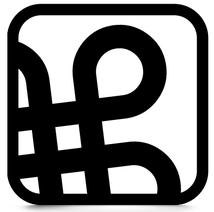


Macoun

Webviewvariationen

Clemens Wagner
macmoonshine@gmx.de

HTML anzeigen

Warum HTML?

- bestehende Inhalte
- bestehende Prozesse
- einfach
- Plattform-Unabhängigkeit

Möglichkeiten

- UIWebView
- WKWebView
- SFSafariViewController

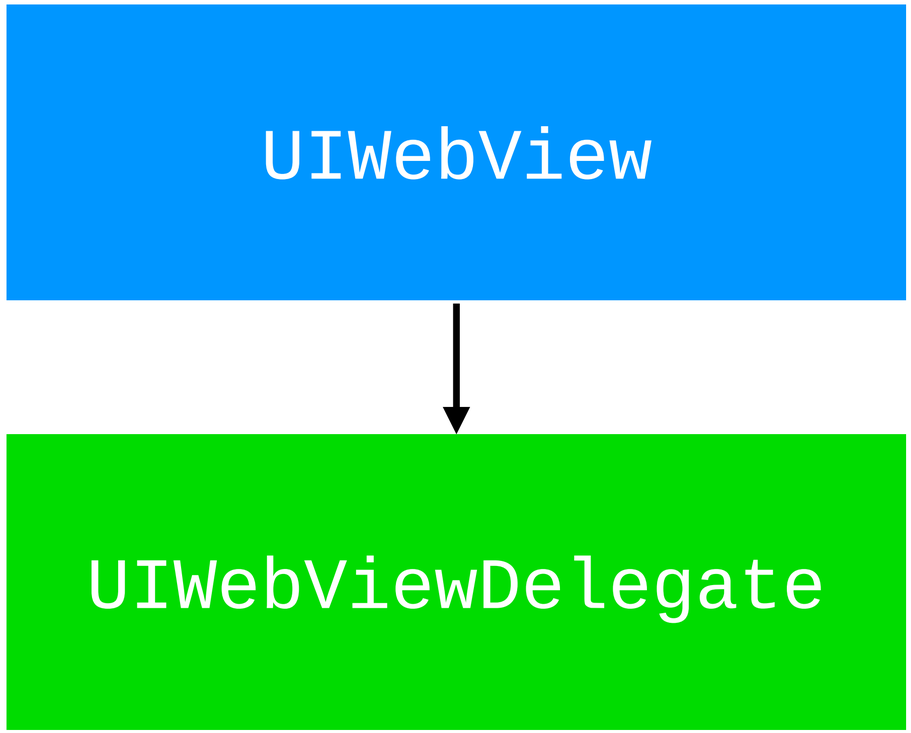
2

8

9

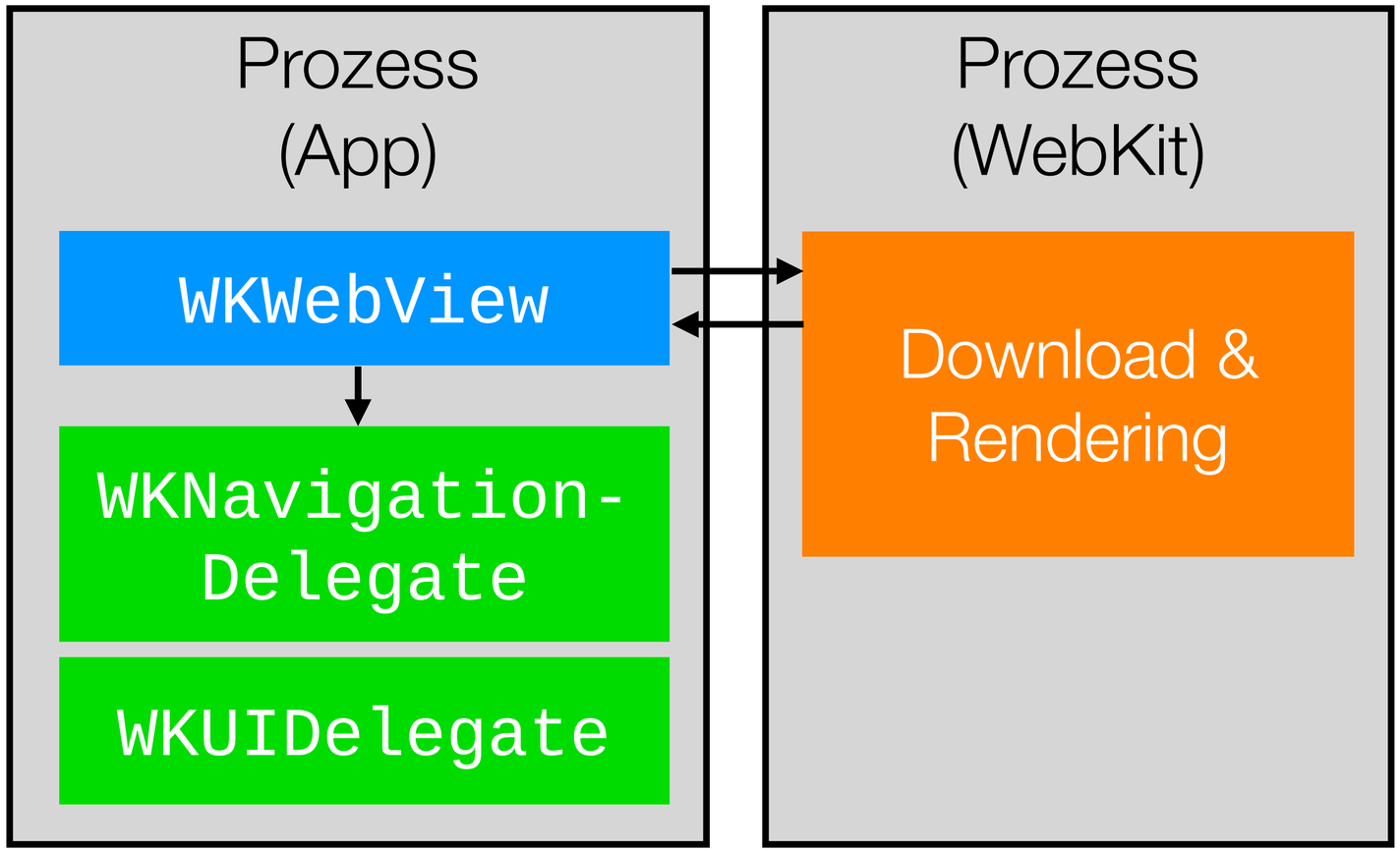
UIWebView

- View + Delegate
- HTML, PDF, SVG, ...
- größte Flexibilität

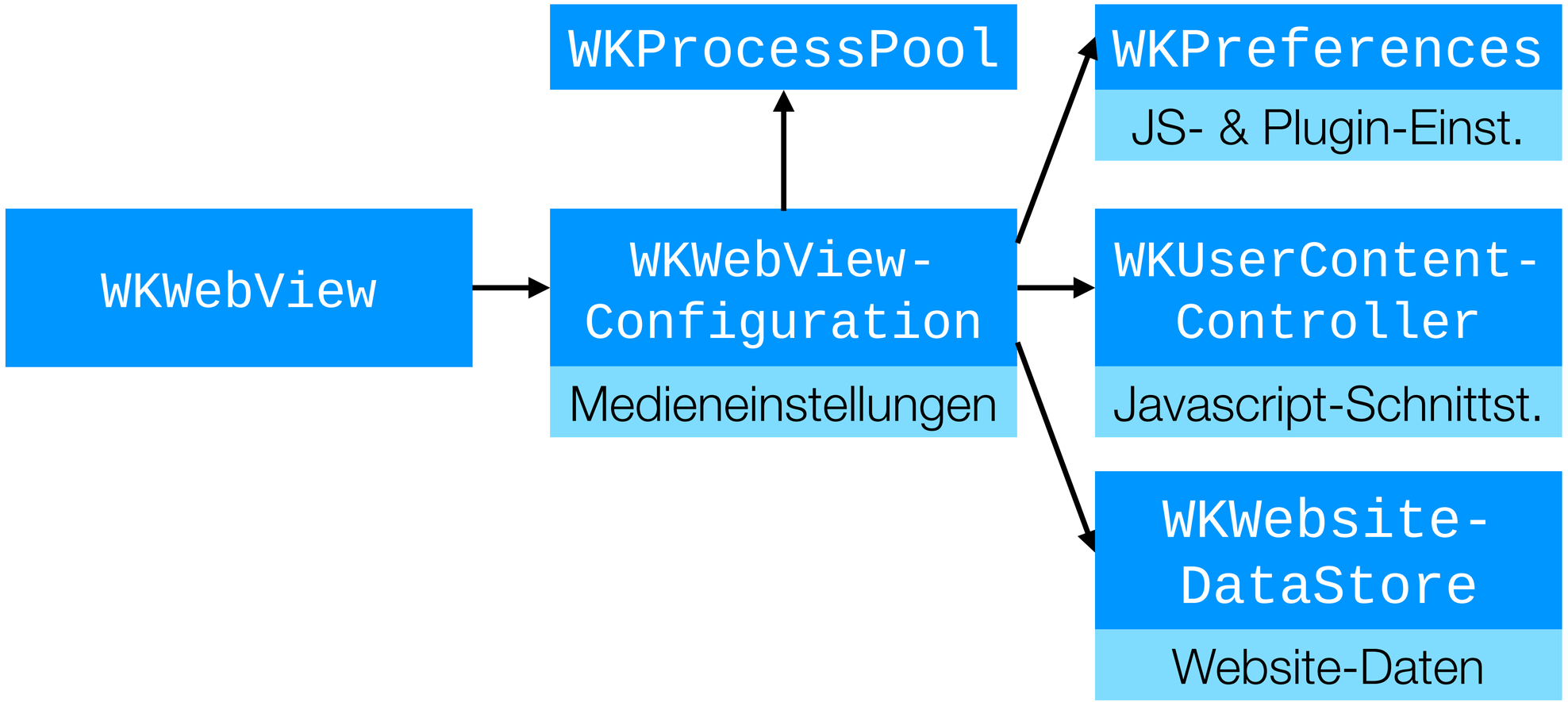


WKWebView

- View & 2 Delegates
- getrennter Prozess
- Stabilität & Sicherheit



WKWebViewConfiguration



SFSafariViewController

- In-App Safari
- einfach
- Standard-Elemente

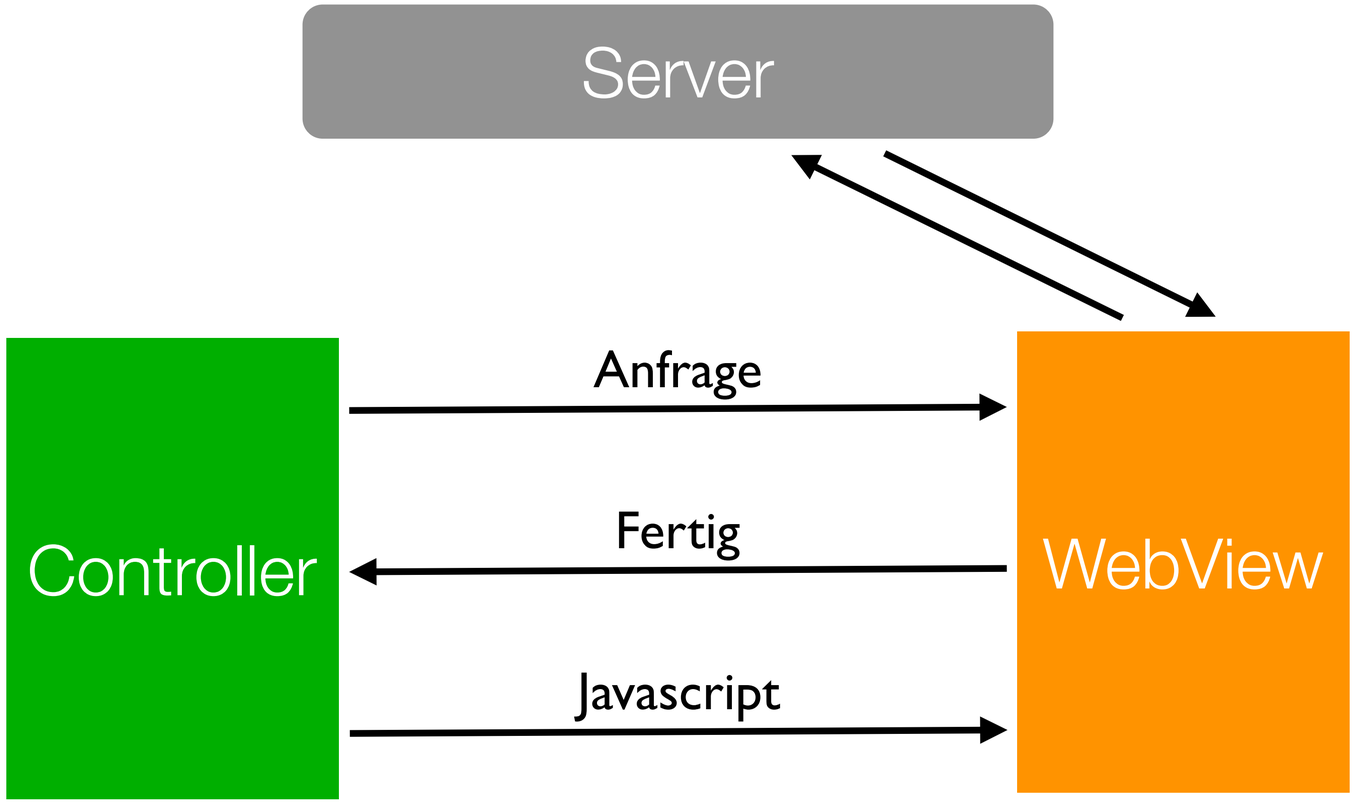
Demo

Web-Inhalte anpassen

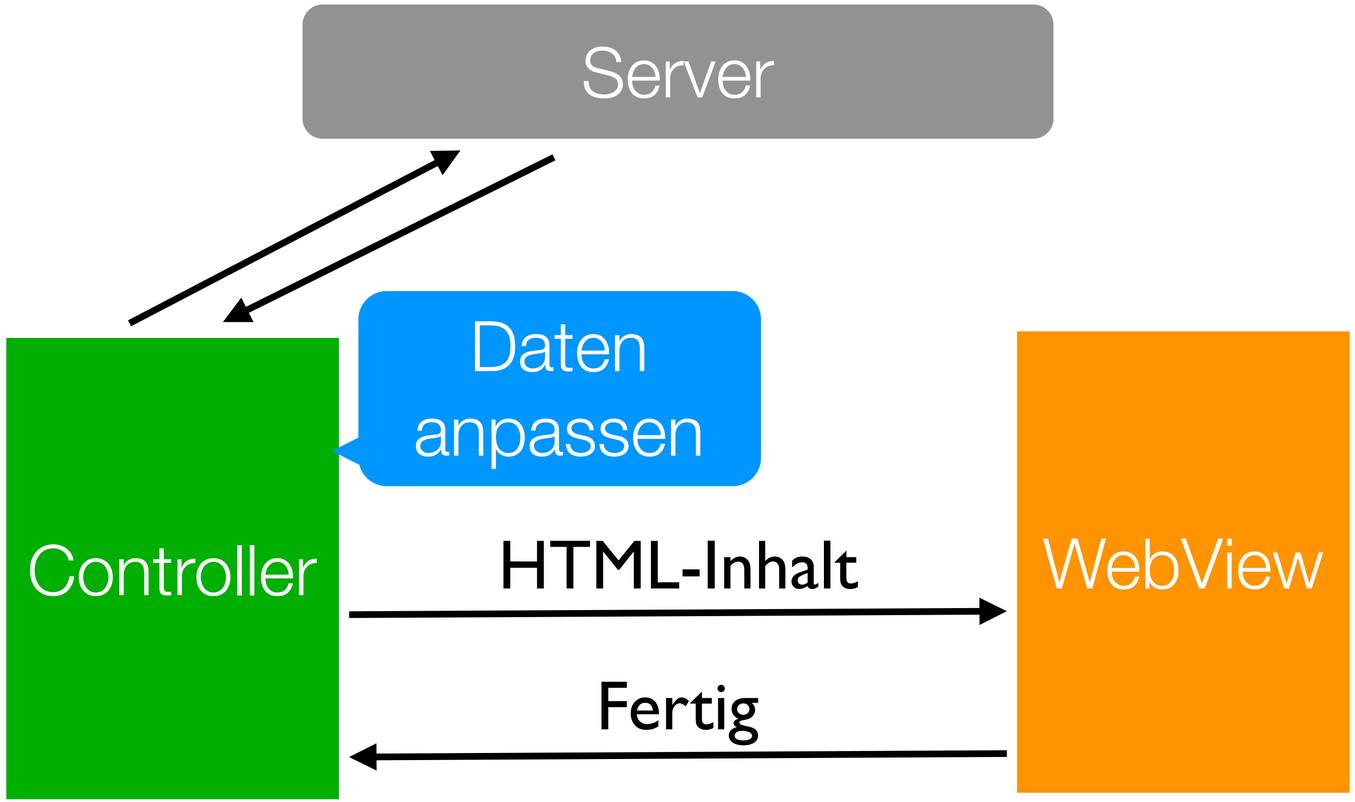
App sieht nach Webseite aus

HTML-Elemente durch native Element ersetzen

Inhalte ändern: Javascript



Inhalte ändern: Vorladen



Inhalt vorladen

```
NSURLSession *theSession = [NSURLSession sharedSession];
NSURLSessionTask *theTask = [theSession dataTaskWithRequest:inRequest
    completionHandler:...];

[theTask resume];
```

Inhalt vorladen

```
^(NSData *inData, NSURLResponse * inResponse, NSError *inError) {
    if(inData == nil) { ... }
    else {
        NSStringEncoding theEncoding = NSUTF8StringEncoding;
        NSString *theContent = [[NSString alloc] initWithData:inData
           encoding:theEncoding];

        // Inhalt anpassen
        [self.webView loadHTMLString:theContent baseURL:inRequest.URL];
    }
}
```

Ressourcen finden

Inhalt anpassen: Suchen

```
<html>
    <head>
        <title>Mein Lexikon</title>
        ...
    </head>
    <body>
        <div id='header'>
```

... und ersetzen

```
<html>
    <head>
        <title>Mein Lexikon</title>
        ...
        <style>form { display: none; height: 0; top: 0; }
          form input { position: absolute; height: 0; top: 0; }</style>
    </head>
    <body>
        <div id='header'>
```

Suchfeld als natives Element

Eingabe an HTML

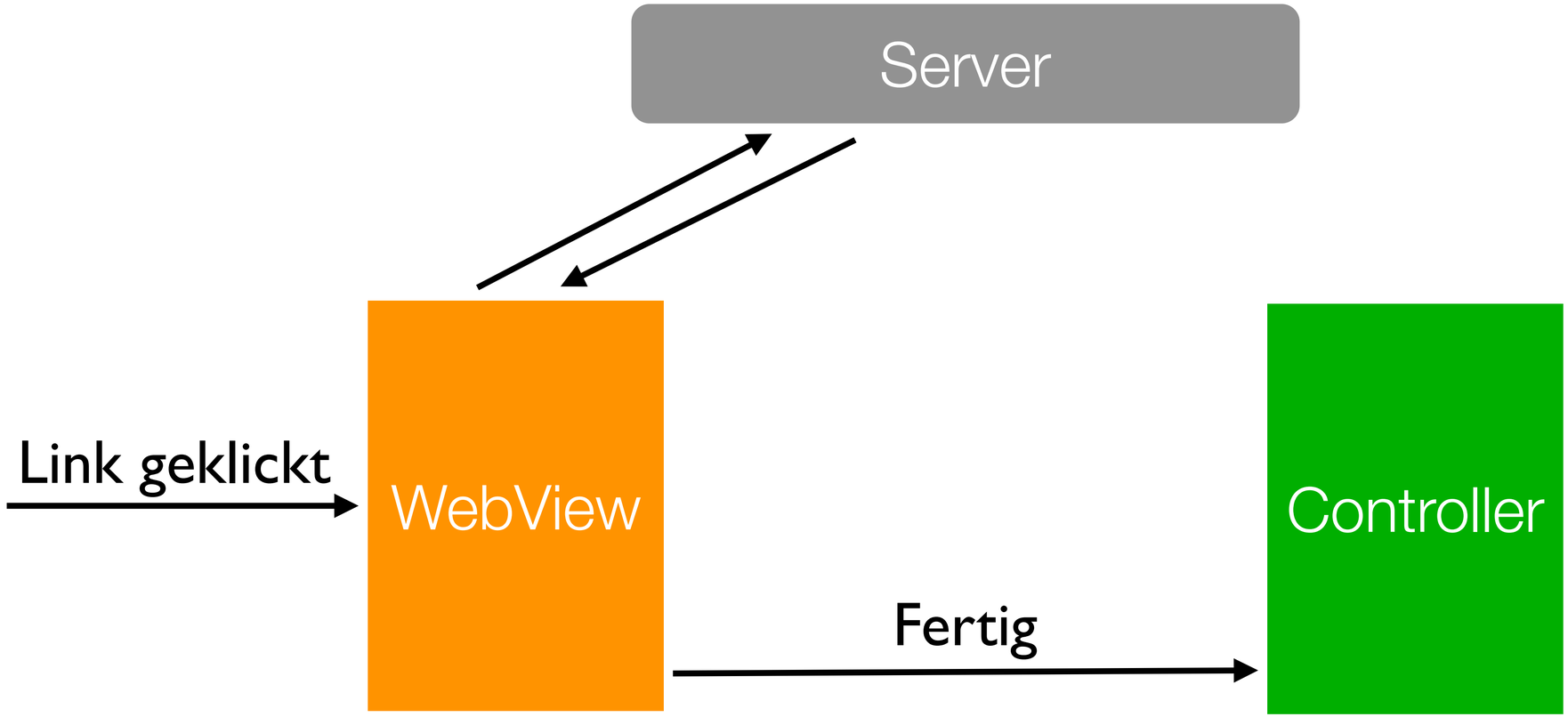
```
-(BOOL)textField:(UITextField *)inTextField shouldChange... {
    if([inText isEqualToString:@"\n"]) { ... }
    else {
        NSString *theText = ...;
        NSString *theScript = [NSString stringWithFormat:
            @"$('#search input[name=q]').val(%@).keydown().keyup();",
                                theText.javaScriptString];

        [self.webView stringByEvaluatingJavaScriptFromString:theScript];
    }
    return YES;
}
```

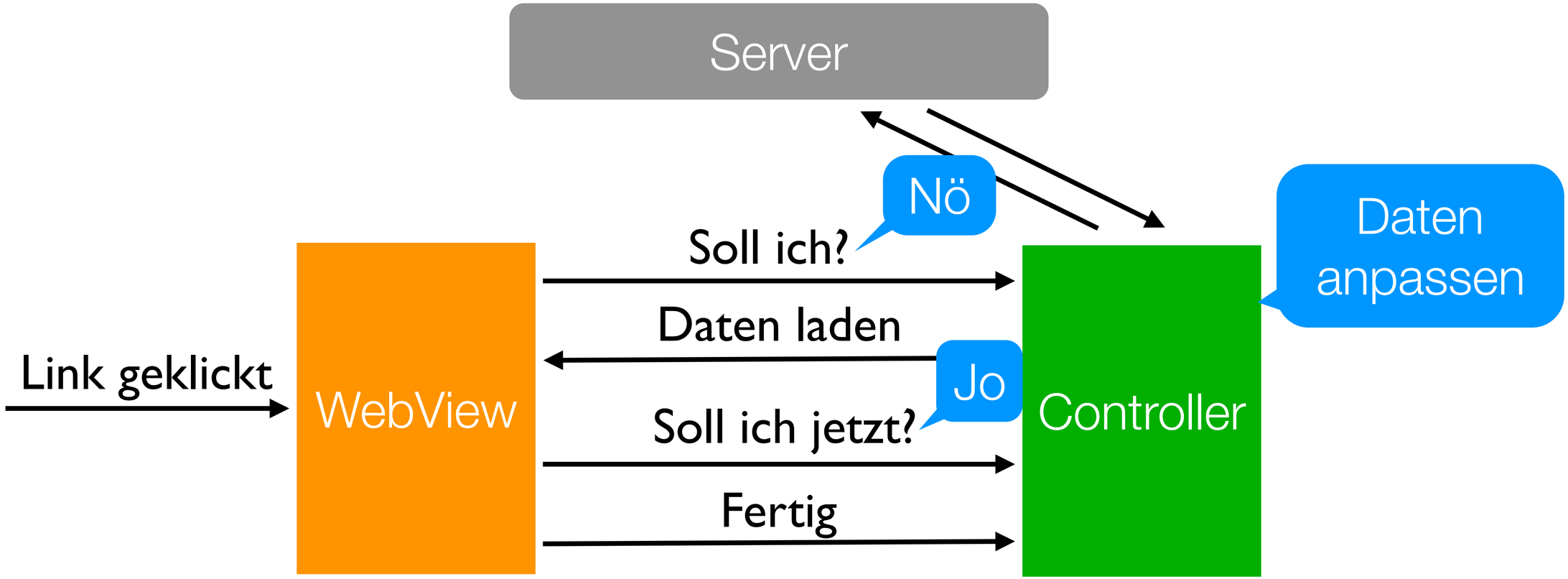
Eingabe an HTML

```
-(BOOL)textField:(UITextField *)inTextField shouldChange... {
    if([inText isEqualToString:@"\n"]) {
        [self.webView stringByEvaluatingJavaScriptFromString:
            @"document.forms.search.submit();"];
        [inTextField endEditing:YES];
    }
    else { ... }
    return YES;
}
```

Was ist mit Links?



Was ist mit Links?



Demo

Ereignisse auslösen

Wozu Ereignisse?

Method-Swizzling in JS

```
function search(inTerm) { ... }

var originalSearch = search;

search = function(inTerm) {
    originalSearch(inTerm);
    // neuen Suchbegriff an App übermitteln
}
```

Ereignis auslösen

```
search = function(inTerm) {
    originalSearch(inTerm);
    window.location.href = "event://search?" + $.param({
        term: inTerm
    });
}
```

eigenes Schema

Ereignistyp / Methode

Ereignis auswerten

```
-(BOOL)webView:(UIWebView *)inWebView
    shouldStartLoadWithRequest:(NSURLRequest *)inRequest
    navigationType:(UIWebViewNavigationType)inNavigationType {
    NSURL *theURL = inRequest.URL;

    if([theURL.scheme isEqualToString:@"event"]) {
        ...
        return NO;
    }
    else { ... }
}
```

Eine Seite pro Webview

- Mehrere Ereignisse problematisch
- Lösung: Frames statt Seiten laden

Frame laden

```
search = function(inTerm) {
    var theFrame = document.createElement("IFRAME");

    originalSearch(inTerm);
    theFrame.setAttribute("src", "event://search?" + $.param({
        term: inTerm
    }));
    document.documentElement.appendChild(theFrame);
    theFrame.parentNode.removeChild(theFrame);
    theFrame = null;
}
```

Skripthandler (WebKit)

```
search = function(inTerm) {
    originalSearch(inTerm);
    window.webkit.messageHandlers.search.postMessage(inTerm);
}
```

Name des Handlers

beliebiges Javascript-Objekt

Skripthandler installieren

```
WKWebViewConfiguration *theConfiguration = ...;
WKUserContentController *theController =
    theConfiguration.userContentController;

[theController addScriptMessageHandler:self name:@"search"];
```

Name des Handlers

Skripthandler implementieren

```
-(void)userContentController:(WKUserContentController *)inController
    didReceiveScriptMessage:(WKScriptMessage *)inMessage {
    if([inMessage.name isEqualToString:@"search"]) {
        self.queryField.text = inMessage.body;
    }
}
```

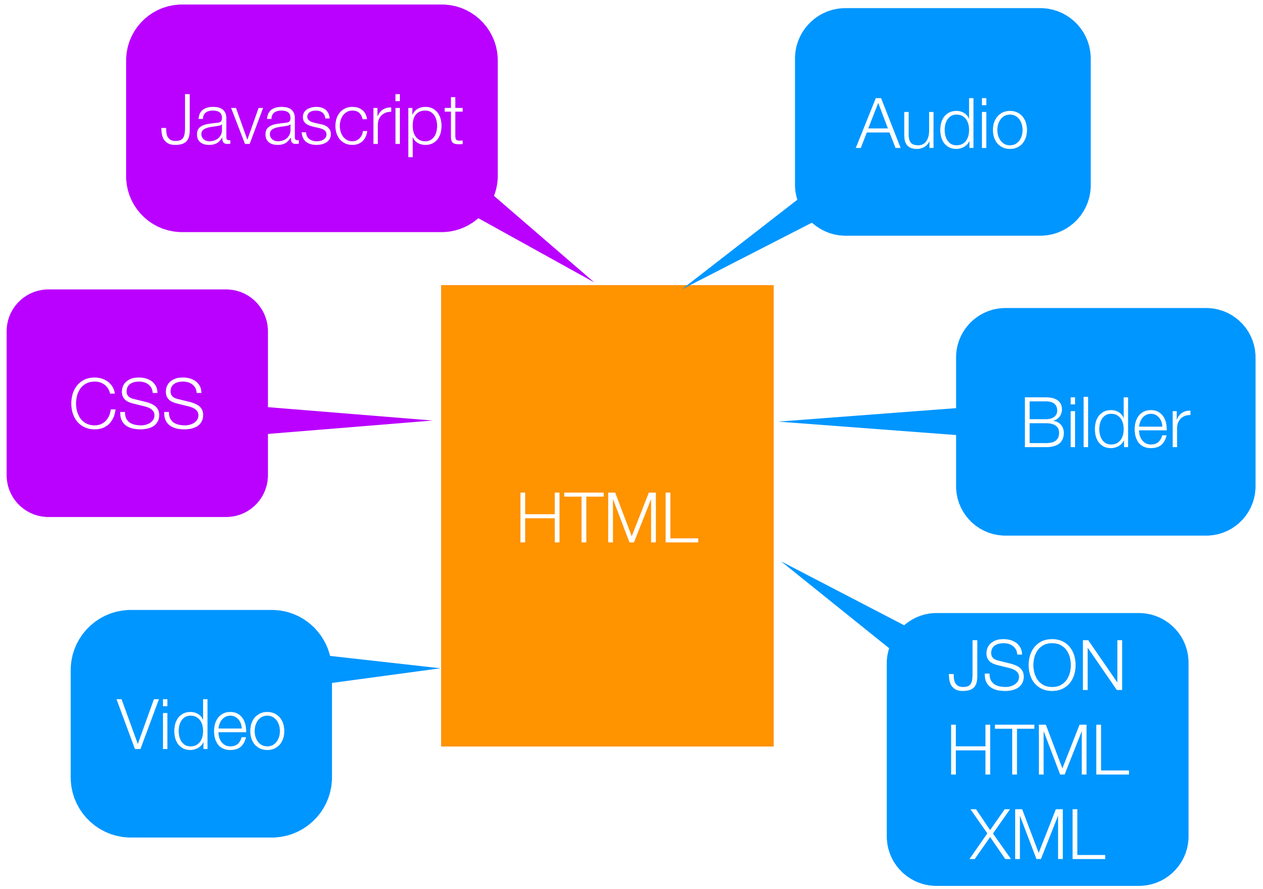
Name des Handlers

Demo

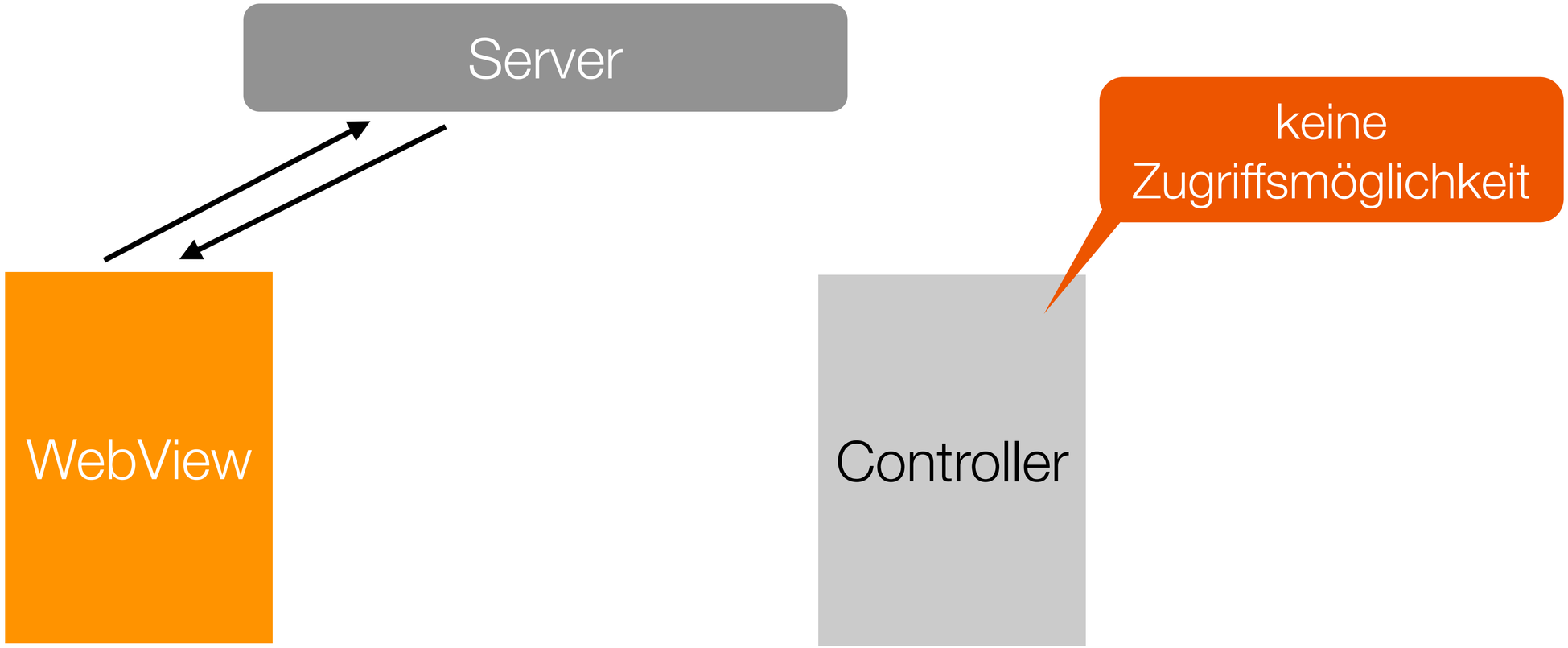
Protokolle

Nachgeladene Inhalte

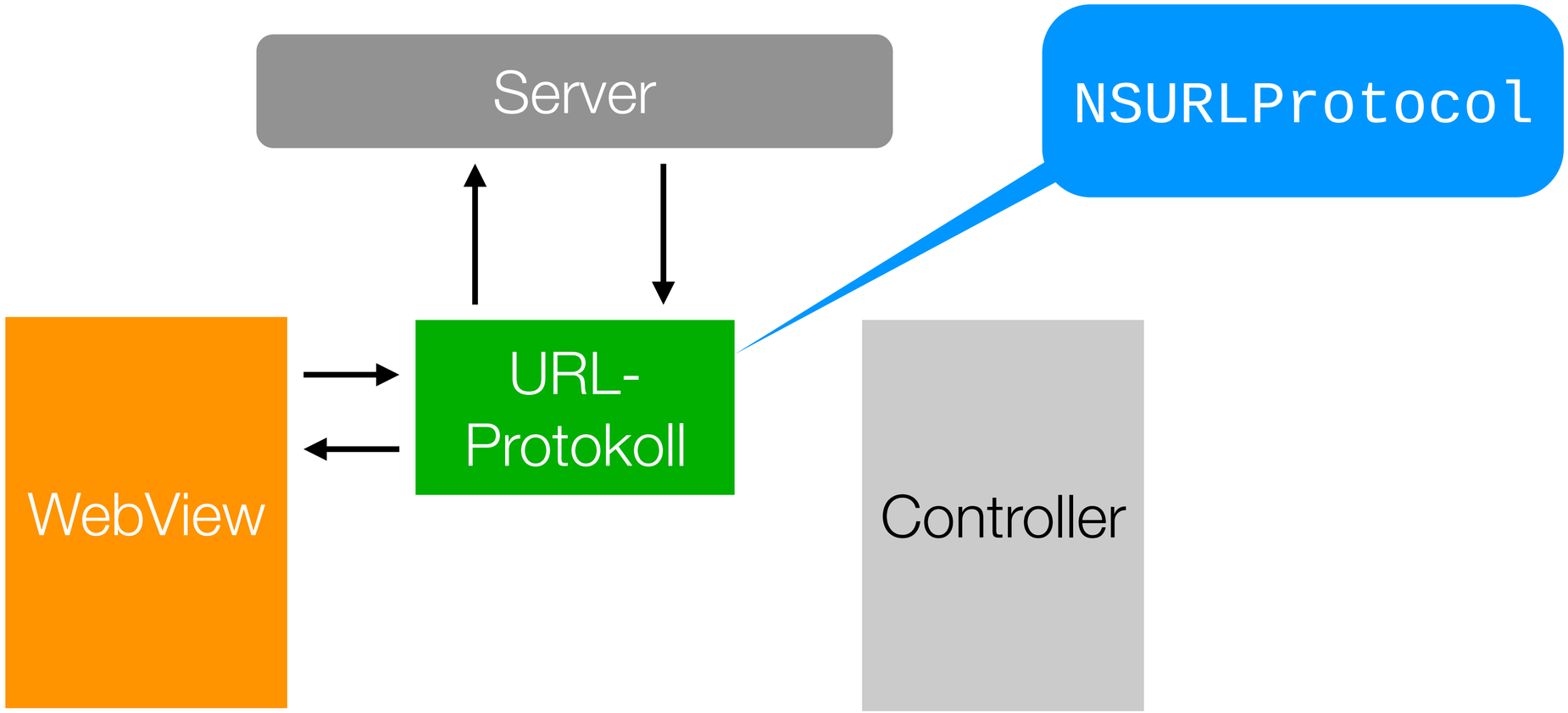
- Verändern
- Generieren
- Caching



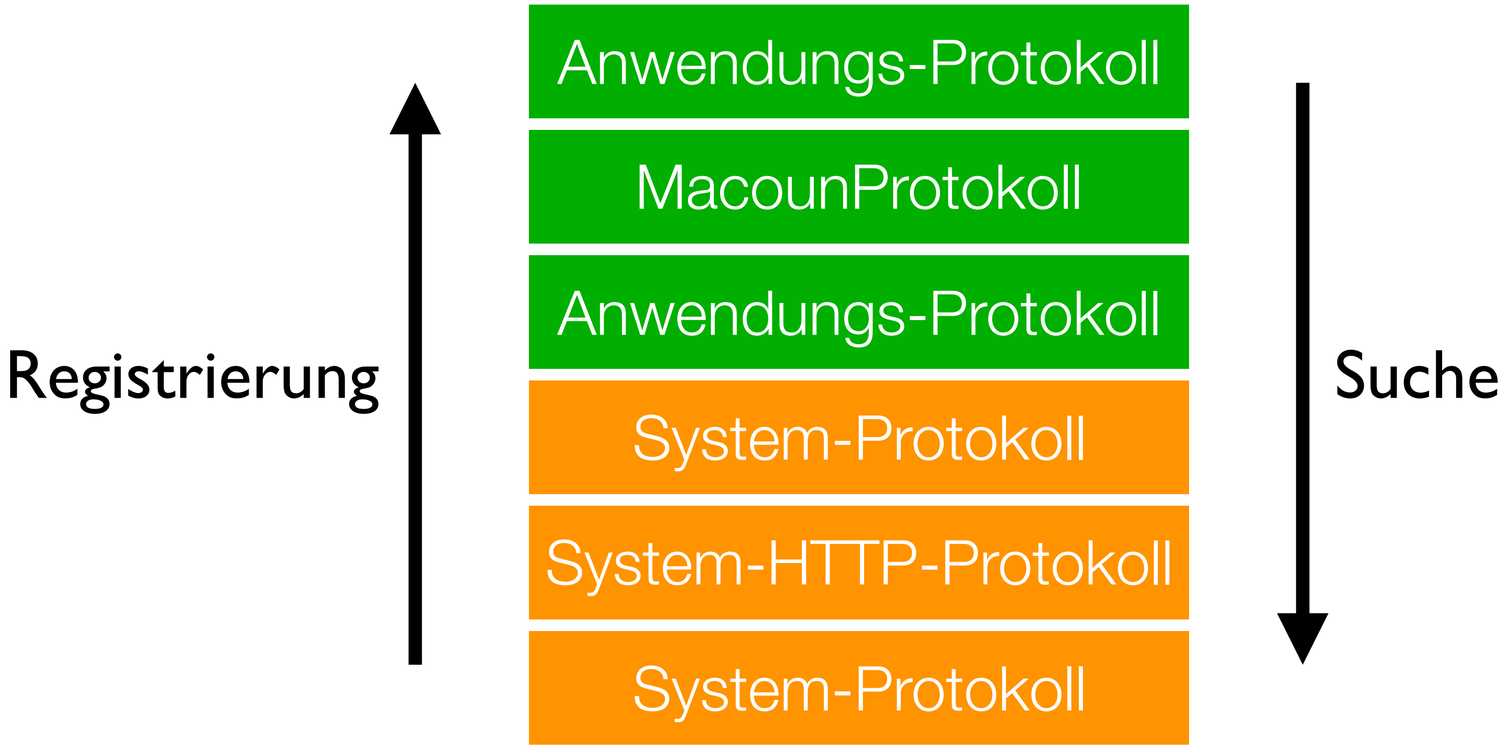
Daten nachladen



URL-Protokolle (UIWebView)



Protokoll-Stack



Protokoll registrieren

```
[NSURLProtocol registerClass:[MacounProtocol class]];
```

```
@implementation MacounProtocol

+ (BOOL)canInitWithRequest:(NSURLRequest *)inRequest {
    NSURL *theURL = [inRequest URL];

    return [theURL.path isEqualToString:@"/autocompletion"];
}
```

Anfrage ausführen (I)

```
- (void)startLoading {
    NSURLSession *theSession = [NSURLSession sharedSession];

    self.task = [theSession dataTaskWithRequest:self.request
        completionHandler:...];
    [self.task resume];
}
- (void)stopLoading {
    [self.task cancel];
    self.task = nil;
}
```

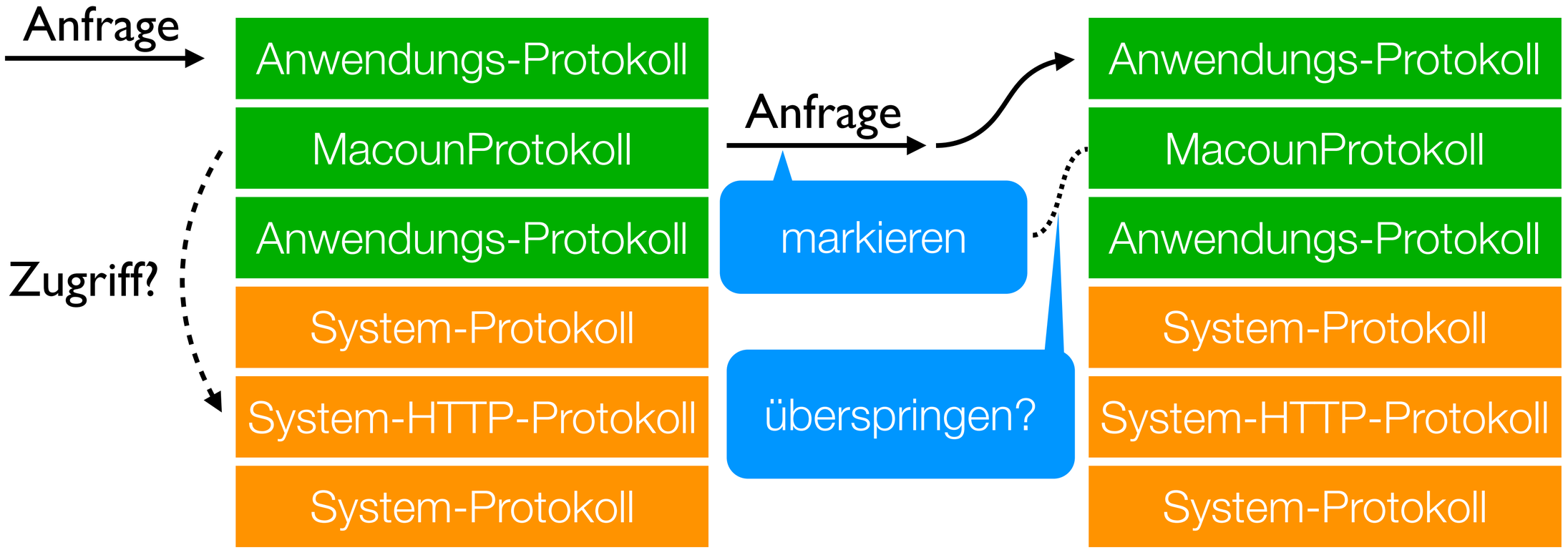
Anfrage ausführen (II)

id<NSURLProtocolClient>

```
^(NSData *inData, NSURLResponse *inResponse, NSError *inError) {
    if(inData == nil) {
        [self.client URLProtocol:self didFailWithError:inError];
    }
    else {
        NSData *theData = [self processData:inData];

        [self.client URLProtocol:self didReceiveResponse:inResponse
            cacheStoragePolicy:NSURLCacheStorageNotAllowed];
        [self.client URLProtocol:self didLoadData:theData];
        [self.client URLProtocolDidFinishLoading:self];
    }
}];
```

Zyklen vermeiden



Anfrage markieren

```
static NSString * const kMacounProtocolSkip = @"kMacounProtocolSkip";

+ (NSURLRequest *)canonicalRequestForRequest:(NSURLRequest *)inRequest {
    NSMutableURLRequest *theRequest = [inRequest mutableCopy];

    [NSURLProtocol setProperty:@YES forKey:kMacounProtocolSkip
        inRequest:theRequest];
    return [theRequest copy];
}
```

Markierte Anfrage überspringen

```
+ (BOOL)canInitWithRequest:(NSURLRequest *)inRequest {
    NSURL *theURL = [inRequest URL];
    id isSkipped = [self propertyForKey:kMacounProtocolSkip
        inRequest:inRequest];

    return ![isSkipped boolValue] &&
        [theURL.path isEqualToString:@"/autocompletion"];
}
```

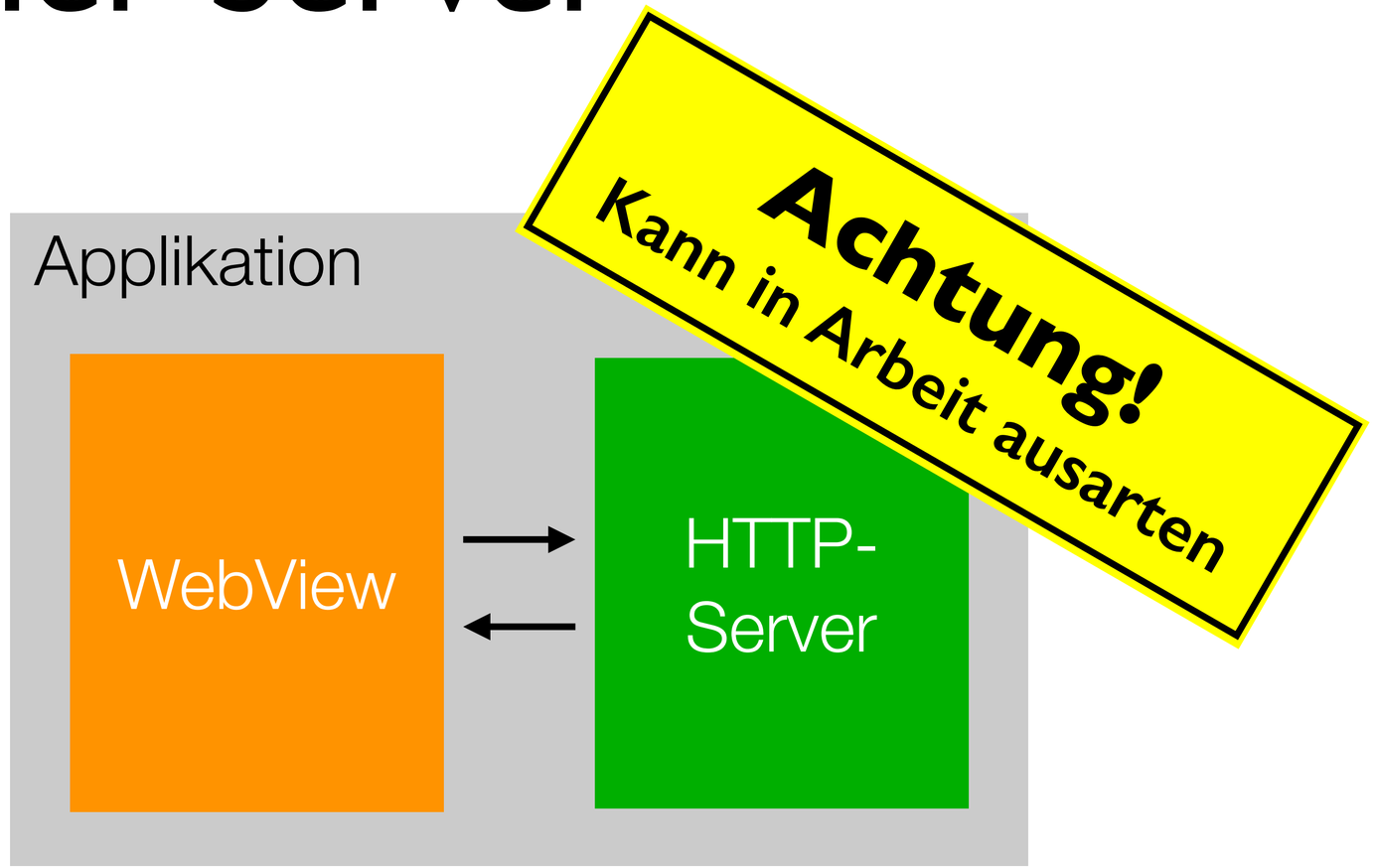
Anwendungsfälle für Protokolle

Demo

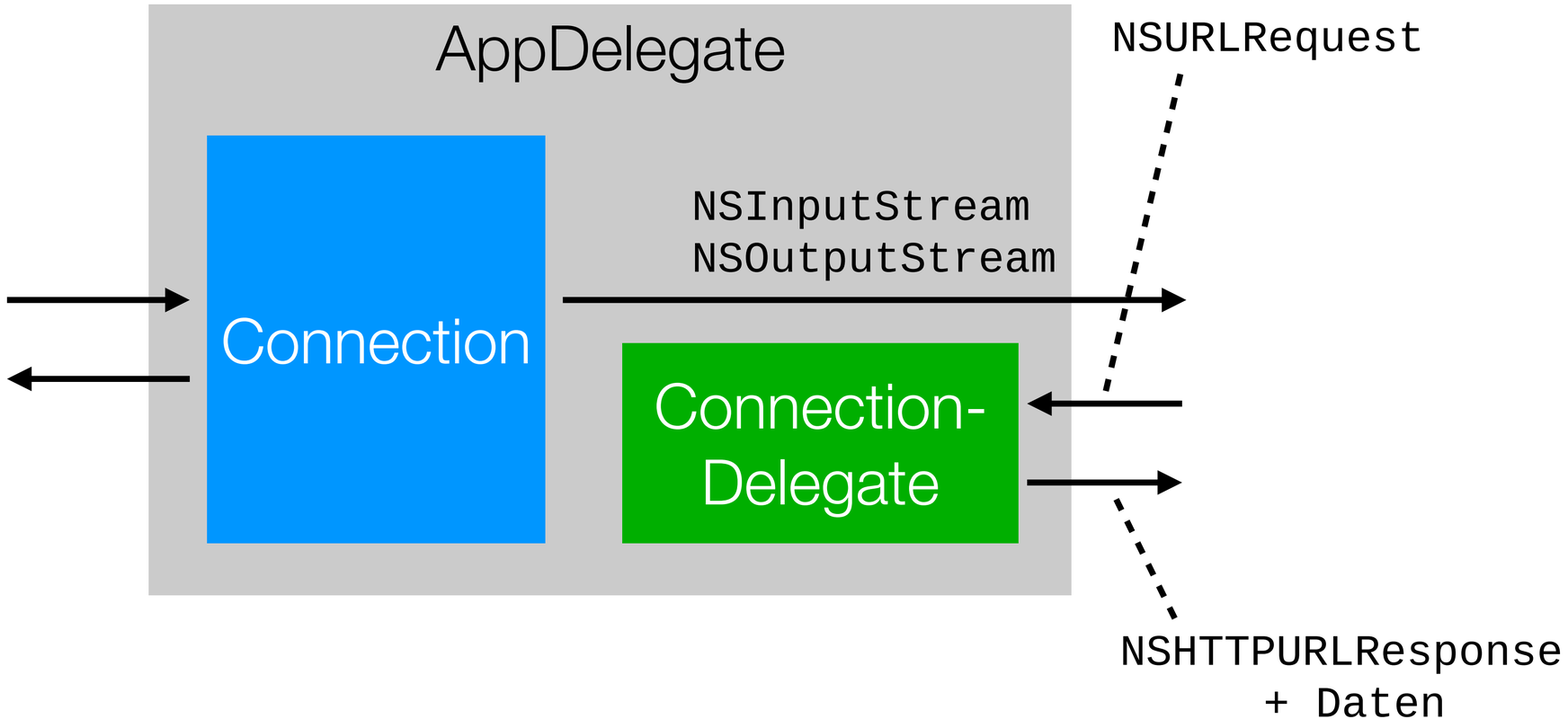
Lokale Server

Lokaler Server

- Anfragen an lokalen HTTP-Server
- `NSNetService`
- 💡 Service auch als Proxy



Lokaler Server

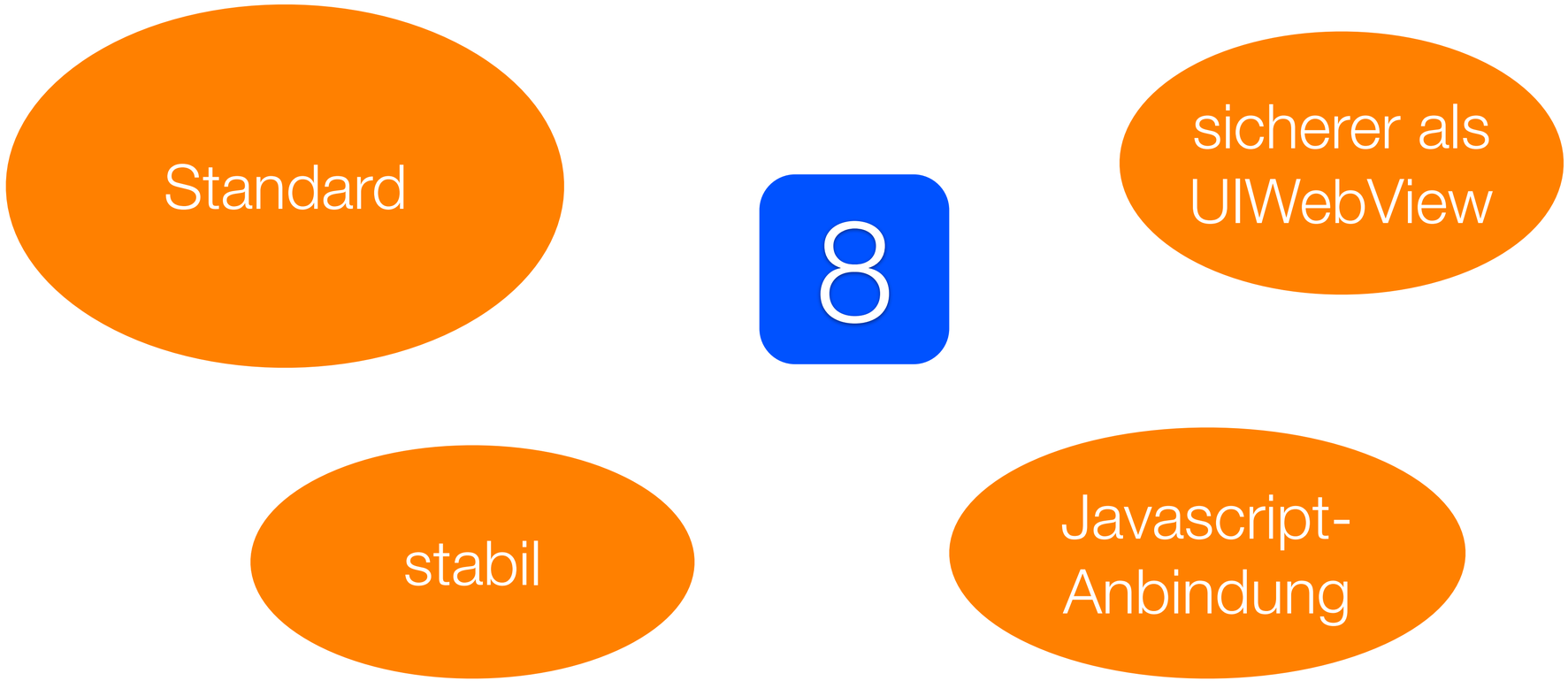


Demo

Entscheidungshilfe

Safari-Viewcontroller

WebKit-Webview



UIKit-Webview

Protokolle

Was ändert sich mit iOS 9?

IPv6

- ab iOS 9 zwingend erforderlich
  - Namen statt Adressen
  - Zugriff: `CFNetwork` / `NSURLSession`

App Transport Security

- HTTPS als Standard
- HTTP für einzelne Domains
- HTTP für Entwicklung
- Ausnahme: `SFSafariViewController`

HTTP für einzelne Domains

```
<!-- Info.plist -->
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSExceptionDomains</key>
    <dict>
    <key>mydomain.com</key>
    <dict>
        <key>NSExceptionAllowsInsecureHTTPLoads</key><true/>
        <key>NSIncludesSubdomains</key><true/>
    </dict>
</dict>
```

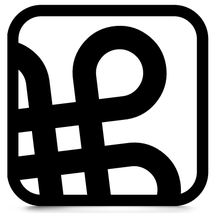
HTTP für Entwicklung

```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

HTTP/2

- Performance-Vorteile
- transparent in NSURLSession
- HTTP/2-Server notwendig

Fragen?



Macoun