store/apps/details?id=com.wirecog.golangace&hl=en_US)
- GolangAce.com (https://golangace.com)

Preparing for the App Review

Tips to Get Through the Review Process

- App Store Review Guidelines (https://developer.apple.com/app-store/review/guidelines/)
- Test on actual devices
- Testing complexity increases with additional capabilities

Thank you

Kamesh Balasubramanian
Founder/CEO, Wirecog, LLC
kamesh@wirecog.com (mailto:kamesh@wirecog.com)
https://www.wirecog.com (https://www.wirecog.com)
@wirecog (http://twitter.com/wirecog)