

Adaptive Apps

Die neue Anpassungsfähigkeit

Thomas Dubiel

CocoaHeads Dresden, 13.08.2014



Bis Xcode 5.x / iOS 7 gibt es:

- Interface Idioms
- Interface Orientation
- Size

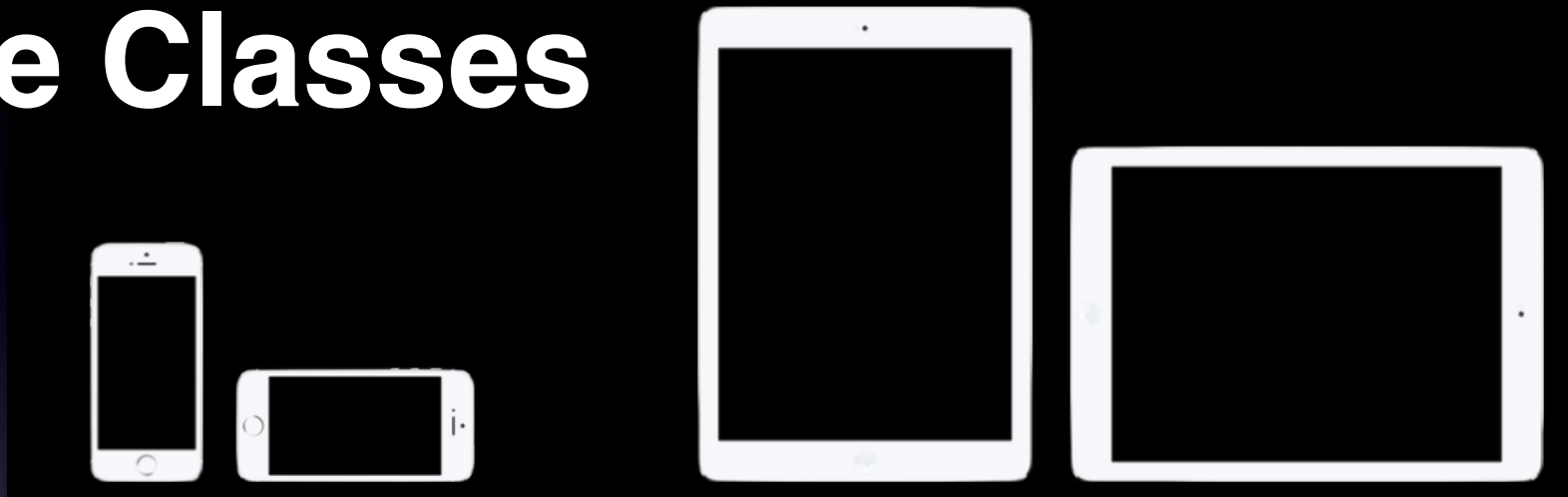
Xcode 6.0 / iOS8 brachte Adaptivity

Adaptive Apps haben:

- Size Classes
- Traits
- Trait Collections
- Trait Environment

Size Classes

Size Classes

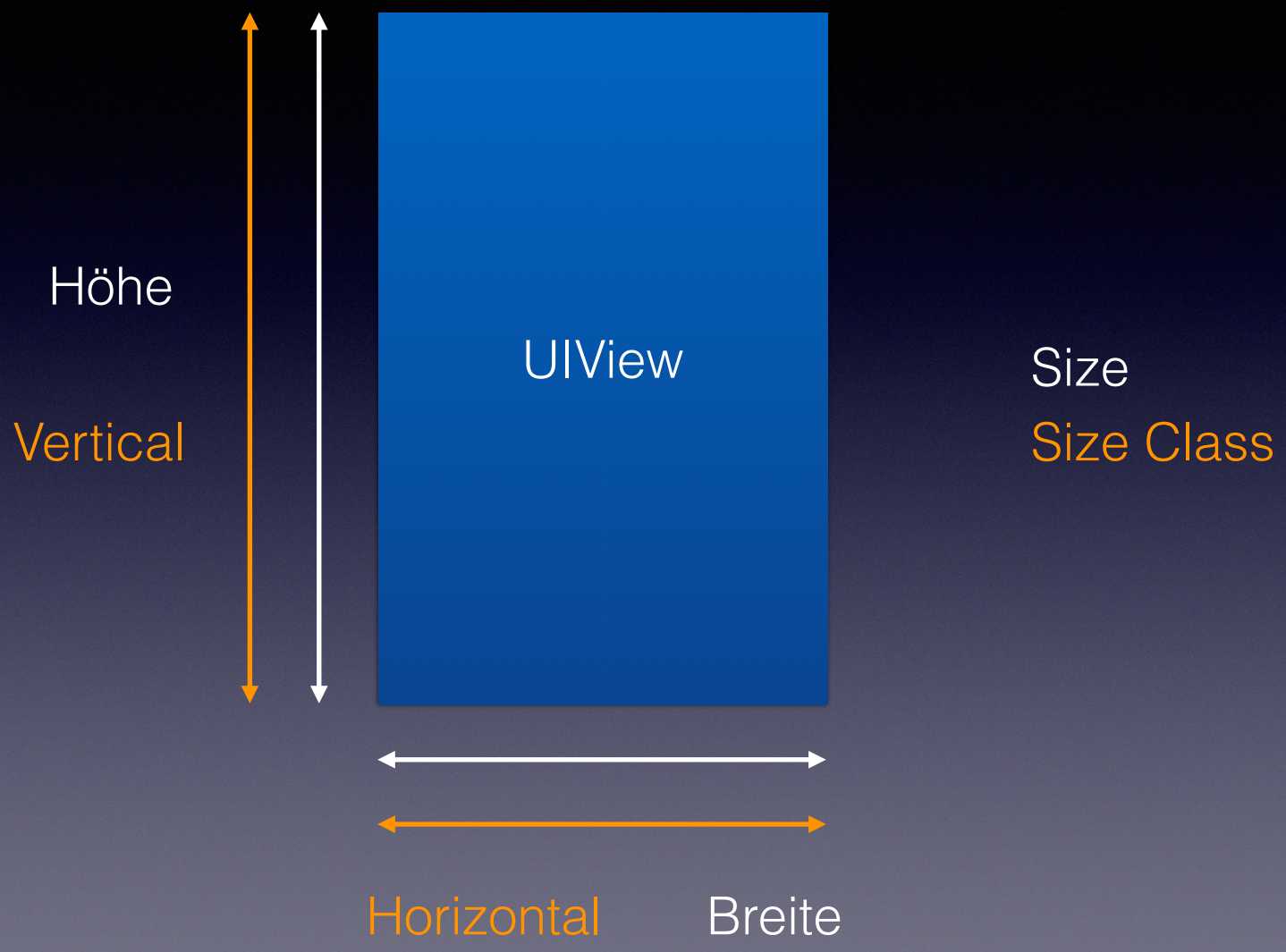


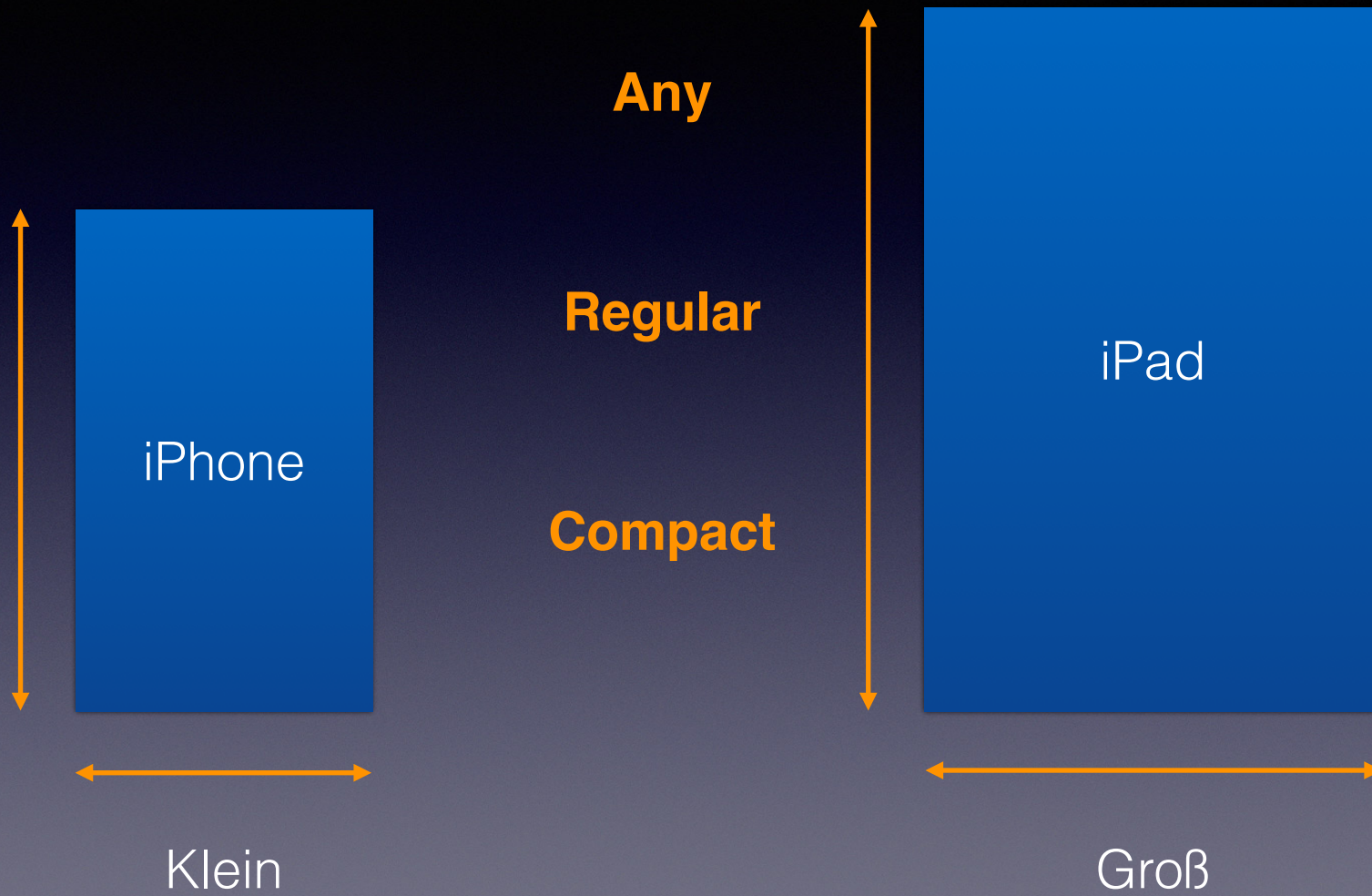
UIInterfaceOrientation

UIUserInterfaceIdiom



What is a „size class“?

It's a trait that coarsely defines the space available





Default Size Classes

		Horizontal	
Vertical		Regular	Compact
	Regular	 iPad	iPhone Portrait iPad Primary Split View Controller
	Compact		iPhone Landscape iPhone Primary Split View Controller

Ein Beispiel



Vertical

	Regular	Compact
Regular	iPad	iPhone Portrait
Compact		iPhone Portrait

Adaptive Apps haben:

- Size Classes
- Traits
- Trait Collections
- Trait Environment

Traits

Traits

```
@property (nonatomic, readonly) UIUserInterfaceSizeClass horizontalSizeClass;  
@property (nonatomic, readonly) UIUserInterfaceSizeClass verticalSizeClass;  
@property (nonatomic, readonly) UIUserInterfaceIdiom userInterfaceIdiom;  
@property (nonatomic, readonly) CGFloat displayScale;
```


Adaptive Apps haben:

- Size Classes
- Traits
- Trait Collections
- Trait Environment

Trait Collections

Trait Collections

horizontalSizeClass	Compact
verticalSizeClass	Regular
userInterfaceIdiom	iPhone
displayScale	2.0

horizontalSizeClass	Compact
verticalSizeClass	Unspecified
userInterfaceIdiom	Unspecified
displayScale	Unspecified

```
+traitCollectionWithHorizontalSizeClass:(UIUserInterfaceSizeClass)horizontalSizeClass;  
+traitCollectionWithVerticalSizeClass:(UIUserInterfaceSizeClass)verticalSizeClass;  
+traitCollectionWithUserInterfaceIdiom:(UIUserInterfaceIdiom)idiom;  
+traitCollectionWithDisplayScale:(CGFloat)scale;
```


Trait Collections

horizontalSizeClass	Compact
verticalSizeClass	Regular
userInterfaceIdiom	iPhone
displayScale	2.0

Contains



horizontalSizeClass	Compact
verticalSizeClass	Unspecified
userInterfaceIdiom	Unspecified
displayScale	Unspecified

– (BOOL)containsTraitsInCollection:(UITraitCollection *)trait;

Trait Collections

horizontalSizeClass	Compact	Contains ✗	horizontalSizeClass	Regular
verticalSizeClass	Regular		verticalSizeClass	Regular
userInterfaceIdiom	iPhone		userInterfaceIdiom	Unspecified
displayScale	2.0		displayScale	Unspecified

– (BOOL)containsTraitsInCollection:(UITraitCollection *)trait;

Trait Collections

horizontalSizeClass	Compact		horizontalSizeClass	Regular		horizontalSizeClass	Regular
verticalSizeClass	Unspecified		verticalSizeClass	Compact		verticalSizeClass	Compact
userInterfaceIdiom	iPhone	+	userInterfaceIdiom	Unspecified	=	userInterfaceIdiom	iPhone
displayScale	Unspecified		displayScale	Unspecified		displayScale	Unspecified

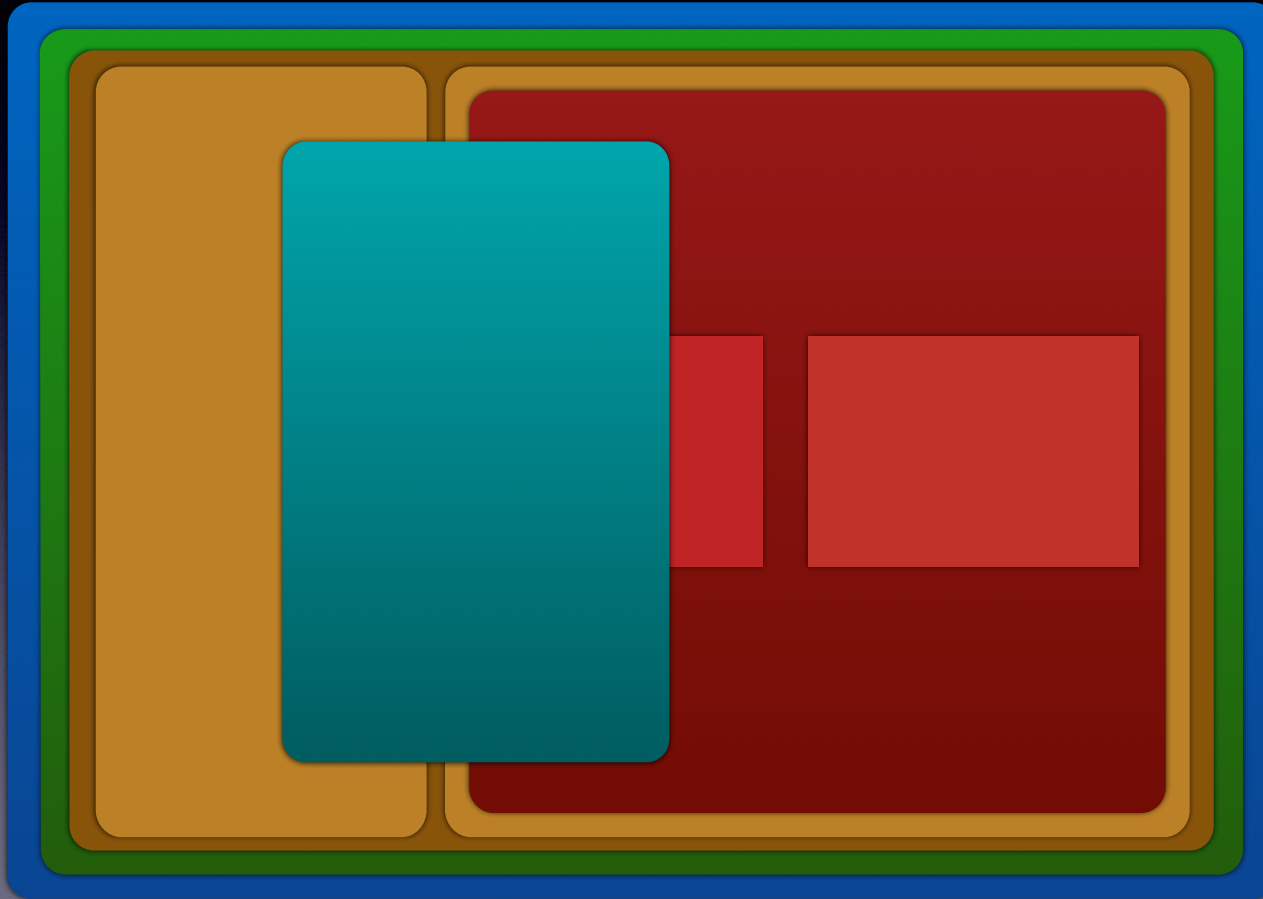
```
+ (UITraitCollection *)traitCollectionWithTraitsFromCollections:(NSArray *)traitCollections;
```


Adaptive Apps haben:

- Size Classes
- Traits
- Trait Collections
- Trait Environment

Trait Environment

Trait Environment



UIScreen

UIWindow

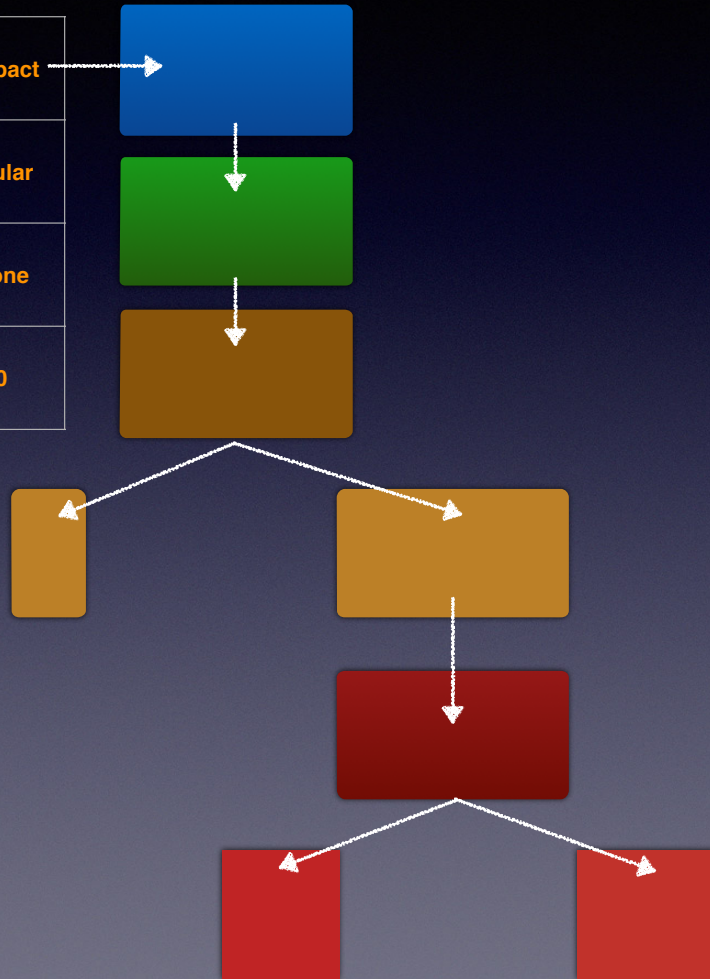
UIViewController

UIView

UIPresentationController

Trait Environment

horizontalSizeClass	Compact
verticalSizeClass	Regular
userInterfaceIdiom	iPhone
displayScale	2.0



UIScreen

UIWindow

UIViewController

UIView

Änderungen im Trait Environment

`-willTransitionToTraitCollection: withTransitionCoordinator:`

UIKitContentContainer:

Benachrichtigt den Container, dass sich die Trait Collection geändert hat. UIKit ruft die Methode bevor die Traits des aktuellen Objektes geändert werden. Auswirkungen auf alle UIViewController und UIViews. Methode kann verwendet werden um Interface basierende Aktionen auszuführen.

UIScreen

UIWindow

UIViewController

UIView

`-viewWillTransitionToSize: withTransitionCoordinator:`

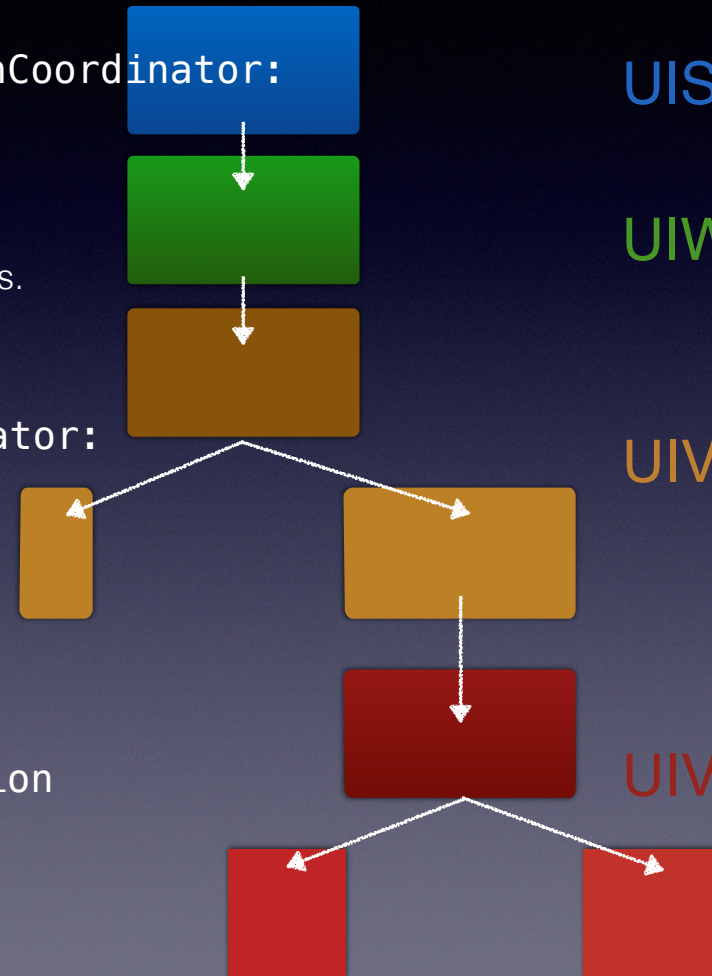
UIKitContentContainer:

Benachrichtigt den Container, dass sich die Größe (size) des Views geändert hat. UIKit ruft die Methode bevor die Größe des Views im präsentierenden Controller geändert wird. Es können abhängige Aktionen zur Größenänderung ausgeführt werden.

`-traitCollectionDidChange: previousTraitCollection`

UITraitEnvironment Protocol:

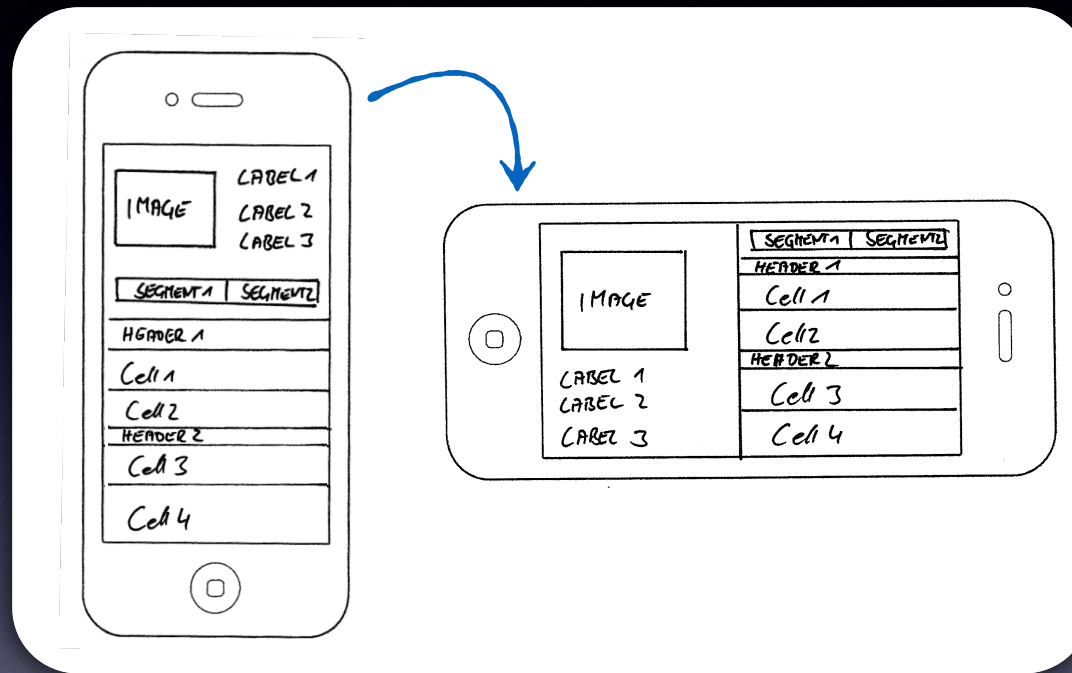
Die Stelle um eigenes Verhalten beim Wechsel einer Trait Collection durchzuführen.



**What does it mean
to us in reality?**

Gerd Oswald

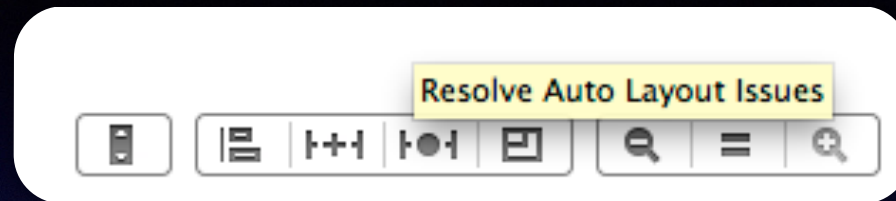
Ein altes Beispiel



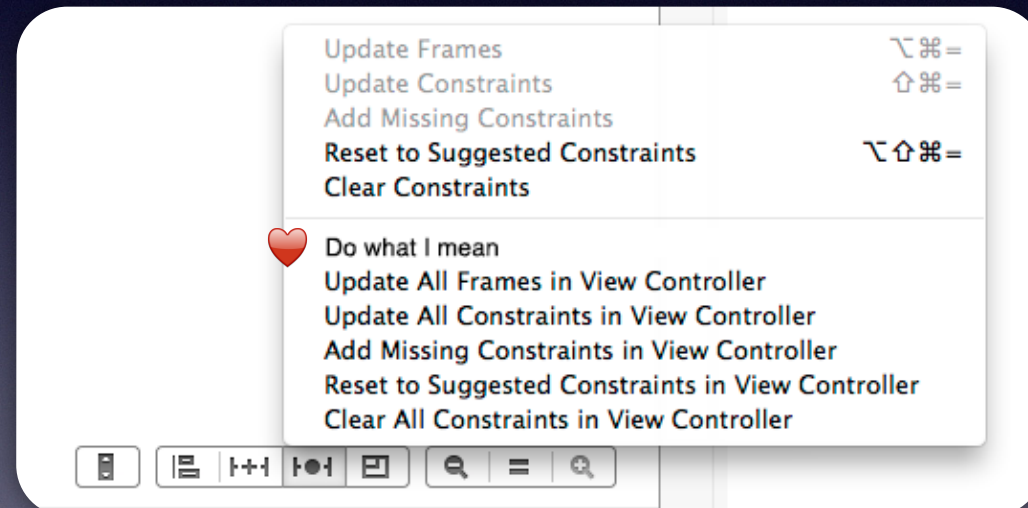
Unterschiedliche Layouts für Porträt und Landscape
heute aber zusätzlich als Universal App.

Die Lösung von damals

1.



2.



... funktioniert immer noch nicht!

Die (echte) Lösung von damals

- Constraints für eine Ausrichtung vollständig in IB
- alle Constraints, die NUR für diese Ausrichtung gelten und NICHT für die andere in eine IBOutletCollection
- in `-updateConstraints` die Constraints für die andere Ausrichtung einmalig erstellen
- dann je nach aktueller Ausrichtung die richtigen Constraints installieren

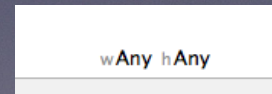
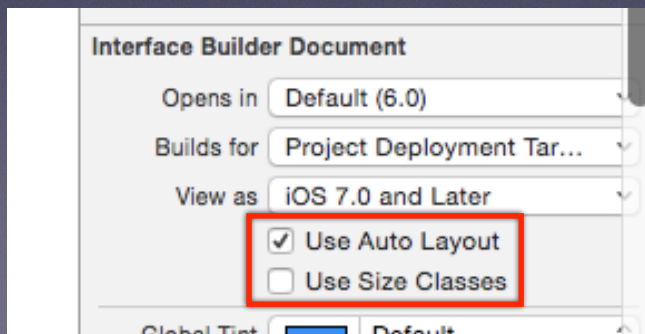
Demo

Die Lösung mit iOS 8

Die alte Lösung aus dem alten Projekt funktioniert wie vorher.

Aber mit der Einschränkung, dass in iOS 8 die Status Bar im Landscape Modus per Default ausgeblendet wird!

Size Classes einschalten:



**Achtung: Ab jetzt kein
Xcode 5.x
Support mehr!**

Die neuen APIs

```
UIInterface.h:
typedef NS_ENUM(NSInteger, UIUserInterfaceSizeClass) {
    UIUserInterfaceSizeClassUnspecified = 0,
    UIUserInterfaceSizeClassCompact     = 1,
    UIUserInterfaceSizeClassRegular     = 2,
} NS_ENUM_AVAILABLE_IOS(8_0);
```

```
UIDevice.h
typedef NS_ENUM(NSInteger, UIUserInterfaceIdiom) {
    UIUserInterfaceIdiomUnspecified = -1,
    #if __IPHONE_3_2 <= __IPHONE_OS_VERSION_MAX_ALLOWED
        UIUserInterfaceIdiomPhone,           // iPhone and iPod touch style UI
        UIUserInterfaceIdiomPad,             // iPad style UI
    #endif
```

Immer wenn sich die Trait Collection ändert möchte ich die Inhalte angezeigt bekommen.

```
-(void)traitCollectionDidChange:(UITraitCollection *)previousTraitCollection
```

Constraints sollen angepasst werden, bevor sich die TraitCollection ändert.

```
-(void)willTransitionToTraitCollection:(UITraitCollection *)newCollection
withTransitionCoordinator:(id<UIViewControllerTransitionCoordinator>)coordinator {

    [super willTransitionToTraitCollection:newCollection withTransitionCoordinator:coordinator];

    [coordinator animateAlongsideTransition:^(id <UIViewControllerTransitionCoordinatorContext> context) {
        [self updateConstraintsForTraitCollection:newCollection];
        [self.view setNeedsLayout];
    } completion:nil];
}

-(void)updateConstraintsForTraitCollection:(UITraitCollection *)collection {
    ...
    if (collection.verticalSizeClass == UIUserInterfaceSizeClassCompact) {
        ...
    }
}
```

Demo

Toll ich kann das selbe machen wie vorher ... nur anders!

Ich will das gleiche Verhalten jetzt auch auf dem iPad haben!

```
-(void)setTraitCollectionForSize:(CGSize)size {  
    if (self.view.traitCollection.userInterfaceIdiom == UIUserInterfaceIdiomPad) {  
        // determine new Size Class  
        UIUserInterfaceSizeClass newSizeClass = (size.width > 768.0 ? UIUserInterfaceSizeClassCompact : UIUserInterfaceSizeClassRegular);  
        // create a new Trait Collection  
        UITraitCollection *newTraitCollection = [UITraitCollection traitCollectionWithVerticalSizeClass:newSizeClass];  
        // assign new Trait Collection to child view = self  
        [self.navigationController setOverrideTraitCollection:newTraitCollection forChildViewController:self];  
    }  
}  
  
-(void)viewWillTransitionToSize:(CGSize)size withTransitionCoordinator:(id<UIViewControllerTransitionCoordinator>)coordinator {  
    // if the interface orientation change set the new trait collection for size  
    [self setTraitCollectionForSize:size];  
}  
  
-(void)viewDidLoad {  
    [super viewDidLoad];  
    // initial set the trait collection for current interface orientation  
    [self setTraitCollectionForSize:self.view.frame.size];  
}
```

Demo

Aber etwas stimmt noch nicht!

- Alle Constraints entfernt welche den grünen und orangenen View beeinflussen
- Alle Constraints entfernt welche den Image View und Label Container View beeinflussen
- IBOutletCollection entfernt
- alle Constraints in updateConstraintsForTraitCollection: für Portrait und Landscape eingefügt

Demo

Ergebnis:

Immer noch zu kompliziert und viel zu viel Coding!

Was bringt mir der Interface Builder?

1. Default Layout bauen (w: Any h: Any)
2. Layout für w: Compact h: Compact anpassen
3. Coding aus ALPViewController löschen

```
@implementation ALPViewController

#pragma mark - Old Methods

-(NSInteger)supportedInterfaceOrientations {
    return UIInterfaceOrientationMaskAll;
}

-(void)viewDidLoad {
    [super viewDidLoad];

    // initial set the trait collection for current interface orientation
    [self setTraitCollectionForSize:self.view.frame.size];
}

#pragma mark - New Methods

-(void)willTransitionToTraitCollection:(UITraitCollection *)newCollection withTransitionCoordinator:(id<UIViewControllerTransitionCoordinator>)coordinator {
    [super willTransitionToTraitCollection:newCollection withTransitionCoordinator:coordinator];

    [coordinator animateAlongsideTransition:^(id<UIViewControllerTransitionCoordinatorContext> context) {
        [self.view setNeedsLayout];
    } completion:nil];
}

-(void)viewWillTransitionToSize:(CGSize)size withTransitionCoordinator:(id<UIViewControllerTransitionCoordinator>)coordinator {
    // if the interface orientation change set the new trait collection for size
    [self setTraitCollectionForSize:size];
}

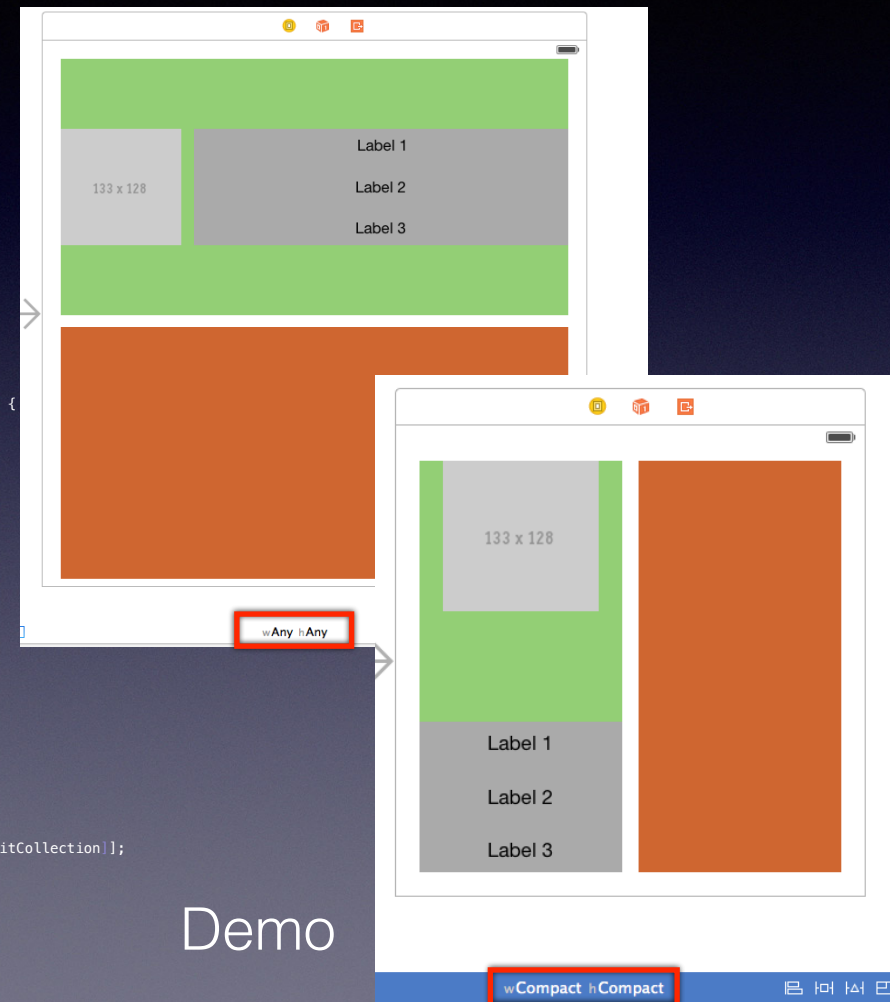
#pragma mark - My Private Methods

-(void)setTraitCollectionForSize:(CGSize)size {
    if (self.view.traitCollection.userInterfaceIdiom == UIUserInterfaceIdiomPad) {
        // determine new Size Class
        UIUserInterfaceSizeClass newSizeClass = (size.width > 768.0 ? UIUserInterfaceSizeClassCompact : UIUserInterfaceSizeClassRegular);

        // create a new Trait Collection
        UITraitCollection *newTraitCollection = [UITraitCollection traitCollectionWithVerticalSizeClass:newSizeClass];
        UITraitCollection *newVerticalTraitCollection = [UITraitCollection traitCollectionWithVerticalSizeClass:newSizeClass];
        UITraitCollection *newHorizontalTraitCollection = [UITraitCollection traitCollectionWithHorizontalSizeClass:newSizeClass];
        UITraitCollection *newTraitCollection = [UITraitCollection traitCollectionWithTraitsFromCollections:[newHorizontalTraitCollection, newVerticalTraitCollection]];

        // assign new trait collection to child view = self
        [self.navigationController setOverrideTraitCollection:newTraitCollection forChildViewController:self];
    }
}

@end
```



Demo

Sehenswert

WWDC 2014:

- 216 - Building Adaptive Apps with UIKit
- 411 - Whats New in Interface Builder
- 221 - Creating Custom iOS User Interfaces

Danke!