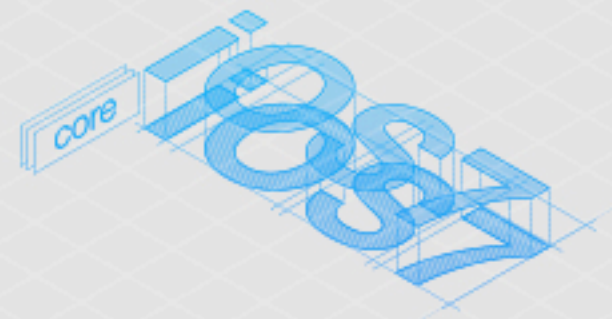


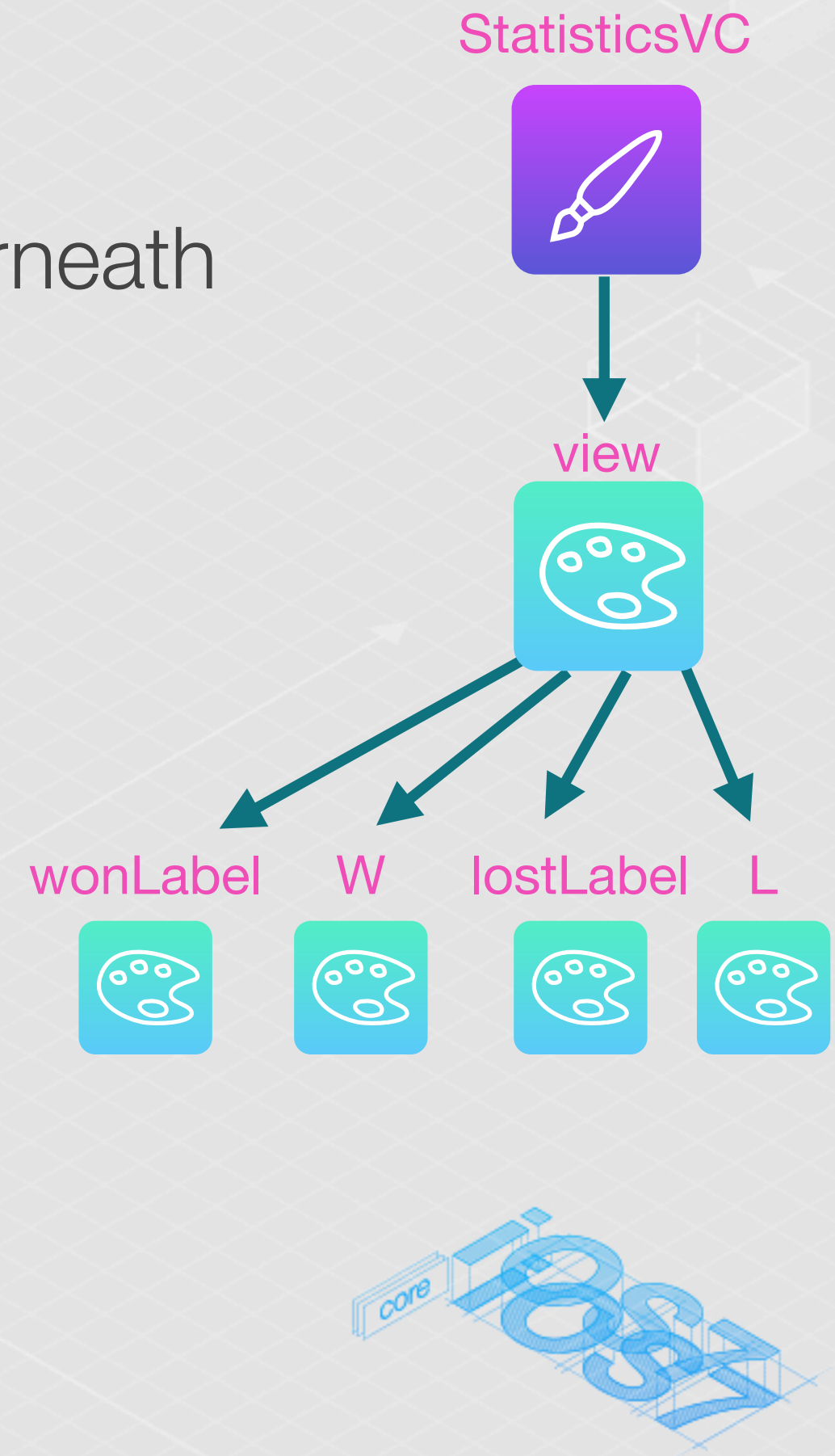
# Always ask yourself these four questions before getting started

1. Does my screen show a status bar?
2. Does my screen show a navigation bar?
3. Does my screen show a tab bar and a navigation bar?
4. Is my main view a scroll view (or a subclass like a table/collection view)?



# Laying out content to avoid the status bar

If you have content that used to sit directly underneath the status bar in iOS 6, it might interfere with the status bar content in iOS 7



# What happened in iOS 6

6



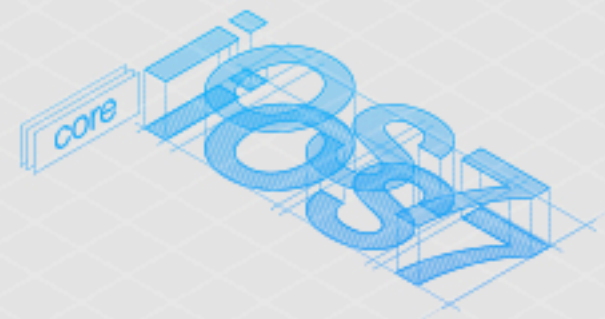
## ViewController.m

```
- (void)loadView
{
    UIView *view = [[UIView alloc] init];
    view.backgroundColor = [UIColor greenColor];
    self.view = view;
}
```

**self.view** height = **screen** height - **status bar** height

548pts high

automatically set to the  
full available width and  
height





# What happens now in iOS 7

7

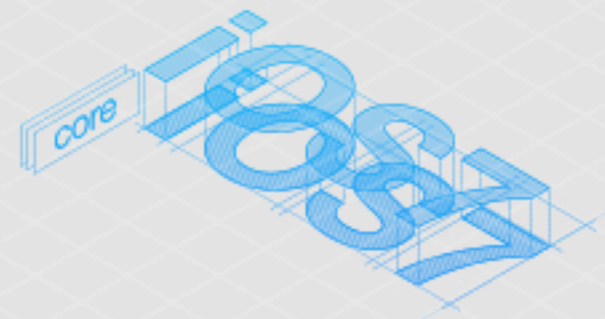


## ViewController.m

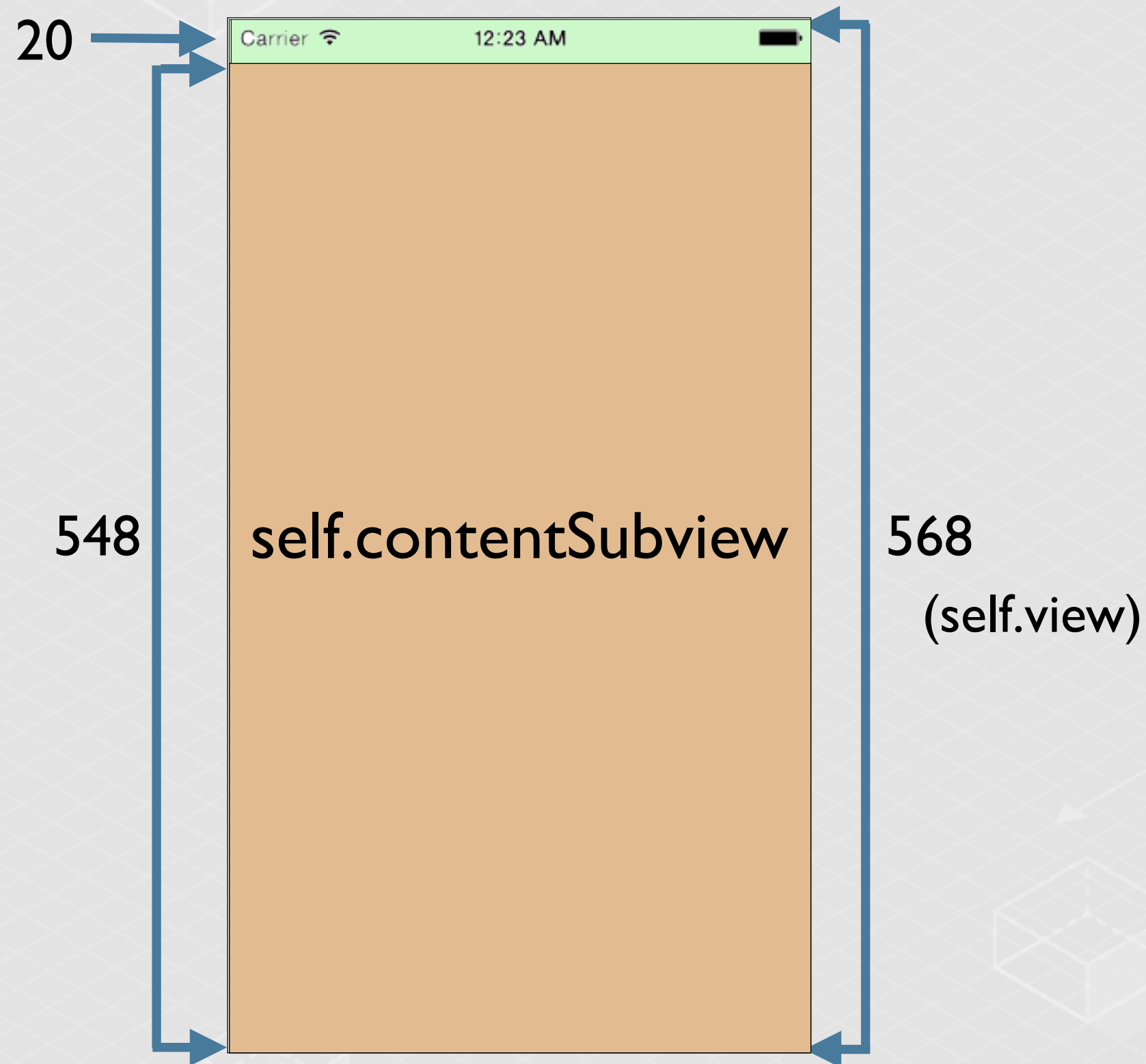
```
- (void)loadView
{
    UIView *view = [[UIView alloc] init];
    view.backgroundColor = [UIColor greenColor];
    self.view = view;
}
```

568pts high

What if you want  
`self.view` to sit  
below the status  
bar?



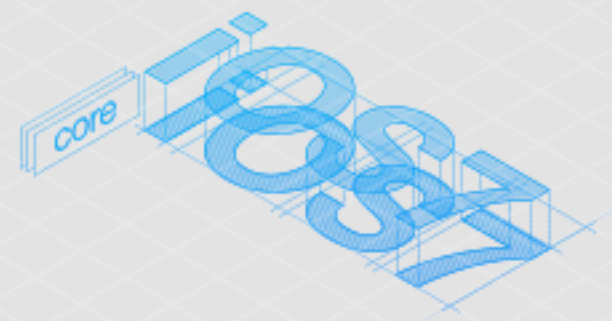
# Putting content in a subview container



## Create a contentView

Add all subviews to the contentView

Set the contentView frame to start below the status bar frame



# Creating a subview container to hold content

## ViewController.h


```
@property (strong, nonatomic) UIView *contentSubview;
```

## ViewController.m

```
- (void)loadView
{
    UIView *view = [[UIView alloc] init];
    view.backgroundColor = [UIColor greenColor];

    self.contentSubview = [[UIView alloc] init];
    self.contentSubview.backgroundColor = [UIColor orangeColor];
    [view addSubview:self.contentSubview];

    self.view = view;
}
```



where do we set the dimensions of contentSubview?

# Moving contentSubview below the status bar area

ViewController.m

set them here instead of viewDidLoad

```
- (void)viewWillLayoutSubviews
{
    [super viewWillLayoutSubviews];

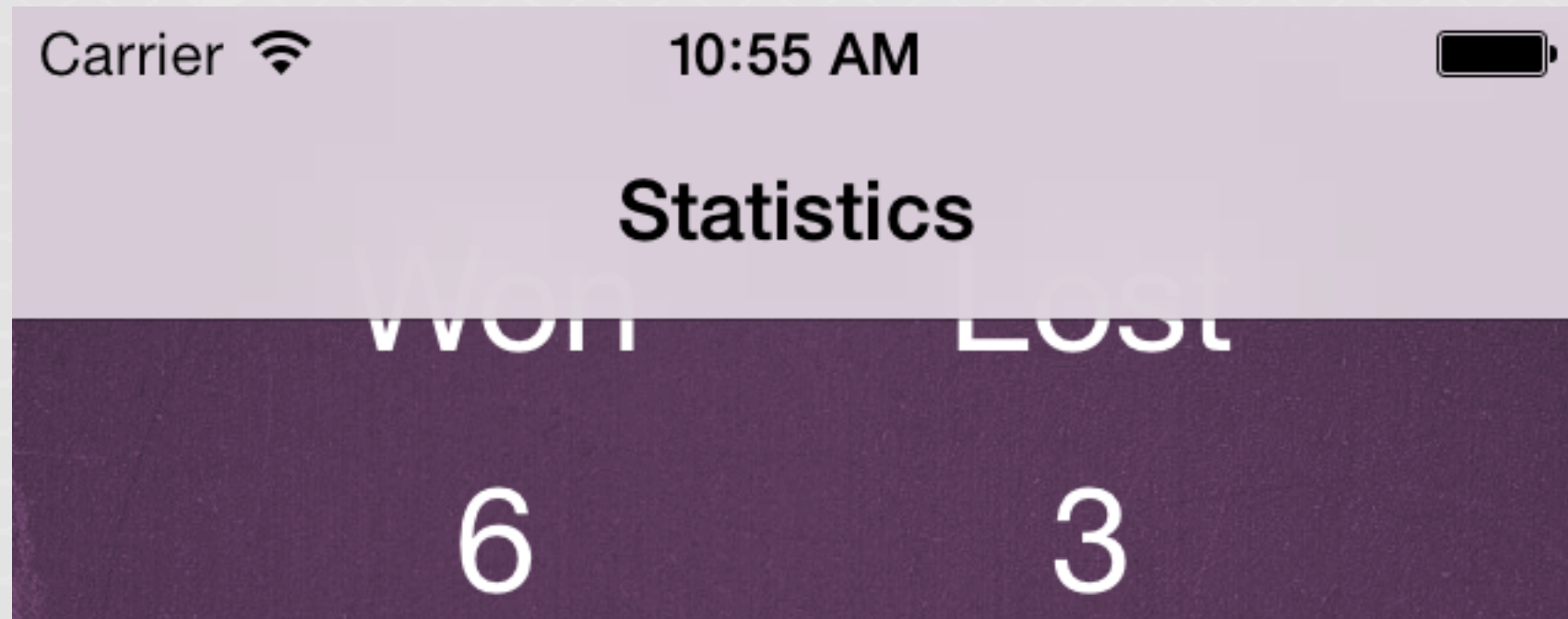
    self.contentSubview.frame = CGRectMake(
        0,
        20,
        CGRectGetWidth(self.view.frame),
        CGRectGetHeight(self.view.frame) - 20
    );
}
```



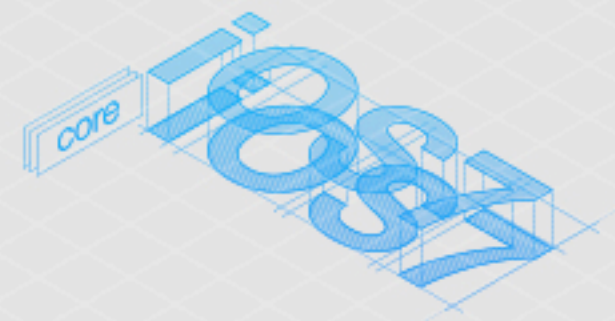
You can't be sure the status bar area will always be 20 pts high



Sometimes the status bar area will be bigger than 20 pts



We need a way to detect this height



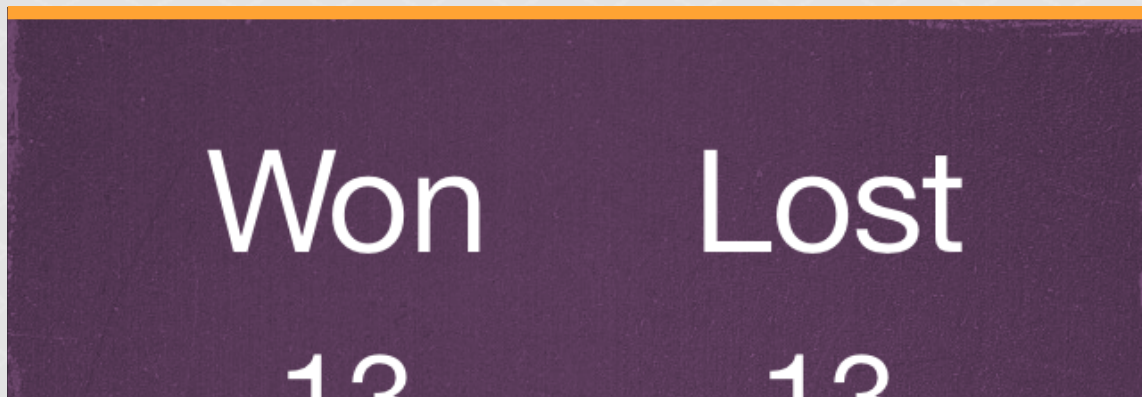


# Use the new `topLayoutGuide` property

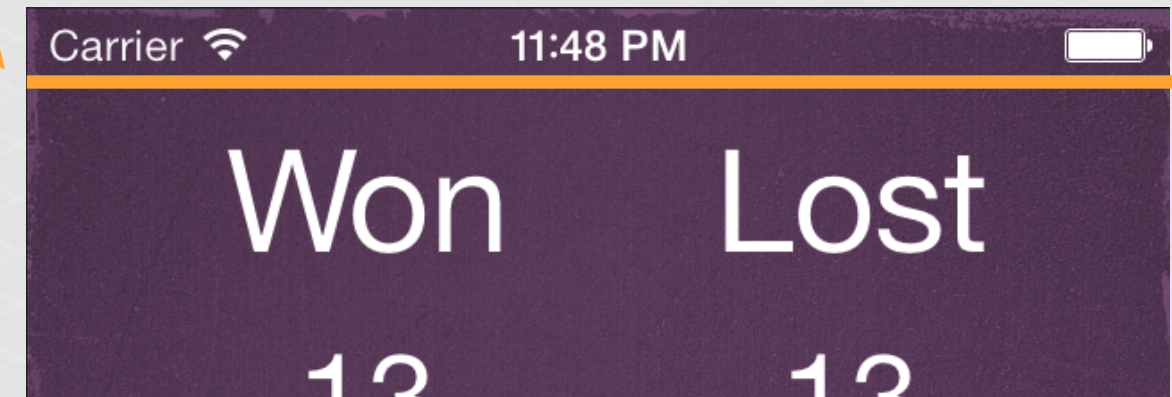
`topLayoutGuide` is a new `UIViewController` property that allows you to get the size of the status bar area to account for it in your view layout.

**`topLayoutGuide`**

if no status bar



if status bar



if no status bar

```
self.topLayoutGuide.length == 0
```

if status bar

```
self.topLayoutGuide.length == 20
```

# Using topLayoutGuide to put the contentSubview underneath the status bar



ViewController.m

```
- (void)viewWillLayoutSubviews ← self.topLayoutGuide isn't set until  
{                                right before this method is called  
    [super viewWillLayoutSubviews];  
  
    self.contentSubview.frame = CGRectMake(  
        0,  
        self.topLayoutGuide.length,  
        CGRectGetWidth(self.view.frame),  
        CGRectGetHeight(self.view.frame) - self.topLayoutGuide.length  
    );  
}
```

# Dimensions of contentView without a status bar in portrait mode

if no status bar



origin.x

0,

origin.y

`self.topLayoutGuide.length`  
0

width

`CGRectGetWidth(self.view.frame),`  
320

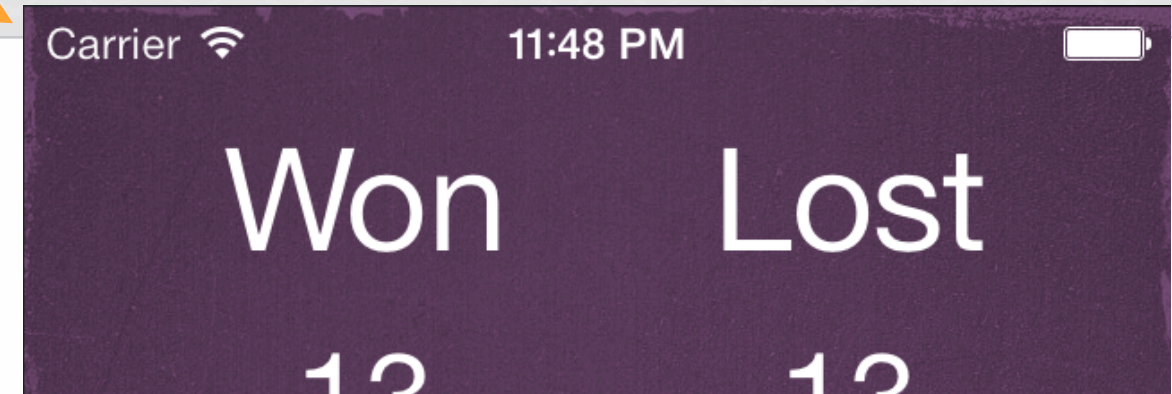
height

`CGRectGetHeight(self.view.frame) - self.topLayoutGuide.length`  
568 - 0 = **568** (full screen)



# Dimensions of contentView with a status bar in portrait mode

if status bar



origin.x

0,

origin.y

`self.topLayoutGuide.length`  
20

width

`CGRectGetWidth(self.view.frame),`  
320

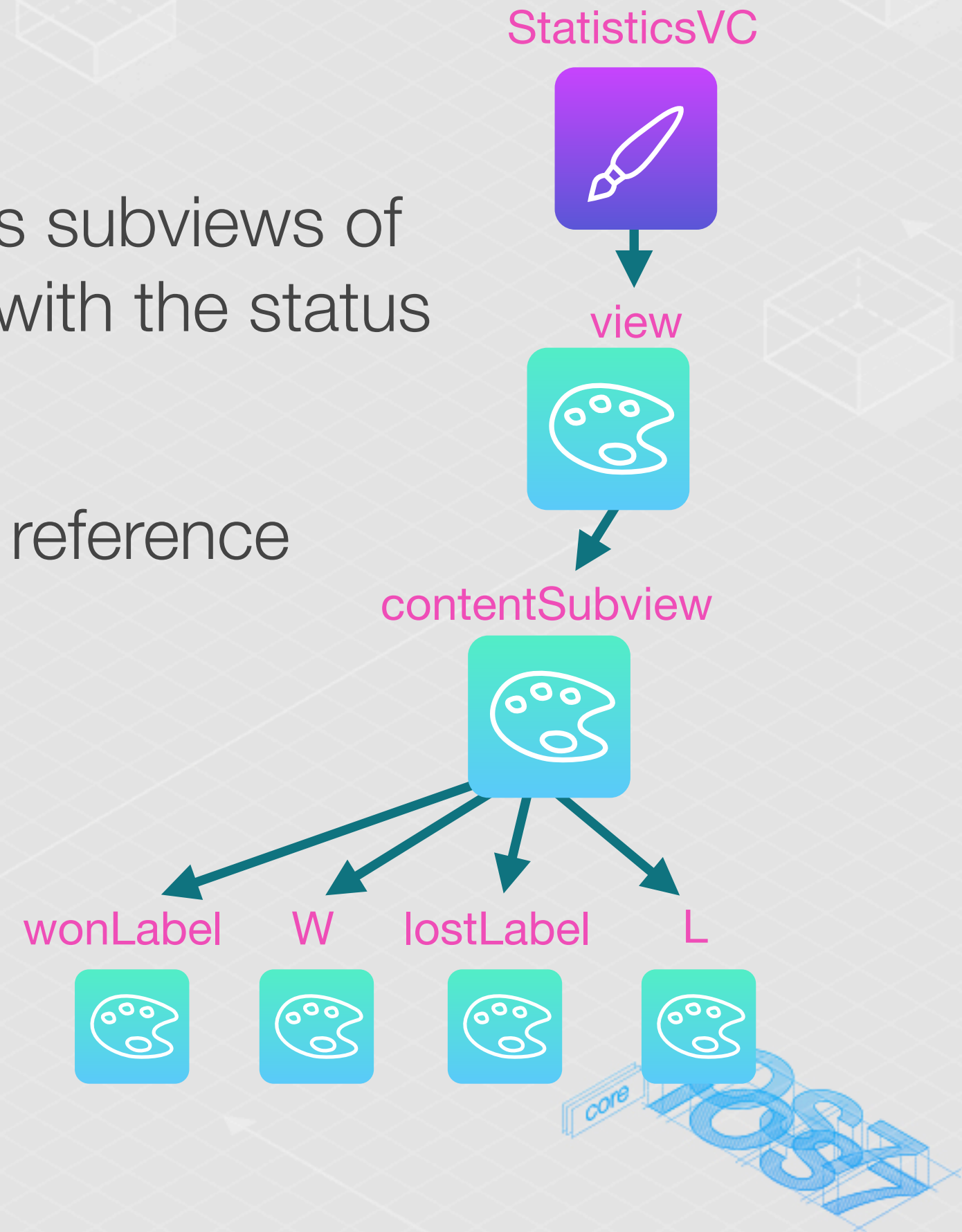
height

`CGRectGetHeight(self.view.frame) - self.topLayoutGuide.length`  
 $568 - 20 = \mathbf{548}$  (almost full screen, just under bar)

# Layout the rest of the subviews

The rest of the labels can now be added as subviews of contentSubview without fear of interfering with the status bar area

Use the **contentSubview frame** as a top reference for subviews instead of topLayoutGuide

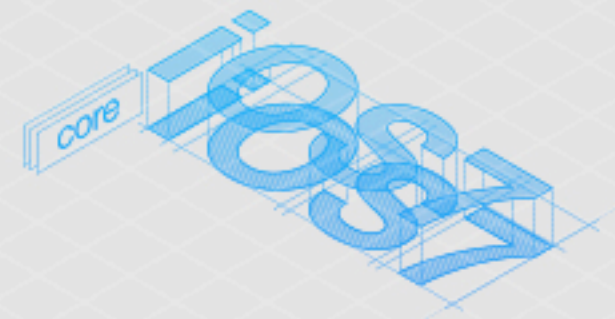


# Adding the rest of the subviews to contentView

ViewController.m

```
- (void)loadView ← add the subviews in loadView
{
    ...
    self.wonLabel = [[UILabel alloc] init];
    [self.contentView addSubview:self.wonLabel];
    ...
}
```

add subviews to the contentView,  
not self.view





# The subview frames are relative to the contentView

## ViewController.m

```
- (void)viewWillLayoutSubviews
{
    [super viewWillLayoutSubviews];

    self.wonLabel.frame = CGRectMake(
        0,
        30, ←
        120,
        30
    );
}
```

contentSubview



wonLabel



relative to the superview (contentSubview)  
means “start 30 pts away from  
the superview edge”

