1 WkWebKit JavaScript

WkWebKit ersetzt den ursprünglichen UIWebView und wurde in unserer App nachträglich hinzugefügt, da war anfangs einen UIWebView verwendet haben. Dieser bietet aber weniger funktionen und wird somit von Apple nicht mehr empfohlen. Die Einbindung des WkWebView ist zum derzeitigen Stand jedoch noch nicht über den Interface Builder möglich und muss somit programatisch erstellt werden. Erzeugen der verwendeten Variable myWebView als WKWebView:

```
var myWebView: WKWebView?
```

Zuweisung des WebViewDelegates ind er Funktion viewDidLoad():

```
        myWebViewDelegate = WebViewDelegate()
        myWebViewDelegate.viewCtrl = self
```

Um JavaScript in die WkWebView einzubinden muss eine WkWebViewConfiguration erstellt werden:

```
        let config = WKWebViewConfiguration()
```

Als nächstes muss der Pfad der JavaScript Datei angegeben werden mit dieser wird der ScriptContent extrahiert. Des weiteren muss der Zeitpunkt des Einfügens in das Dokument angegeben werden. Dies wird der WkWebViewConfiguration hinzugefügt genauso wie ein ScriptMessageHandler der auf klick Ereignisse reagiert:

```
let scriptURL = NSBundle.mainBundle().pathForResource("main", ofType:
 "js")

let scriptContent = try! String( contentsOfFile: scriptURL!, encoding:
NSUTF8StringEncoding)

let script = WKUserScript(source: scriptContent, injectionTime:
.AtDocumentStart, forMainFrameOnly: true)

config.userContentController.addUserScript(script)

config.userContentController.addScriptMessageHandler(self, name:
 "onclick")
```

Die WkWebViewConfiguration hat nun alle erforderlichen Komponenten und kann zusammen mit der Angabe des Frames welcher hier die Grenzen des ContinerViews sind als SubView des ContainerViews gesetzt werden:

```
self.myWebView = WKWebView(frame: containerView.bounds, configur
self.containerView.addSubview(myWebView!)
```