Notes Udemy iOS 13

Image assets

Starter Image bundles were provided in this tutorial.

Canva.com. she creates a 1024x1024 custom image in the main screen then pick some shapes, and gradient backgrounds to match the diamond background.

1. AppIcon.co 2 tabs, 1 for app icons. Like on your iPhone main screen
    1. 2 ‘Image sets for icons within a specific model (4s, 5, 8, X)
        1. Screens have the same screen sizes. but resolutions will be bigger as technology advances. Pixels will be so small/sharp they will almost disappear to the human eye.
    2. Finally, dragging all these photos with a contents.json will automatically sort your app icon sets, the downloaded files will mimic your app directory, so pretty self explanatory. You have to manually move it from folder to folder, and not plop it into Xcode like 3x,2x,1x Image set
        1. You can mark off android, and just use what platforms you are targeting.

She shows how you can change the the view from a different device in the layout editor in the bottom, ‘View as’, in this case this will only look good on a 11, as we didn’t set the constraints out the right way.

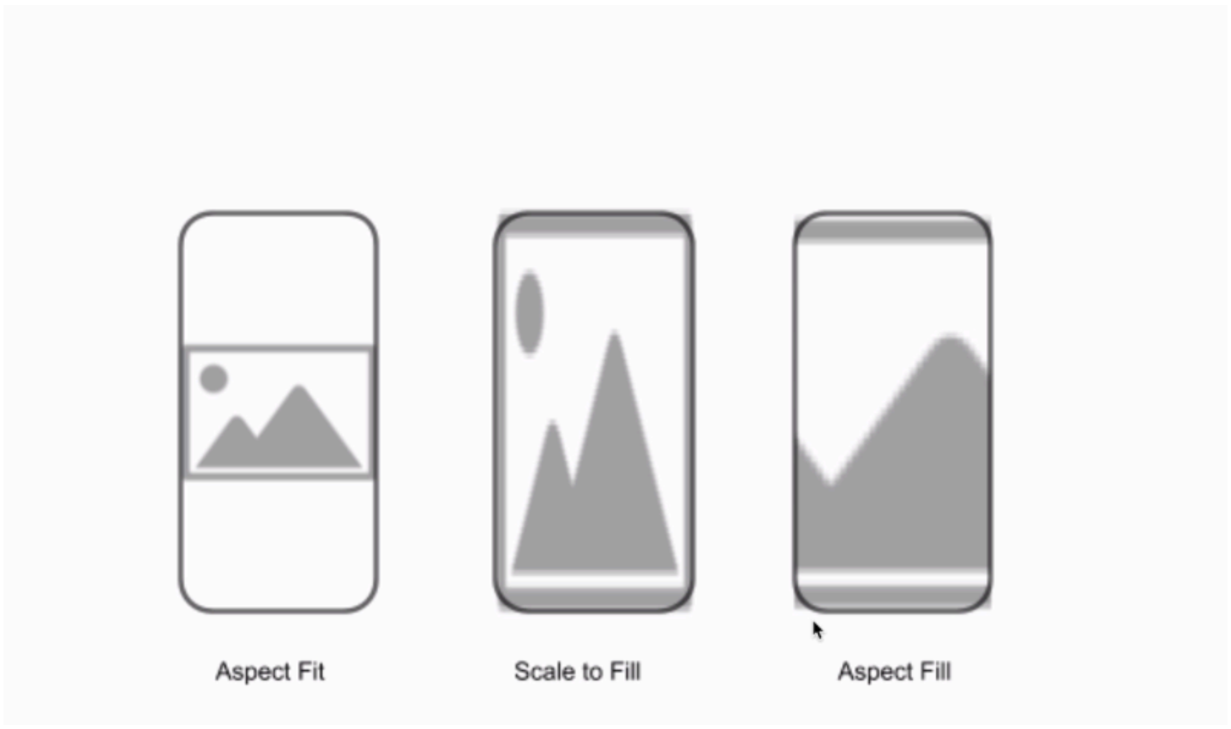
Before launch you MUST run your application on a Apple device, preferably iPhone

Dicee Game, Section 3

Close all Xcode instances!

There is a git clone feature in the top bar when Xcode is running, copy the cloning directory,(Big green button)

Below is the attribute Inspector’s way of stretching/filling image views, best is to either aspect fit or aspect fill,// as scale to fill
Is too pixelated, tho pixelation might be a good design choice



Holding the Option Key/alt+clicking+dragging image element will create a Complete copy of the previous element

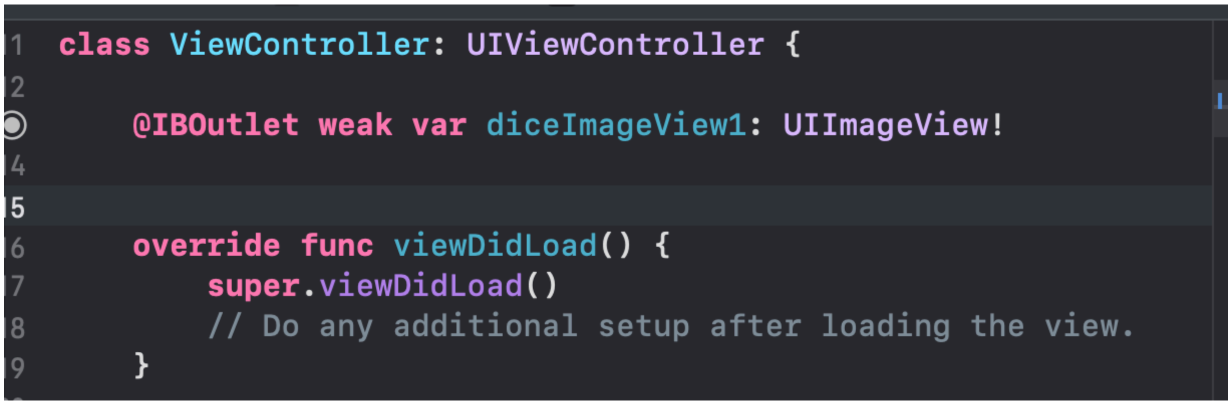
Swapping images is as simple as selecting image, and changing it in this example it was DICE 1 to DICE TWO

In case a user on the other side of the world is using your app, we cannot change dice one to dice two. Which is why
We will use code for dices to swap.
—>I thought she meant live updates, but really just front end code.

viewController.Swift is like your JS file, while the storyboard is like an XML/HTML Front end/design apparatus all in one. Kinda
Amazing

If you go to the top right of your storyboard, or .Swift file you will see a editor option menu,
'The paragraph' icon, selecting 'Assistant will allow you to put your .swift and xml

next to each other.



```swift
class ViewController: UIViewController {

    @IBOutlet weak var diceImageView1: UIImageView!


    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
    }
```

Adding variable objects from your Design Builder/storyboard allows you to manipulate view objects using code

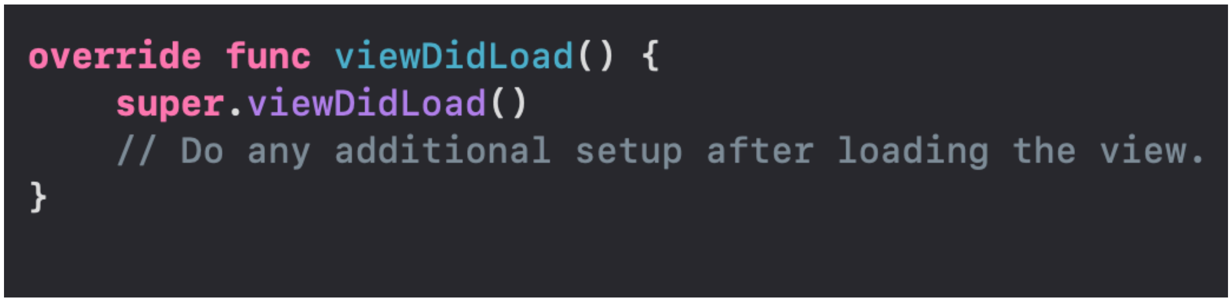
Simply holding the control key(Not command) and placing it in between a class declaration and Override Function allows you
To add a the variable automatically. Though, a manual way might be more beneficial in project with a larger scope.
Focusing on single page app for now. Basic camelCase rules apply here for variable declaration.

Right clicking the storyboard and clicking and opening as source code, gives us that classic HTML/XML look.

Changing the name of the variable in the storyboard will cause the app to crash, if it is linked with the wrong name.

Simply disconnecting(Right click element) and reconnecting via the .swift file will be enough to refresh it and reconnect

(+) there should be a plus line to the far left of the variable decleration in .swift



```swift
override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
}
```

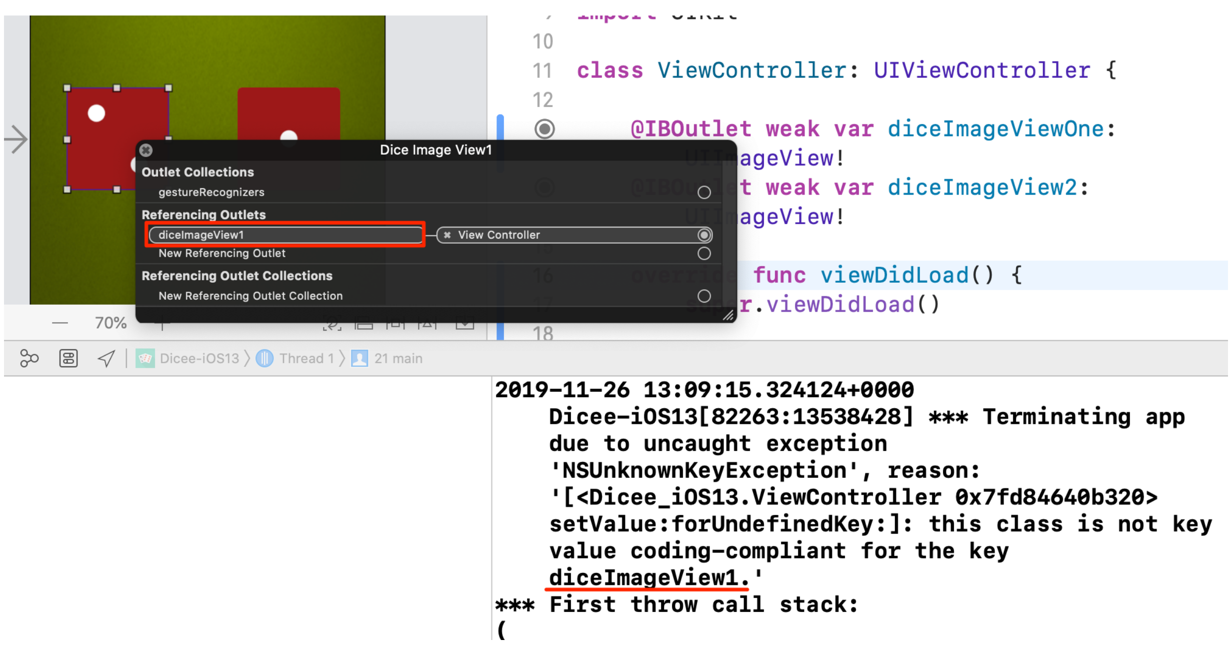
The function above executes, when the page is loaded, kinda like a jS on load
So far only two lines of code, that will be activated when the page is loaded, since the second line is just a comment
All that execute, is super.viewDidLoad()

She wants to change the 1 into 6, in order to change the 1 to a 6, we need to us a

who.what=Value
Element(Object).Property=(value=6)

Swift is case sensitive, when you see the error (Unresolved Identifier,) = basically I can't find the value

Compilation Error: "signal SIGABRT and "Not Key Value Coding Compliant"



```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var diceImageViewOne:
        UIImageView!
    @IBOutlet weak var diceImageView2:
        UIImageView!

    override func viewDidLoad() {
        super.viewDidLoad()
```

```
2019-11-26 13:09:15.324124+0000
    Dicee-iOS13[82263:13538428] *** Terminating app
    due to uncaught exception
    'NSUnknownKeyException', reason:
    '[<Dicee_iOS13.ViewController 0x7fd84640b320>
    setValue:forUndefinedKey:]: this class is not key
    value coding-compliant for the key
    diceImageView1.'
*** First throw call stack:
(
```

Just locate the conflicting line of code in the console/debugger and cancel the connection, at the very top. Usually its a
Connection issue.

Right click all the UI elements or the most suspected and cancel the connection, I would say to run and commit and run again, to keep your code things safe

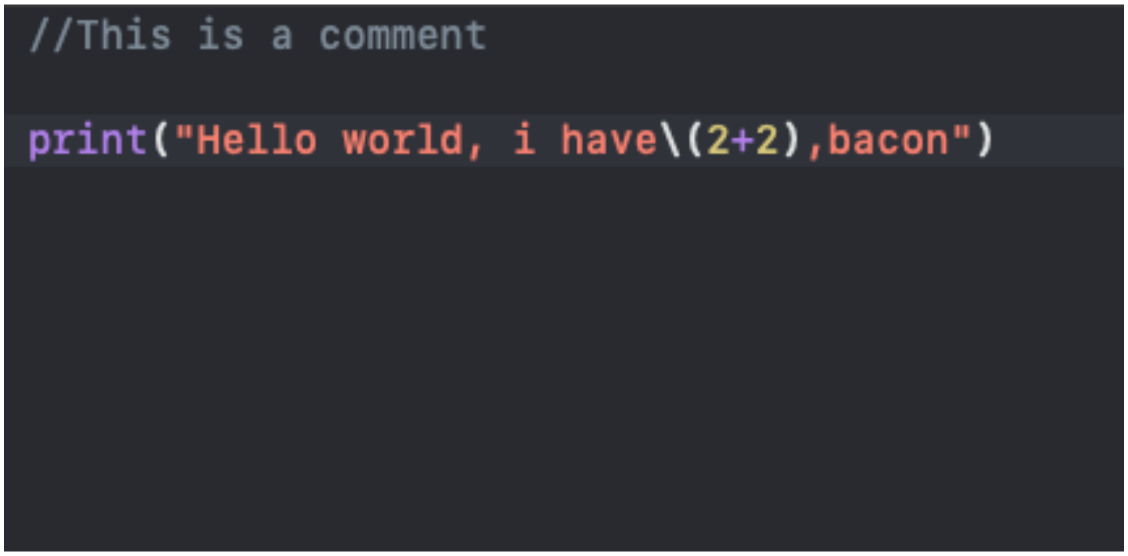
Command + Shift
Comments out the code

Mixing data types in a swift print statement, very similar to most languages

The Forward slash is key, as you are used to the lower backslash \, the comma is also required
Adding in a parenthesis with an isolated/different data type

Text/Letters= strings

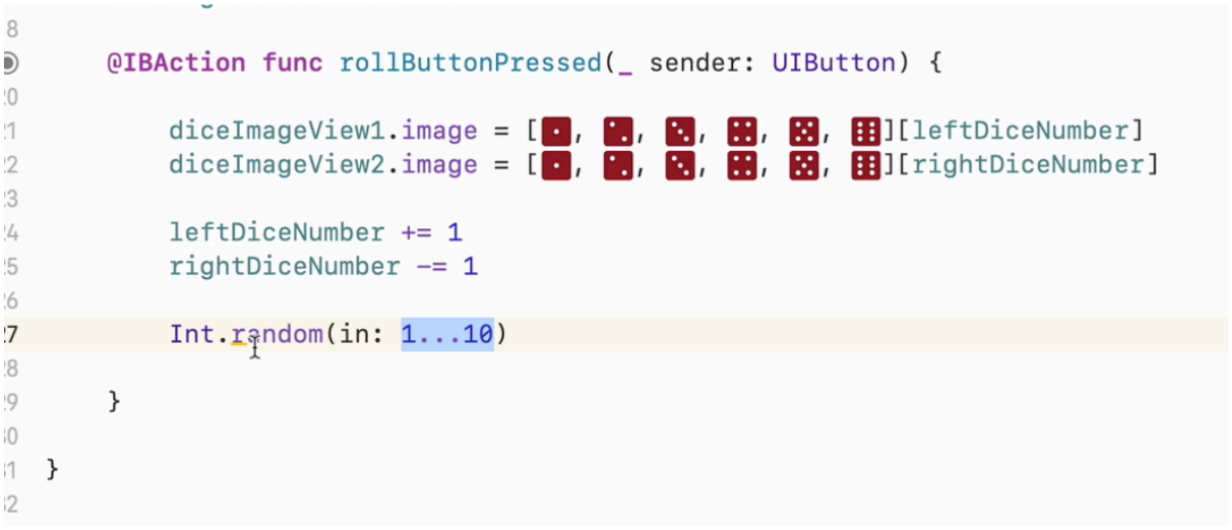
This is called string interpolation incase you forgot(You did), strin



```
//This is a comment

print("Hello world, i have\(2+2),bacon")
```

Went through some array syntax earlier, much of the same,

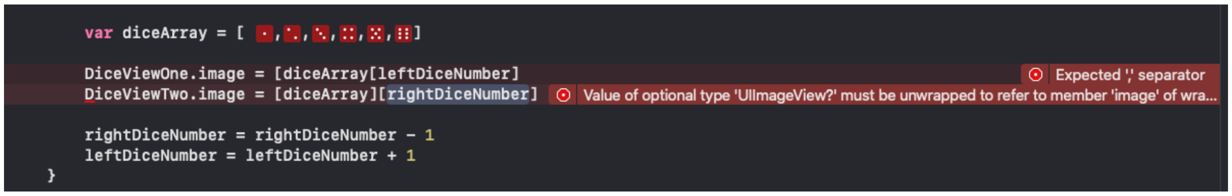
myVariable += 1 (myVariable=myVariable+1)



```
    @IBAction func rollButtonPressed(_ sender: UIButton) {

        diceImageView1.image = [⚀, ⚁, ⚂, ⚃, ⚄, ⚅][leftDiceNumber]
        diceImageView2.image = [⚀, ⚁, ⚂, ⚃, ⚄, ⚅][rightDiceNumber]

        leftDiceNumber += 1
        rightDiceNumber -= 1

        Int.random(in: 1...10)

    }

}
```

How to create a random number in a given range 1-10, I remember I couldn't even

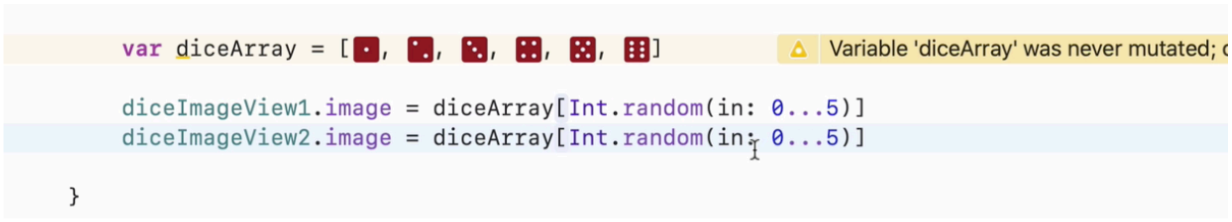
understand this
We haven't added it to the function yet, as we will go through syntax as well



```
    var diceArray = [ ⚀,⚁,⚂,⚃,⚄,⚅]

    DiceViewOne.image = [diceArray[leftDiceNumber]
    DiceViewTwo.image = [diceArray][rightDiceNumber]

    rightDiceNumber = rightDiceNumber - 1
    leftDiceNumber = leftDiceNumber + 1
}
```

Expected ',' separator
Value of optional type 'UIImageView?' must be unwrapped to refer to member 'image' of wra...

You can't just wrap the variable like an array, the correct way to do this is treat it like a normal object being used by the array

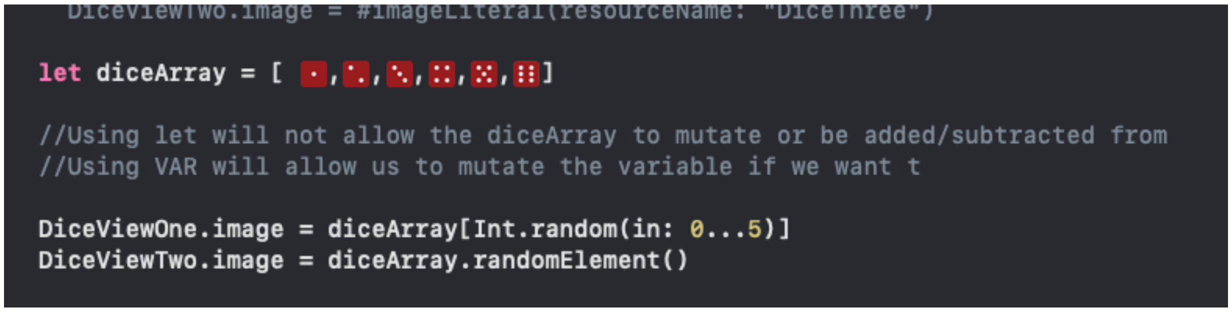


```
        var diceArray = [⚀, ⚁, ⚂, ⚃, ⚄, ⚅]

        diceImageView1.image = diceArray[Int.random(in: 0...5)]
        diceImageView2.image = diceArray[Int.random(in: 0...5)]

    }
```

Variable 'diceArray' was never mutated; c

This is a great Dry technique, and also happened to make our Array Randomly sort infinitely, so we kinda have true
Dice game.

'Let' is known as a constant



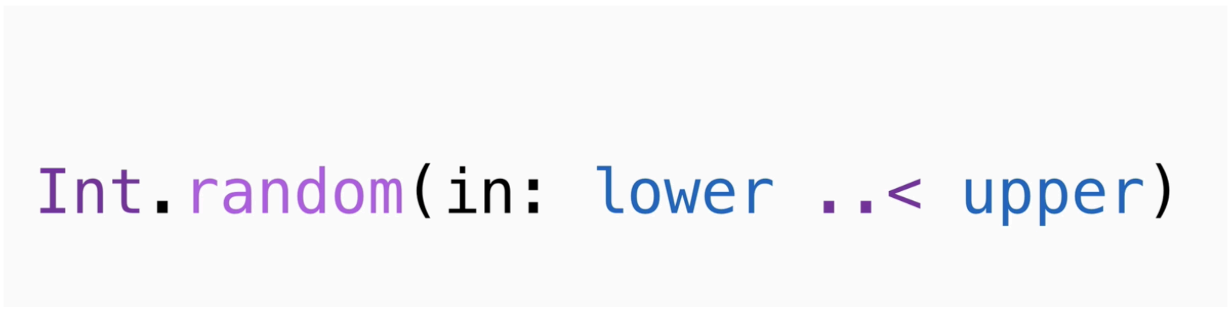
```
  DiceViewTwo.image = #imageLiteral(resourceName: "DiceThree")

let diceArray = [ ⚀,⚁,⚂,⚃,⚄,⚅]

//Using let will not allow the diceArray to mutate or be added/subtracted from
//Using VAR will allow us to mutate the variable if we want t

DiceViewOne.image = diceArray[Int.random(in: 0...5)]
DiceViewTwo.image = diceArray.randomElement()
```

There are two ways of getting random elements/integers from out app

Let a = 3

A =5

This will create a syntax error, as the constant cannot be changed. The content of the box will never change, the computer
Knows it will never change, less 'weight'

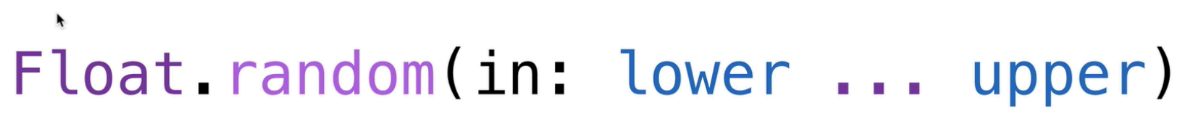


```
Int.random(in: lower ..< upper)
```

..< create a limit, doesn’t include the upper.

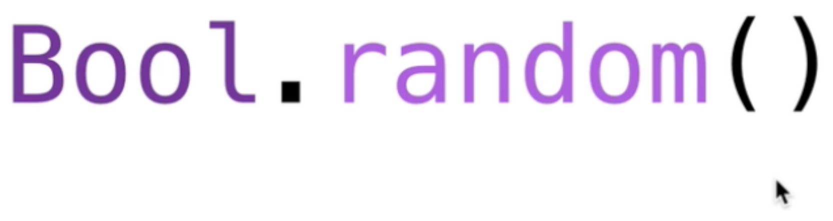
… while, the ‘…’ will include the upper. I can do a JS comparison, but its kinda the same <=, < vs ..<

Example of random with the use of Float



```
Float.random(in: lower ... upper)
```

Example of a Bool random, either

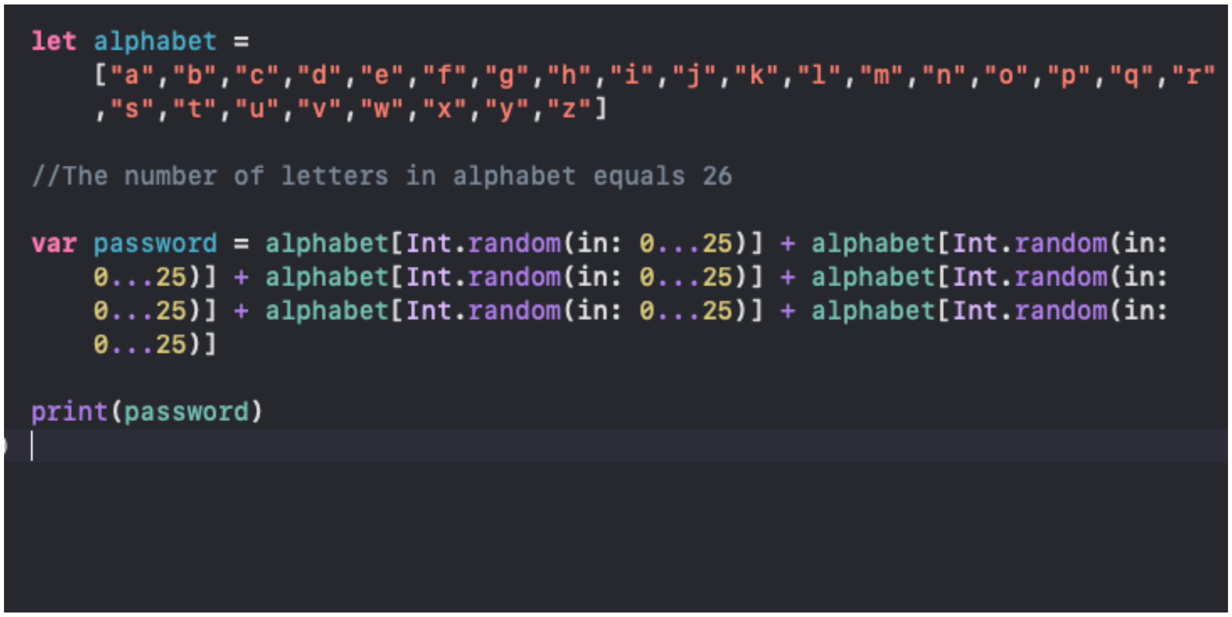


```
Bool.random()
```

Array.shuffle

array.shuffle()

Var bacon = [‘I like bacon’, ‘I eat bacon’, ‘we eat bacon’]
Array.shuffle shuffles the array positions ie([0,1,2].shuffle() = [1,2,0] etc
While the output will be like a [

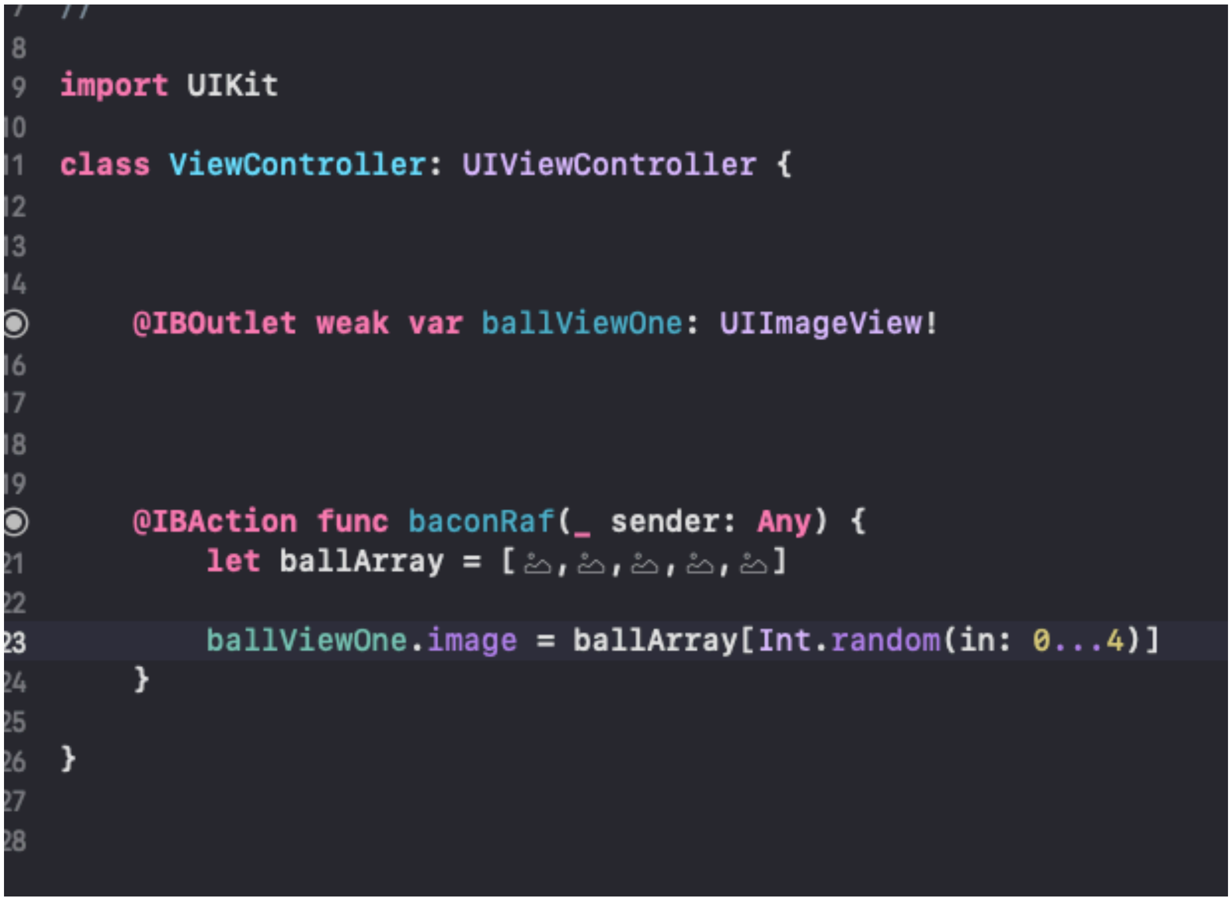


```
let alphabet =
    ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r"
    ,"s","t","u","v","w","x","y","z"]

//The number of letters in alphabet equals 26

var password = alphabet[Int.random(in: 0...25)] + alphabet[Int.random(in:
    0...25)] + alphabet[Int.random(in: 0...25)] + alphabet[Int.random(in:
    0...25)] + alphabet[Int.random(in: 0...25)] + alphabet[Int.random(in:
    0...25)]

print(password)
```

Here is the result of the password challenge the parameter were very clear and it was a nice challenge. I like the pacing so far
Instinctively I didn’t count how many character I counted, but met the quota.

I tried using array.shuffle, but forgot the let Constant is immutable(Can’t alter it)

The first password I printed is ‘qeyequ’ sounds like Spanish Que Que! Lol



```
//

import UIKit

class ViewController: UIViewController {


    @IBOutlet weak var ballViewOne: UIImageView!



    @IBAction func baconRaf(_ sender: Any) {
        let ballArray = [🏞,🏞,🏞,🏞,🏞]

        ballViewOne.image = ballArray[Int.random(in: 0...4)]
    }

}
```

This is the end result of my page, I had a connection issue for a action/image view I deleted that I couldn't find using the IDE
Its dangerous to restart an entire project because of it. But in this case it was doable.

If I could redo the entire thing I would probably just

This is the section in Layouts and Constraints

Constraints are set so that your app can scale to any apple device.

Once the constraints are set on a layout you can build

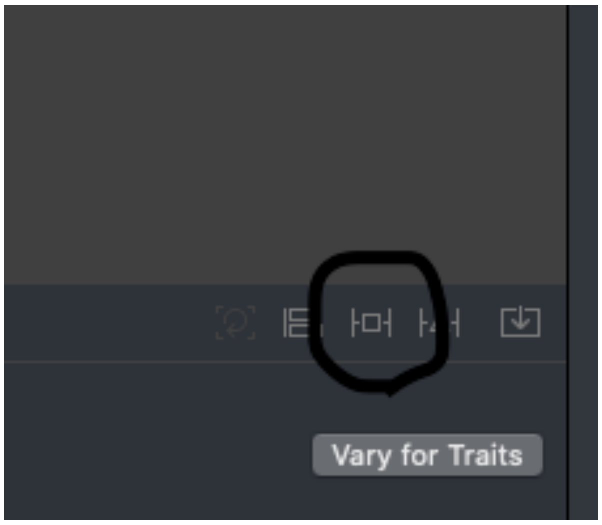
V Vlada· 16 days ago 1

found another solution: add constraints one by one (not 4 altogether). Type "0" for top constraint and tap "Add Constraints". Adjust background to fill into frame and then add just left constraint. Repeat till you do all four. Worked for me

By adding constraints one by one, we can avoid the problem of the safe area interfering leaving blank white area.

Since the Update IOS 13, SWIFT 5…

The constraints operate a little differently. Be sure to check the view log, as she will teach IOS documentation later

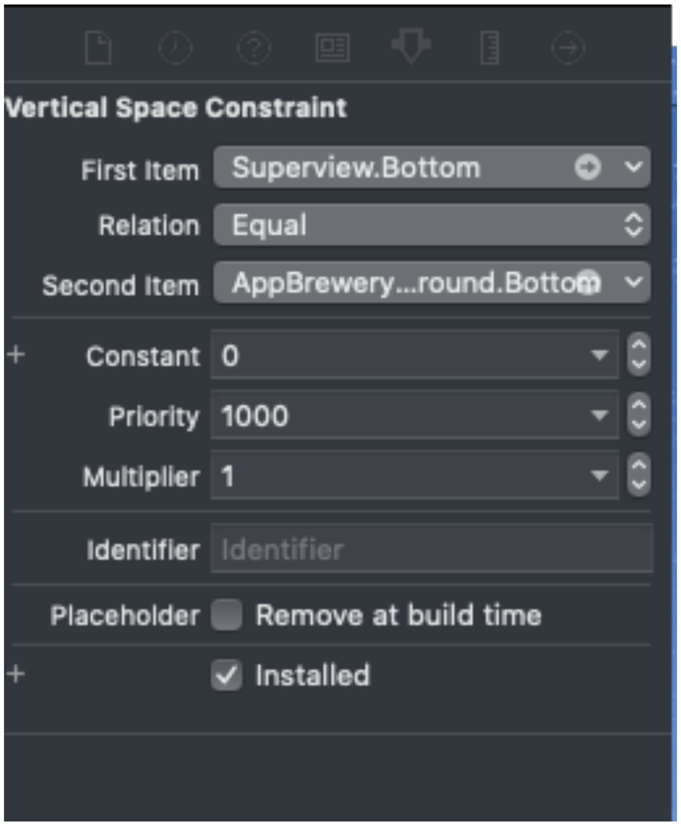


In the bottom right hand side is constraints setter, once clicked you will see a cross +, clicking one at a time red

—

You can also fix this by highlighting the specified constraint. Going all the way to the right to the attribute inspector.

And changing the offending constant to 0, much easier tbh


Vertical Space Constraint

First Item Superview.Bottom

Relation Equal

Second Item AppBrewery...round.Bottom

Constant 0

Priority 1000

Multiplier 1

Identifier Identifier

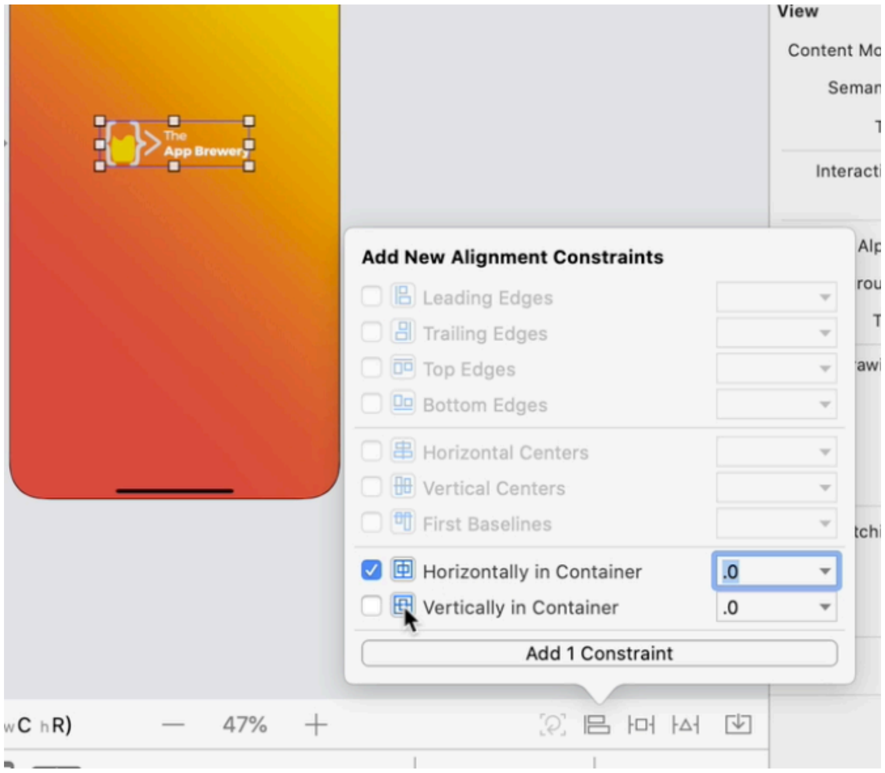
Placeholder ☐ Remove at build time

✓ Installed

This is under the condition that we are just doing portrait view, she has yet to go through landscape calibration

The safe Area concept seems difficult to grasp right now, but its a learn once and learn it FOR once concept. Despite UI updates or changes. The concept will fundamentally stay the safe.

Aligning the logo to the constraints

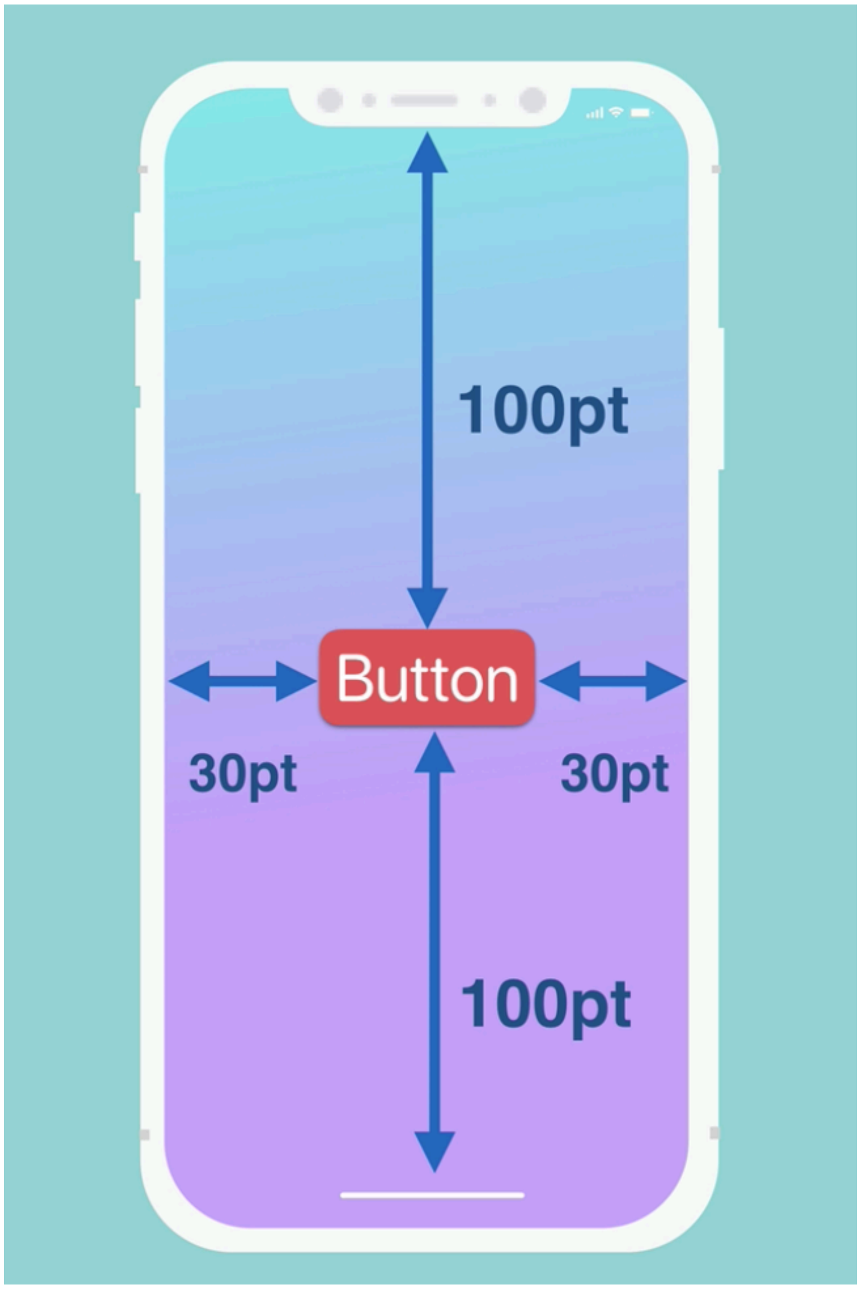


Add New Alignment Constraints

- [ ] Leading Edges
- [ ] Trailing Edges
- [ ] Top Edges
- [ ] Bottom Edges
- [ ] Horizontal Centers
- [ ] Vertical Centers
- [ ] First Baselines
- [✓] Horizontally in Container .0
- [ ] Vertically in Container .0

Add 1 Constraint

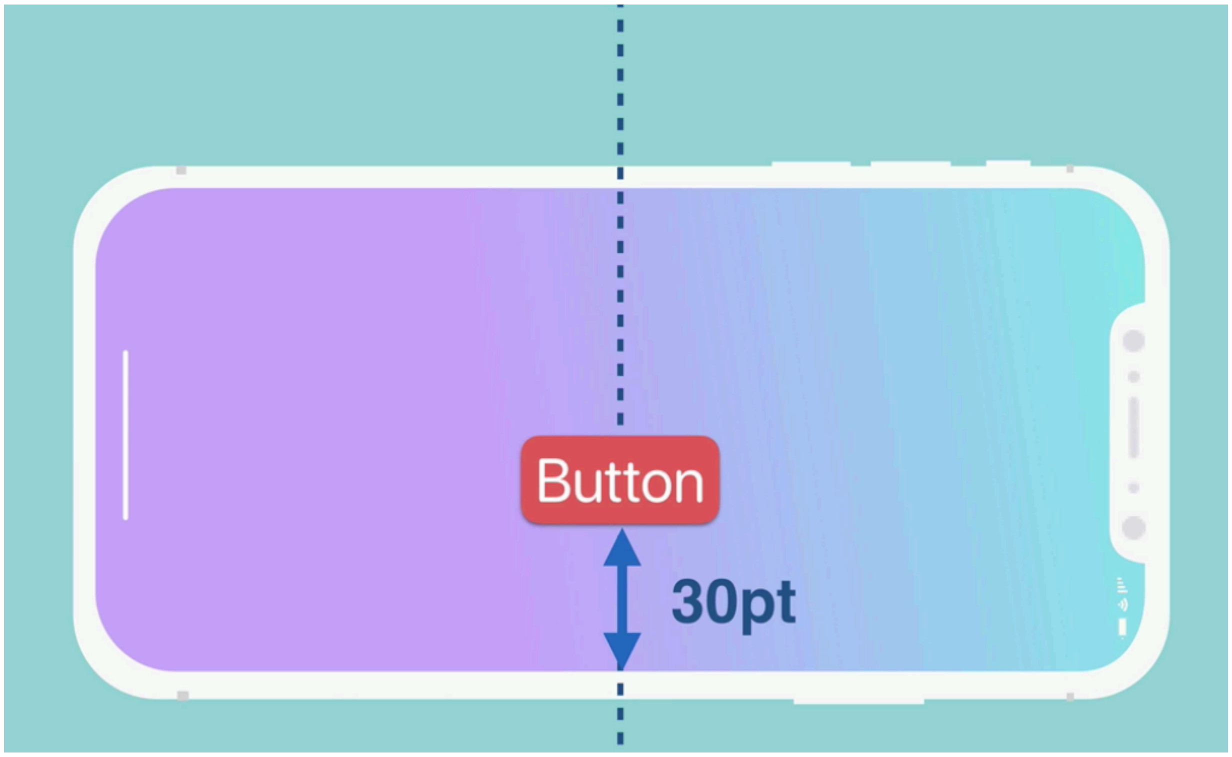
As easy as it looks, now we can flip it horizontal and vertical and not worry about



Constraints are just defining the distance, between an element and it's container

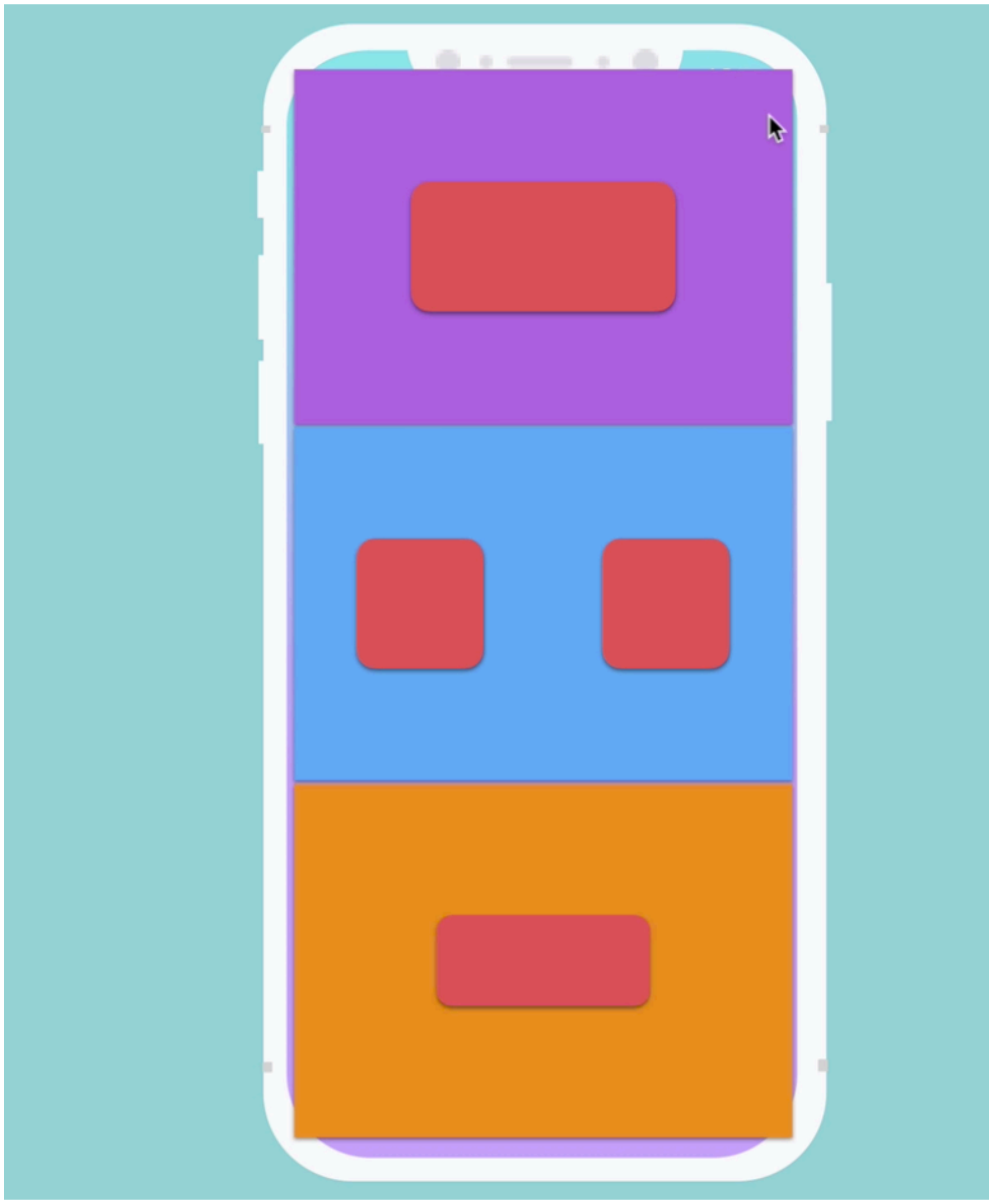
The 'Pining technique works well, until you flip it into landscape, then everything breaks

By 'Aligning as mentioned in the previous photo, we can center the button in both portrait and landscape views.

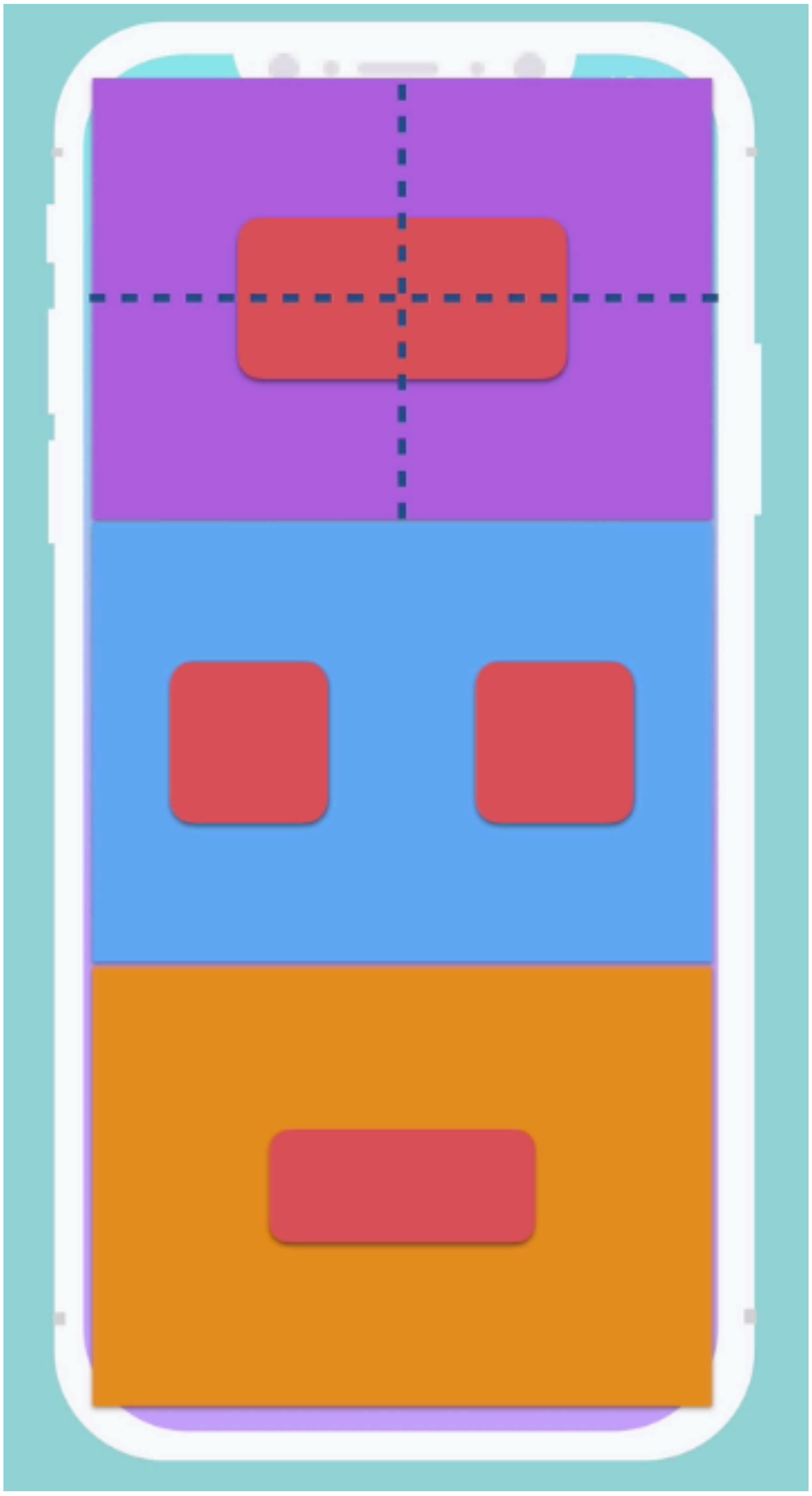


Above is an example of combining alignment with constraint pins, your app will have buttons that will need to located elsewhere besides the center, in this case center horizontal, and lower vertical(30pt)

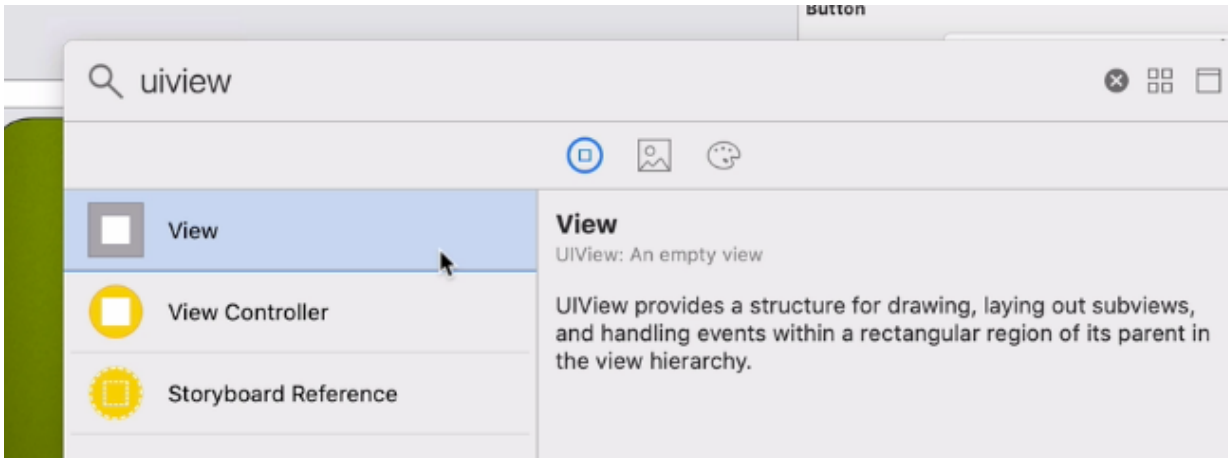
Personally my apps will stay in fixed locations for now, as I don’t like the landscape mode.



You will have apps with multiple buttons, in this example we are separating buttons into 3 containers. Within these 3 containers
We have our buttons aligned.



In this example we are still separating into 3 different containers. Instead of aligning we will be using constraints, along with the new container



View

UIView: An empty view

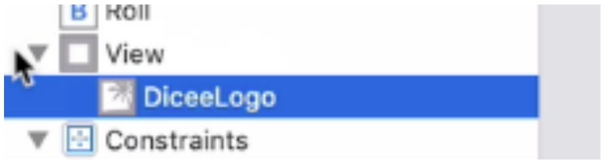
UIView provides a structure for drawing, laying out subviews, and handling events within a rectangular region of its parent in the view hierarchy.

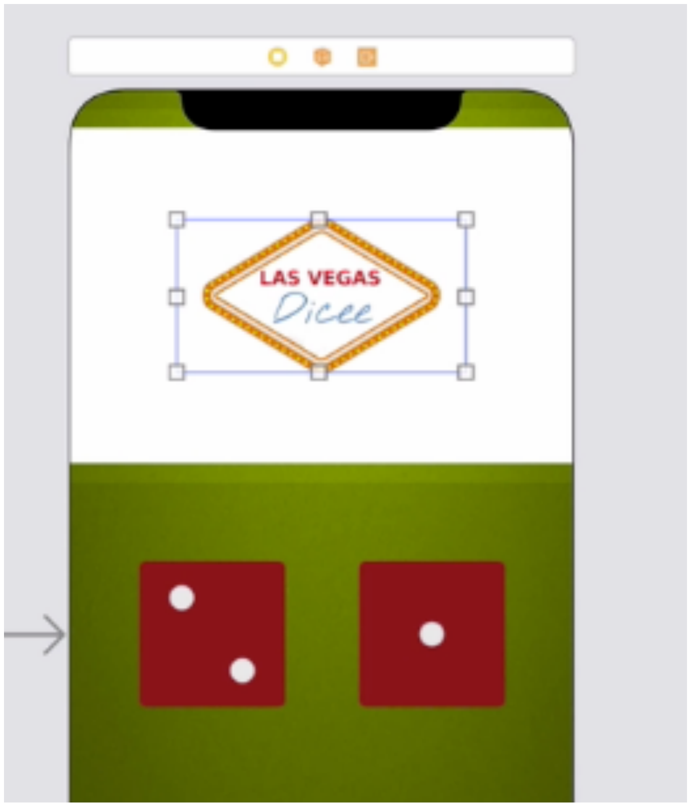
To Create a View, we will have to add the UIView

Embedding, In this case we are embedding Dice logo inside our new white container.

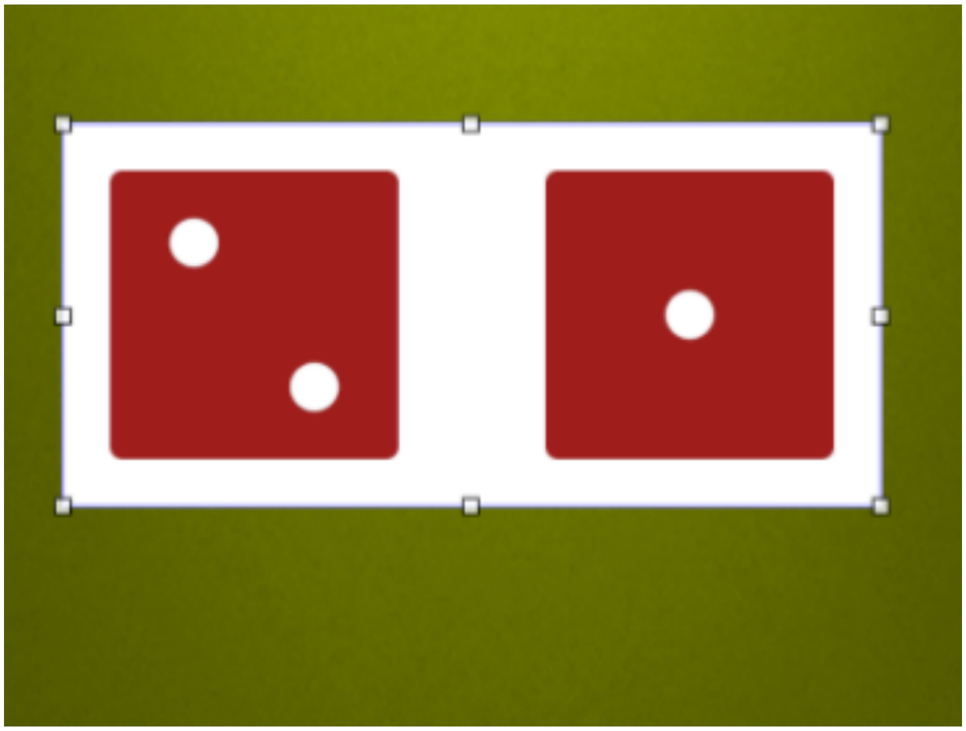
It's one thing to stack an element on top of a container, it's another to lock it in.

Simply Drag and drop the element where into the corresponding view/ container

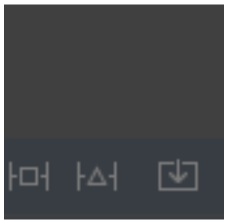




Holding Command + clicking on both dices AND
Going to the top nav bar, and clicking EDITOR/embedIn/view

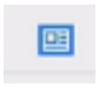


This is better IMO



Highlighting and clicking the Download button on the bottom right. Will Also do the trick

Highlighting Containers and Clicking the 'Identity Inspector' Attribute in the top right corner



 Below document you find 'Label'

We can now add constraints to the labels/logos in the views and add them to the designated areas

Command + Click all the other containers you will get