

7

Teil II

Overview

- GIT in Xcode ✓
- Die Grundlagen (40min) ✓
- Programmieraufgabe (0,5h – 1h) ✓
- Erweiterte UIElemente (30min)
- Programmieraufgabe (1h - 2h)

Teil II






- UI-Elements:
 - UITableView
 - UINavigationController
 - TabBarController
- Daten senden und Empfangen per HTTP
 - synchron
 - asynchron
- JSON in Obj-C

UITableView



UITableView Styles

plain

More		Edit
	Audiobooks	>
	Tones	>
	Genius	>
	Purchased	>
	Downloads	>

grouped

< Settings Maps	
DISTANCES	
In Miles	✓
In Kilometers	
MAP LABELS	
Always in English	<input checked="" type="checkbox"/>
PREFERRED DIRECTIONS	
Driving	✓
Walking	

UITableView

- UITableView benötigt zwei Delegates:
 - UITableViewDataSource
 - UITableViewDelegate
- normalerweise werden beide von einem Controller implementiert

```
@interface SomeViewController ()<UITableViewDataSource, UITableViewDelegate>
```

```
@end
```

```
@implementation SomeViewController
```

```
....
```

UITableViewDataSource







- stellt die anzuzeigenden Daten bereit
- erstellt die Zellen der Table
- Benötigte Methoden:

```
-(NSInteger)tableView:(UITableView *)tableView  
    numberOfRowsInSection:(NSInteger)section;
```







```
-(UITableViewCell *)tableView:(UITableView *)tableView  
    cellForRowAtIndexPath:(NSIndexPath *)indexPath;
```

NSIndexPath like a C-Struct
contains row and section







UITableView Cells

Default Cell Style	
	Text Label
	Dahlia
	Daisies
	Dandelion
	Echinacea
	Lavender

UITableViewCellStyleDefault

Value 1 Cell Style	
	Text Label Detail text label
	Dahlia This is a dahlia
	Daisies These are daisies
	Dandelion This is a dandelion
	Echinacea This is echinacea
	Lavender This is a field of lavender

UITableViewCellStyleValue1

Subtitle Cell Style	
	Text Label Detail text label
	Dahlia This is a dahlia
	Daisies These are daisies
	Dandelion This is a dandelion
	Echinacea This is echinacea
	Lavender

UITableViewCellStyleSubtitle

Value 2 Cell Style	
Text Label	Detail text label
Dahlia	This is a dahlia
Daisies	These are daisies
Dandelion	This is a dandelion
Echinacea	This is echinacea
Lavender	This is a field of lavender

UITableViewCellStyleValue2

UITableViewDataSource

- Bevorzugte Art Zellen anzulegen:

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell* cell = [tableView dequeueReusableCellWithIdentifier:@"standard"];

    if(!cell){
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:@"standard"];
    }

    cell.textLabel.text = @"Text";
    cell.detailTextLabel.text = @"Detail Text";
    cell.imageView.image = [UIImage imageNamed:@"image.png"];
    return cell;
}
```

UITableViewDelegate

- Benötigt um weitere Funktionen bereitzustellen
- Besitzt nur optionale Methoden

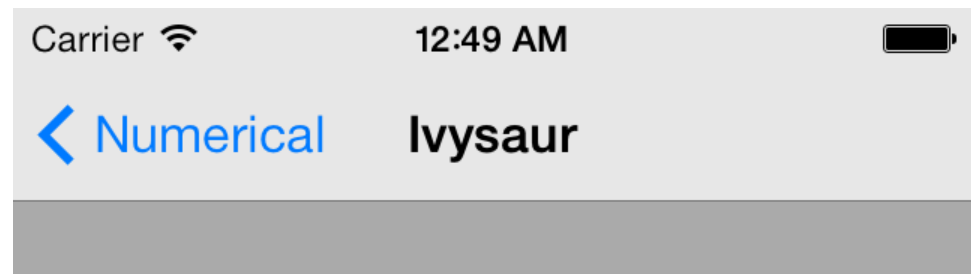
z.B. für User-Interaktion:

// called when user selects a row in the table

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:  
    (NSIndexPath *)indexPath;
```

UINavigationController

- Stellt Vorwärts/Rückwärts Funktionen bereit
- Views werden in einen Stack gelegt
- erstellt automatisch einen Balken am oberen Rand



UINavigationController

Initialisierung im AppDelegate:

```
UIViewController *viewController1 = [[FirstViewController alloc]
                                     init];
viewController1.title = @"Some Title";

UINavigationController* nav1 = [[UINavigationController alloc]
initWithRootViewController:viewController1];

self.window.rootViewController = nav1;
```

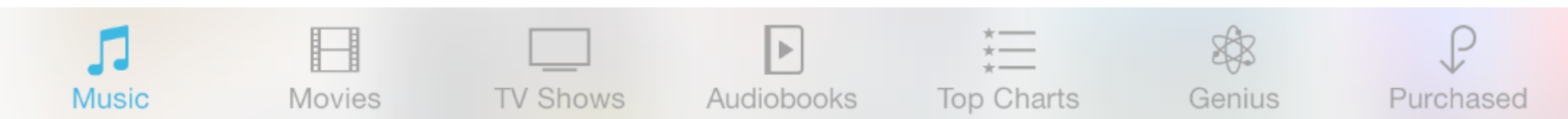
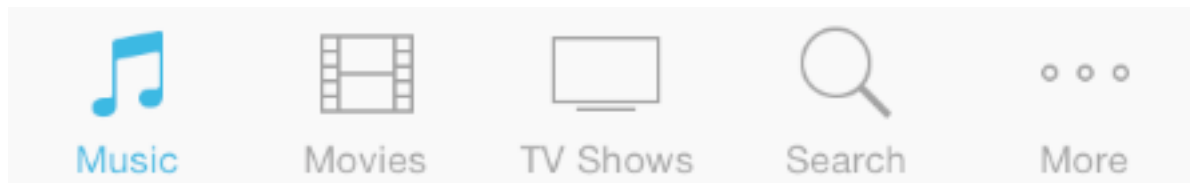
UINavigationController Usage

```
[self.navigationController pushViewController: CONTROLLER  
                                           animated:YES];
```

```
[self.navigationController popViewControllerAnimated:YES];
```

```
[self.navigationController  
    popToRootViewControllerAnimated:YES];
```

UITabBarController



UITabBarController

- Top Level Controller
- Gruppiert mehrere Controller in Tabs
- Views können über Index adressiert werden
- erstellt automatisch die TabBar
(Platz lassen!!)

UITabBarController

Initialisierung in AppDelegate:

```
UIViewController *view1 = [[FirstViewController alloc] init];  
view1.title = @"Some Title";  
view1.tabBarItem.image = [UIImage imageNamed:@"imagename"];  
  
//...nochmal für view2  
  
UITabBarController tabbar = [[UITabBarController alloc] init];  
tabbar.viewControllers = @[view1, view2]; // magic array initialization  
  
self.window.rootViewController = tabbar;
```


UITabBarController + UINavigationController

```
UIViewController *viewController1 = [[FirstViewController alloc] init];  
UINavigationController* nav1 = [[UINavigationController alloc]  
    initWithRootViewController:viewController1];
```

```
UIViewController *viewController2 = [[SecondViewController alloc] init];  
UINavigationController* nav2 = [[UINavigationController alloc]  
    initWithRootViewController:viewController2];
```

```
UITabBarController* tabs = [[UITabBarController alloc] init];  
tabs.viewControllers = @[nav1, nav2];
```

```
self.window.rootViewController = self.tabBarController;
```

JSON in Obj-C

- Java-Script-Object-Notation
- einfache Benutzung in ObjC
- Benötigt keine externen Bibliotheken

JSON Beispiel Code:

```
+(NSArray*) parseJSON:(NSData*) json {
    NSError* error;
    NSArray* array = [NSJSONSerialization JSONObjectWithData:json
                                                            options:kNilOptions error:&error];

    if(!error){ // kein Fehler aufgetreten,
        return array;
    } else {
        return nil;
    }
}
```

Woher kriege ich NSData* json?

einfach: [@"string" dataUsingEncoding: NSUTF8StringEncoding]; // NSData
besser: nächste Folie 😊

Synchrones empfangen von Daten

- iOS bietet Funktionen um über HTTP Daten zu empfangen
- Dazu werden folgende Klassen benötigt:
 - NSURL
 - NSURLRequest
 - NSURLConnection

Beispiel für synchrones Empfangen

```
-(NSData*) useSynchronousFetch {  
    NSURL* url = [NSURL URLWithString:@"http://google.de"];  
  
    NSURLRequest* request = [NSURLRequest requestWithURL:url  
                             cachePolicy:NSURLRequestReloadIgnoringLocalCacheData  
                             timeoutInterval:5.0];  
  
    NSURLResponse *response;  
    NSError *error;  
  
    NSData *responseData = [NSURLConnection sendSynchronousRequest:request  
                                           returningResponse:&response error:&error];  
  
    if(!error){  
        return responseData;  
    } else {  
        return nil; // oder NSURLResponse checken warum Fehler auftrat  
    }  
}
```

kleines Intermezzo

- Selector
 - identifiziert Methode durch ihre Signatur
 - @selector (methodname) // keine Parameter
 - @selector (methodname:) // 1 Parameter
 - @selector (methodname:name:) // 2 Parameter
- Handler
 - inline C-Funktion
 - ^(parameters){ block }

Daten asynchron Empfangen


Bevorzugte Variante Daten zu empfangen, denn UI wird nicht blockiert.

Möglichkeiten:

- Handler
- Delegate
- Selector

Handler

```
[NSURLConnection sendAsynchronousRequest:(NSURLRequest *)  
queue:(NSOperationQueue *)  
completionHandler:^(NSURLResponse *, NSData *, NSError *)handler]
```



C-Function mit 3 Parametern,
wird nicht auf dem UIThread ausgeführt

Delegate

```
[[NSURLConnection alloc] initWithRequest:NSURLRequest  
delegate:self startImmediately:YES];
```

Delegate sollte NSURLConnectionDataDelegate implementieren:

- (void)connection:(NSURLConnection *)con
didReceiveResponse:(NSURLResponse *) res;
- (void)connection:(NSURLConnection *)con
didReceiveData:(NSData *)data;

Selector

1. Einen Selector auf einem nicht UIThread ausführen:

```
[self performSelectorInBackground:@selector(nonUIMethod:)
    withObject:OBJ];  
// nonUIMethod: kann nun synchron Daten empfangen
```

2. Einen Selector wieder auf dem UIThread ausführen:

```
[self performSelectorOnMainThread:@selector(methodOnUIThread:)
    withObject:OBJ waitUntilDone:YES];
```

OBJ kann benutzt werden um ein Objekt zu übergeben,
zB ein NSString oder NSArray für mehrere Objekte

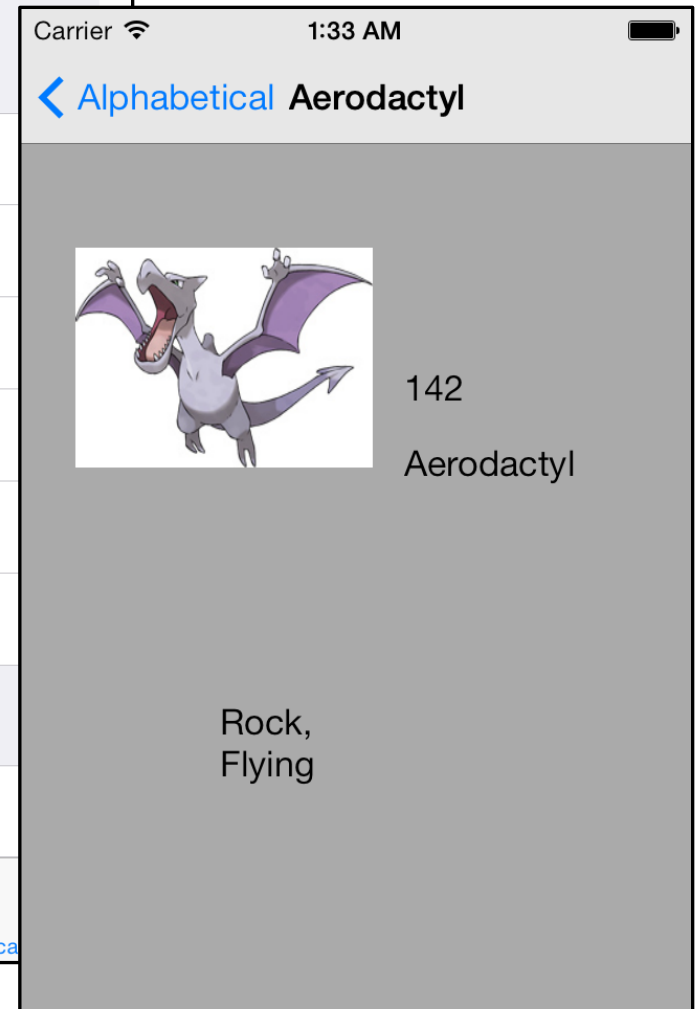
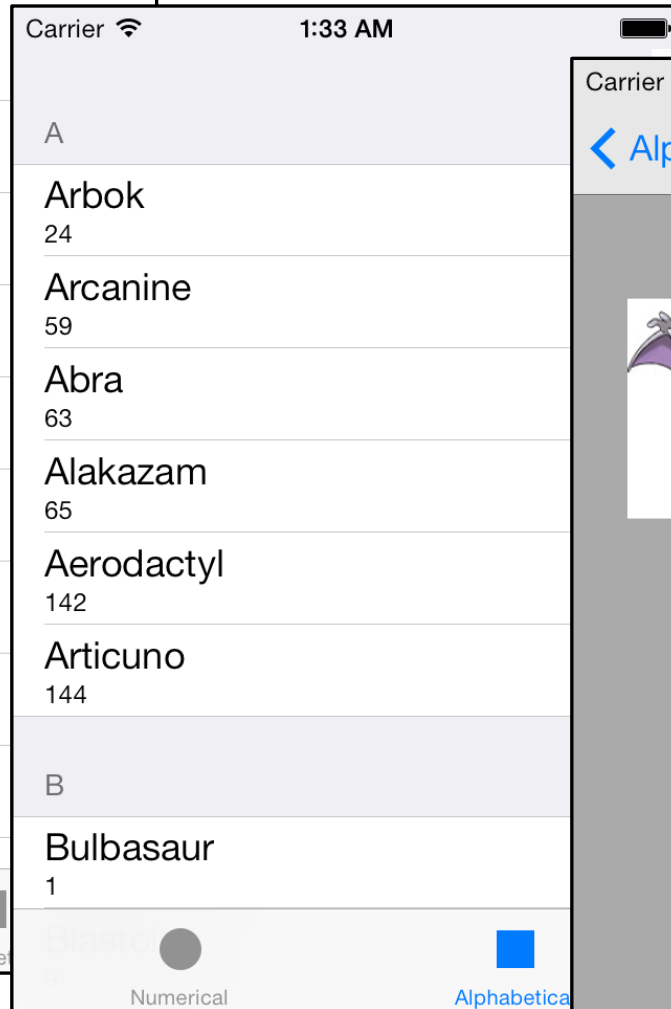
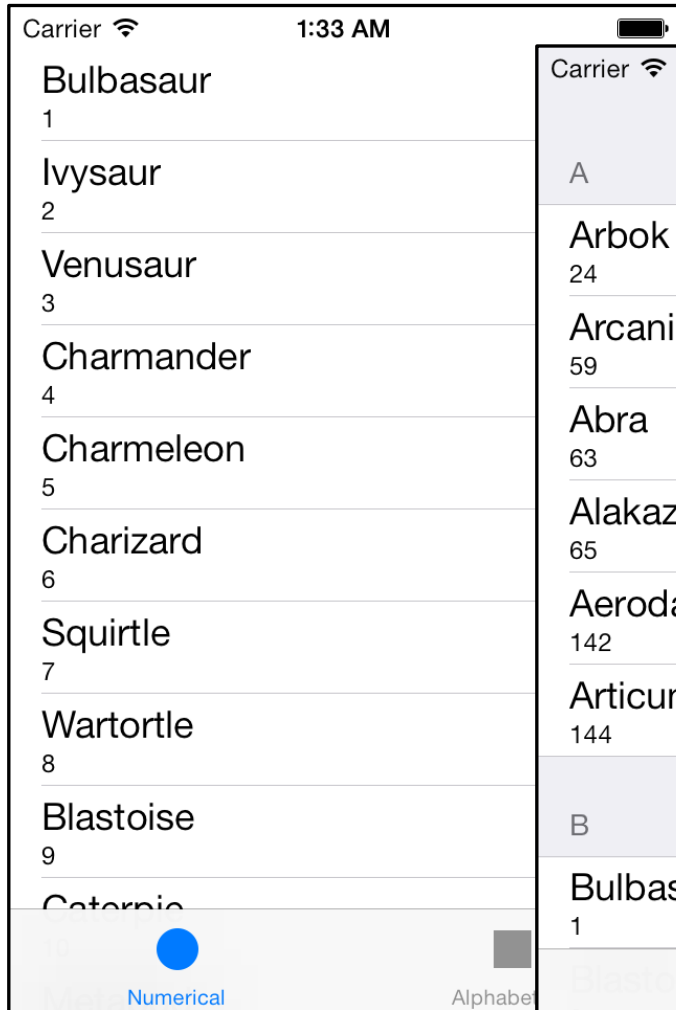
Hinweise

- `UIActivityIndicatorView` benutzen um den Nutzer über andauernde Prozesse zu informieren
- Den globalen `NetworkActivityIndicator` benutzen um Netzwerkaktivität zu signalisieren

Zweite Aufgabe

- Der Pokedex
- Requirements:
 - Liste der Pokemons
 - JSON gibt's hier: <http://posdorfer.de/mlab/pokemon.json>
 - Detailansicht des selektierten Pokemons
 - Name, Nummer, Typen und Bild
 - NavController benutzen
- **Erweitert:** TabBarController benutzen um eine zweite Liste mit einer anderen Sortierung und „Grouped-Style“ anzuzeigen

Zweite Aufgabe



Nützliche Informationen

- JSON

Code enthält: Nummer, Name, Typen und Bild-URL

Result besteht aus:

NSArray von NSDictionary's mit `NSInteger`, `NSString`, `NSArray`, `NSURL`

- Arrays können sortiert werden:

- `NSArray*` sortedArray = [array
sortedArrayUsingComparator:`NSComparator`];

- Grouped-Style:

- Optionale Methoden vom `UITableViewDelegate`
 - benötigt `NSDictionary` um Sections und Header zu speichern