

tvOS watchOS

IZA, Martin Hrubý, FIT VUT, 2018





Apple TV, Apple Watch

- Apple TV (2007) — iPod pro televizi. Nemá dotykové rozhraní, takže změna do iOS.
- Apple Watch — první pokus o nositelné zařízení.
 - Reálné možnosti HW (a baterie) ovlivnily podobu OS (watchOS) a koncepci aplikací.
 - Od počátku koncipovány jako "rozšíření" iPhone.
 - CPU S1, 8GB HDD. Kvóty pro uložení (obrázky, hudba).

Apple TV, Apple Watch

- U obou zařízení Apple definuje nějaké druhy kvót pro uložení.
- Zatížení procesoru:
 - Apple TV — bez omezení, plná aplikace typu iOS.
 - Apple Watch — (obavy o baterii) — Watch nemá sloužit pro výpočty, ale především jako uživatelský terminál (notifikace, hlasové ovládání, Apple Pay, autentizace).
- Personálnost — Watch (zcela), TV (lze Apple ID, tj. iCloud). TV však je zařízení domácnosti.

Apple TV a tvOS

- tvOS je odvozeno od iOS, avšak jsou tam jisté odlišnosti.
 - práce s uložištěm,
 - programování v JavaScript (volitelně),
 - CloudKit. Odstraněn WebKit.
- Apple TV: CPU A8 (iPhone 6), 32(64) GB HDD, 2GB RAM, wifi / ethernet, HDMI, dálkový ovladač.

tvOS

- Koncept zvýrazněného prvku UI, přecházení mezi zvýrazněnými prvky.
 - jako formulář nebo web,
 - "focus", zprávy: update-focus.
- Stisk OK, posouvání, stisk Zpět.
- Emulátor: lze ovládat šipkami.
- Přihlášení do Cloud, Apple ID.

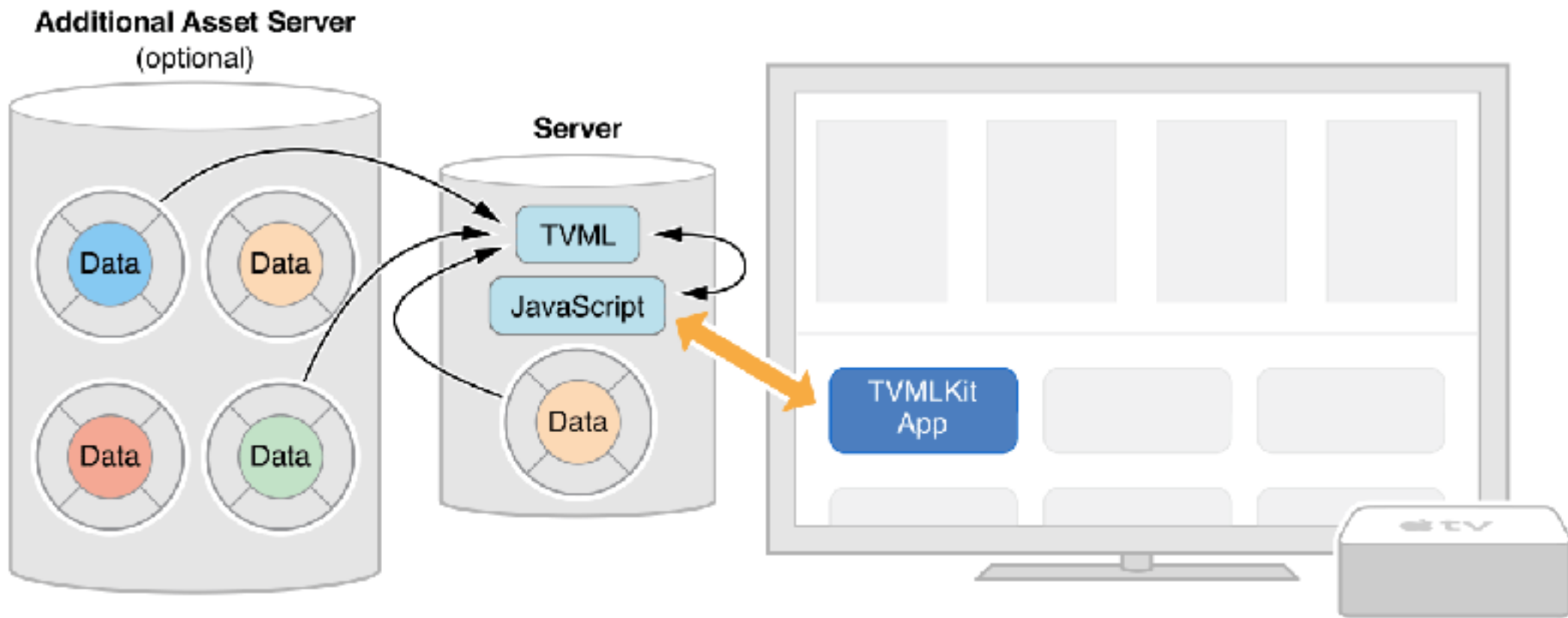
Kvóty pro aplikaci

- App Bundle — 4GB. Herní zařízení.
 - On-demand resources — kvóta rozšířena až na 24GB.
- Lokální data aplikace — 500 KB (UserDefaults).
 - Aplikace smí využívat možnosti CloudKit, tj. Key-Value Storage (1MB),
 - ... a CKAsset z CloudKit.
- CKAsset — data jsou pod správou OS.
- CloudKit nemůže předpokládat lokální cache.

Client-Server App, JavaScript

- Je jeden z přístupů vytváření aplikace.
 - Typicky pro aplikace typu klient-server (síťová DB, stránky).
- Binárka aplikace (Swift) slouží pouze jako zavaděč JavaScript-ového programu.
- TVML — Apple's TV Markup Language.
- Zřejmě celá aplikace je mimo rámec Foundation (další služby).

Schema tv-app s TVML



Zahájení projektu v XCode

- Dvě varianty:
 - normální nativní aplikace — `singleView`, umožňuje `CoreData`, má `storyBoard`.
 - TVML aplikace.
- TVML — dvojí protokoly `AppDelegate`.
 - `TVApplicationControllerDelegate`
 - dvojí rozhraní, přeposílání zpráv.
 - start JS aplikace a řízení app z OS (background / foreground).

Nativní tvOS Aplikace

- UIApplicationDelegate — podle iOS.
- CoreData, UserDefaults, CloudKit. SpriteKit.
- Kvóty.
- Bundle — zdroje.
- StoryBoard.

tvOS StoryBoard

- ViewController (UIViewController), View.
- @IBAction, @IBOutlet.
- SpriteKit View.



```
// UI !!! ViewController
class HlavniStranka: UIViewController {
    //
    @IBOutlet var lab: UILabel!
    // Event: Primary Action Triggered
    @IBAction func stiskni() {
        //
        lab.text = "udelano"
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view, typically from a
nib.
    }
}
```

Alert

```
@IBAction func stiskni() {  
    // ViewController pro obrazku s hlasenim  
    let alertController = UIAlertController(title: "Hello",  
                                           message: "Nemackej me",  
                                           preferredStyle: .alert)  
  
    // tlacitko a jeho akce  
    let cancelAction = UIAlertAction(title: "Cancel",  
                                     style: .cancel)  
    {  
        action in print("Stisk Cancel")  
    }  
  
    // tlacitko a jeho akce  
    let okAction = UIAlertAction(title: "OK",  
                                 style: .default)  
    {  
        action in print("Stisk OK")  
    }  
  
    // ...  
    alertController.addAction(cancelAction)  
    alertController.addAction(okAction)  
    // zobrazeni VC  
    self.present(alertController, animated: true) {  
        // po zobrazeni VC proved:  
    }  
}
```

Hello

Nemackej me

OK

Cancel

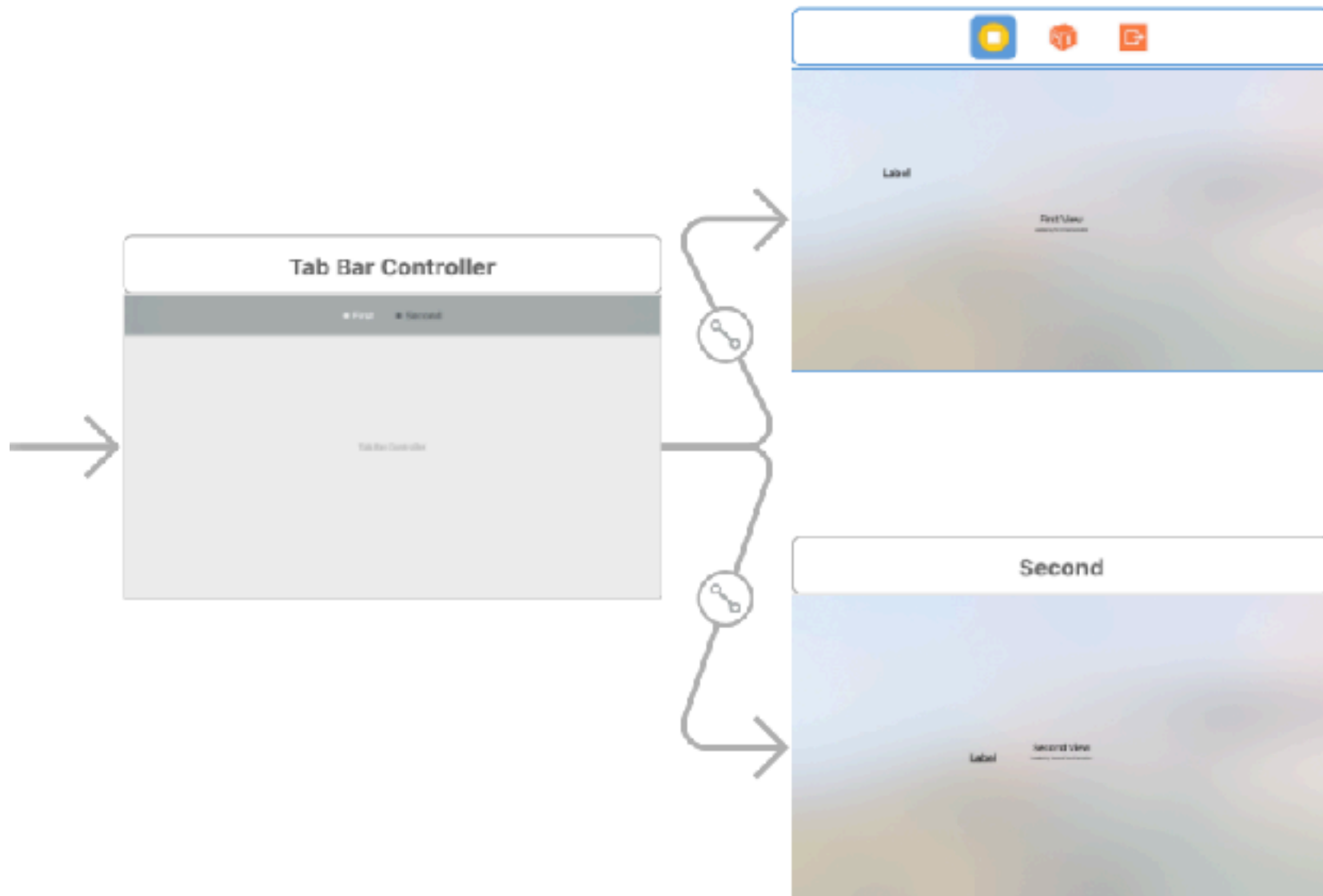
Focus engine v TV

- Není touch ovládání, není klávesnice, myš.
Máme dálkový ovladač.
- Focus — jeden prvek je "focused" (zvýrazněný).
Typicky se jeho view zvětší, zvýrazní.
- Přecházení mezi focused se děje na základě stromu Views, *bez explicitní topologie*.
 - ve všech směrech.

Koncepty UI v tv-app

- TVML. SpriteKit.
- TabBar app.
- Tabulkový VC.
- Collection VC.
- Katalogový koncept — tabulka a collectionView.

Tabbed app



Tabulka v tvOS

- UITableViewController, DataSource, apod.
 - prototyp UITableViewCell.
 - Koncept totožný s iOS.
- Focusing: UIFocusUpdateContext
 - nextFocusedItem — znamená aktuální zvýrazněný prvek
 - canFocusItem... —> Bool

```

class TVMyTable: UITableViewController {
    //
    override func numberOfSections(in tableView: UITableView) -> Int {
        return 1;
    }
    //
    override func tableView(_ tableView: UITableView,
                            numberOfRowsInSection section: Int) -> Int
    {
        //
        return 100;
    }
    //
    override func tableView(_ tableView: UITableView,
                            cellForRowAt indexPath: IndexPath) ->
UITableViewCell
    {
        //
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell",
for: indexPath)

        //
        cell.textLabel?.text = "Ahoj \(indexPath.row)"

        //
        return cell
    }
}

```

didUpdateFocus

```
override func didUpdateFocus(in context: UIFocusUpdateContext,  
                             with coordinator:  
UIFocusAnimationCoordinator)  
{  
    if let nf = context.nextFocusedItem {  
        print("Next \(nf)")  
        //  
        if let nfi = context.nextFocusedView as? UITableViewCell {  
            //  
            nfi.textLabel?.text = "focused"  
        }  
  
        //  
        if let pfi = context.previouslyFocusedView as? UITableViewCell {  
            //  
            pfi.textLabel?.text = "was focused"  
        }  
    }  
}
```

focused

Ahoj 1

Ahoj 2

Ahoj 3

Ahoj 4

Ahoj 5

Ahoj 6

Ahoj 7

Ahoj 8

Ahoj 9

Ahoj 10

Ahoj 11

UINavigationController

- V tvOS existuje.
- push / pop, storyboard.
- Horní lišta pro tlačítka, obvyklý postup implementace tlačítek.
- Tlačítka jsou "fokusevatelná".

NavigationItem



Stiskni

was focused

Ahoj 1

Ahoj 2

Ahoj 3

Ahoj 4

Ahoj 5

Ahoj 6

Ahoj 7

Ahoj 8

Ahoj 9

Ahoj 10

Ahoj 11

```
class TVMyTable: UITableViewController {
    //
    @objc func akce(_:AnyObject) {

    }
    //
    override func viewDidLoad() {
        //
        super.viewDidLoad()

        //
        let but = UIBarButtonItem(barButtonSystemItem: .action, target:
self, action: #selector(akce(_)))
        let butA = UIBarButtonItem(title: "Stiskni", style: .done, target:
self, action: #selector(akce(_)))

        //
        self.navigationItem.rightBarButtonItem = [butA, but]
    }
```

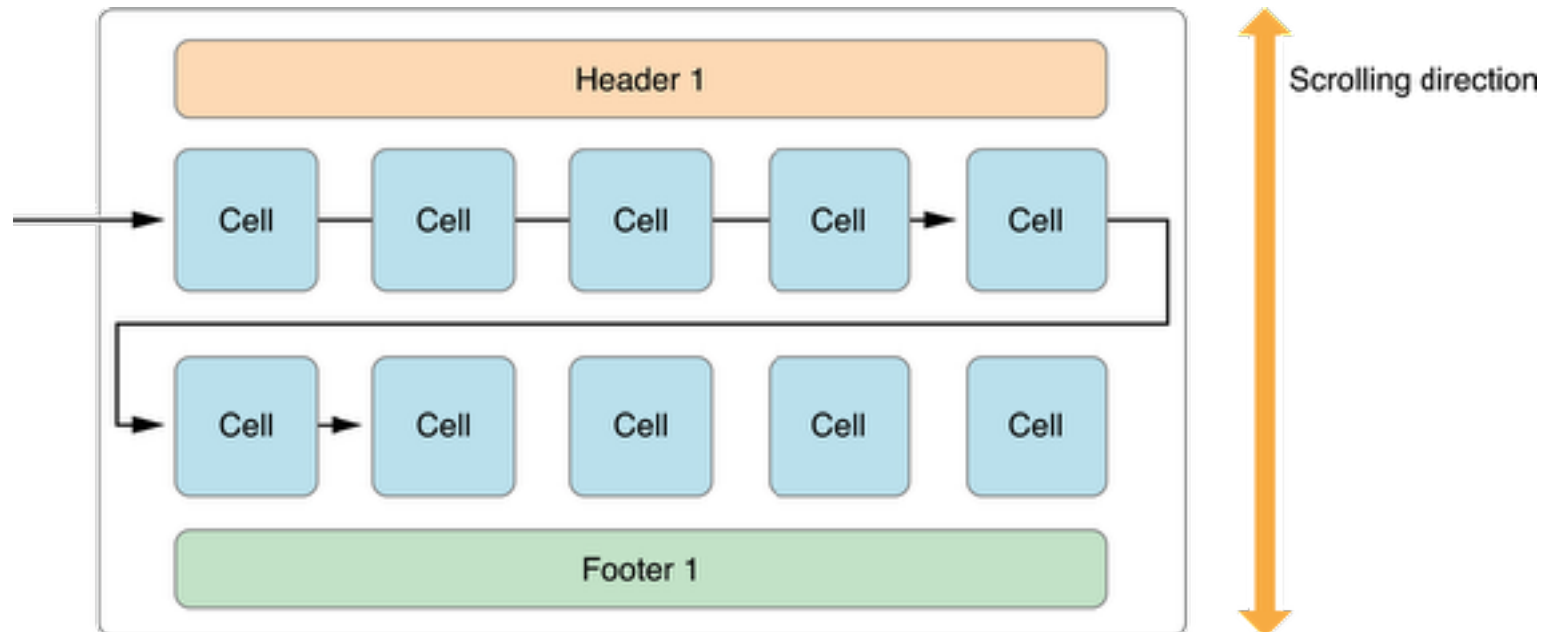
Collection View

- Geometrie
 - Další součást dataSource (layout) — generování metadat o geometrii pro buňky collection. Má podporovat i focusing (záznamy podle zadaného Rect).
 - Obvyklá / Implicitní (Flow) — maticová,
 - Obecná / Custom — pak je nutno implementovat UICollectionViewLayout.
- Posuv — buď horizontální nebo vertikální.
 - zarovnání na frame.



Collection view jako Flow

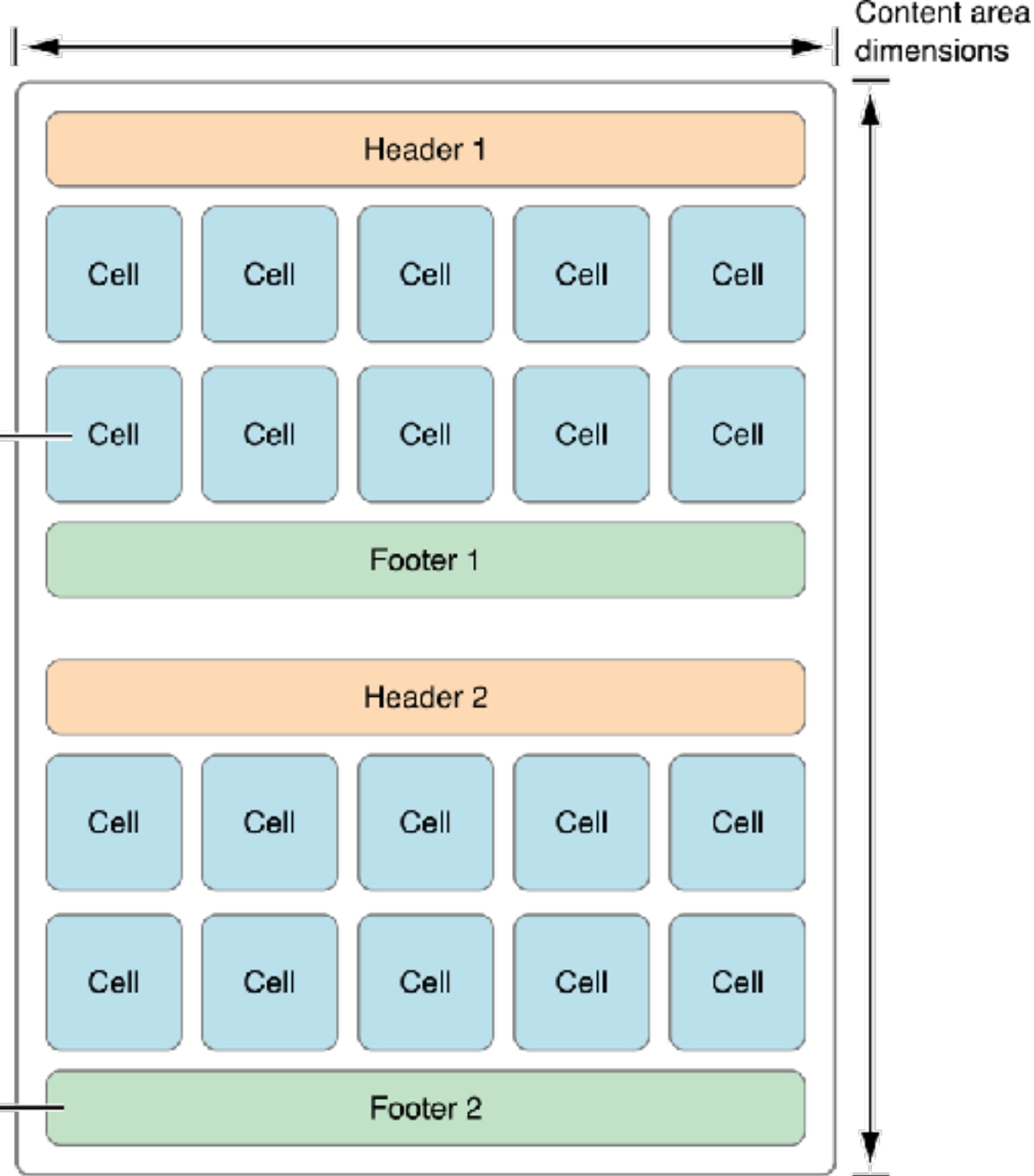
- Koncept totožný s TableView. DataSource:
 - sekce, počet buněk na sekci, generování cell.
- Animace pro focus na buňku.



View Layout Information

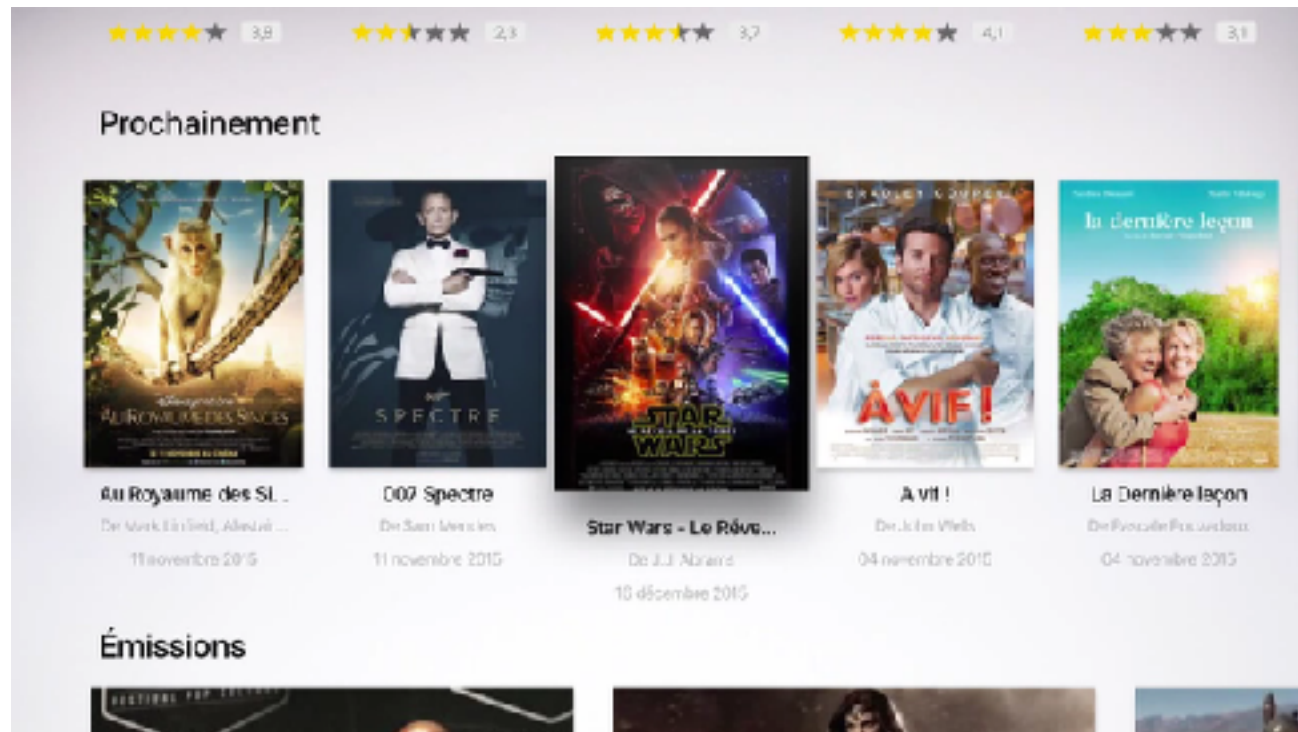
Cell (0,6):
Size: (50, 50)
Center: (40,145)
Alpha: 1.0
...

Footer 2:
Size: (200, 20)
Center: (115,400)
Alpha: 1.0
...



Katalogové UI v tvOS (iOS)

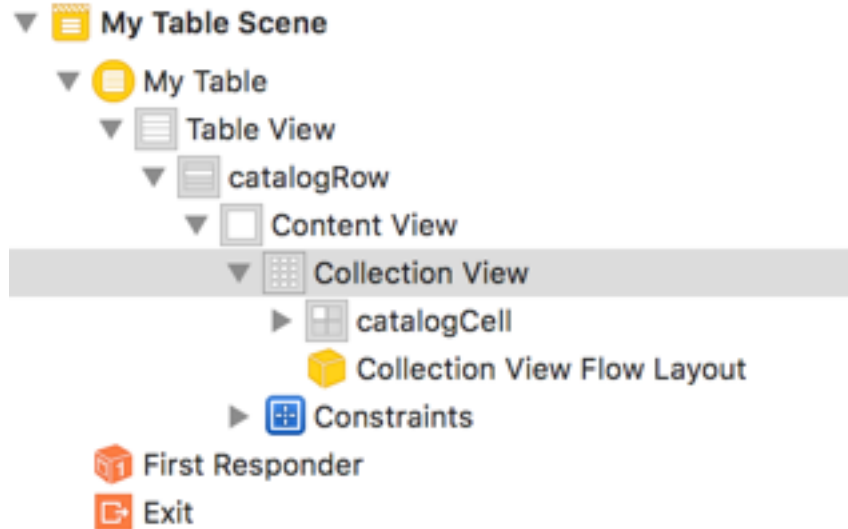
- Koncept "focus".
- Kombinace TableView a CollectionView.
- Základ většiny aplikací na AppleTV.



Katalogové UI, konstrukce

- TableView definuje vertikální dimenzi a scrollování nahoru / dolů.
- collectionView definuje horizontální dimenzi.
- TableViewCell obsahuje collectionView pro řádek *a je jeho dataSourceem*.
- 2D datový model — sekce, obsahy sekcí.

Storyboard model



Outlets	
dataSource	* catalogRow
delegate	* catalogRow
prefetchDataSource	

Outlet Collections	
gestureRecognizers	

Referencing Outlets	
New Referencing Outlet	

Referencing Outlet Collections	
New Referencing Outlet Collection	

Collection View	
Items	1
Layout	Flow
Scroll Direction	Horizontal
Accessories	☐ Section Header ☐ Section Footer
Prefetch	☐ Prefetching Enabled

Katalog, implementace Tabulky

```
// VC pro Tabulku-> radky katalogu
class TVCatalogTable: UITableViewController {
    // !!!
    override func tableView(_ tableView: UITableView,
                             canFocusRowAt indexPath: IndexPath) -> Bool {
        // !!! bunka tabulky (radek)
        // neni samostatne focusovatelna
        return false
    }
    //
    override func tableView(_ tableView: UITableView,
                             cellForRowAt indexPath: IndexPath) ->
UITableViewController
    {
        //
        let cell = tableView.dequeueReusableCell(withIdentifier:
"catalogRow", for: indexPath) as! TVCatalogRow

        //
        return cell
    }
}
```

```

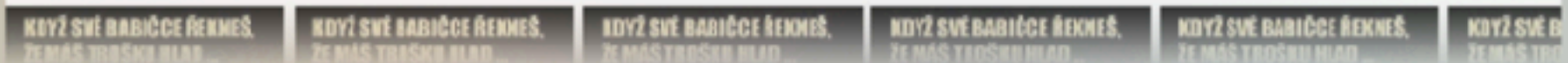
// Table-Cell -> CollectionView
class TVCatalogRow: UITableViewCell, UICollectionViewDataSource,
    UICollectionViewDelegate{
    // def
    func numberOfSections(in collectionView: UICollectionView) -> Int {
        return 1
    }
    // podle modelu
    func collectionView(_ collectionView: UICollectionView,
        numberOfItemsInSection section: Int) -> Int
    {
        //
        return 20
    }
    //
    func collectionView(_ collectionView: UICollectionView,
        cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell
    {
        //
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier:
"catalogCell", for: indexPath) as! TVCatalogCell

        //
        cell.image.image = __obrazek!

        //
        return cell
    }
}

```

Demo Katalogového VC



Buňka, animace při focus

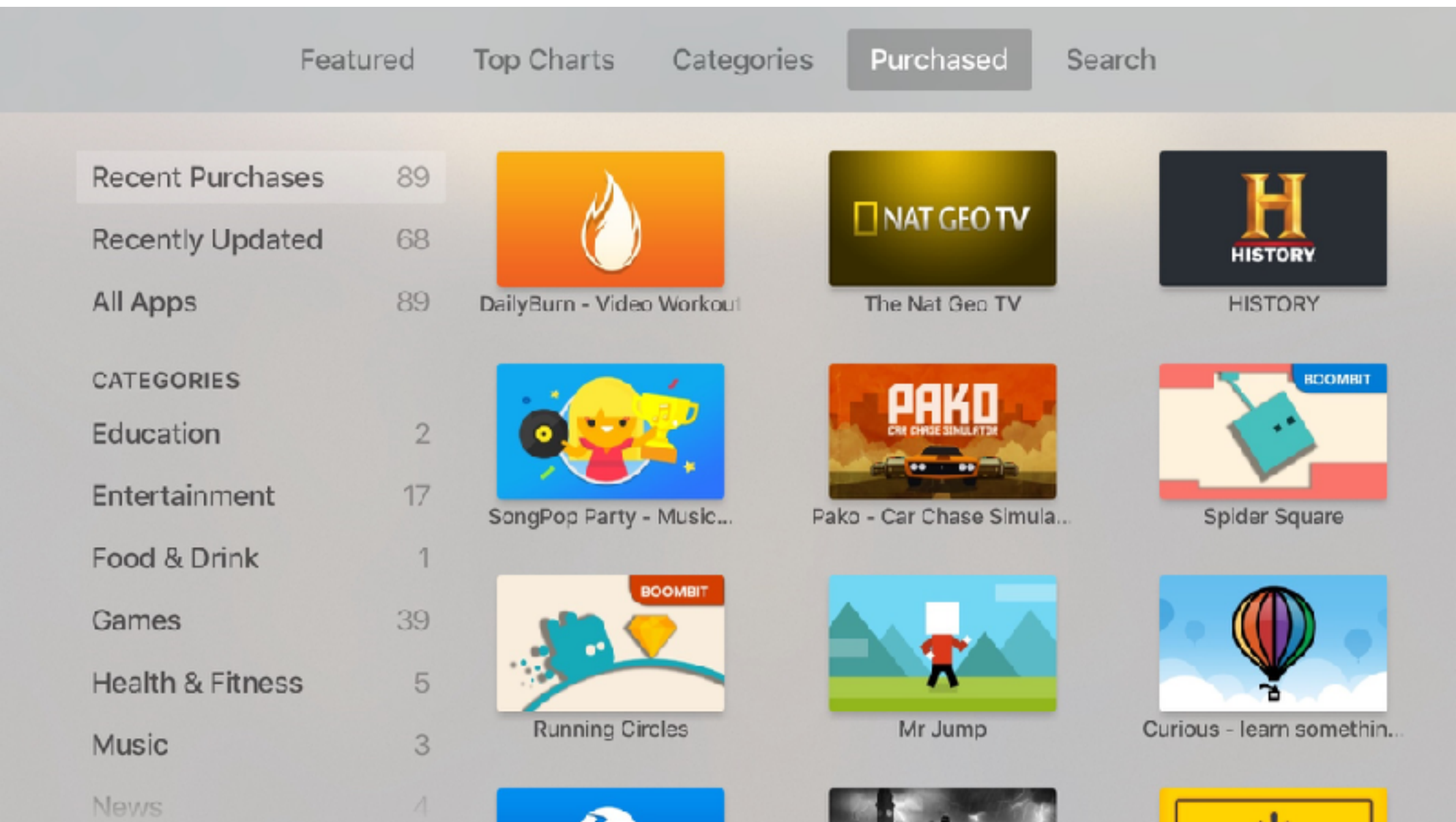
```
func collectionView(_ collectionView: UICollectionView, didUpdateFocusIn
context: UICollectionViewFocusUpdateContext, with coordinator:
UIFocusAnimationCoordinator) {

    if let previousIndexPath = context.previouslyFocusedIndexPath,
        let cell = collectionView.cellForItem(at: previousIndexPath) {
        cell.contentView.layer.borderWidth = 0.0
        cell.contentView.layer.shadowRadius = 0.0
        cell.contentView.layer.shadowOpacity = 0
    }

    if let indexPath = context.nextFocusedIndexPath,
        let cell = collectionView.cellForItem(at: indexPath) {
        cell.contentView.layer.borderWidth = 10.0
        cell.contentView.layer.borderColor = UIColor.black.cgColor
        cell.contentView.layer.shadowColor = UIColor.black.cgColor
        cell.contentView.layer.shadowRadius = 20.0
        cell.contentView.layer.shadowOpacity = 0.9
        cell.contentView.layer.shadowOffset = CGSize(width: 0, height:
0)

        //
        collectionView.scrollToItem(at: indexPath, at:
[.centeredHorizontally, .centeredVertically], animated: true)
    }
}
```

Komplexní aplikace

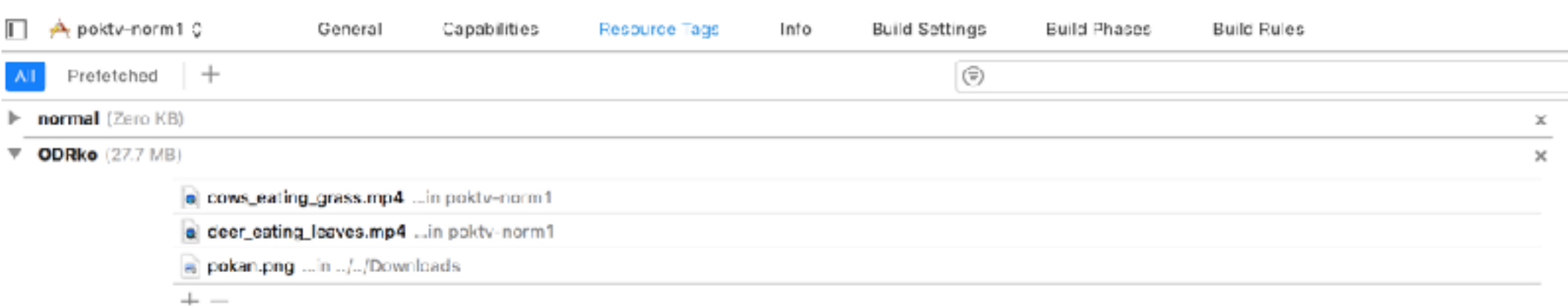


On-demand resources (ODR)

- Je to část app-bundle, která se nestahuje s aplikací, ale až dynamicky v průběhu potřeby
- Bundle.main — hlavní bundle aplikace
- Aplikace generuje žádost na ODR zdroje,
 - tvOS stáhne zdroje a doplní bundle,
 - zpráva, že už ODR nejsou potřeba, tvOS zdroje smí uvolnit.
- Zdroje mají v XCode přidělen "resourceTag"

ResourceTag

- Zdroje bez tagu jsou automaticky v Bundle.main
- <https://www.raywenderlich.com/173062/demand-resources-tvos-tutorial-2>



Stav ODR zdrojů

- Not downloaded, Downloaded, In Use, Purged
- Aplikace nepříkazuje SMAZAT, pouze o zdroj žádá, zdroj přestává potřebovat

On Demand Resources

Tag	Size	Status
ODRko	28.7 MB	Purged

Open Files

Descriptor	Type	Device	Size/Offset	Inode	Path
0r	CHR	50 331 850	0	308	/dev/null
1u	CHR	268 435 457	0	3 171	/dev/tty001
2u	CHR	268 435 457	0	3 171	/dev/tty001
3r	REG	0	0	8 693 921 529	/Users/hrubym/Library/Developer/CoreSimulator/Devices/B1C27C6F-B78E-4B96-99D0-63427857147
5r	REG	0	0	8 590 192 531	/System/Library/ColorSync/Resources/ColorTables.data

Žádost o zdroj

```
class ODRka {
    // doba zivota "req" urcuje dobu zivota zadanych zdroju
    var req: NSBundleResourceRequest? = nil

    //
    static let shared = ODRka()
    // ref na ODR zdroj po stazeni
    var pokan: UIImage? = nil

    // zadost o ODR
    func startRequest() {
        //
        req = NSBundleResourceRequest(tags: ["ODRko"])
        //
        req!.loadingPriority = NSBundleResourceRequestLoadingPriorityUrgent
        // zahajeni download
        req!.beginAccessingResources { (error) in
            // blok se mi spusti v GlobalQueue
            if error == nil {
                // dale do GlobalQueue zadam pristup na bundle
                DispatchQueue.global().async {
                    //
                    self.pokan = UIImage(named: "pokan.png")
                }
            }
        }
    }
}
```

Uvolnění ODR

- Zdroj se uvolní, když ho BRR přestane referencovat, tj. když objekt BRR zanikne.

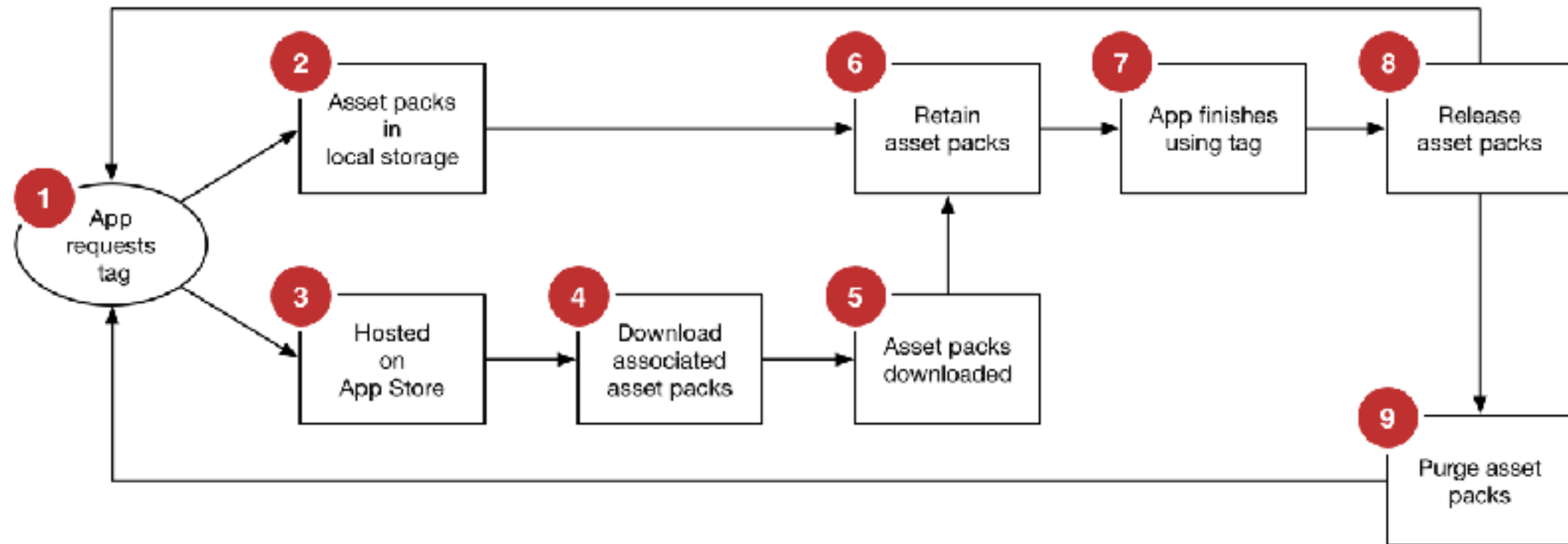
```
class ODRka {  
    // doba zivota "req" urcuje dobu zivota zadanych zdroju  
    var req: NSBundleResourceRequest? = nil  
    //  
    func freeRequest() {  
        // req?.endAccessingResources()  
        if let _req = req {  
            //  
            _req.endAccessingResources()  
        }  
  
        // explicitni dealokace  
        req = nil  
    }  
}
```

Persistence ODR žádostí

- Obyčejný lifecycle BRR: vznikne, download, přístupný ODR, ukončení požadavku na zdroj, dealokace BRR. Pak smí tvOS ODR smazat, ale nemusí.
- Opakované spuštění aplikace. Aplikace zůstává na pozadí (drží BRR).
- Když se aplikace ukončí: ODR se zřejmě uvolní pro smazání. Priorita (preservation).

Lifecycle ODR

- Aplikace by zřejmě měla při každém spuštění předpokládat potřebu downloadu, tj. procházet cyklem žádostí.



tvOS app podle TVML

- TVML = Television Markup Language (Apple)
- <https://www.raywenderlich.com/114886/beginning-tvos-development-with-tvml-tutorial>
 - kód generovaným XCodem raději tentokrát ne
- Bude třeba nějaký server
 - Desktop / client / js /
 - počáteční skript application.js

application.js

```
App.onLaunch = function(options) {  
  // 1  
  var alert = createAlert("Hello World", ""); //leaving 2nd parameter with an empty string  
  navigationDocument.presentModal(alert);  
}  
  
// 2  
var createAlert = function(title, description) {  
  var alertString = `<?xml version="1.0" encoding="UTF-8" ?>  
    <document>  
      <alertTemplate>  
        <title>${title}</title>  
        <description>${description}</description>  
      </alertTemplate>  
    </document>`  
  var parser = new DOMParser();  
  var alertDoc = parser.parseFromString(alertString, "application/xml");  
  return alertDoc  
}
```

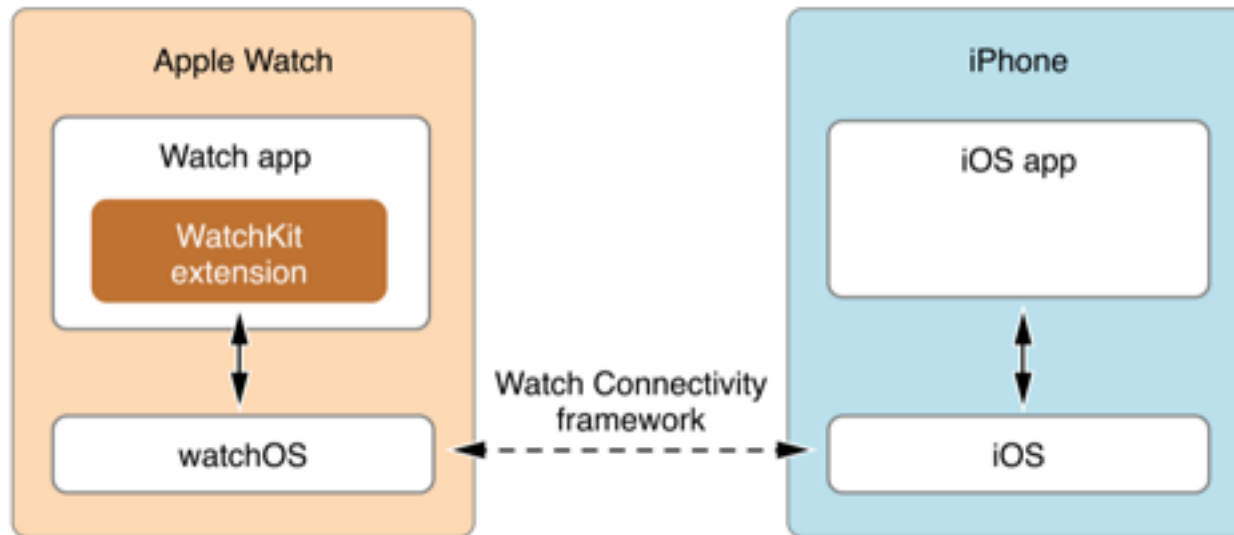
tvOS a CloudKit

- Podporovaný koncept.
- Nejspíš bez lokální cache (kvóta). Určuje to použitelnost aplikace — sledování změn v DB (diff fetch). Kompletní fetch DB s omezením na počet záznamů (řazeno např. podle času).
- Nutno prozkoumat chování CKAsset.

Závěr o tvOS

- Jaký druh aplikací tvOS podporuje — nepříliš otevřený pro zpracovávání dat (kvóty).
- Apple ID u AppleTV — v podstatě jedno-uživatelské zařízení.
 - Aplikace jako kalendář, mail apod. mají zde omezenou smysluplnost.
- Multimedia. Katalogy. Filmy. Hry (spritekitové).
- Zcela ne Web.

watchOS



Úvod

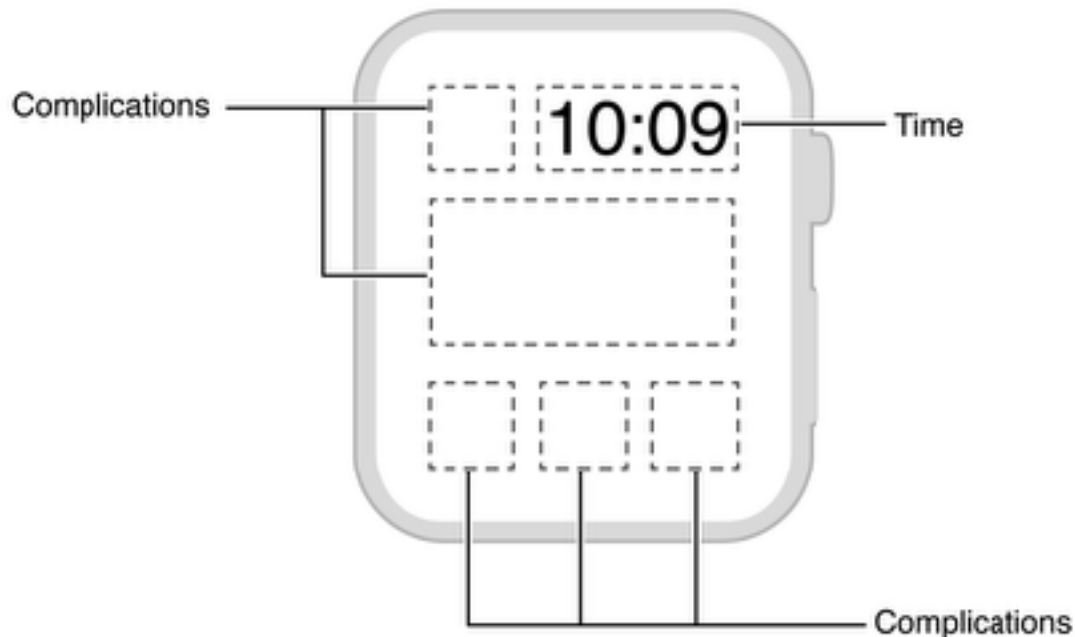
- Apple Watch — watchOS 2, 3, 4.
- Typicky jde o spojení aplikace pro iPhone a pro Watch.
 - iPhone — zajišťuje data,
 - watch — terminál pro zobrazení.
- Aplikace pracuje v režimu *Extension* párové iPhonové aplikace.
 - Koncept extensions v aplikacích (do notif. centra, tlačítko "akce").

Podoba aplikace

- Watch Extension — aplikace spojená s iOS / iPhone aplikací.
 - Watch Notifikace.
- Prostudujeme:
 - Tvorbu uživatelského rozhraní.
 - Komunikaci iOS+watchOS.
 - Přenos dat - UD, CloudKit, Messages.
- Přístupnost UIKit, Foundation — tedy ne.

Podoba aplikace

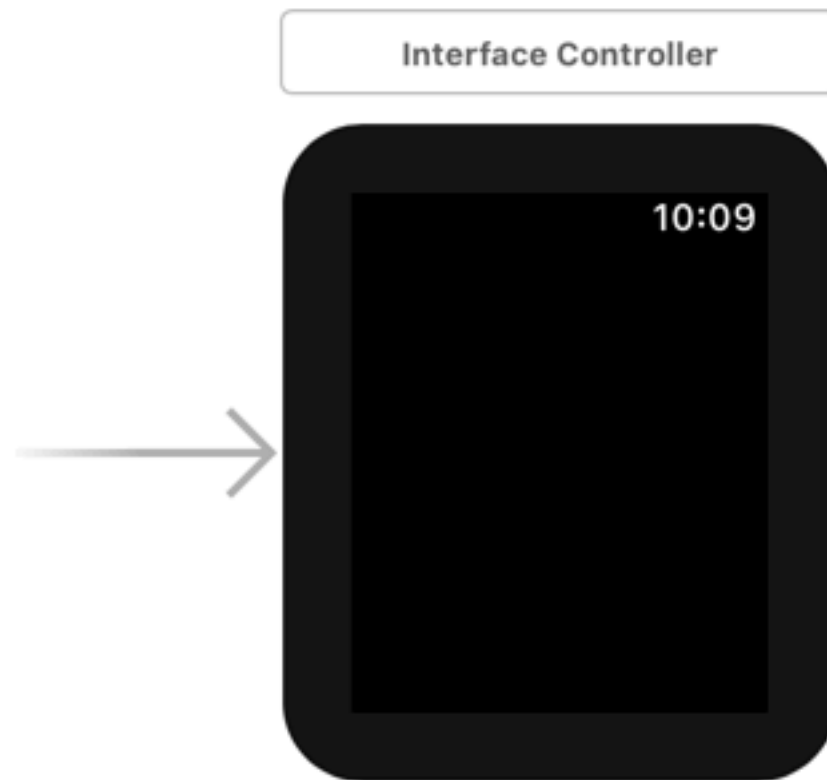
- Notifikace — specializovaný VC.
- Komplikace — komponenta "scény s hodinkami".



Sestavení v XCode

- Zvolíme watchOS app.
- Vygeneruje se:
 - aplikace pro iOS
 - WatchKit App
 - WatchKit Extension
 - popř. StoryBoard pro notifikaci.

Storyboard pro watch



Aplikace watch

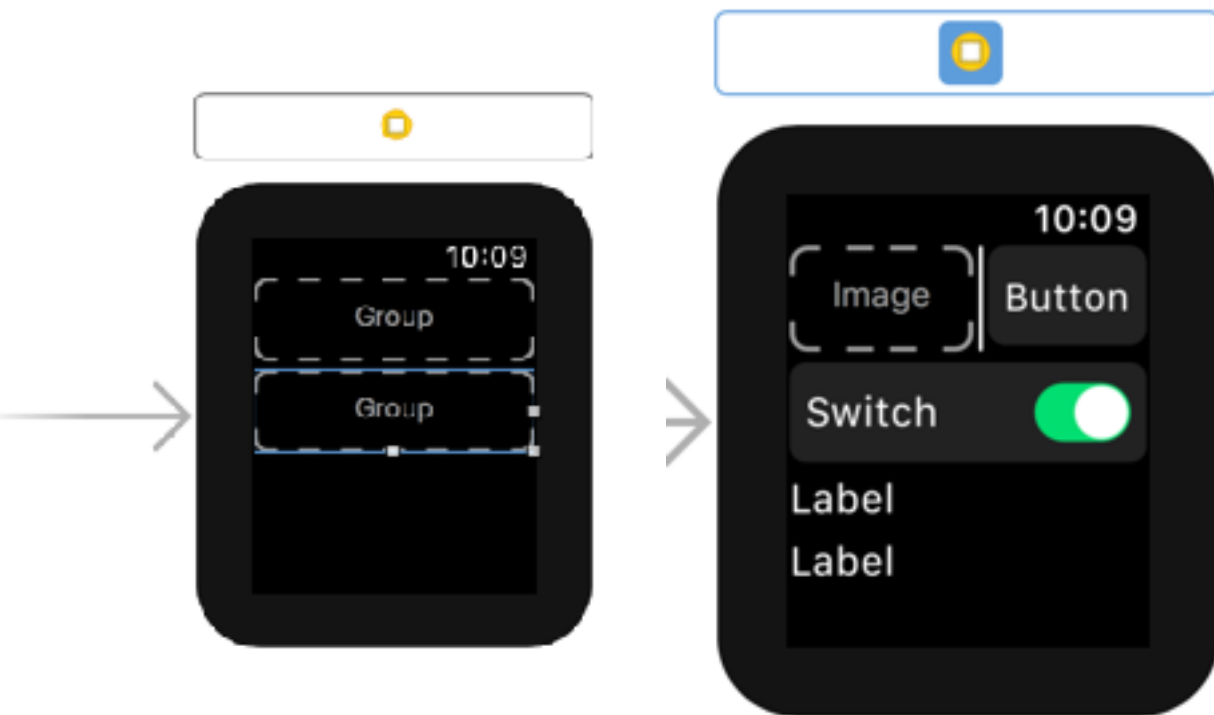
- ViewController — WKInterfaceController
 - awake
 - willActivate
 - didDeactivate
- Extension Delegate (AppDelegate)
 - WKExtensionDelegate
 - obdobné zprávy jako u iOS/macOS

WatchKit (WK)

- WKExtension — obdoba UIApplication.
- WKExtensionDelegate
- WKInterfaceController — každá scéna (obrazovka) má svůj VC, teď IC.
- Outlety jsou WK*
 - zjevně odlišná implementace od UIKitu, iOS.

Geometrie obrazovky

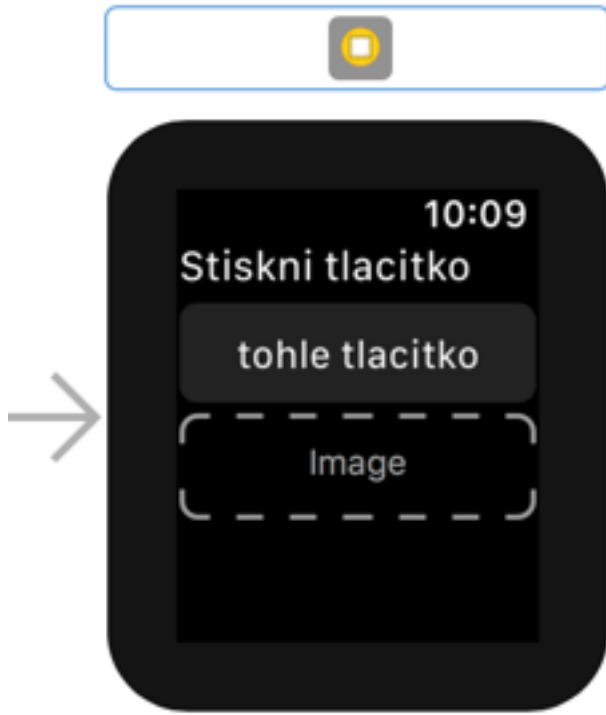
- V zásadě opět vede na tabulku:
 - posloupnost Group — vertikální/horizontální layout — tvoří formulář UI,
 - Outlets — label, button, pickery, obrázky. Tabulka.



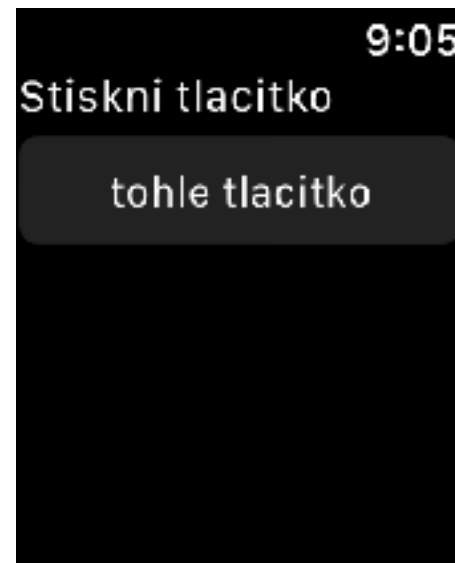
Základy UI

- ViewController, scéna, StoryBoard.
- Ovládací prvky — WK — Label, Button, ...
 - `setText(_ s: String)`
- IBOutlet, IBAction

Demo základů UI

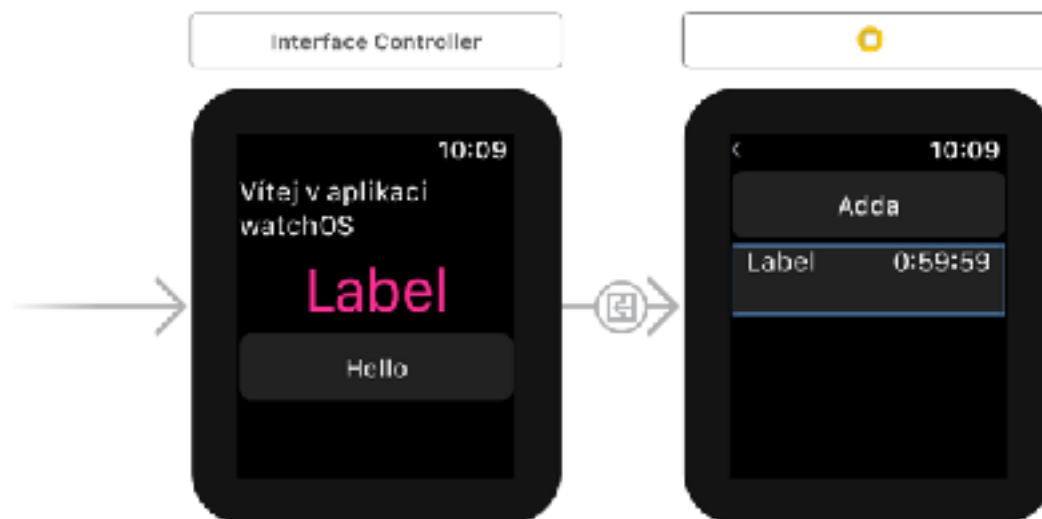


```
class WWHlavni: WKInterfaceController {  
    //  
    @IBOutlet weak var pic: WKInterfaceImage!  
    //  
    @IBAction func tlacitko1() {  
        //  
        if let _pic = UIImage(named: "pokan.png") {  
            //  
            pic.setImage(_pic)  
        }  
    }  
}
```



NavigationController

- Je implicitní. Segue — show (push).



Tabulky

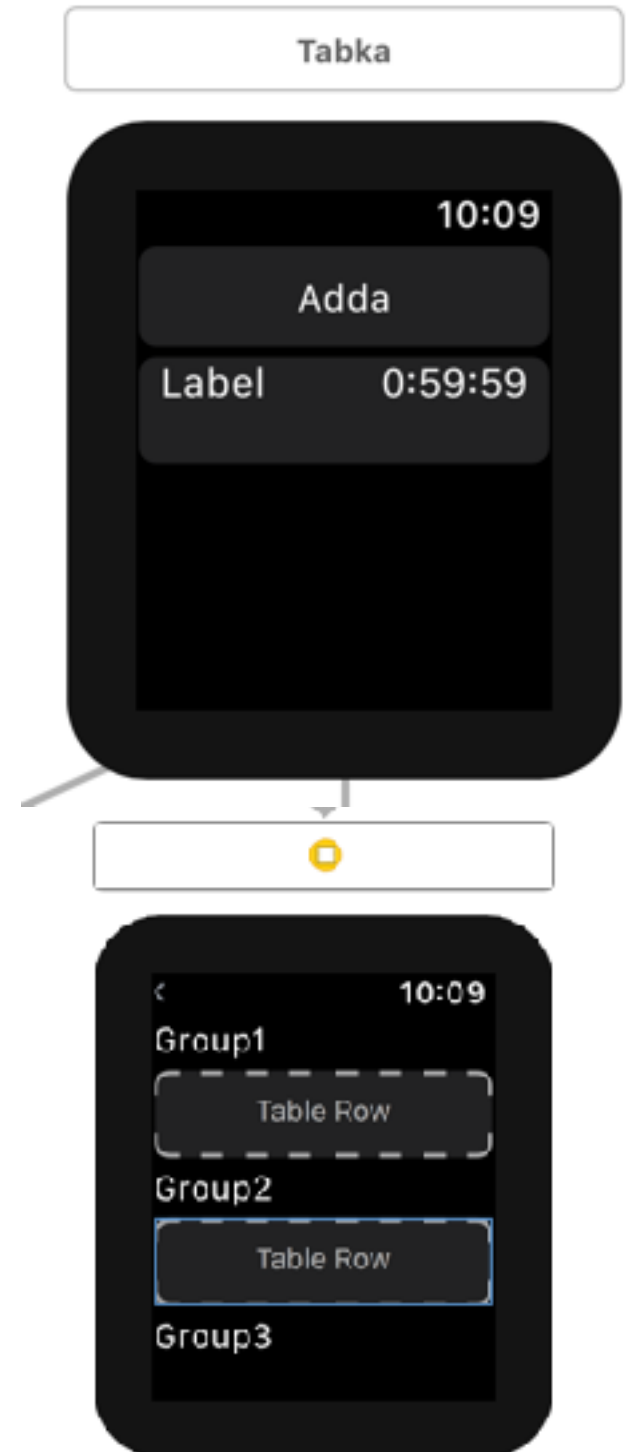
- RowController — obdoba UITableViewCell
 - identifikován stringem ve StoryBoard.
- Předpokládá se heterogenita řádků (rowContr).
- Zjednodušené rozhraní:
 - `setNumberOfRows(n, type:String)`
 - `insertRow(idx, type)`.
- Objekt buňky instancuje Tabulka při vložení.

Tabulky: rowController

- Třída odvozená od NSObject.
 - outlets ovládacích prvků
 - ve StoryBoard pojmenovaná
 - instance vytváří tabulka insertRow, setNumberOf

Demo

```
class TabkaRow: NSObject {  
    //  
    @IBOutlet var text: WKInterfaceLabel!  
    @IBOutlet var tima: WKInterfaceTimer!  
    //  
    func runt() {  
        //  
        tima.start()  
    }  
    // ??? musi to byt takto  
    override init() {  
        super.init()  
    }  
}
```



Demo, TableVC

```
class Tabka: WKInterfaceController {
    //
    @IBOutlet var table: WKInterfaceTable!
    // datovy model...
    var datas: [String] = ["jedna", "dva", "tri"]
    //
    override func awake(withContext context: Any?) {
        //
        super.awake(withContext: context)
        //
        table.setNumberOfRows(datas.count, withRowType: "hehe")
        //
        for i in 0..
```

Datová komunikace

- iOS prý synchronizuje iPhone-app UD do watchOS-app.
- Cloud Key-Value Storage — podle dokumentace je dostupné, realně není.
- WatchConnectivity.
- CloudKit — ???.

WatchConnectivity

- Datové rozhraní pro párovou komunikaci iPhone+Watch.
 - Základem existence watchApp je být v páru s iPhonem.
 - od watchOS3 už smí na CloudKit samostatně (sítě).
- WCSSession, WCSSessionDelegate
- sendMessage, delegate: didReceiveMessage

WC-conn, demo Message, iOS

```
class ViewController: UIViewController, WCSessionDelegate {
    //
    func session(_ session: WCSession,
                 activationDidCompleteWith activationState:
WCSessionActivationState,
                 error: Error?)
    {
        //
        print("Session activated")
    }
    //
    override func viewDidLoad() {
        super.viewDidLoad()

        //
        if WCSession.isSupported() {
            //
            WCSession.default.delegate = self
            //
            WCSession.default.activate()
        }
    }
}
```

WC-conn, demo Message, iOS

```
// iOS part
@IBAction func tlacko() {
    //
    if WCSSession.default.isPaired, WCSSession.default.isReachable {
        // datova zprava [String:Any]
        let _dt = [ "cas" : Date().description ]
        // zakodovat, co nejdriv odeslat
        // lze cekat na odpoved
        //
        WCSSession.default.sendMessage(_dt, replyHandler: nil,
errorHandler: nil)
    } else {
        print("Notpaired")
    }
}
```

WC-conn, demo Message, watch

- Zpráva sendMessage

- z iOS do watchOS neprobudí watch

- z watchOS do iOS probudí iPhone aplikaci

```
class InterfaceController: WKInterfaceController, WCSSessionDelegate {  
    //  
    @IBOutlet var cas: WKInterfaceLabel!  
    // procedura pro zpracovani prichodi zpravy  
    func session(_ session: WCSSession,  
                didReceiveMessage message: [String : Any])  
    {  
        //  
        print("Message")  
  
        //  
        if let _val = message["cas"] as? String {  
            //  
            cas.setText(_val)  
        }  
    }  
}
```

WCSession, obousměrná komun.

- Odesílatel zprávy může obdržet odpověď (opět ve formátu key-value).
- Aplikace si nad WCSession může zřídit interní protokol (metadata ve zprávách).
Centralizovaný singleton pro správu zpráv.
- pokyny k plánování notifikací.

WConn, interní aplikační protokol

- [String: Any], key: "descr" — adresát zprávy

```
// globalni API pro WConn
class WConnManager: NSObject, WSessionDelegate {
    //
    static let shared = WConnManager()
    //
    var working: Bool { return WSession.isSupported &&
WSession.default.isReachable }
    //
    func startup() {
        //
        if WSession.isSupported() {
            //
            WSession.default.delegate = self
            WSession.default.activate()
        }
    }
    // funkce pro poslani zpravy
    // obsluha protokolu WSessionDelegate
}
```

Notifikace

- Sestavit notifikaci.
- Naplánovat — trigger. Notifikační požadavek
 - unikátní ID požadavku
- OS rozhodne, zda-li zobrazí notifikaci na telefonu nebo hodinkách ??
- Poslání notifikace do hodinek —
WCSession.message, lokálně naplánovat
- Push.



Zřízení notifikace

```
@IBAction func notifyButton() {
    // objekt notifikace
    let content = UNMutableNotificationContent()
    // nějaký obsah
    content.title = "Ukaz se"
    content.body = "Tady to zazní... "
    content.sound = UNNotificationSound.default()
    // unikátní ID pro každou notifikaci
    let _uuid = UUID().uuidString
    // spouštěcí podmínka, napr. timer
    let trigger = UNTimeIntervalNotificationTrigger.init(timeInterval:
5, repeats: false)
    // zřízení požadavku na notifikaci
    let request = UNNotificationRequest.init(identifier: _uuid, content:
content, trigger: trigger)
    //
    let center = UNUserNotificationCenter.current()
    //
    center.add(request) { (error) in
        print(error)
    }
}
```

Závěr

- Jaké aplikace se hodí na TV a Watch?
 - Fitness pro Apple Watch? ;)
- Zbývá probrat: macOS, přehled knihoven, UserExperience