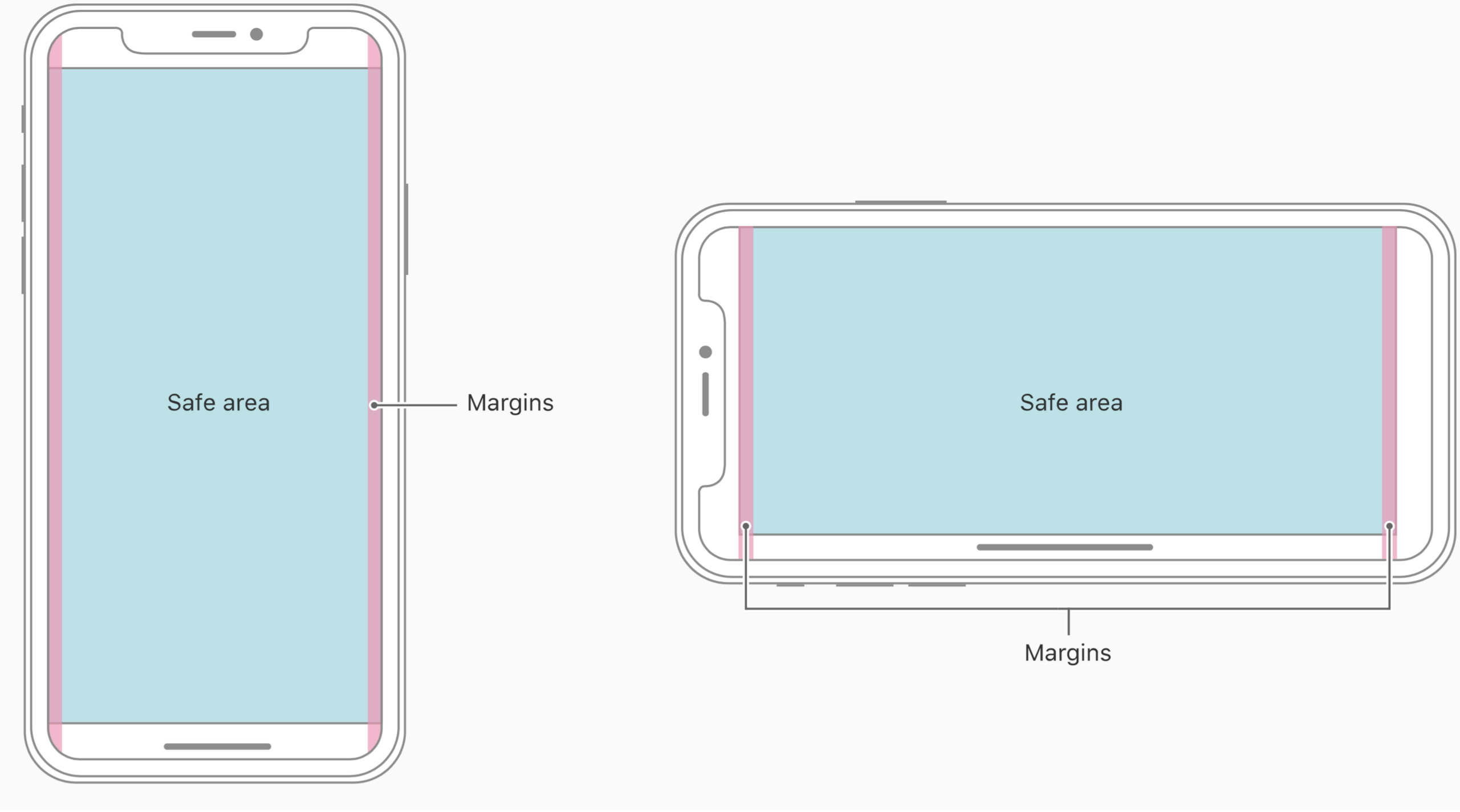
Safe Area Insets

The Truth. The Way. The Light.



If you *only* watched WWDC videos, all you'd know is that "safe areas are a new way to layout your content".

The *real* reason is they are basically the only hope we have of avoiding those fancy rounded corners and notches.

Generally, they mean "here is a place we think is a good place to put your stuff"

Ignoring for a second the weird corners and notches, I think safe areas are a simplification/extension of the existing APIs.

Pre-iOS 11
UIViewController

```
var topLayoutGuide: UILayoutSupport
var bottomLayoutGuide: UILayoutSupport
```

We've had layout guides, which was suppose to let you avoid UI chrome, navigation and tab bars. They helped with navigation bars changing sizes in landscape

But they were only available on view controllers, which is limiting. They are also static, clunky to get values out of (if you weren't using them as anchors), and hard to customize/extend

You could add you own guides, but then you had to remember to use them

UIView

```
var safeAreaInsets: UIEdgeInsets
var safeAreaLayoutGuide: UILayoutGuide
func safeAreaInsetsDidChange()

var insetsLayoutMarginsFromSafeArea: Bool
```

UIViewController

```
var additionalSafeAreaInsets: UIEdgeInsets
```

This is UIView-based, not UIViewController-based which I think is an interesting change from UIKit APIs in the past.

They also *update* throughout a view's lifecycle, which is an important distinction from the past

Insets are values, that are always up to date. Guides are for layout constraints and also provide anchors

The one nod to controllers is the ability for the controller to give a bit extra insets to its root view. Seems like only positive values have any affect.

Safe areas propogate down, so every view has sane/valid safe areas for its frame of reference.

Blue represents the current safe areas in this view controller

We can optionall add more additional areas

The view has a label (that prints out current safe areas insets) tied to constraints, we can toggle if margins respect safe areas

The view itself isn't tied to any safe areas, so I can move it freely. However, the safe areas are being updated in the view as it moves

and if insetsLayoutMarginsFromSafeArea is true, they are adjusted as well

Don't forget, instead of margins, you can also use safe areas layout guides

UIScrollView
and UITableView
and UICollectionView

Things are a little different for UIScrollView

Pre-iOS 11
UIViewController

```
var automaticallyAdjustsScrollViewInsets: Bool
```

iOS 7 was a palet swap and a whole bunch of hacks. One of them was this.

```
init              -> contentInset = {0, 0, 0, 0}
viewDidLoad       -> contentInset = {0, 0, 0, 0}
// Some hidden magic
viewWillAppear -> contentInset = {64, 0, 49, 0}
```

So, if a scroll view exists… *somewhere*, find it & just nudge it a bit

The rules were finding it were opaque, when this changes happened was opaque

```
init           -> contentInset = {0, 0, 0, 0}
viewDidLoad    -> contentInset = {0, 0, 0, 0}
viewWillAppear -> contentInset = {64, 0, 49, 0}
...
someFunc        -> contentInset = {64+10, 0, 49, 0}
```

If you wanted to change it in the future, you had to remember to use the current value, otherwise it would be lost

```
init           -> contentInset = {0, 0, 0, 0}
viewDidLoad    -> contentInset = {100, 0, 0, 0}
viewWillAppear -> contentInset = {64, 0, 49, 0}
```

of it you set it too early (say, to fit a fancy parallaxing header), it would just be blown away.

UIScrollView

```
var contentInsetAdjustmentBehavior:
  UIScrollViewContentInsetAdjustmentBehavior

.automatic,
.scrollableAxes,
.never,
.always
```

Now, this has been replaced with contentInsetAdjustmentBehavior

UIScrollView

```
var contentInsetAdjustmentBehavior:
  UIScrollViewContentInsetAdjustmentBehavior

.automatic,
.never
```

.automatic seems to be automaticallyAdjustsScrollViewInsets = true on steroids

.never is kind of automaticallyAdjustsScrollViewInsets = false but worse

Because it not just about avoiding navigation bars, there are other concerns now

This adjustment is based on the safe areas, so its all propogating down the right way, including any additional insets from a view controller

iOS 11

```
var contentInset: UIEdgeInsets //Completely ours
var adjustedContentInset: UIEdgeInsets // Ours + Safe Area Magic
```

This is what automaticallyAdjustsScrollViewInsets adjusts when contentInsetAdjustmentBehavior is set to something other then .never, adjustedContentInset will account for safe areas, keeping the intial "unscrolled" scroll away from it. Safe ares are *not* propogated into the subviews

I say "our", but that is kind of a lie. RefreshControlls still mess with it.

Never

.never*

After some experimenting, this is what I came with. .never is an easy hack, but things get weird/unexpected very quickly

Demo

Scroll view's behave a little different. Safe areas adjust content insets, unless .never, then safe areas are propogated to subviews

blue view that just ignores safe areas all together

Red view label uses safe areas (but not margins)

And, again, a box that prints out its current safe areas (tied to margins)

~~Best~~ Practices

- Never `.never`
- Think about each view, not necessarily a root view controller
- Read `adjustedContentInsets` not `contentInsets`
- Content (labels, buttons) should probably use `safeAreaLayoutGuides`
- Backgrounds should be to `frame`

Additional Resources

https://twitter.com/smileyborg/status/907723616137052160
https://developer.apple.com/videos/play/fall2017/201/
https://developer.apple.com/videos/play/fall2017/801/
https://useyourloaf.com/blog/safe-area-layout-guide/
https://useyourloaf.com/blog/supporting-iphone-x/

The first link is a UIKit engineer, basically describing to us the weird sticking behavior was not a bug, but working as indended. Without us telling him, he described our "bug" to a T.

Thanks Jen Kelley David Johnson

Thanks to Jen and David. We worked through all this over the course of week while trying to fix our fancy/custom navigation headers.

@donnellyk
Reverb