get back to the playground

...emotionally!

Manu Rink
Technical Evangelist
@
Microsoft

[*0]

We all loved Playgrounds

… back then

[*15]

… and still do!

[*1]

[*3]

[*2]

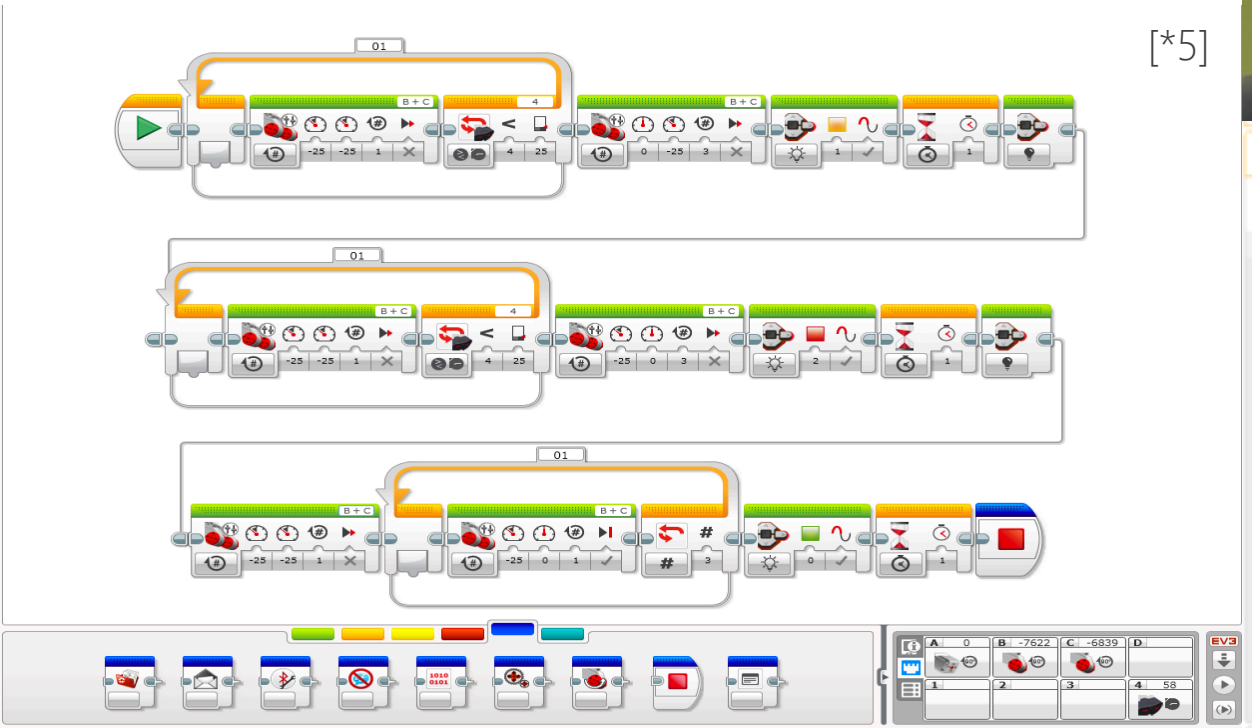
Coding is lot like playing.

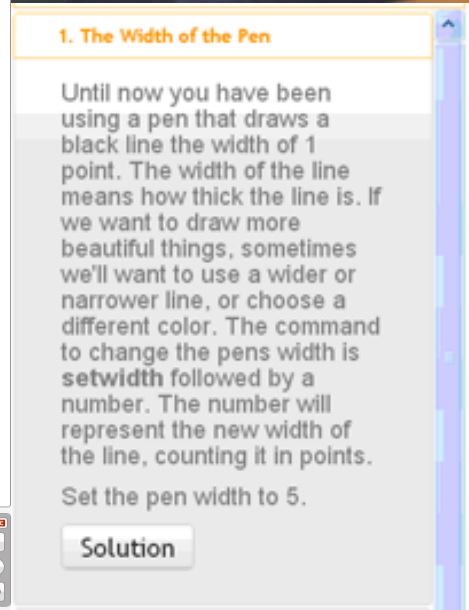
exploring, failing, learning

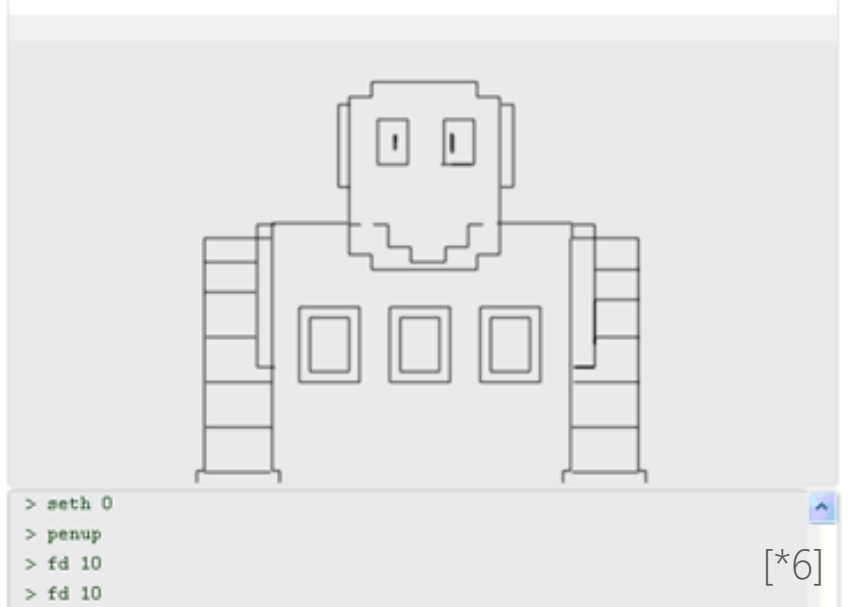
with (mostly) a lot of excitement!

[*4]

[*5]





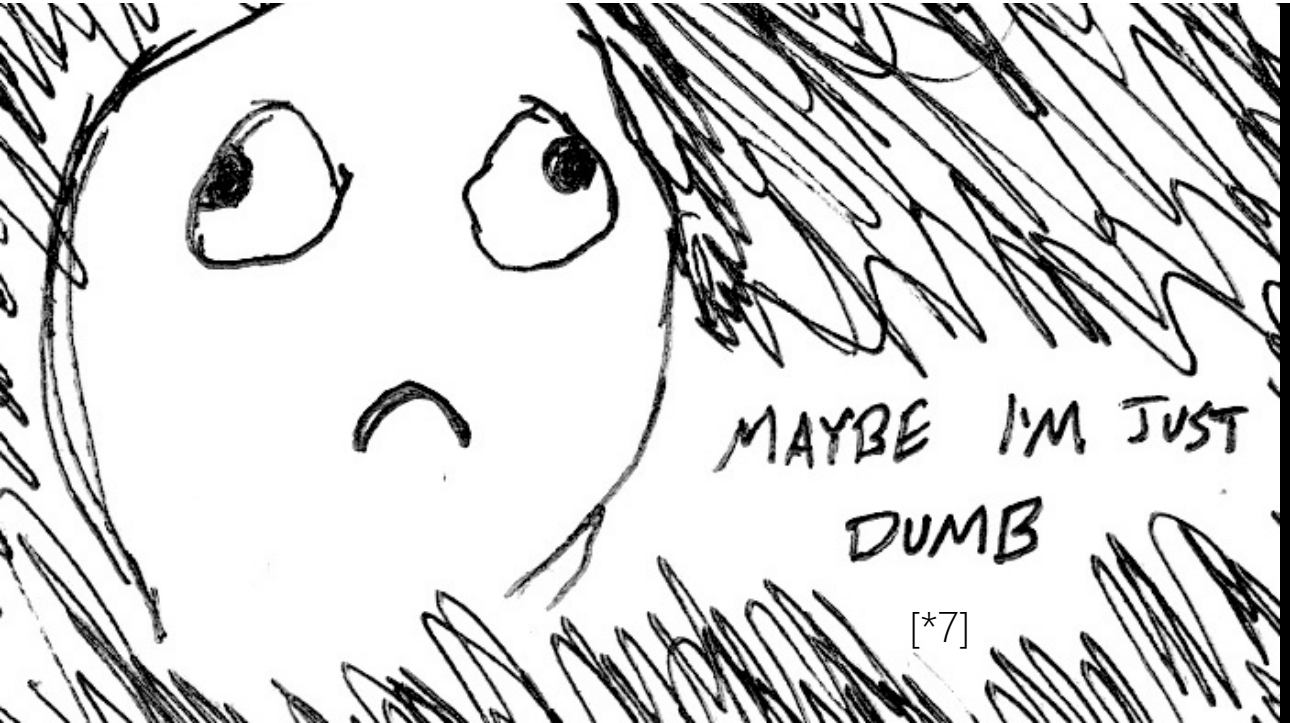


[*6]

Learning new things in coding
mostly **lacks exitement** and
~~can be~~ is often **frustrating**.

"My Adventure lacks excitement - aka - my code is broken"

- codecademy.org

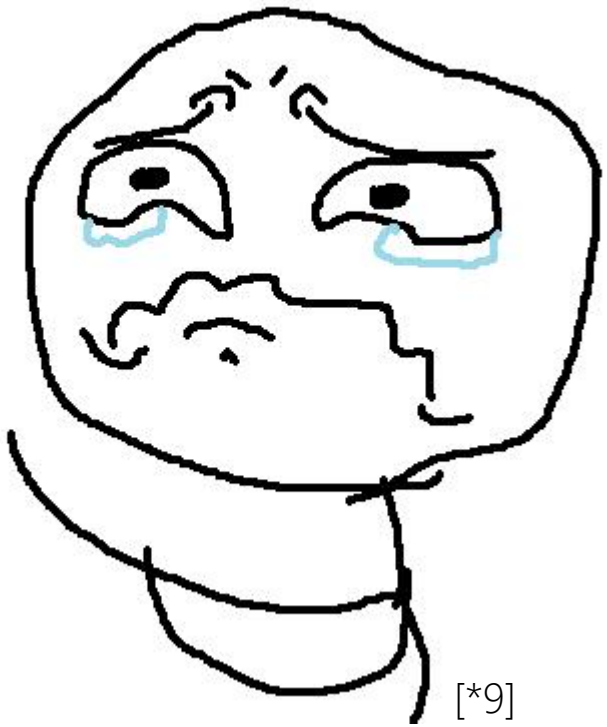


[*7]

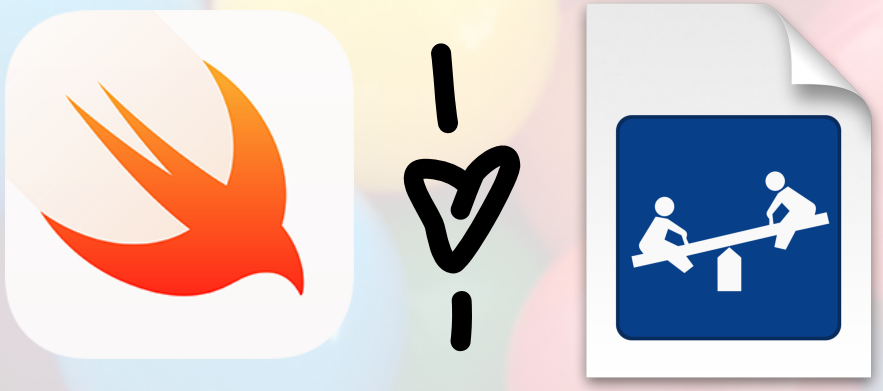
[*8]

missing
incomplete
outdated
wrong
absurd

] documentation ==



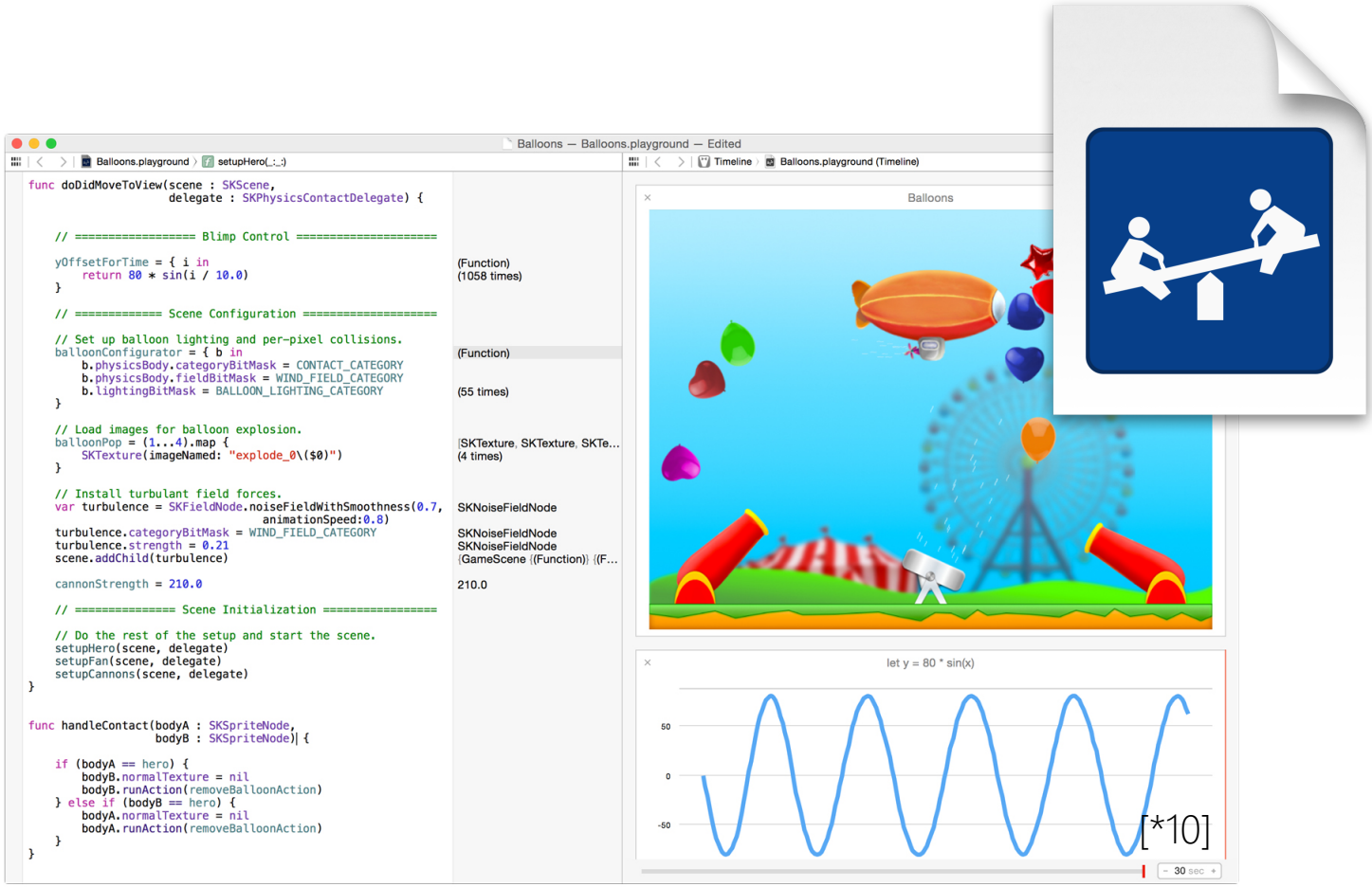
[*9]



Playgrounds

for learning, documenting and exploring code

Learning and compreheding things through immediate exploration !! Literally get „hands-on“ by touching it!



since

iOS8

supports

with

iOS10

additionally

supports

and

A Playground is ...

... a super quick way for swift code experiments

Layout nicely with markdown-ish syntax

Extend source by using basically every SDK, framework and lib for development

Guide the user with links through navigation keywords

Hide code for focusing on one essential thing at a time

Interact with code through the „live view“

Easily edit code by using placeholders aka literals for color, images and files

Embedded result views inline

Choose OS for targeted execution

Running ComputerVision

Kick start Cognitive Services
- First steps
  - Sources
  - Resources
    - justtext.txt
    - keepcalm.png
- ComputerVision M
  - Sources
  - Resources
    - justtext.txt
- Emotions
  - Sources
  - Resources
- Faces
  - Sources
  - Resources
- Sources
  - CognitiveServices.swift
  - Ext.swift
- Resources
  - Aaron.jpg
  - beach.png
  - containers.png
  - Giugli.png
  - highway.png
  - Jan.png
  - Les.jpg
  - Nazuki.png
  - nightcity.png
  - Owen_Family.jpg
  - Tiffany.jpg
  - woman_blue.png
  - wood.png

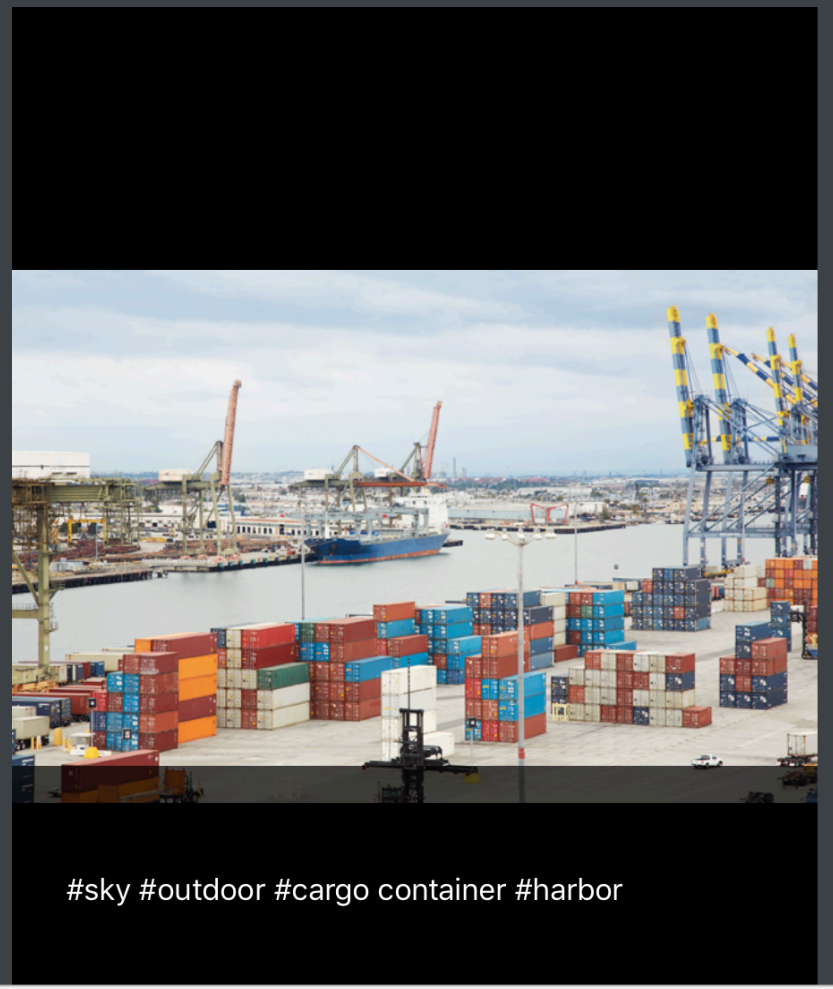
Filter

Kick start Cognitive Services 〉 ComputerVision

```
let backgroundView = UIView(frame: CGRect(x: 0, y: myView.bounds.height-170,
    width: myView.bounds.width, height: 200))
backgroundView.backgroundColor =
backgroundView.alpha = 0.7

myView.addSubview(preview)
myView.addSubview(backgroundView)
myView.addSubview(textLabel)

func showTagsForImage (_ photo : UIImageView, _ confidence : Double) {
    let manager = CognitiveServices()
    textLabel.text = "... gimme a sec - getting your tags!"
    manager.retrievePlausibleTagsForImage(photo.image!, confidence) { (result,
        error) -> (Void) in
        DispatchQueue.main.async(execute: {
            if let _ = error {
                print("omg something bad happened: \(error)")
            } else {
                print("seems like all went well: \(result)")
            }
            setTagsAsDescription(result)
        })
    }
}

func setTagsAsDescription (_ tags : [String]?) {
    if (tags?.count)! > 0 {
        textLabel.text = ""
        for tag in tags! {
            textLabel.text = textLabel.text! + "#" + tag + " "
        }
    } else {
        textLabel.text = "Uh noez! No tags could be found for this image :("
    }
}

//#-end-hidden-code
/*:
 * experiment:
 Every part of the description of the picture will be returned with a certain
     confidence. A good value is 0.85 for nice fitting results. But go a head
     and play around with this value and see, with what funky descriptions the
     "computer" may come along
```

```
UIView
UIView
UIView
UIView
UIView
UIView
<Kick_start_Cognitive_Se...
UILabel
<Kick_start_Cognitive_Se...
()
"seems like all went well:...
UILabel
(4 times)
```

Manual 〉 ComputerVision.xcplaygroundpage (Timeline)




```
            name = "cargo container";
        },
            {
            confidence = "0.4507218599319458";
            name = harbor;
        }
    );
}
["sky", "outdoor", "cargo container", "harbor"]
seems like all went well: Optional(["sky", "outdoor", "cargo container", "harbor"])
```

A Playground Book is ...

... a perfekt place for exploring and learning

Always-on live view with various styling possibilities

Organise content into chapters with pages

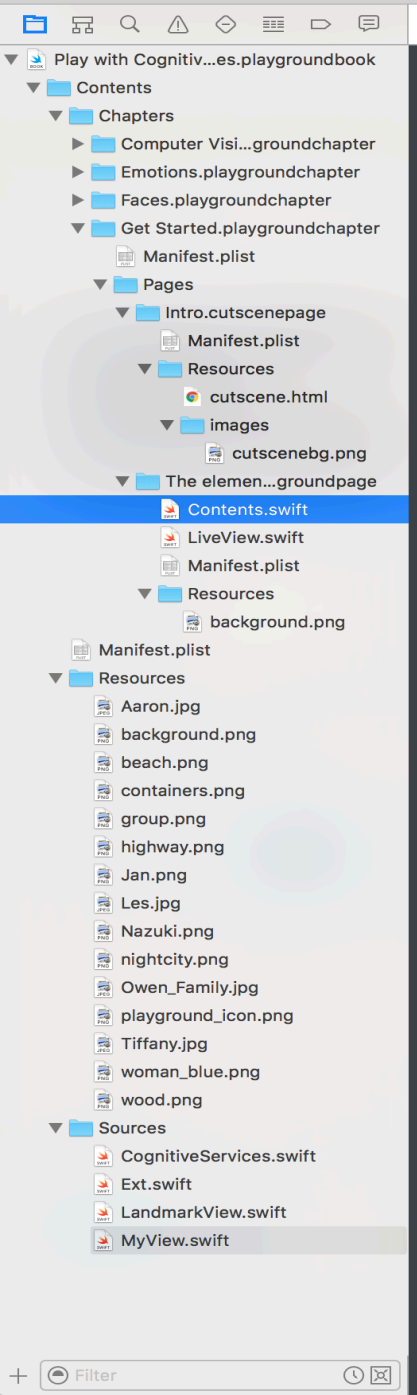
Use cut scenes as special pages for editorial freedom

Limit and control code completion suggestions

Style the appearance in the „store“

Reset the content of the book or single pages

... with way more complexity!

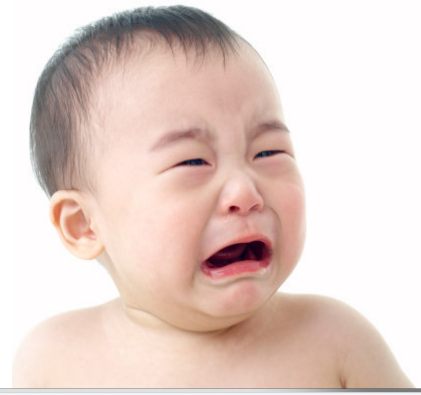



```swift
public class MyView : UIViewController {

    let preview = UIImageView()
    let textLabel = UILabel()
    let backgroundView = UIView()
    let landmarkView = MyLandmarkView()

    var confidence = 0.85

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

        view.frame = CGRect(x: 0, y: 0, width: 520, height: 768)
        let imageBGView = UIImageView(image: UIImage(named:"background.png")!)
        view.addSubview(imageBGView)

        preview.frame = view.bounds
        preview.contentMode = .scaleAspectFit

        textLabel.frame = CGRect(x: 30, y: view.bounds.height-200, width: 350, height:
            110)
        textLabel.lineBreakMode = .byWordWrapping
        textLabel.numberOfLines = 5
        textLabel.textColor = .white
        textLabel.text = "This label makes place for your description :)"

        backgroundView.frame = CGRect(x: 0, y: view.bounds.height-210, width: view.
            bounds.width, height: 210)
        backgroundView.backgroundColor = .black
        backgroundView.alpha = 0.5

        landmarkView.frame = view.bounds
        landmarkView.backgroundColor = .clear

        view.addSubview(preview)
        view.addSubview(backgroundView)
        view.addSubview(textLabel)
        view.addSubview(landmarkView)
        view.bringSubview(toFront: landmarkView)

        makeLandmarkViewVisible(false)
    }

    public func setTheDescription(_ message: String) {
        textLabel.text = message
    }

    public func setTheTextColor(_ color: UIColor) {
        textLabel.textColor = color
    }

    public func setTheImage(_ image: UIImage) {
        preview.image = image
    }

    public func reply(_ message: String) {
        textLabel.text = message
    }

    public func makeLandmarkViewVisible(_ visible: Bool) {
        landmarkView.alpha = visible ? 1.0 : 0.0
    }

    public func updateImage ( _ image: UIImage) {
        preview.image = image
        view.setNeedsDisplay()
        textLabel.text = "upated image ... wohooo \(image)"
    }

    /**
     cognitive services functions
     called from the LiveViewMessageHandler
     */
```



```swift
/*:
 # First steps with Playgrounds

 The first thing we want to try is to get hands on with what we know already - dealing with out beloved UIKit. You can use all elements of UIKit as you are used to. In this example we want to build our environment for our further examples. Let's warm up, put your hands on the playground and get started!

 * callout(What to do):
 Just choose a picture you like, then enter a text you think fits to the image. To make it nice looking, choose a color for your text.
 */
//#-hidden-code
import PlaygroundSupport
import UIKit
import Foundation

guard #available(iOS 9, OSX 10.11, *) else {
    fatalError("Life? Don't talk to me about life. Here I am, brain the size of a planet, and they tell me to run a 'playground'. Call that job satisfaction? I don't.")
}

func setDescription(_ message: String) {
    let page = PlaygroundPage.current
    if let proxy = page.liveView as? PlaygroundRemoteLiveViewProxy {
        proxy.send(.string(message))
    }
}

func setMyTextColor(_ color: UIColor) {
    let page = PlaygroundPage.current
    if let proxy = page.liveView as? PlaygroundRemoteLiveViewProxy {
        let message : String = color.toHexString()
        proxy.send(.string(message))
    }
}

func chooseImage (_ imageData: Data) {
    let page = PlaygroundPage.current
    if let proxy = page.liveView as? PlaygroundRemoteLiveViewProxy {
        proxy.send(.data(imageData))
    }
}
//#-end-hidden-code
let image = /*#-editable-code*/ /*#-end-editable-code*/
let dataImage = UIImagePNGRepresentation(image)
chooseImage(dataImage!)
setDescription(/*#-editable-code */"Description goes here!"/*#-end-editable-code*/)
setMyTextColor(/*#-editable-code */ /*#-end-editable-code*/)
/*:
 * callout(What did we learn?):
 So we are done with the basics. We created a `UIImageView` with an embedded `UIImage`. Our description area consists of an `UIView`, which background color and alpha is adjustable. And above this background view we added a `UILabel`, which shows our nice descriptive text for the picture.
 */

//: Horray! Let's get to our [next adventure](@next)!
```

A Playground book is complex ...



[*11]

No Preview in Xcode

Executes solely in the iPad Swift Playgrounds app

Cumbersome stucturing with plists

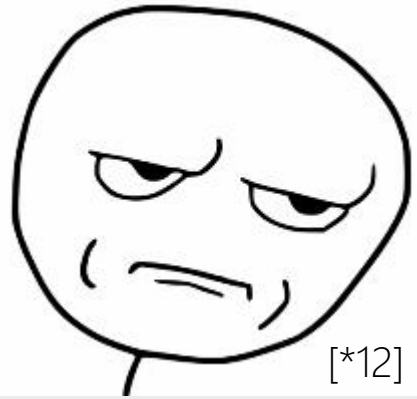
Content of chapters andpages needs to be defined in a bunch of plists

Separate communication for always-on live view

Fairly complex protocol for transfering data from the playground source to the live view

A Playground Book structure ...

... is defined by various manifest.plist files



[*12]

Ready | Today at 18:43

Play with Cognitive S...ices.playgroundbook › Manifest.plist › No Selection

Manual › P › C › C › G › Manifest.plist › No Selection

- Play with Cognitiv...es.playgroundbook
  - Contents
    - Chapters
      - Computer Visi...groundchapter
      - Emotions.playgroundchapter
      - Faces.playgroundchapter
      - Get Started.playgroundchapter
        - Manifest.plist
        - Pages
          - Intro.cutscenepage
            - Manifest.plist
            - Resources
          - The elemen...groundpage
            - Contents.swift
            - LiveView.swift
            - Manifest.plist
            - Resources
    - Manifest.plist
    - Resources
    - Sources

Filter

Key	Type	Value
Root	Dictionary	(7 items)
Version	String	1.0
ContentVersion	String	1.0
Name	String	Play with Cognitive Services
ContentIdentifier	String	com.ms.demo.CSPlayground
DeploymentTarget	String	ios10.0
ImageReference	String	playground_icon.png
Chapters	Array	(4 items)
Item 0	String	Get Started.playgroundchapter
Item 1	String	Computer Vision.playgroundchapter
Item 2	String	Emotions.playgroundchapter
Item 3	String	Faces.playgroundchapter

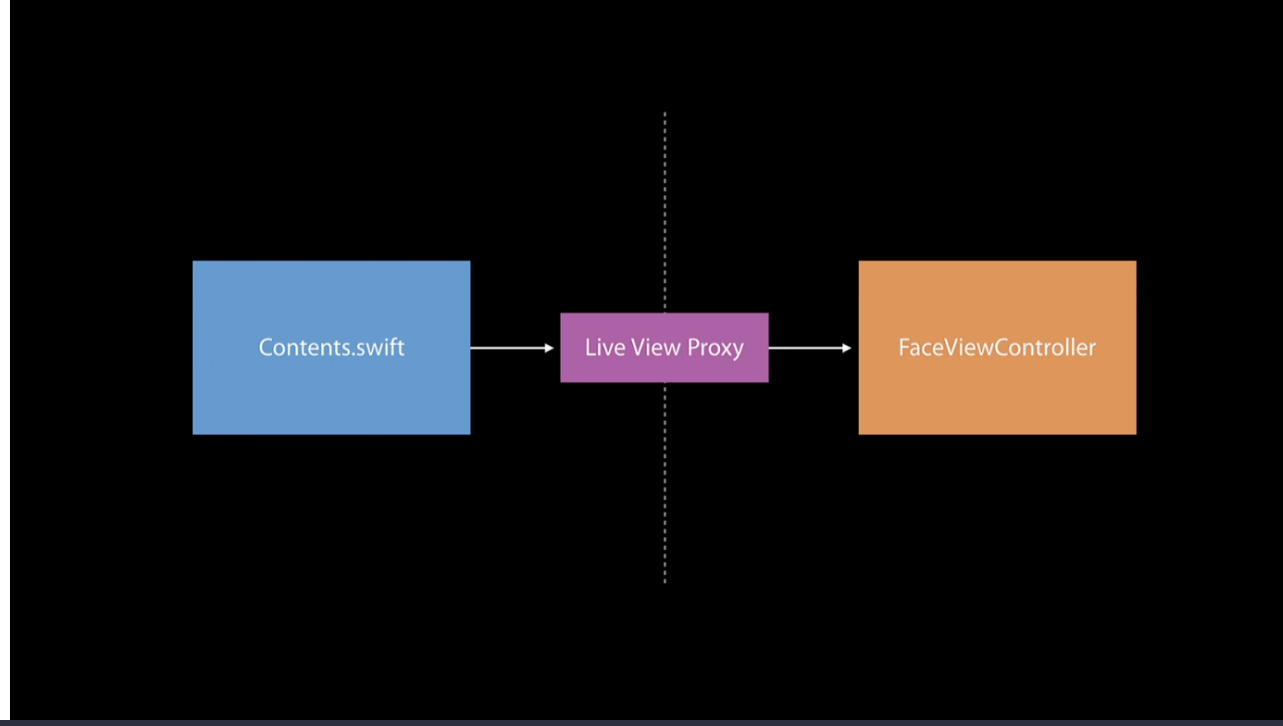
Key	Type	Value
Root	Dictionary	(3 items)
Version	String	1.0
Name	String	Get Started
Pages	Array	(2 items)
Item 0	String	Intro.cutscenepage
Item 1	String	The elements.playgroundpage

... › Manifest.plist › No Selection

Key	Type	Value
Root	Dictionary	(5 items)
Version	String	1.0
Name	String	The Elements
LiveViewMode	String	VisibleByDefault
LiveViewEdgeToEdge	String	YES
PosterReference	String	background.png

Playground Book

Always-on live-view specialities and utilities




```
public enum PlaygroundValue {
    case array([PlaygroundValue])
    case dictionary([String: PlaygroundValue])
    case string(String)
    case data(Data)
    case date(Date)
    case integer(Int)
    case floatingPoint(Double)
    case boolean(Bool)
}
```

```
let store = PlaygroundPage.current.keyValueStore

store["Greeting"] = .string("Hello, WWDC!")

if case let .string(greeting)? = store["Greeting"] {
    print(greeting)     // "Hello, WWDC!"
}
```

Playground Book

Always-on live-view communication

between the Content.swift
and the instane of the live view

1 Send data of type `PlaygroundValue` from Content.swift to the live view by using the `send function` of the `PlaygroundRemoteLiveViewProxy`

2 Implement the `PlaygroundRemoteLiveViewProxyDelegate` with ist `remoteLiveViewProxy(...)` method and assign an instance of it to the proxys `delegate` of the current `PlaygroundPage`

1

```
import PlaygroundSupport
func say(_ message: String) {
    let page = PlaygroundPage.current
    if let proxy = page.liveView as? PlaygroundRemoteLiveViewProxy {
        proxy.send(.string(message))
    }
}
//#-end-hidden-code

say(/*#-editable-code*/"<#Knock, knock!#>"/*#-end-editable-code*/)
```

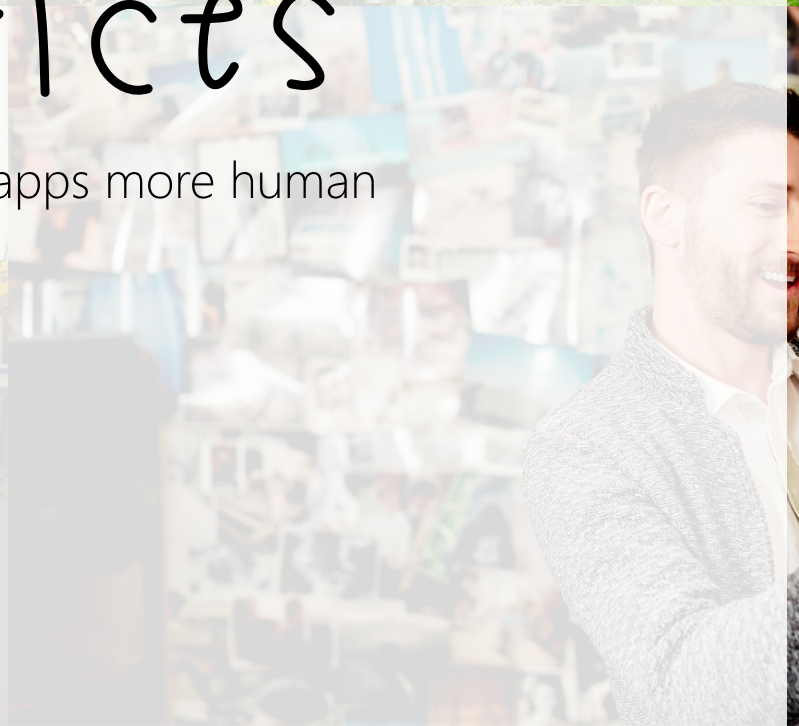
2

```
let page = PlaygroundPage.current
page.needsIndefiniteExecution = true
let proxy = page.liveView as? PlaygroundRemoteLiveViewProxy
class MyClassThatListens: PlaygroundRemoteLiveViewProxyDelegate {
   func remoteLiveViewProxy(_ remoteLiveViewProxy: PlaygroundRemoteLiveViewProxy,
                            received message: PlaygroundValue) {
        if case let .string(text) = message {
            doSomethingWithString(text)
        }
    }
}
let listener = MyClassThatListens()
proxy?.delegate = listener
```










cognitive services

for making your apps more human

Let the API do the work for you...

Vision

Computer Vision

Emotion

Face

Video

Speech

Bing Speech

Custom Recognition

Speaker Recognition

Language

Bing Spell Check

Language Understanding

Linguistic Analysis

Text Analytics

WebLM

Knowledge

Academic

Entity Linking

Knowledge Exploration

Recommendations

Search

Bing Autosuggest

Bing Image Search

Bing News Search

Bing Video Search

Bing Web Search

... what do YOU want to do?

computer vision API

[

Analyse
Describe
Thumbnail
OCR

computer vision API

Analyse an image

https://api.projectoxford.ai/vision/v1.0/analyze[?visualFeatures][&details]

URL parameters

Visual features
Categories, Tags, Description, Faces, ImageType, Color, Adult
Details
Currently just “Celebrities” is supported

Header

Content-Type
application/json, application/octet-stream, multipart/form-data
Ocp-Apim-Subscription-Key
Get your key from “My account” at https://www.microsoft.com/cognitive-services/.
You might have to create an account first.

Documentation: https://www.microsoft.com/cognitive-services/en-us/computer-vision-api/documentation

API Reference https://dev.projectoxford.ai/docs/services/56f91f2d778daf23d8ec6739/operations/56f91f2e778daf14a499e1fa

computer vision API

JSON result for analyzed image

- Details “Celebrities”
- Visual features “Categories, Tags, Description, Adult”


```
"categories": [
    {
        "name": "abstract_",
        "score": 0.00390625
    },
    {
        "name": "people_",
        "score": 0.83984375,
        "detail": {
            "celebrities": [
                {
                    "name": "Satya Nadella",
                    "faceRectangle": {
                        "left": 597,
                        "top": 162,
                        "width": 248,
                        "height": 248
                    },
                    "confidence": 0.999028444
                }
            ]
        }
    }
],
"adult": {
    "isAdultContent": false,
    "isRacyContent": false,
    "adultScore": 0.0934349000453949,
    "racyScore": 0.068613491952419281
},
```

```
"tags": [
    {
        "name": "person",
        "confidence": 0.98979085683822632
    },
    {
        "name": "man",
        "confidence": 0.94493889808654785
    },
    {
        "name": "outdoor",
        "confidence": 0.938492476940155
    },
    {
        "name": "window",
        "confidence": 0.89513939619064331
    }
],
"description": {
    "tags": [
        "person",
        "man",
        "outdoor",
        "window",
        "glasses"
    ],
    "captions": [
        {
            "text": "Satya Nadella sitting on a bench",
            "confidence": 0.48293603002174407
        }
    ]
}
```

computer vision API 1.0

Describe an image

https://api.projectoxford.ai/vision/v1.0/describe[?maxCandidates]

URL parameters - maxCandidates

Header - see “Analyse”

Get Thumbnail

https://api.projectoxford.ai/vision/v1.0/generateThumbnail[?width][&height][&smartCropping]

URL parameters – width, height, smartCropping

Header - see “Analyse”

OCR

https://api.projectoxford.ai/vision/v1.0/ocr[?language][&detectOrientation]

URL parameters – language, detectOrientation

Header - see “Analyse”

computer vision API

On the left:
JSON result for OCR

On the right:
JSON result for Describe


```
"language": "en",
"textAngle": -2.0000000000000338,
"orientation": "Up",
"regions": [
    {
        "boundingBox": "462,379,497,258",
        "lines": [
            {
                "boundingBox": "462,379,497,74",
                "words": [
                    {
                        "boundingBox": "462,379,41,73",
                        "text": "A"
                    },
                    {
                        "boundingBox": "523,379,153,73",
                        "text": "GOAL"
                    },
                    {
                        "boundingBox": "694,379,265,74",
                        "text": "WITHOUT"
                    }
                ]
            }
        ]
    }
]
```

```
{
    "description": {
        "tags": [
            "person",
            "man",
            "outdoor",
            "window",
            "glasses"
        ],
        "captions": [
            {
                "text": "Satya Nadella sitting on a bench",
                "confidence": 0.48293603002174407
            },
            {
                "text": "Satya Nadella is sitting on a bench",
                "confidence": 0.40037006815422832
            },
            {
                "text": "Satya Nadella sitting in front of a building",
                "confidence": 0.38035155997373377
            }
        ]
    },
    "requestId": "ed2de1c6-fb55-4686-b0da-4da6e05d283f",
    "metadata": {
        "width": 1500,
        "height": 1000,
        "format": "Jpeg"
    }
}
```

emotion API

- Recognition in images
- Recognition in videos
- Recognition with rectangles

emotion API beta

Emotion Recognition

https://api.projectoxford.ai/emotion/v1.0/recognize

Header

Content-Type
application/json, application/octet-stream, multipart/form-data
Ocp-Apim-Subscription-Key
Get your key from "My account" at https://www.microsoft.com/cognitive-services/.
You might have to create an account first.

Documentation: https://www.microsoft.com/cognitive-services/en-us/emotion-api/documentation

API Reference https://dev.projectoxford.ai/docs/services/56f91f2d778daf23d8ec6739/operations/56f91f2e778daf14a499e1fa

emotion API beta

JSON result for Emotion Recognition of an image.

For every detected face the API returns
- the face rectangle
- the list of emotions with scores


```
[
  {
    "faceRectangle": {
      "left": 68,
      "top": 97,
      "width": 64,
      "height": 97
    },
    "scores": {
      "anger": 0.00300731952,
      "contempt": 5.14648448E-08,
      "disgust": 9.180124E-06,
      "fear": 0.0001912825,
      "happiness": 0.9875571,
      "neutral": 0.0009861537,
      "sadness": 1.889955E-05,
      "surprise": 0.008229999
    }
  }
]
```

emotion API beta

Emotion Recognition in videos

https://api.projectoxford.ai/emotion/v1.0/recognizeinvideo[?outputStyle]

URL parameters – outputStyle [aggregate, perFrame]

Header - see "Recognition"

Result on 202 - video operation status/result as URL

Emotion Recognition with Face Rectangles

https://api.projectoxford.ai/emotion/v1.0/recognize?faceRectangles={faceRectangles}

URL parameters – faceRectangles (left, top, width, height)

Header - see "Recognition"

Recognition in Video Operation Result

https://api.projectoxford.ai/emotion/v1.0/operations/{oid}]

URL parameters – oid (URL from Emotion Recognition in videos)

Header - see "Recognition"

Result: Status of recognition operation. On SUCCEEDED -> JSON can be retrieved from processingResult field.
https://www.microsoft.com/cognitive-services/en-us/emotion-api/documentation/howtocallemotionforvideo

Face API

- Detect
- Verify
- Identify
- Find Similar

Face API 1.0

Detect Faces

https://api.projectoxford.ai/face/v1.0/detect[?returnFaceId][&returnFaceLandmarks][&returnFaceAttributes]

URL parameters

returnFaceId
faceId needed if face should later be attached to a person
returnFaceLandmarks
get position of e.g. eyes, pupils, nose, eyebrows,…
returnFaceAttributes
get attributes “age, gender, smile, facialHair, headPose, glasses” for a face

Header

Content-Type
application/json, application/octet-stream, multipart/form-data
Ocp-Apim-Subscription-Key
Get your key from “My account” at https://www.microsoft.com/cognitive-services/.
You might have to create an account first.

Documentation: https://www.microsoft.com/cognitive-services/en-us/face-api/documentation/overview

API Reference https://dev.projectoxford.ai/docs/services/563879b61984550e40cbbe8d/operations/563879b61984550f30395236

Face API 1.0

JSON result for Face Detection of an image.

For every detected face the API returns

- the faceId
- the list of faceLandmarks
- requested attributes of the face


```
"faceId": "c5c24a82-6845-4031-9d5d-978df9175426",
"faceRectangle": {
    "width": 78,
    "height": 78,
    "left": 394,
    "top": 54
},
"faceLandmarks": {
    "pupilLeft": {
        "x": 412.7,
        "y": 78.4
    },
    "pupilRight": {
        "x": 446.8,
        "y": 74.2
    },
    "noseTip": {
        "x": 437.7,
        "y": 92.4
    },
    "mouthLeft": {
        "x": 417.8,
        "y": 114.4
    },
    "mouthRight": {
        "x": 451.3,
        "y": 109.3
    },
    "eyebrowLeftOuter": {
        "x": 397.9,
        "y": 78.5
    },
    "eyebrowLeftInner": {
        "x": 425.4,
        "y": 70.5
    },
    "eyeLeftOuter": {
        "x": 406.7,
        "y": 80.6
    },
    "eyeLeftTop": {
        "x": 412.2,
        "y": 76.2
    },
"faceAttributes": {
    "age": 71.0,
    "gender": "male",
    "smile": 0.88,
    "facialHair": {
        "mustache": 0.8,
        "beard": 0.1,
        "sideburns": 0.02
    }
},
"glasses": "sunglasses",
"headPose": {
    "roll": 2.1,
    "yaw": 3,
    "pitch": 0
}
```

Face API 1.0

Find Similar Faces

https://api.projectoxford.ai/face/v1.0/findsimilars

URL parameters – faceId, faceListId, faceIds, maxNumOfCandidatesReturned, mode [matchPerson, matchFace]

Header - see "Detect"

Verify a face

https://api.projectoxford.ai/face/v1.0/verify

Request Body
Face2Face Verification: faceId1, faceId2
Face2Person Verification: faceId, personGroupId, personId

Header - see "Detect"

Identify a face

https://api.projectoxford.ai/face/v1.0/identify

Request Body – faceIds, personGroupId, maxNumOfCandidatesReturned, confidenceThreshold

Header - see "Detect"

Face API 1.0

APIs for creating, populating, training and maintaining persons, person groups and face lists

▼ Person

- POST Add a Person Face
- POST Create a Person
- DELETE Delete a Person
- DELETE Delete a Person Face
- GET Get a Person
- GET Get a Person Face
- GET List Persons in a Person Group
- PATCH Update a Person
- PATCH Update a Person Face

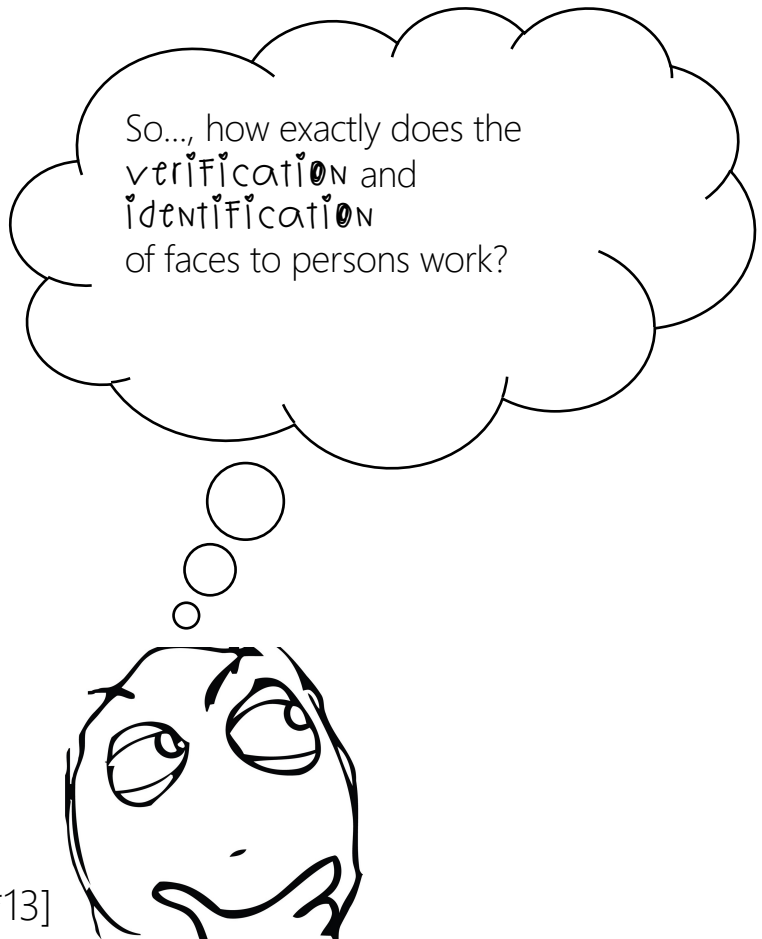
▼ Person Group

- PUT Create a Person Group
- DELETE Delete a Person Group
- GET Get a Person Group
- GET Get Person Group Training Status
- GET List Person Groups
- POST Train Person Group
- PATCH Update a Person Group

▼ Face List

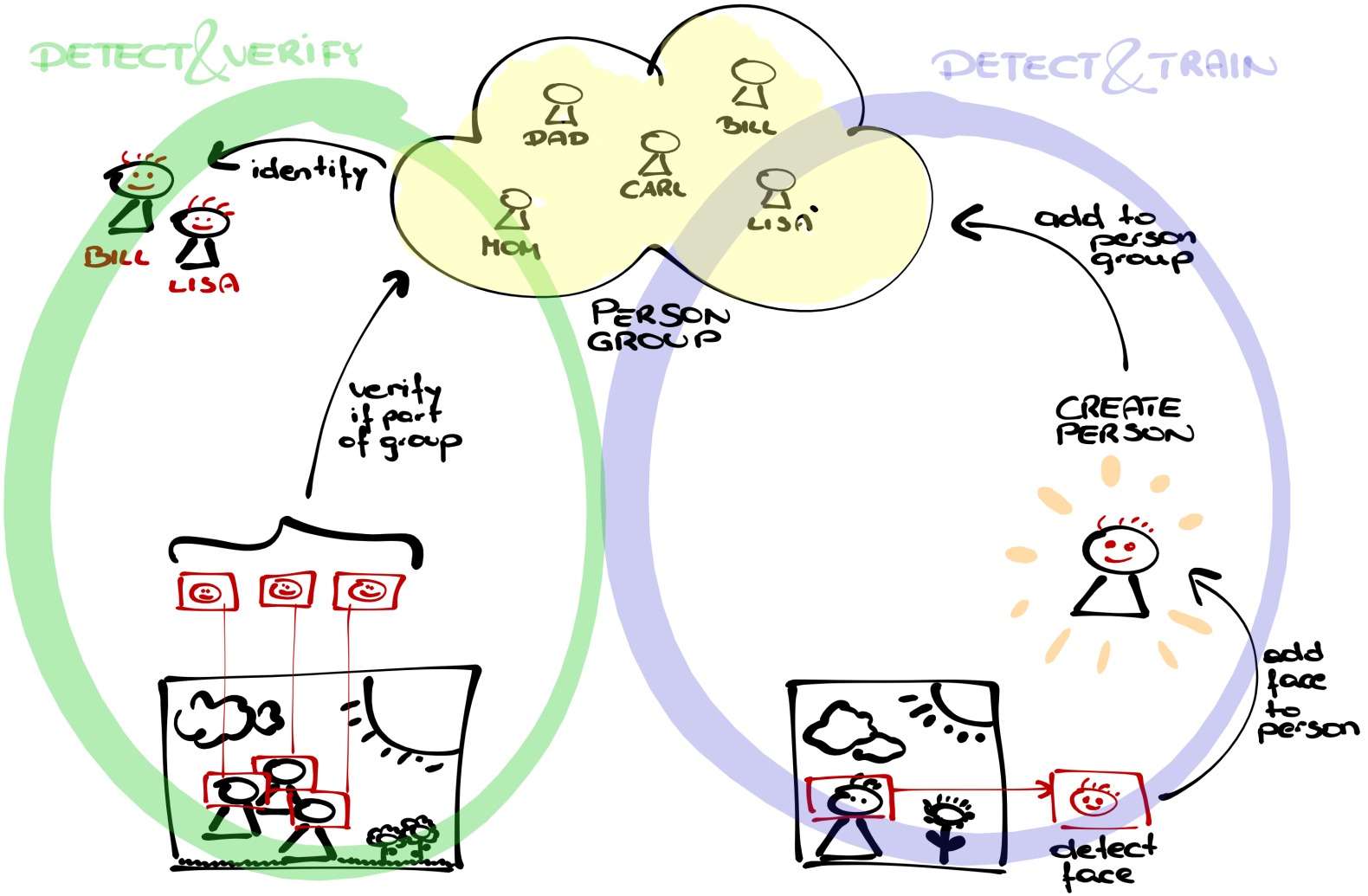
- POST Add a Face to a Face List
- PUT Create a Face List
- DELETE Delete a Face from a Face List
- DELETE Delete a Face List
- GET Get a Face List
- GET List Face Lists
- PATCH Update a Face List

Fact API 1.0



[*13]

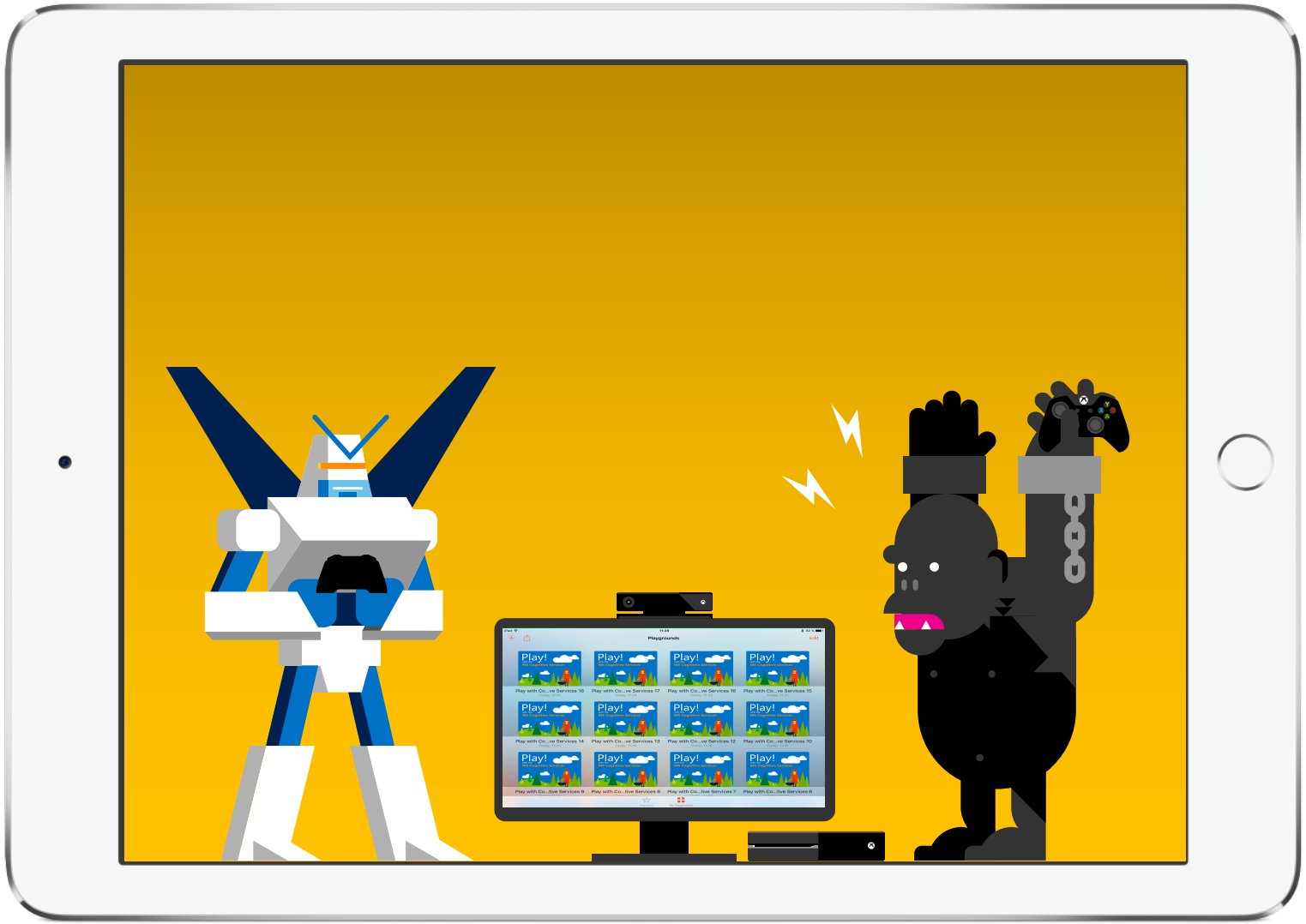
[*14]



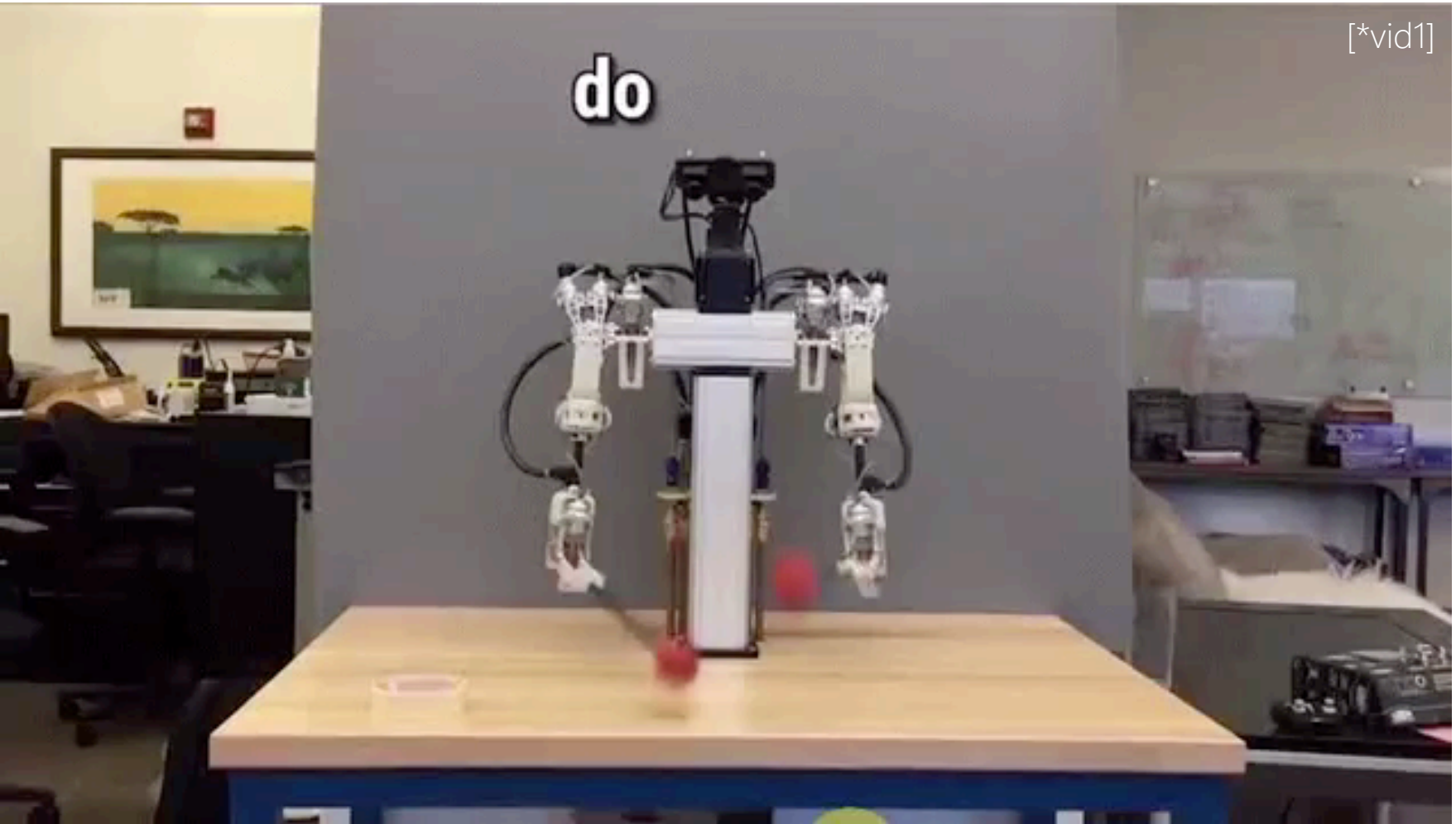
Bringing it all together

Cognitive Services

demo



Let's get started
make our tech more human!



Disney Research created a telepresence robot which feels human in his interactions.

Link to the paper:
https://s3-us-west-1.amazonaws.com/disneyresearch/wp-content/uploads/20160503162533/A-Hybrid-Hydrostatic-Transmission-and-Human-Safe-Haptic-Telepresence-Robot-Paper.pdf

The necessities…

Tech material links
- https://github.com/codePrincess/playgrounds
- http://ericasadun.com
- https://itunes.apple.com/us/book/playground-secrets-power-tips/id982838034
- https://developer.apple.com/library/prerelease/content/documentation/Xcode/Conceptual/swift_playgrounds_doc_format/
- https://developer.apple.com/library/ios/documentation/Xcode/Reference/xcode_markup_formatting_ref/
- https://github.com/ashfurrow/playgroundbook

Fonts
- Segoe UI Light/Normal – MS Standard Font for Decks
- Child wish: http://www.dafont.com/de/childswish.font

Images
[*0] https://metrouk2.files.wordpress.com/2015/10/jl_picture_14.jpg
[*1] http://xn--bllebad-und-mehr-vnb.de/images/product_images/original_images/Art1006088.jpg
[*2] https://www.nycgovparks.org/photo_gallery/full_size/19014.jpg
[*3] http://www.shareable.net/sites/default/files/styles/blog-header-large/public/blog/top-image/PlaygroundHeader.jpg?itok=036ZlixY
[*4] http://edtechtimes.com/wp-content/uploads/2016/01/kids-playing-games-880x440.jpg?resolution=1024,1
[*5] http://blog.grunick.com/wp-content/uploads/2015/09/Mindstorms.png
[*6] https://cdn.brainpop.com/games/turtleacademy/screenshot1.png
[*7] https://techcrunch.com/2014/05/24/dont-believe-anyone-who-tells-you-learning-to-code-is-easy/
[*8] https://media.giphy.com/media/13HgwGsXF0aiGY/giphy.gif
[*9] http://memesvault.com/wp-content/uploads/Sad-Meme-04.jpg
[*10] http://swiftplayground.org/blog_images/playground-screenshot.jpg
[*11] https://img.buzzfeed.com/buzzfeed-static/static/2015-12/2/14/enhanced/webdr01/enhanced-17255-1449085183-9.jpg
[*12] http://media.tumblr.com/tumblr_m5cw224lps1qcwic6.jpg
[*13] http://www.vccoaching.com/wp-content/uploads/2014/12/meme-thinking-face-1920x1080.jpg
[*14] Copyright by Manuela Rink, @codePrincess
[*15] http://documama.org/wp-content/uploads/2013/02/IMG_3614.jpg
[*16] http://i3.kym-cdn.com/photos/images/facebook/000/210/119/9b3.png

Videos
[vid1] https://www.youtube.com/watch?v=whqCf0onDWU

[*16]

Merci :)

says

Manu Rink
Technical Evangelist

marink@microsoft.com
@codeprincess

Microsoft