

Kliensoldali technológiák

Bevezetés

Dr. Kővári Bence | kovari.bence@vik.bme.hu

Követelmények

Kliensoldali technológiák

Dr. Kővári Bence, BME, AUT

kovari.bence@vik.bme.hu

Elérhetőség

- Kővári Bence
 - > QB 221
 - > Kovari.bence@vik.bme.hu
- Tárgyhonlap
 - > <https://edu.vik.bme.hu> → Kliensoldali technológiák



Oktatási forma

- Előadás
 - > minden hétfő 8:15 (Teams)
 - > A félév során lesznek előre rögzített és lesznek élő előadások is.
 - > Az élő előadásokról felvételei, vagy azokkal egyenértékű szerkesztett videók utólag megtekinthetőek lesznek
 - > Moodle kvíz minden előadáshoz! Kötelező? 
- Gyakorlat
 - > Otthon, **önállóan** megoldandó feladatok Moodle-ben kiírva
 - > Beadási határidő Neptun csoporttól és naptári ünnepektől függetlenül, mindig páros hét vasárnapja (2,4,6,8,10,12)



cloud.bme.hu

Oktatási forma

- Szünetek, áthelyezések
 - > Március 15. (Nemzeti ünnep) – előadás elmarad
 - > Április 5. (Húsvét) – előadás elmarad

Számonkérések

- Gyakorlatok
 - > Jegyzőkönyv

- > Részvétel
 - 2 gyakorlat hagyható ki
 - Gyakorlat pótlására nincs lehetőség

Számonkérések (távolléti oktatáshoz igazítva)

- ZH → elmarad

• Házi feladat

- > A tárgyhonlapon meghirdetett feladatok valamelyikét, kell teljesíteni
- > Lehet saját feladatot is hozni, ezt a gyakorlatvezetővel egyeztetni kell
- > Bemutatás: utolsó két héten
- > Pótbeadás: pótlási héten
- > Pontozás: 0-40 pont, 50%-ot el kell érni az aláíráshoz

Osztályozás

Megszerezhető pontok

ZH	20 pont
Házi	40 pont
Vizsga	60 pont
Összesen	100 pont

Ponthatárok

0- 49 pont	1
50- 59 pont	2
60- 69 pont	3
70- 84 pont	4
85- 100 pont	5

Tematika

Szoftver-technikák

C#

Tervezési minták

Adatvezérelt rendszerek

EF

WebAPI

JSON

Mobil- és webes szoftverek

Android

HTML

JavaScript

Kliens oldali technológiák

XAML

Xamarin

TypeScript

Tematika

1. Kliens oldali alkalmazások (Dr. Kővári Bence)

2-5. Grafikus felhasználói felület, XAML (Albert István)

6. Multiplatform fejlesztés, Xamarin (Albert István)

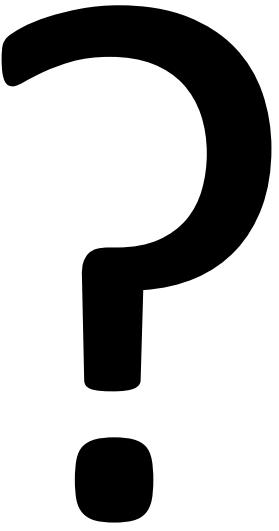
7. Ergonómia (Dr. Kővári Bence)

8-9. TypeScript (Szabó Gábor)

10-11. Angular (Szabó Gábor)

12. Webes keretrendszerök (Szabó Gábor)

Kliens oldali technológiák



Bevezető

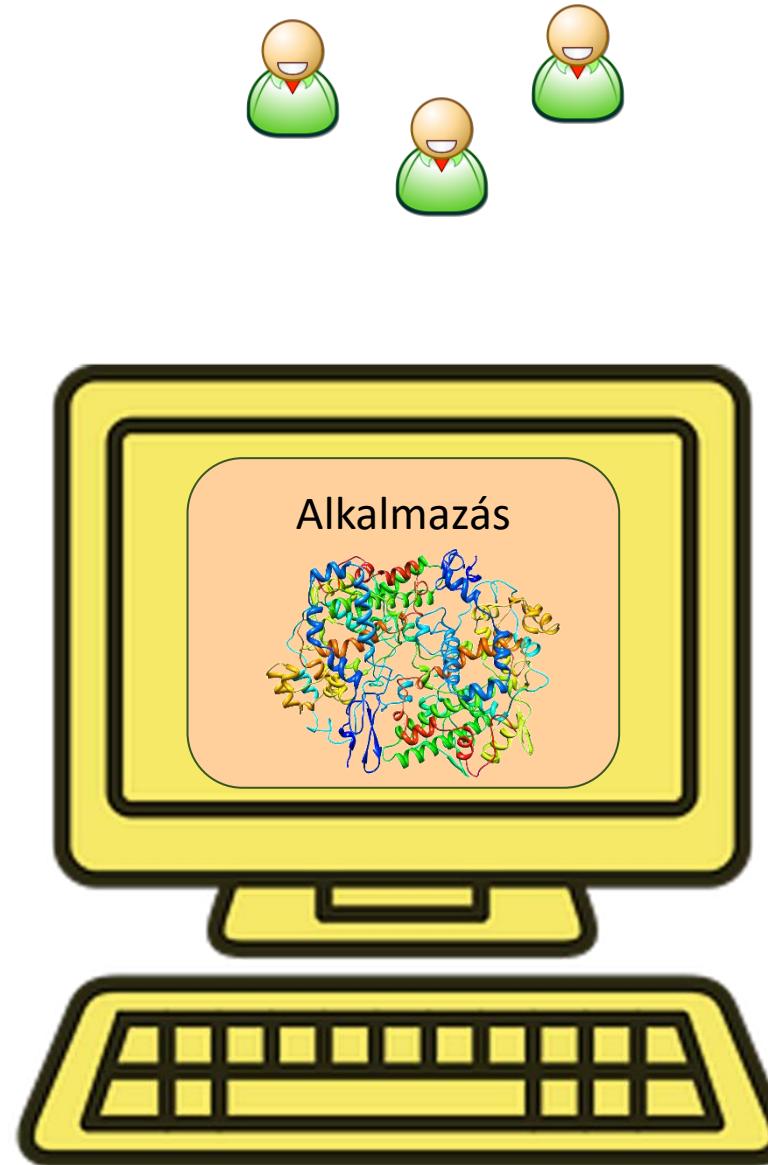
Kliensoldali technológiák

Dr. Kővári Bence, BME, AUT

kovari.bence@vik.bme.hu

Kliensoldali szoftver?

~Egy olyan szoftver, mely a felhasználó eszközén fut, s mellyel a felhasználó közvetlen interakcióba lép.



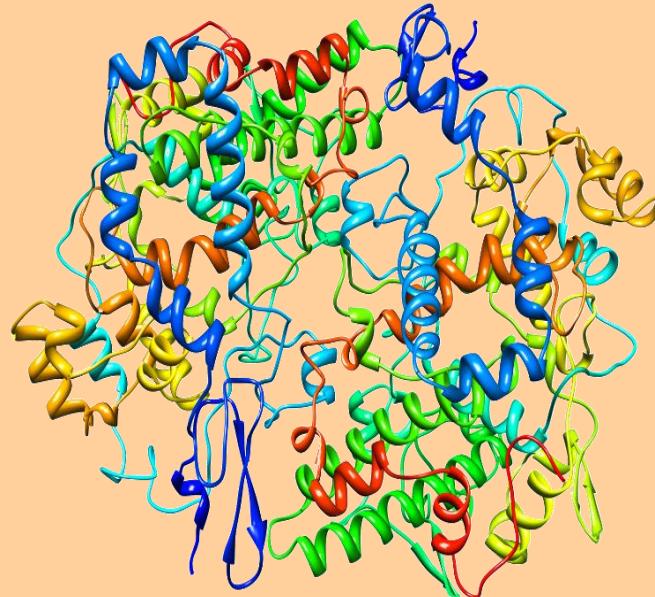
„Klasszikus” monolit alkalmazás



Egy futtatható fájl

Adatkezelés, UI kezelés,
üzleti logika stb. mind
egyben

Alkalmazás



Többrétegű alkalmazás

Erős kohézió

A logikailag összekapcsolódó funkciókat egy-egy modulba, a modulokat egy-egy rétegbe koncentráljuk

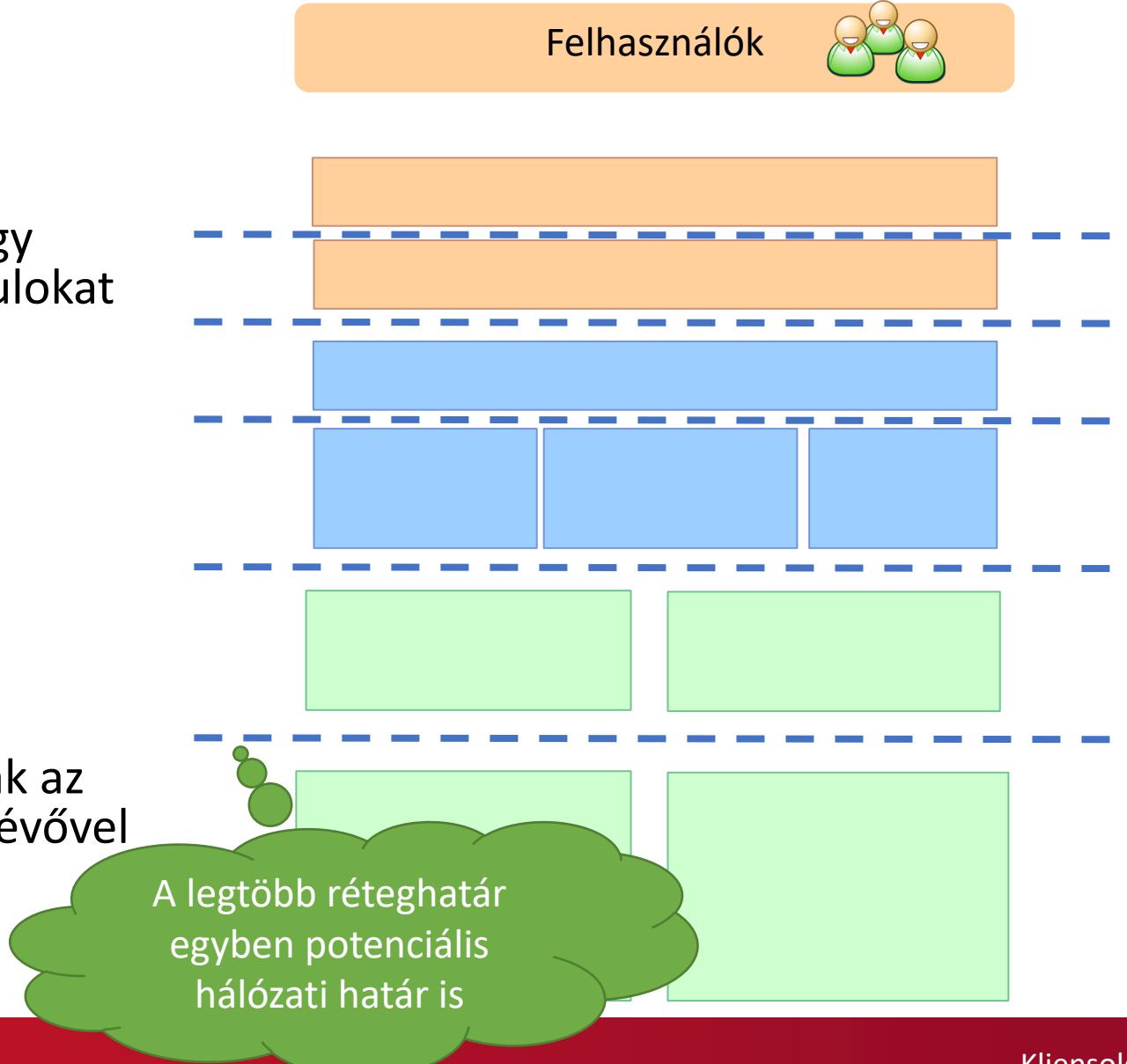
Gyenge csatolás

Modulok közti függőségek visszaszorítása

Réteges szerkezet

Minden réteg csak az alatta és fölötté lévővel kommunikál

Felhasználók

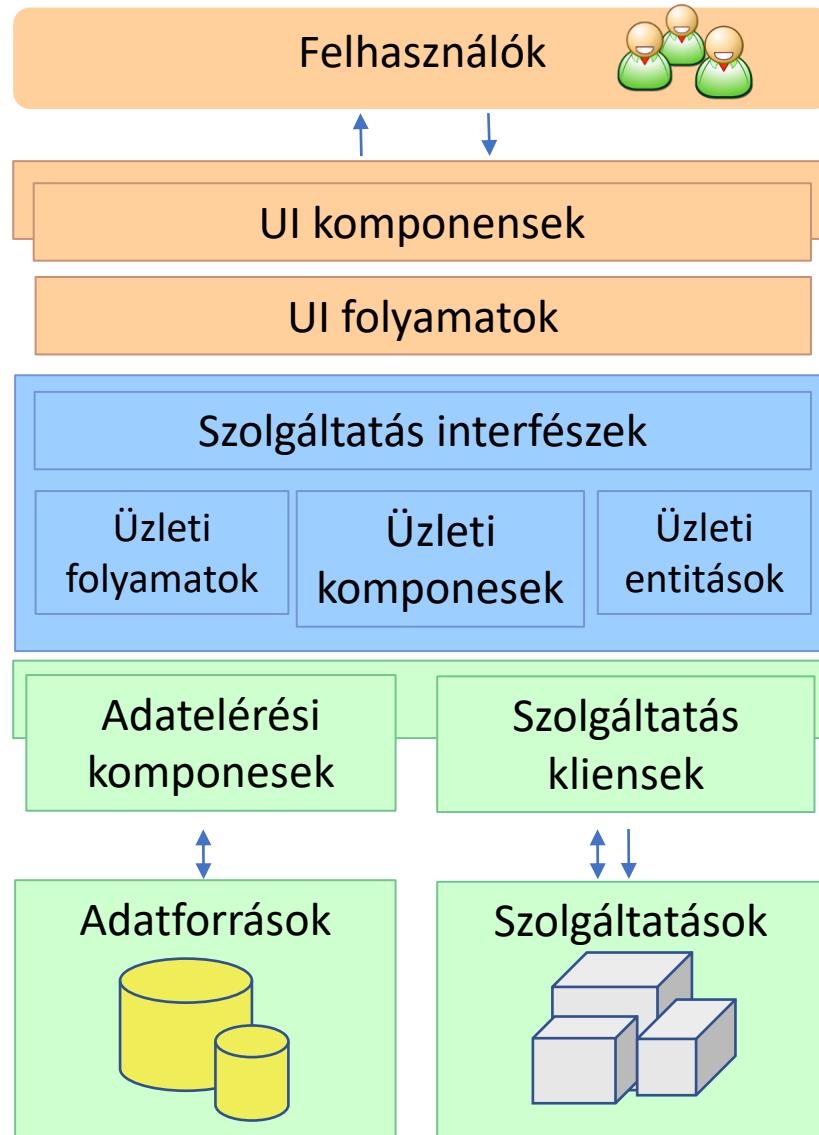


3 rétegű architektúra

Megjelenítési réteg
Presentation Layer (UI)

Üzleti logika réteg
Business Logic Layer (BLL)

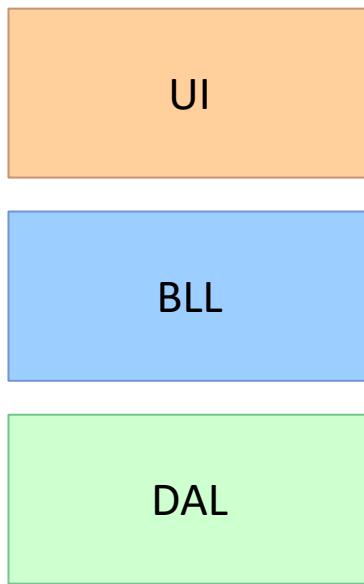
Adatelérési réteg
Data Access Layer (DAL)



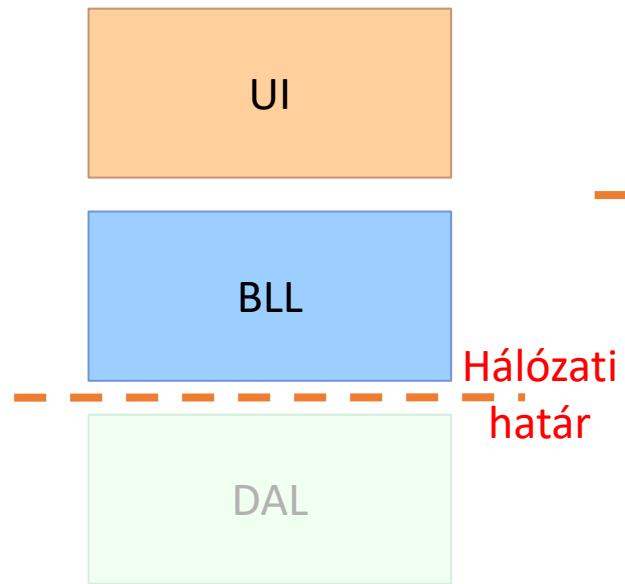
Megjelenítési réteg feladatai

- Adatok megjelenítése
 - > Felhasználóbarát megjelenítés
 - Lista → sorrendezett, csoportosított, kereshető nézet
 - > Adatok lekérése a felhasználói interakció szerint
 - > Státusz információ biztosítása (szerkesztett adatok)
- Felhasználói input ellenőrzése
 - > Formátum, kötelező megadni
- Egyszerű transzformációk
 - > Termék név megjelenítésben → Termék azonosító alsóbb rétegekben
- Lokalizáció

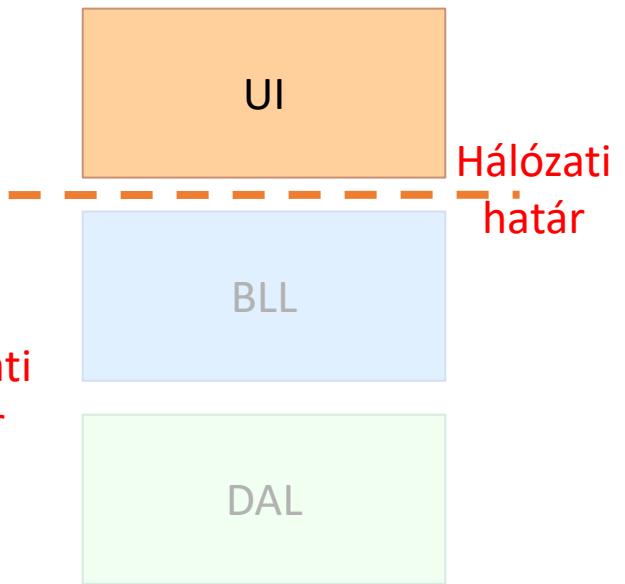
Önálló szoftver



Vastagkliens



Vékony kliens



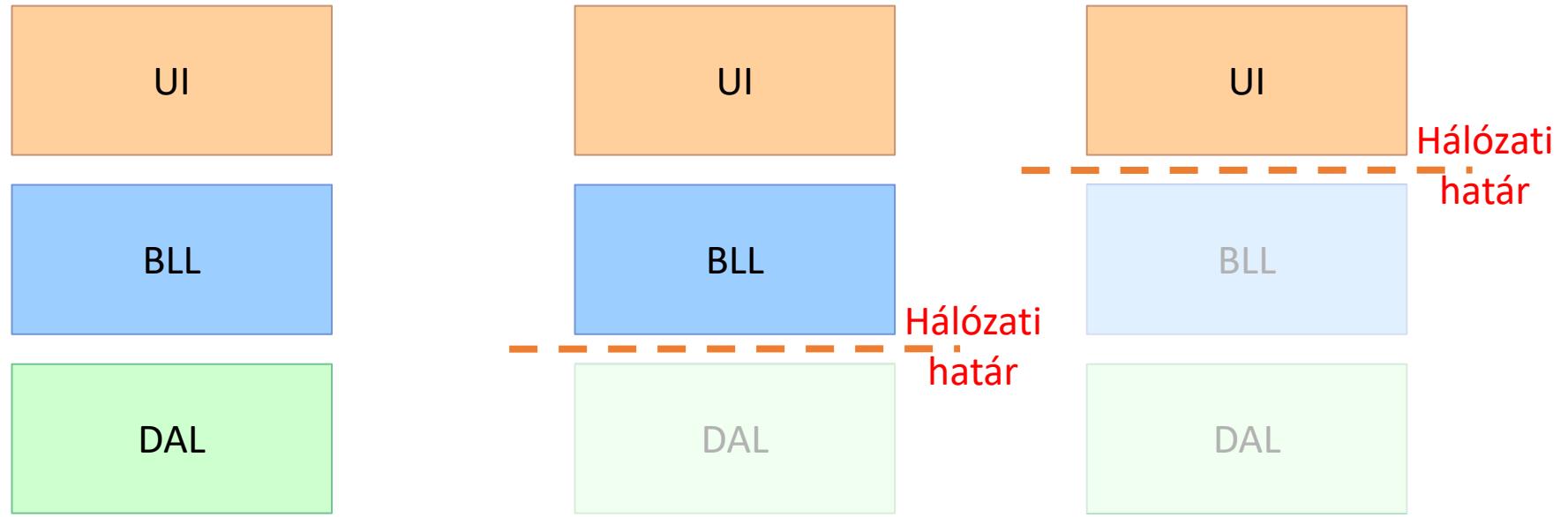
Word, Excel,
PowerPoint

Single player játékok
Photoshop, Paint

Onenote, Edzésnapló,
Levelező kliens
Multiplayer játékok
Bittorrent kliens

Gmail, Office365
Böngésző alapú
alkalmazások
PuTTY, SSH, Telnet

Kliens oldali technológiák



Kliens oldali technológiák

Bár az egyes technológiák dominánsabbak lehetnek adott típusú kliensalkalmazásnál, többnyire nem kizártlagos a használatuk.

Hálózati
háttér

DAL

DAL

DAL



.NET

Kliensoldali technológiák

Dr. Kővári Bence, BME, AUT

kovari.bence@vik.bme.hu

.NET kliens technológiák története

2002 .NET 1.0, Windows Forms

2006 .NET 3.0 (Vista), WPF

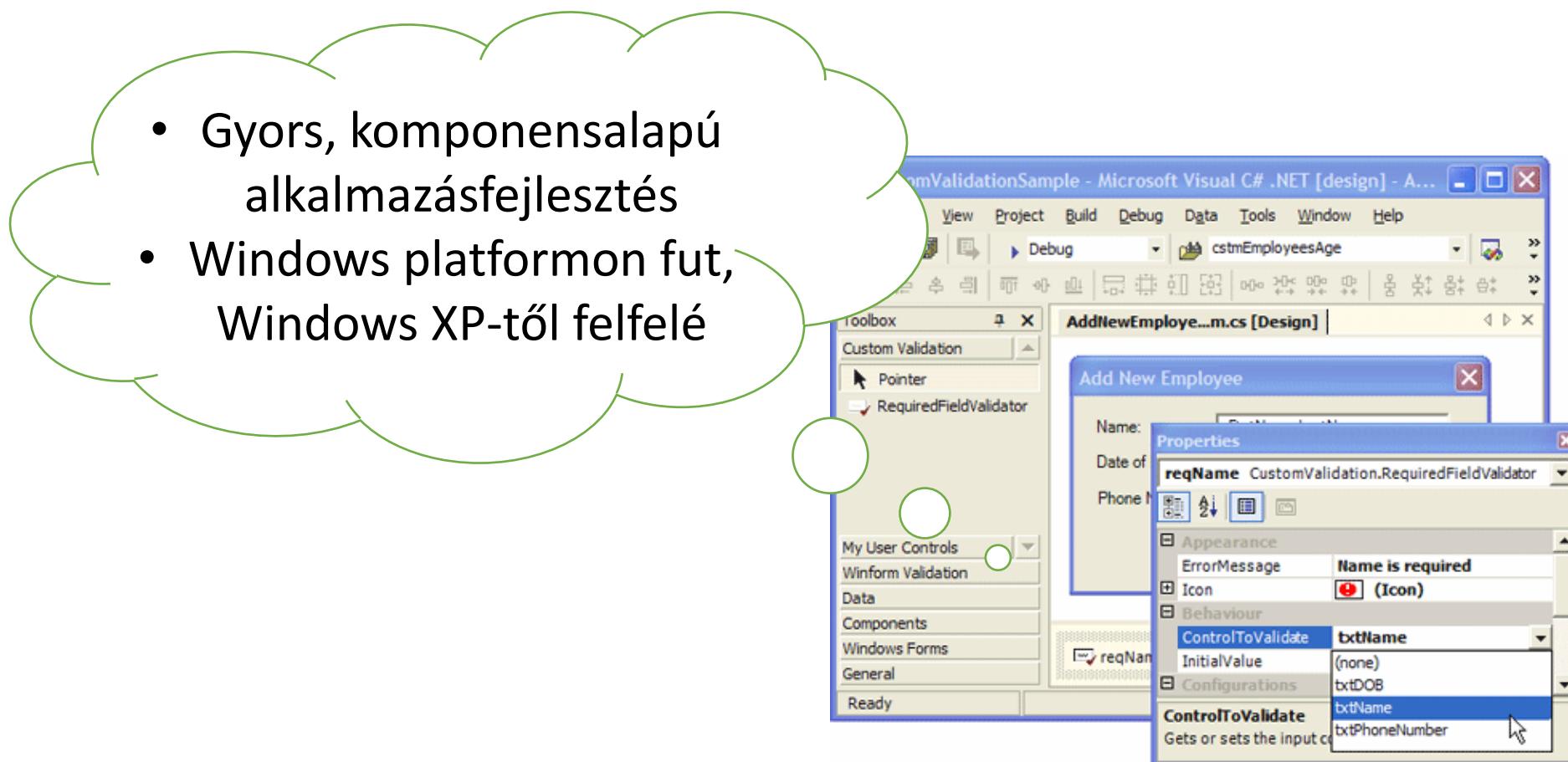
2007 Silverlight

2012 Windows 8, WinRT

2015 Windows 10, UWP

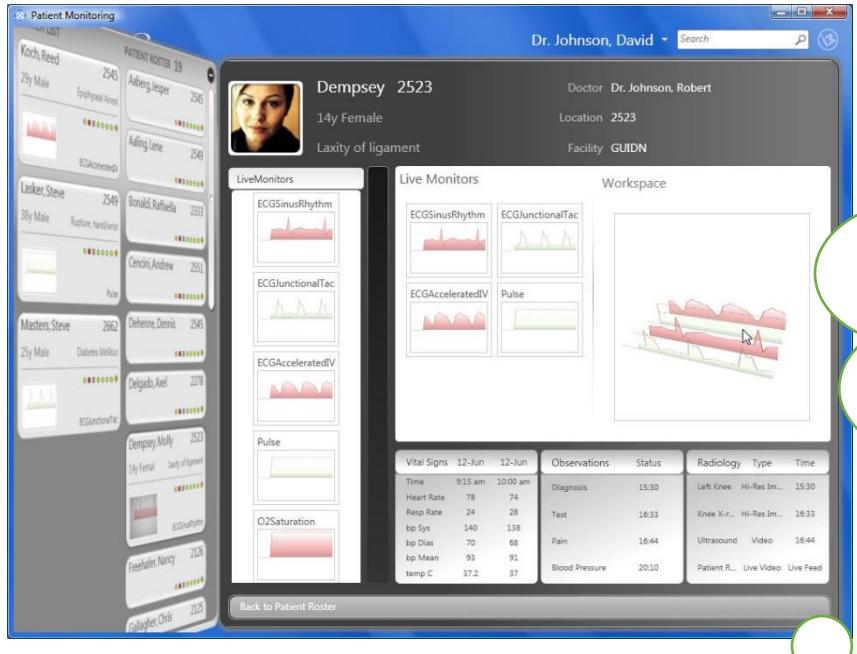
.NET kliens technológiák története

2002 .NET 1.0, Windows Forms



.NET kliens technológiák története

2006 .NET 3.0 (Vista), WPF



- Újszerű, vektorgrafikus
- XAML felületleíró nyelv
- Windows XP-től felfelé



```
<StackPanel>
    <Button Content="Click Me" Width="150" Margin="10"/>
</StackPanel>
</Window>
```

.NET kliens technológiák története

2007

Silver



- .Net alapú, böngészőben fut
- Windows XP, MacOS, Symbian
- Windows Phone 7 is ezen alapult
- Sosem érte el a kritikus piaci tömeget

.NET kliens technológiák története

2012 Windows 8, WinRT

- Windows 8 + Windows Phone 8

2015 Windows 10, UWP

- Universal Windows Platform



Mobil

Kliensoldali technológiák

Dr. Kővári Bence, BME, AUT

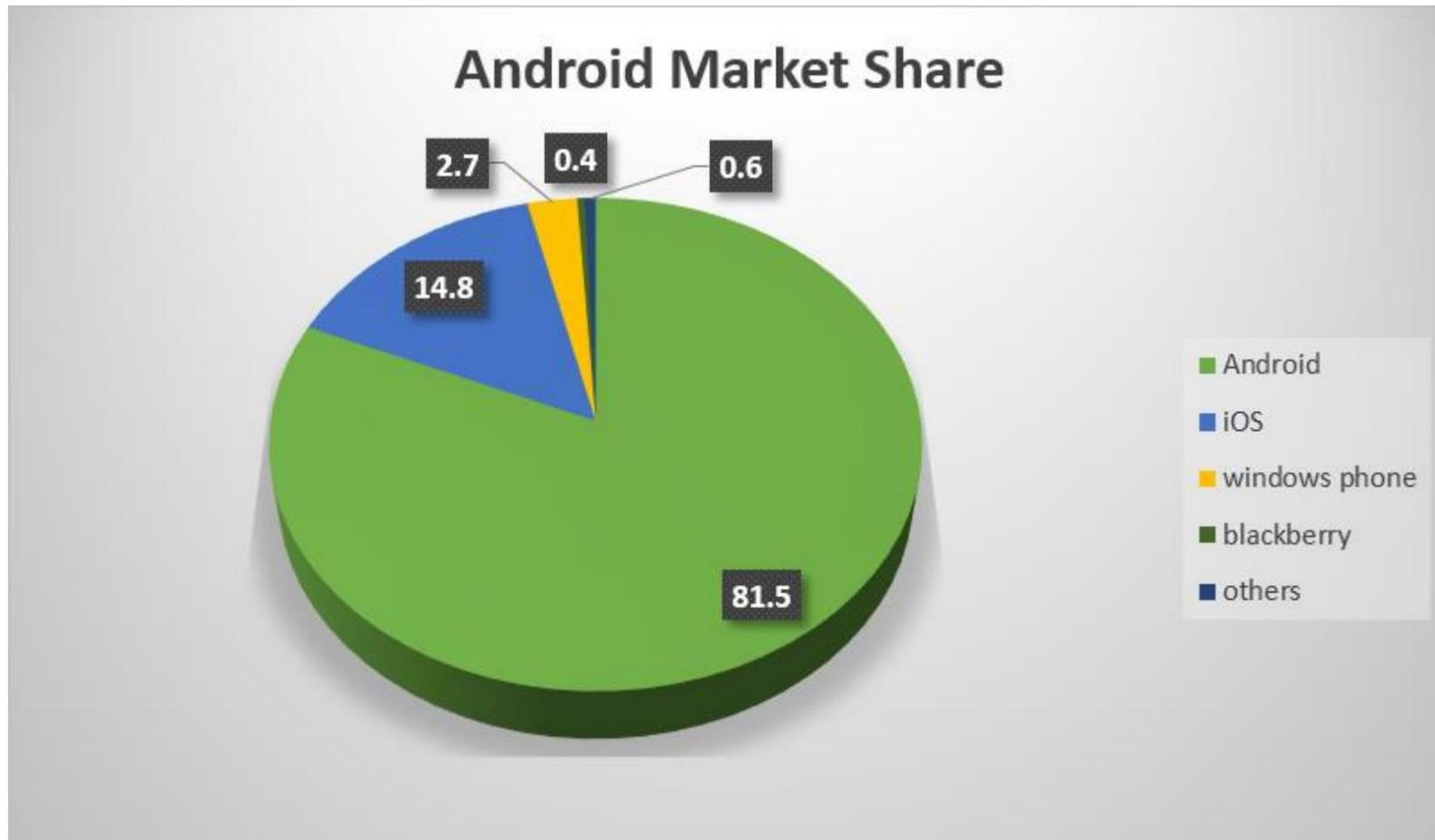
kovari.bence@vik.bme.hu

Mobil szoftverplatformok

- Natív megoldások
- Java ME
- Symbian OS
- Windows Phone
- Android
- iOS

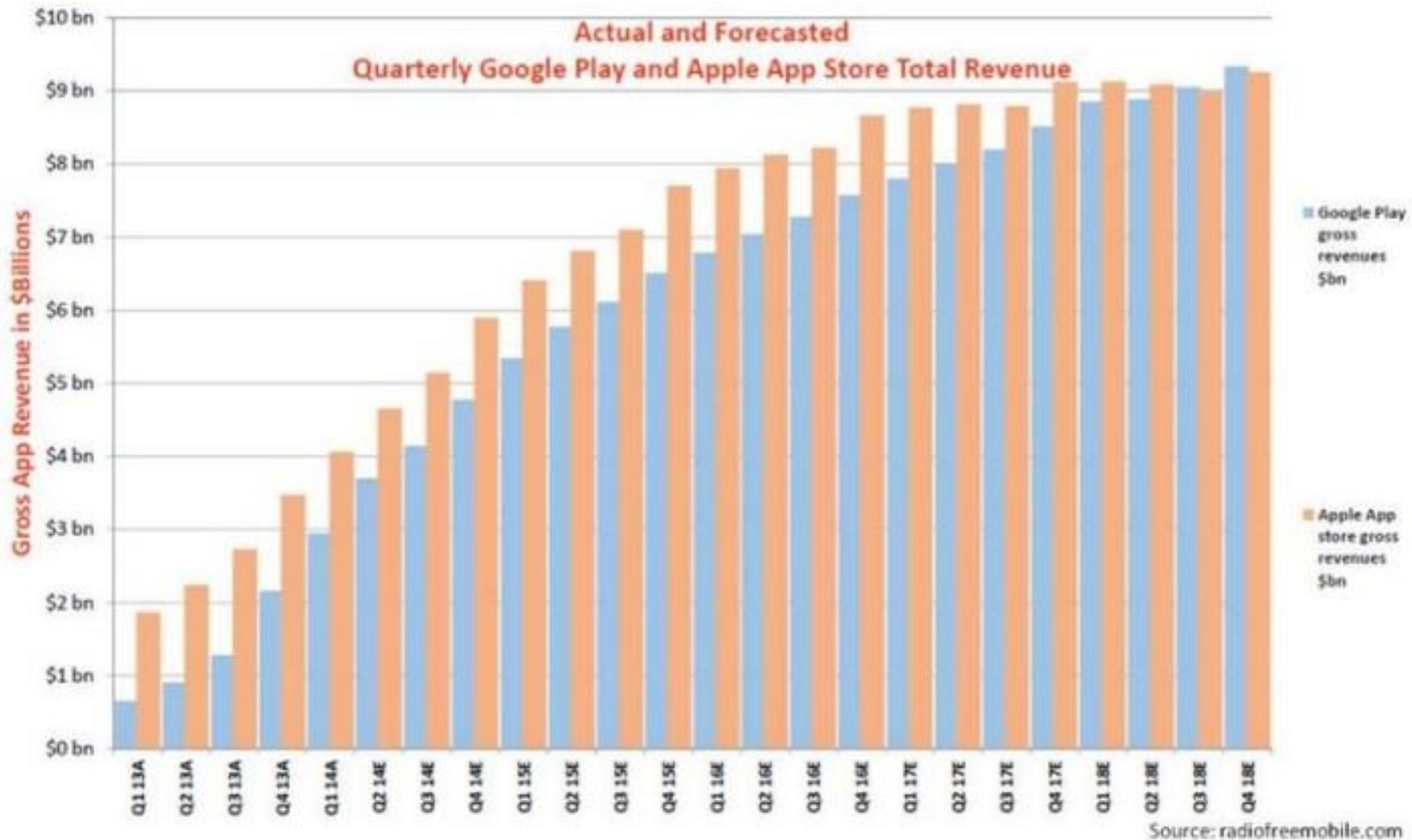
→ Mobil- és webes szoftverek (VIAUAC00)

Miért NEM CSAK Android?



forrás: <http://codecondo.com/average-salary-of-an-android-programmer-in-2015>

Miért NEM CSAK Android



Miért NEM CSAK Android

- A felhasználók több platformon veszik igénybe a szolgáltatásokat
- minden egyedi platformra fejleszteni az alkalmazásunkat költséges
- Kisebb-nagyobb kompromisszumokkal elérhető, hogy egy közös kódmezőt használjunk a különböző platformokon

Multiplatform mobil fejlesztés

- Apache Cordova (aka PhoneGap)
 - > HTML 5 + CSS alapú felület leírás
 - > JavaScript alapú programozás
 - > Lehetőség natív komponensek illesztésére
- Xamarin
 - > XAML alapú felületleírás
 - > C# alapú programozás
 - > Lehetőség natív komponensek illesztésére
- HTML 5
 - > Böngésző alapú (nem natív mobilos) alkalmazások



Webes technológiák

Kliensoldali technológiák

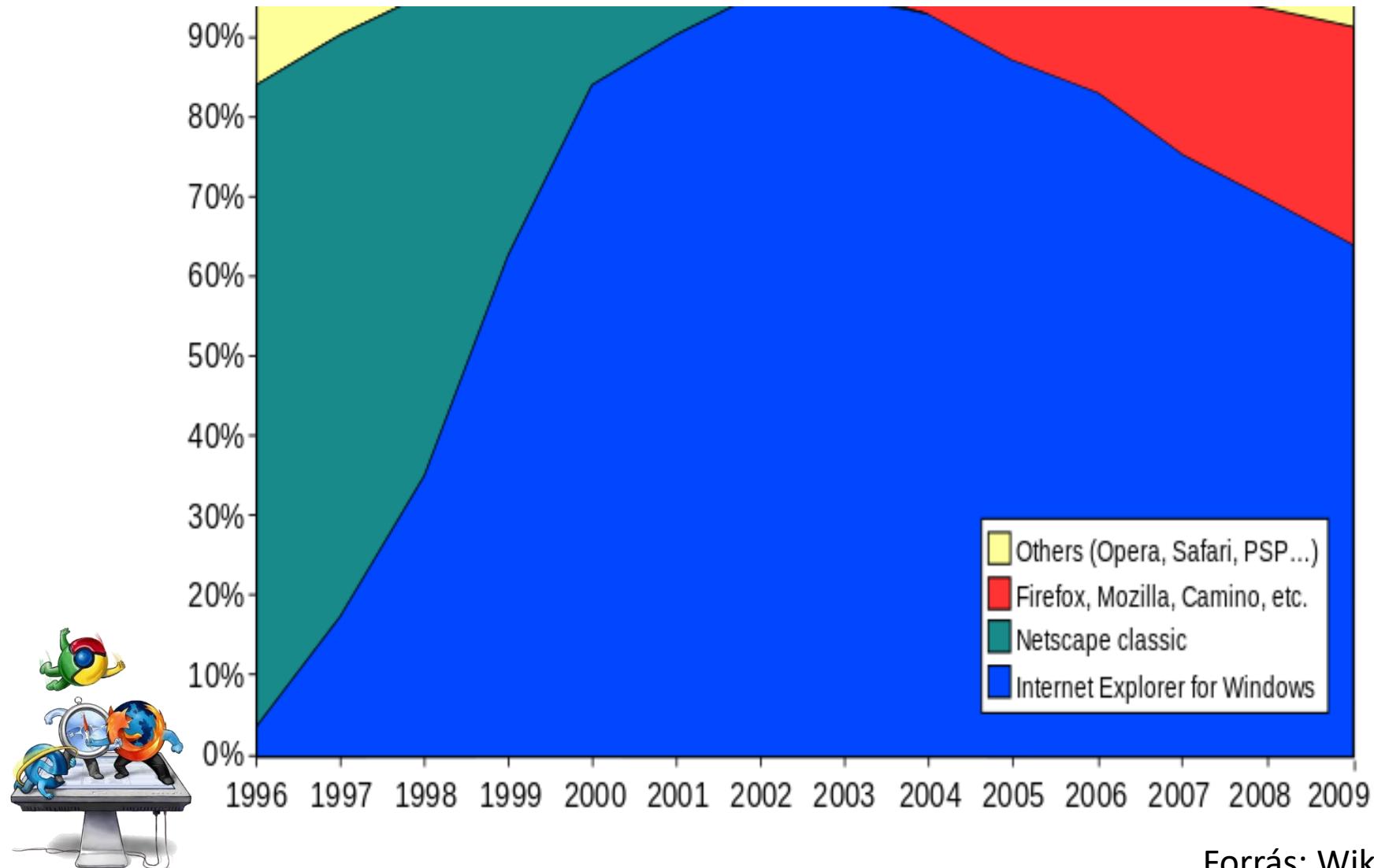
Dr. Kővári Bence, BME, AUT

kovari.bence@vik.bme.hu

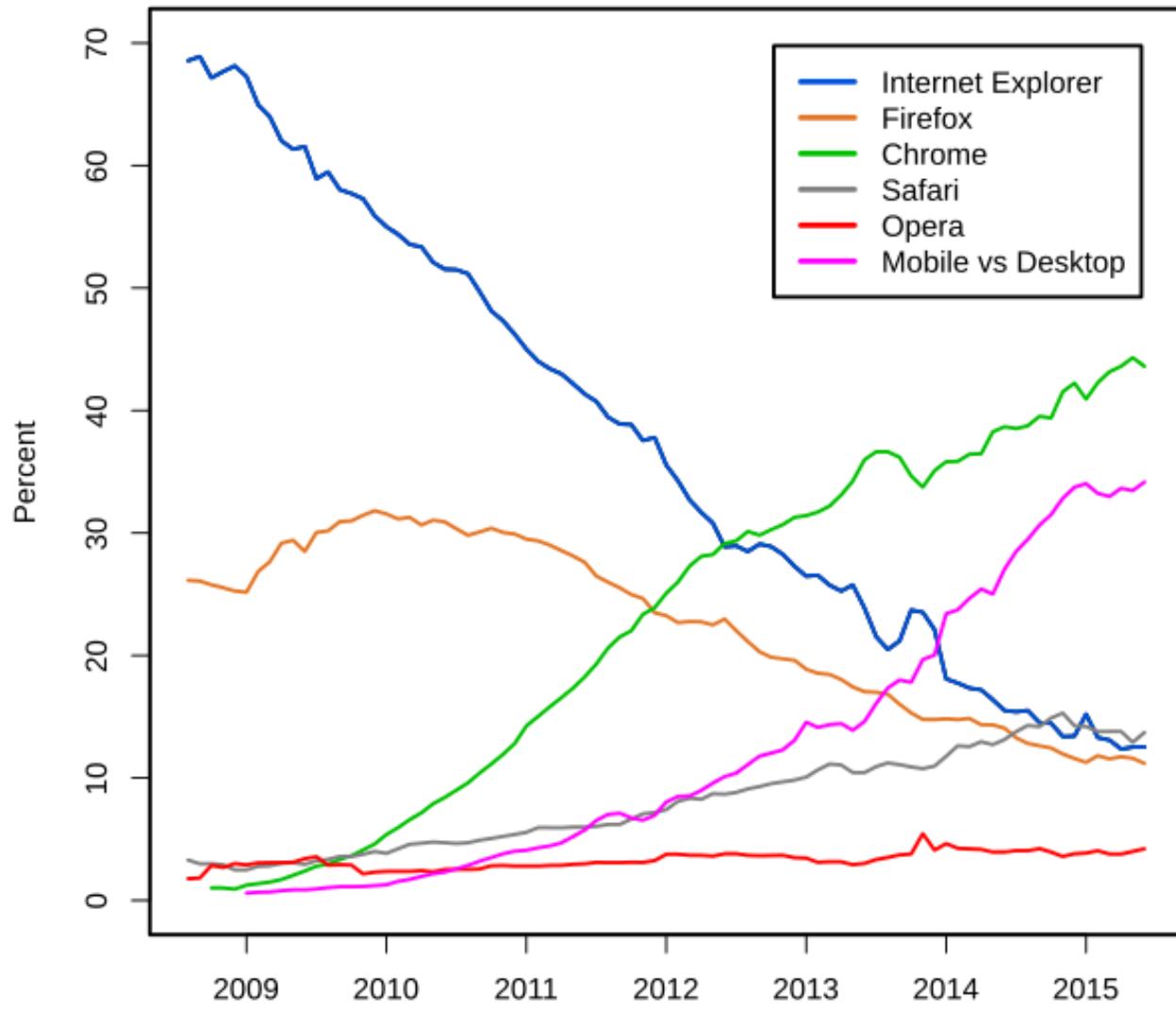
1. Böngésző háború (1995-2009)



1. Böngésző háború (1995-2009)



2. Böngésző háború

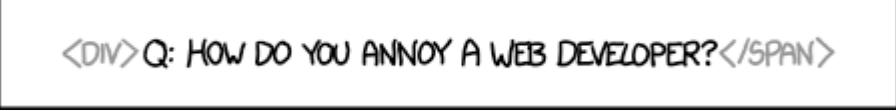


Forrás: Wikipedia, StatCounter

Böngésző háborúk

Böngészők közt éveken át jelentős kompatibilitási problémák voltak

- HTML szabvány eltérő értelmezése



<DIV>Q: HOW DO YOU ANNOY A WEB DEVELOPER?

Forrás: xkcd.com

- CSS eltérő értelmezése
- Sebessékgülönbségek
(elsősorban JS futtatásnál)

Böngészők ma...

A mai böngészők nagy mértékben szabványkövetők, kielégítő teljesítménnyel.

<http://kangax.github.io/compat-table/es6/>

Webes technológiák

- HTML5 + CSS
 - > Nem feltétlenül a legoptimálisabb, de a leghasználtabb programozási nyelv (C#, Java, C++ előtt áll)
 - > Számos keretrendszer épül rá



Kihívások a kliensoldalon

Kliensoldali technológiák

Dr. Kővári Bence, BME, AUT

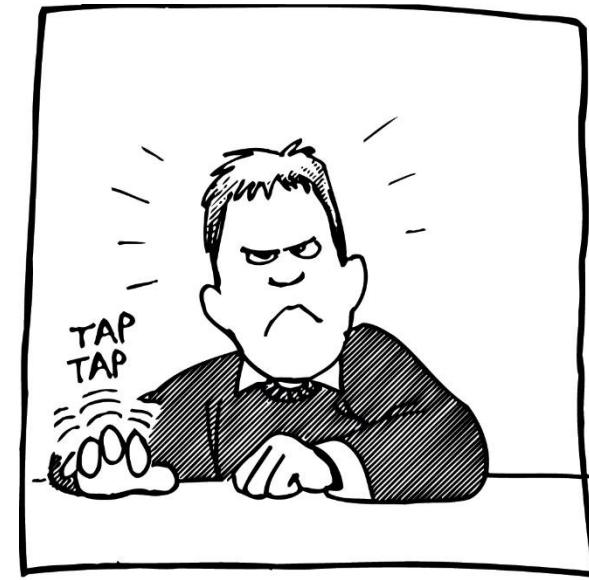
kovari.bence@vik.bme.hu

Kihívások a kliensoldalon

- Környezet
 - > Heterogén (Windows, Android, iOS ...)
 - > Ismeretlen (egyedi konfigurációk)
 - > Ellenőrizetlen (jogosultságok, adatbiztonság...)
- Böngészők
 - > Kisebb különbségek akadnak köztük

Kihívások a kliensoldalon

- Felhasználó
 - > Heterogén
 - > Kiszámíthatatlan
 - > Türelmetlen
 - > ...



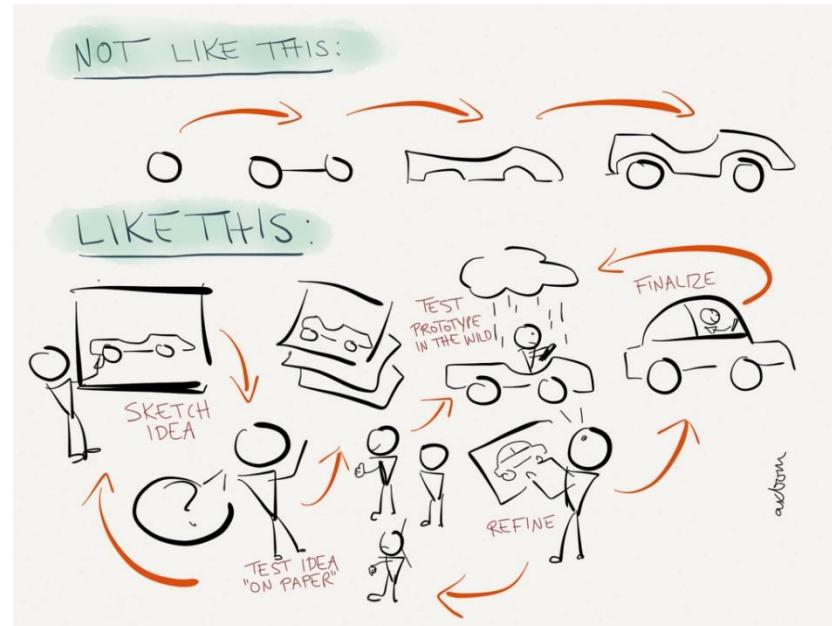
Kihívások a kliensoldalon

- Hibakezelés
 - > Hibák detektálása
 - > Hibák kezelése
 - > Felhasználói visszajelzés?



Kihívások a kliensoldalon

- Ergonómia
 - > Prototipizálás, sketch
 - > Felhasználói élmény (UX) tervezés
 - > Intuitív felhasználói felületek
 - > Reszponzivitás



És legvégül pedig ezt itt.

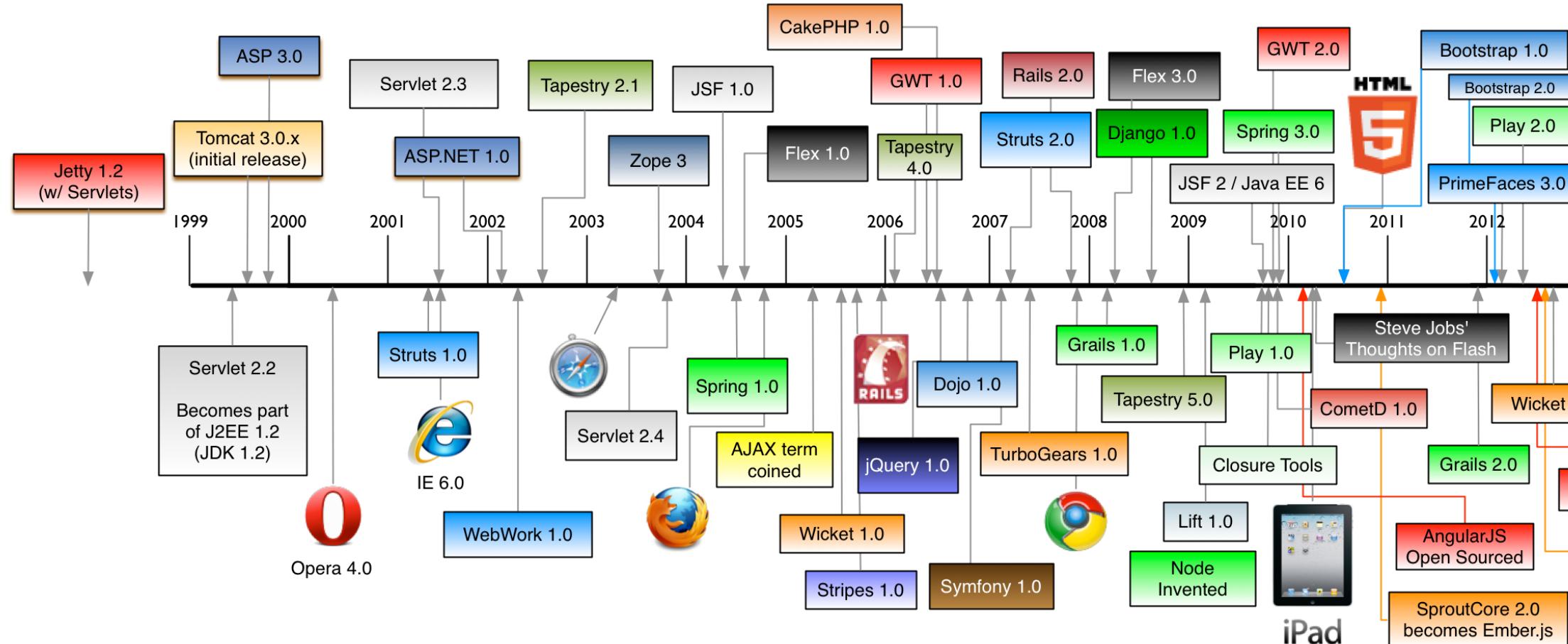
**Ezt olvasod el
először.**

Aztán pedig majd ezt.

Utána ezt a sort.

Kihívások a kliensoldalon

- Rohamosan változó kliensoldali stack



<https://github.com/mraible/history-of-web-frameworks-timeline>

Esettanulmány

Angular tapasztalat van

Classic MVC, vagy Single Page Application legyen?

Web teszt: Selenium és Visual Studio webtest tapasztalatunk van

ORM: Entity Framework 6, Code first

Felhasználókezelés: ASP.NET Identity

Continuous integration: ha kell TFS-en is be tudunk lőni

Bolob-ok és egyéb Azure szolgáltatások használata

PowerShell, minél több folyamat automatizálására

Reactive Extensions: Megfontolható a használata

SignalR

Scriptnyelv: TypeScript szímpatikusnak tűnik mindenkinél

REST-es api-t dokumentálni kell

Swagger.io-t az Androidos csapat már többször használta

Naplázás: Log4net

Monitoring: Application Insights praktikus rá az Azure Stack-en

Részletek egy webes project kickoff jegyzőkönyvéből...

2016 január

Kliensoldali fejlesztés ma

Számos HTML5 alapú, vagy mobil fejlesztés

UI keretrendszerök

UI komponens készletek

Üzleti alkalmazásoknál még mindig sok vastagkliens fejlesztés (C#, Java)

Kliensoldali fejlesztés holnap

- HTML5, JS, CSS
- Bot framework
- Wearables
- IoT
- 3D
- Holographic, AR
- Voice

Windows Store alkalmazás fejlesztés

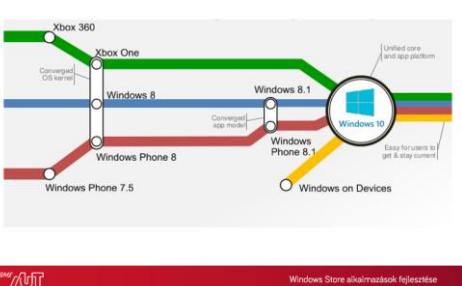
< Albert István >



Automatizálási és
Alkalmazott
Informatikai Tanszék

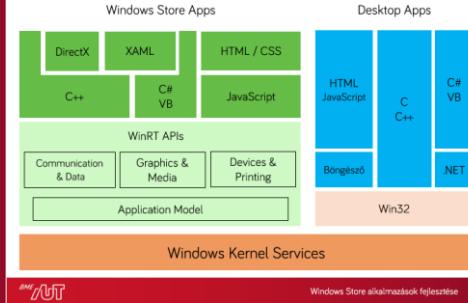
Tartalom

Windows 10 - konvergencia



Windows Store alkalmazások fejlesztése

A „nagy” Windows platform



Windows Store alkalmazások fejlesztése

Windows Runtime (WinRT)

- Új platform API
- A Win32 „utódja”
- Natív API
- COMponens orientált
- Programozási nyelv független
 - C#, VB, C++, JS, ...

- Elsődleges natív API, nem a Win32-re épít!



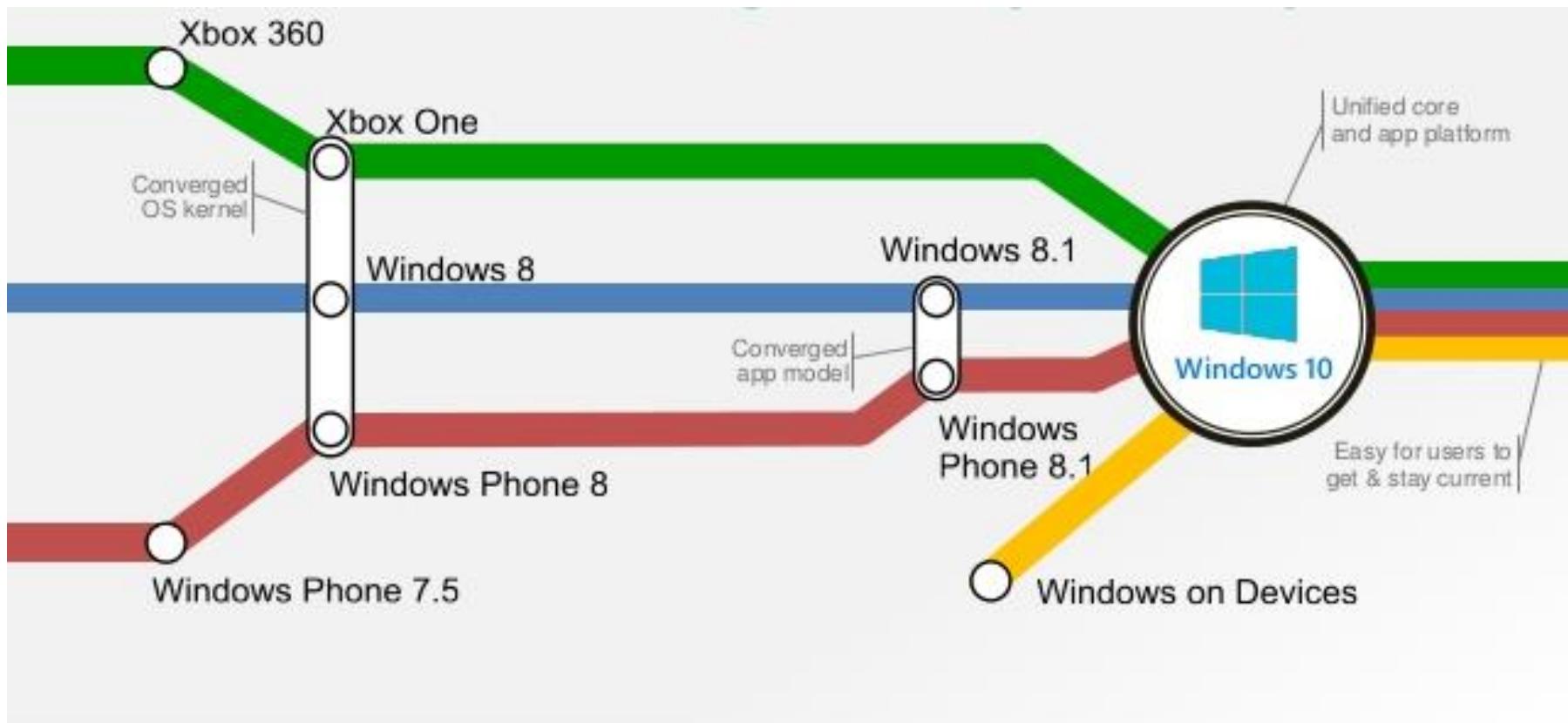
Windows Store alkalmazások fejlesztése

.NET Foundation: új megközelítés



Windows Store alkalmazások fejlesztése

Windows 10 - konvergencia



Széleskörű eszköztámogatás - form faktor

- IoT
 - > Windows 10 IoT Core
- Tablet + Hibrid
- Notebook
- Desktop
- Surface Hub
- HoloLens
- Xbox One
- Surface Neo



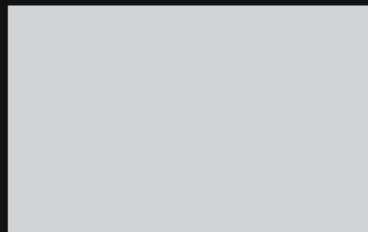
Széleskörű eszköztámogatás

- Támogatott architektúrák:
 - > x86 és x64
 - > ARM – (újra?)
- Támogatott beviteli eszközök:
 - > Érintés / Toll
 - > Egér
 - > Billentyűzet
 - > Xbox controller, Kinect, Hololens gesztusok
- Támogatott képernyőméretek:
 - > 0" - 4" - 7" - 10" - 30"+

Windows 8.x: „Szakítsunk a múlttal”

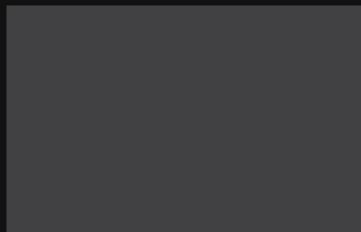
← Grid Template

Collection title lorem 0



0.0 ÅSed nisl nibh, eleifend posuere.

Phasellus faucibus



0.2 ÅSed nisl nibh, eleifend posuere.

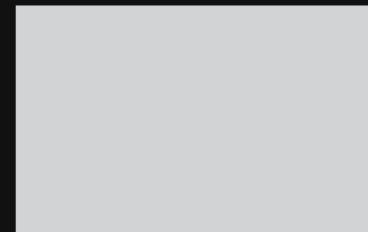
Phasellus faucibus



0.4 ÅSed nisl nibh, eleifend posuere.

Phasellus faucibus

Collection title lorem 1



1.0 ÅSed nisl nibh, eleifend posuere.

Phasellus faucibus



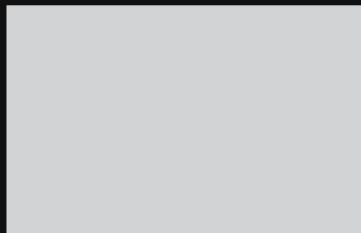
1.2 ÅSed nisl nibh, eleifend posuere.

Phasellus faucibus



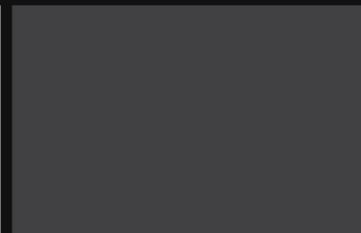
0.1 ÅSed nisl nibh, eleifend posuere laoreet egestas, porttitor quis lorem.

Phasellus faucibus



0.3 ÅSed nisl nibh, eleifend posuere laoreet egestas, porttitor quis lorem.

Phasellus faucibus



0.5 ÅSed nisl nibh, eleifend posuere laoreet egestas, porttitor quis lorem.

Phasellus faucibus



1.1 ÅSed nisl nibh, eleifend posuere laoreet egestas, porttitor quis lorem.

Phasellus faucibus



1.3 ÅSed nisl nibh, eleifend posuere laoreet egestas, porttitor quis lorem.

Phasellus faucibus

Windows 10: konszolidáció

The screenshot shows the Windows 10 Calendar application interface. The title bar says "Naptár". Below the title bar are navigation buttons: "Nap" (Day), "Munkahét" (Work week), "Hét" (Week), "Hónap" (Month), and "Ma" (Today). On the left, there's a sidebar with "+ Új esemény" (New event), a month calendar for July 2016, and sections for "Microsoft account" and "Calendar". The main area displays the month of July 2016 with days from 06. 27. to 31. The days are labeled with their corresponding dates: 27., 28., 29., 30., 1., 2., 3., 4., 5., 6., 7., 11., 12., 13., 14., 15., 16., 17., 18., 19., 20., 21., 22., 23., 24., 25., 26., 27., 28., 29., 30., 31.

Windows 10: konszolidáció

The screenshot shows the Windows Store interface. At the top, there is a navigation bar with tabs: Áruház, Kezdőlap, Alkalmazások, Játékok, Zene, and Filmek + TV. A search icon is also present. Below the navigation bar, there is a large image of the game FarmVille 2: Country Escape, which features a vibrant farm scene with various crops and animals. Below this main image, there are smaller thumbnail images of other games like Minecraft and a bicycle wheel. At the bottom of the screen, there are three categories: Alkalmazás-toplisták és -kategóriák, Játéklisták és -kategóriák, and Kiemelt. There are also sections for "Önnek ajánljuk" (Recommendations) and "Az összes megjelenítése" (All versions). The recommendations section includes icons for LastPass, Another Case Solved, Power Planner, Microsoft Treasure Hunt, and ABC News.

Áruház

Kezdőlap Alkalmazások Játékok Zene Filmek + TV

FarmVille 2: Country Escape
Go on a farm adventure with friends

Alkalmazás-toplisták és -kategóriák Játéklisták és -kategóriák Kiemelt

Önnek ajánljuk

LastPass Another Case Solved Power Planner Microsoft Treasure Hunt ABC News

Az összes megjelenítése

UWP alkalmazás modell - package

- Telepítés egysége
 - > Ezt telepítjük (Windows Store) (*.appx)
 - > Nincsenek saját installerek, nincs ClickOnce
 - > Izolált környezet: törlés után nem marad nyoma
- Tartalma
 - > Alkalmazáshoz tartozó fájlok
 - Bináris, XAML, képek, stb...
 - > Manifest
 - Alkalmazás info (név, csempék, ...)
 - Jogosultságok (Capabilities)
 - Alkalmazás képességek (kontraktok és kiegészítések)

Fájlok kezelése

- App container
 - > Sandbox koncepció - korlátozott fájlkezelés
- Hozzáférés:
 - > Package könyvtára (alkalmazás fájlok)
 - csak olvasható
 - > Alkalmazás adatai
 - AppData könyvtárban
 - Teljes hozzáférés
 - > Documents, Music, Photo, Video
 - Manifest-ben jogot kell kérni
 - Fájl típus hozzárendelés
 - > FilePicker
 - A felhasználó által kiválasztott fájl, mappa

Korlátrozott multitaszk

- Csak az előtérben futó alkalmazások kapnak erőforrást
 - > Mobil jellegzetesség: teljesítmény, akku kímélése
 - > Desktopon csak akkor „alszik el” az app, ha minimize-oljuk
- Kompenzációk
 - > Háttérben futó folyamatok (erős korlátozásokkal)
 - Fájl le/feltöltés, zenelejátszás, GPS
 - > Push Notification: értesítések
 - csempe frissítés, toast üzenetek, ...
- De egyre jobban lazítják ezeket a megkötéseket
 - > Extended execution, Long running bgtask

Windows Store

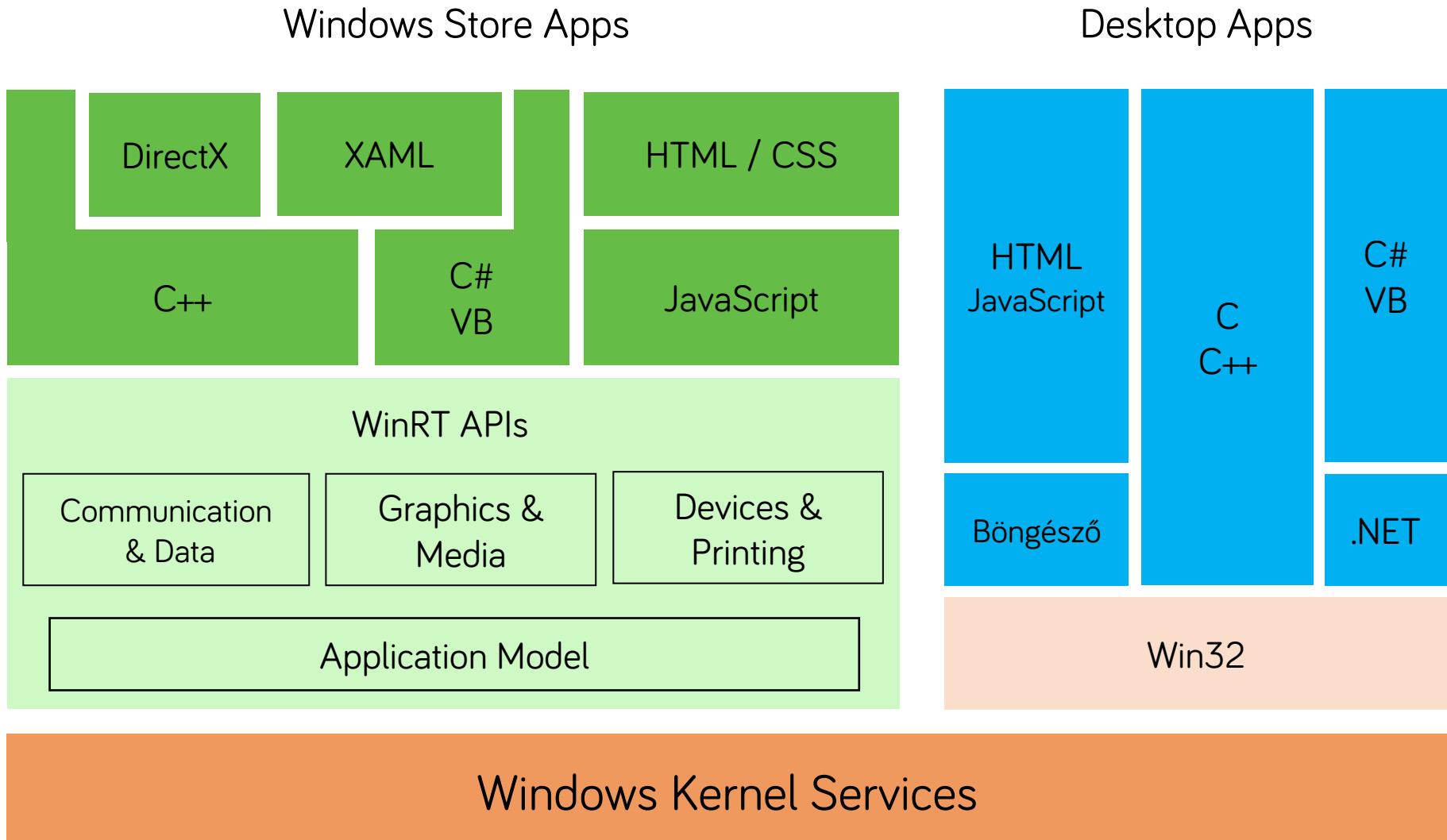
- Egységes áruház az összes alkalmazásnak
- Ingyenes és fizetők alkalmazások
- 20% a Store-é, 80% a fejlesztőé
- Szolgáltatások
 - > Alkalmazáson belüli fizetés
 - 3rd-party fizetés 2016 óta nem támogatott ☹
 - > Automatikus alkalmazás frissítés
 - > Hirdetés
 - > Próbaverziók
 - > Elemzések, bugriportok

IoT - Windows 10 IoT Core

- Lecsupaszított rendszer beágyazott eszközökre
 - > Raspberry Pi2, 3, DragonBoard 410c stb.
- UWP appok futtathatóak rajtuk
 - > Speciális API-k eléréséhez plusz extension SDK
- Kétféle üzemmód:
 - > Headed (default)
 - UWP UI app + sok UI nélküli app
 - > Headless
 - Csak UI nélküli alkalmazások
- Magas és alacsony szintű API-k
 - > Windows.Devices névtér
 - > Scanner, Nyomtató (2D/3D), Point of Service, DLNA, SMS, ...
 - > USB (HID), Bluetooth (WPD), AllJoyn, Gpio, I2C



A „nagy” Windows platform



Desktop alkalmazások

- .NET Fx 4.8 (C#, VB)

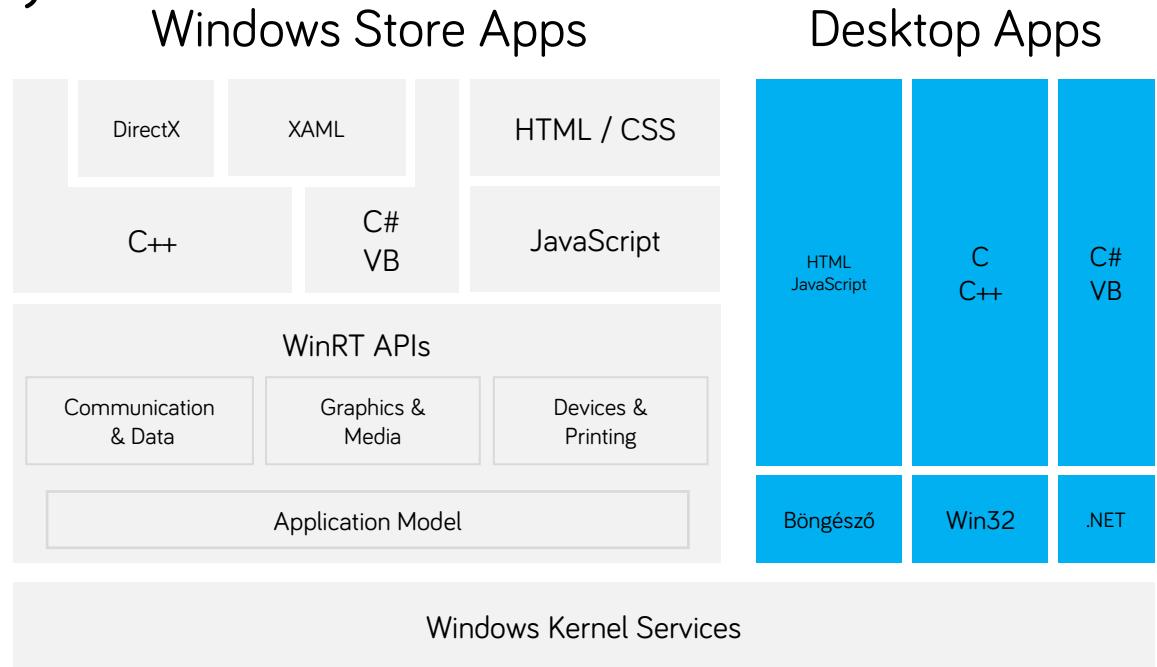
- > WPF
 - > WinForms
 - > Console

- C/C++ Win32

- Web

- > HTML, JS

- Stb...



Windows Store (UWP) alkalmazások

1. C#, VB

> XAML

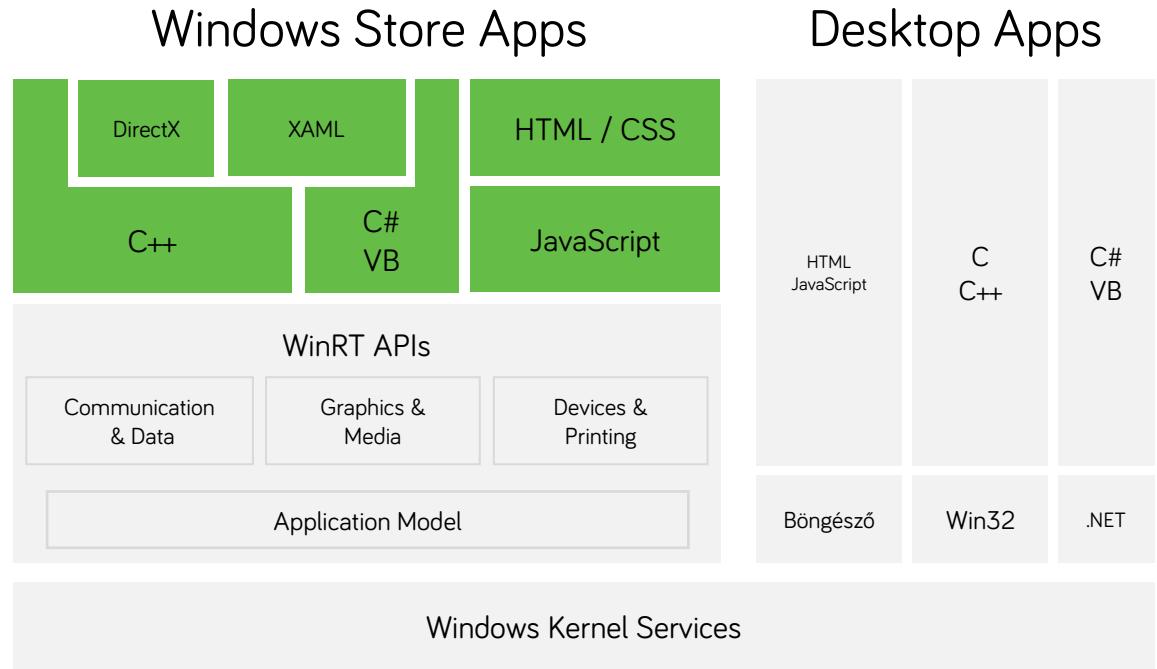
2. C++

> XAML

> DirectX

3. JavaScript

> HTML/CSS



„Válaszd azt, amihez értesz!”

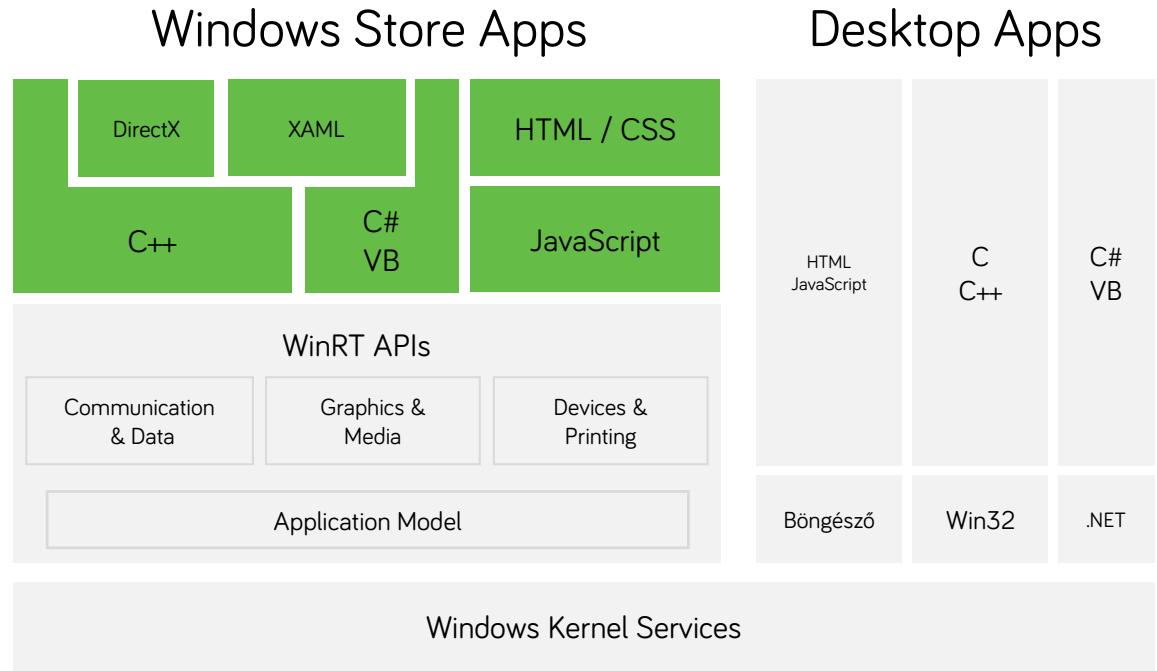
UI keretrendszer (HTML)

- Edge
 - > HTML 5
 - > CSS
 - > Windows specifikus kiterjesztések
 - > Hosted vagy packaged
- HW gyorsítás
- Fejlesztés
 - > JavaScript
 - > TypeScript
 - > ...

Platform bridge-ek

1. Web bridge

- > Online weboldal „becsomagolása”

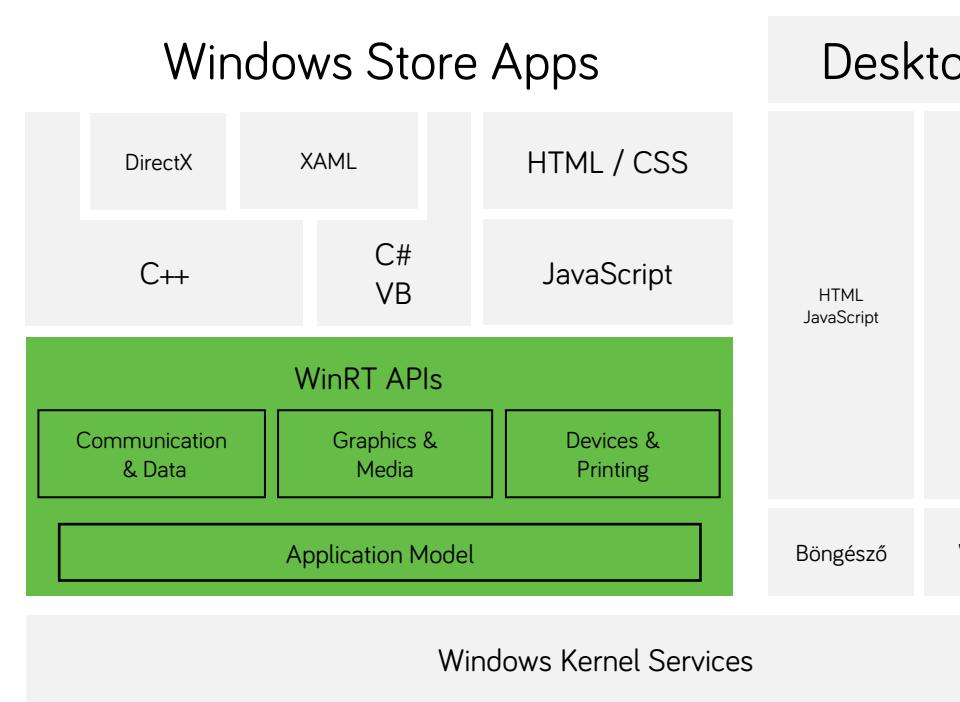


2. Desktop app bridge

- > Tetszőleges hagyományos app Store-ban terjesztésére
- > Egységes, biztonságos telepítés

Windows Runtime (WinRT)

- Új platform API
- A Win32 „utódja”
- Natív API
- COMponens orientált
- Programozási nyelv független
 - > C#, VB, C++, JS, ...
- Elsődleges natív API, nem a Win32-re épít!



Miért született meg a WinRT?

- Az operációs rendszert közvetlenül lehessen programozni különböző nyelvi környezetekből
 - > Nem kell kompatibilisnek maradni a korábbi Win32 alapú technológiákkal
- Új alkalmazás/OS paradigmák bevezetése
 - > Sandboxing (app container), biztonság
 - > Energiatakarékosság
 - > Aszinkronitás
 - > Multiplatform
 - > Mobil alkalmazás koncepció



UI keretrendszer (XAML)

- WinRT része (natív)
- WPF „utódja”
 - > Teljesítmény szempontok, natív kód
 - > Vannak eltérések a WPF-hez képest
- HW gyorsítás, DirectX
 - > Amit csak lehet, azt a GPU-n: rajzolás, animációk
- Fejlesztés
 - > C#, VB
 - > C++

COM alapok

- Bináris, nyelv- és platform-független szabvány komponensek együttműködéséhez
 - > Tetszőleges nyelven írt komponensek
 - > Out-of-process és távoli hívások támogatása
- Mutató alapú struktúra, mutató alapú hívások
- Interfész központú
- (Az OLE, ActiveX, DirectX technológiák mind COM alapúak)

COM alapfunkciók

- Funkciók interfész alapú összefogása
- Komponensek egyedi azonosítása (GUID) és példányosítása
- Alap interfész minden komponenshez (IUnknown):
 - > Feltételezett interfész lekérdezése (QueryInterface)
 - > Objektum életciklus támogatás (AddRef, Release)
- Opcionális metaadat exportálás (IDL fájl)

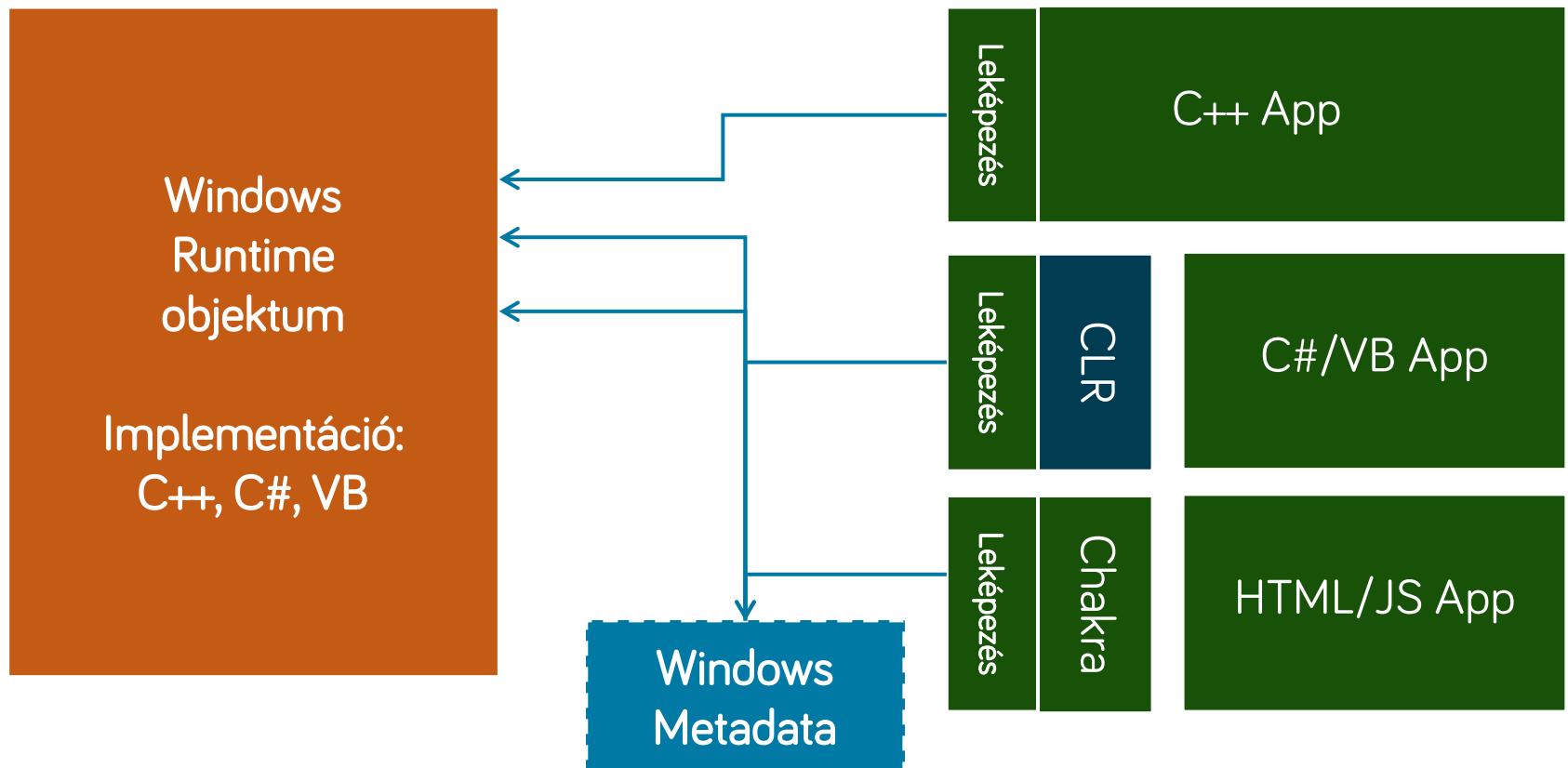
COM újdonságok a WinRT-ben

- Kötelező IInspectable interfész megvalósítás dinamikus bindinghoz (JavaScript)
 - > Megvalósított interfések lekérdezése
 - > String alapú azonosítás – WinMD hivatkozás
- .NET stílusú delegate-ek és események
- Generikus interfések
- Statikus metódusok (a factory osztályon)
- Nincsenek továbbra sem publikus mezők
- Kötelező metaadatok: WinMD fájlok

Windows Metadata (.winmd)

- WinRT komponensek leírása
- C++ vagy C#/VB fordítók által automatikusan generált 
- Az ECMA 334-es szabványnak megfelelő bináris formátum
 - Azonos formátum, más szemantika
- Nincs benne implementáció csak metaadat
- Többnyelvű támogatás
- Teljes Intellisense támogatás statikus típusinformációk alapján

Windows Runtime (WinRT)



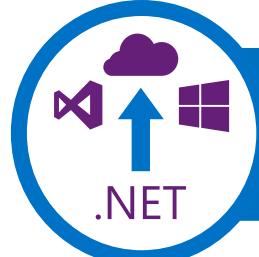
A C# kód

```
using Windows.Storage;
using System;
using System.IO;
using System.Threading.Tasks;
class Sample {
    static async Task WriteAsync(StorageFolder wrtfolder, string filename, string
text) {
        var wrtFile = await wrtFolder.CreateFileAsync(filename);
        wrtStream = await wrtFile.OpenStreamForWriteAsync(FileAccessMode.ReadWrite);
        using (Stream stream = wrtStream.OpenWrite()) {
            using (var writer = new StreamWriter(stream)) {
                writer.WriteLine(text);
            }
        }
    }
}
```

The diagram illustrates various components of the C# code:

- native namespace**: Points to the `Windows.Storage` namespace.
- native type**: Points to the `StorageFolder` type.
- native method**: Points to the `CreateFileAsync` method.
- C# feature**: Points to the `var` keyword and the `await` keyword.
- managed argument passed to native API**: Points to the `FileAccessMode` enum value.
- managed type returned from a seemingly native method**: Points to the `StreamWriter` type.

.NET Foundation: új megközelítés



.NET innováció

Core innovációk a meglévő és
jövőben alkalmazásokhoz



Rugalmasság és agilitás

Folyamatos, moduláris kiadások



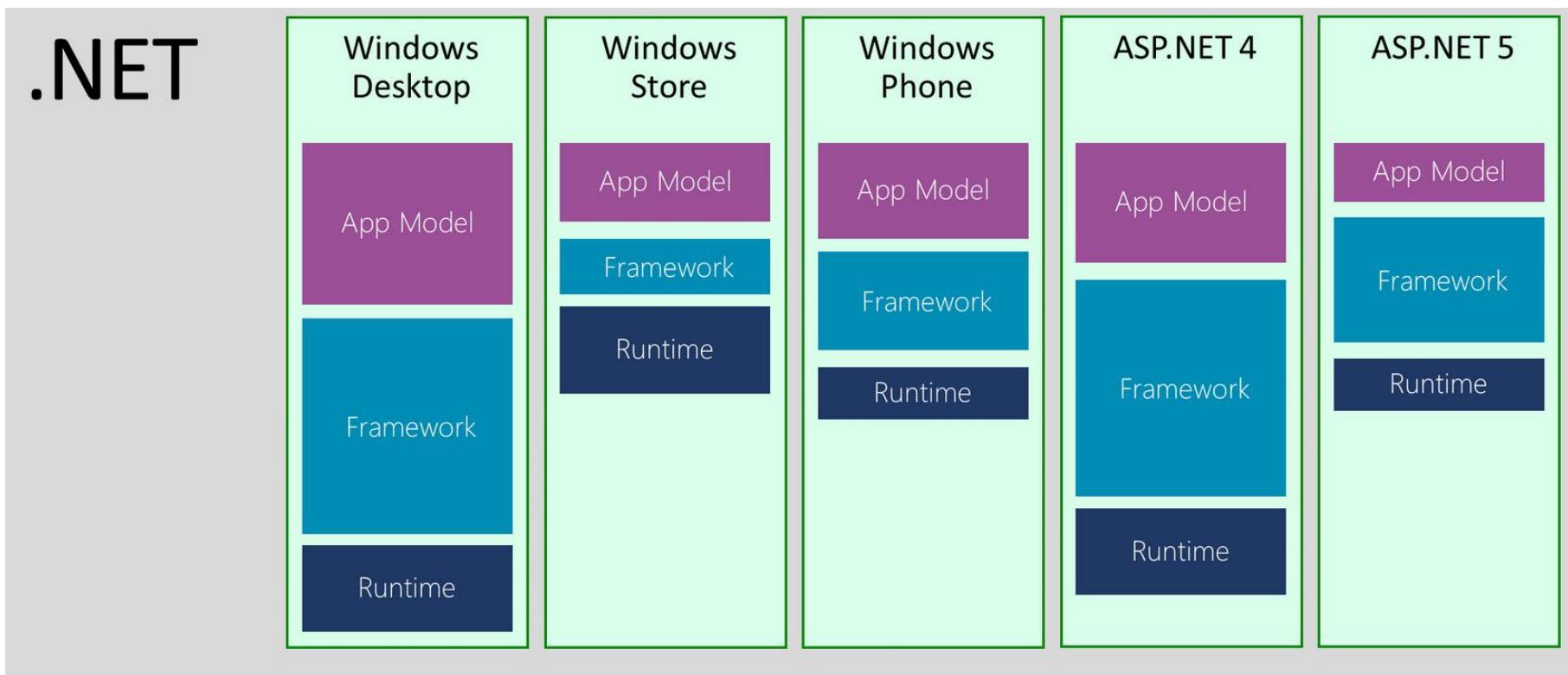
Nyitottság

Átlátható, nyitott, közösségeg-
vezérelt

Futtatókörnyezetek és keretrendszerek

- Futtatókörnyezet: CLR
 - > GC, JIT, ClassLoader, Marshalling, PAL, ...
- Keretrendszer: BLC
 - > List<T>, StringBuilder, FileStream, ...
- Több implementáció
 - > .NET Framework 4 . X
 - > .NET Core
 - > (...NET CF, Rotor, Silverlight, .NET for WinRT)

Különböző .NET stackek



.NET Framework 4.8

- A meglévő, klasszikus asztali .NET keretrendszer következő verziója
 - > Néhány új feature, bugfixes
- Monolitikus
 - > ASP.NET, WCF, WPF, WF, WinForms, ...
- Windows 10-től előtelepítve!

Gép szintű monolitikus keretrendszer

- Előnyei
 - > Központilag kezelhető – vállalati környezet
 - > Kisebb helyigény (mintha alkalmazásonként lenne)
 - > Natív kód megosztása az alkalmazások között
- Hátrányai
 - > Közös függőség bevezetése a gépen futó különböző alkalmazások kötött
 - > Frissítés esetén kompatibilitási problémák
 - > Adminisztrátor jogosultságok kellenek

Ami változott...

Run on Windows



Run everywhere

.NET as system component



Deploy with app

Run on VM (CLR)



Compile to native

Black box compilers



Open compiler APIs

Edit in Visual Studio



Use your favorite editor

Proprietary

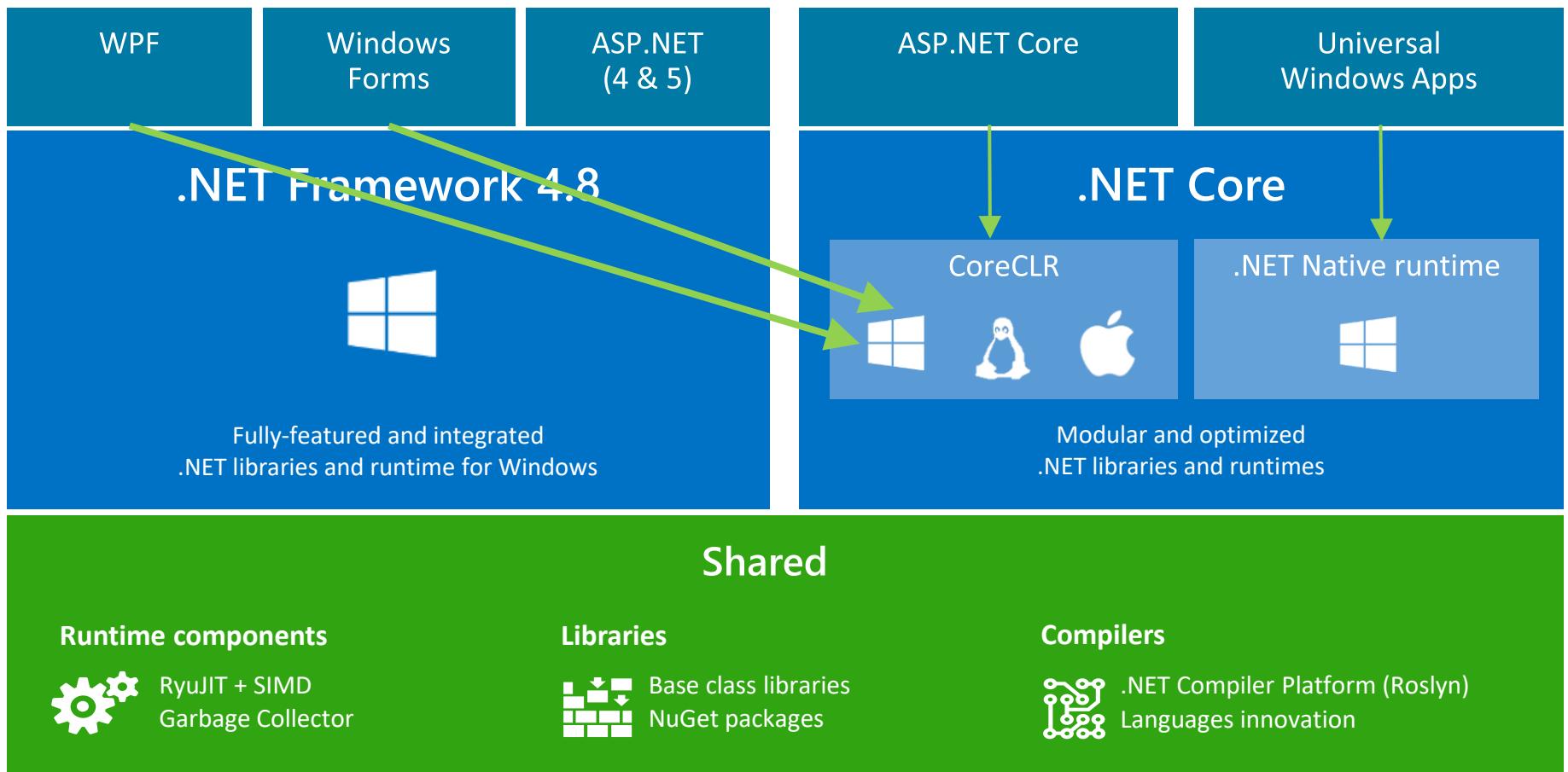


Open source

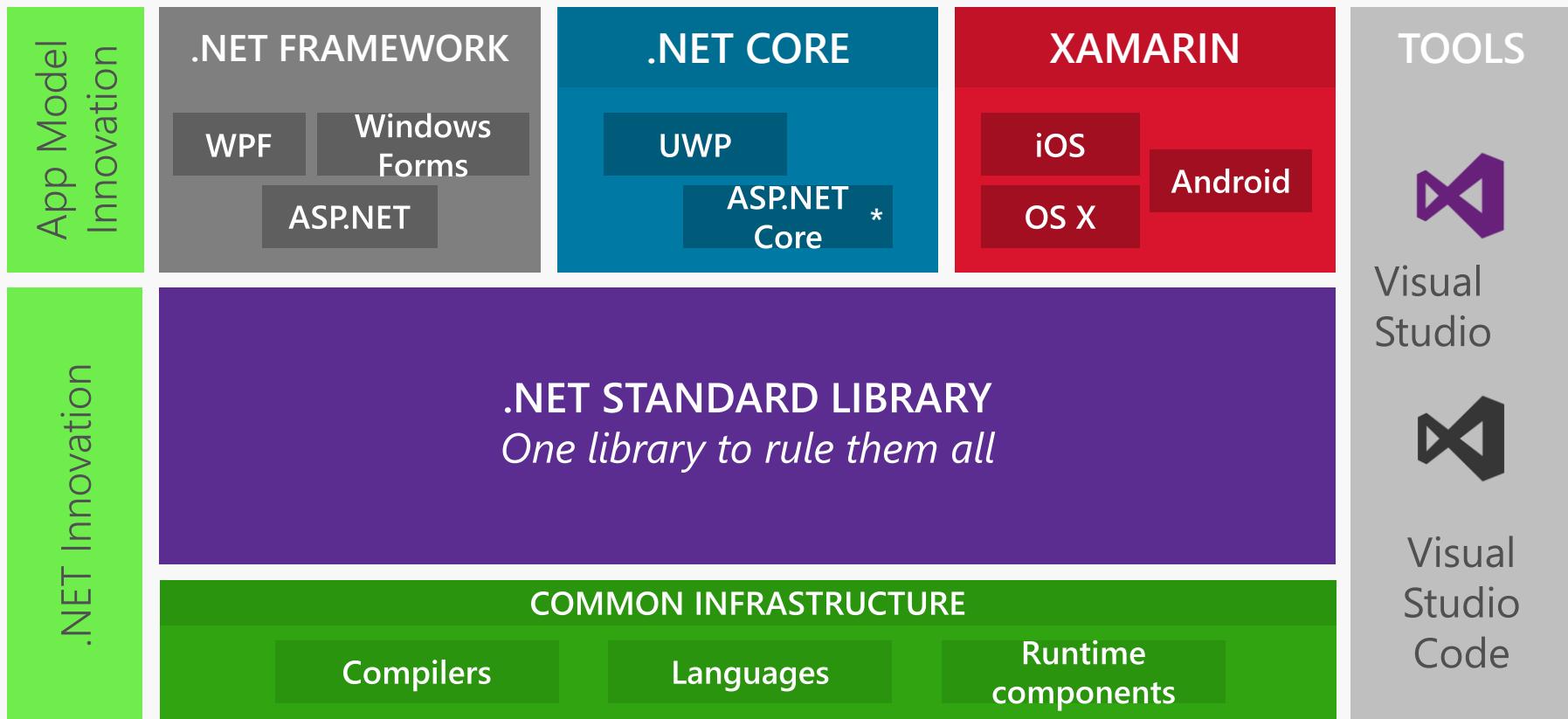
.NET Core 1.0

- A nagy keretrendszer egy forkja, amit erősen refaktoráltak
- Csak egy minimális BCL tartozik hozzá
 - > Gyorsabb: pl hideg indítás: 13 sec helyett 3 sec
- Moduláris
 - > minden további funkció NuGet csomagokban
 - > A csomag neve = a szerelvény neve
- Nyílt forráskódú
- Cross-platform: Windows, Linux, Mac OSX
- Side-by-side telepítés

.NET



.NET innovációs terek



.NET Platform Standard

- .NET Standard Library
 - > Mint a C++ standard lib
- Egy „API” minden .NET környezetben és platformon
 - > Csak API contract, nincs implementáció
 - > A megírt kód mindenhol újra felhasználható
 - > **A ráépülő könyvtárak mindenütt használhatóak**
- Megvalósítás
 - > netstandard.dll, referencia szerelvény (assembly)
 - > Type-forwarding: a platform adja az implementációt

.NET Standard 2.1 ~ 32k+ API

XML

XLinq • XML Document • XPath • Schema • XSL

SERIALIZATION

BinaryFormatter • Data Contract • XML

NETWORKING

Sockets • HTTP • Mail • WebSockets

IO

Files • Compression • MMF

THREADING

Threads • Thread Pool • Tasks

CORE

Primitives • Collections • Reflection • Interop • Linq

.NET Standardra épülő könyvtárak

- .NET Standard nélkül – túl sok verzió
 - > A könyvtárat külön le kell fordítani és publikálni minden .NET platformra, amit támogatni szeretnénk
 - > Sok „felesleges” munka, a kód gyakran nem is változik semmit – ha változik, az extra nehézség
- .NET Standard – optimális könyvtár fejlesztés
 - > A könyvtár megcélozza például a .NET Standard 2.0-t
 - > minden API-t használhat, ami ott fel van sorolva (pl HttpClient, ...)
 - > Az alkalmazás készítésénél használhatjuk a könyvtárat, ha az alkalmazás célplatformja (pl Unity / Xamain / ...) támogatja azt a .NET Standard verziót

.NET Standard

- A platformok egységesítése
 - > A jövőben töredézettség elkerülése
- minden jövőbeni platform verziójának ezen standardok mentén kell fejlődni

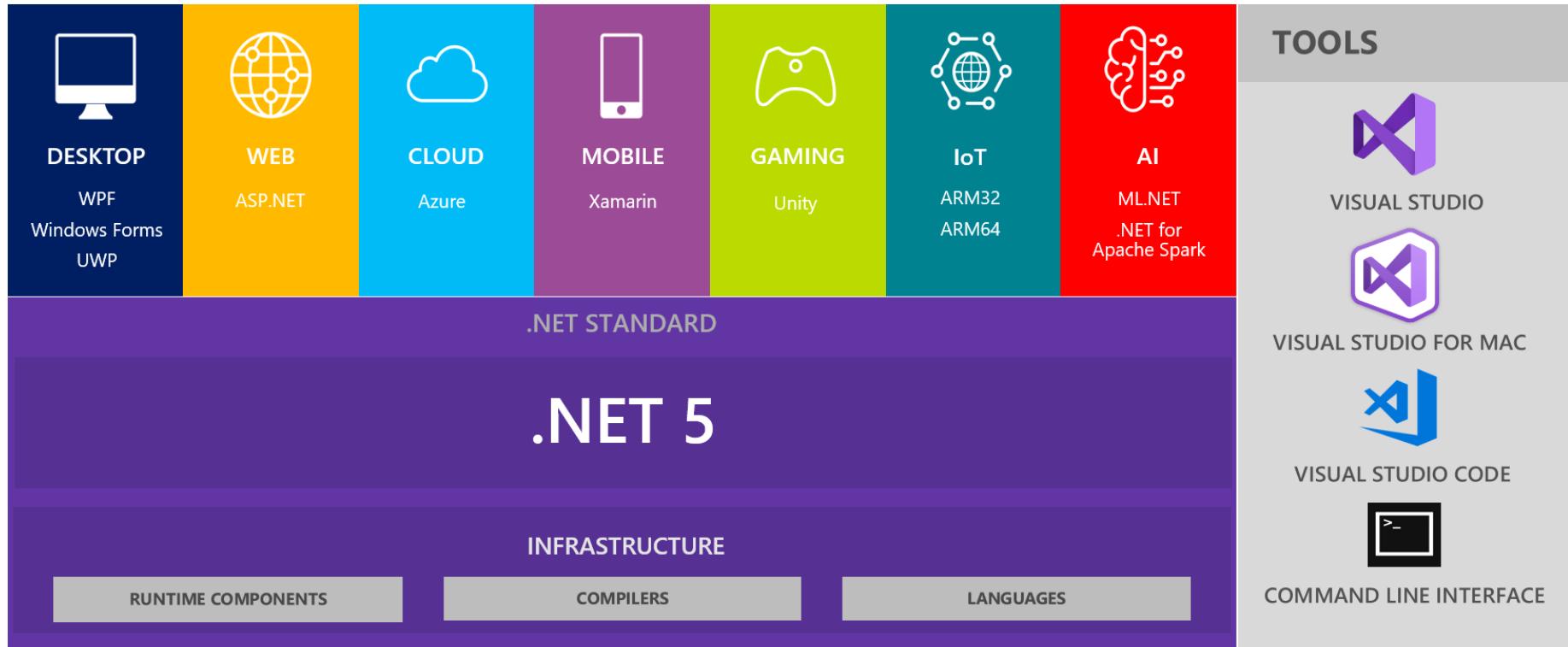
Miért lett nyílt forráskódú?

- Cross-platform
 - > Ez nem egy technikai követelmény, de tipikus
 - > „Olcsóbb”
- Gyorsabb bugfixing, jobb dizájn
 - > Rögtön kapnak visszajelzést: hibák, API struktúra, ...
 - > Több lehetőség próbálkozásra
- Hatékonyabb a Microsoft belső működése
 - > minden csapat egyszerűen hozzáfér a kódhoz...
- A .NET Framework miért csak „source open”?
 - > Túl sok gépen van rajta, a kompatibilitás túl fontos

A .NET story

- Indulás 2002 éve: .NET Framework
 - > Túl monolitikus, több .NET implementáció 🤦
- .NET Core 2017
 - > Refaktor, írjuk újra, legyen kicsi... 🤝
- .NET Standard
 - > Közös könyvtárak támogatása 🎉
- .NET 5: 2020
 - > Új, egységes .NET 🌟
 - > A .NET Core következő verziója

.NET – A unified platform



.NET 5

- „The One .NET”
- Egy platform
- Egységes .NET SDK
 - > Egyetlen Base Class Library minden platformon
 - > Mobil fejlesztés .NET 5-ön (Xamarin)
- Natív, felhős és webes alkalmazások

A „XAML” platform

- 2006. WPF – C#, .NET, Windows, Win32
- 2012. UWP – C++, Windows, WinRT
- 2014. Xamarin Forms – Android, iOS
- 2021? WinUI – XAML (UWP stílus)
 - > UWP és Win32 alkalmazások támogatása
 - > Windows + .NET 5
- 2022? MAUI – XAML (Xamarin stílus?)
 - > Xamarin -> Android, iOS, macOS, Windows
 - > Single project, Blazor, .NET 6

Kérdések?

Albert István
ialbert@aut.bme.hu

XAML 1

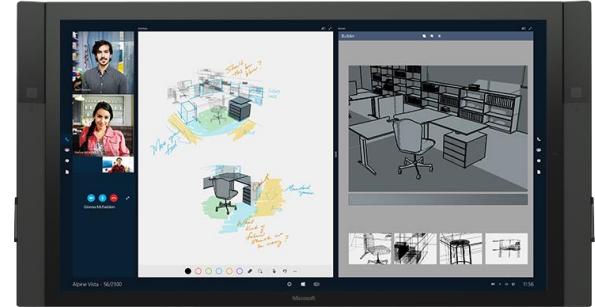
< Albert István >



Automatizálási és
Alkalmazott
Informatikai Tanszék

Miért XAML?

- Felhasználói felület összállítására
- Klasszikus asztali alkalmazás (WPF)
 - > LOB – üzleti terület
 - > Win32 elérése – nagy szabadság
- Universal Windows Platform
 - > Mobil, IoT
 - > Tablet + Hibrid
 - > Notebook, desktop
 - > Surface Hub, HoloLens, Xbox One



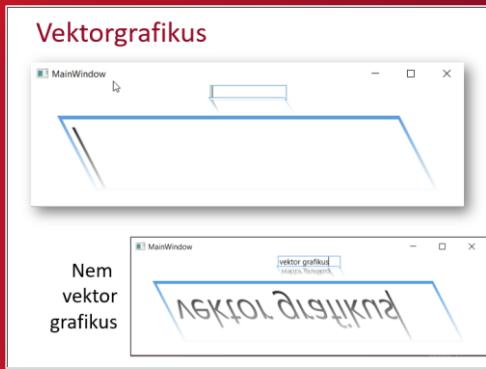
XAML történelem és változatok

- 2006, .NET Framework 3.0 – WPF
 - > A leggazdagabb XAML platform
 - > Felügyelt kódban
 - > Kényelmetlenségekkel
- 2007, Silverlight böngésző plugin
 - > Natív de butább
- 2010, Silverlight for Windows Phone 7
- 2012, WinRT, Windows 8
- 2014, Universal Windows Platform
- 2021, WinUI 3

Mire jó a XAML?

- Deklaratív, objektum fa példányosító nyelv
- Objektumok
 - > Tipikusan vezérlők
 - > Vezérlő tulajdonságok
- Fa
 - > Jól követhető hierarchia
- A vektor grafikus megjelenítés a rajzoló alrendszer feladata!

Tartalom



XAML és kód

- XAML elemek és attribútumok

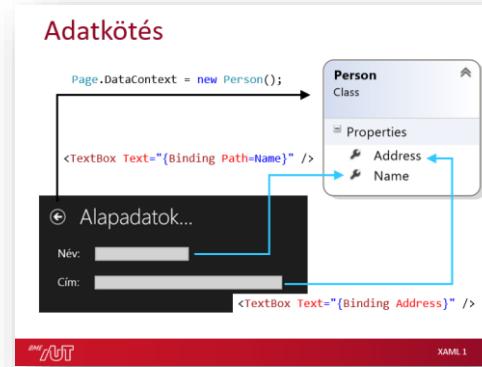
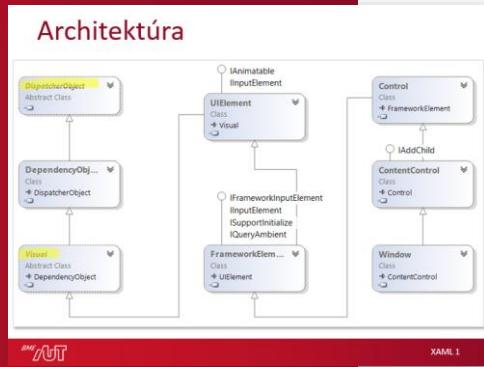
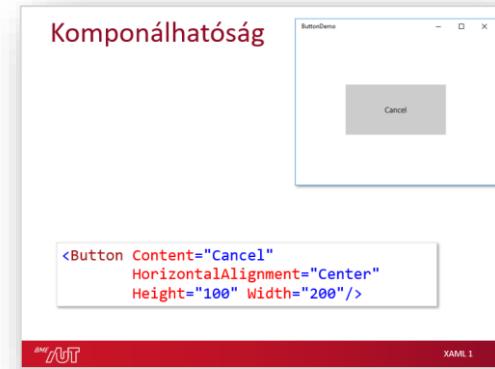
> XAML

```
<Button
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    Content="OK"
    Click="Button_Click"/>
```

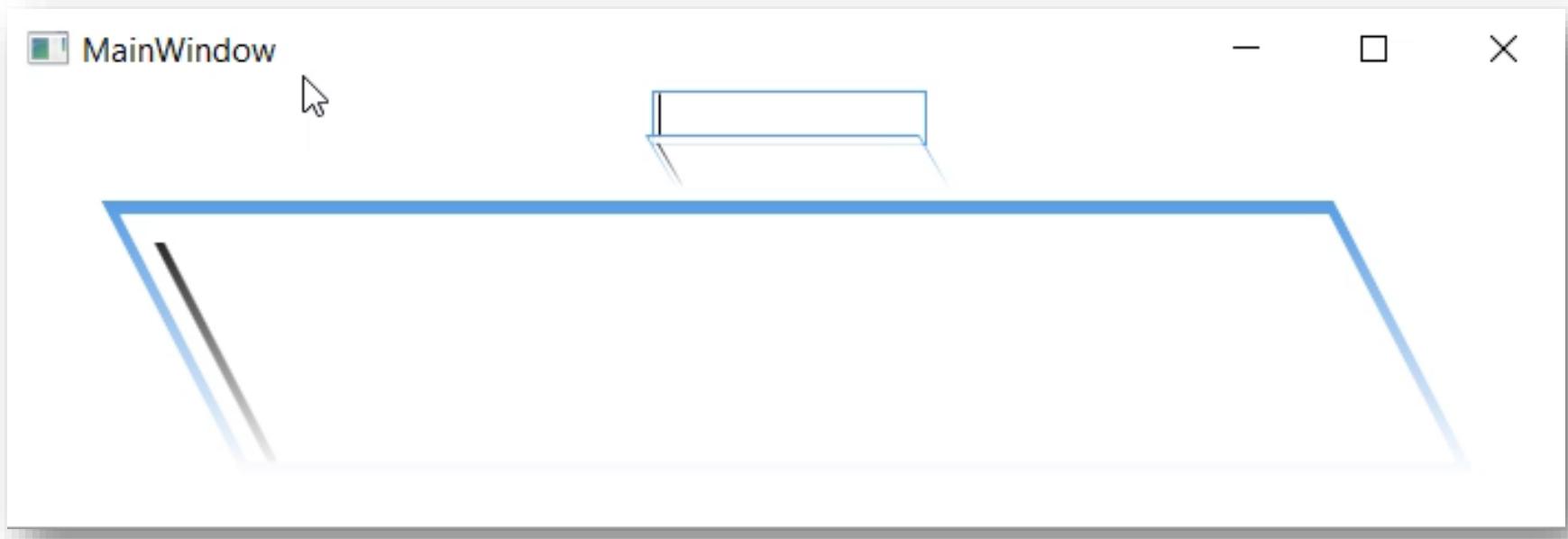
> C#

```
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

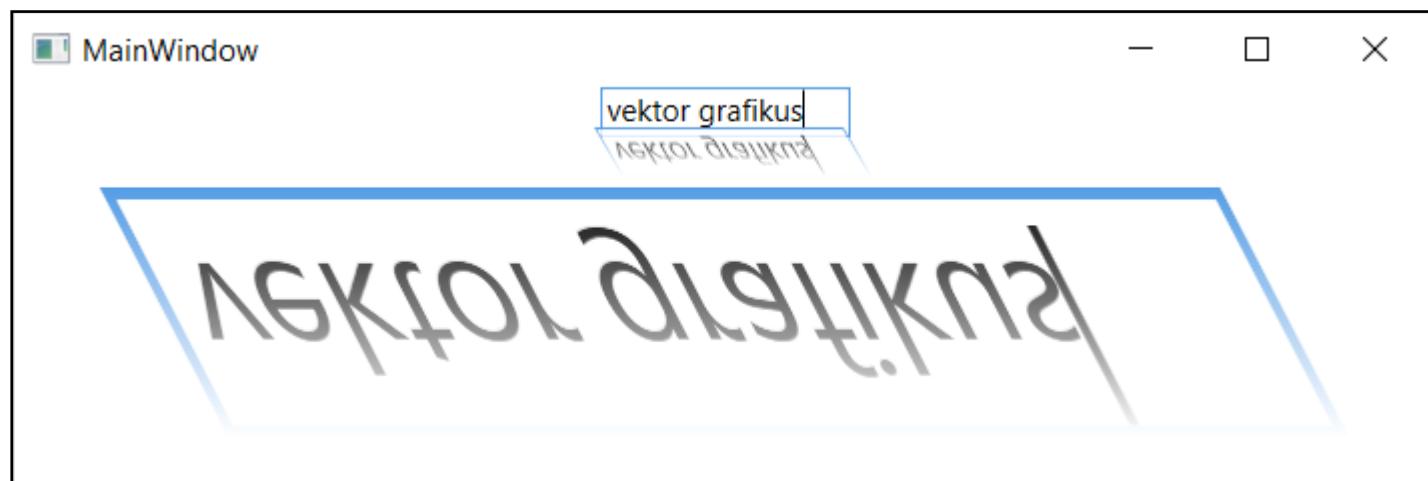
Button b = new Button();
b.Content = "OK";
b.Click += Button_Click;
```



Vektorgrafikus

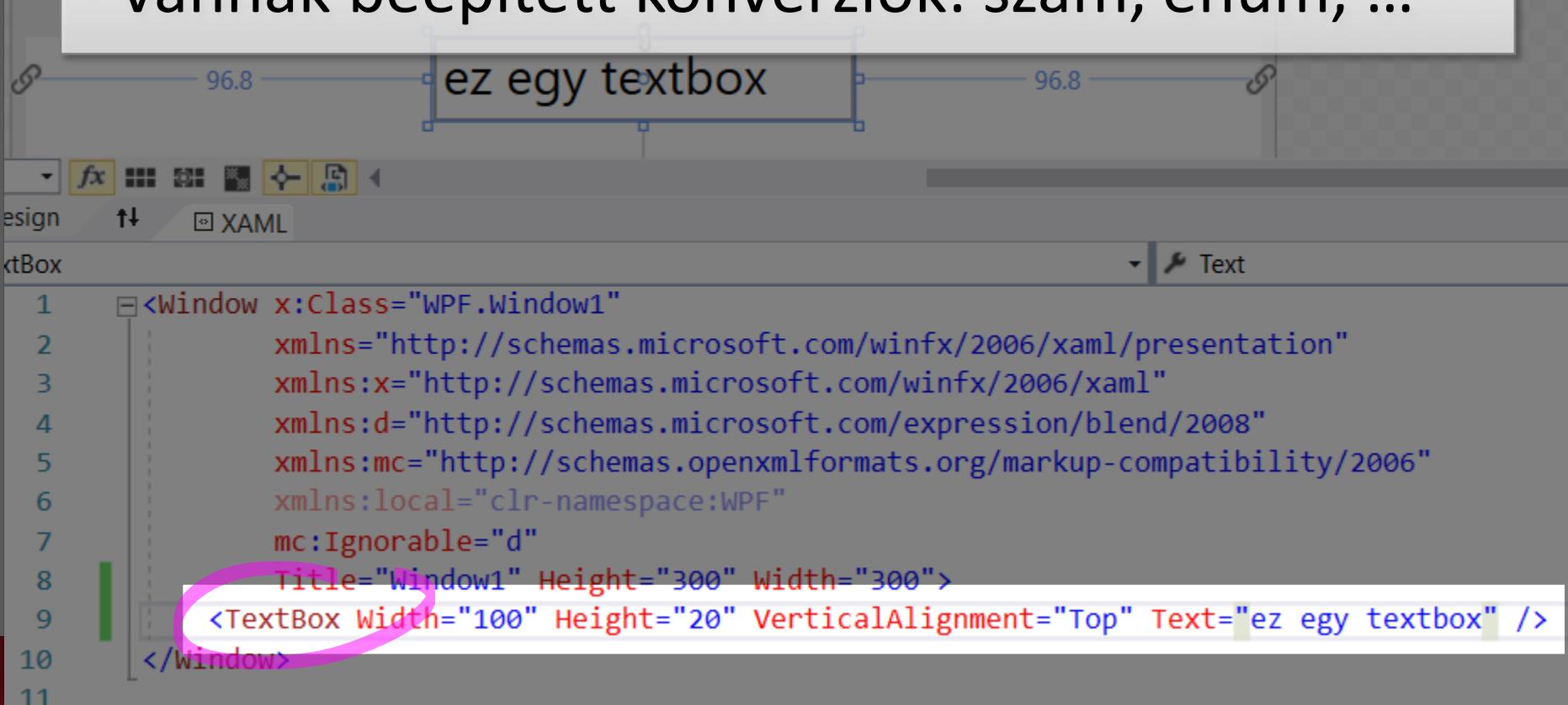


Nem
vektor
grafikus



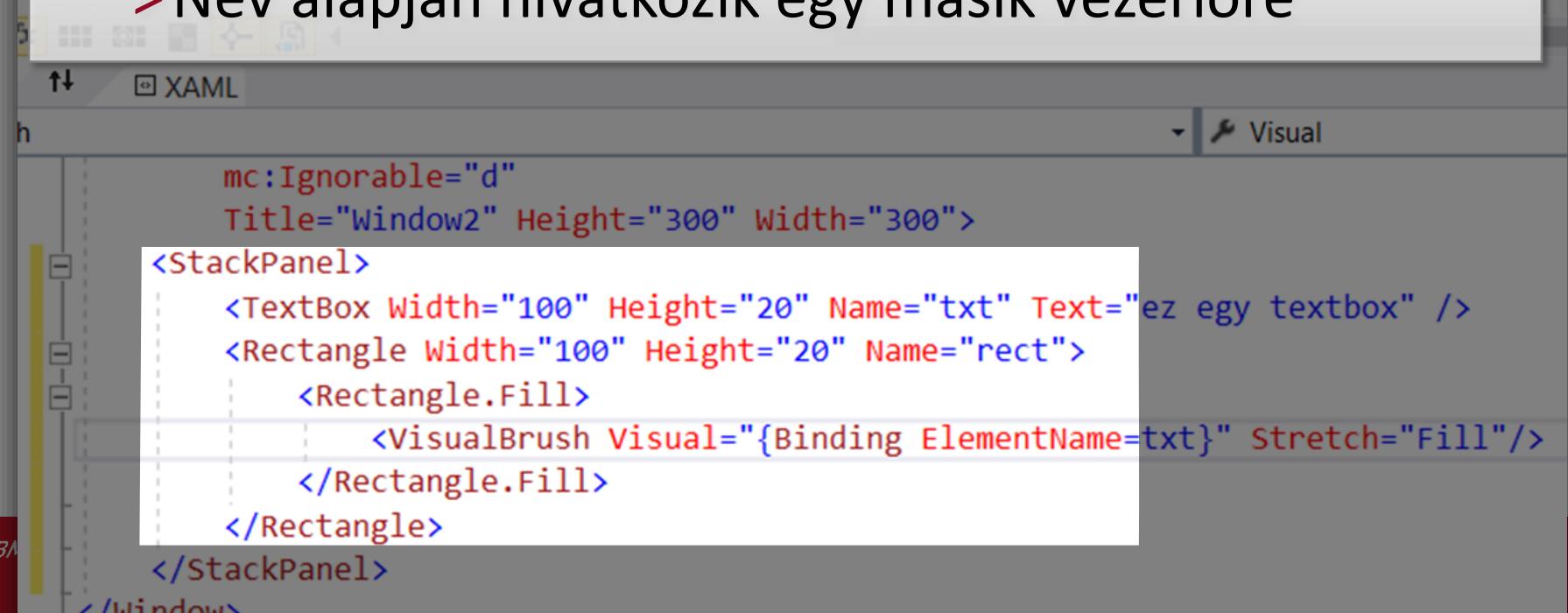
Objektum példányosítás

- A tegek példányosítandó osztály nevek
- Az attribútumok tulajdonságok
- Vannak beépített konverziók: szám, enum, ...



Hierarchia, összetett tulajdonságok

- A StackPanel egymás alá rendezi az elemeket
- A Rectangle Fill tulajdonsága nem egyszerű szöveggel van megadva
 - > Külön objektum van a tulajdonsághoz rendelve
 - > Név alapján hivatkozik egy másik vezérlőre

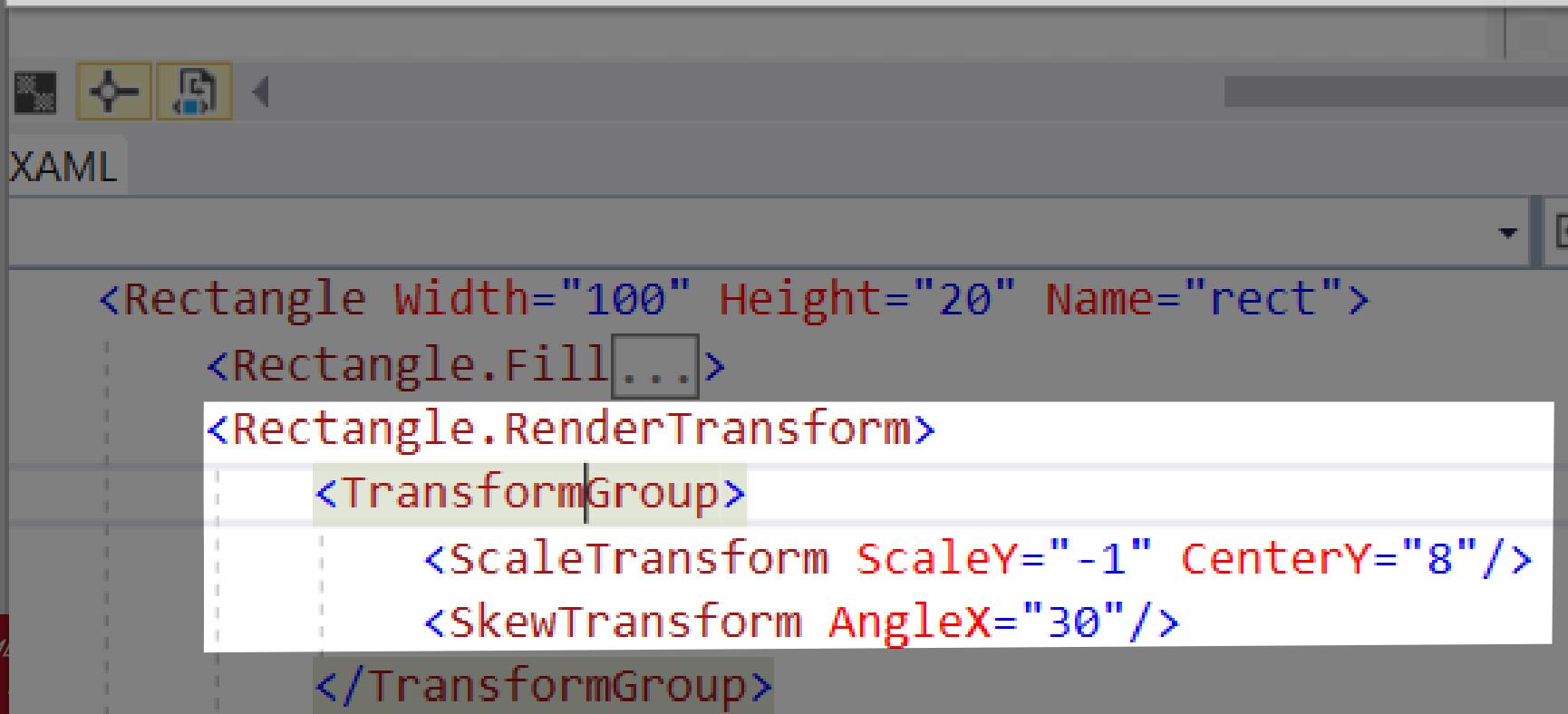


The screenshot shows a development environment with a toolbar at the top, followed by tabs for 'XAML' and 'Visual'. Below the tabs is a code editor displaying XAML code. The code defines a window with a title, height, and width. Inside the window, there is a stack panel containing a text box and a rectangle. The rectangle's fill is set to a visual brush that binds to the text box's element name. The 'Visual' tab is selected, showing a preview of the window with its components.

```
mc:Ignorable="d"
Title="Window2" Height="300" Width="300">
<StackPanel>
    <TextBox Width="100" Height="20" Name="txt" Text="ez egy textbox" />
    <Rectangle Width="100" Height="20" Name="rect">
        <Rectangle.Fill>
            <VisualBrush Visual="{Binding ElementName=txt}" Stretch="Fill"/>
        </Rectangle.Fill>
    </Rectangle>
</StackPanel>
</Window>
```

2D affin transzformációk

- Tetszőleges objektum/vezérlő transzformálható
- Affin trafó: tükrözés, forgatás, eltolás, nyírás
- Nem mindegy a transzformáció sorrendje!



The screenshot shows a WPF application interface. At the top, there's a toolbar with icons for selection, transformation, and other tools. Below it is a 'XAML' tab. The main area displays XAML code for a rectangle with various transforms applied.

```
<Rectangle Width="100" Height="20" Name="rect">
    <Rectangle.Fill>...
```

The code block shows the XAML definition for a rectangle named "rect". It includes a fill definition and a RenderTransform section containing a TransformGroup with a ScaleTransform (ScaleY="-1", CenterY="8") and a SkewTransform (AngleX="30").

```

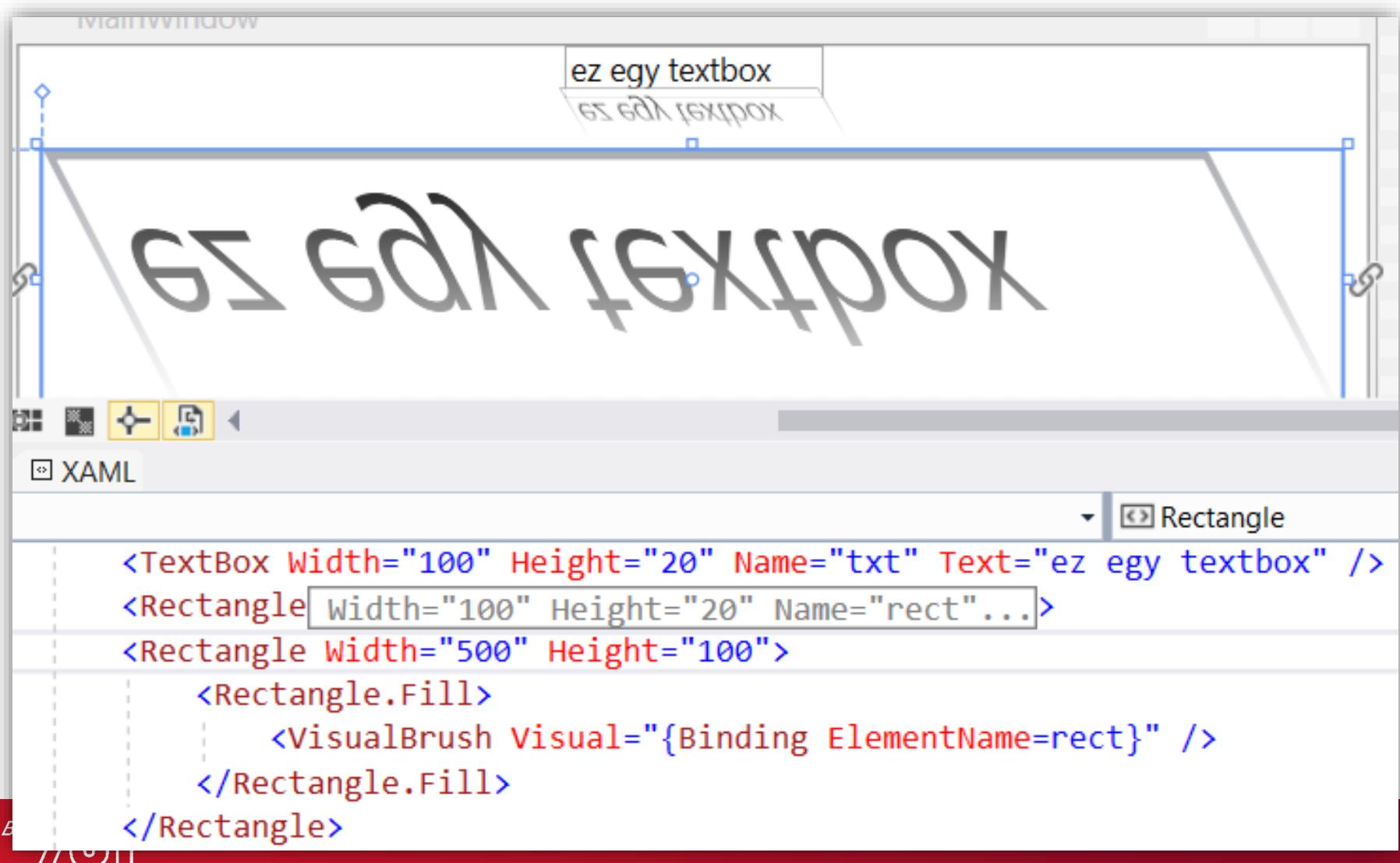
        <Rectangle.RenderTransform>
            <TransformGroup>
                <ScaleTransform ScaleY="-1" CenterY="8"/>
                <SkewTransform AngleX="30"/>
            </TransformGroup>
        </Rectangle.RenderTransform>
    </Rectangle>
```

Ecset, átmenetek, átlátszóság

- Rajz elem kitöltése: ecset – Brush leszármazottak
 - > Lineáris és radiális átmenetek
 - > Átmenet pontok specifikálása
- A szín minden esetben tartalmaz alfa csatornát!
- Minden vezérlőhöz (UIElement) tartozik
OpacityMask tulajdonság

```
<Rectangle Width="100" Height="20" Name="rect">
    <Rectangle.Fill>...
    <Rectangle.OpacityMask>
        <LinearGradientBrush EndPoint="0 1">
            <GradientStop Offset="0" Color="#00000000"/>
            <GradientStop Offset="1" Color="Black"/>
        </LinearGradientBrush>
```

Vektor grafikus nagyítás



XAML és kód

- XAML elemek és attribútumok

> XAML

```
<Button  
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
    Content="OK"  
    Click="Button_Click"/>
```

> C#

```
using Windows.UI.Xaml;  
using Windows.UI.Xaml.Controls;  
  
Button b = new Button();  
b.Content = "OK";  
b.Click += Button_Click;
```

CLR típusok hivatkozása: UWP

- Saját névtér használata

```
<Page  
    x:Class="App3.Views.MainPage"  
    xmlns:local="using:App3.Views"  
    xmlns:controls="using:MyControls.Controls">  
  
    <controls:MyControl />  
</Page>
```

- A „local” szokott hivatkozni az aktuális osztály névterére
- Másik szerelvény használatához sem kell több, csak a névtér és a hivatkozás

CLR típusok hivatkozása: WPF

- Saját névtér használata

```
<Page
    x:Class="App3.Views.MainPage"
    xmlns:sys="clr-namespace:System; assembly=mscorlib">

    <sys:String>Hello</sys:String>
</Page>
```

- Nem elég a névtér, ki kell írni a hivatkozott szerelvényt!
- Az assembly elhagyható, ha azonos szerelvényben van a típus
 - Használható XmlNsDefinitionAttribute szerelvény szinten

Markup kiterjesztések

- A tulajdonságok megadásánál használt '{' jel egy markup kiterjesztést aktivál

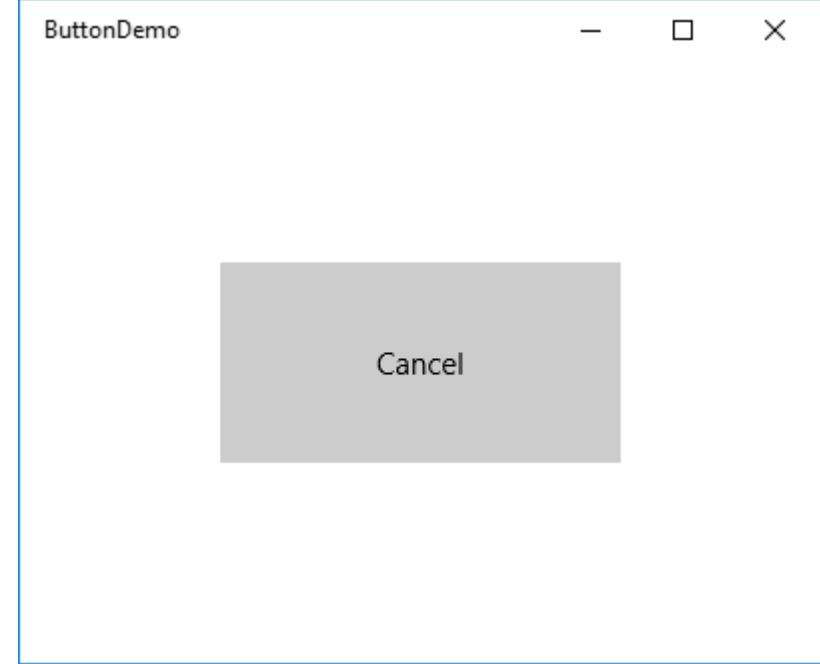
```
<TextBlock Text="{Binding}" ...>
```

- Ezek a kifejezések egymásba ágyazhatók
- Tipikusan speciális hivatkozásokat jelentenek
 - > Adatkötés
 - > Erőforrás hivatkozás
 - > Sablon hivatkozás

XAML metódusok

- **FindName**
 - > Név alapján keres vezérlőt
 - > Az InitializeComponent is ezt használja
- **XamlReader . Load**
 - > XAML fájl betöltése és objektum gráf építés dinamikusan

Komponálhatóság



```
<Button Content="Cancel"  
       HorizontalAlignment="Center"  
       Height="100" Width="200"/>
```

Gomb 2

ButtonDemo

- □ ×

Cancel (Are you sure?)

```
<Button HorizontalAlignment="Center"  
        Height="100" Width="200">  
    <TextBlock>  
        <Run Text="Cancel"/>  
        <Run FontWeight="Bold"  
            Text="(Are you sure?)"/>  
    </TextBlock>  
</Button>
```

Gomb 3



```
<Button HorizontalAlignment="Center"
        Height="100" Width="250">
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Cancel" Margin="0,0,10,0,,"
                   VerticalAlignment="Center"/>
        <CheckBox Content="(Are you sure?)" 
                  FontWeight="Bold"/>
    </StackPanel>
</Button>
```

Gomb 4

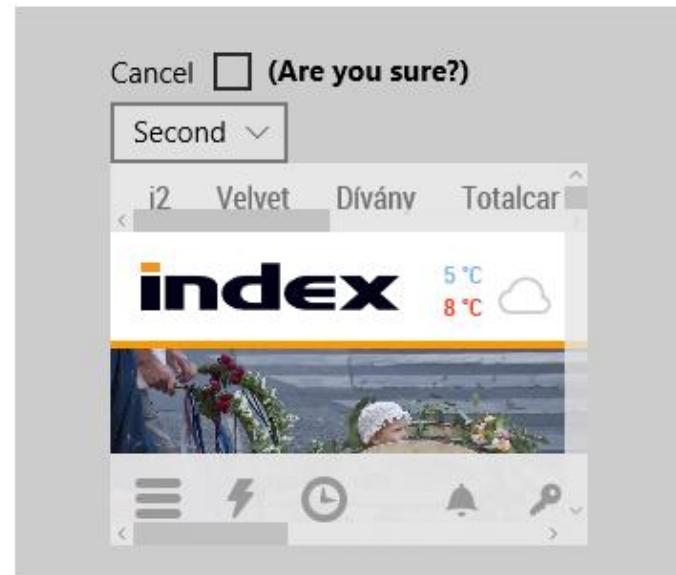
ButtonDemo

- □ ×



```
<Button HorizontalAlignment="Center" Height="100">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Cancel" ... />
            <CheckBox Content="(Are you sure?)" />
        </StackPanel>
        <ComboBox>
            <ComboBoxItem>First</ComboBoxItem>
            <ComboBoxItem>Second</ComboBoxItem>
            <ComboBoxItem>Third</ComboBoxItem>
        </ComboBox>
    </StackPanel>
</Button>
```

Gomb 5



```
<Button HorizontalAlignment="Center" Height="300" Width="350">
    <StackPanel Orientation="Vertical">
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="Cancel" Margin="0,0,10,0"
                VerticalAlignment="Center"/>
            <CheckBox Content="(Are you sure?)" FontWeight="Bold"/>
        </StackPanel>
        <ComboBox>
            <ComboBoxItem>First</ComboBoxItem>
            <ComboBoxItem>Second</ComboBoxItem>
            <ComboBoxItem>Third</ComboBoxItem>
        </ComboBox>
        <WebView Source="http://index.hu/" Height="200" Width="250" />
    </StackPanel>
</Button>
```

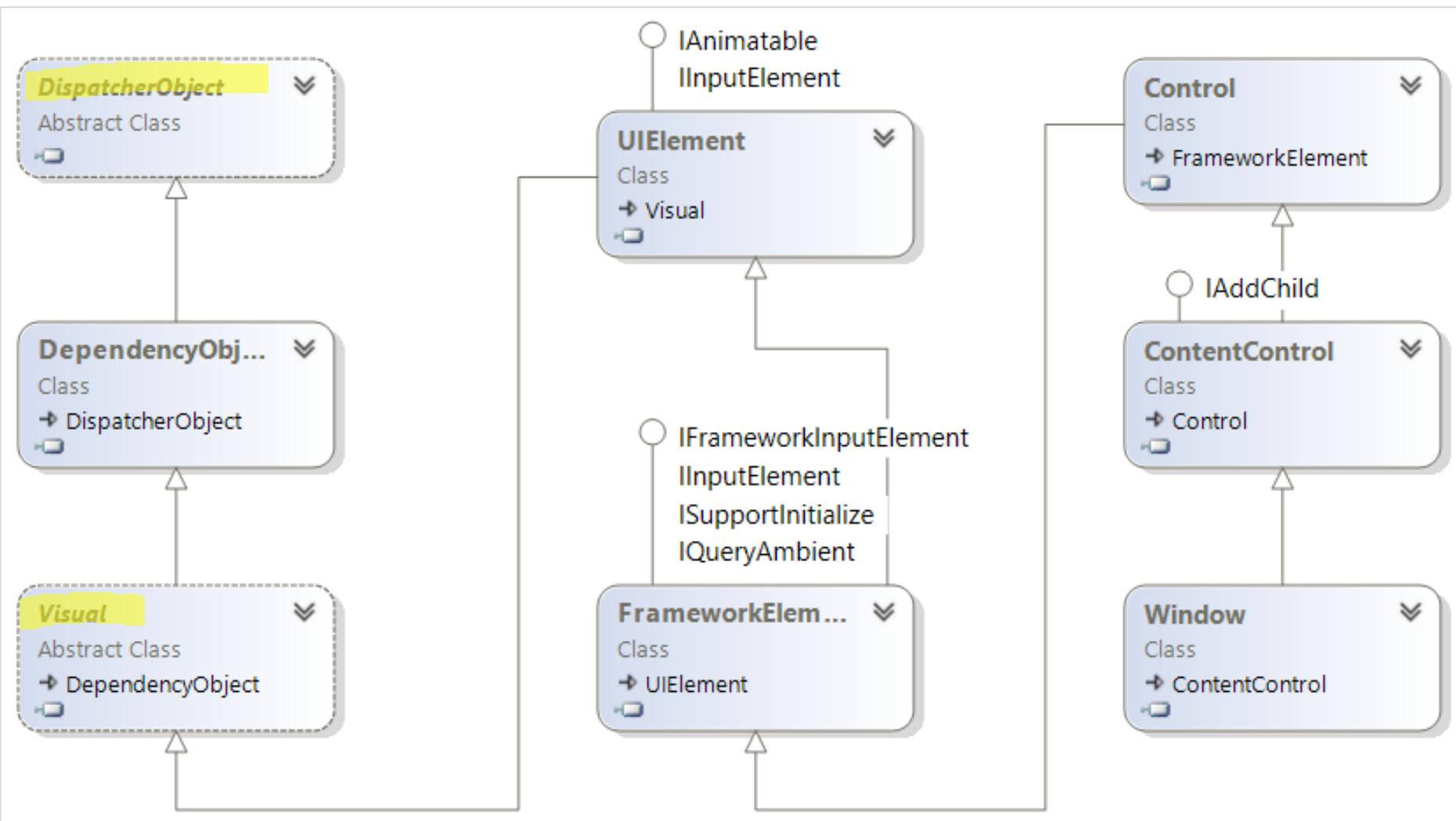
Komponálhatóság

- A vezérlők tetszőleges tartalmat meg tudnak jeleníteni
 - > Például: Button, Content
 - > Content bármi lehet: szöveg, kép, más vezérlők:
 - StackPanel + szöveg + videó
- Az összetett vezérlők más vezérlőkből állnak
 - > Az összeépítésre nagyon kevés megkötés van
 - > Például: ListView
 - ScrollView a görgetéshez
 - minden listaelem egy másik tetszőleges vezérlő

Testreszabható megjelenés

- Deklaratívan, objektumokkal leírható az egyes vezérlők megjelenése
 - > Vonal, háttér stb.
- Leírhatók a különböző állapotokhoz tartozó változások
 - > Például letiltott vezérlő
- Az állapotok eseményekhez köthetők
 - > Például megnyomják a gombot
- A változások animálhatók

Architektúra



Architektúra – egyszálú

- Használj **async**-ot!
- minden objektum (vezérlő stb) egy szálhoz van rendelve, csak arról a szálról elérhető
 - > Más szálról hívott műveletek kivételt dobnak
 - > A Dispatcher tartja nyilván
- UWP - CoreDispatcher / WPF - DispatcherObject
 - > Központi üzenet kezelő komponens
 - > minden UI vezérlő hivatkozik a sajátjára
 - > Szál ellenőrzés: HasThreadAccess
 - > RunAsync/Invoke: aszinkron végrehajtás
 - > ProcessEvents/Run: üzenetek végrehajtása

Dispatcher példa (WPF)

```
1 reference
void calculatePI()
{
    double pi = 2;
    const int n = 50;
    for (int i = n; i > 0; i--)
        pi = pi * i / (2 * i + 1) + 1;
    pi *= 2;

    this.Dispatcher.InvokeAsync(() => txt.Text = pi.ToString() );
}

0 references
private void Window_Loaded(object sender, RoutedEventArgs e)
{
    Task.Run((Action)calculatePI);
}
```

DependencyProperty

- Egységes tulajdonság tároló rendszer
 - > Adatkötés, adatkötéshez értesítés
 - > Stílusok kezelése
 - > Animáció
 - > Öröklés
 - > Alapértelmezett értékek
 - > Futás időben egyes objektumokhoz felvehető és lekérdezhető értékek
- DependencyObject alaposztály
 - > Kb minden „XAML” osztályhoz

DependencyProperty használata

- Hagyományos propertyként jelenik meg
 - > Mind XAML-ben mind C#-ban

```
<Grid>
    <TextBox Height="20" Width="100" VerticalAlignment="Top" Text="alma" Name="txt" />
</Grid>
```

- Használhatjuk a kulcs-érték pár beállítást is

```
txt.Text = "narancs";
DependencyObject d = txt;
d.SetValue(TextBox.TextProperty, "kávé");
```

(field) DependencyProperty TextBox.TextProperty
Identifies the TextBox.Text dependency property.

DependencyProperty deklarálása 1

```
public class Button : ButtonBase
{
    public bool IsDefault
    {
        get { return (bool)GetValue(Button.IsDefaultProperty); }
        set { SetValue(Button.IsDefaultProperty, value); }
    }

    public static readonly DependencyProperty IsDefaultProperty =
        DependencyProperty.Register(
            nameof(IsDefault),
            typeof(bool),
            typeof(Button),
            new PropertyMetadata(false, OnIsDefaultChanged));
}

private static void OnIsDefaultChanged(
    DependencyObject o, DependencyPropertyChangedEventArgs e) {}
```

DependencyProperty deklarálása 2

- Konvenció

```
public static readonly DependencyProperty xxxxProperty
```

- DependencyProperty.Register()

- > A visszaadott objektum csak egy **kulcs** ez érték globális tárolóból való elkéréséhez

- DependencyObject

- > GetValue(DependencyProperty dp)
 - > SetValue(DependencyProperty dp, object value)

- A csomagoló getter/setter és callback opcionális

- > A XAML miatt a csomagoló property szükséges, de a JIT fordító úgyis kioptimalizálja (reméljük)

Csatolt tulajdonságok (attached properties)

- Az objektumhoz tetszőleges kulcsú érték eltárolható
 - > Ugyanaz a DP tárolási mechanizmus
 - > Kulcs: a DP definíciójakor létrehozott statikus objektum
- Bárki eltárolhat bármit és lekérdezhet bármit
- Tipikusan vezérlő hierarchiában használjuk
 - > Canvas: rajta lévő elemek pozíciója
 - > Grid: rajta lévő elemek helye a táblázatban

Csatolt tulajdonságok használata

- Kicsit speciálisabb hívás, főleg C#-ban

```
<Canvas>
    <TextBox Canvas.Left="35" Canvas.Top="47"
        Height="20" Width="100" VerticalAlignment="Top" Text="alma" Name="txt" />
</Canvas>
```

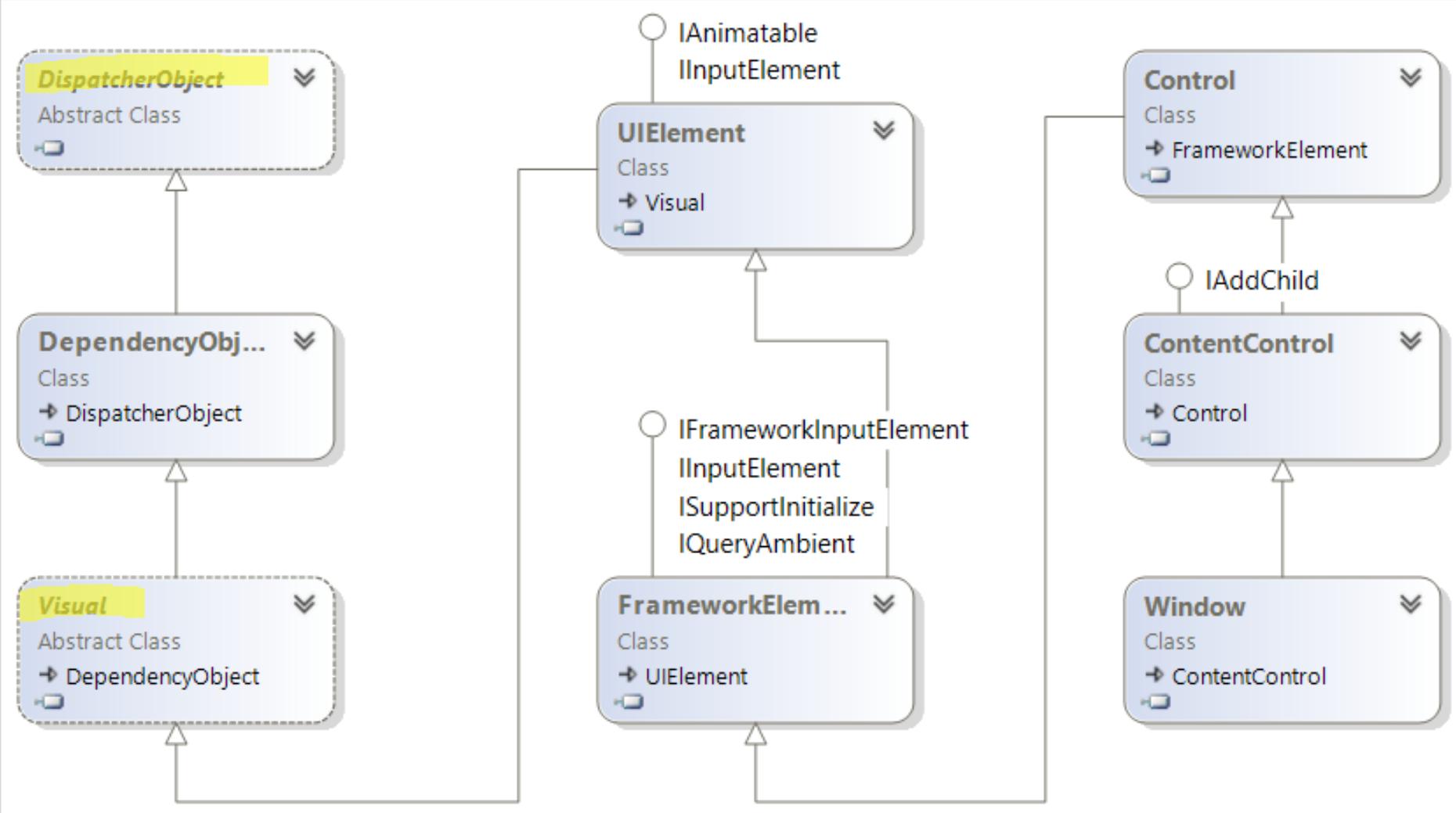
```
txt.Text = "narancs";
DependencyObject d = txt;
d.SetValue(TextBox.TextProperty, "kávé");

txt.SetValue(Canvas.LeftProperty, 30);
d.SetValue(Canvas.TopProperty, 42);
Canvas.SetLeft(txt, 33);
```

⊕ void Canvas.SetLeft(UIElement element, double length)

Sets the value of the System.Windows.Controls.Canvas.Left attached property for a given dependency object.

UI keretrendszer alaposztályai



DependencyObject

- object
- **DependencyObject** : object
 - > Dependency- és AttachedProperties
 - Get/Set/Clear, ...
- Dispatcher: melyik szálhoz tartozik
 - > HasThreadAccess: az szálból meg lehet-e hívni
 - > RunAsync: művelet végrehajtása a UI szálon
 - > ProcessEvents: UI szálon a klasszikus üzenetkezelő ciklus hívja, az eseményeket dolgozza fel

UIElement

- Object
- **DependencyObject** : object
- **UIElement** : DependencyObject
 - > Van vizuális megjelenése
 - > Beérkező inputokat kezel

UIElement

- Megjelenés, láthatóság
 - > Transformáció
 - > Tranzíciók
- Befoglaló téglalap, alap layout koncepció
 - > Két fázis: Measure + Arrange
- Eseménykezelés
 - > Drag&Drop
 - > Fókusz és billentyűzet események
 - > Érintés és egér események

FrameworkElement

- Object
- **DependencyObject**: object
- **UIElement**: DependencyObject
- **FrameworkElement**: UIElement
 - > Implementált layout rendszer
 - > Életciklus események (Loaded)
 - > DataContext alapú adatkötés

FrameworkElement

- Layout
 - > Width, MinWidth, ActualWidth, ...
 - > Horizontal/VerticalAlignment
 - > Margin
 - > LayoutUpdated, SizeChanged események
- Nevesített vezérlők: Name, FindName metódus
- Adatkötés (SetBinding)
- Stílus kezelés
- Lokális erőforrásgyűjtemény
- Loaded / Unloaded események
 - > A fő logikai fához tartozás eseményei

Control

- Object
- **DependencyObject**: object
- **UIElement**: DependencyObject
- **FrameworkElement**: UIElement
- **Control** : FrameworkElement
 - > Saját vezérlő sablon

Control

- Vezérlő-szerű működés
 - > Saját megjelenés
 - > Logika és megjelenítés szétválasztása vezérlősbablonnal
 - > IsEnabled
 - > Fókusz állítás és TabStoppok
- Tartalom koncepció
 - > Padding, HorizontalContentAlignment, ...
 - > Háttérszín és keret
 - > Tipográfiai beállítások

DP-k szerepe az öröklési hierarchiában

- **DependencyObject**
 - > Központosított tár, alapértelmezett érték
 - > Invalidálás, értesítés
 - > Csatolt tulajdonságok (attached properties)
 - > Adatkötés (binding)
- **UIElement : DependencyObject**
 - > Animáció
- **FrameworkElement : UIElement**
 - > Stílusok
 - > Öröklés

A megjelenés testreszabása

- minden vezérlő **tartalma** testreszabható
- többféle tartalom a vezérlő típusától függően
 - > Egyszerű, például Button (: ContentControl)
 - > Listás, például ListView (: Selector : ItemsControl)
- A tartalmat a ContentPresenter jeleníti meg
 - > Alapértelmezett tartalom: ToString() – szöveg
 - > De testreszabható!

ContentControl : Control

- Content tulajdonság (object): a tartalom
 1. UIElement: megjeleníti azt az objektum fát
 2. ContentTemplate adott: használja a sablont
 3. ToString-et használ
- ContentTemplate (DataTemplate)
 - > A tartalom megjelenítését lehet vele testreszabni
- ContentTemplateSelector: sablonválasztó logika
- ContentTransitions: animáció...

ButtonBase : ContentControl

- Kezeli az egéreseményeket
 - > Click esemény
- Tulajdonságok
 - > IsPressed
 - > ClickMode: Release, Press, Hover
- Integrálódik a Command mintával (MVVM)
 - > Command, CommandParameter, CommandTarget

Button, RepeatButton, HyperlinkButton

- Button

- > Nincs IsDefault, IsCancel
- > Flyout: amit meg kell jeleníteni...

Submit Query

OK

- RepeatButton

- > Delay property
- > Interval property
- > Nyomva tartva többször lövi el a Click eseményt

- HyperlinkButton

- > NavigateUri, automatikusan böngészőt nyit

[index.hu](#)

ToggleButton, CheckBox

- **ToggleButton : ButtonBase**
 - > IsChecked: lehet null is
 - > IsThreeState
 - > Checked, ... események
 - **CheckBox : ToggleButton**
 - > Saját megjelenése van
 - **RadioButton : ToggleButton**
 - > Saját megjelenése van
 - > GroupName-mel lehet csoportba foglalni azokat amiből csak egyet lehet kiválasztani
- All of above

All of above

All of above

All of above

ItemsControl : Control

- Egyszerű listás megjelenítés kiegészítő funkciók nélkül (törzselés, kijelölés, elemkattintás)
 - > Gyakran használjuk, ha listát kell csak megjeleníteni
- ItemsSource: a megjelenítendő gyűjteményre mutat
- ItemTemplate: az elemek megjelenését írja le
- ItemsPanel: elem elrendező vezérlő
- GroupStyle: csoportok stilizálása
 - > Fejléc, tartalom, tartalom elrendező panel
 - > GroupStyleSelector: elemenként eltérő megjelenés
- ItemTemplateSelector: dinamikus ItemTemplate saját logika alapján

Container osztályok

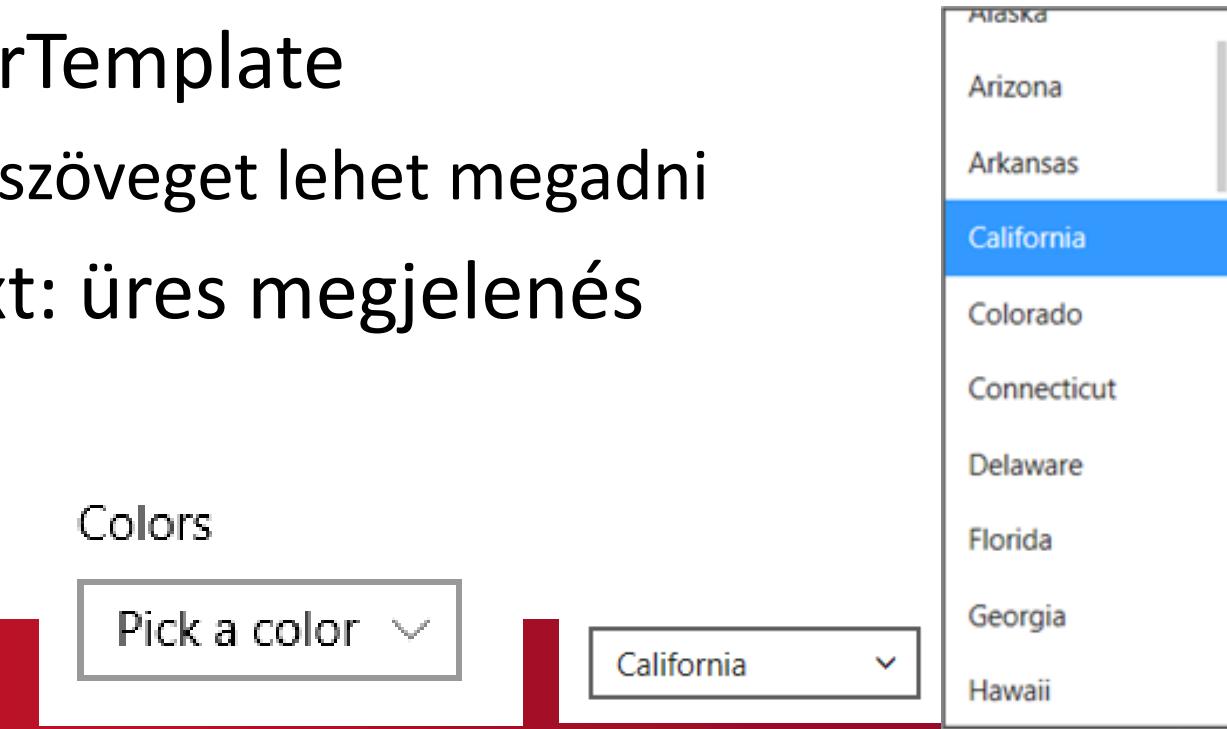
- minden megjelenített listaelemhez van egy Container osztály, ami egy ContentControl
 - > Pl ListViewItem, ComboBoxItem, ...
- Az ItemContainerGenerator-on keresztül lehet elérni a container itemeket
- A container elemek felelősek a kiválasztott elem háttérszínéért, stb.
 - > ItemContainerStyle, ...Selector, ...Transitions
 - > GetContainerForItem, PrepareContainerForItem...
- Például a kijelölést nem a beépített módon szeretnénk jelölni

Selector : ItemsControl

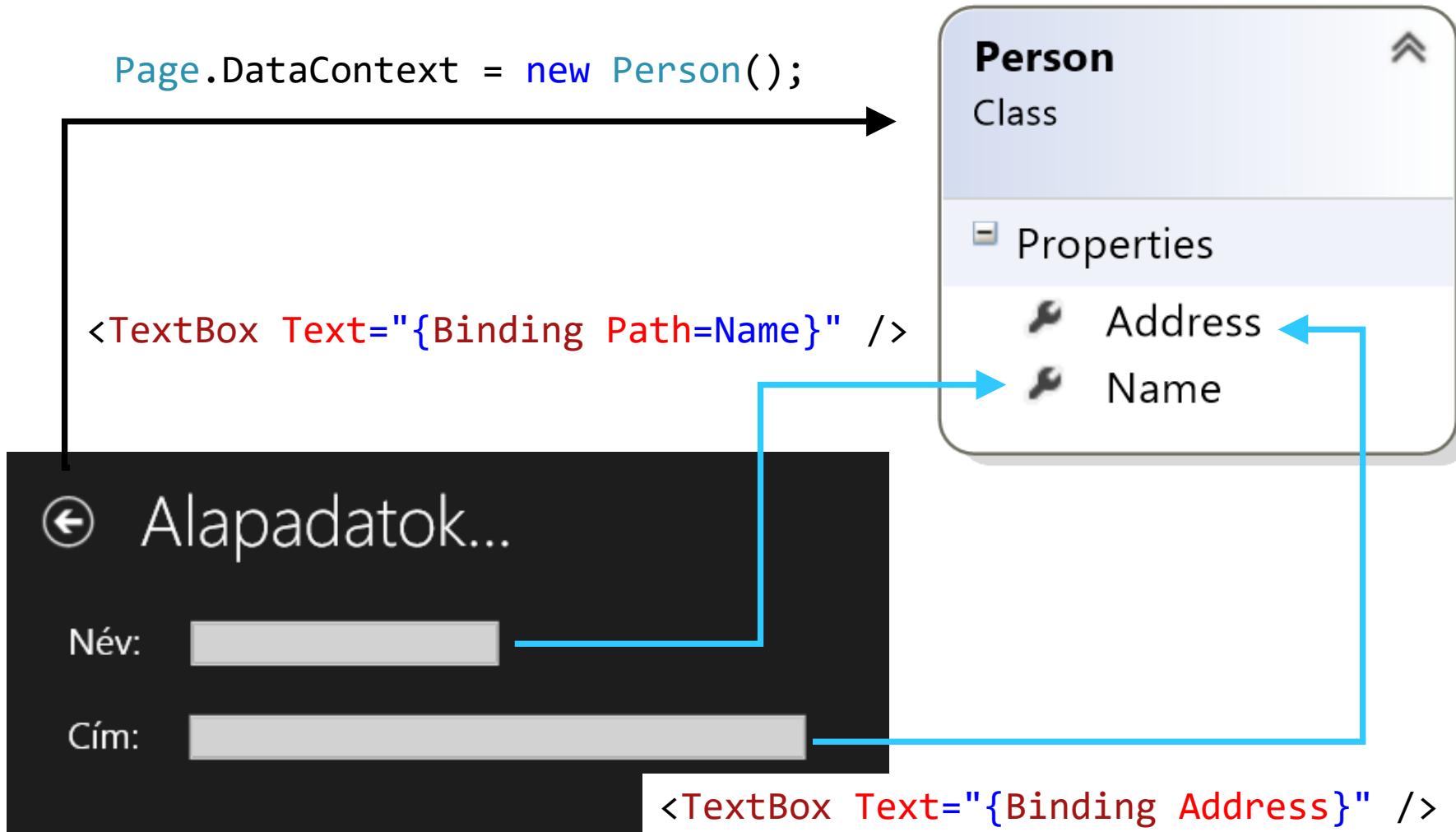
- Az elemek kiválaszthatóak
- SelectedIndex: 0 alapú index
- SelectedItem
 - > A kiválasztott objektum
- SelectedValue
 - > SelectedValuePath által hivatkozott tulajdonsága a kiválasztott objektumnak
- SelectionChanged esemény

ComboBox : Selector

- IsDropDownOpen
 - Programozottan is lenyitható
- DropDownOpened/Closed események
- IsEditable (=false ☹)
- Header, HeaderTemplate
 - A felette lévő szöveget lehet megadni
- PlaceholderText: üres megjelenés helyett

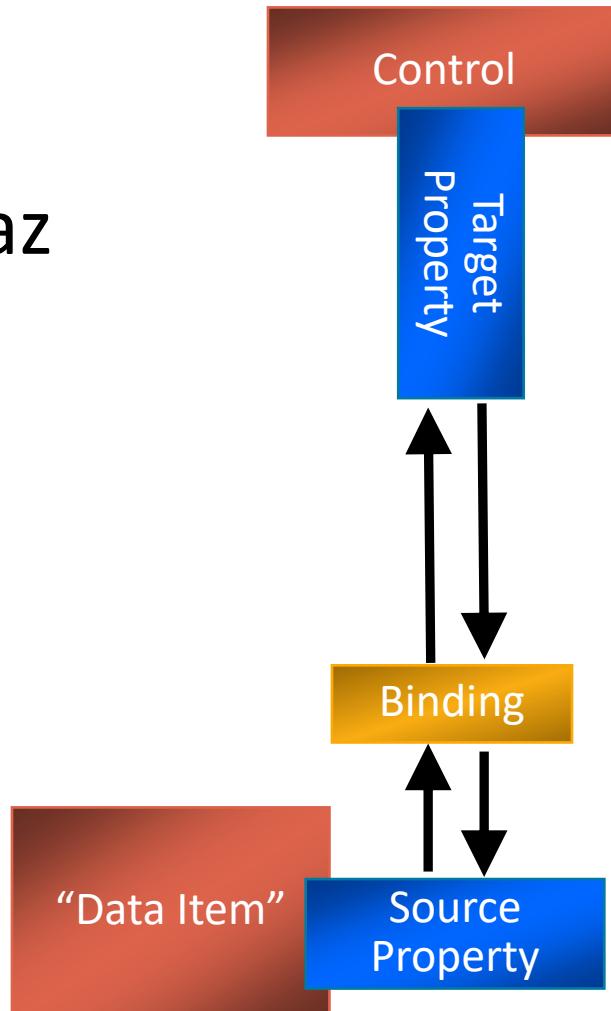


Adatkötés



Adatkötés XAML-ben

- Adatkötés célja: egy vezérlő tulajdonságainak hozzákötése az adatforráshoz
- Target: dependency property
- Source: tetszőleges property
- Szintaktika
 - > {Binding} markup extension



Binding osztály

- Az adatkötés jellemzőit írja le
- Binding osztály
 - > Source : forrás objektum, tetszőleges típus
 - > Path : forrás property, elmaradhat
- Cél objektum : csak DependencyObject
- Cél property : csak dependency property

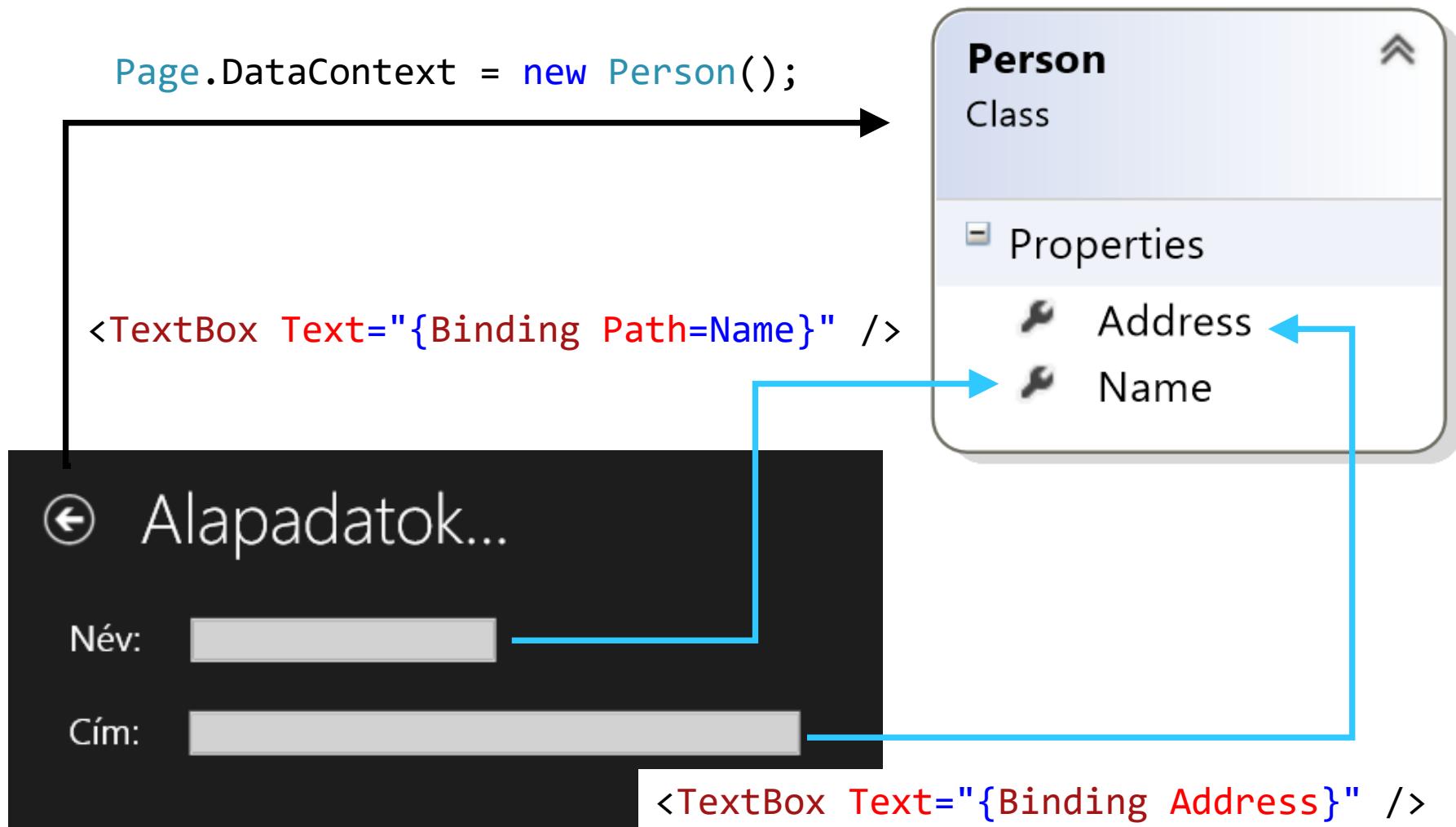
Binding . Source

- Tetszőleges CLR objektum
- Alapértelmezett forrás a logikai fában a cél objektumhoz rendelt DataContext értéke
 - > Ha a cél objektumnak nincs DataContextje explicit megadva, akkor a logikai fában a szülőtől örökli
 - > A Source explicit beállítható másik objektumra
- Nem állítható együtt az **ElementName** és **RelativeSource** tulajdonságokkal

Binding . Path

- Az adatforrás pontos elemét határozza meg
- **PropertyName: egyszerű tulajdonság név**
- A.B: elérési útvonal a propertyken keresztül
- (Canvas.Left): hivatkozás csatolt tulajdonságra
- Items[0]: indexelt elem
- „/”: gyűjtemény esetén az aktuális elem kiválasztása
- „.”: a teljes forrás objektum kiválasztása
- A fenti lehetőségek kombinálhatók
 - > pl „Items/s[1]”

Példa: source, DataContext, path



Példa source, explicit

- Explicit beállításra a leggyakoribb példa az erőforrás gyűjteményből történő hivatkozás

```
<Page.Resources>
    <local:Person x:Key="MyPerson" Name="Eric Cartman" Age="8"/>
</Page.Resources>

<TextBlock Text="{Binding Source={StaticResource MyPerson}, Path=Name}"/>
```

Binding . ElementName

- Az adatkötés forrása lehet egy másik vezérlő
- Az ElementName tulajdonság kell hivatkozzon a forrás vezérlő nevére

```
<Button Content="Press"  
       IsEnabled="{Binding ElementName=MyCheckBox, Path=.IsChecked}" />  
<CheckBox x:Name="MyCheckBox" Content="Enable my button"/>
```

Binding . RelativeSource

- Binding . RelativeSource beállítása
 - > Saját objektumon lévő propertyhez köt
 - > Tipikusan sablonokban, stílusokban használják
- Típusai (RelativeSourceMode):
 - > Self: az adat forrás megegyezik a cél objektummal
 - > TemplatedParent: sablonoknál az adatforrás
 - Ha nem akarsz konvertert stb. rákötni, akkor a {Binding XXX, RS={RS TemplatedParent}} helyett jobb a {TemplateBinding XXX}

Binding . Mode

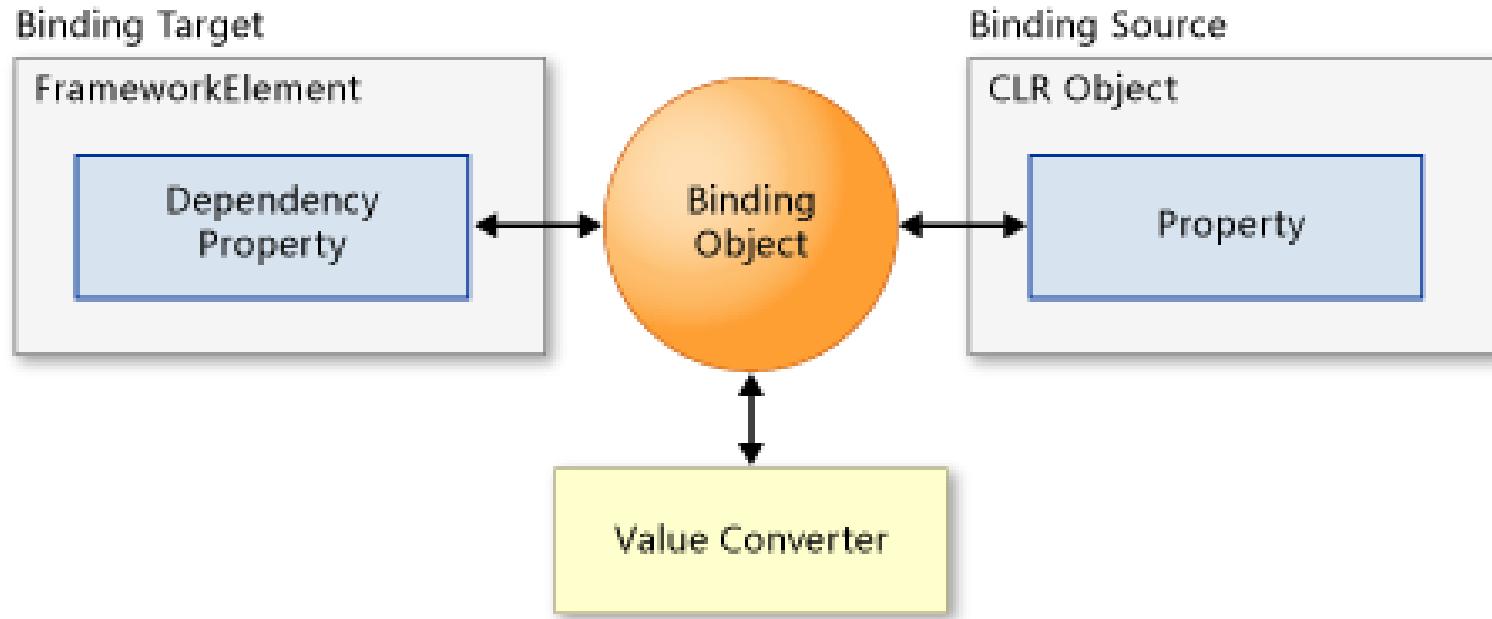
- Az adatkötés módját adja meg
- **TwoWay**: kétirányú kötés, a forrás- és cél property változása frissíti a másikat
- **OneWay**: csak a cél property frissül amikor a forrás megváltozik
- **OneTime**: a kötés létrehozásakor kerül végrehajtásra
- **OneWayToSource**: csak a forrás frissül, ha a cél megváltozik

Binding egyebek

- TargetNullValue
 - > Alapértelmezett érték ha az adatforrás null
- FallbackValue
 - > Ha a hivatkozott property nincs
- UpdateSourceTrigger
 - > Mikor kerüljön az adat frissítésre
 - > PropertyChanged: amikor a property megváltozott
 - > Explicit: csak amikor kódból meghívjuk

Binding . Converter

- Konvertálás: Binding.Converter
 - > Adatcsere előtt az adat transzformálható
 - > IValueConverter implementáció példány



IValueConverter

- Cél: Adatok konvertálása az adatkötés során

```
public interface IValueConverter
{
    object Convert(object value, Type targetType,
                  object parameter, string language);

    object ConvertBack(object value, Type targetType,
                      object parameter, string language);
}
```

- Paraméterek:
 - > Object: ez tartalmazza a konvertálandó értéket
 - > Type: Convert esetén a vezérlő tulajdonságának a típusa
 - > Object: opcionális paraméter
 - > Language: lokalizációhoz kultúra információ
- ConvertBack, csak ha kétirányú

IValueConverter

- Használat előtt példányosítani kell!
- Lépések:
 1. Konverter implementálása
 2. Konverterből statikus erőforrásként példány létrehozása
 3. Adatkötésnél konverter megadása

IValueConverter példa

```
public class BooleanToVisibilityConverter : IValueConverter
{
    public object Convert(object value, Type targetType, ...)
    {
        return (value is bool && (bool)value)
            ? Visibility.Visible : Visibility.Collapsed;
    }
    public object ConvertBack(object value, Type targetType, ...)
    {
        return value is Visibility
            && (Visibility)value == Visibility.Visible;
    }
}
```

IValueConverter példa

- Bool alapján vezérelt láthatóság

```
<Grid.Resources>
    <local:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
</Grid.Resources>

<TextBox Visibility="{Binding IsVisible,
    Converter={StaticResource BooleanToVisibilityConverter}}" />
```

Adatforrás változás jelzés

- Property Changes
 - > INotifyPropertyChanged interfész
 - > PropertyChanged event pattern
 - A szövegesen hivatkozott tulajdonság megváltozott
- Collection Changes
 - > INotifyCollectionChanged interfész
 - Új elemek vagy elemek mozgatása/törlése/változása, illetve reset
 - > ObservableCollection<T> : INCC

INPC példa, C# 6

```
class Person : INotifyPropertyChanged
{
    string name;
    public string Name
    {
        get { return name; }
        set
        {
            if(name == value) return;
            name = value;
            PropertyChanged?.Invoke(this,
                new PropertyChangedEventArgs(nameof(Name)));
        }
    }

    // INPC:
    public event PropertyChangedEventHandler PropertyChanged;
}
```

C# 5: Caller Info attribútumok

```
class Person : INotifyPropertyChanged
{
    string name;
    public string Name
    {
        get { return name; }
        set
        {
            if(name == value) return;
            name = value;
            OnPropertyChanged();
        }
    }

    void OnPropertyChanged([CallerMemberName] string prop = null)
    {
        PropertyChanged?.Invoke(this, new
PropertyChangedEventArgs(prop));
    }

    // INPC:
    public event PropertyChangedEventHandler PropertyChanged;
}
```

De ez így nagyon sok kód, nem?

- Általában kivezetjük őssosztályba

```
protected bool SetProperty<T>(ref T storage, T value,
    [CallerMemberName] String propertyName = null)
{
    if (object.Equals(storage, value)) return false;
    storage = value;
    this.OnPropertyChanged(propertyName);
    return true;
}

// Használata
private string name;
public string Name
{
    get { return name; }
    set { SetProperty(ref name, value); }
}
```

Adatforrás változás jelzés listák esetén

- `INotifyCollectionChanged` interfész
 - > Új elemek
 - > Mozgatás
 - > Törlés
 - > Változás
 - > Reset
- `ObservableCollection<T>` : INCC

Listás adatkötés

- ItemsControl . ItemsSource propertyra kell kötni
 - > A ListBox, stb. az ItemsControlból származik
- Köthetjük közvetlenül az adatforrást
 - > List<T>, ObservableCollection<T> leszármazottak
 - > IList, IList<object>, IEnumerable, IEnumerable<object> interfész implementálók
- Köthetünk „okosabb” burkoló osztályt:
- Nézetek: CollectionView alaposztály
 - > Aktuális elem nyilvántartása
 - > Adatlista rendezése, szűrése és csoportosítása (!)

IICollection View 1 : IE, INCC

- „Okos” csomagoló gyűjtemény
- GetDefaultView statikus metódussal hozzuk létre
 - > SourceCollection property: a forrás
- Aktuális elem nyilvántartás
 - > CurrentItem / Position
 - > MoveCurrentTo...
- Rendezés
 - > SortDescription lista
 - > Property név és rendezési irány

ICollectionView létrehozás példa

```
public class CustomerViewModel
{
    private ICollectionView _customerView;

    public ICollectionView Customers
    {
        get { return _customerView; }
    }

    public CustomerViewModel()
    {
        IList<Customer> customers = GetCustomers();
        _customerView = CollectionViewSource.GetDefaultView(customers);
    }
}
```

CollectionView aktuális elem példa

```
<ListBox ItemsSource="{Binding Customers}" IsSynchronizedWithCurrentItem="True" />
```

```
IList<Customer> customers = GetCustomers();
ICollectionView _customerView = CollectionViewSource.GetDefaultView(customers);
_customerView.CurrentChanged = CustomerSelectionChanged;

private CustomerSelectionChanged(object sender, EventArgs e)
{
    // React to the changed selection
}
```

ICollectionView 2 : IE, INCC

- Szűrés
 - > Filter: Predicate alapú, elemenként lehet döntení
- Csoportosítás
 - > PropertyGroupDescription lista
 - > Megadható, melyik property legyen a csoportosító szempont
 - > Lekérdezhetők a kialakult csoportok
 - Név + benne lévő elemek
- Késleltetett frissítés (DeferRefresh)

CollectionView szűrés példa

```
IICollectionview _customerView = CollectionViewSource.GetDefaultView(customers);
_customerView.Filter = CustomerFilter

private bool CustomerFilter(object item)
{
    Customer customer = item as Customer;
    return customer.Name.Contains(_filterString);
}
```

```
public string FilterString
{
    get { return _filterString; }
    set
    {
        _filterString = value;
        NotifyPropertyChanged("FilterString");
        _customerView.Refresh();
    }
}
```

CollectionView rendezés példa

```
IICollectionview _customerView = CollectionViewSource.GetDefaultView(customers);
_customerView.SortDescriptions.Add(
    new SortDescription("LastName", ListSortDirection.Ascending));
_customerView.SortDescriptions.Add(
    new SortDescription("FirstName", ListSortDirection.Ascending));
```

```
ListCollectionView _customerView = CollectionViewSource.GetDefaultView(customers);
                                         as ListCollectionView;
_customerView.CustomSort = new CustomerSorter();

public class CustomerSorter : IComparer
{
    public int Compare(object x, object y)
    {
        Customer custX = x as Customer;
        Customer custY = y as Customer;
        return custX.Name.CompareTo(custY.Name);
    }
}
```

CollectionView csoportosítás példa

```
<ListBox ItemsSource="{Binding Customers}">
    <ListBox.GroupStyle>
        <GroupStyle.HeaderTemplate>
            <DataTemplate>
                <TextBlock Text="{Binding Path=Name}" />
            </DataTemplate>
        </GroupStyle.HeaderTemplate>
    </ListBox.GroupStyle>
</ListBox>
```

A csoport Name
propertyje, neve

```
ICollectionView _customerView = CollectionViewSource.GetDefaultView(customers);
_customerView.GroupDescriptions.Add(new PropertyGroupDescription("Country"));
```

Az azonos országba
tartozó elemek kerülnek
egy csoportba

CollectionView csoportosítás példa

```
<ListView.GroupStyle>
    <GroupStyle>
        <GroupStyle.HeaderTemplate>
            <DataTemplate>
                <TextBlock FontWeight="Bold" FontSize="14" Text="{Binding Name}"/>
            </DataTemplate>
        </GroupStyle.HeaderTemplate>
    </GroupStyle>
```

A csoport Name
propertyje, neve

```
</ListVi
CollectionView view = (CollectionView)CollectionViewSource.GetDefaultView(lvUser);
PropertyGroupDescription groupDescription = new PropertyGroupDescription("Sex");
view.GroupDescriptions.Add(groupDescription);
```

Name	Age
Male	
John Doe	42
Sammy Doe	13
Female	
Jane Doe	39

Az azonos neműek
kerülnek egy
csoportba

CollectionViewSource

- Tetszőleges gyűjteményhez készít valamilyen ICollectionView leszármazott példányt
 - > Source: a forrás gyűjtemény
 - > View: az elkészített nézet, ami tud rendezni, szűrni, csoportosítani stb.
- A forrás gyűjtemény legyen minél okosabb
 - > IList: indexelt bejárás, elemszám
 - > INCC: változás értesítés
- XAML-ben is használható, paraméterezhető

Az adatkötés teljesítményproblémái

- A Binding kényelmes de lassú
> - Reflectiont használ
 - Részben emiatt sem használta a Win10 előtt a Microsoft sem a XAML-t a fontosabb termékeiben (a VS-t kivéve)
- A Windows Store-os Office megírásához viszont muszáj volt XAML-t használni ☺
> - De az már natív kód, nem C#!

Compiled Binding

- Ugyanolyan deklaratív szintaxis, mint a „hagyományos” Binding esetén
- Mindent felold fordítási időben
 - > Code-behind kódgenerálás
 - > Nincs reflection
 - > Az elrontott adatkötésekkel fordítási idejű hibát kapunk
 - > Magabiztosabb IntelliSense, jobb debugger élmény

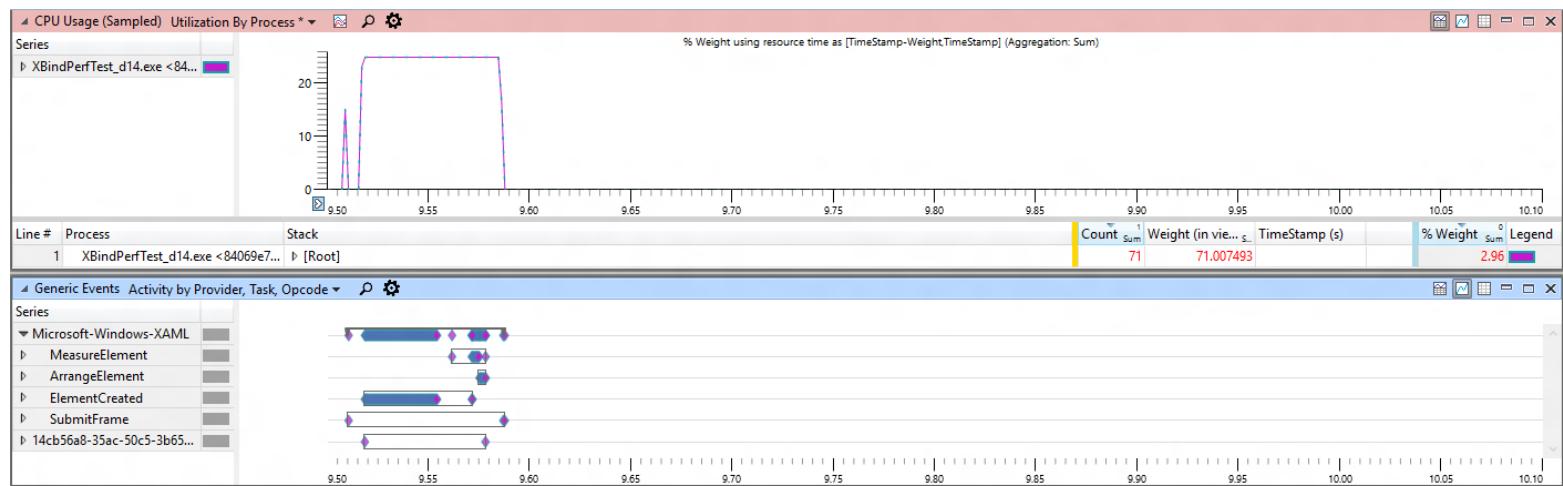
```
<TextBlock Text="{x:Bind User.Name, Mode=OneWay}" />
```

CPU használat összehasonlítása

Classic
Binding

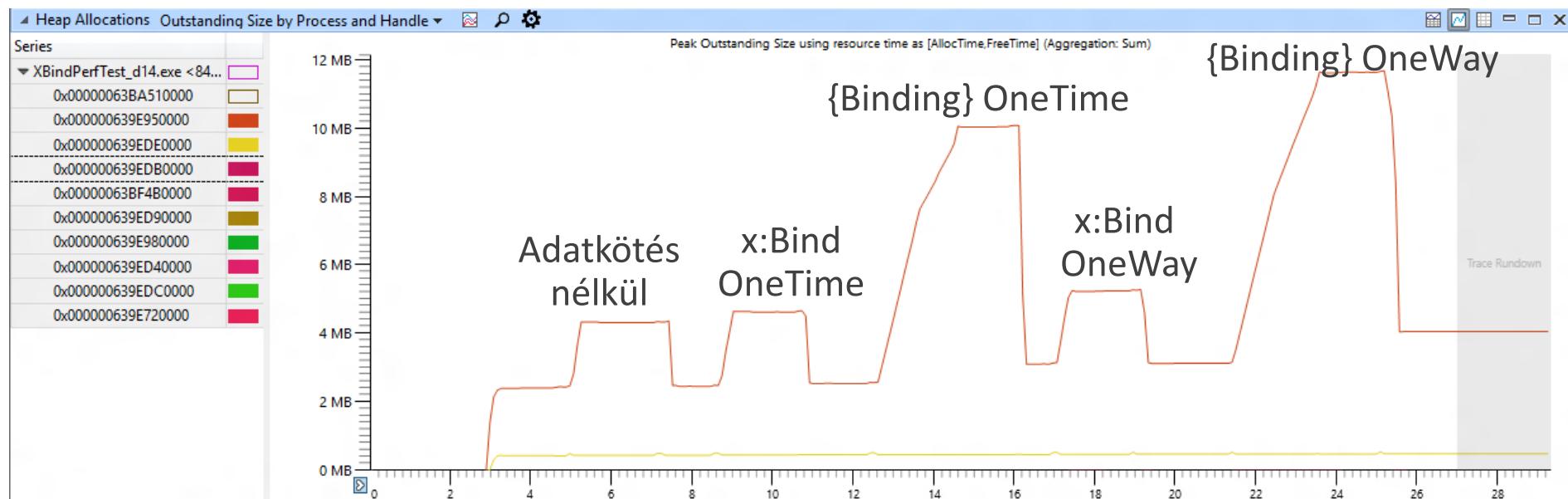


Compiled
Binding



Links: [Windows Performance Analyzer](#), [EventSource](#)

Memóriahasználat összehasonlítása



Mérés: 1600 db <Border> hátterének adatkötése

Binding és x:Bind eltérései

- Az x:Bind erősen típusos
 - > A DataContext **mindig** az aktuális Page/UserControl (vagyis annak a code behind osztálypéldánya)
 - Mintha sima Bindingnál azt írnánk: DataContext = this;
 - Az x:Bind a FrameworkElement.DataContext átállításával nem foglalkozik!
 - Így van fordítás idejű típusinformációja
- Az alapértelmezett Mode nem OneWay, hanem a OneTime!
 - > Túl sok erőforrás olyanra, amit nem is használunk

Tippek a OneTime bindinghoz

- A Loading esemény inicializál minden kötést
 - > Ha aszinkron adatunk van, ez OneTime esetén túl korai...
 - > Bindings.Update() hívással frissíthetőek a OneTime bindingok!
 - Ez akkor is hasznos, ha változásértesítés nélküli forráshoz kell kötnünk (de amúgy tudjuk, mikor frissül)
- Egy jó nagy „Update” hatékonyabb, mint sok kicsi INPC
 - > Ha tudjuk, hogy sok elem fog változni:
 1. Binding.StopTracking();
 2. await AdatokFrissitese();
 3. Bindings.Update();

x:Bind jellemzői

- Adatkonverzió
 - > Ha a cél property string: ToString()
 - > Ha a forrás adat string: a Xaml parser működésével azonos konverzió
 - > Convertereket továbbra is lehet használni
 - De általában nem kell – ld. következő slide
- Változások követése
 - > INotifyPropertyChanged, INotifyCollectionChanged, IObservableVector és persze DependencyProperty
 - > TwoWay Binding esetén a UI property továbbra is DP kell legyen
 - > TextBox továbbra is LostFocusra frissít 😞

Függvényhívás kötése

- `Text="{x:Bind GetTags(VM.Item.Title, VM.Item.Tags)}"`
 - > Konverter / multibinding helyett
 - > Ha bármelyik paraméter változik, újra meghívódik a függvény
- Lehet benne dictionary elérés is
 - > `Text="{x:Bind File.Properties['Artist'].Name}"`
- Lehet benne kasztolás is
 - > `Visibility="{x:Bind ((Visibility)VM.IsStuffVisible)}"`
 - > Csak Visibility, mert ezt égették be a parserbe 😞

Mit *nem* tud az x:Bind

- RelativeSource=Self és ElementName
 - > De mivel eleve a Page a DataContext, nem is hiányzik. (Amit az ElementName-be írnánk, az eleve a DC része.)
- RelativeSource=TemplatedParent
 - > ControlTemplate-be NEM írható x:Bind, de a TemplateBinding használata OK, már optimalizálták
- DataContext dinamikus átállítása
 - > De ez amúgy se volt jó minta, főleg lustaságból használtuk, hogy ne kelljen hosszú PropertyPath-t írni...
- A {Bindingnak} is maradnak azért használati esetei
 - > A sablonok megtárgyalása után még visszatérünk rá!

Események kötése

- Az x:Bind eseménykezelőkhöz is tud kötni
 - > Nincs több kényszeredett DelegateCommand
- Milyen függvényhez köthetünk?
 - > Paraméter nélküli
 - `void Poke() {...}`
 - > Eseményparaméterekhez passzoló
 - `void Poke(object s, RoutedEventArgs e) {...}`
 - > Eseményparaméterek ōseihez passzoló
 - `void Poke(object s, object e) {...}`
 - > Overloadot nem támogat

DependencyProp változásértesítések

- Windows 10 előtt csak „saját” dependency propertyhez tudtunk változásesemény-kezelőt írni
 - > Beépített DP-k változásakor csak körülményesen lehetett megoldani
 - > Mostmár lehet regisztrálni beépített DP-k-hez is saját handlert

```
<Button x:Name="target" Content="{x:Bind Name}" />
```

```
// Register for changes on “target.Content” property
target.RegisterPropertyChangedCallback(
    ContentControl.ContentProperty, ContentChanged);
```

```
// Handler
void ContentChanged(DependencyObject sender, DependencyProperty prop)
{
    Object newContent = (sender.GetValue(prop));
}
```

Kérdések?

Albert István
ialbert@aut.bme.hu

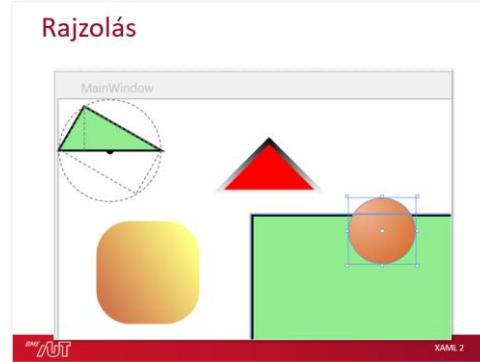
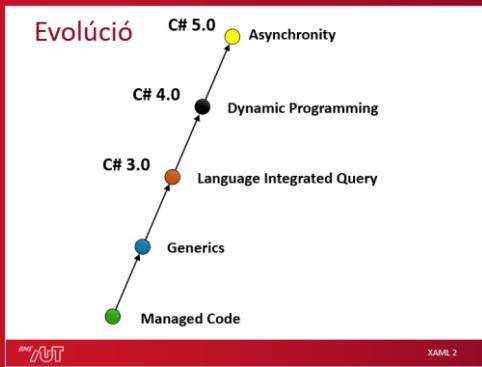
XAML 2

< Albert István >



Automatizálási és
Alkalmazott
Informatikai Tanszék

Tartalom



Fájlként csatolt erőforrások

- Visual Studio-ban Content-ként csatolt fájlok
 - Egyszerű hivatkozás XAML-ből, pl:
`<Image Source="images/logo.png" />`
 - Programozottan, pl:
`var uri = new Uri("ms-appx:///images/logo.png");`
- Lokális fájlrendszerből
 - `<Image Source="ms-appdata:///local/images/logo.png" />`
 - `<Image Source="ms-appdata:///roaming/images/logo.png" />`

Animáció

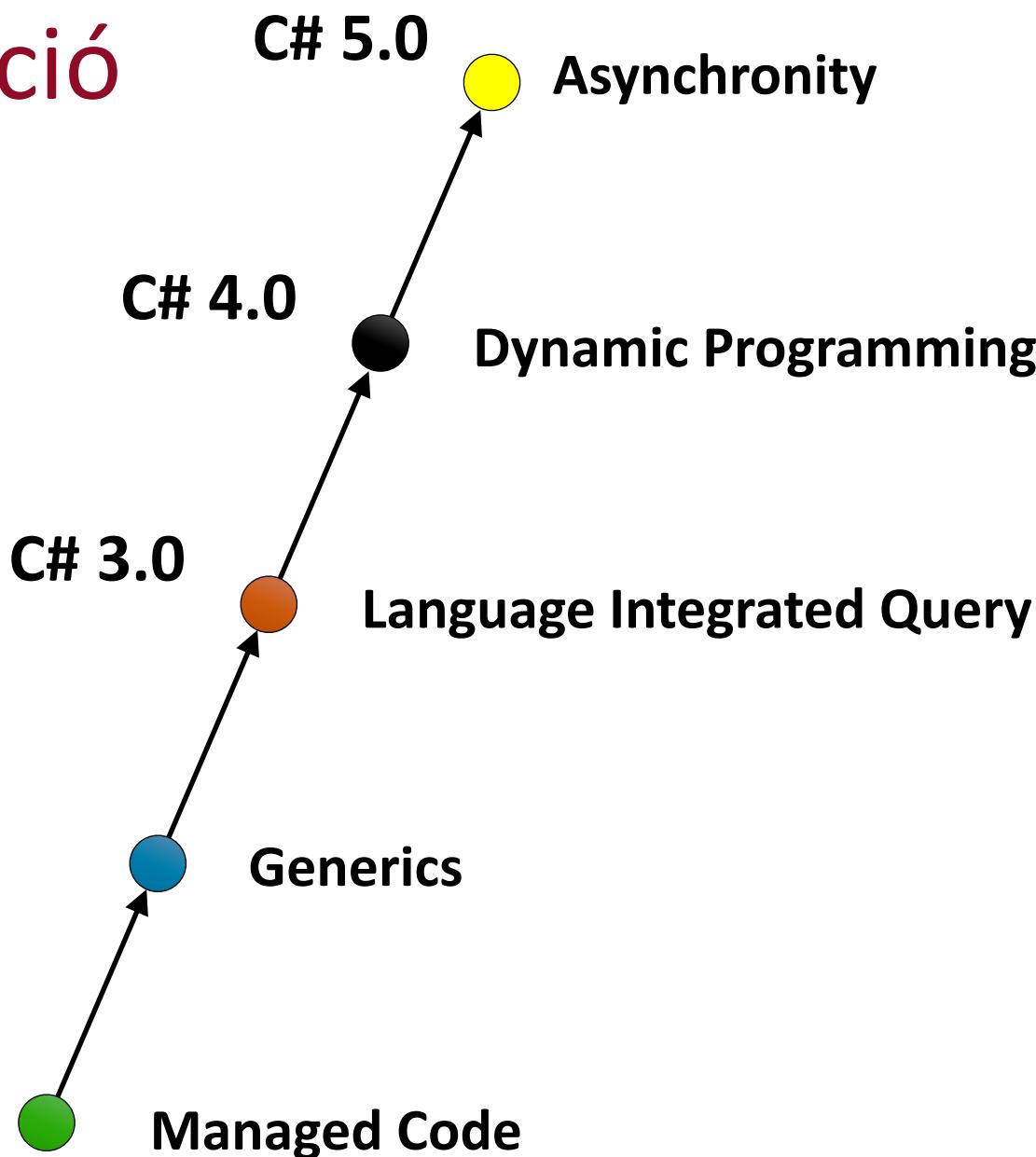


XAML 2

Kérdések?

Albert István
ialbert@aut.bme.hu

Evolúció



Miért fontos az aszinkronitás?

- Egyre több hálózati alkalmazás / művelet – lásd O 365
 - Több késleltetés
 - Több UI ‚responsiveness’ probléma
 - Több skálázási probléma
- Aszinkron programozás
 - Reszponzív, skálázható alkalmazások alapja
 - Csak aszinkron API-k: JavaScript, Silverlight, **WinRT**

Mitől aszinkron?

- Szinkron → Megvárjuk a visszatérési értéket/választ
 - `string DownloadString(...);`
- Aszinkron → Rögtön visszatér, későbbi feldolgozás
 - `void DownloadStringAsync(..., Action<string> callback);`
- Előnyök
 - UI responsiveness: UI szál válaszol a felhasználónak
 - Szerver skálázódás: a szálat más kérések használhatják
- Van összefüggés a párhuzamossággal és többszálúsággal de nem ugyanaz!
 - Például UI szálon is lehet aszinkron művelet

Szinkron vs. aszinkron

```
var data = DownloadData(...);  
ProcessData(data);
```

Szál



DownloadData

ProcessData

```
DownloadDataAsync(... , data => {  
    ProcessData(data);  
});
```

Szál



DownloadDataAsync

ProcessData

Aszinkronitás manuálisan

```
... method1()
{
    GetResultCompleted += callback1;
    GetResultAsync( ... );
    // kilep a metodus
}

... callback1() ←
{
    // háttérszalon fut
    // hibakezeles stb...
    Dispatcher( ..., showResult); // áthívás UI szálra
}

... showResult() ←
{
    Textbox1.Text = ...;
}
```

The diagram illustrates the asynchronous flow of control. It starts with the execution of `method1()`. Inside `method1()`, the `GetResultCompleted` event is registered with `callback1`, and then `GetResultAsync` is called. This triggers the orange arrow pointing to `callback1()`. In `callback1()`, the `Dispatcher` is used to invoke the `showResult` method on the UI thread, which is indicated by the purple arrow pointing to `showResult()`.

Jelenlegi megoldások nehézkesek

- Háttér szálak használata
- Explicit szálkezelés
- Értesítési mechanizmusok
- Versenyhelyzetek
- Kódszétdarabolódás

Taszkok - TAP

- Task osztály: az egységnyi, párhuzamosan vagy aszinkron végrehajtható feladatot tart nyilván
 - > Van saját egyedi azonosítója (Id)
 - > Lekérdezhető a feladat státusza
 - > A dobott kivétel
 - > Lehet várni a befejezésére (egyre vagy többre)
 - > Folytatások vagy hibakezelő felfűzése
- A Task<TResult> olyan feladat, aminek van visszatérési értéke, a Task-ból származik
 - > Típusosan lekérdezhető az eredmény
- Létrehozás new-val vagy TaskFactory osztállyal

Taszk példa

```
// Create a task
var taskA = new Task(() =>
    Console.WriteLine("Hello from taskA."));

// Start the task.
taskA.Start();

// OR USE THE FACTORY METHOD:
// var taskA = Task.Factory.StartNew(() =>
//     Console.WriteLine("Hello from taskA."));

// Output a message from the joining thread.
Console.WriteLine("Hello from the calling thread.");

/* Output:
 * Hello from the calling thread.
 * Hello from taskA.
 */
```

Folytatásos feladatok

- A feladatok láncba fűzhetőek, az egyik lefutása után automatikusan elindulhat a következő
 - > A egyik művelet eredménye lehet a következő bemenő paramétere
- Több előzmény taszkhoz is kapcsolható folytatás
 - > Any és All támogatás
 - > Az összes előzmény eredményt megkapja
- Speciális folytatások is beállíthatók a TaskContinuationOptions paraméterrel
 - > Például hiba vagy abortálás esetén más-más folytatás fussen le

Folytatásos taszkok

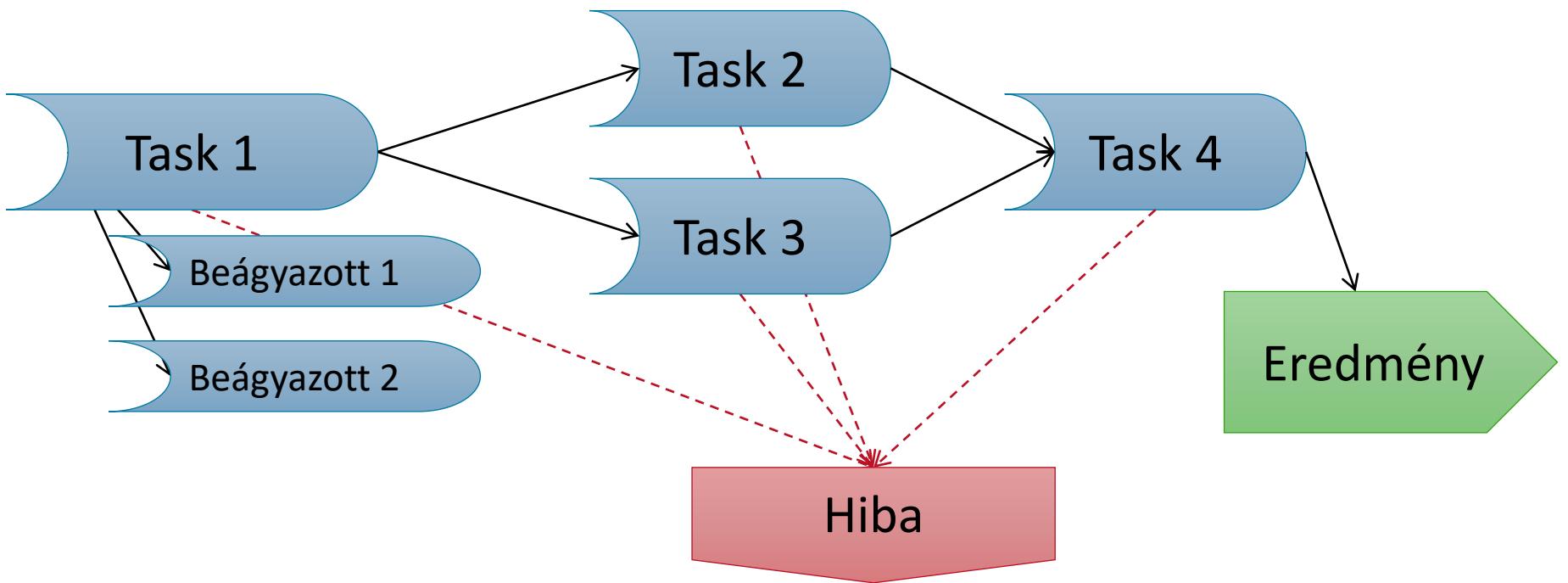
```
try
{
    var firstTask = new Task(
        () => CopyDataIntoTempFolder(path));

    var secondTask = firstTask.ContinueWith((t) =>
        CreateSummaryFile(path));

    firstTask.Start();
    //...
}

catch (AggregateException e)
{
    Console.WriteLine(e.Message);
}
```

Összetett művelet háló



Kivétel kezelés

- A taszkok által dobott kivételek bekerülnek egy AggregateException kivételbe (InnerExceptions)
- Wait hívások, Result property elérése dobja a kivételt
- A Task . Exception propertyn keresztül is lekérdezhető a kivétel
- Ha a Task példány úgy kerül finalizálásra, hogy a fentiek egyike sem hívódik, akkor a teljes folyamat leáll (mint más kivételeknél)

Kivétel kezelés példa

```
var task1 = Task.Factory.StartNew(() =>
{
    throw new MyCustomException("baj van!");
});

try
{
    task1.Wait();
}
catch (AggregateException ae)
{
    // Rethrow anything
    foreach (var e in ae.InnerExceptions)
    {
        if (e is MyCustomException)
            Console.WriteLine(e.Message);
        else
            throw;
    }
}
```

Taszkok megszakítása

- CancellationToken alapú (TPL-től független)
 - > CancellationTokenSource osztály hozza létre a tokent
 - > A Task példányokhoz a token hozzárendelhető
- Kooperatív: a taszon belül kell figyelni a tokent
 - > Polling
 - > Wait...
 - > Callback regisztráció
 - > Több token összeköthető egy tokenbe
- OperationCanceledException kivételt kell dobni
 - > Ebből tudja a TPL, hogy megszakították a taszkot

Művelet megszakítás

```
static void CancelWithThreadPoolMiniSnippet()
{
    //Thread 1:The Requestor - Create the token source.
    CancellationTokenSource cts = new CancellationTokenSource();
    ThreadPool.QueueUserWorkItem(DoSomeWork, cts.Token);

    // Request cancellation
    cts.Cancel();
}

static void DoSomeWork(object obj)
{
    CancellationToken token = (CancellationToken)obj;
    for (int i = 0; i < 100000; i++)
    {
        Thread.SpinWait(500000); // Simulating work.
        if (token.IsCancellationRequested)
            throw new OperationCanceledException(token);
    }
}
```

Aszinkronitás manuálisan

```
... method1()
{
    GetResultCompleted += callback1;
    GetResultAsync( ... );
    // kilep a metodus
}

... callback1()
{
    // háttérszalon fut
    // hibakezeles stb...
    Dispatcher( ..., showResult); // áthívás UI szálra
}

... showResult()
{
    Textbox1.Text = ...;
}
```

Asznkronitás nyelvi szinten

- A fenti kód az új async és await kulcsszavakkal

```
private async void btnDoWork_Click(...)  
{  
    try  
    {  
        int result = 0;  
        await ThreadPool.RunAsync(  
            delegate { result = Compute(); });  
        txtResult.Text = result.ToString();  
    }  
    catch (Exception exc)  
    {  
        txtResult.Text = exc.Message;  
    }  
}
```

Másik példa ThreadPool nélkül

```
double calculatePI()
{
    double pi = 2;
    const int n = 50;
    for (int i = n; i > 0; i--)
        pi = pi * i / (2 * i + 1) + 1;
    return 2* pi;
}
```

1 reference

```
private async void Page_Loaded(object sender, RoutedEventArgs e)
{
    double result = await Task.Run((Func<double>)calculatePI);
    txtResult.Text = result.ToString();
```

A nyelvi megoldás előnyei

- Nem kell manuálisan visszahívni a UI szálba
- Nem kell az állapotot hívásonként feldolgozni, hanem használható szokásos a kivételkezelés
- A program struktúrája olvasható marad, nem kell callbackekre feldarabolni
- Az aszinkron hívások egymás után fűzhetők, jól komponálódnak
- A metódus az awaitek mentén feldarabolódik

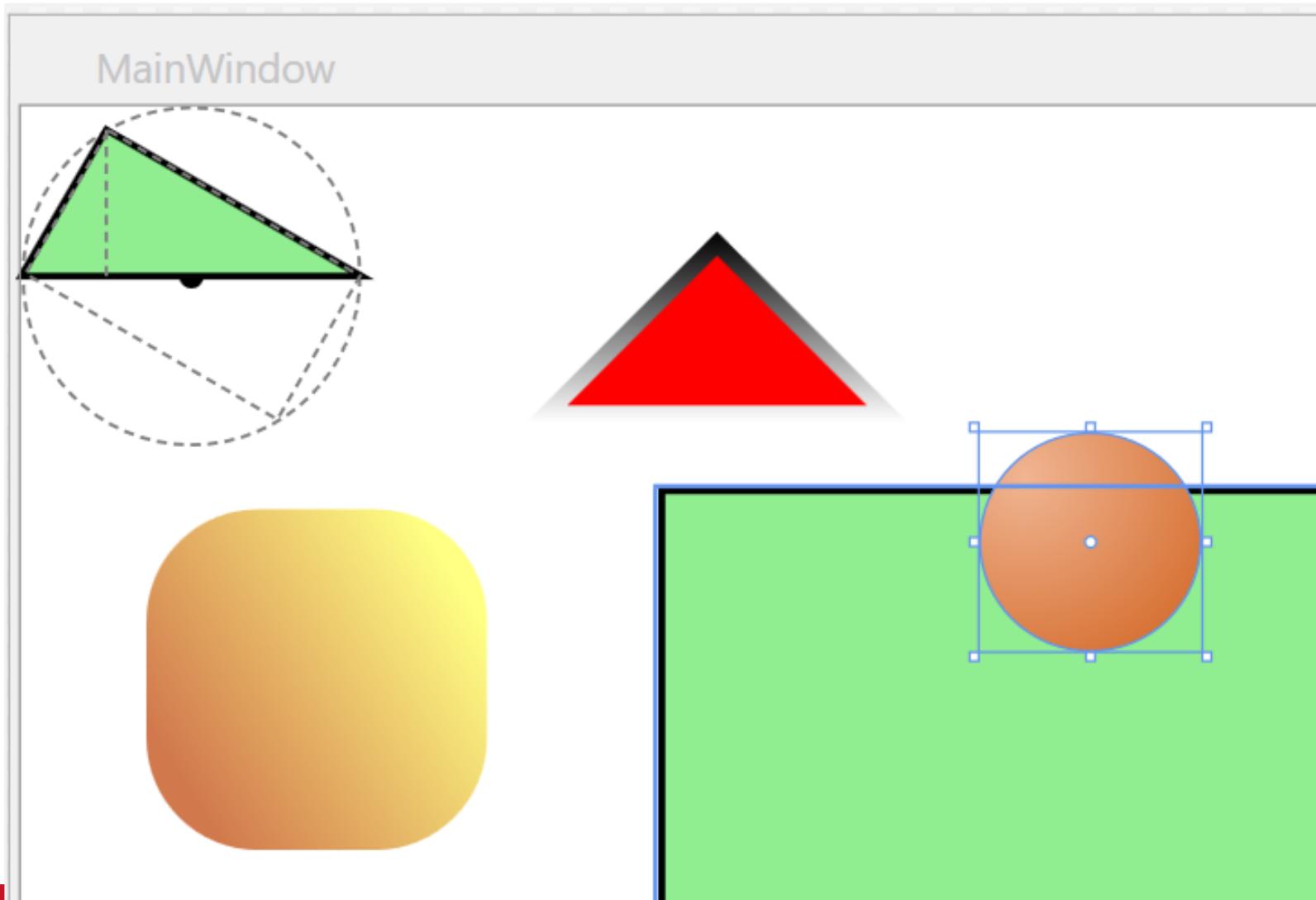
A nyelvi megoldás lépései

- A ‚yield return’-höz hasonlóan itt is egy állapotgép követi a metódus végrehajtását
 - > minden metódushoz egy külön osztály készül
- A metódus az **await**tel jelzett aszinkron hívás helyén felfüggeszti a futását és kilép
 - > A lokális változók és a végrehjtás helye elmentődik
 - > A hívó visszakapja a vezérlést
 - > Amint a művelet befejeződik, a metódus folytatódik
- Bármilyen típus visszaadható, ami megfelel az await mintának:
 - > GetAwaiter metódus => IsCompleted, OnCompleted, GetResult tagokkal rendelkező obj.

Aszinkron nyelvi elemek

- “**async**” módosító kulcsszó
 - > A metódus vagy lambda kifejezés aszinkron módon futhat
 - > A fordító állapotgépet készít hozzá
 - > A metódust a fordító feldarabolja a await hívások mentén
- “**await**” művelet **visszaadhatja** a vezérlést a szálnak
 - > Amikor az aszinkron hívás (task) véget ér, „folytatódik” a metódus végrehajtása
 - > **await** csak **async** metódusokban használható

Rajzolás



Ellipszis, kör

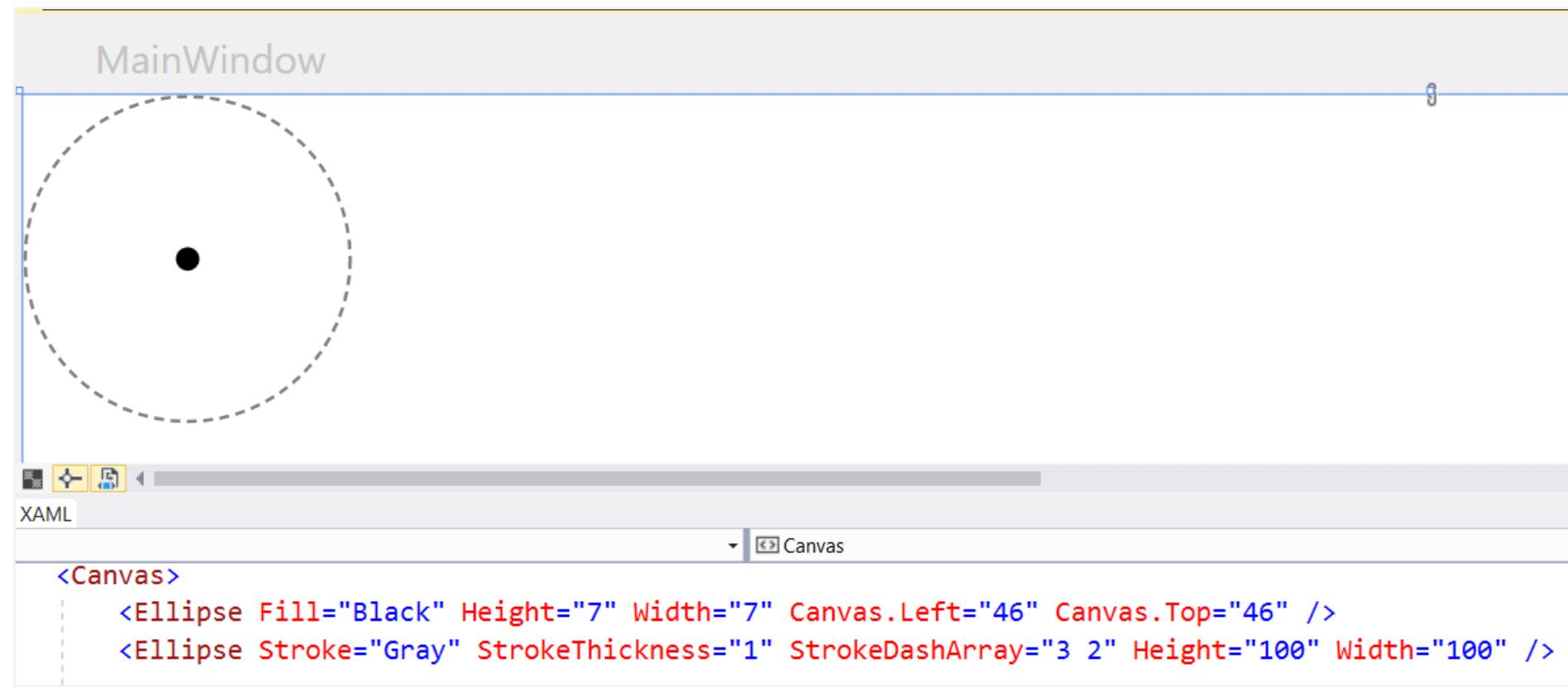
MainWindow



The screenshot shows a Windows application development environment. At the top, there's a title bar with the text "MainWindow". Below it is a main window area containing a single black ellipse. The bottom half of the screen is occupied by a "XAML" editor. The XAML code is as follows:

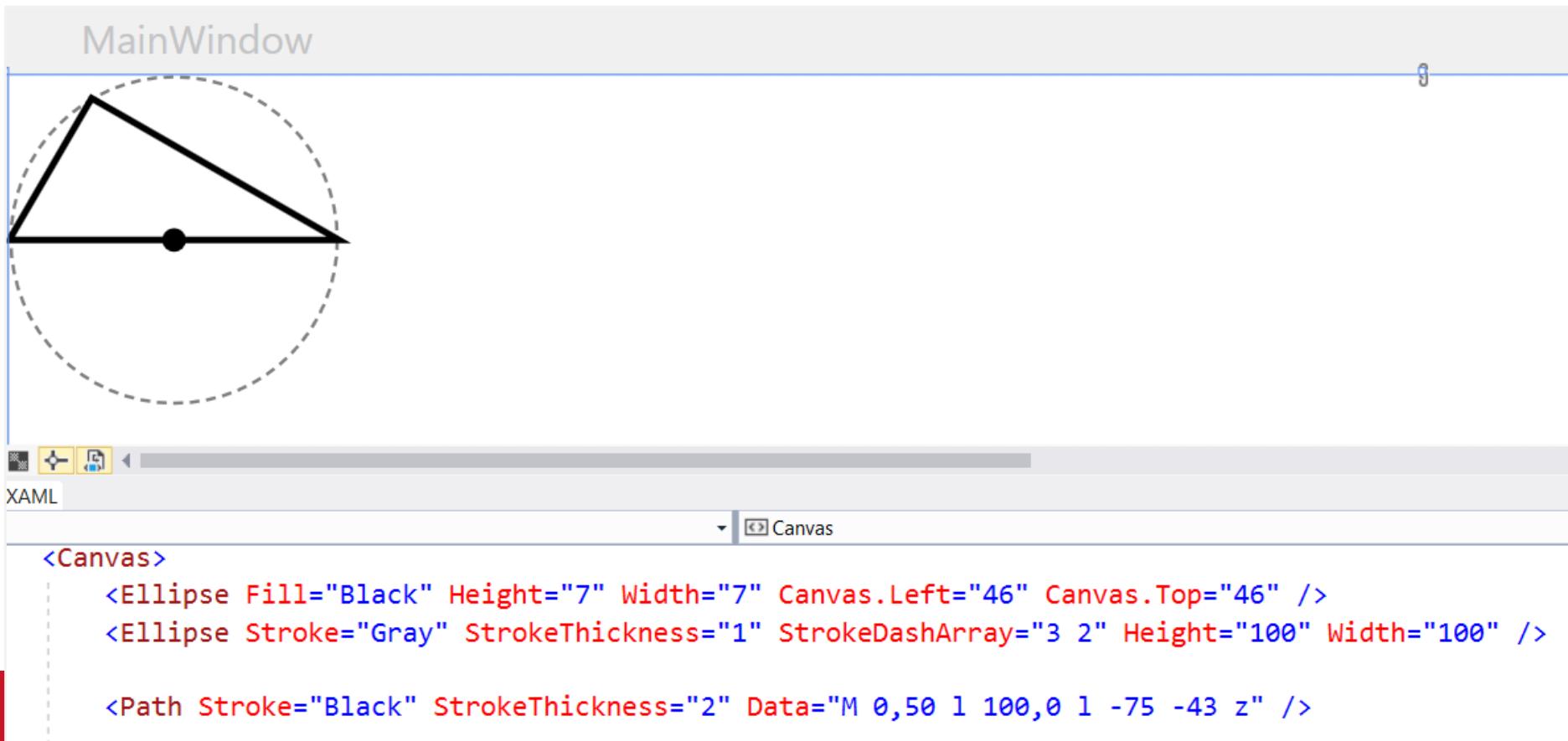
```
<Canvas>
    <Ellipse Fill="Black" Height="7" Width="7" Canvas.Left="46" Canvas.Top="46" />
```

Vonal vastagság, szaggatott vonal



Tetszőleges alakzat

- Path: tetszőleges geometria, több fajta görbe



Téglalap, transzformáció

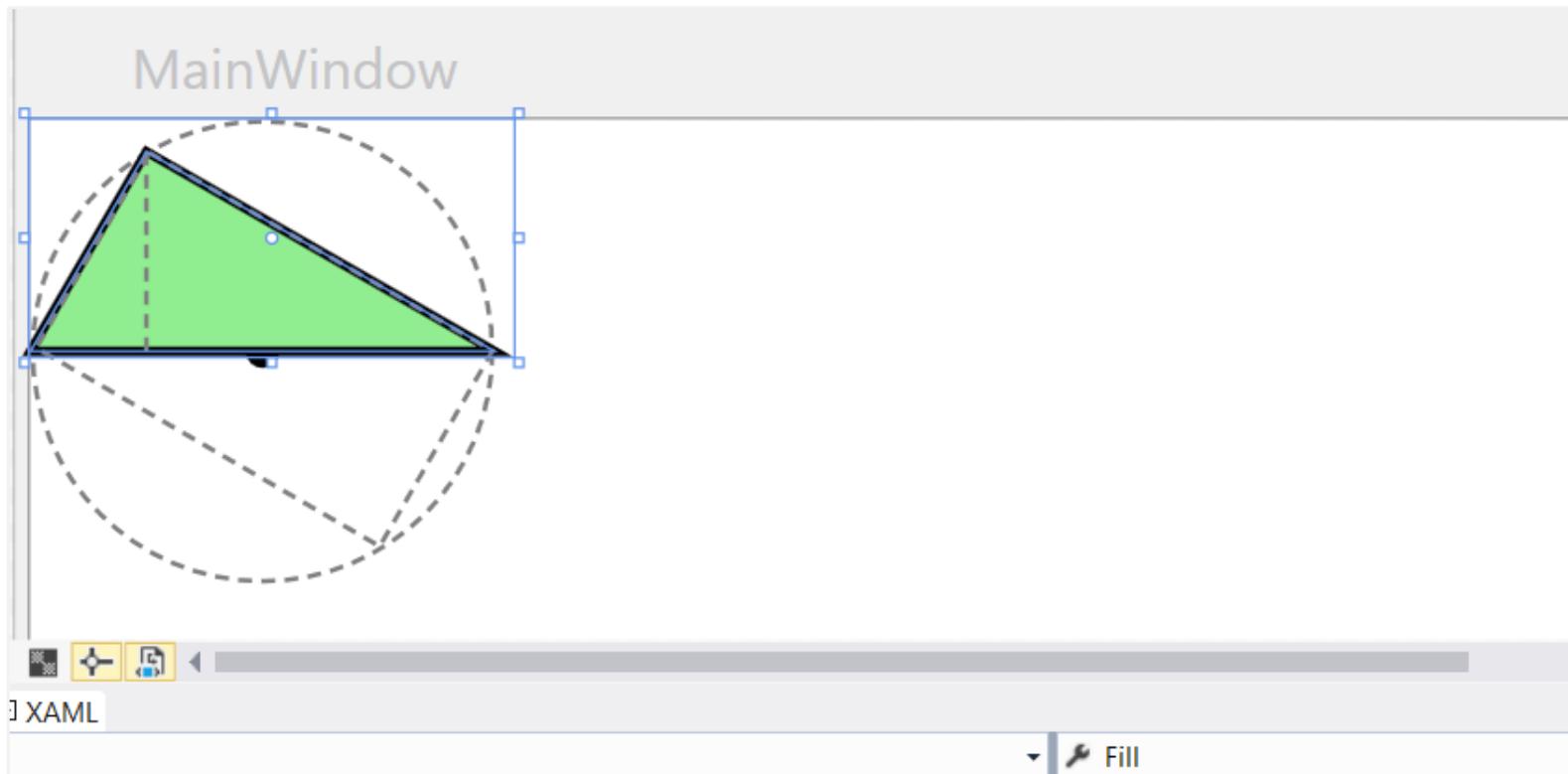
MainWindow

XAML

```
<Canvas>
    <Ellipse Fill="Black" Height="7" Width="7" Canvas.Left="46" Canvas.Top="46" />
    <Ellipse Stroke="Gray" StrokeThickness="1" StrokeDashArray="3 2" Height="100" Width="100" Canvas.Left="46" Canvas.Top="46" />
    <Path Stroke="Black" StrokeThickness="2" Data="M 0,50 1 100,0 1 -75 -43 z" />
    <Line X1="25" Y1="50" X2="25" Y2="7" Stroke="Gray" StrokeThickness="1" StrokeDashArray="3 2" />
    <Rectangle Canvas.Top="25" Width="86" Height="50" Stroke="Gray" StrokeThickness="1">
        <Rectangle.RenderTransform>
            <RotateTransform Angle="30"/>
        </Rectangle.RenderTransform>
    </Rectangle>
```

/UI XAML 2

Zárt alakzatok kitöltése



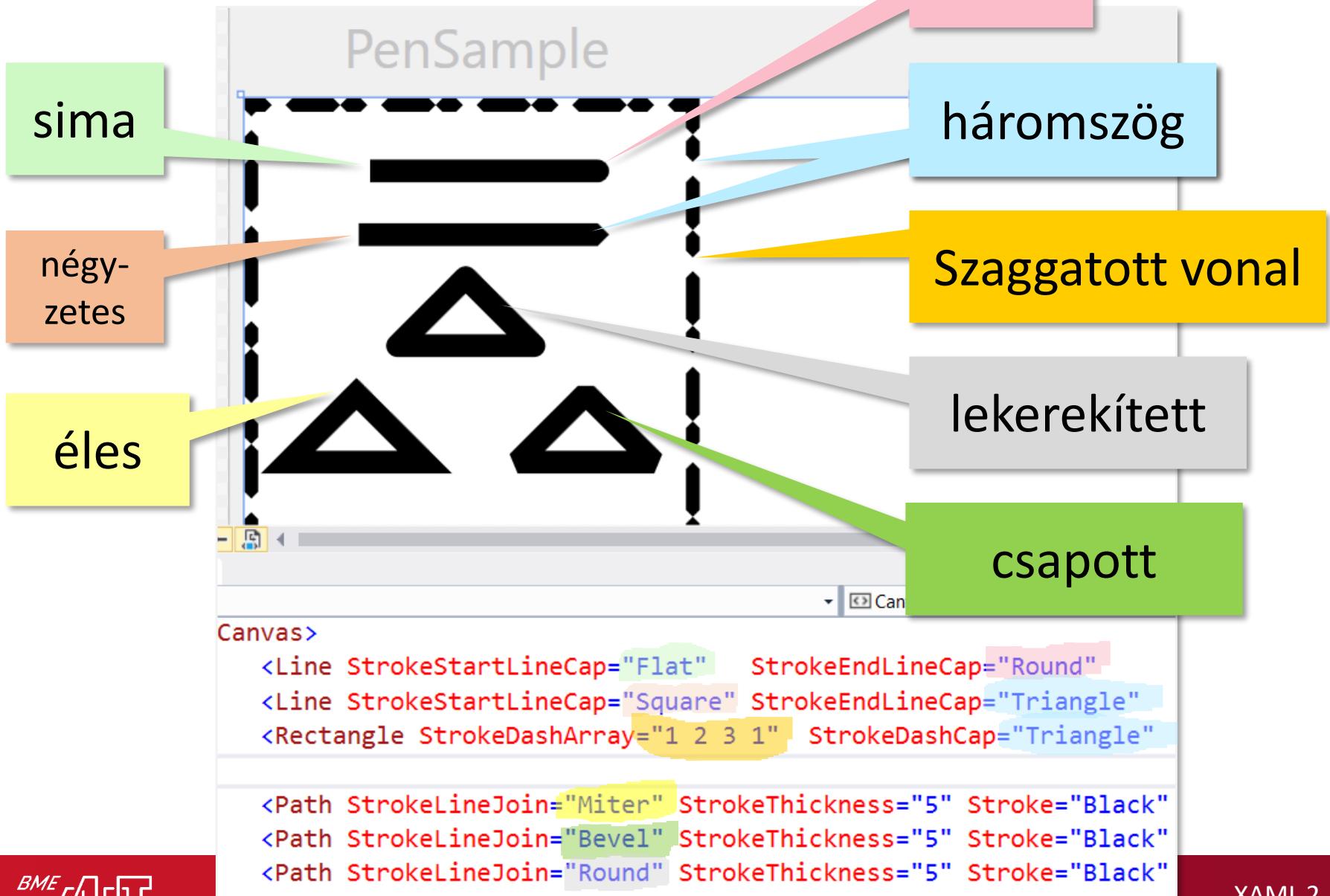
```
<Canvas>
    <Ellipse Fill="Black" Height="7" Width="7" Canvas.Left="46" Canvas.Top="50" />
    <Ellipse Stroke="Gray" StrokeThickness="1" StrokeDashArray="3 2" Canvas.Left="25" Canvas.Top="7" />
    <Path Stroke="Black" Fill="LightGreen" StrokeThickness="2" Data="M 25,50 L 75,50 L 25,70 Z" />
    <Line X1="25" Y1="50" X2="25" Y2="7" Stroke="Gray" StrokeThickness="1" />
    <Rectangle Canvas.Top="25" Width="86" Height="50" Stroke="Gray" />
        <Rectangle.RenderTransform>
            <RotateTransform Angle="30" />
        </Rectangle.RenderTransform>

```

Grafika készítése Shape osztályokkal

- **FE:** eseménykezelés, layout, transzformációk stb.
- **Festés/ecset:** Brush tulajdonság
- **Körvonal/toll:** Pen tulajdonság
 - > A körvonalat alkotó vonalak kinézete befolyásolható
- **Stretch:** a nagyítás módja
- **Rectangle, Line, Polygon, Polyline, Ellipse**
- **Path:** tetszőleges geometriai elem halmaz

Vonal testreszabása

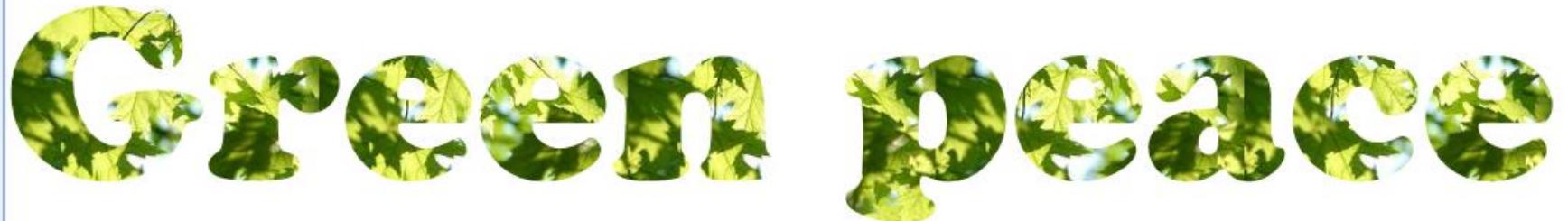


Pen és Brush

- Pen (vagy Stroke): kontúr vonalak formázása
 - > Vonalkitöltés, ami egy Brush
- Brush: alakzat kitöltésének formázása
 - > Átlátszóság
 - > Transformáció
 - > SolidColorBrush, LinearGradientBrush
 - > ImageBrush: kép (tipikusan bitmap)
 - > WebViewBrush: HTML tartalom

Ecset használata szöveghez

Brushes

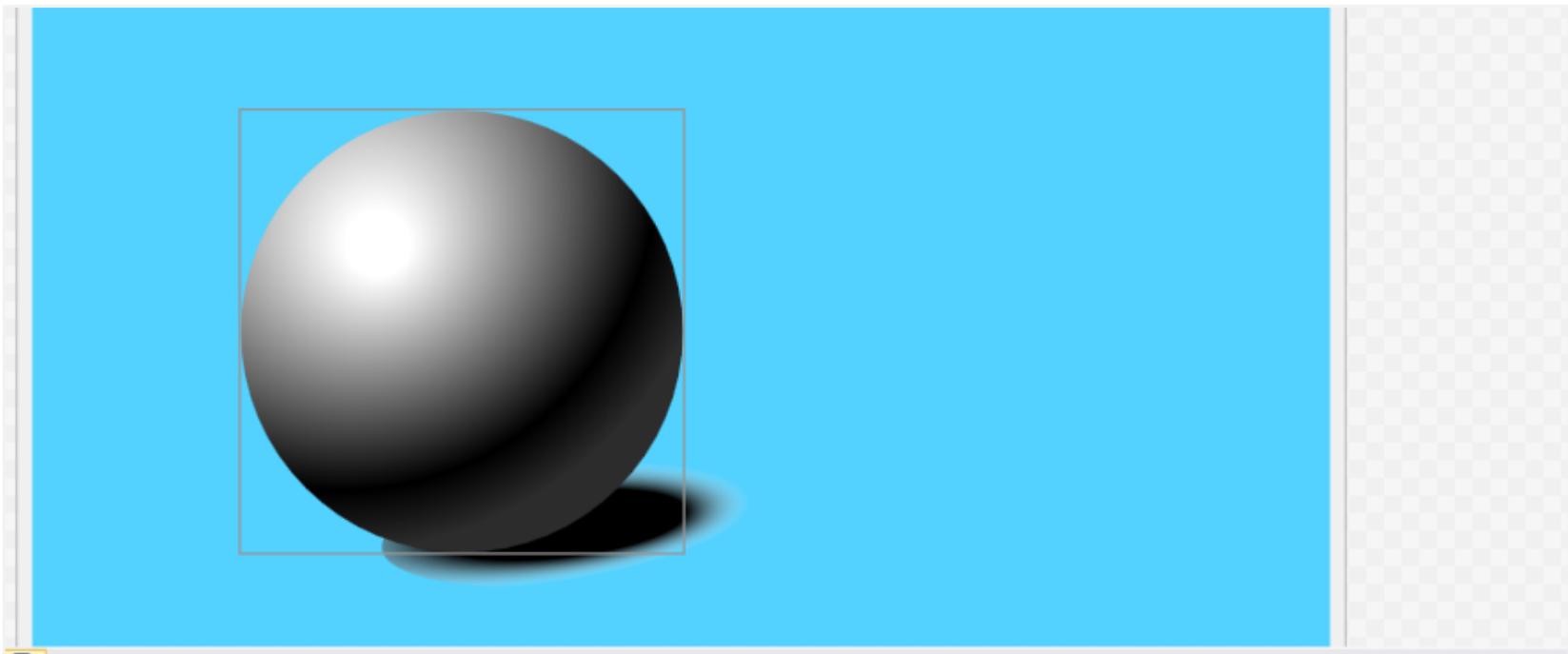
The text "Green peace" is displayed in a large, bold font. The letters are filled with a repeating pattern of green leaves, giving them a textured, organic appearance.

XAML

Canvas

```
<Canvas>
    <TextBlock Text="Green peace" FontSize="45" FontFamily="Cooper Black" >
        <TextBlock.Foreground>
            <ImageBrush ImageSource="Image3.png" TileMode="Tile" Viewport="0,0,0.2,1" />
        </TextBlock.Foreground>
    </TextBlock>
</Canvas>
```

Átmenetes ecsetek



```
<Ellipse Height="100" Width="100" Canvas.Left="47" Canvas.Top="72">
    <Ellipse.Fill>
        <RadialGradientBrush GradientOrigin="0.32,0.32" Center="0.2,0.1"
            RadiusX="0.9" RadiusY="0.9">
            <GradientStop Color="Black" Offset="0.794"/>
            <GradientStop Color="White" Offset="0.084"/>
            <GradientStop Color="#FF2C2C2C" Offset="1"/>
```

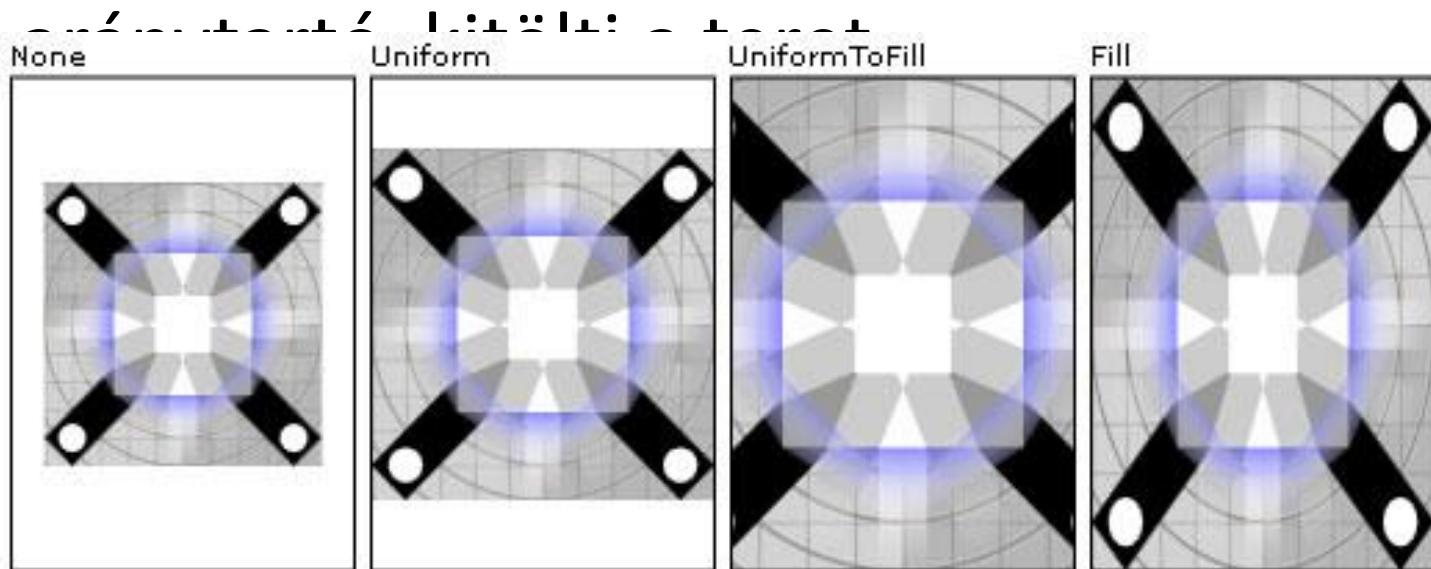
ViewBox : FrameworkElement

- Általános nagyító elem
- Child: a benne lévő tartalom amit felnagyít vagy lekicsinyít a rendelkezésre álló helynek megfelelően
- StretchDirection: az átméretezés iránya
 - > nagyítás / kicsinyítés / minden kettő
- Stretch: az átméretezés módja



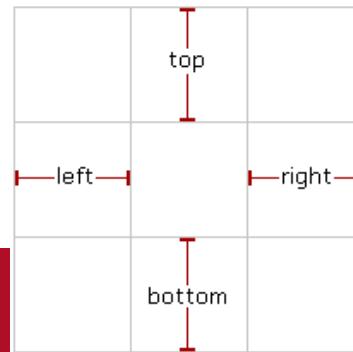
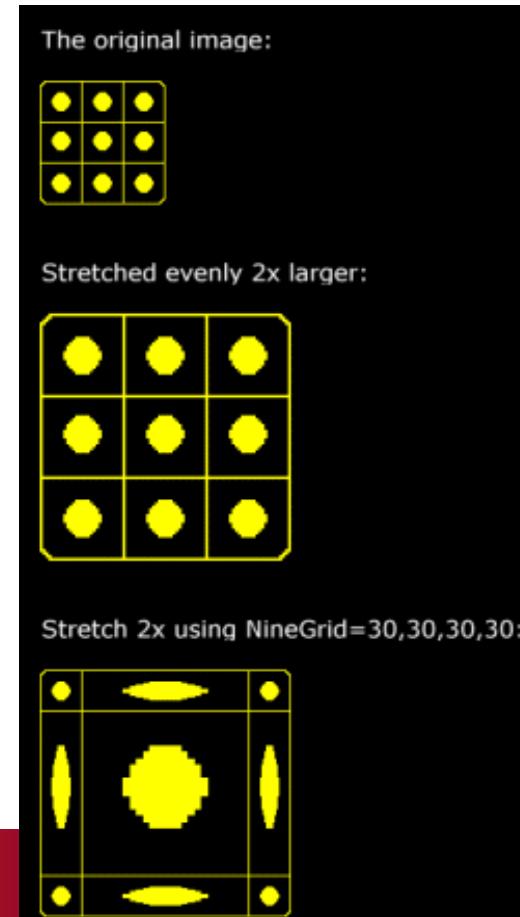
Stretch: átméretezés módja

- None: megmarad az eredeti méret
- Uniform: aránytartó úgy, hogy a teljes tartalom megjelenik, nem kerül levágásra semmi
- UniformToFill: aránytartó úgy, hogy kitölți a teret
- Fill: nem



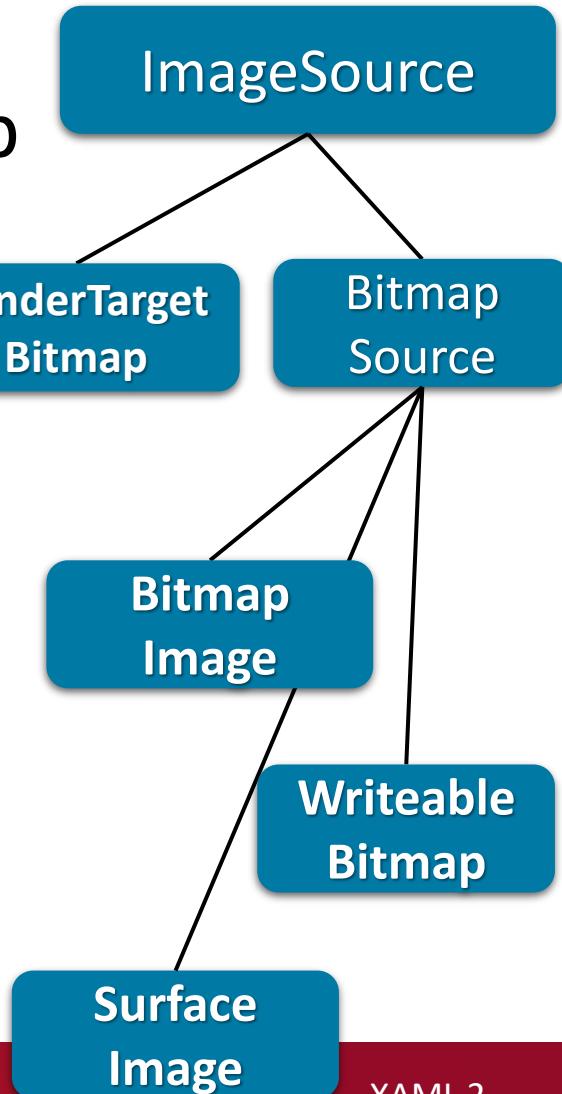
Image

- FrameworkElement-ből származik
 - > Elrendezés, transzformáció, input
- Source (Uri): forrás beállítása
 - > Formátumok: WIC
- ImageFailed / Opened események
 - > Aszinkron képmegnyitás
- Stretch: nagyítás módja
- NineGrid: speciális átméretezés keretes képekhez



Kép kezelés

- ImageSource alaposztály
- BitmapSource: rasztergrafikus kép
 - > Szélesség, magasság
- BitmapImage: kép forrás, pl. fáj
 - > DecodePixelHeight/Width
 - > Betöltés események
- WriteableBitmap
 - > Lassú, közvetlen írás/olvasás
- SurfaceImage: DirectX interop



RenderTargetBitmap

- Tetszőleges XAML tartalom képpé konvertálásához
 - > Dinamikus kép készítés, Share, nyomtatás stb.
 - > Videó, saját DirectX tartalom nem kapható el
- RenderAsync metódus
 - > Forrás XAML, pixel méretek (opcionális)
- GetPixelsAsync
 - > Kimásolja a pixeleket

Metaadatok és kódolás

- Gazdag, kiterjeszthető metaadat támogatás
 - > Exif, tEXt, IFD, IPTC, XMP natív és felügyelt környezetben
- BitmapEncoder és BitmapDecoder osztályok
 - > Beépített formátumok: BMP, GIF, JPEG, PNG, TIFF, WMP
 - > Tetszőleges Codec használható GUID alapján
 - > Több képkocka támogatása: GIF, TIFF
 - > Előnézeti kép, ha van
- WMP: jpeg alternatíva
 - > 30%-kal jobb tömörítés hasonló minőség mellett
 - > Gazdag szabvány, kevesebb erőforrást használ stb.

Fájlként csatolt erőforrások

- Visual Studio-ban Content-ként csatolt fájlok

> Egyszerű hivatkozás XAML-ből, pl:

```
<Image Source="images/logo.png" />
```

> Programozottan, pl:

```
var uri = new Uri("ms-appx:///images/logo.png");
```

- Lokális fájlrendszerből

```
<Image Source=  
      „ms-appdata:///local/images/logo.png” />
```

```
<Image Source=  
      „ms-appdata:///roaming/images/logo.png” />
```

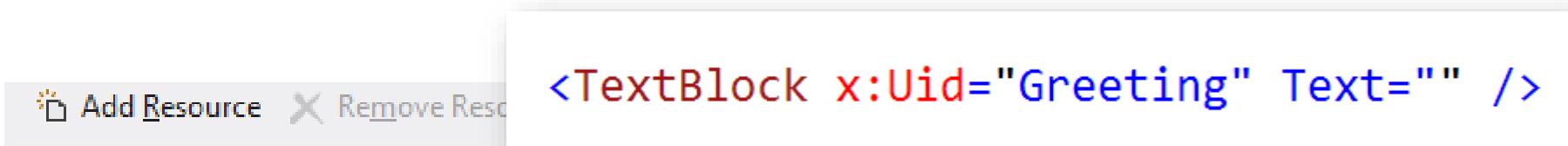
Szöveges erőforrások

- Egyszerűbb modell mint a hagyományos .NET
 - > Csak szöveges erőforrások
 - > Nincs VS kódgenerálás, típusos hivatkozás 😞
 - > Egyetlen erőforrás fájl a kimenet (PRI)
- Több erőforrás fájlt (.resw) lehet létrehozni
 - > A különböző mappában lévőket összefűzi
 - > Alapértelmezett név: “Resources.resw”
 - Ha más a neve, az Uid-ben szerepelnie kell
 - > ResourceLoader osztály fogja betölteni
 - > GetForCurrentView() metódus (DPI aware)

```
var loader = new Windows.ApplicationModel.Resources.ResourceLoader.GetForCurrentView();
var text = loader.GetString("Farewell");
```

Szöveges erőforrások és vezérlők

- A vezérlőknek van x:Uid XAML attribútuma
 - > Ez hivatkozik a szöveges erőforrás kulcsára
 - > Uid: „/resx fájl név/erőforrás név”
- Az erőforrás fájlban a kulcs első eleme az Uid a második a beállítandó tulajdonság



	Name	Value	Comment
▶	Greeting.Text	Hello	A welcome greeting
	Greeting.Width	20	Width of the welcome greeting
	Farewell	Goodbye	A goodbye
*			

Erőforrások lokalizálása

- A sztring és képi erőforrások kultúránként különbözhetnek
- Aktuális nyelv lekérdezése / beállítása:
 - > CultureInfo . DefaultThreadCurrentUICulture
- A kultúra információt az erőforrások elérési útvonalába lehet kódolni
 - > A mappa nevébe (pl: images/hu-hu/poster1.png)
 - > A fájlnévbe (pl: images/poster1.lang-hu-hu.png)
 - > Szöveg erőforrásokhoz is (../hu-hu/strings.resw)
 - > Betöltésnél nem kell odaírni a nyelvet, automatikus
- ResourceManager / Context: további funkciók

XAML erőforrások

- Deklarálás a Resources tegen belül

```
<Page xmlns="..." xmlns:x="..." Title="Simple Window">
    <Page.Resources>
        <SolidColorBrush x:Key="backgroundBrush">
            Yellow</SolidColorBrush>
        <SolidColorBrush x:Key="borderBrush">
            Red</SolidColorBrush>
    </Page.Resources>
```

- Felhasználás a StaticResource-szal

```
<Button Background=
    "{StaticResource backgroundBrush} ... />
```

- Név alapján keres az erőforrások között
- Dekomponálás, erőforrásfájlok összefűzése

```
<ResourceDictionary.MergedDictionaries> ...
```

Animació

The screenshot shows a news application interface with a dark header bar containing the 'CONTOSO NEWS' logo. Below the header, there are two main sections: 'top stories' and 'politics'.

top stories 10 more

- TECH** Cell phones have a huge impact on how long...
- POLITICS** New bills in Washington are bringing up new topics for politi...
- HEALTH** A recently approved drug that may help...
- LOCAL** Helicopter crashes in town near army base
- NATIONAL** Interesting prison reform ideas concerning correction facili...
- LUCERNE SOCIAL** Lucerne Social
a new way to connect

politics 10 more

- POLITICS** Politicians discuss budgets and economic...
- POLITICS** Congress moves forward on new bills
- POLITICS** Sometimes the camera does lie
- POLITICS** Latest developments in the 2012 Election

Animáció elemei



```
<Canvas>
    <Image Height="81" Source="mahjong_seasons.jpg"
        Name="spring" Height="81" Source="sprin
            <Image.RenderTransform>
                <TranslateTransform x:Name="springTranslate" />
            </Image.RenderTransform>
    </Image>
```

```
        <Storyboard x:Key="springAnim">
            <DoubleAnimation From="0" To="100"
                Storyboard.TargetName="springTranslate"
                Storyboard.TargetProperty="X"
            />
        </Storyboard>
```

```
</spri<private void Image_PreviewMouseLeftButtonUp(object sender, MouseEventArgs e)
{
    Storyboard sb = (Storyboard)FindResource("springAnim");
    sb.Begin();
}
```

void Storyboard.Begin() (+ 12 overloads)

Applies the animations associated with this `Storyboard` to their targets and initiates them.

Mit

Hogyan

Mikor

Animáció elemei

1. Mit?

- > Csak Dependency Propertyt!

2. Hogyan?

- > Típusos leíró osztályok
- > Timeline osztály leszármazottak

3. Mikor?

- > Storyboard objektum
- > Manuálisan, programozottan
- > Automatikusan állapot átmenetek esetén

Timeline alaposztály 1.

- Leírja az animáció időbeli lefutását
- **Duration:** az animáció teljes hossza
 - > Pl: „0:0:5”: 5 másodperc
- **AutoReverse:** megfordul-e a végén
- **BeginTime:** az animáció indulási ideje
 - > Összetett animációk esetén
- **RepeatBehavior**
 - > Count: hányszor ismétlődjön
 - ha AutoReverse, akkor az oda-vissza lejátszás 1-nek számít
 - > Duration: összesen mennyi ideig
 - > Forever (statikus): végtelenséggig

Timeline alaposztály 2.

- **FillBehavior**
 - > **HoldEnd**: az animáció végén a property megtartja az utolsó értéket
 - Ilyenkor a property értékének explicit beállítása nem okoz változást
 - Megoldás: Stopra kell állítani és a StoryBoard . Completed eseményére feliratkozni, ott beállítani
 - > **Stop**: a végén a property felveszi az egyébként beállított értéket
 - Lásd DP prioritások



Timeline alaposztály 3.

- **SpeedRatio**: sebesség relatív állítása. Általában a szülő animáción belüli elemek hangolására használják.
- **AllowDependentAnimations**: UI szalon futó animációk engedélyezése
 - > Jellemzően ami a layout újraszámolását igényli, például elemek szélessége
- **Completed** esemény: az animáció végetért

Timeline leszármazottak

- minden animálandó típushoz külön animációs osztály tartozik
 - > DoubleAnimation
 - > ColorAnimation
 - > PointAnimation
 - > + ... AnimationUsingKeyFrames
 - > ObjectAnimationUsingKeyFrames
 - Például Visibility állításhoz
- megadják az animáció pontos adatait

DoubleAnimation

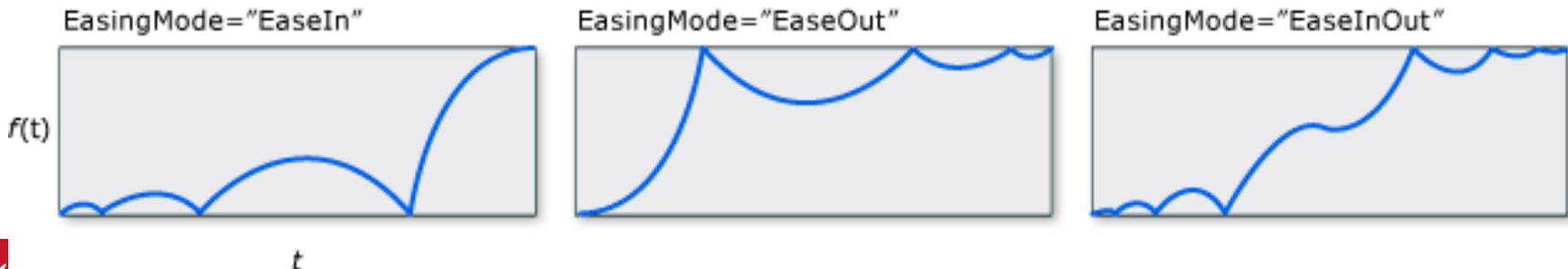
- **From (double?)**
 - > az animáció indulási pontja
 - > Null: az aktuális érték vagy az előző animáció vége
- **To (double?)**
 - > Az animáció végpontja
 - > Null: az aktuális érték vagy az előző animáció vége
- **By (double?)**
 - > Relatív animáció (az aktuális vagy kezdőponthoz képest By-jal megnövelt értékig)
- **EasingFunction:** idő-érték függvény, az animáció lefutásának módját befolyásolja

Példa a DoubleAnimation-re

```
<DoubleAnimation  
    From="1.0"  
    To="0.0"  
    Duration="0:0:3"  
    AutoReverse="True"  
    RepeatBehavior="Forever"  
/>
```

Easing függvények

- Idő-érték függvény megadása ami a lineáris lefutás helyett elasztikus, pattogó, **organikus** stb. hatást kelt
- EaseMode: belépő/kilépő/be- és kilépő is
- Csak beépített függvényeket lehet használni:
 - > BackEase, **BounceEase**, CircleEase, CubicEase, ElasticEase, ExponentialEase, PowerEase, **QuadraticEase**, QuarticEase, QuinticEase, SineEase



Easing példa

No easing



QuadraticEase



Easing példa kód

```
<Storyboard x:Key="springAnim">
    <DoubleAnimation From="0" To="50" Duration="0:0:1"
        Storyboard.TargetName="springTranslate1"
        Storyboard.TargetProperty="X"
    />
    <DoubleAnimation From="0" To="50" Duration="0:0:1"
        Storyboard.TargetName="springTranslate2"
        Storyboard.TargetProperty="X">
        <DoubleAnimation.EasingFunction>
            <QuadraticEase EasingMode="EaseOut" />
        </DoubleAnimation.EasingFunction>
    </DoubleAnimation>
</Storyboard>
```

DoubleAnimationUsingKeyFrames

- Kulcs pontok megadása (KeyFrames)
- Kulcspont:
 - > Típusos, például DoubleKeyFrame stb.
 - > Idő: KeyTime
 - Egy csomagolt TimeSpan
 - > Érték: Value
 - Lehet double, Color, ... a kulcspont típusától függően
 - > A kulcspontok között több féle módon lehet interpolálni

Példa a BounceEase-re

```
<DoubleAnimationUsingKeyFrames  
    Duration="0:0:10"  
    EnableDependentAnimation="True">  
  
    <!-- Folyamatosan 500-ig mozog 3 másodperc alatt az aktuális  
        poziciótól. -->  
    <LinearDoubleKeyFrame Value="500" KeyTime="0:0:3" />  
  
    <!-- Hirtelen 400-ra ugrik a 4. másodpercben. -->  
    <DiscreteDoubleKeyFrame Value="400" KeyTime="0:0:4" />  
  
    <!-- 6 másodperc alatt visszakerül 0-ra. -->  
    <SplineDoubleKeyFrame  
        KeySpline="0.6,0.0 0.9,0.00" Value="0" KeyTime="0:0:6" />  
  
</DoubleAnimationUsingKeyFrames>
```

Kulcs pont interpolációk

- DiscreteDoubleKeyFrame
 - > Nincs interpoláció, ugrik a pontok között
- LinearDoubleKeyFrame
 - > Időben lineáris interpoláció
- EasingDoubleKeyFrame
 - > Easing fügvénnyel megadott interpoláció
- SplineDoubleKeyFrame
 - > Spline kontrol pontok megadásával

Animáció Storyboarddal

- **Animáció:** tetszőleges DependencyProperty értékének interpolációja adott értékek között
- A Storyboardon belül adhatók meg az egyes lefutások, típuspecifikus osztályokkal
 - > Például: DoubleAnimation stb.
- A Storyboard csatolt tulajdonságai választják ki az animálandó tulajdonságot és objektumot
 - > TargetName, TargetProperty
- Vezérlés: Storyboard objektumok segítségével
 - > Kódból vagy XAML-ből
 - A Storyboard is TimeLine leszármazott
 - From, To/By, Duration, Repeat, Reverse, ...
 - > Begin, Pause, Seek, GetCurrentTime, ...

Animáció példa

```
<Window.Resources>
    <Storyboard x:Key="bounceAnim">
        <DoubleAnimation From="-20" Duration="0:0:0:0.1" Storyboard.TargetName="shadowGradient" Storyboard.TargetProperty="Offset" EasingFunction=<BounceEase Bounces="6" Bounciness="1.5" />>
        <DoubleAnimation From="10" Duration="0:0:02" Storyboard.TargetName="shadowGradient" Storyboard.TargetProperty="Offset" EasingFunction=<BounceEase Bounces="6" Bounciness="1.5" />>
        <DoubleAnimation From="0.2" Duration="0:0:0:0.1" Storyboard.TargetName="shadowGradient" Storyboard.TargetProperty="Offset" EasingFunction=<BounceEase Bounces="6" Bounciness="1.5" />>
        <DoubleAnimation From="0.1" Duration="0:0:02" Storyboard.TargetName="shadowGradient" Storyboard.TargetProperty="Offset" EasingFunction=<BounceEase Bounces="6" Bounciness="1.5" />>
    </Storyboard>
</Window.Resources>
<Canvas Background="#FF53D9">
    <Ellipse Height="28" Width="28" Canvas.Left="151" Canvas.Top="72" Name="ball" Fill="Yellow" Stroke="Black" StrokeThickness="3" Data="M 151 72 A 28 28 0 0 1 123 100 Z" />
    <Ellipse Height="100" Width="100" Canvas.Left="151" Canvas.Top="151" Name="shadow" Fill="Black" Opacity="0.2" Stroke="Black" StrokeThickness="3" Data="M 151 151 A 100 100 0 0 1 151 251 Z" />
    <Path StrokeThickness="3" Canvas.Left="58.205" Canvas.Top="88.884" Data="M 58.205 88.884 A 28 28 0 0 1 123 100 Z" />
    <Path StrokeThickness="3" Canvas.Left="113" Canvas.Top="84" Data="M 113 84 A 100 100 0 0 1 151 251 Z" />
</Canvas>
```

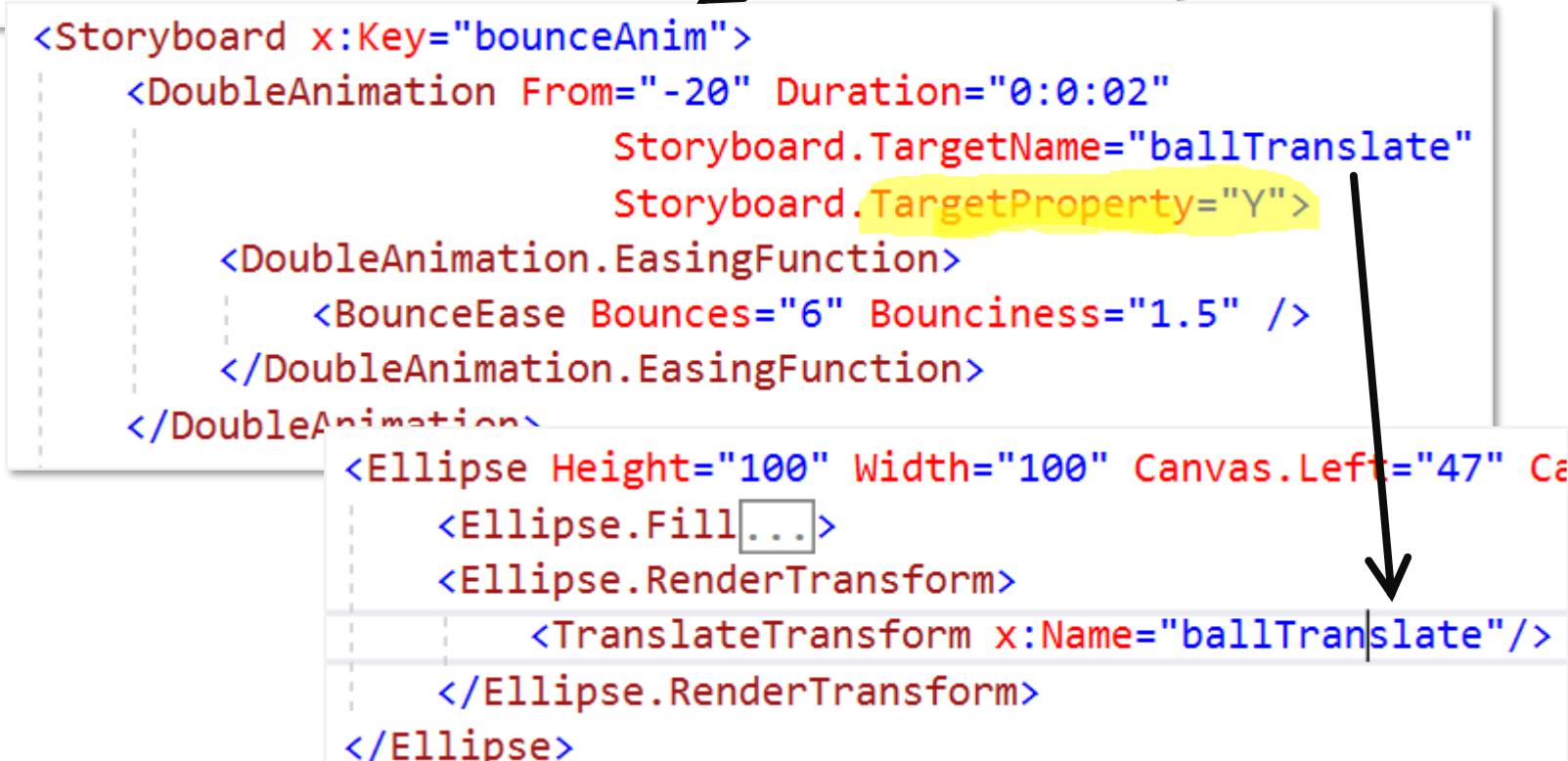


Storyboard példa

```
private void ball_MouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    Storyboard sb = (Storyboard)FindResource("bounceAnim");
    sb.Begin();
}

<Storyboard x:Key="bounceAnim">
    <DoubleAnimation From="-20" Duration="0:0:02"
        Storyboard.TargetName="ballTranslate"
        Storyboard.TargetProperty="Y">
        <DoubleAnimation.EasingFunction>
            <BounceEase Bounces="6" Bounciness="1.5" />
        </DoubleAnimation.EasingFunction>
    </DoubleAnimation>
    <Ellipse Height="100" Width="100" Canvas.Left="47" Canvas.Top="150">
        <Ellipse.Fill>...</Ellipse.Fill>
        <Ellipse.RenderTransform>
            <TranslateTransform x:Name="ballTranslate"/>
        </Ellipse.RenderTransform>
    </Ellipse>

```



Beépített animációk (ThemeAnimations)

- Előre beépített animációk
 - > Timeline-ból származnak, testreszabhatók
 - > Jó default értékekkel vannak paraméterezve
 - > Storyboardon belül használhatók
 - > Nem automatikusak, manuálisan kell őket elindítani
- FadeInThemeAnimation: megjelenéshez
- FadeOutThemeAnimation: eltünéshez
- RepositionThemeAnimation: mozgatáshoz
- Drag And Drop, Swipe, Pop, Split

Példa a FadeOutThemeAnimation-ra

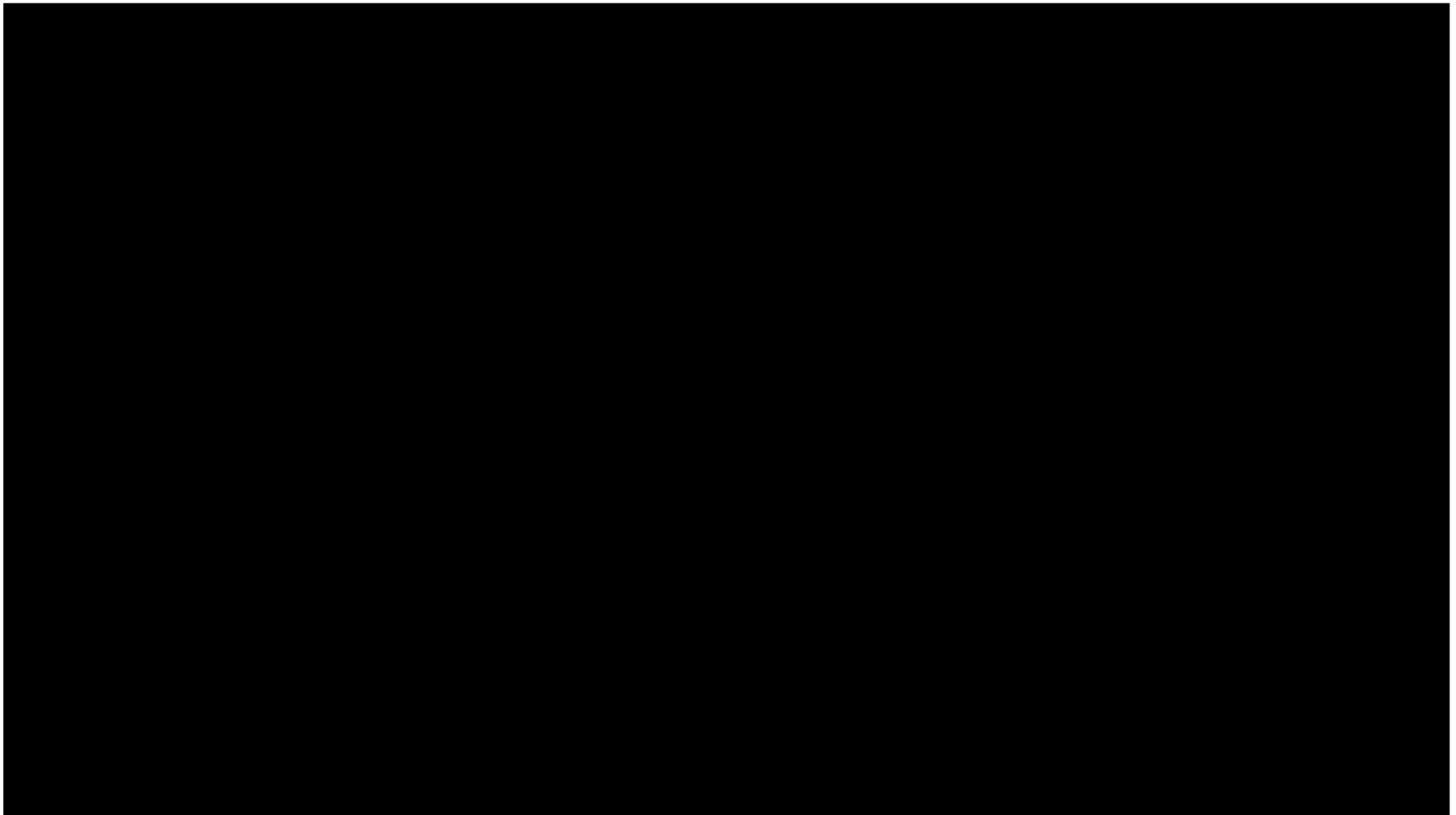
```
<StackPanel>
    <StackPanel.Resources>
        <Storyboard x:Name="EnterStoryboard">
            <FadeOutThemeAnimation
                Storyboard.TargetName="myRectangle" />
        </Storyboard>
        <Storyboard x:Name="ExitStoryboard">
            <FadeInThemeAnimation
                Storyboard.TargetName="myRectangle" />
        </Storyboard>
    </StackPanel.Resources>

    <Rectangle x:Name="myRectangle"
        PointerEntered="Rectangle_PointerEntered"
        PointerExited="Rectangle_PointerExited"
        Fill="Blue" Width="200" Height="300" />
</StackPanel>
```

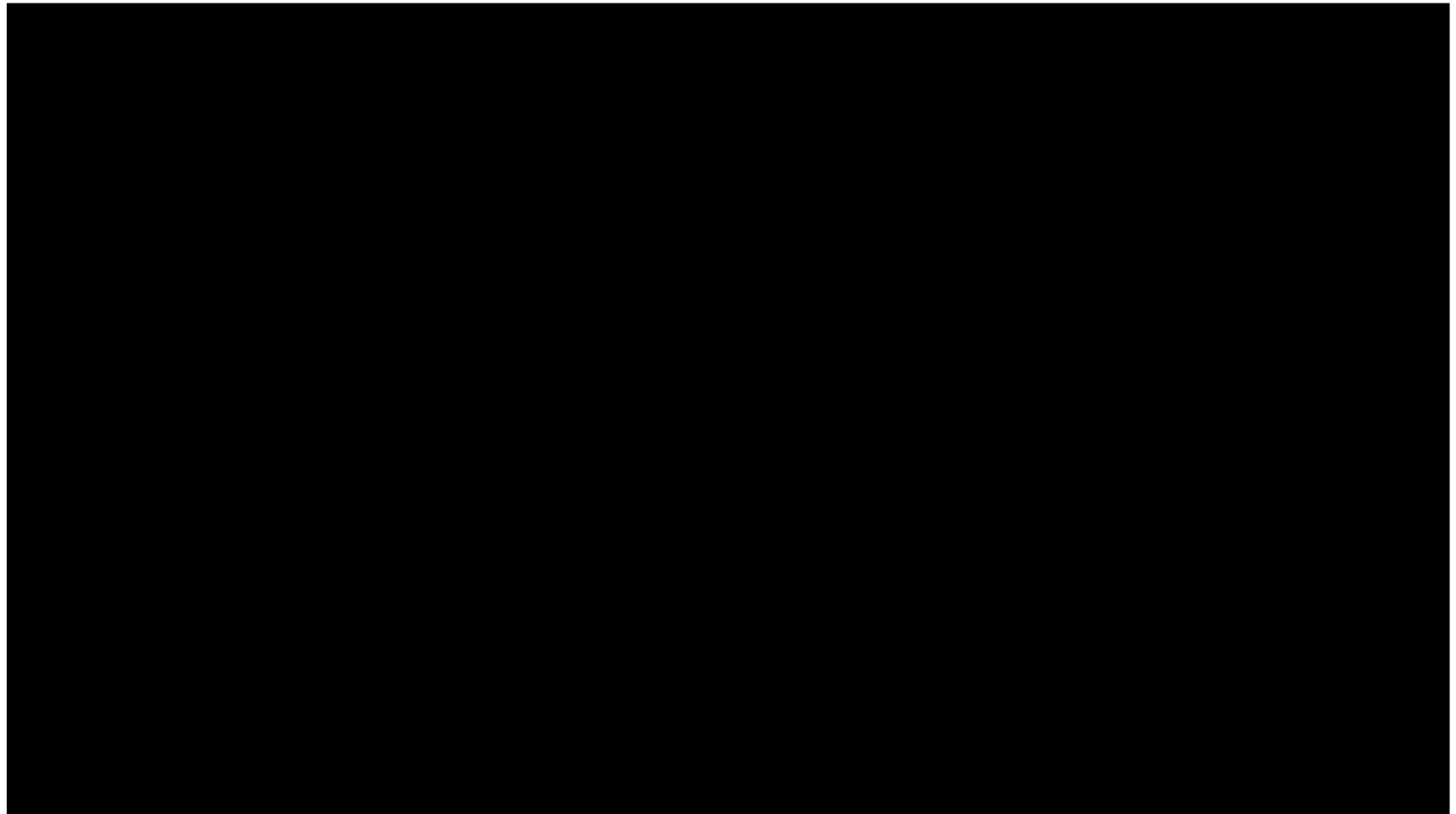
Tranzíciók

- Automatikus animációk
 - > Állapotváltozáskor, átmozgatáskor stb.
- Egyesével
 - > UIElement . Transitions
- Összes gyerekelemre
 - > Panel . ChildrenTransitions
- Korlátosztottan, de paraméterezhetők

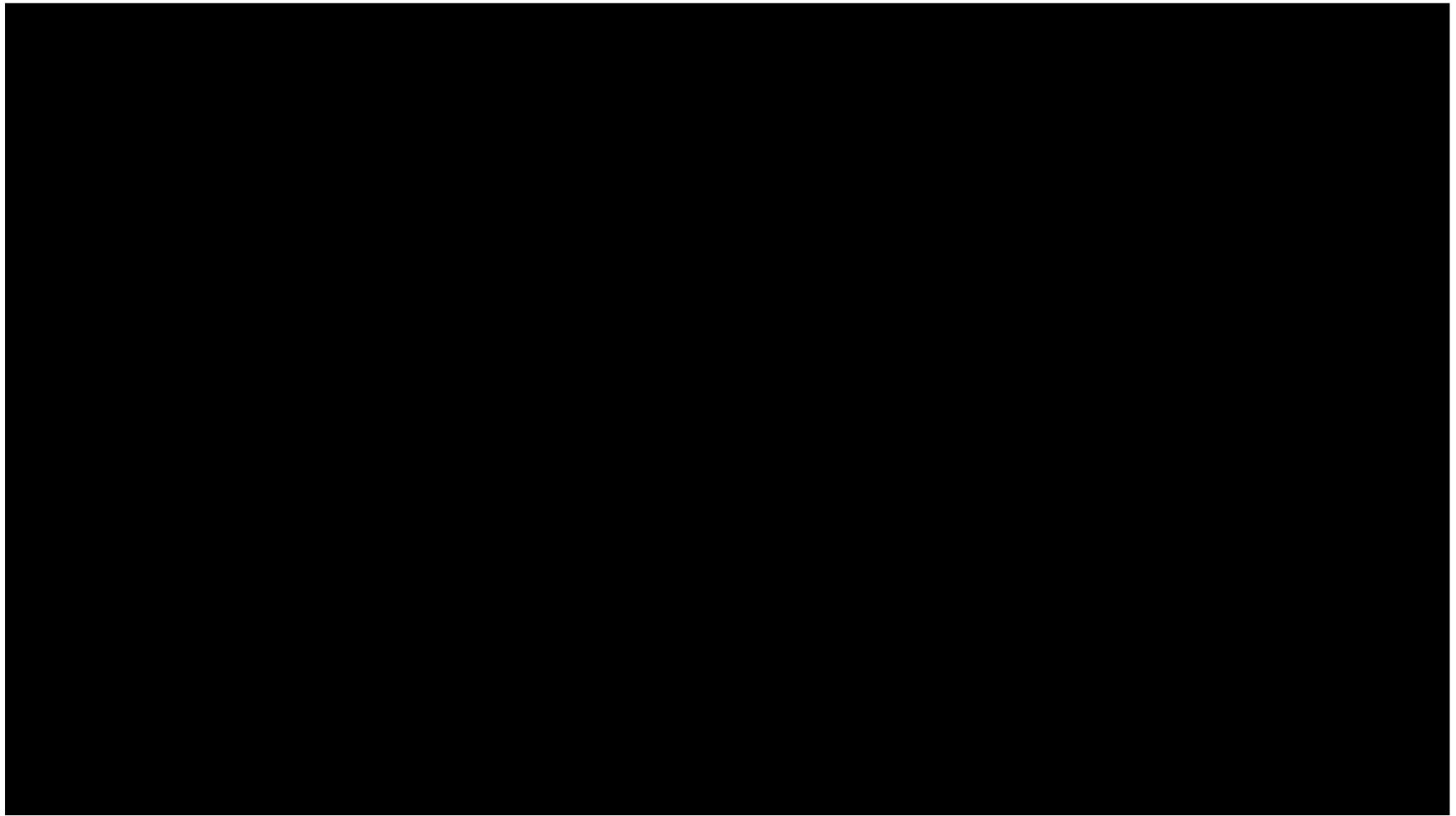
Content



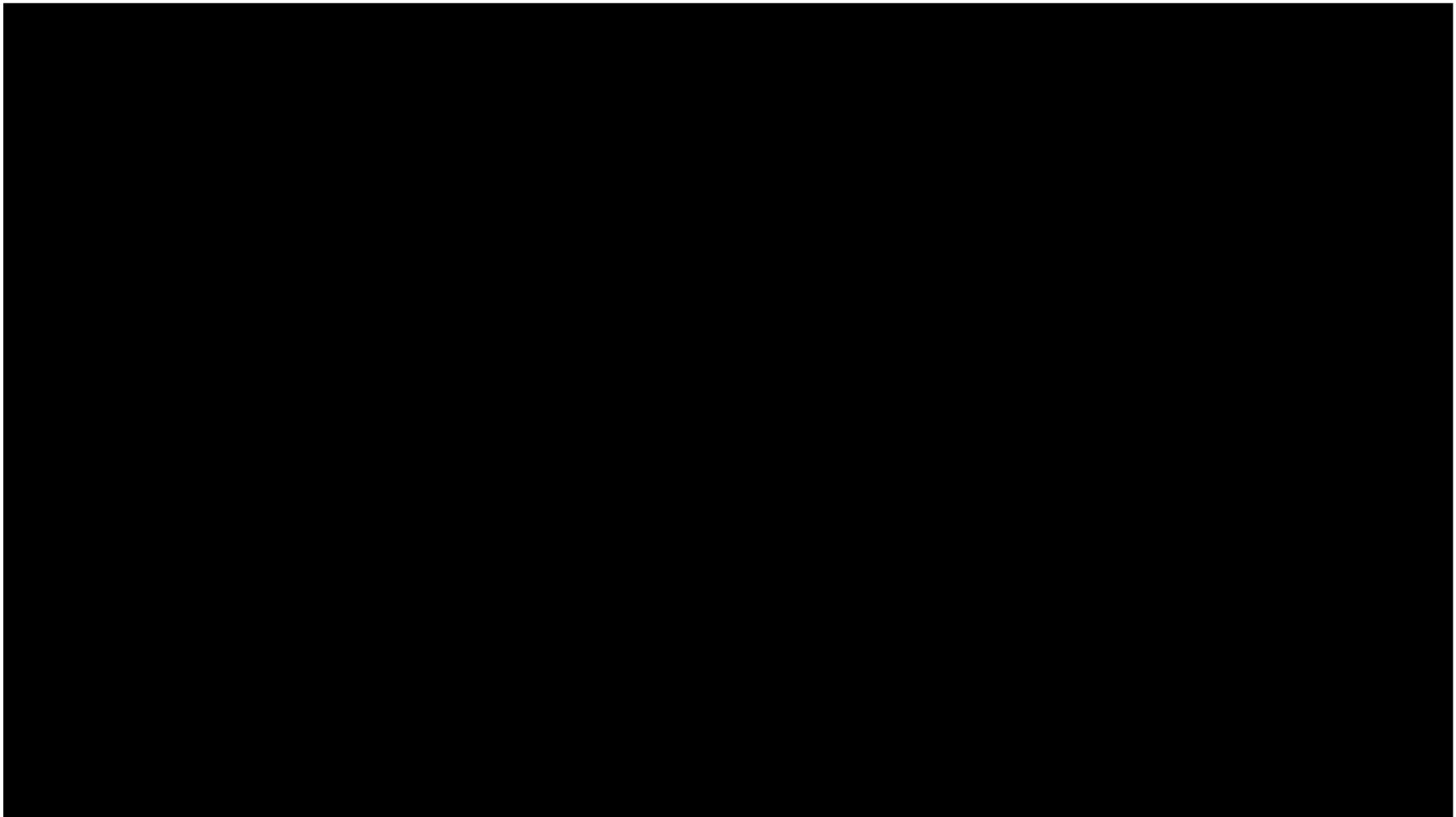
Entrance



AddDelete



Reposition



Beépített tranzíciók

- EntranceThemeTransition
 - > Megjelenik az elem
- ContentThemeTransition
 - > A vezérlő tartalma megváltozik
- AddDeleteThemeTransition
 - > Új gyerek elem kerül egy panelre vagy törlődik
- RepositionThemeTransition
 - > Elem pozíciója megváltozik
- EdgeUI/PaneThemeTransition
 - > Appbar / beállítások
- Popup: tooltip
- Reorder: Drag&Drop művelet

Példa a ContentThemeTransition-re

- A tartalom cserélődése automatikusan indítja az animációt

```
<ContentControl  
    x:Name="ContentHost"  
    PointerPressed="ContentHost_PointerPressed">  
    <ContentControl.ContentTransitions>  
        <TransitionCollection>  
            <ContentThemeTransition />  
        </TransitionCollection>  
    </ContentControl.ContentTransitions>  
  
    <Rectangle Height="200" Width="200" Fill="Orange" />  
</ContentControl>
```

Animation metrics

- Beépített animációk paraméterei kérhetők el
- Ha saját animációk így lesznek konzisztensek a rendszer animációival
- AnimationDescription
 - > Effect: Expand, Reveal, ShowPopup, SwipeSelect,
...
 - > EffectTarget: Primary, Content, Incoming, Tapped,
...
- Animations gyűjtemény
- Időparaméterek

Animációk teljesítmény szempontból

- Independent animáció
 - > Csak rajzoláshoz/megjelenéshez kapcsolódik
 - > Rajzoló szálra fut, max FPS
 - > Például:
 - RenderTransform, Projection
 - Canvas.Left, Top, ...
 - Opacity, Color
- Dependent animáció
 - > UI szálra fut, lassú lehet
 - > Például:
 - Width, Height – elrendezésen változtat!

XAML 3

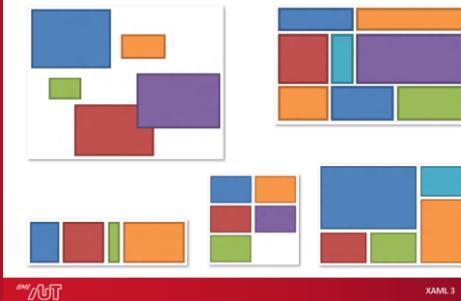
< Albert István >



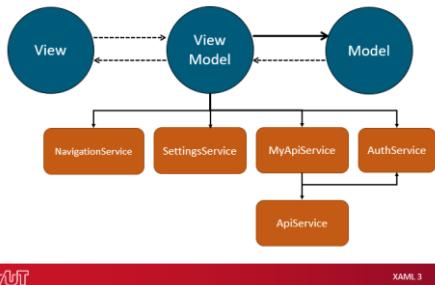
Automatizálási és
Alkalmazott
Informatikai Tanszék

Tartalom

Elrendezés



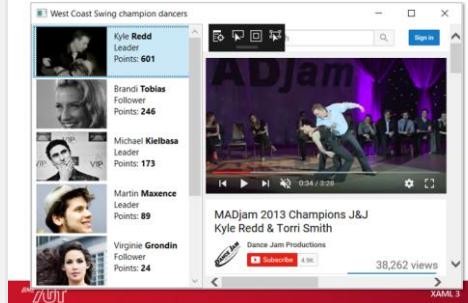
MVVM minta XAML technológiákhoz



Függőségek kezelése

- Az osztályok nagyon gyakran függenek más osztályuktól/szolgáltatásuktól
 - Ez megkötésekkel jár
 - > Függőségek lecserélése, frissítése az osztály módosítását vonja maga után
 - > Fordítás időben elérhetőknek kell lennie a függőségeknek
 - > Nehéz tesztelni az osztályt, mockolni a függőségeit
 - > Az osztály felelőssége a függőségeinek példányosítása/konfigurálása

Sablonok



Adaptivitás

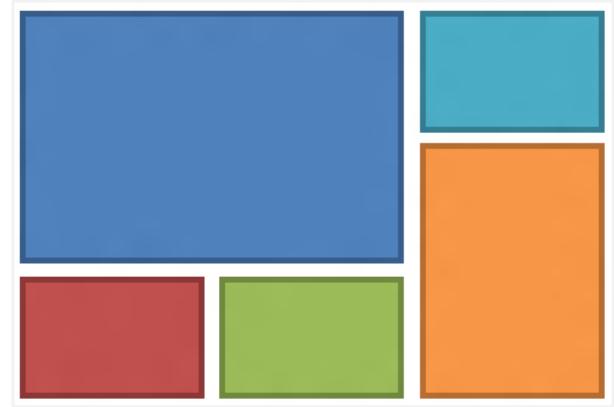
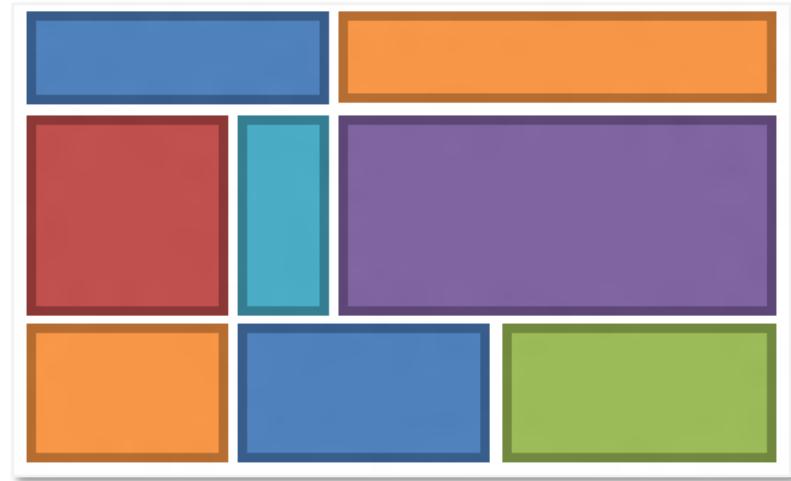
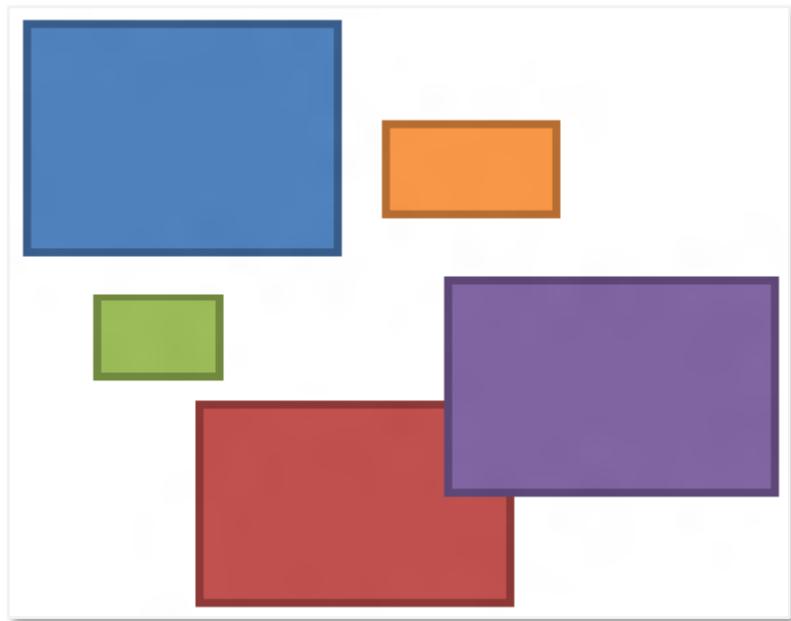
- Képernyő méret, pixel sűrűség



- Felhasználói felület arányai



Elrendezés



Elrendezés

- Speciális vezérlők végzik a rajtuk lévő vezérlő elrendezését
- Egyedi algoritmusok
 - > Táblázat jellegű, sorfolytonos stb.
- Paraméterek a rendező vezérlőn
- Paraméterek a rendezendő vezérlőkön
- Az elrendezéshez tartoznak általános tulajdonságok

Méretezés – UIElement . Visibility

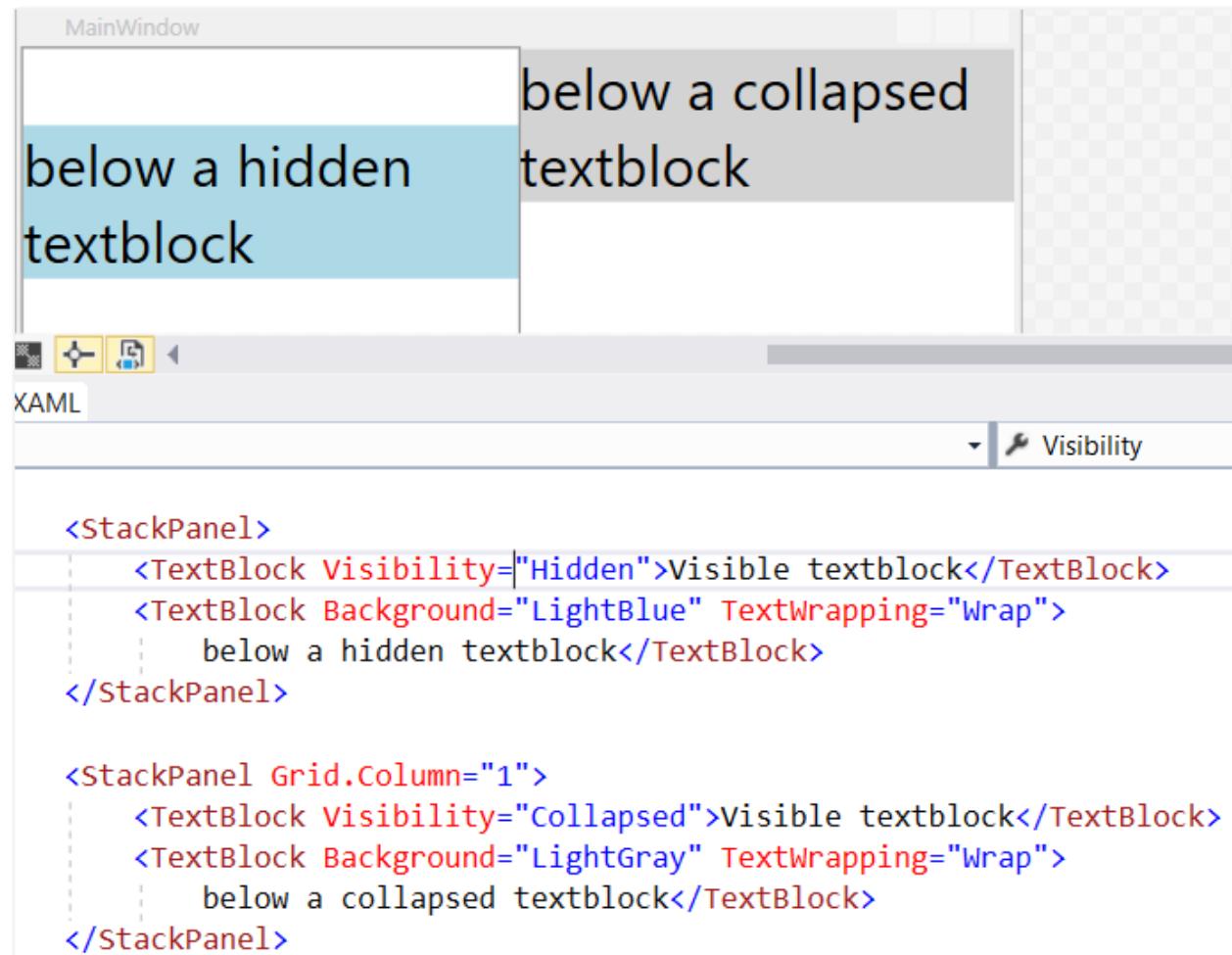
- Visibility – enum nem bool!

> **Visible**: látszik

> **Collapsed**:

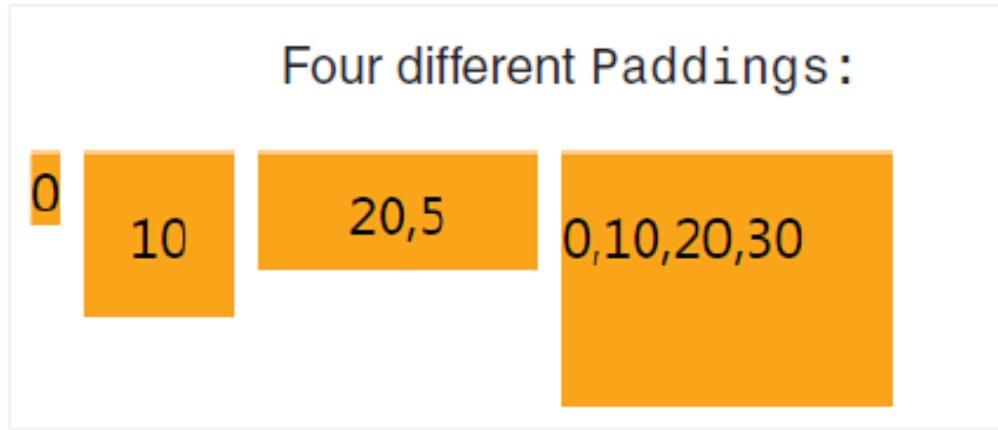
nem látszik,
és nem is
foglal helyet

> WPF bónusz:
Hidden

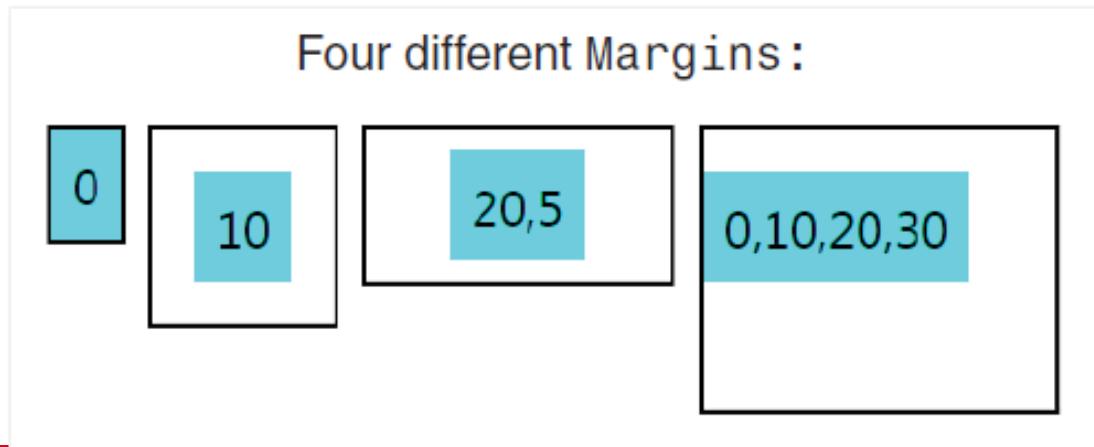


Méretezés – Margin, Padding

- Padding:

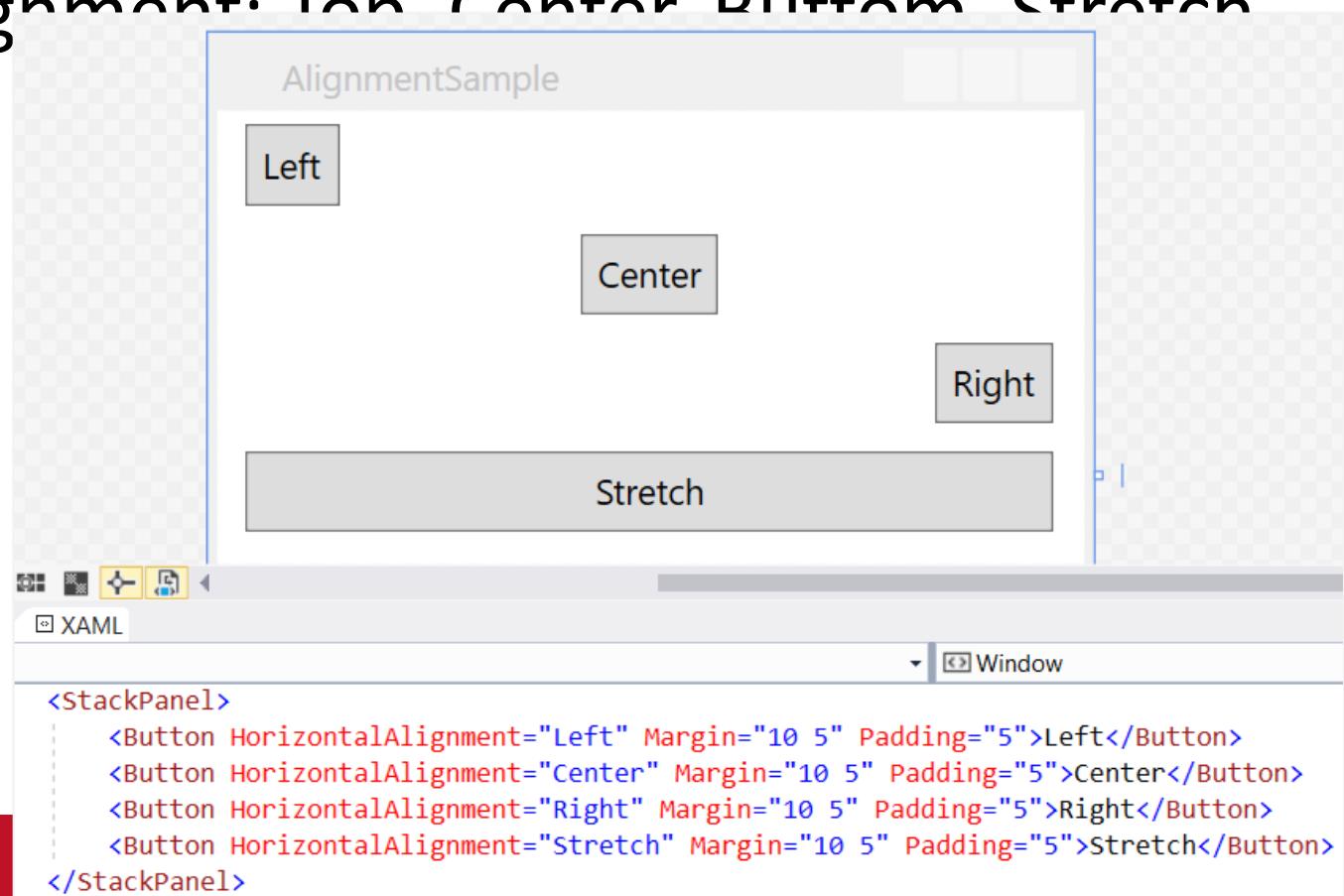


- Margin (Thickness):



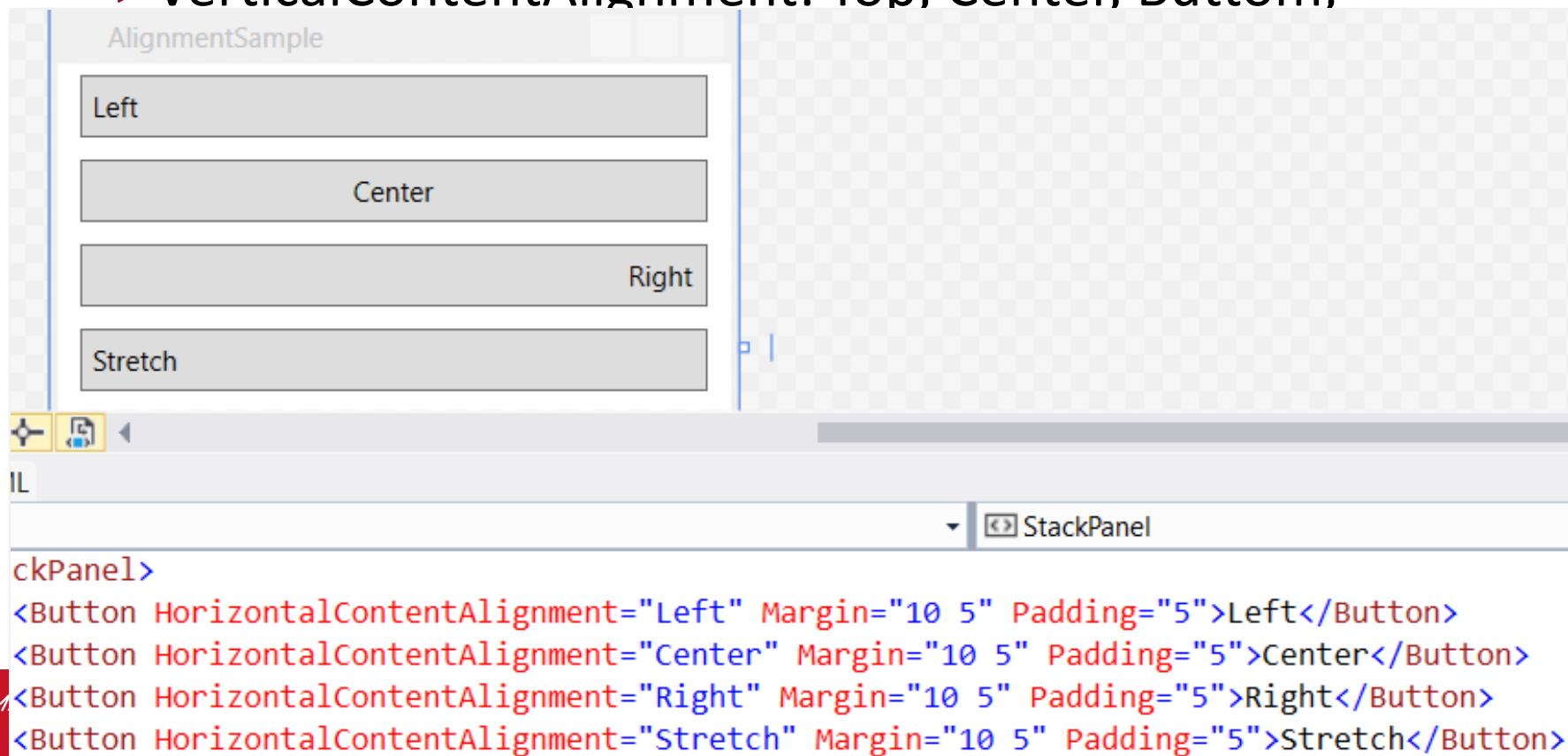
Pozícionálás - vezérlő

- HorizontalAlignment: Left, Center, Right, Stretch
- VerticalAlignment: Top, Center, Bottom, Stretch



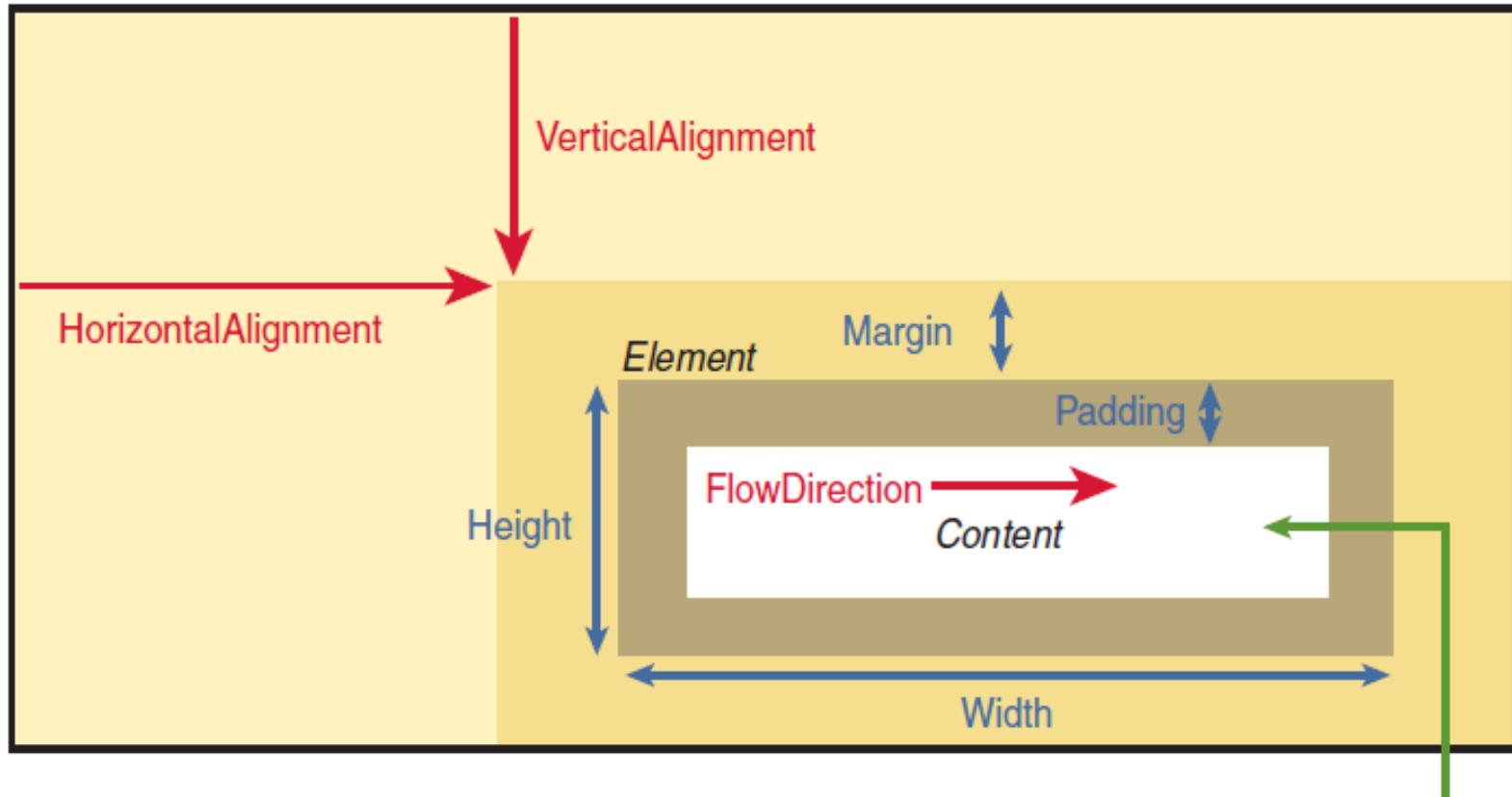
Pozícionálás - tartalom

- Tartalom igazítás a Control osztályban
 - > HorizontalContentAlignment: Left, Center, Right, Stretch
 - > VerticalContentAlignment: Top, Center, Bottom,



Méretezés

Panel



LayoutTransform
RenderTransform

Általános paraméterek

- **Horizontal/Vertical Alignment**
 - > A szülőn belül hova kerüljön
 - > Left/Top, Right/Bottom, Center, Stretch
- **Margók:** Left, Right, Top, Bottom
 - > Az elem széleinek távolsága a szülő vagy egyéb vezérlők megfelelő széleitől
 - > Az ActualHeight/Width-en kívül értelmezettek
- **Padding:** csak néhány elemen van (pl border)
 - > A benne lévő elemek távolsága a panel szélétől

Méretezés – Width, Height

- Tulajdonságok:
 - > Width, Height (alapértelmezett Auto = Double.NaN)
 - > MinWidth, MinHeight (alapértelmezett 0)
 - > MaxWidth, MaxHeight (def: Double.PositiveInfinity)
- Számold tulajdonságok:
 - > DesiredSize -> layout készítéskor számolja
 - > RenderSize -> renderelés után számolja
 - > ActualHeight, ActualWidth -> renderelés után

Framework szintű elrendezés

- A Panel gyerekeinek elrendezése
- Az elemek pozíójának, méretének változása új layout számítást indít – lassú lehet
 - > Aszinkron: a méret változás nem indukál szinkron layout számítást (2 queue: measure, arrange)
- Két fázisú rekurzív rendező algoritmus
 - 1. Measure – felmérés**
 - 2. Arrange – elrendezés**
- minden UIElement egy befoglaló téglalapban van

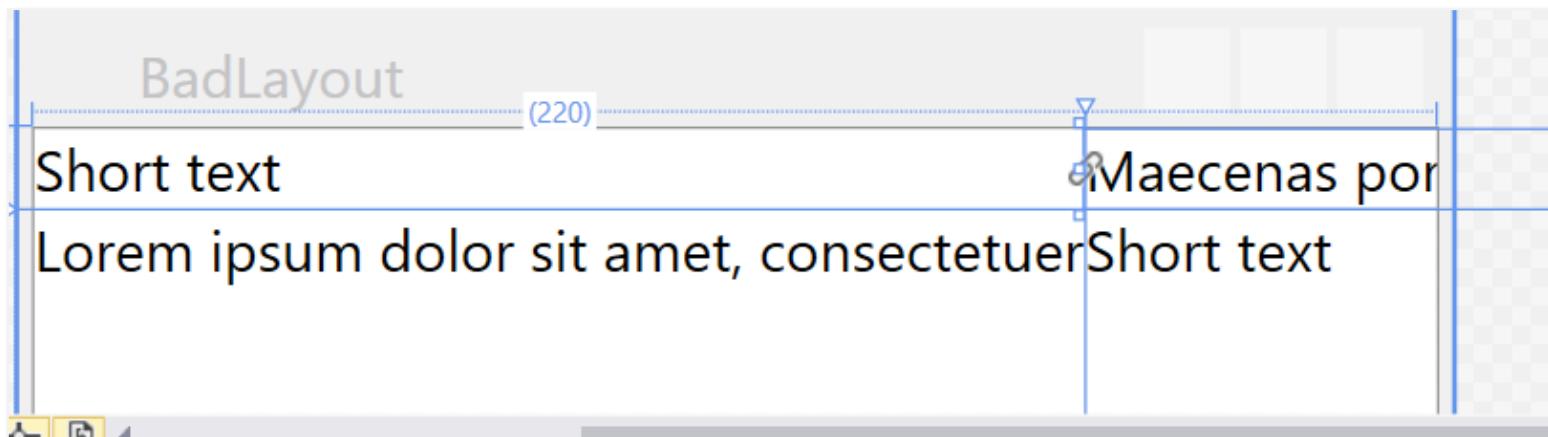
1. Fázis: felmérés (Measure)

- UIElement . Measure(AvailableSize)
- A panel hívja a gyerekein megadva, hogy mennyi a legnagyobb lehetséges méret
 - > ScrollViewer esetén például végtelen!
- Az ideális méretet a gyerek adja meg (DesiredSize)
- Általános paraméterek figyelembe vétele
 - > Clip, Max/Min Width/Height, Margin

2. Fázis: elrendezés (Arrange)

- A Panel (aki rendez) létrehoz egy befoglaló téglalapot mindenek elemnek a panel saját algoritmusának megfelelően
- ArrangeCore hívódik a befoglaló téglalappal:
 - > Utolsó simítások: ArrangeSize, FinalSize
 - > Vezérlő elhelyezése a téglalapon belül (Alignment)

A két fázis problémái...



The screenshot shows a Windows Forms application's designer interface. A `Grid` control is displayed with the following structure:

```
<Grid>
    <RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </RowDefinitions>
    <ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto"/>
    </ColumnDefinitions>
    <TextBlock Grid.Row="0" Grid.Column="0" TextWrapping="Wrap">short text</TextBlock>
    <TextBlock Grid.Row="0" Grid.Column="1" TextWrapping="Wrap">
        Maecenas porttitor congue massa. Fusce posuere ac
        ultricies nunc. Nullam id nisi non turpis
        euismod lacinia. Sed et diam ut
        neque. Ut euismod, nisl id
        tincidunt
    </TextBlock>
    <TextBlock Grid.Row="1" Grid.Column="0" TextWrapping="Wrap">
        Lorem ipsum dolor sit amet, consectetur
        adipiscing elit. Donec
    </TextBlock>
    <TextBlock Grid.Row="1" Grid.Column="1" TextWrapping="Wrap">short text</TextBlock>
</Grid>
```

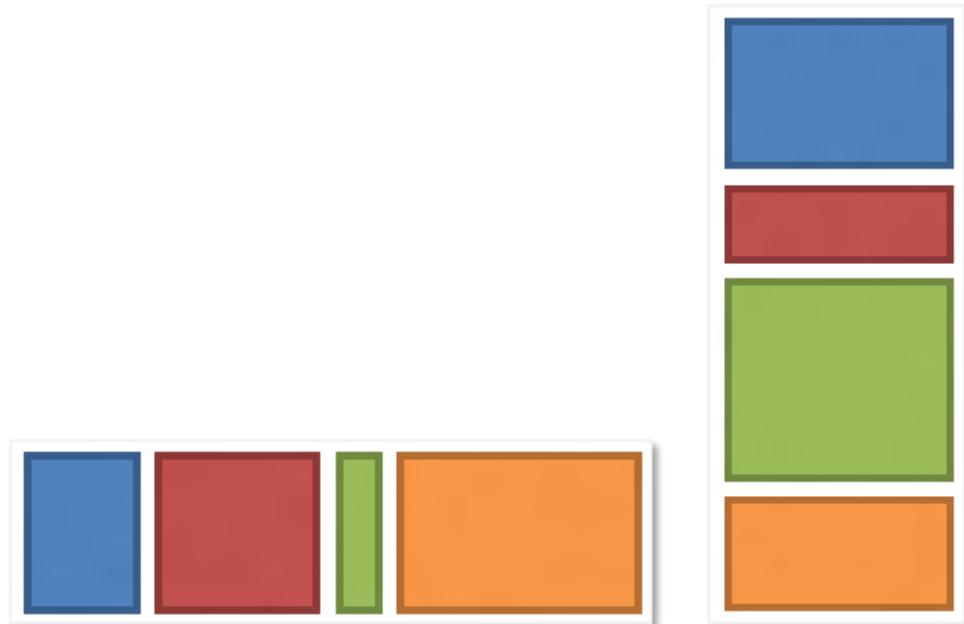
```
BME / UI XAML 3
```

Panel alaposztály

- Gyűjtő vezérlő
 - > Children: gyerek elemek
 - > Background: háttér
 - > ChildrenTransitions: tranzíció animációk (UWP)

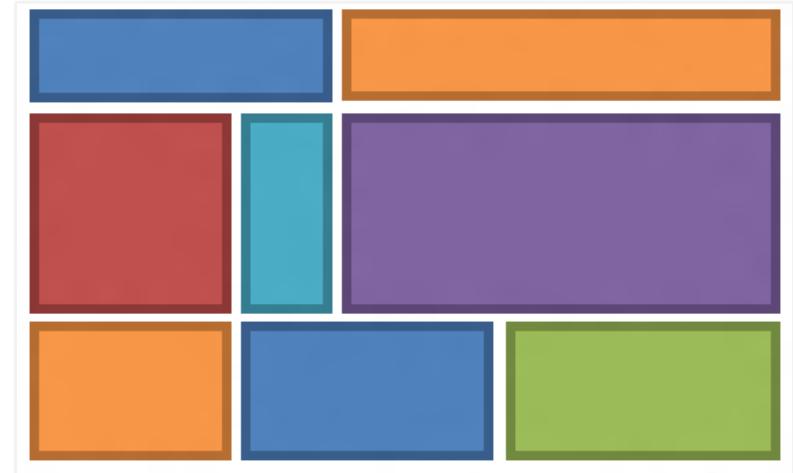
StackPanel

- Elemek egymás alatt vagy mellett
- Orientation
 - > Horizontal
 - > Vertical
- Virtualizált változat:
 - > VirtualizingStackPanel



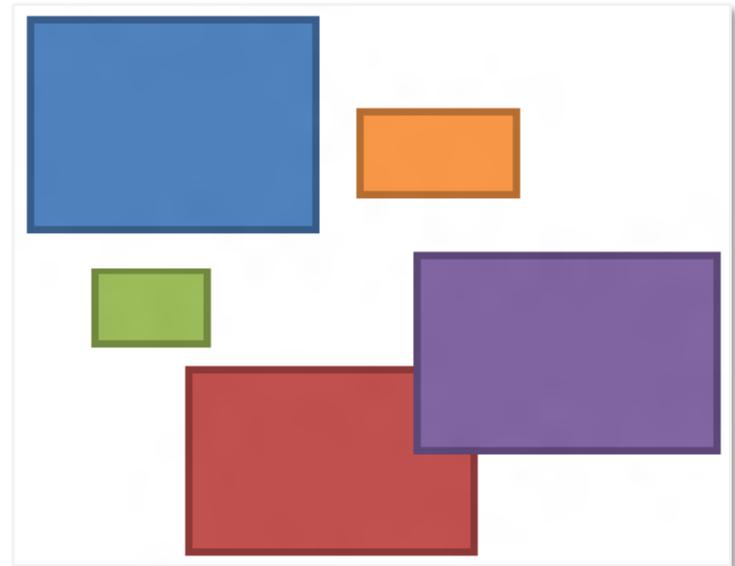
Grid

- Táblázatos megjelenítés
- Sorok/oszlopok megadása:
 - > Fix pixel méret
 - > Automatikus (a benne lévő vezérlőktől függ)
 - > Arányos (*)
- Gyerekelemek elhelyezése
 - > Grid.Row
 - > Grid.Column
 - > Grid.**RowSpan**
 - > Grid.**ColumnSpan**



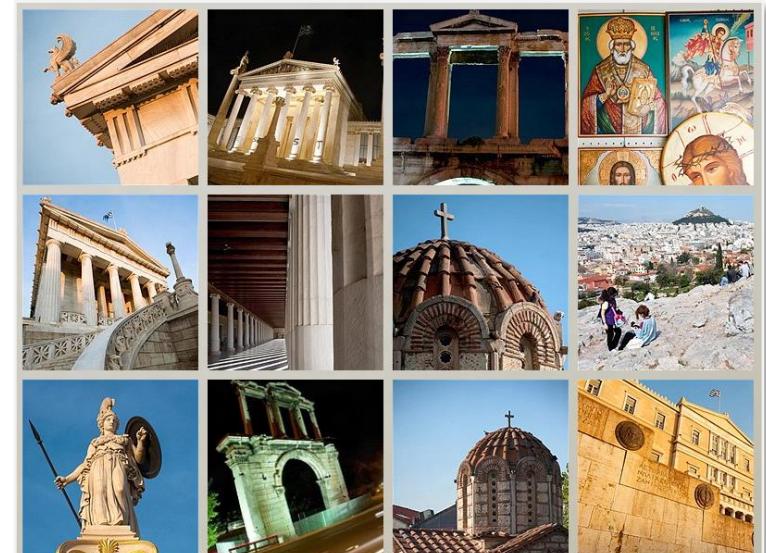
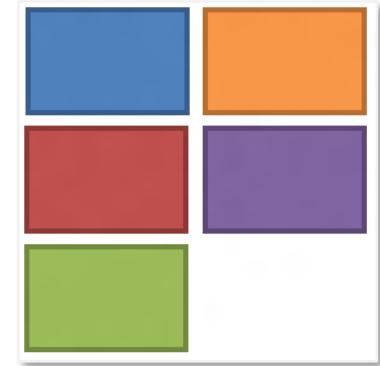
Canvas

- Gyerek vezérlők szabadon elhelyezhetők fix pixel koordinátákkal:
 - > Canvas . Left, Right, Top, Bottom
- Zindex állítás csatolt tulajdonságokkal
- A vezérlők átfedhetnek!



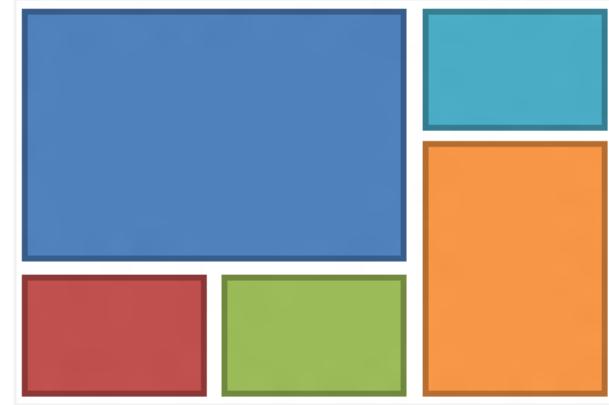
WrapGrid

- Sor/oszlop-folytonos megjelenítés
- Elemek mérete:
 - > ItemHeight
 - > ItemWidth
 - > Ha nincs beállítva, akkor az első elem mérete alapján
- Egyéb tulajdonságok:
 - > MaximumRowsOrColumns
 - > Orientation
- Csak listáknál használható
- Virtualizált



VariableSizedWrapGrid

- Elemek mérete:
 - > ItemHeight, ItemWidth csatolt tulajdonság
 - > Ha nincs beállítva, akkor az első elem mérete alapján
- Különböző elemméretek (attached property)
 - > RowSpan, ColumnSpan
- Egyén tulajdonságok:
 - > MaximumRowsOrColumns
 - > Orientation
- Csak listáknál használható
- Nem virtualizált! ☹

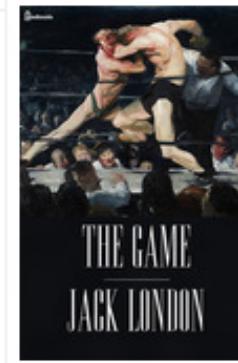


RelativePanel - csak UWP

- Az elemek helyét *egymáshoz képest* kell megadni, és a panel kiszámolja az „ideális” elrendezést
- Előny: rugalmas, adaptív elrendezés, kevesebb panellel

Példa: RelativePanel nélkül

```
<Grid>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Image x:Name="img" .../>
    <StackPanel Grid.Column="1" ...>
        <TextBlock x:Name="title" .../>
        <TextBlock x:Name="authors" .../>
        <TextBlock x:Name="summary" .../>
        <Button Content="Download" .../>
    </StackPanel>
</Grid>
```



The Game

Jack London

On the eve of their wedding, twenty-year-old sweethearts...
the prizefight that pits her fair young lov...

Download

RelativePanel

```
<RelativePanel>
    <Image x:Name="img" .../>
    <TextBlock x:Name="title"
        RelativePanel.RightOf="img"
        RelativePanel.AlignTopWith="img" .../>
    <TextBlock x:Name="authors,,"
        RelativePanel.RightOf="img"
        RelativePanel.Below="title" .../>
    <TextBlock x:Name="summary,,"
        RelativePanel.RightOf="img"
        RelativePanel.Below="authors" .../>
    <Button Content="Download" RelativePanel.RightOf="img"
        RelativePanel.AlignBottomWithPanel="True" .../>
</RelativePanel>
```



The Game

Jack London

On the eve of their wedding, twenty-year-old sweethearts have a secret desire to view her only rival: the "game" of the prizefight that pits her fair young love against

Download

Sablonok

West Coast Swing champion dancers

Photo	Name	Role	Points
	Kyle Redd	Leader	601
	Brandi Tobias	Follower	246
	Michael Kielbasa	Leader	173
	Martin Maxence	Leader	89
	Virginie Grondin	Follower	24

Sign in

ADjam

0:34 / 3:28

MADjam 2013 Champions J&J
Kyle Redd & Torri Smith

DANCE JAM PRODUCTIONS

Dance Jam Productions

Subscribe 4.9K

38,262 views

Sablonok feladata, típusai

- Azonos típusú adatok megjelenítése egységes módon
 - > minden adat objektumhoz azonos vizuális fa jön létre
 - > A megjelenés paramétereit az objektum tartalmazza
- Vezérlők (például gomb) egységes megjelenése
- A sablonok típusai
 - > Adat sablon (DataTemplate): adat megjelenítése
 - > Vezérlő sablon (ControlTemplate): vezérlő
 - > Listázó panel (ItemsPanelTemplate): elem listázás

Adat sablon példa

```
<Window.Resources>
    <DataTemplate DataType="{x:Type local:Dancer}">
        <StackPanel Orientation="Horizontal">
            <Image Source="{Binding Image}" Height="70"
                  Width="100" Stretch="UniformToFill" />
            <TextBlock Margin="10 5">
                <Run Text="{Binding FirstName}" />
                <Bold><Run Text="{Binding LastName}" /></Bold>
                <LineBreak/>
                <Run Text="{Binding Role}" />
                <LineBreak/>
                <Run Text="Points:" />
                <Bold><Run Text="{Binding Point" />
            </TextBlock>
        </StackPanel>
```

West Coast Swing champion dancers



Kyle **Redd**
Leader
Points: **601**



Brandi **Tobias**
Follower
Points: **246**

Adat sablon (DataTemplate)

- Az adatok megjelenítését a **ContentPresenter** kezeli
 - > UIElement / Sablon keresés / ToString()
- minden ContentControl tartalmaz egy ContentPresentert
- Sablon választási logika
 - > Explicit: **ItemTemplate** tulajdonsága
 - > Implicit: a sablon **DataTemplate** tulajdonsága ki van töltve
 - > Adatelemenként: **DataTemplateSelector** használatával
 - > Programozottan a **LoadContent** példányosít

ContentPresenter használata

Adatok
egységes
sablonnal

Tervezés
idejű
adatforrás

ContentPresenter
közvetlen használata

The screenshot shows a user interface in Microsoft Blend. At the top, there's a title bar with the text "GridCP". Below it is a stack panel containing three items, each with a photo and some text. The first item is for "Kyle Redd", the second for "Brandi Tobias", and the third for "Michael Kielbasa". Each item includes a photo, a name, a role (Leader or Follower), and a points count (601, 246, or 173). Below the stack panel is a toolbar with several icons. At the bottom of the screen, the XAML code for the UI is visible:

```
d:DataContext="{StaticResource sampleDancerList}"
Title="GridCP" Height="300" Width="300">
<StackPanel>
    <ContentPresenter Content="{Binding Path=.[0]}" />
    <ContentPresenter Content="{Binding Path=.[1]}" />
    <ContentPresenter Content="{Binding Path=.[2]}" />
    <ContentPresenter Content="{Binding Path=.[3]}" />
</StackPanel>
```

ContentControl és Presenter

```
<Button Margin="10" Width="50" Height="20"  
       Content="OK"  
<Button Margin="10" Width="120" Height="50"  
       <TextBlock>  
           <Bold><Run Text="Yes"/></Bold>  
           <LineBreak/>  
           <Run Text="Keep the items!" />  
       </TextBlock>  
</Button>  
<Button Margin="10" Width="200" Height="80"  
       Content="{Binding Path=. [0]} />
```

templateSample

OK

Yes

Keep the items!

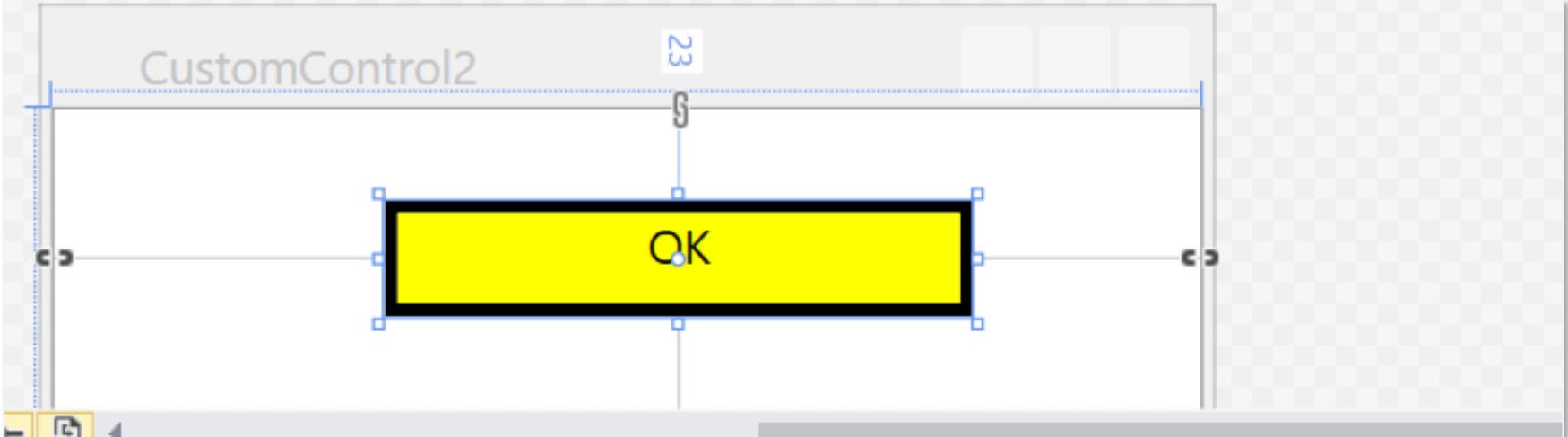


Kyle **Redd**
Leader
Points: **601**

Vezérlő sablonok

- A XAML keretrendszerben a vezérlő kódja független a megjelenéstől
- A vezérlő vizuális megjelenését és vizuális működését a **ControlTemplate** határozza meg
- A template-ben jelzni kell a tartalom helyét
 - > ContentControlnál a **ContentPresenter** osztály
 - Button esetén ide kerül a tartalom (Content)
 - > ItemsControl-nál az **ItemsPresenter** (vagy a Panel, IsItemsHost=true) ad helyet az elemeknek
 - például ListBoxItemek

Vezérlő sablon példa 1.



The screenshot shows the Windows Forms Designer interface. At the top, there's a title bar with the text "CustomControl2". Below it is a control named "CustomControl2" which contains a single button. The button has a yellow background, a black border with a thickness of 3 pixels, and the text "OK" centered within it. The designer interface includes a toolbar at the bottom left and a status bar at the bottom right.

```
<Button Content="NO, IT'S NOT OK" Height="30" Width="150" HorizontalA
    <Button.Template>
        <ControlTemplate>
            <Border Background="Yellow" BorderBrush="Black"
                BorderThickness="3">
                <TextBlock HorizontalAlignment="Center" Text="OK" />
            </Border>
        </ControlTemplate>
    </Button.Template>
</Button>
```

The code in the screenshot is XAML-like XML. It defines a button with the content "NO, IT'S NOT OK". Inside the button's template, there is a control template that contains a border with a yellow background and a black border thickness of 3. Inside the border, there is a text block with horizontal alignment set to center and the text "OK". The content "NO, IT'S NOT OK" and the text "OK" are highlighted with red boxes, indicating they are the focus of the example.

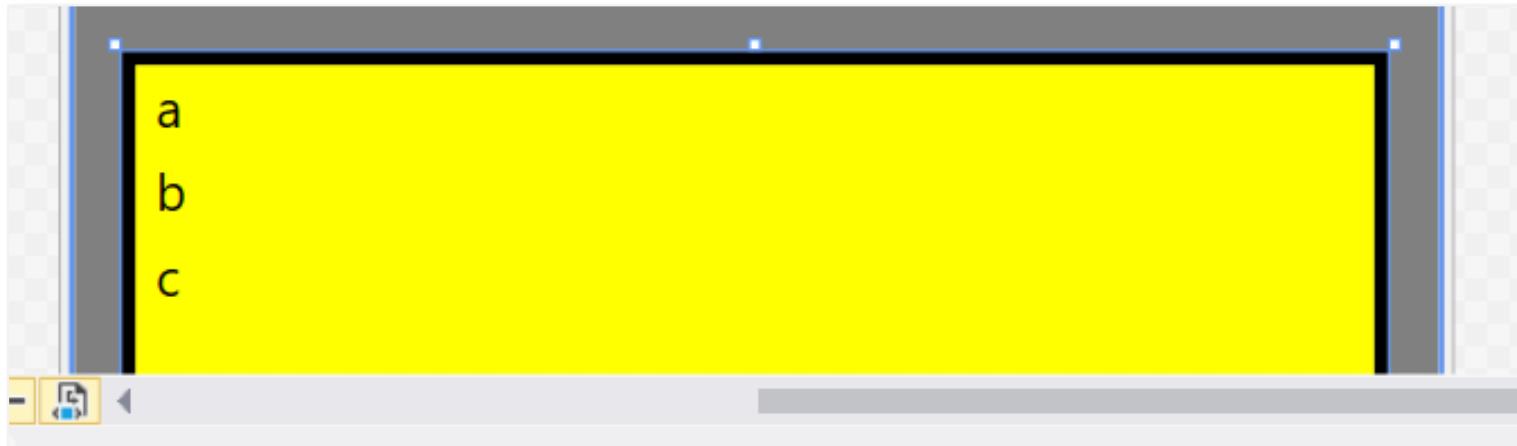
Vezérlő sablon példa 2.

The screenshot shows the Windows Presentation Foundation (WPF) Designer interface. At the top, a window titled "CustomControl3" displays a yellow button with a black border and padding, containing the text "NO, IT'S NOT OK". The button is centered within a grid layout. Below the designer, the XAML code is displayed:

```
<ControlTemplate x:Key="myButton" TargetType="Button">
    <Border Background="Yellow" BorderBrush="Black" Padding="10 3"
        HorizontalAlignment="Center" VerticalAlignment="Center"
        BorderThickness="3">
        <ContentPresenter />
    </Border>
</ControlTemplate>
</Window.Resources>
<Button Content="NO, IT'S NOT OK" Template="{StaticResource myButton}"
    Background="Red"
    Height="30" Width="150" HorizontalAlignment="Center" VerticalAlignme
```

The XAML code defines a control template named "myButton" for a Button control. The template consists of a Border element with a yellow background, black border, and padding of 10 on the left and right and 3 on the top and bottom. The ContentPresenter is used to display the button's content. This template is then applied to a Button element with the following properties: Content set to "NO, IT'S NOT OK", Template set to the static resource "myButton", Background set to "Red", Height set to "30", Width set to "150", HorizontalAlignment set to "Center", and VerticalAlignment set to "Center".

ItemsControl sablon példa



```
<ListBox Background="Yellow" Height="200" Width="300">
    <ListBox.Template>
        <ControlTemplate>
            <Border Background="{TemplateBinding Background}"
                    BorderBrush="Black" BorderThickness="3">
                <ItemsPresenter />
            </Border>
        </ControlTemplate>
    </ListBox.Template>
    <ListBoxItem>a</ListBoxItem>
    <ListBoxItem>b</ListBoxItem>
    <ListBoxItem>c</ListBoxItem>
```

Vezérlők vizuális állapotai

- ..VisualStateGroups: fületlen állapot csoportok
 - > A vezérlő csoportonként pontosan egy állapotban van
- ..VisualState: állapotok egy csoporton belül
- Például ComboBox állapotai:
 - > CommonStates
 - Normal, Disabled, MouseOver, ...
 - > FocusStates
 - Focused, Unfocused, ...
 - > ValidationStates:
 - Valid, InvalidFocused, ...

Állapot váltás

- VisualStateManager
 - > . GotoState(FE control, state, useTransition)
- Beépített vezérlők testreszabása
 - > Dokumentációban rögzített csoport és állapotnevek
 - > A vezérlő-logika hívja a GotoState metódust
 - > Az állapotoknál megfelelően kell elnevezni a sablonban
- Saját manager is készíthető, származtatással

Állapotok: VisualState

- Az állapothoz a nevén kívül a megjelenés leírása tartozik (Storyboarddal)
- Storyboard állapothoz
 - > Az adott állapot vizuális megjelenését írja le
 - > Tipikusan animáció nélkül!
- Transition – animáció generálás automatikusan
 - > Az animációk lefutását definiálják
 - > Melyik állapotból melyik másikba
 - > Mennyi idő alatt (GeneratedDuration)
 - > Milyen lefutással (GeneratedEasingFunction)

VisualState példa

Yes

```
<Window.Resources>
```

```
    <VisualStateManager.VisualStateGroups>
        <VisualStateGroup Name="CommonStates">
            <VisualState Name="Normal" />
            <VisualState Name="PointerOver">
                <Storyboard>
                    <ColorAnimation Storyboard.TargetName="background" Duration="0:0:0.5"
                        Storyboard.TargetProperty="Color" To="Red" />
                </Storyboard>
            </VisualState>
            <VisualStateGroup.Transitions>
                <VisualTransition To="PointerOver" />
            </VisualStateGroup.Transitions>
        </VisualStateGroup>
    </VisualStateManager.VisualStateGroups>
```

```
private void button_MouseEnter(object sender, MouseEventArgs e)
{
    VisualStateManager.GoToState(button, "PointerOver", true);
}
```

Stílusok

- Tetszőleges tulajdonságok beállítása
FrameworkElement-ből származó osztályokon

```
<Style TargetType="ListBoxItem">  
    <Setter Property="Opacity" Value="0.5" /> ...
```

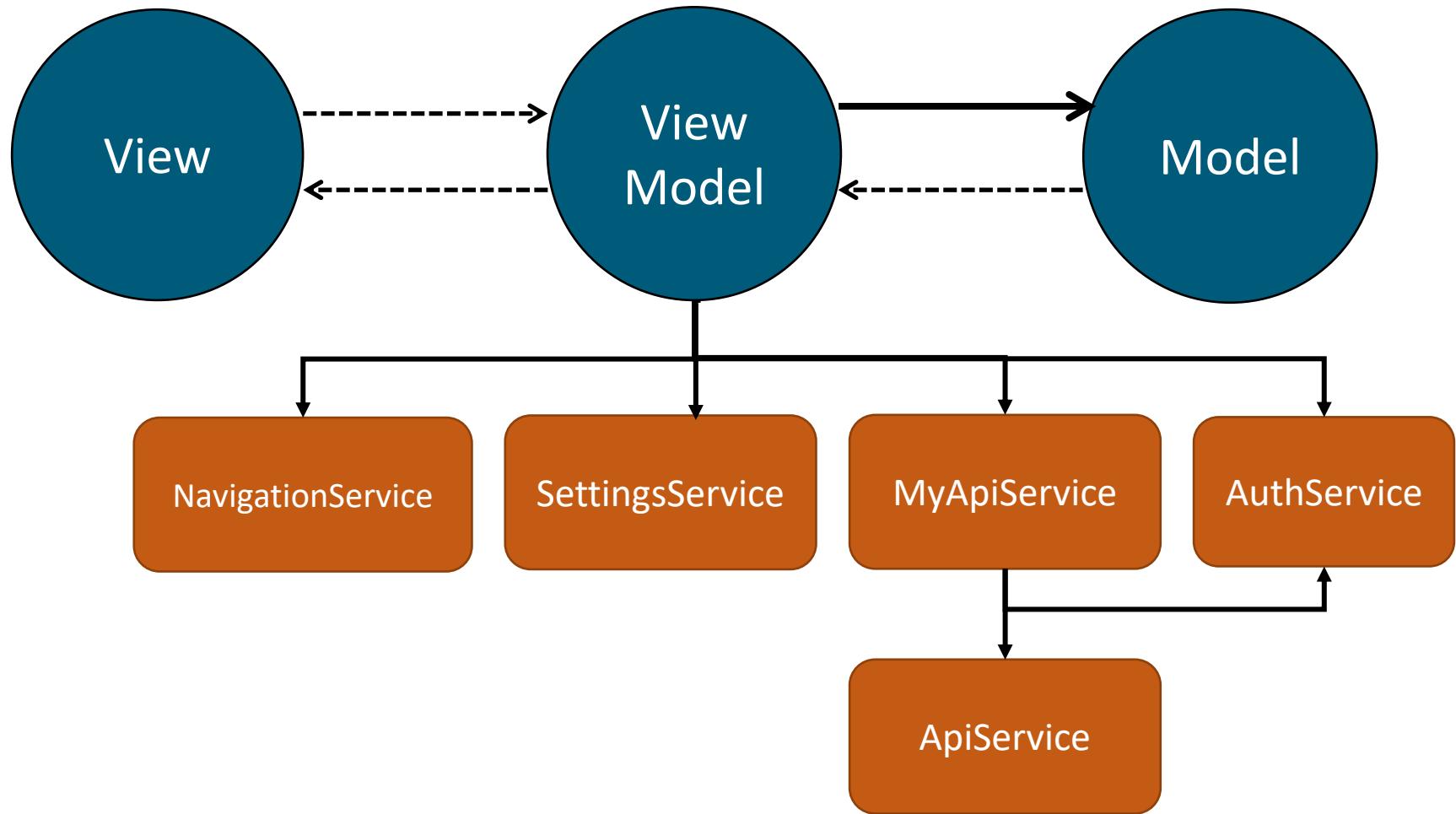
- Egy vezérlőhöz egyszerre csak egy stílus tartozhat
 - > De a stílusok származhatnak egymásból
- Tipikusan erőforrásként definiáljuk a stílusokat
 - > TargetType megadásával automatikusan érvényre jut
 - > Kivéve ha neve van a stílusnak (explicit)
- minden beépített vezérlőnek van saját stílusa, az határozza meg a kinézetét a Template-tel

Stílus példa

- TextBlock tulajdonságokat állít be
- Ezek utána példányonként változtathatóak
 - > Lásd DP prioritás

```
B <Style TargetType="TextBlock" x:Key="TextBlockStyle">  
    <Setter Property="Foreground" Value="Navy"/>  
    <Setter Property="FontSize" Value="14"/>  
    <Setter Property="VerticalAlignment" Value="Bottom"/>  
    <Setter Property="Background">  
        <Setter.Value>  
            <LinearGradientBrush StartPoint="0,0.5" EndPoint="1,0.5">  
                <GradientStop Color="White" Offset="0.0"/>  
                <GradientStop Color="Navy" Offset="1"/>  
            </LinearGradientBrush>  
        </Setter.Value>  
    </Setter>  
B </Style>
```

MVVM minta XAML technológiákhoz

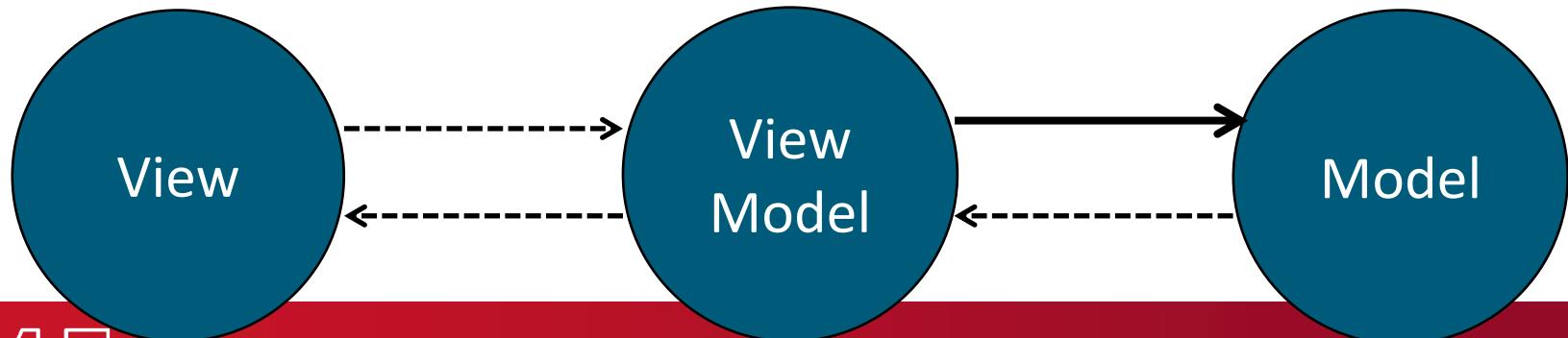


MVVM minta nélkül

- Hol van a kód?
 - > Page vagy UserControl code-behind
 - > Modell osztályok, üzleti logikát, adathozzáférést végző osztályok, ...
- Probléma:
 - > A code-behindban keveredik az alkalmazás logika a megjelenítéshez tartozó részekkel
- Ez miért gond?
 - > Komplexebb alkalmazásokban áttekinthetetlen
 - > Nehéz karbantartani
 - > Nem válnak el a funkciók (biztos, hogy ugyan az a személy fejleszti a kettő...?)

MVVM minta felépítés

- Megoldás: válasszuk külön az alkalmazáslogikát a megjelenítéstől
- Model:
 - > Domain osztályok, adathozzáférés
- ViewModel:
 - > Alkalmazás logika (pl. adatok frissítése - távoli hívások, validáció, modell módosítása, adattranszformáció)
 - > Felületen megjelenő adatok
 - > Absztrakt nézet (pl. nézet állapotok, kiválasztott listaelem)
- View:
 - > Csak a megjelenítéshez kapcsolódó részek



MVVM általános alapelvek

- **Model** nem hivatkozik senkire
- **ViewModel** és Model kapcsolata
 - > Hivatkozás vagy burkolás
- **View** és ViewModel
 - > ViewModel általában a DataContext
 - > Adatkötés (INotifyPropertyChanged)
 - > Command
- A VM általában nem hivatkozik a Viewra
 - > Ha nem elég az adatkötés akkor interfész implementálhat a view (pl. dialógusablakot kell feldobni az üzleti logika hatására)

Model

- Domain specifikus adatosztály
 - > Üzleti objektum, szerver oldali adatokból építjük
- **Tartalmilag a felülethez alkalmazkodjon**
 - > Ne adjunk át felesleges adatokat
 - > Egy felület megjelenítéséhez ne kelljen többször a szerverhez fordulni
 - > A szolgáltatás interfészt a UI határozza meg
- Formára legyen független a UI típusuktól stb

ViewModel

- A felülethez kapcsolódó logika van benne
- Felület állapotait tartalmazza és vezeti ki
 - > UI specifikus propertyk megjelenítési technológiától függetlenül (IsVisible vs Visibility)
- A felület állapot átmeneteit implementálja
 - > Navigáció
 - > Felhasználói input ellenőrzés
 - > Hiba visszajelzés, hiba állapot
- A domain adatokat a Modelből veszi
 - > A szerver oldalt hívva hozza létre a Modelek

View

- Gazdag deklaratív leírás: XAML
- Adatkötéssel implicit hivatkozik a ViewModelre és Modelre
- A code-behind tipikusan üres
 - > Command mintával kapcsolódunk a ViewModelhez
 - > Speciális eseménykezelőket lehet implementálni
- A ViewModel eseményeken vagy interfészen keresztül érheti el
 - > Melyiknek milyen előnye van?

A View felépítése

- XAML
 - > Adatkötéssel a DataContext-en keresztül a ViewModelhez vagy Modelhez van kötve
- Code-behind (.cs/.vb fájl)
 - > Inicializáló kód (pl DataContext beállítása stb)
 - > Tipikusan eseménykezelők implementálása

Command minta

- X:Bind esetén nem kell! ☺
- Ne kelljen eseménykezelőt írni a code-behindba
 - > Túl nagy kísértés ☺
 - > Leválasztott logika, tesztelhetőség, újrafelhasználás
- A View-ból az eseményeket közvetlenül a ViewModel kapja meg
- A menü és gomb az **ICommand** interfészen keresztül támogatják az események átadását
 - > A többi eseményhez és vezérlőhöz is készíthető csatolt tulajdonság, ami támogatja az ICommandot

Az ICommand interfész

Execute(object param)

- > A műveletet végre kell hajtani

bool CanExecute(object param)

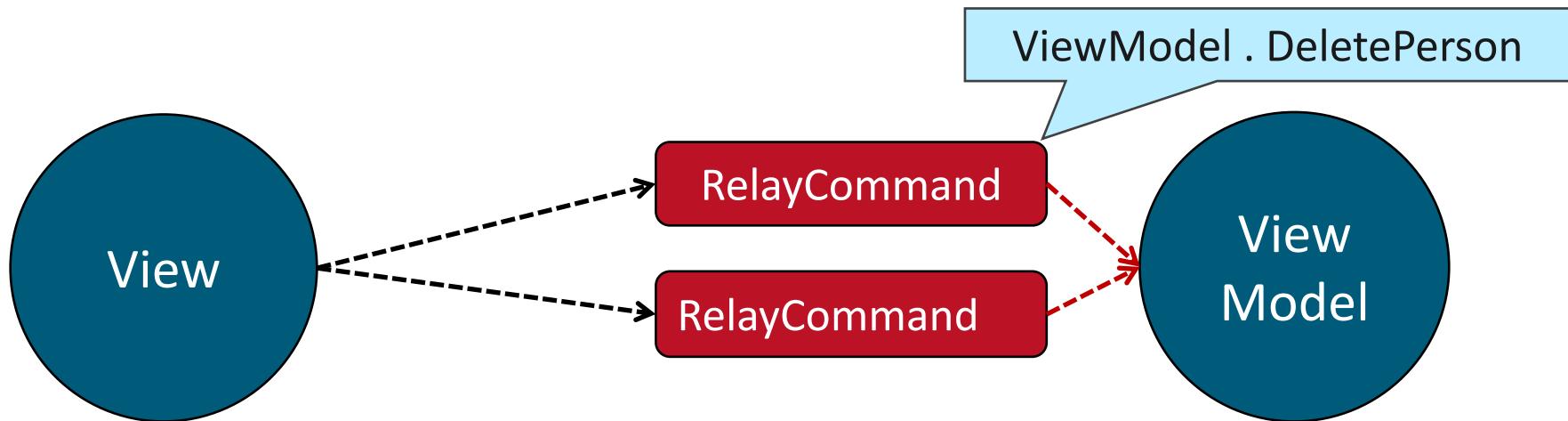
- > Végrehajtható-e a művelet az aktuális UI állapotban?

EventHandler CanExecuteChanged

- > A végrehajthatóság megváltozott

Relay/DelegateCommand

- A ViewModel minden eseményhez egy külön híd objektumot publikál
 - > Ez a híd implementálja az ICommandot
 - > A hidat adatkötni kell a Button Click eseményére
 - > A híd az esemény hatására behív a ViewModel-be
 - > Például DeletePersonCommand



Command példa

```
public DelegateCommand<string> SaveDocCommand { get; }

public MainViewModel()
{
    SaveDocCommand = new DelegateCommand<string>(SaveDoc);
}

private void SaveDoc(string param)
{
    // ...
}

<Button Content="Press"
        Command="{Binding SaveDocCommand}"
        CommandParameter="MyParam"/>
```

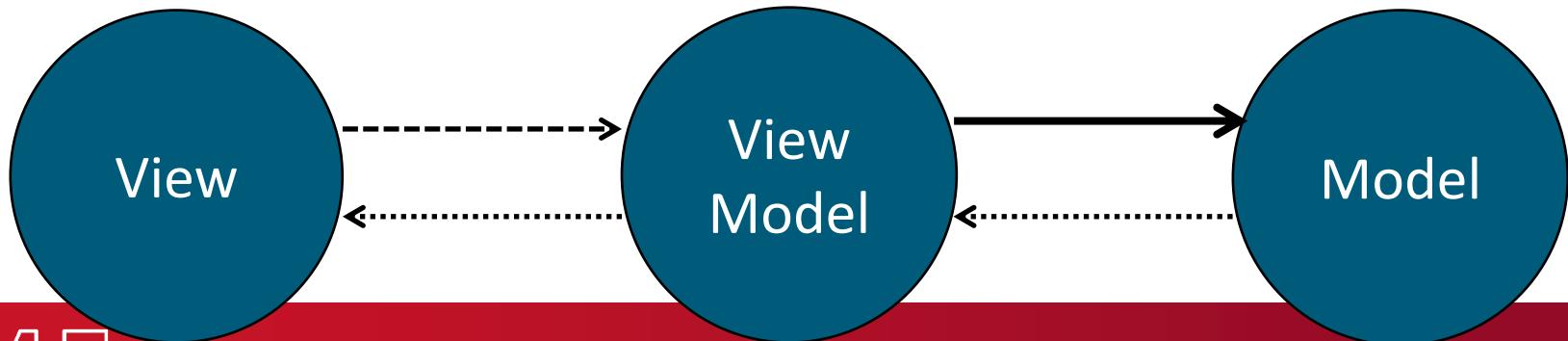
Ha nincs Command támogatás - Behaviors

- Alapból csak a gomb vezérlők támogatják a command mintát
 - > Mit tegyünk ha máshol is kellene?
- Vezérlők kiterjesztése Behavior objektumokkal
 - > Hasonló az attached property mechanizmushoz
 - > Vannak beépített behaviorök
 - Pl.: Esemény – Command összekapcsolás

```
<TextBlock Text="Sample">
  <i:Interaction.Behaviors>
    <ic:EventTriggerBehavior EventName="Tapped">
      <ic:InvokeCommandAction Command="{Binding SaveDocCommand}" />
    </ic:EventTriggerBehavior>
  </i:Interaction.Behaviors>
</TextBlock>
```

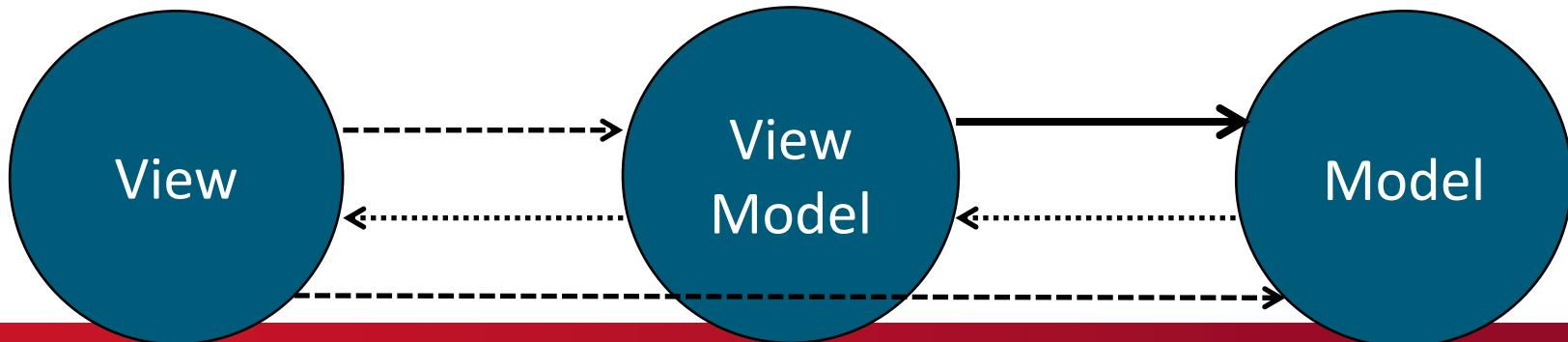
Strict MVVM

- View nem hivatkozik a Modelre csak a VM-re
 - > Az adatkötésekben csak a ViewModel tulajdonságai szerepelnek, a Model tulajdonságai nem
- minden Model osztály be van csomagolva ViewModel osztályba (pl. Person -> PersonViewModel)
 - > A Model felületen megjelenő tulajdonságait a ViewModel tulajdonságai csomagolják
 - > ViewModel implementálja a felületlogikához szükséges interfészeket (pl. INPC, IEditableObject)
 - > Általában kódgenerálás
- A Model teljesen független lehet a felületlogikától
 - > Például nem kell INPC a Modelbe



Relaxed MVVM

- View hivatkozik a Modelre
 - > A hivatkozás implicit az adatkötésekben
 - > A Model tulajdonságait közvetlenül kötjük a felületre
- ViewModel hivatkozik a Modelre
 - > Publikus tulajdonságokon keresztül elérhető a Model (pl. List<Person> People)
 - > A Modelnek támogatni kell a szükséges interfészeket (pl. INPC, IEditableObject)
 - Pl. WCF proxy generátor ezt részben biztosítja, a további bővítéshez partial kiegészítés a Model osztályokhoz
- A ViewModel eseménnyel értesítheti Viewt



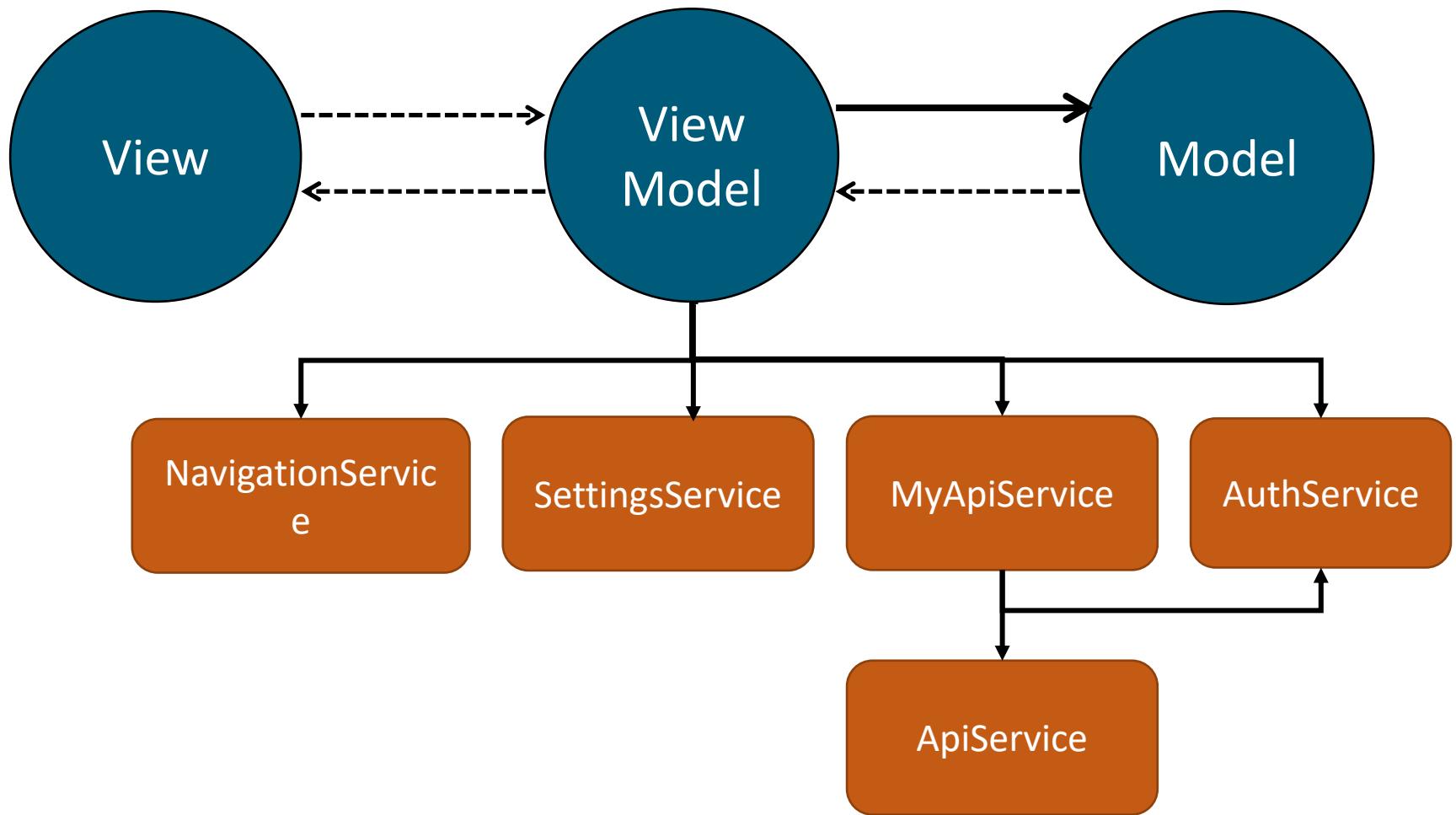
Hány VM legyen?

- ViewModel minden nézethez: OK.
- ViewModel minden UserControlhoz?
 - > Gyakran üres, nincs hozzá tartozó logika
 - > A módosítások eseményekként kerülnek kivezetésre amire a tartalmazó nézethez tartozó VM iratkozik fel
- ViewModel minden domain osztályhoz?
 - > Ld. Strict MVVM
 - > Gyakran csak egyszerű csomagolássá válik
 - > Gyűjtemények/objektum fák esetén hivatkozásoknál kell csomagolni!
 - > Főleg többrétegű alkalmazásnál lehet nehézkes

Szolgáltatások

- Ha csak a ViewModelbe rakkának a teljes üzleti logikát, nem sokban különbözne egy összehányt xaml.cs fájltól
- Szervezzük a logika darabjait szolgáltatás osztályokba
 - > Pl.: Rest API adatelérés, NavigationService stb.
- A ViewModel ezeket felhasználva működik, próbáljuk vékonyan tartani a VM-eket!
- IoC, DI...

Szolgáltatások



MVVM minta hátrányai, korlátai

- Fő hátrány: implementációs overhead
- Amikor nem érdemes használni:
 - > Túl egyszerű alkalmazás
 - > Statikus adatok
 - > Nem hagyományos üzleti alkalmazás
 - > Teljesítménykritikus alkalmazások

MVVM minta előnyei

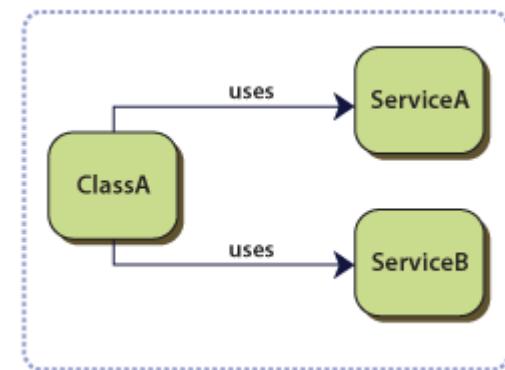
- Áttekinthetőség
- Felelősségek szétválasztása
- Laza csatolás
- Könnyű módosíthatóság
- Tesztelhetőség
 - > Pl. Unit test a ViewModelhez
- Újrafelhasználhatóság
 - > Generikus ViewModel osztályok
 - > ViewModel megosztása appok között (pl. Webshop appok – ugyanaz az üzeleti logika, csak más UI)
 - > Pl. Windows Phone és Windows 8 (Portable Class Library)

MVVM könyvtárak

- Használunk létező könyvtárakat nem kell újra feltalálni a kereket
- Gyakran a Dependency Injection mintát megkövetelik
- Prism, MvvmLightToolkit, Caliburn Micro
- Template10
 - > Nem library hanem inkább alkalmazás sablon

Függőségek kezelése

- Az osztályok nagyon gyakran függenek más osztályuktól/szolgáltatásuktól



- Ez megkötésekkel jár
 - > Függőségek lecserélése, frissítése az osztály módosítását vonja maga után
 - > Fordítás időben elérhetőeknek kell lennie a függőségeknek
 - > Nehéz tesztelni az osztályt, mockolni a függőségeit
 - > **Az osztály felelőssége a függőségeinek példányosítása/konfigurálása**

Függőségek, laza csatolás

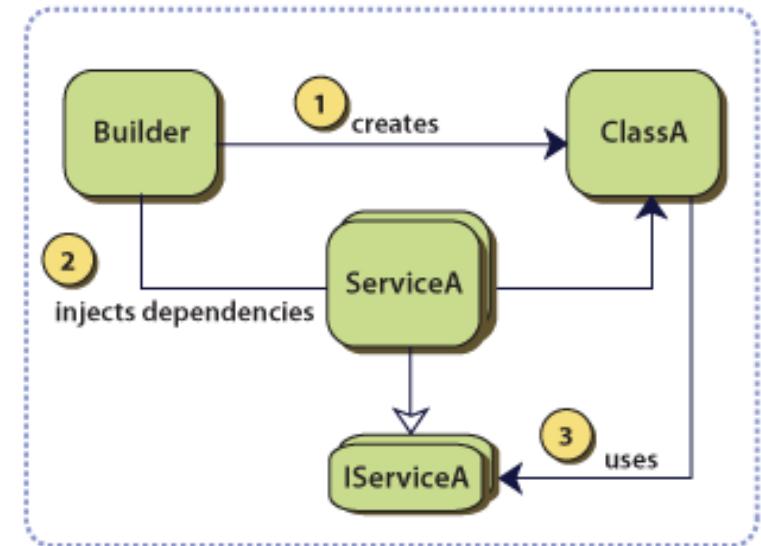
- A kódban felhasznált komponensek konfigurációjának/létrehozásának és felhasználási helyének szétválasztása!
- Az osztály kódjának függetlenítése a felhasznált komponensek *implementációjától*
- Inversion of Control
 - > Factory minták
 - > Service Locator
 - > Dependency Injection

Inversion of Control (IoC)

- Általános fogalom
- Bármire utalhat, amikor a vezérlés megfordul
 - > Esemény vezérelt programozás (Windows)
- Nem a komponens vezényel, hanem Őt használják fel, hívják meg, irányítják
- Változtatják meg a viselkedését ...

Dependency Injection (DI)

- Szoftver tervezési minta: a függőségek átadásra kerülnek a függő objektumba
 - > Az átadás lehet manuális vagy automatikus (injection)
- Laza csatolás
- A minta elemei
 - > A függőség interfésze
 - > A függőség implementációja
 - > A függő/kliens objektum
 - > (injektor objektum)



Interfész

- minden külső függőség funkciót írjuk le egy interfésszel
 - > Explicit (deklaratív) a kapcsolat (szerződés) leírása
 - > Cserélhető az implementáció
- A különböző megoldások az implementáció létrehozásának helyét, konfigurálását mondják meg
 - > Például megkaphatja a konstruktorból...

Injektálás

- Konstruktor alapú
 - > Előnye, hogy jól láthatóak a kötelező függőségek
- Sok paraméter lehet
 - > Áthidalható, ha egyetlen paramétert kap, amiben minden szükséges függőséget átad a hívó – de akkor elveszítjük az deklaratív függőség leírást
- Property alapú
 - > Használható az opcionális függőségek megadására
 - > Nehéz áttekinteni a függőségek listáját
 - > Publikus interfész módosul!
- IoC konténerrel érdemes használni

Injektálás - példa

```
public class MainPageViewModel : ViewModelBase
{
    private readonly INavigationService _navigationService;
    private readonly IDataService _dataService;

    [Dependency]
    public ISomeService SomeService { get; set; }

    public MainPageViewModel(
        INavigationService navigationService,
        IDataService dataService)
    {
        _navigationService = navigationService;
        _dataService = dataService;
    }
}
```

IoC / DI konténer

- Az objektumok létrehozásáért, konfigurálásáért és életciklusáért az alkalmazás felső szintje felel
 - > Ne az alkalmazás alacsony szintjén dőljenek el ezek a kérdések
 - > Például ne az adatelérési réteg határozza meg a naplózó komponenst
- Objektum gráfok létrehozása automatikusan
- Függőségek automatikus, rekurzív kielégítése
 - > Az egyes objektumokat úgy példányosítja, hogy azok függőségeit (konkrét implementációkat) is ő hozza létre
 - > Konstruktor, property injektálás

IoC / DI konténer –példa (Autofac)

```
private Autofac.IContainer container;

private void ConfigureDependencies()
{
    var builder = new Autofac.ContainerBuilder();

    // típus regisztráció interfészhez, minden új példány jön létre
    builder.RegisterType<DataService>().As<IDataService>().InstancePerDependency();
    // típus regisztráció interfészhez, egy példány minden (singleton)
    builder.RegisterType<SomeService>().As<ISomeService>().InstancePerLifetimeScope();
    // saját singleton példány regisztráció interfészhez
    builder.RegisterInstance(NavigationService).As<INavigationService>();

    // típus regisztráció interfész nélkül, szelektált property injektálással
    builder.RegisterType<MainPageViewModel>().InstancePerDependency()
        .PropertiesAutowired((p, o) => p.GetCustomAttributes<DependencyAttribute>().Any());

    container = builder.Build();
}

// használat, függőségek rekurzívan feloldásra kerülnek
container.Resolve<MainPageViewModel>();
```

Service Locator minta

- Egyetlen (singleton) objektum, ami tartalmazza az összes szükséges szolgáltatást
 - > Gyakran az IoC konténer, ami globálisan elérhető
- A logika közvetlenül kéri el a konténertől a függőségeket
- Általában próbáljuk kerülni a használatát
 - > Imperatív megközelítés, a deklaratív ctor/prop injektálással szemben, (kód olvashatóság csökken)
 - > Akkor jó, ha nincs ctor/prop injektálásra lehetőség, például futás idejű paraméterünk van

MVVM és DI közösen

- ViewModel-ek a szolgáltatás interfészekre hivatkoznak
- Ezekből egy-egy példányt függőség injektálással kérnek el
- A VM még inkább újrafelhasználható lesz
 - > Nem függ platformspecifikus implementációtól
 - Pl.: Xamarin, UWP átjárás
- A legtöbb MVVM keretrendszer elvárja és támogatja a DI minta alkalmazását

ViewModel példányosítása 1

- De ki példányosítja a ViewModelt?
 - > Több megoldás is elterjedt
- Egy központi NavigationService osztályok keresztül navigálunk új nézetekre (Page)
- Sikeres navigáció esetén a NavigationService fogja az IoC konténerből elkérni a ViewModelt a függőségeivel (DI)
- Az elkért ViewModelt a NavigationService állítja a nézet DataContextjére vagy egy típusos propertyre (x:Bind)

ViewModel példányosítása 2

- ServiceLocator mintával XAML-ben adatkötéssel
- ```
<Page ...
 DataContext="{Binding MainPage,
 Source={StaticResource ViewModelLocator}}" />
```
- A ViewModelLocator egy alkalmazás szintű erőforrás könyvtárban található objektum
  - Tulajdonságokat publikál a különböző lehetséges ViewModel típusokkal
    - A tulajdonságok gettere a konténerből példányosítja a ViewModel-t
  - A visszaadott objektumot adatkötéssel állítjuk be az oldal DataContextjére

# ViewModel példányosítása 3

- Attached Propertyvel módosítjuk a View-t
- Az attached property értékkadásának „mellékhatásaként” fut le
  - > Az AP elkéri a ViewModel példányt a konténerből
  - > Majd beállítja az AP-hez tartozó objektum DataContextjére
- Gyakran névkonvenciót használ
  - > MainPage → MainPageViewModel

<Page...

`mvvm:ViewModelLocator.AutoWireViewModel="True" />`

# x:Bind és az MVVM (ismétlés)

- Hogy használhassuk az x:Bind adatkötést szükségünk van a code-behindban egy segéd tulajdonságra, amihez köthetünk

```
public MainPageViewModel ViewModel =>
 (MainPageViewModel)DataContext;
```

Text="{x:Bind ViewModel.Text}"

# MVVM és DI közösen

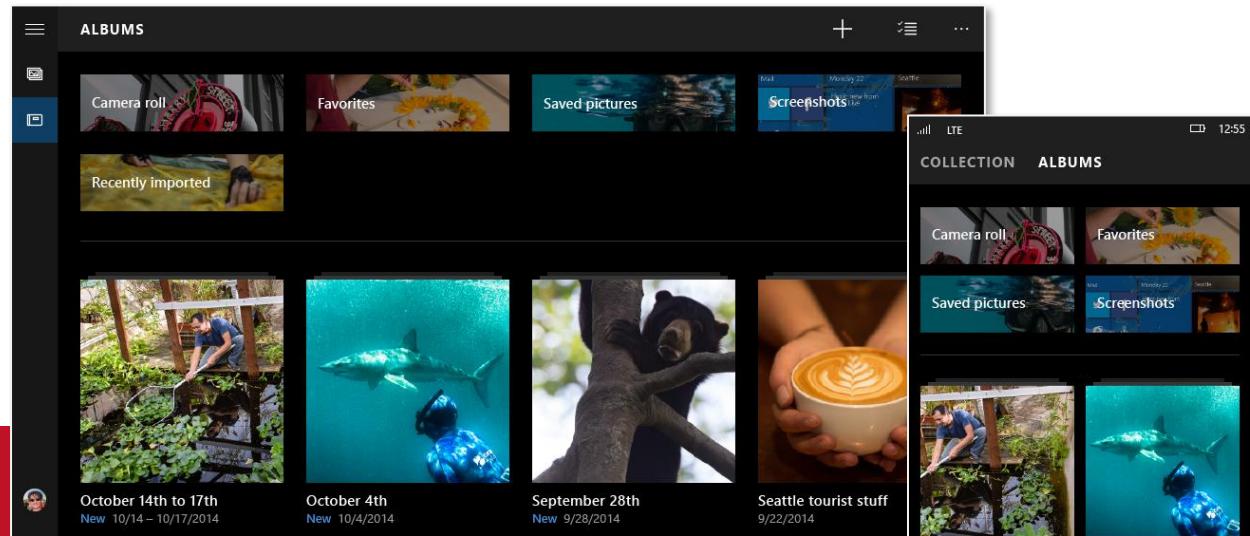
- Előnyök
  - > Vezeti a fejlesztő(k) kezét és szemét
  - > Lazán csatolt architektúra
  - > Tesztelhetőség, hordozhatóság, karbantarthatóság növelése
  - > SOLID elvekhez konvergáló kód
- Hátrányok
  - > Fejlesztési overhead
- Elterjedt IoC könyvtárak:
  - > Autofac, Unity, Ninject, SimpleIoC, stb.

# Adaptivitás

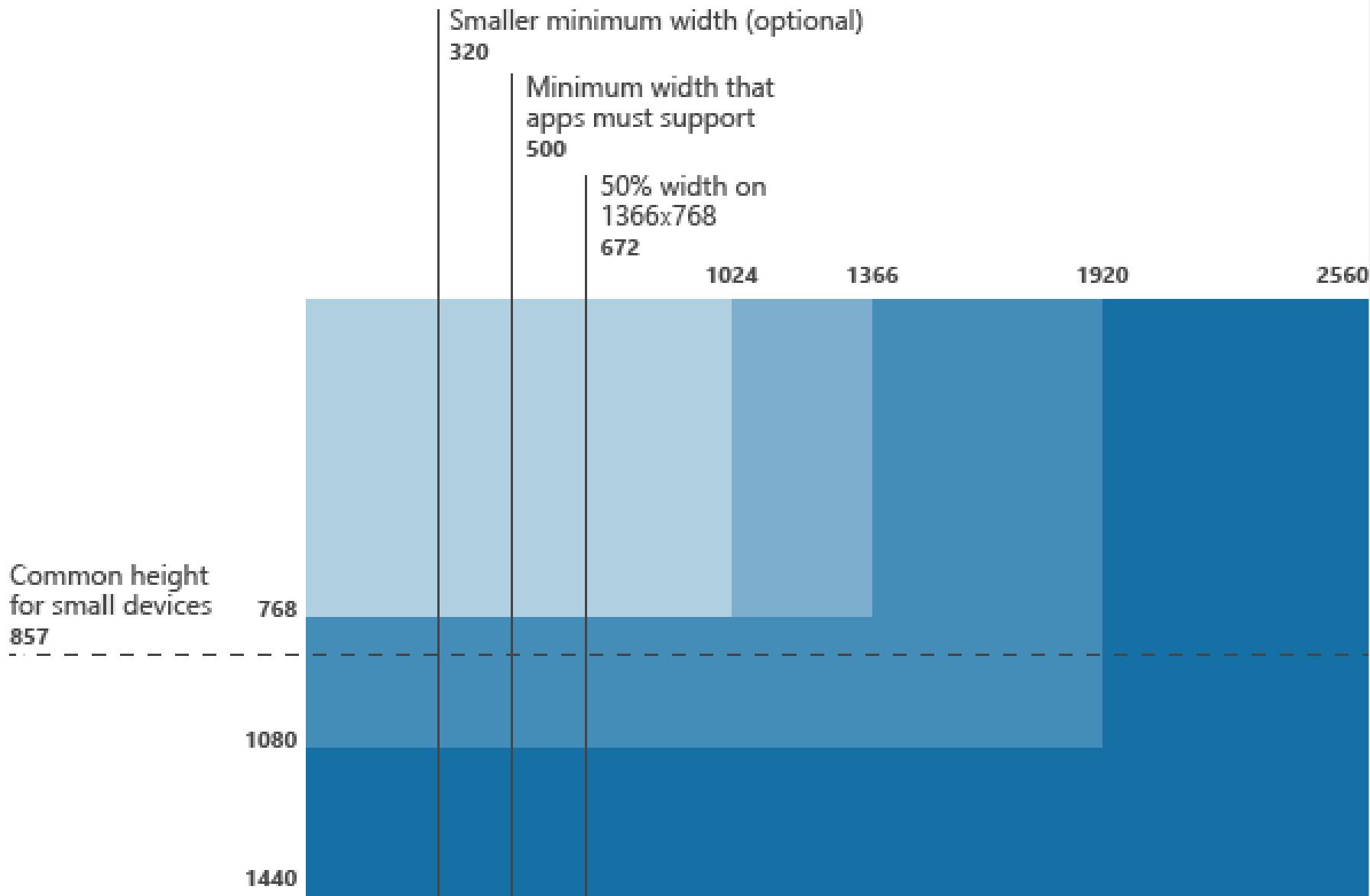
- Képernyő méret, pixel sűrűség



- Felhasználói felület arányai



# Több képernyőméret és -arány támogatása

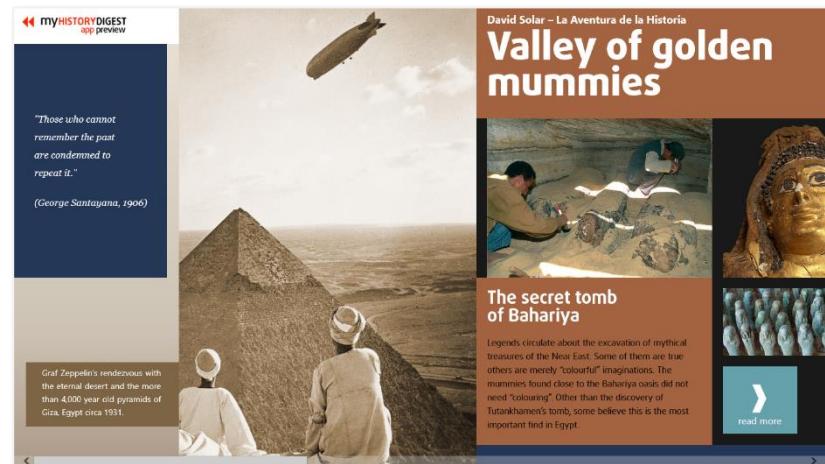


# Fix layout

- Pixelpontos tervezés (PSD)
- ViewBox, Canvas, Grid
- minden képernyőmáreten pontosan ugyanaz látszik



- Előnye:
  - > Egyszerű
  - > A design pontosan le másolható
- Hátránya:
  - > Nem adaptív - arányok
  - > Változó szövegméret



# Adaptív layout

- Nagyobb képernyőn több információ látszik
- Előnye:
  - > Kényelmes mennyiségű tartalom
  - > Megfelelő szövegméret
  - > Pixelpontos képméret
- Hátránya:
  - > Adaptívra kell megtervezni a felületet



11.6" 1366x768



30" 2560x1600

# Pixelsűrűség

- Ha fizikai pixelekben számolunk:
  - > Minél nagyobb a pixelsűrűség, annál kisebb a tartalom
  - > Pl: egy elem 100 pixel széles



11.6" 1366x768



11.6" 1920x1080

# Pixelsűrűség

- Nőjön a tartalom is a pixelsűrűséggel!
- Azonos méretű eszközön, ugyanakkora tartalom
- Pl: egy listaelem 100 pixel / 140 pixel széles



11.6" 1366x768



11.6" 1920x1080

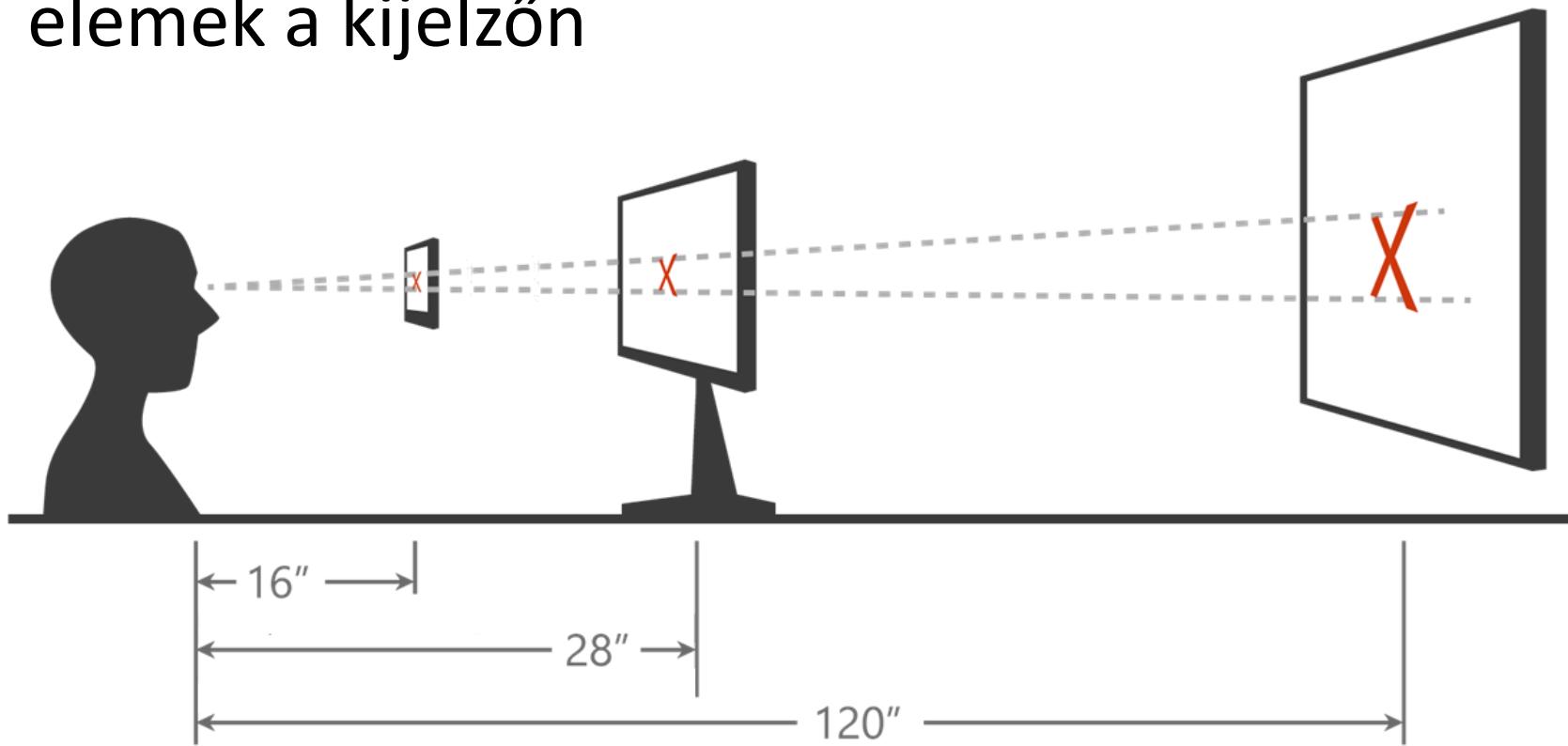
# Windows 10: még szélesebb eszközpaletta

- A képernyőméret és a fizikai felbontás mellett a *szemtől való tipikus távolság* is szemponttá válik



# Windows 10: még szélesebb eszközpaletta

- A skálázási algoritmus azt próbálja elérni, hogy *ugyanakkora a látószöget foglaljanak el az elemek a kijelzőn*



# Hogyan?

- $100\% = 96\text{DPI}$ -s eszköz, kb. 70 cm-re a szemtől
  - > A DPI és a távolság egyaránt növeli (továbbra is sávosan)
- Pár példa:

| Display size, device type, and aspect ratio | Physical resolution and pixel density | Effective resolution and scale factor | Visual size of an effective pixel |
|---------------------------------------------|---------------------------------------|---------------------------------------|-----------------------------------|
| 4" phone; 5x3                               | 480x800 & 233 DPI                     | <b>320x533</b> at 150%                | 0.020°                            |
| 6" phone; 9x16                              | 1440x2560 & 489 DPI                   | <b>411x731</b> at 350%                | 0.022°                            |
| 12" 2 in 1; 3x2                             | 2160x1440 & 216 DPI                   | <b>1440x960</b> at 150%               | 0.020°                            |
| 24" 4K desktop; 16x9                        | 3840x2160 & 183 DPI                   | <b>1920x1080</b> at 200%              | 0.022°                            |
| 84" 4K wall; 16x9                           | 3840x2160 & 53 DPI                    | <b>2560x1440</b> at 200%              | 0.018°                            |

# Mennyire tipikusak az egyes szintek?

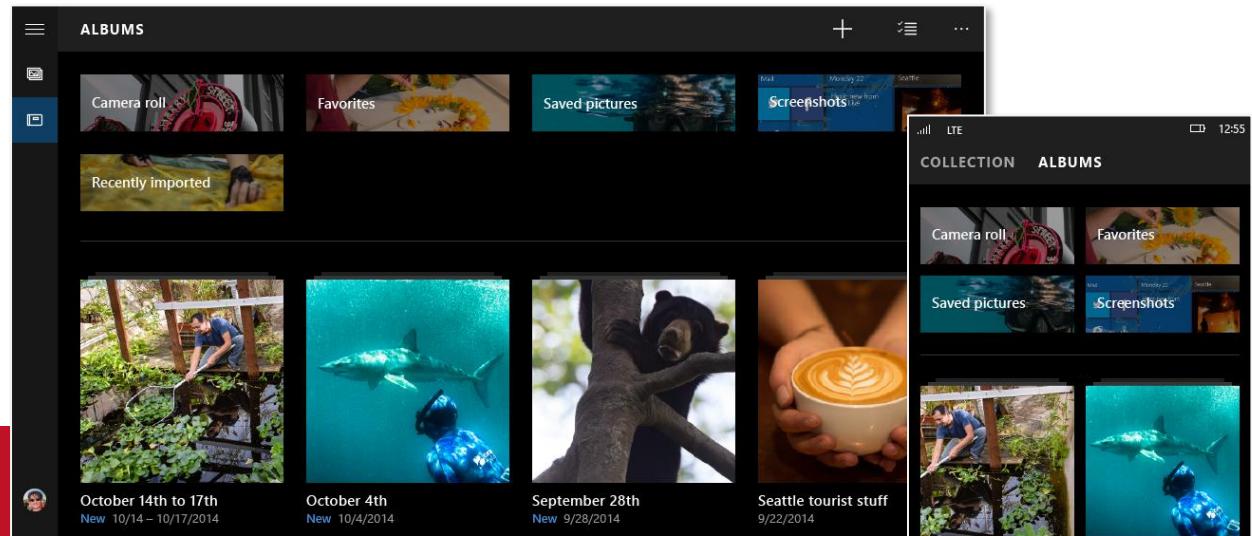
| Scale Factor | Importance                                                                  | Examples                                                             |
|--------------|-----------------------------------------------------------------------------|----------------------------------------------------------------------|
| 100%         | Most PC upgrades and low-cost laptops<br>Some low-cost tablets<br>No phones | 10-12.5" 1366x768 laptop<br>30" 2560x1440 monitor                    |
| 125%         | Some PC upgrades and low-cost laptops<br>Many low-cost phones               | 13.3" 1600x900 laptop<br>27" 3200x1800 monitor<br>4.5" 480x800 phone |
| 150%         | New mainstream laptops<br>Many low-cost phones                              | 10.6-13.3" 1080p laptop<br>28" 4K monitor<br>5" 960x540 phone        |
| 200%         | New premium laptops<br>Many mid-range phones                                | 13.3" 2560x1440 laptop<br>24" 4K monitor<br>5" 1280x720 phone        |
| 250%         | New high end, some 4K laptops<br>Some mid-range phones                      | 13.3" 3200x800 laptop<br>15.6" 4K laptop<br>5.7" 1080p phone         |
| 300%         | New 4K laptops<br>High-end phones                                           | 13.3" 4K laptop<br>4.5" 1080p phone                                  |
| 400%         | Premium phones                                                              | 5.2" 2560x1440 phone                                                 |

# Adaptivitás

- Képernyő méret, pixel sűrűség



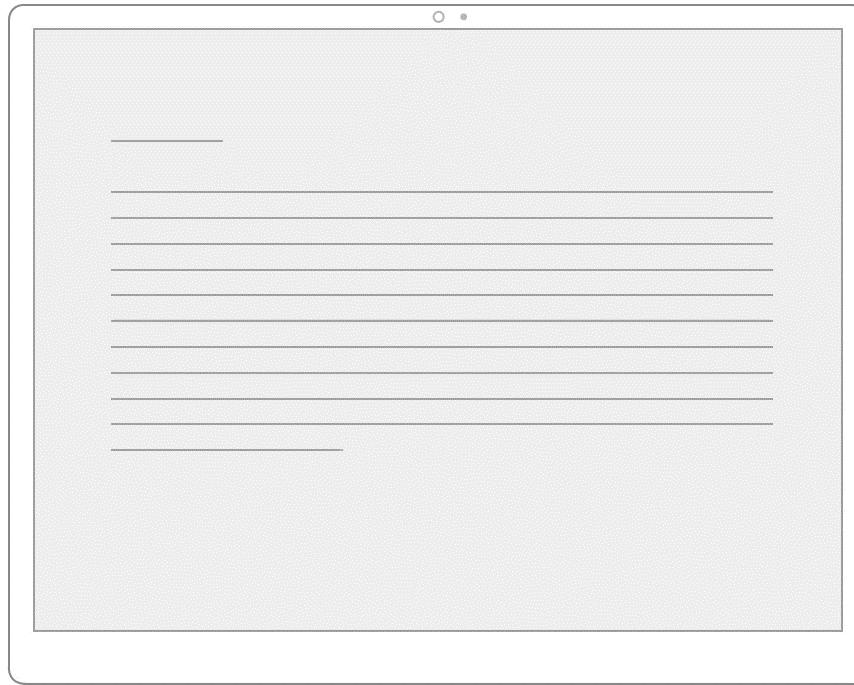
- Felhasználói felület arányai



# Adaptivitási stratégiák

- „6 R”
  1. Resize (átméretezés)
  2. Reflow (áttördelés)
  3. Reposition (áthelyezés)
  4. Reveal (felfedés)
  5. Re-architect (átrendezés)
  6. Replace (lecsérélés)

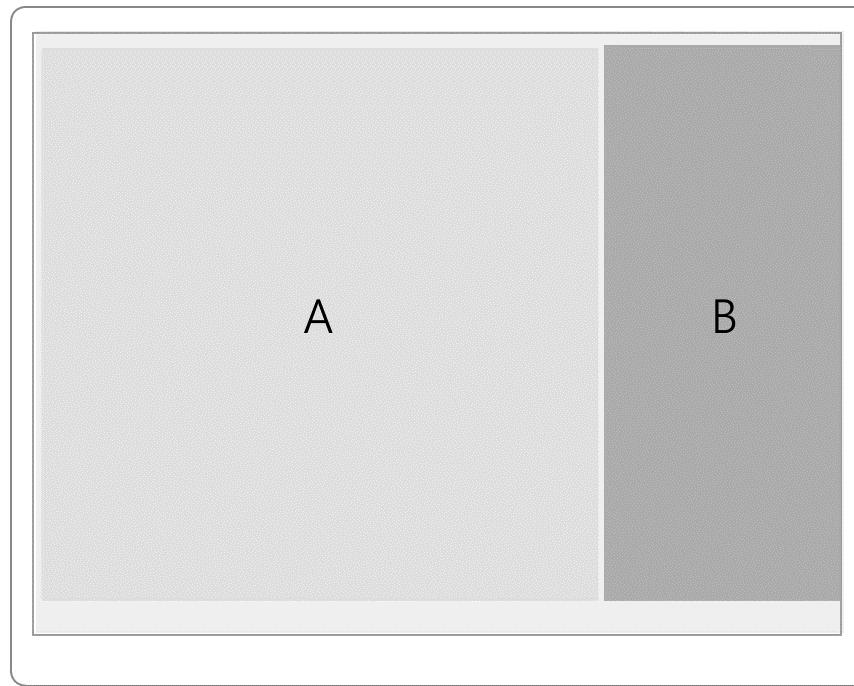
# 1. Resize



## 2. Reflow



# 3. Reposition

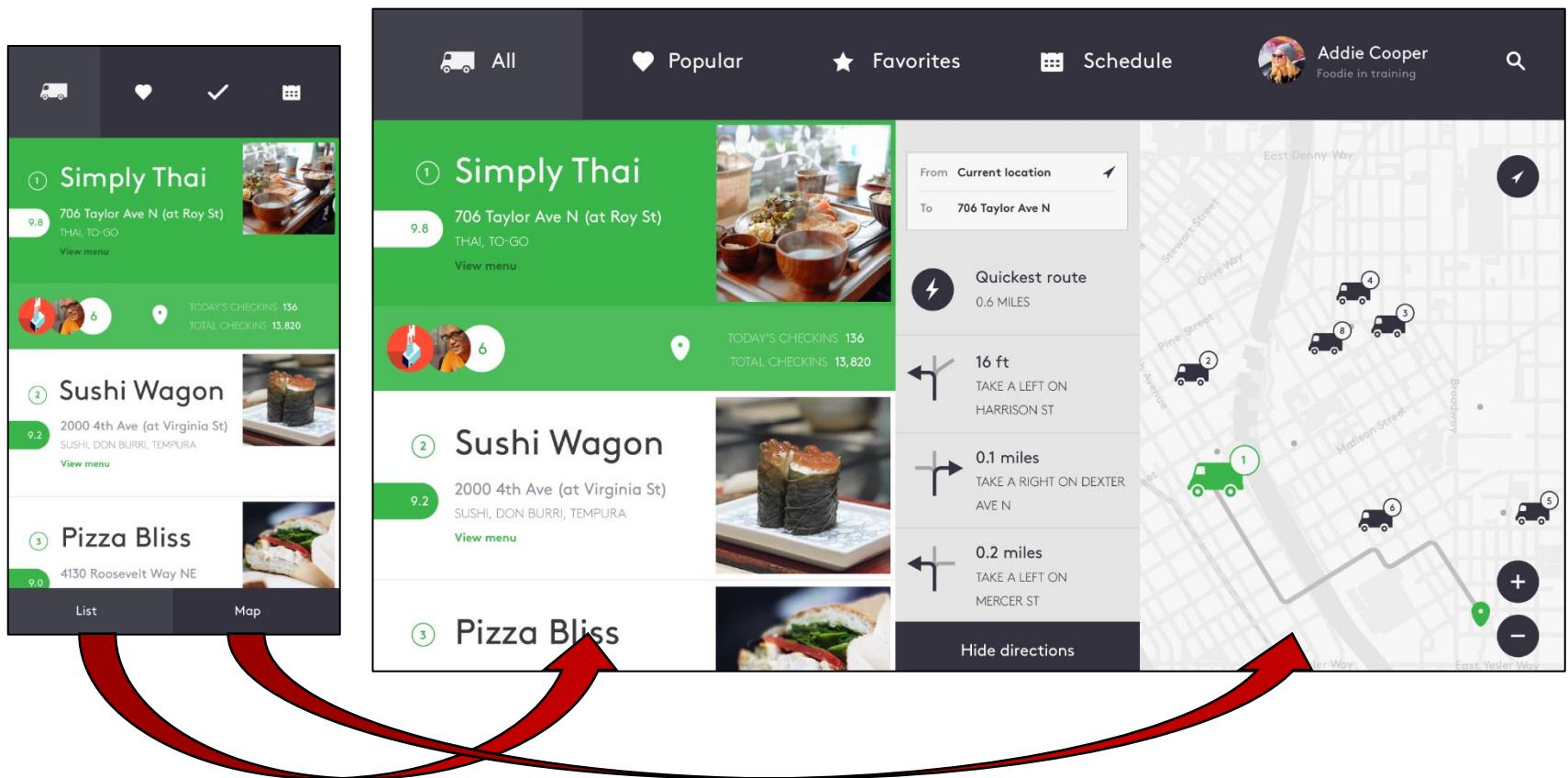


# Reposition pélða

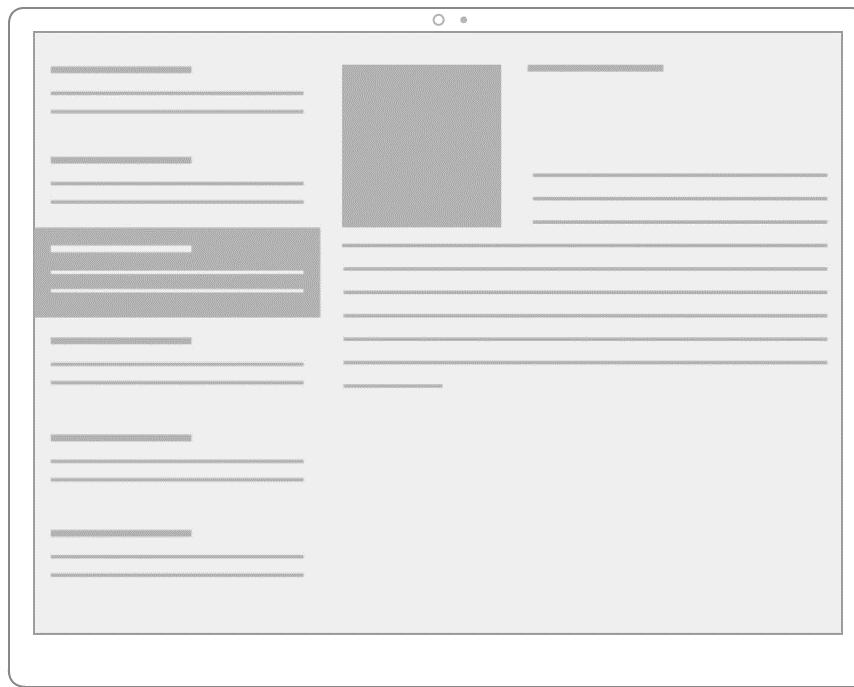
The screenshot shows the Windows Store interface. At the top, there's a search bar and navigation links for Home, Apps, Games, Music, and Movies & TV. Below that is the app page for "Evernote Touch" by Evernote. The page includes a green icon with a white elephant logo, the app name, its rating (4.5 stars from 3,548 reviews), and a "Share" button. A "Phone + Tablet + PC" badge indicates it's a universal app. The description states: "Evernote is an easy-to-use, free app that helps you remember everything across all of your devices. You can create a plaintext note on your Windows tablet, and then open it on your smartphone or any other". A "More" link is present. At the bottom, there's a "Free" badge and a "Screenshots" section showing PC and mobile device screenshots.

The screenshot shows the same Windows Store page for "Evernote Touch", but with a large red arrow pointing from the left side towards the "Screenshots" section. The page layout is identical to the first one, featuring the app icon, name, rating, share options, and a detailed description. The "Free" badge is at the bottom. The "Screenshots" section displays images of the app's interface on both PC and mobile devices. Below the screenshots, there's a "Ratings and reviews" section with a 4.2 rating from 18,654 reviews, and a "Features" section listing various sync and organization features.

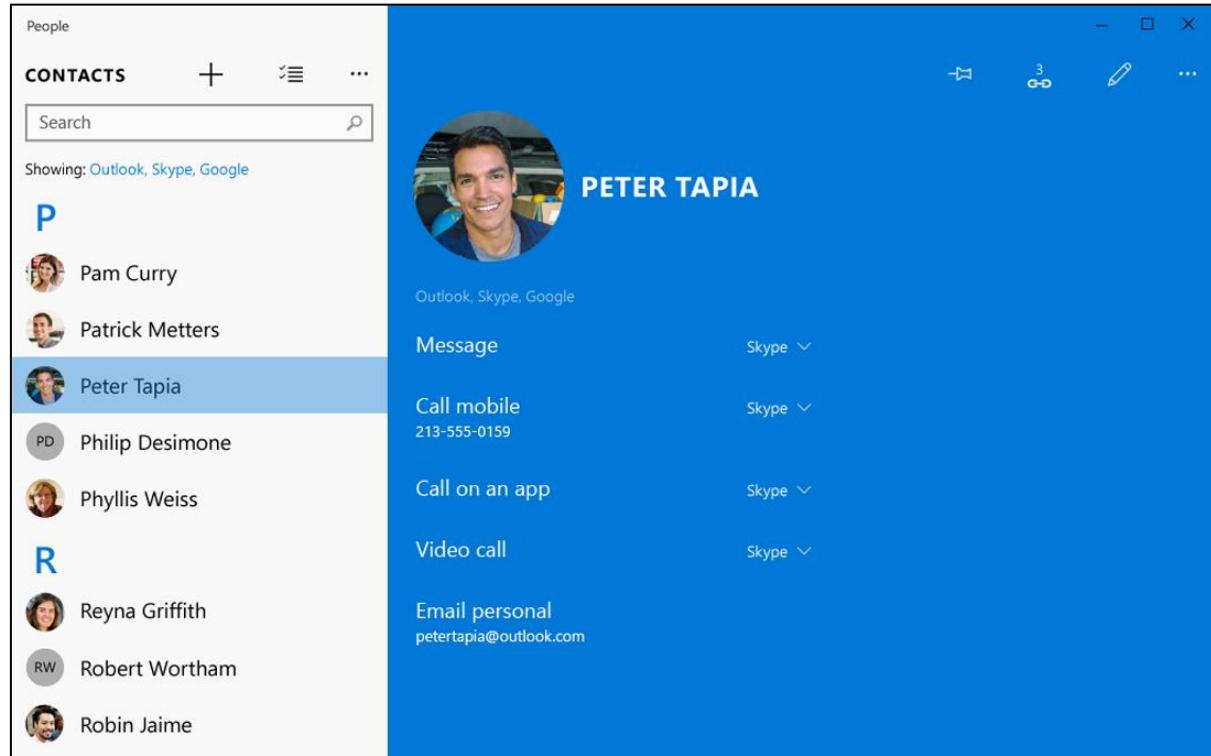
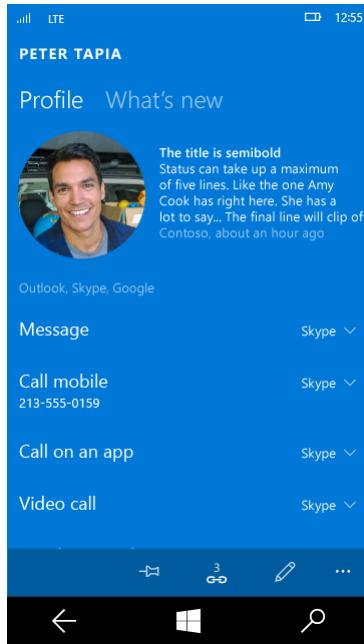
# 4. Reveal



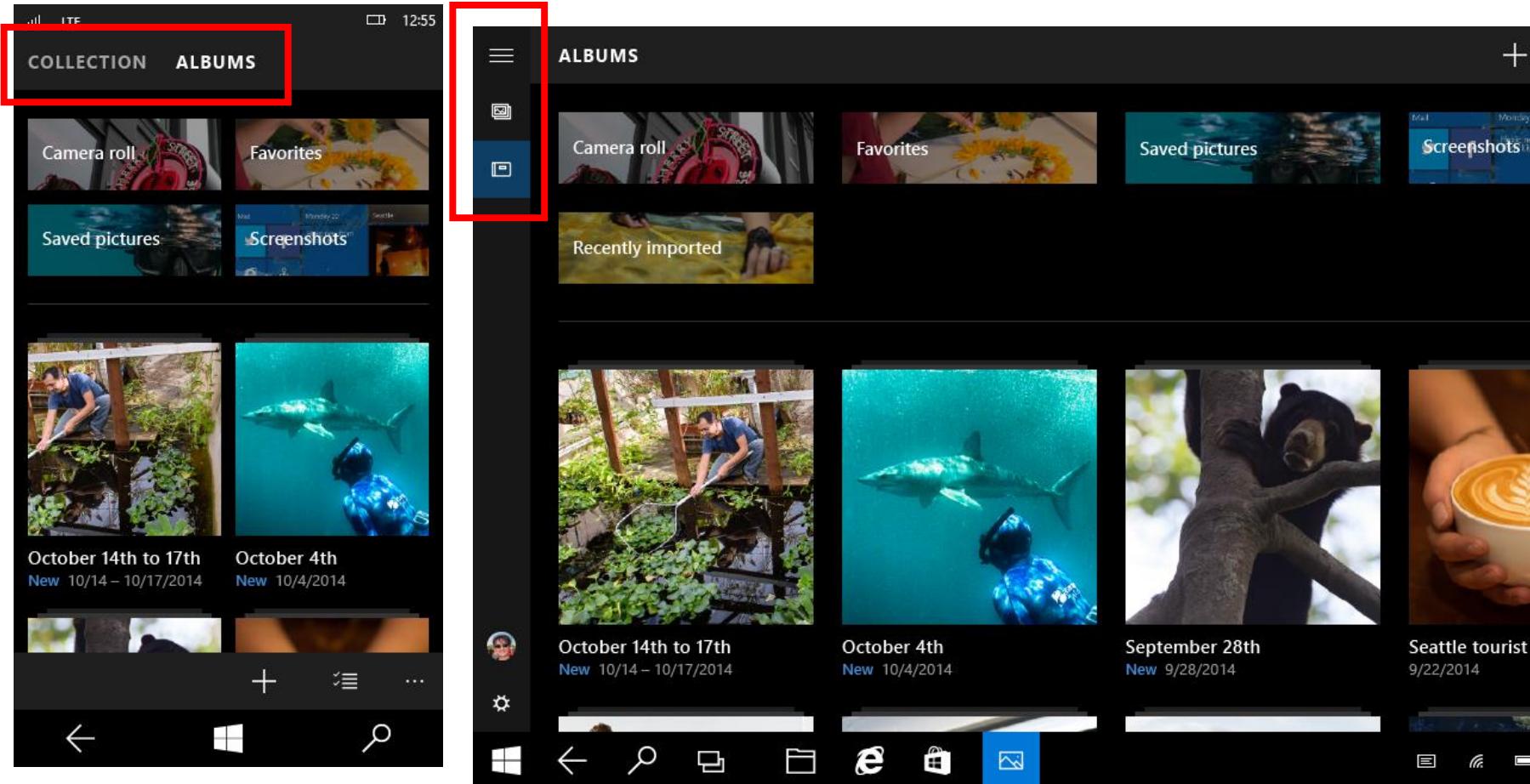
# 5. Re-architect



# Re-architect példa



# 6. Replace



# Adaptivitás megvalósítása

- Tipikusan az összes layoutot egyetlen Page-en definiáljuk
  - > A különböző vezérlőket ki/be kapcsoljuk
  - > Váltogatunk az adatsablonok között
- Vizuális állapotokkal – VisualStateManager
  - > „Animációkkal” váltunk állapotot
  - > Visibility tulajdonság
  - > ItemTemplate tulajdonság

# Az állapotváltás is lehet deklaratív

- Windows 10-től
  - > Mostantól nem kell az állapotváltás kedvéért feliratkozni a SizeChanged-re

```
<VisualStateGroup x:Name="WindowSizeStates">
 <VisualState x:Name="WideState">
 <VisualState.StateTriggers>
 <AdaptiveTrigger MinWindowWidth="720" />
 </VisualState.StateTriggers>
 <VisualState.Setters>
 <Setter Target="MySplitView.IsPaneOpen" Value="True" />
 <Setter Target="MySplitView.DisplayMode" Value="Inline"/>
 </VisualState.Setters>
 </VisualState>
</VisualStateGroup>
```

# Adaptációs stratégia

- A „MinWindowWidth” property: „ha az ablak szélesebbé válik 720-nál, aktiváld ezt az állapotot, ha pedig keskenyebbé, akkor deaktiváld.”
1. Tehát alapból a lehető legkeskenyebb képernyőre (ablakméretre) írjuk meg a XAML-t
  2. A képernyő fokozatos „szélesedését” VisualState-ekkel kezeljük le – pl. úgy, hogy elkezdünk paneleket bekapcsolgatni (Reveal stratégia).

# Az állapotváltás is lehet deklaratív

- Nem csak képernyőméretre tud reagálni
  - > Pontosabban *alapból* csak arra, de kiterjeszhető!
- A StateTriggerBase-ből kell leszármazni
  - > Feliratkozunk a szükséges eseményekre
  - > Saját propertyket definiálunk a konfiguráláshoz (mint az AdaptiveTriggernél a MinWindowWidth)
  - > Az egész nagyon hasonlít a Behaviors SDK-hoz
- Rengeteg (nem hivatalos) kiegészítő Trigger letölthető már

# Néhány példa letölthető triggerre

<https://github.com/dotMorten/WindowsStateTriggers>

- **DeviceFamilyStateTrigger**
  - > Trigger based on the device family (Desktop, Mobile, IoT or Team)
- **NetworkConnectionStateTrigger**
  - > Trigger if internet connection is available or not
- **OrientationStateTrigger**
  - > Trigger based on portrait/landscape mode
- **IsNullOrEmptyStateTrigger**
  - > Trigger if an object is null, or if a String or IEnumerable is empty
- **IsTypePresentStateTrigger**
  - > Trigger if a type is present (ie hardware backbutton etc)
- **InputTypeTrigger**
  - > Trigger based on the PointerType you're using on the TargetElement
- **RegexStateTrigger**
  - > This trigger evaluates a regular expression against a string and triggers if a match is found.
- **CompositeStateTrigger**
  - > This trigger combines other triggers using, And, Or or Xor to create even more powerful triggers.

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

# Xamarin 1+2

< Albert István >



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Tartalom

## Mi a Xamarin?

- Technológia, amivel minden nagyobb mobil platform elérhető
  - > Natív felhasználói felület
  - > Natív teljesítmény
  - > Közös kód a platformok között
- > C# & .NET 5



Xamarin 1

## iOS / Objective-C specialitások

- Target-action minta – .NET-es események
- Protokoll – interfész opcionális metódusokkal
- Message – „metódus hívás”
- Delegate – „delegáció” tervezési minta, NEM a .NET-es delegate-re utal (metódus referencia)

Xamarin 1

## INotifyPropertyChanged interfész

- A XAML-ben szokás INPC interfész
- Itt is MVVM-et használunk
- Itt is vannak keretrendszerek!

Xamarin 1

## Alapfogalmak

- Xamarin
- Mono
- Xamarin.Android
- Xamarin.iOS
- Xamarin.Mac
- C# de a szokásos Android és iOS architektúra
  - > MVP és MVC
  - > minden platform specifikus API elérhető
- Xamarin.Forms



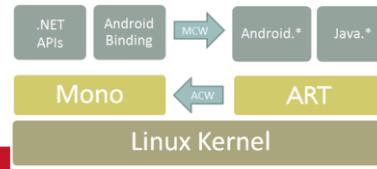
Aufsoft

25

Xamarin 1

## Xamarin . Android

- Egy alkalmazást két futtatókörnyezet szolgál ki
- Mono
  - > A CLR és HAL C-ben van, a kernelt hívja
  - > Az osztálykönyvtár C#, így érhető el a fájlrendszer stb.
  - > A platform funkciók jelentős része csak az Android Runtime Java API-n érhető el – csomagoló osztályok



Aufsoft

Mono

ART

Linux Kernel

## Xamarin.Forms

- Nyílt forráskód
- Elsősorban platform független UI megvalósítására
- UI definíálása XAML-lel vagy C#-ban
- Kiterjeszthető, az egyedi igények szerint
- Mikor válasszuk?



Xamarin.Forms

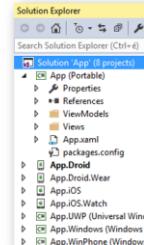


Xamarin.iOS, Xamarin.Android

Xamarin 1

## Xamarin projekstruktúra

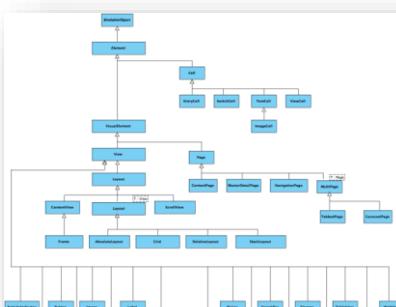
- .NET Standard / Shared / PCL
  - > Teljes egészében platform független kód
  - > Az alkalmazás core része
- Platform specifikus projektek
  - > Android, iOS, UWP
  - > Jellegzetes platform specifikus kód megosztások
    - UWP
    - iOS, Droid



Aufsoft

51

Xamarin 1



Aufsoft

59

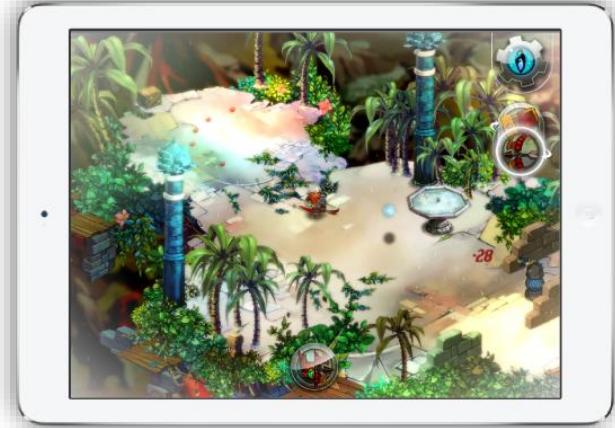
Xamarin 1

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

# Mi a Xamarin?

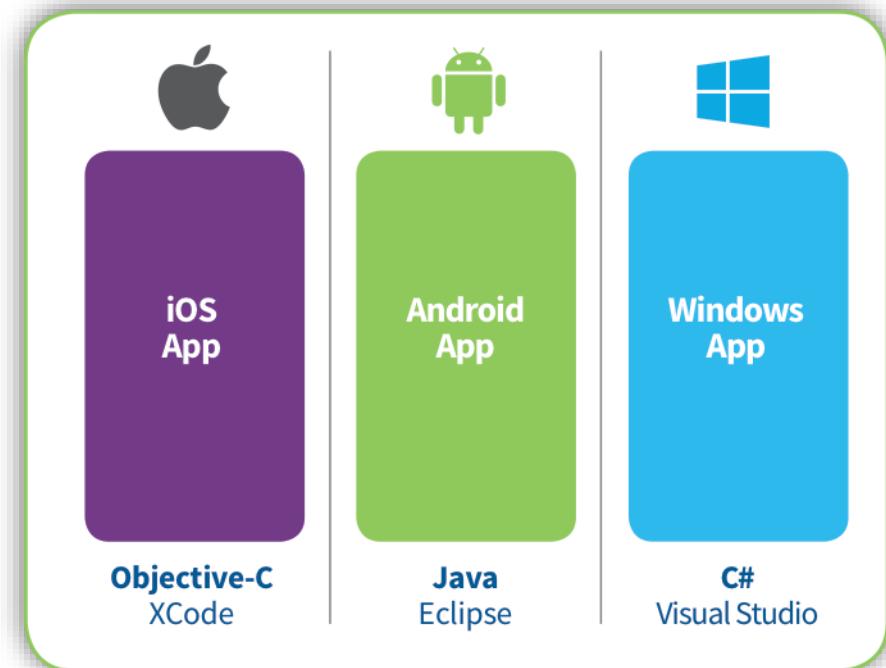
- Technológia, amivel minden nagyobb mobil platform elérhető
  - > Natív felhasználói felület
  - > Natív teljesítmény
  - > Közös kód a platformok között
  - > C# & .NET 5



# Natív silós megközelítés

## Több natív alkalmazást készítünk

- Több csapat
- Több kód bázis
- Különböző eszközök
- Fejlesztés
- Karbantartás
- HR menedzsment



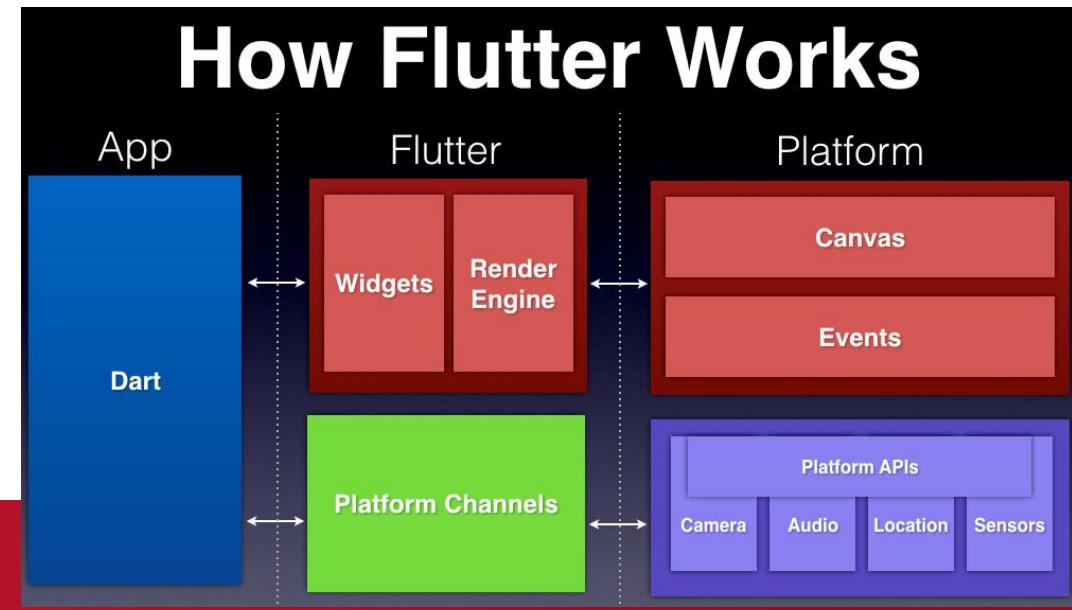
# Webes megközelítés

- Legkisebb közös többszörös
- Böngésző töredézettség
- Valójában egy platformra tervezük és végül több platformon futtatjuk
- Korlátozott felhasználói és fejlesztői élmény, alaprendszer



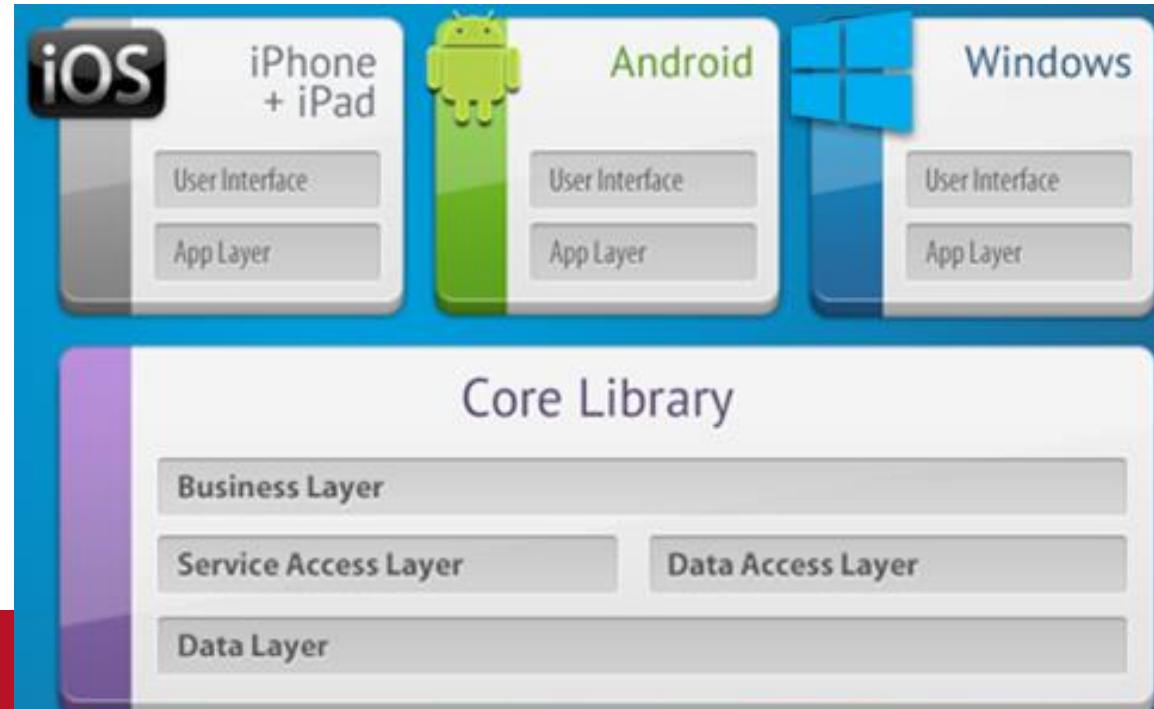
# Flutter – egy másik cross-platform technológia

- Dart programozási nyelv
- Nem natív UI elemek használata
  - > minden platformon ugyanúgy néz ki az app!
  - > nincs „legkisebb közös többszörös” probléma
- Teljes natív API elérés

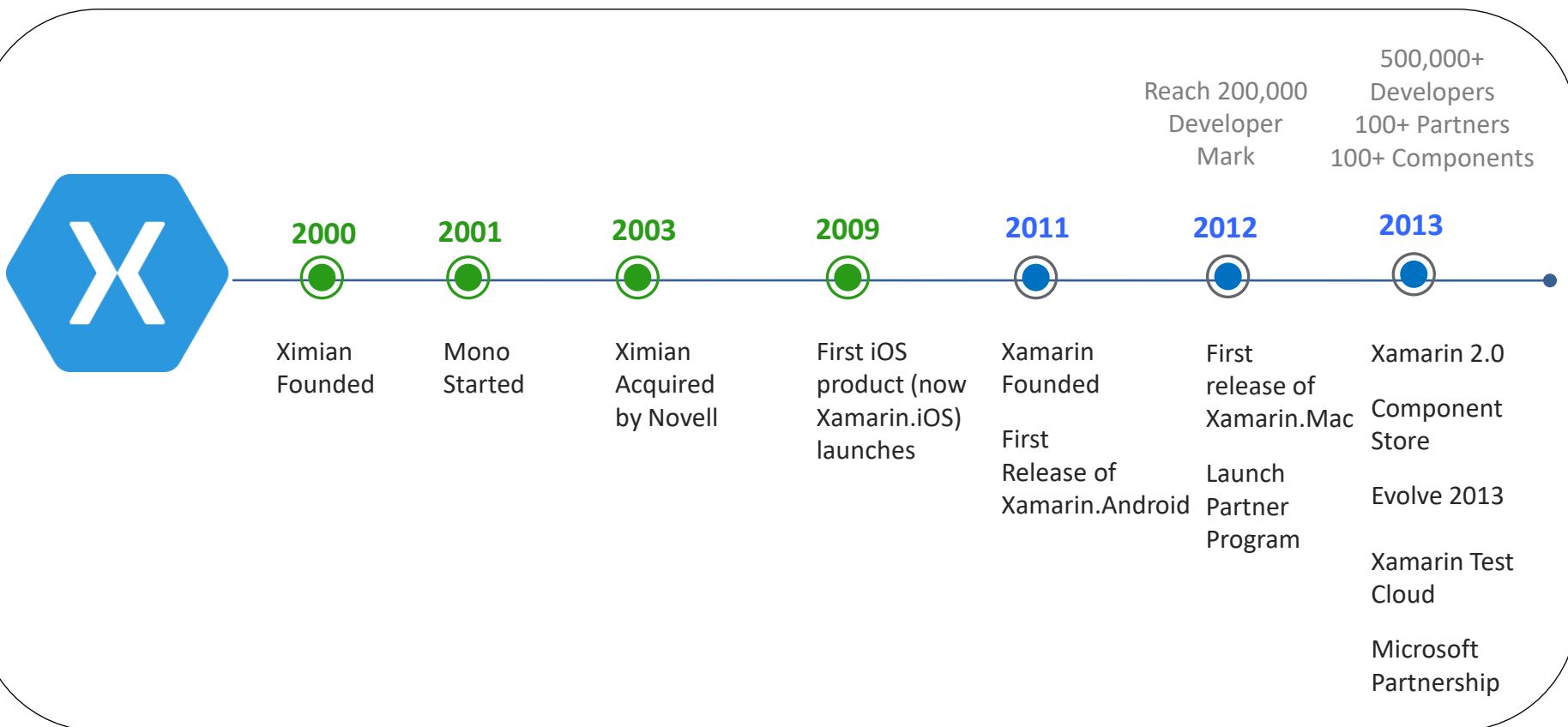


# Xamarin megközelítés

- Natív felhasználói felület
- Közel natív teljesítmény
- Közös kód a platformok között
- C# & .NET 5
- Teljes API elérés



# Xamarin történelem

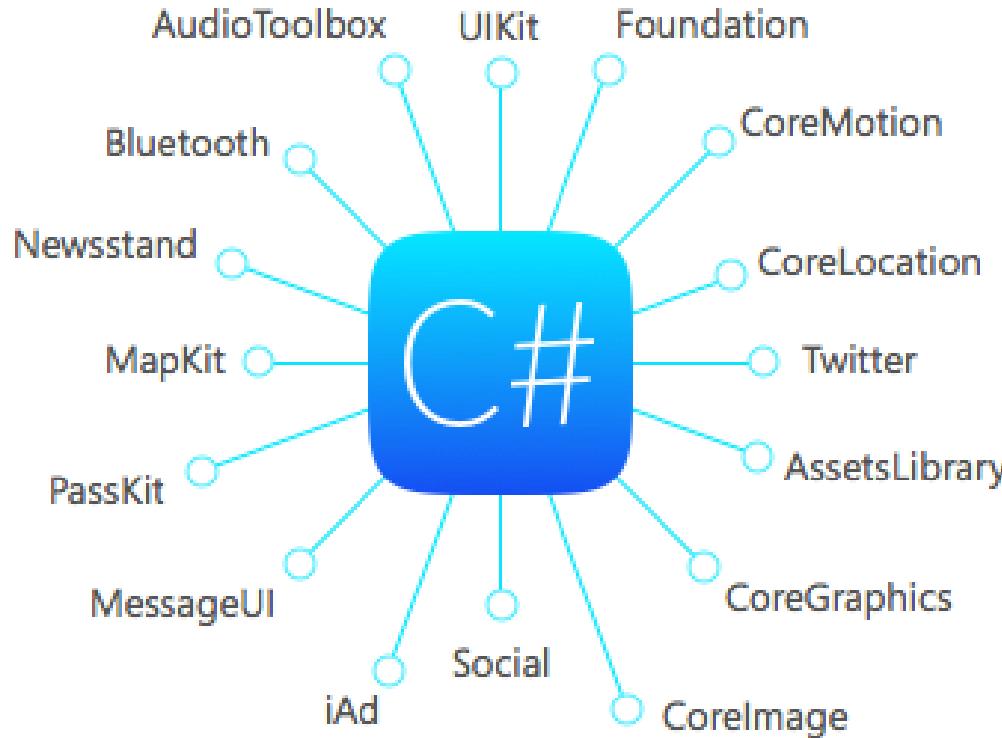


# MAUI

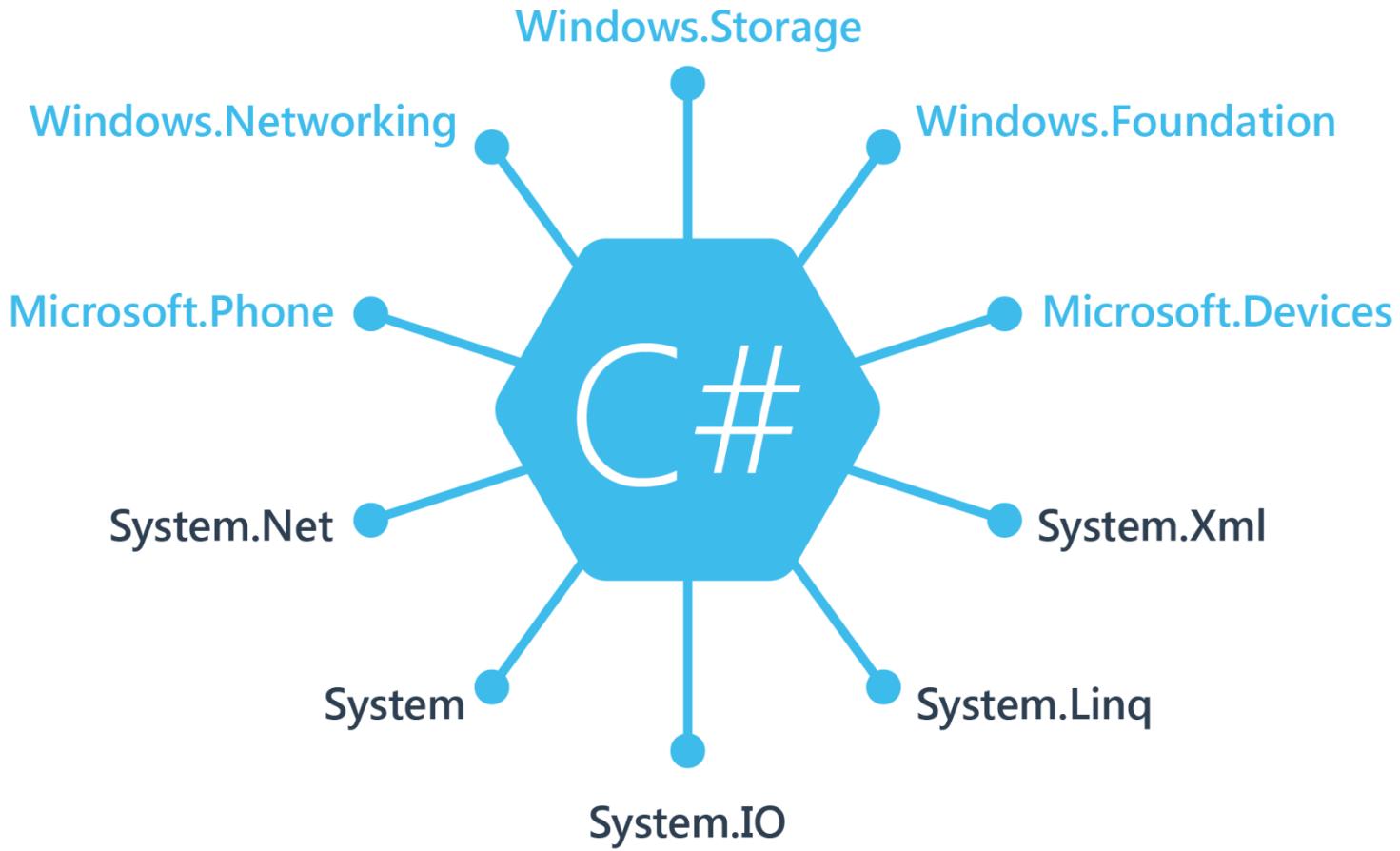
- .NET Multi-platform App UI
- .NET 6-ra épül
  - > Natív API elérés + közel natív teljesítmény
- Xamarin és Xamarin.Forms utódja
  - > Újraírt vezérlő koncepció
  - > *Natív vezérlők*
- Mobil és desktop
  - > WinUI -> Windows / Mac Catalyst is
- Egyetlen projekt fájl

# 100%-os API lefedettség

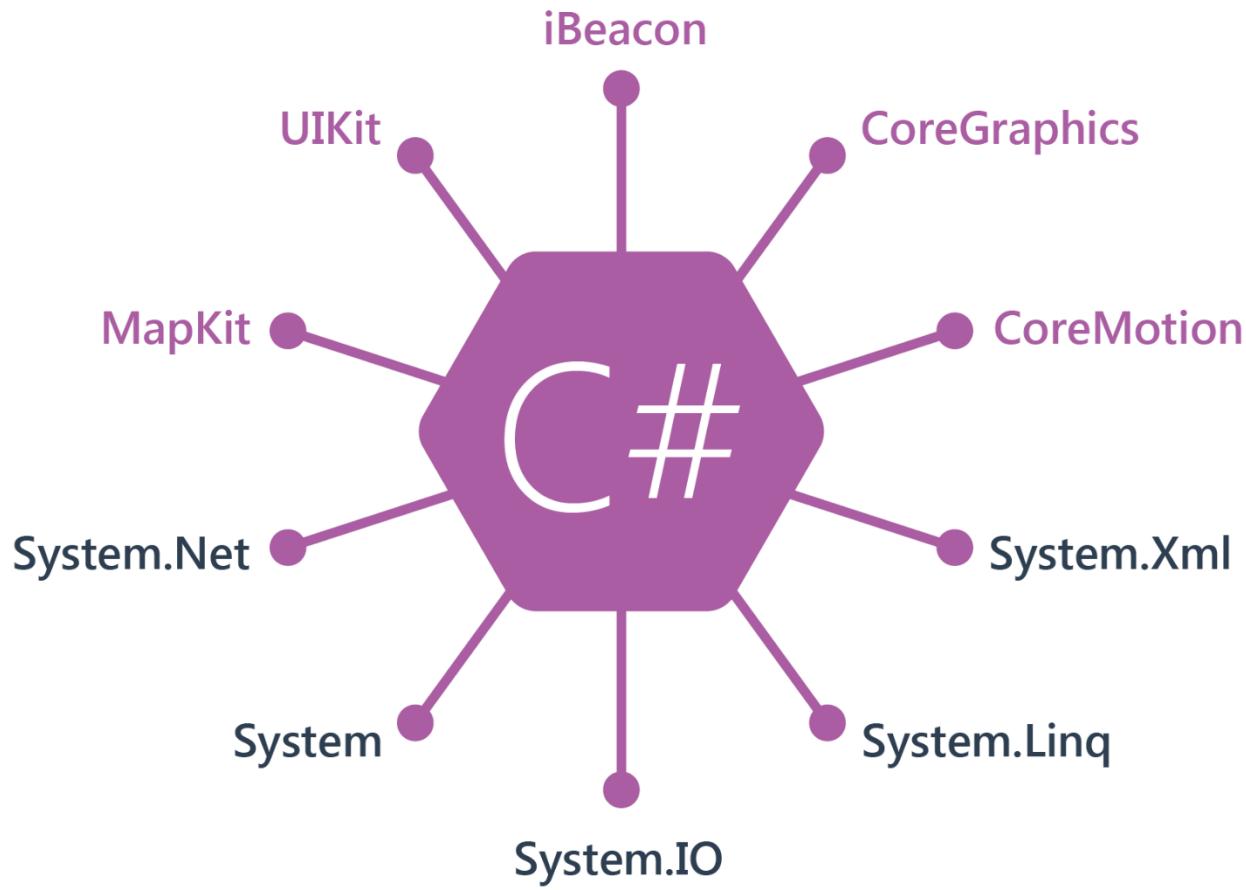
- Bármi, amit meg tudsz írni Swiftben/Objective-C-ben, Kotlinban/Javaban, megírhatod C#-ban, Visual Studio-val és Xamarin-nal!



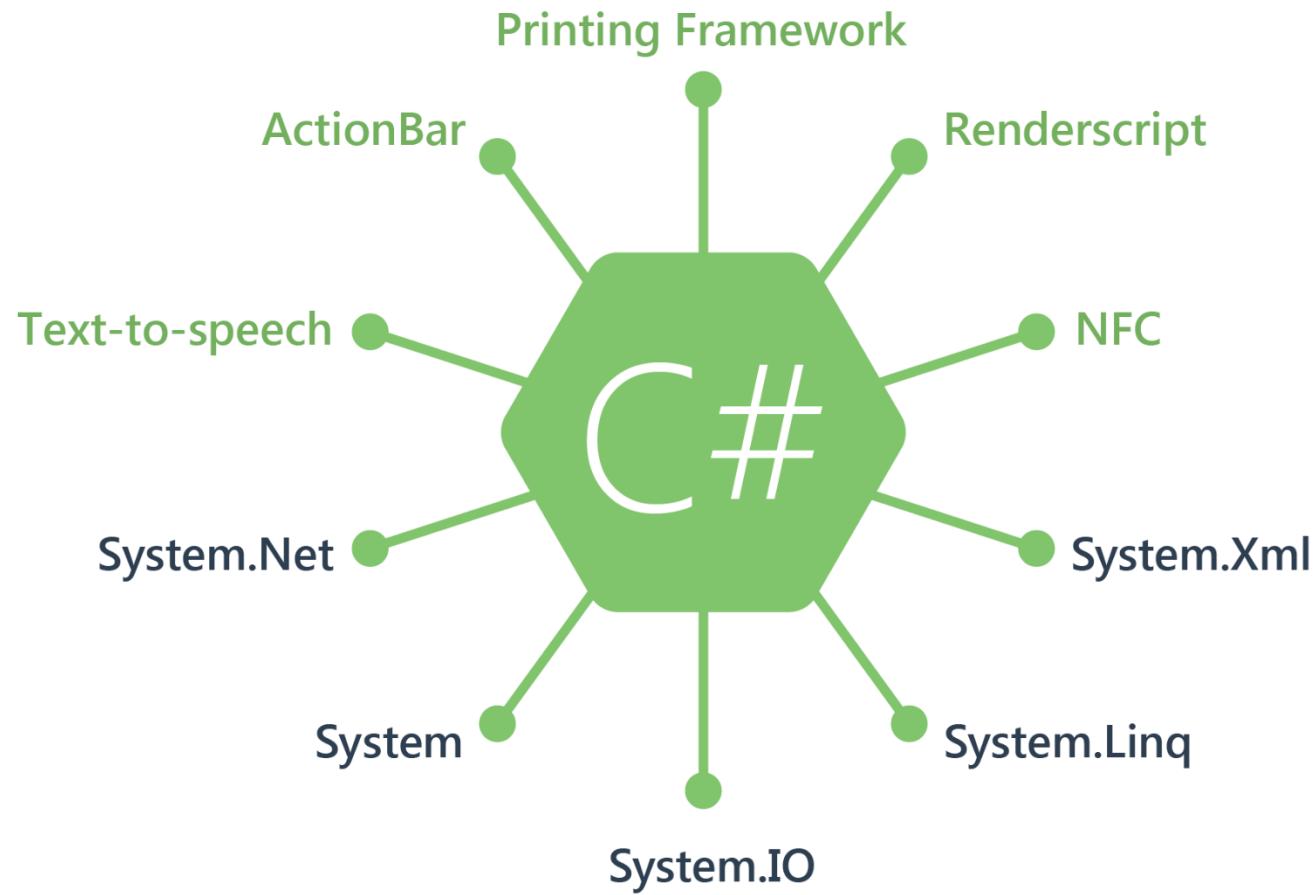
# Windows APIs



# iOS APIs

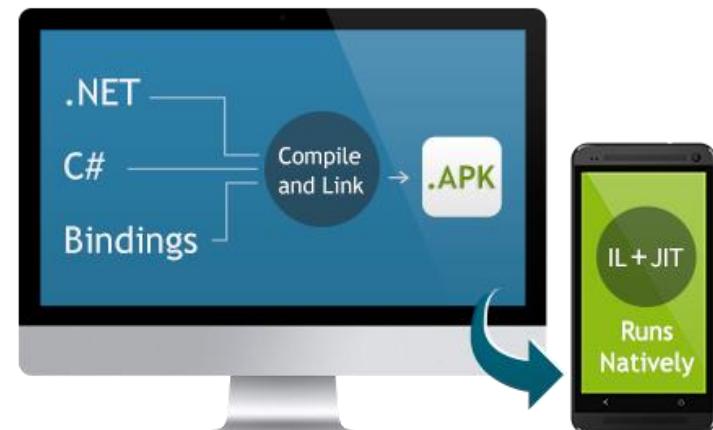
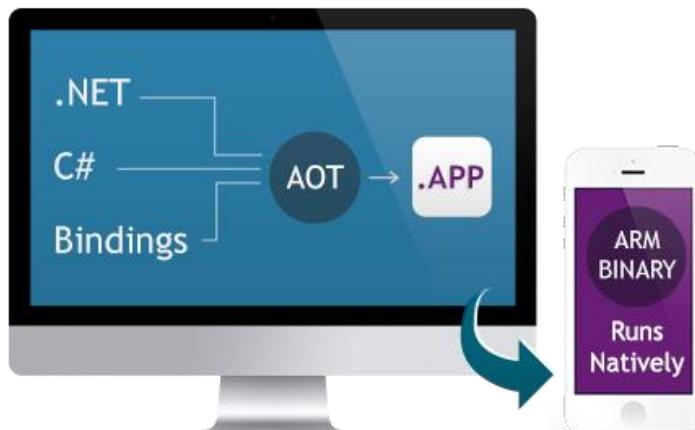


# Android APIs



# Teljesítmény

- iOS: Ahead-Of-Time fordítás ARM-ra
- Android: Just-In-Time bytecode fordítás
- A teljesítmény a natívval közel azonos
  - > De minden áthívás a natív platformba tartalmaz overheadet!





A mítikus kód-újrafelhasználás...

# Xamarin megközelítés



iOS C# UI

Android C# UI

Windows C# UI

Azure

Linux/Mono  
CoreCLR

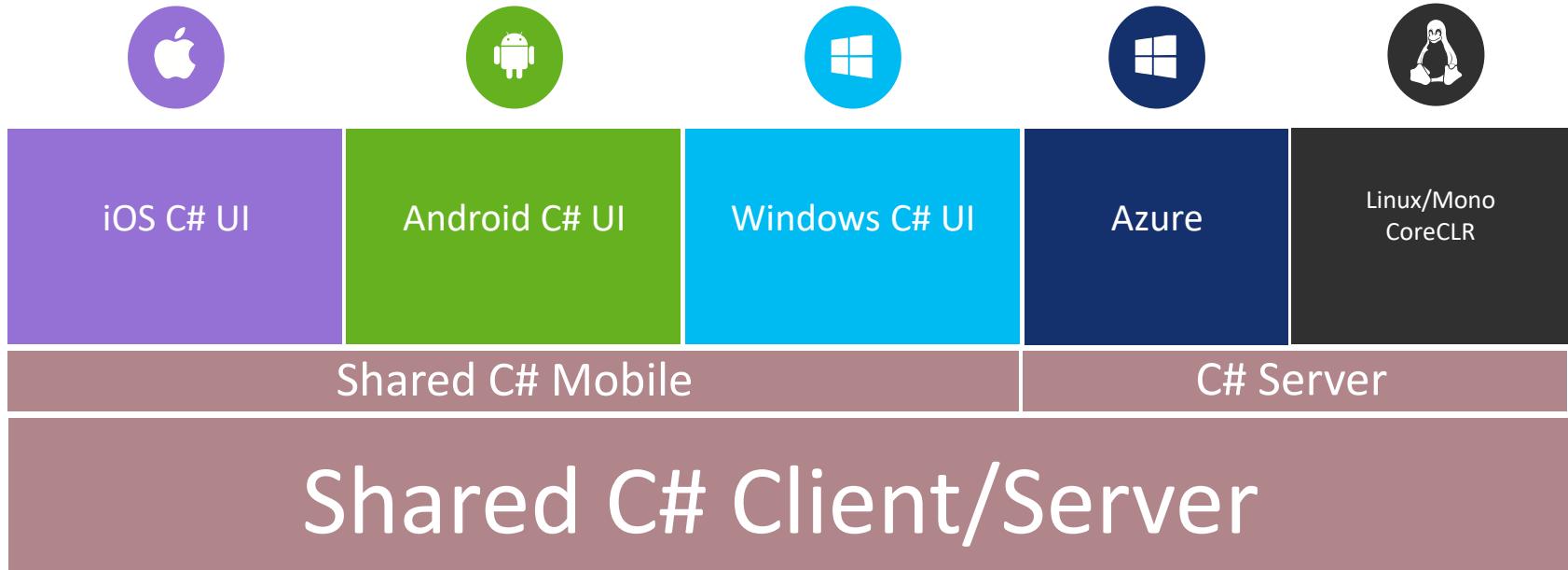
Shared C# Mobile

C# Server

Shared C# Client/Server

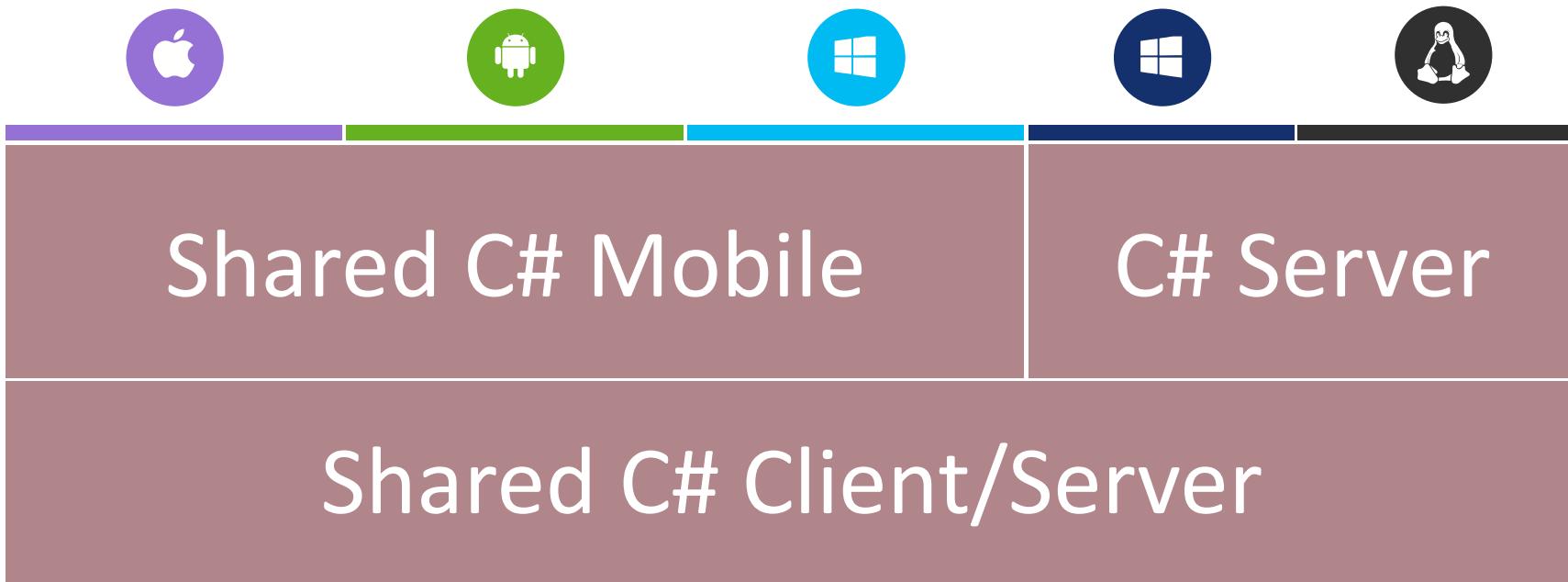
Shared C# codebase • 100% native API access • High performance

# Xamarin megközelítés - ?



Shared C# codebase • 100% native API access • High performance

# Xamarin.Forms-szal még több közös kód



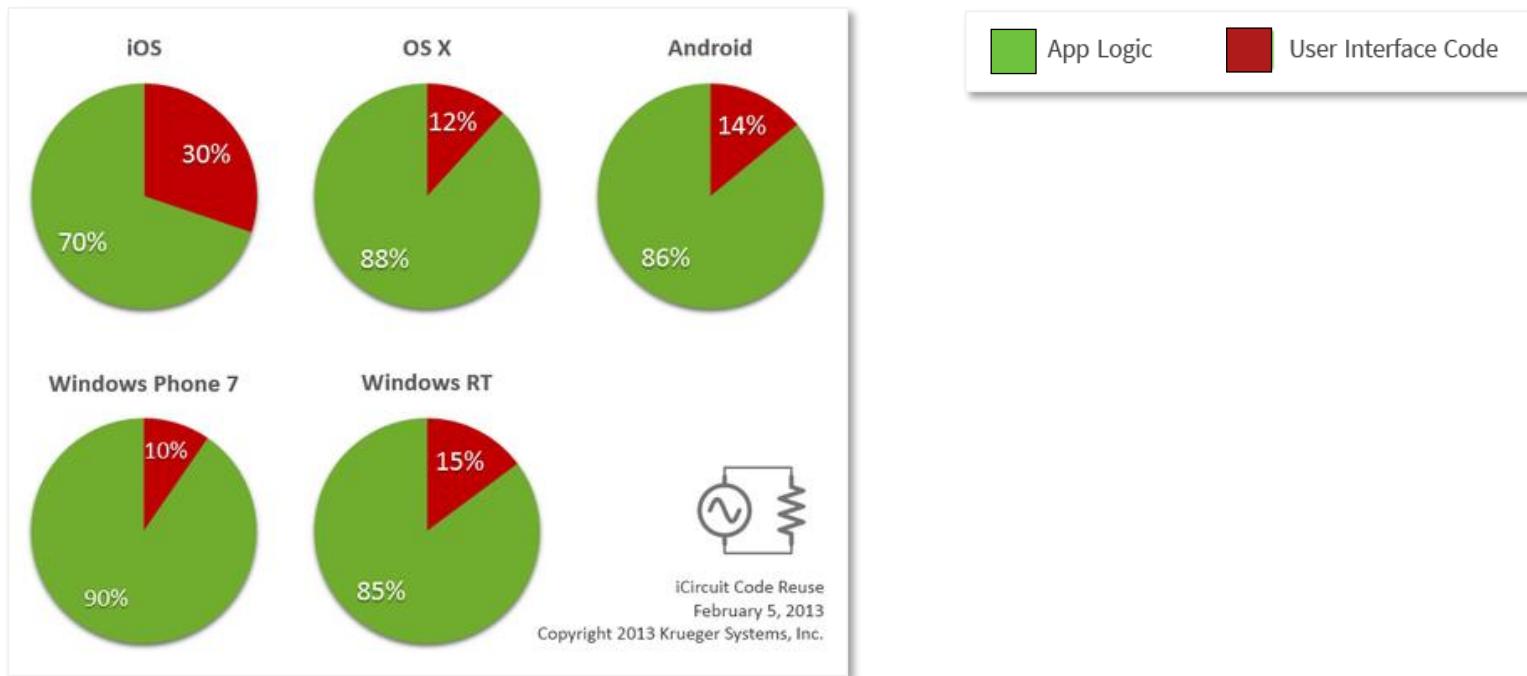
Shared C# codebase • 100% native API access • High performance

# Kód megosztás általában

- Tipikusan megosztható
  - > Adat elérés
  - > Szolgáltatás elérés
  - > Kliens oldali logikák
  - > View-modellek
- Nem megosztható natív Xamarinnal
  - > Felhasználói felület
    - Ha nem használunk Xamarin.Forms-ot
  - > Esemény kezelők
  - > Platformhoz kötött szolgáltatások

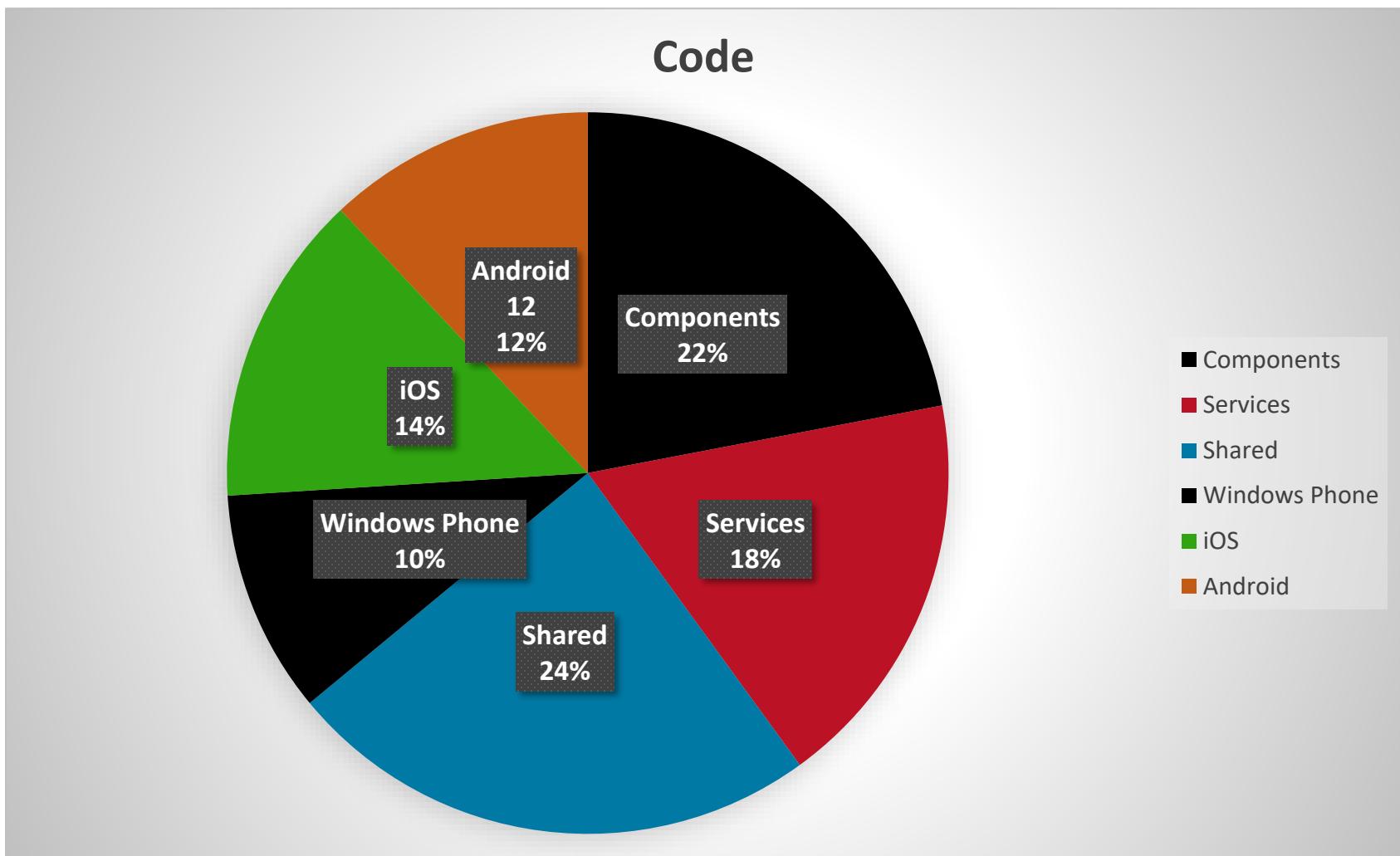
# Code Sharing: Accelerate Development

- Akár 90%-os közös kód
- Meglévő könyvtárak felhasználása
  - > NuGet Support



Statisztika: iCircuit

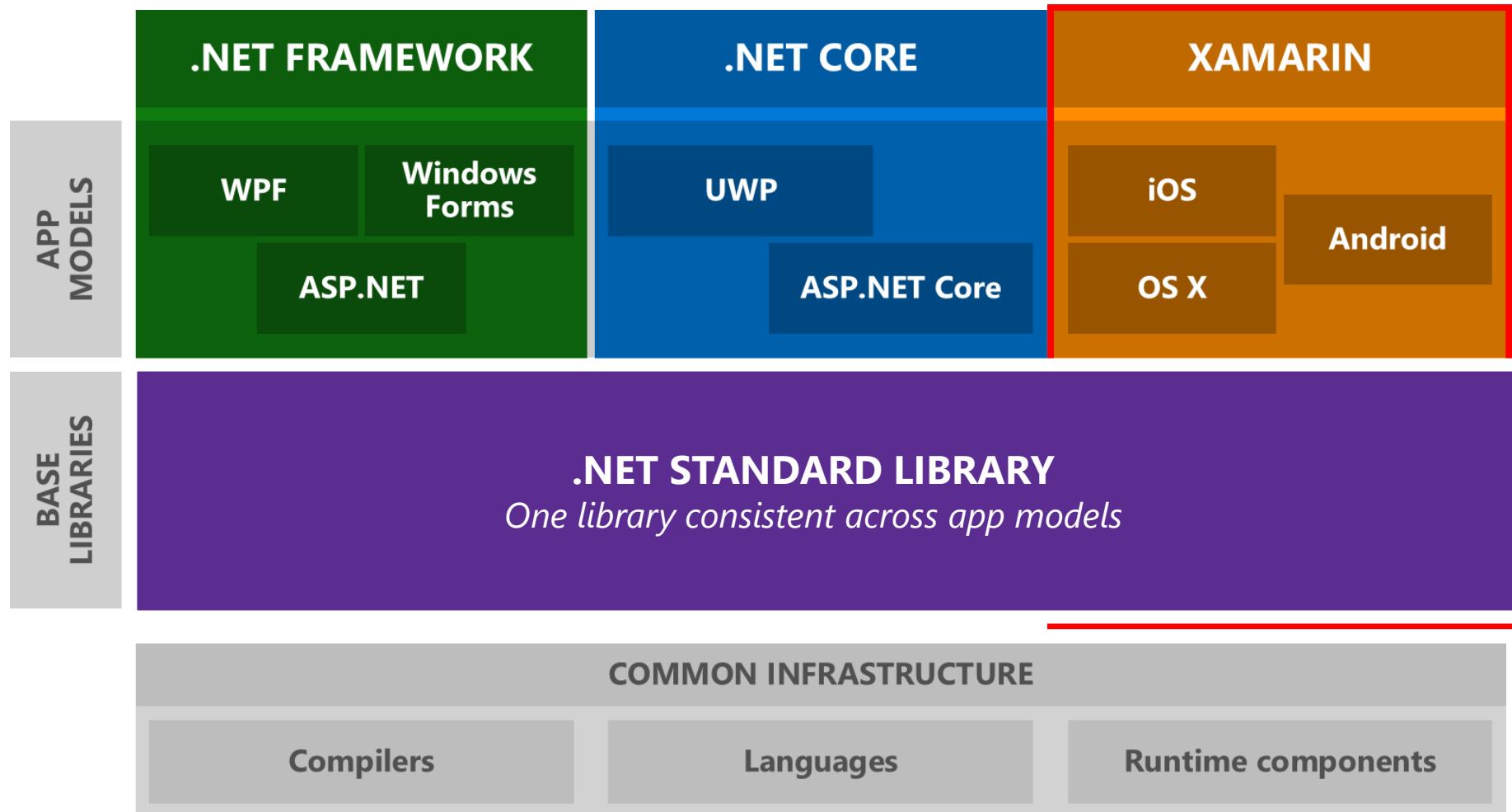
# Statisztikák a Xamarin-tól



# Megosztott kód formája

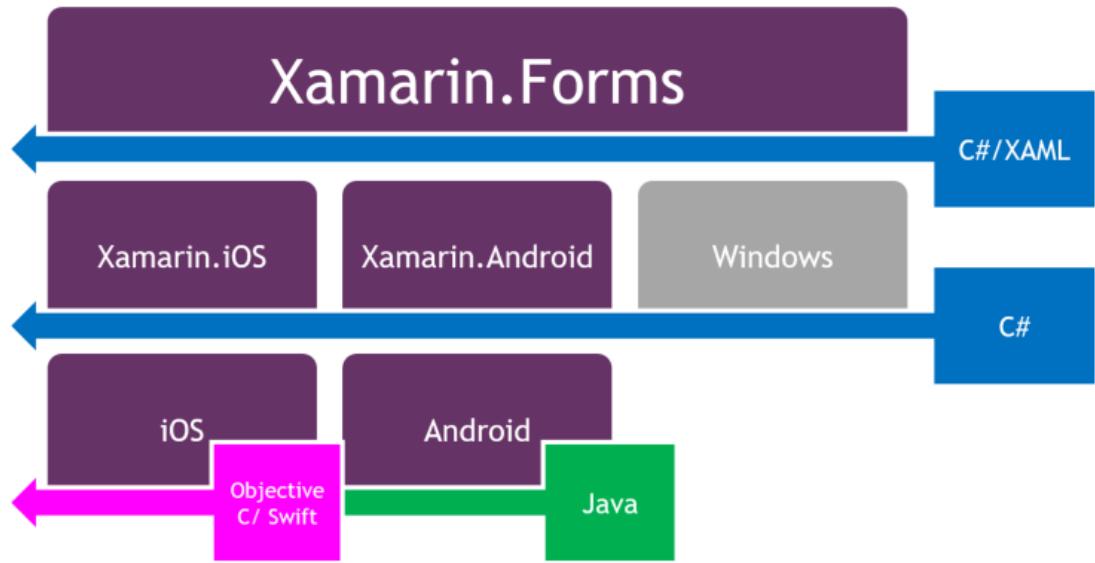
- .NET Standard Library Project:
  - > Javasolt módszer
  - > Beállítható a .NET Standard verziószám
  - > Készítsünk interféseket, ahol platform specifikus hívásokra van szükség
- „Shared projects”:
  - > A kód „másolása” a platform projektekbe
  - > #if direktívák használata platform specifikus kódhoz
- PCL: Portable Class Libraries
  - > Régi megoldás, .NET Standard előtt

# .NET az egyes platformokon

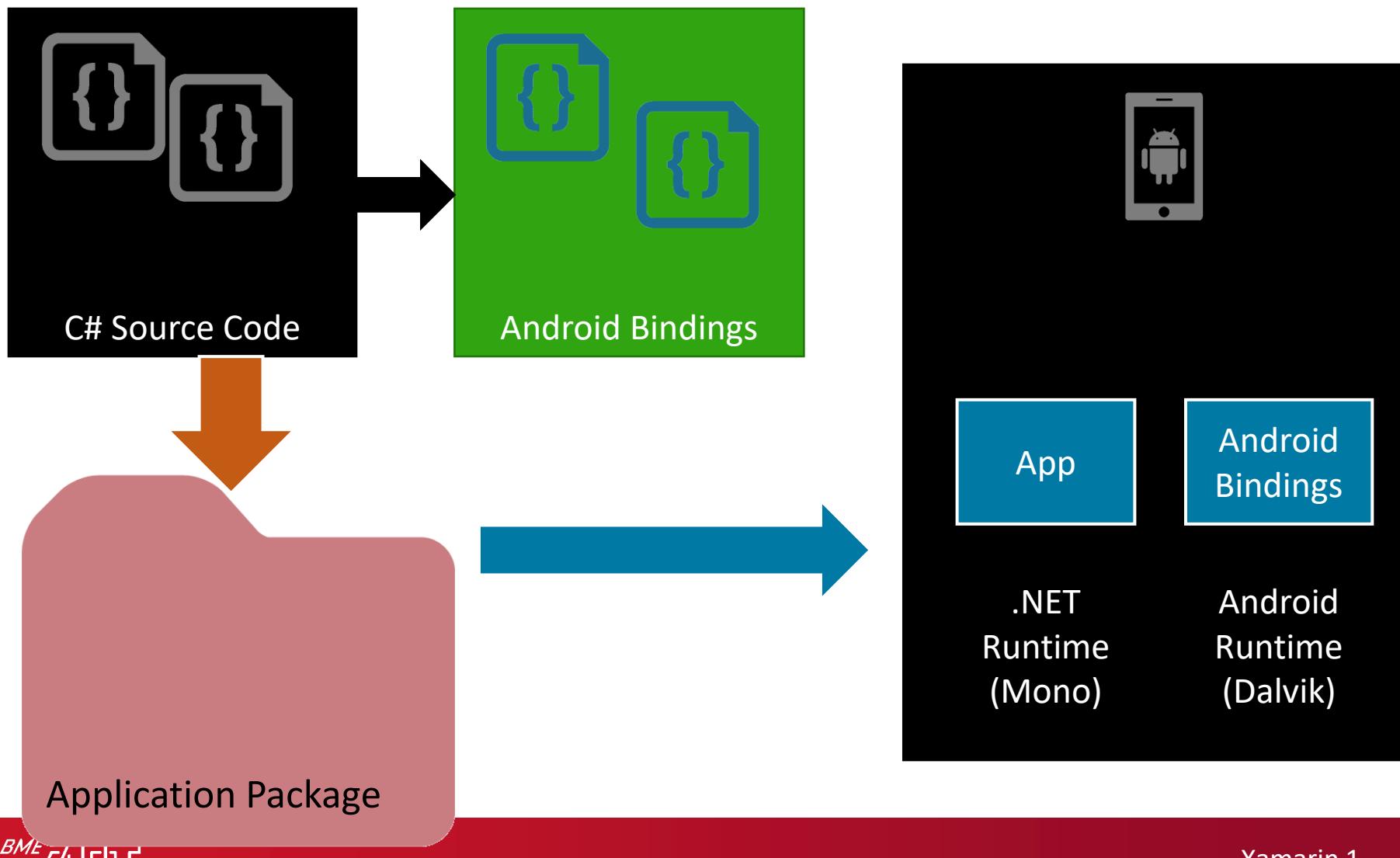


# Alapfogalmak

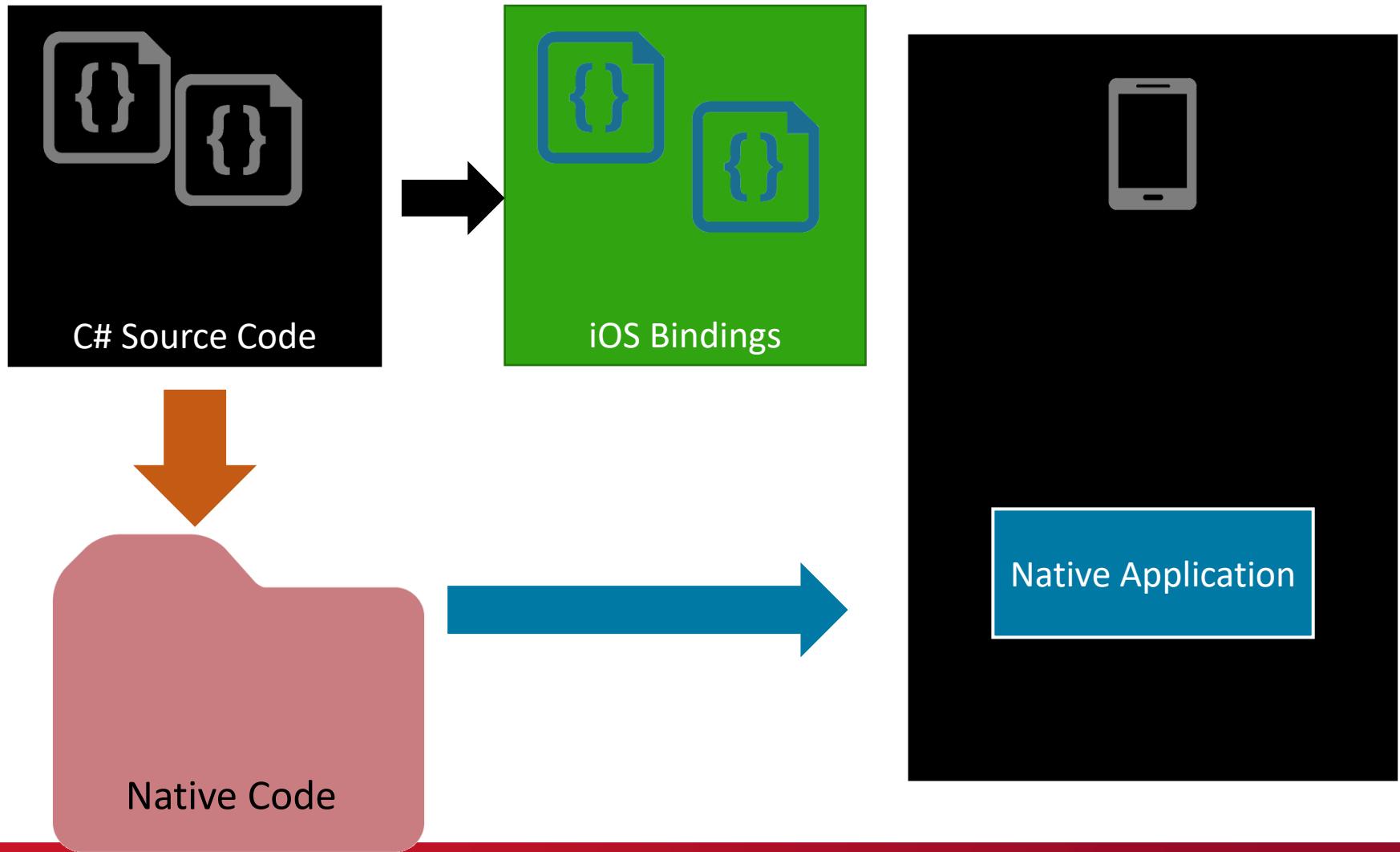
- Xamarin
- Mono
- Xamarin.Android
- Xamarin.iOS
- Xamarin.Mac
- C# de a szokásos Android és iOS architektúra
  - > MVP és MVC
  - > minden platform specifikus API elérhető
- Xamarin.Forms



# C# on Android



# C# on iOS



# Platform virtualizáció (language binding)

- Xamarin.iOS, Xamarin.Android: ezek végzik el a hívást a natív platform API-ba a .NET-es hívásból
  - > .NET-es kód: IL kód (már nem C#)
- iOS: az IL kódot Mac-en fordítja natív ARM kóddá úgy, mint az Objective-C fordító
- Android: az IL kódot a Mono CLR futtatja

# Xamarin tervezési szempontok

- Kövesse a .NET Design Guideline-okat
- Származhassunk az Android / Objective-C osztályokból
- Objective-C / Swift / JavaBean tulajdonságok C#-ban is tulajdonságokként jelenjenek meg
- Használjunk C# metódus referenciákat egy-metódusos interfészek és protokollok helyett
- Típusos API-t használjunk ami biztonságot és IntelliSense-t ad
- IDE-ben is működjön a dokumentáció
- A gyakori Java / Swift / Objective-C feladatok legyenek könnyűek, a nehezek legyenek lehetségesek
- Tetszőleges Java / Swift / Objective-C könyvtár meghívható legyen

# Példa: pozíció meghatározás

- A Xamarin dokumentáció egy-az-egyben lehivatkozza az Android doksit
  - > Ott van az „autentikus” információ
- Android:

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
```

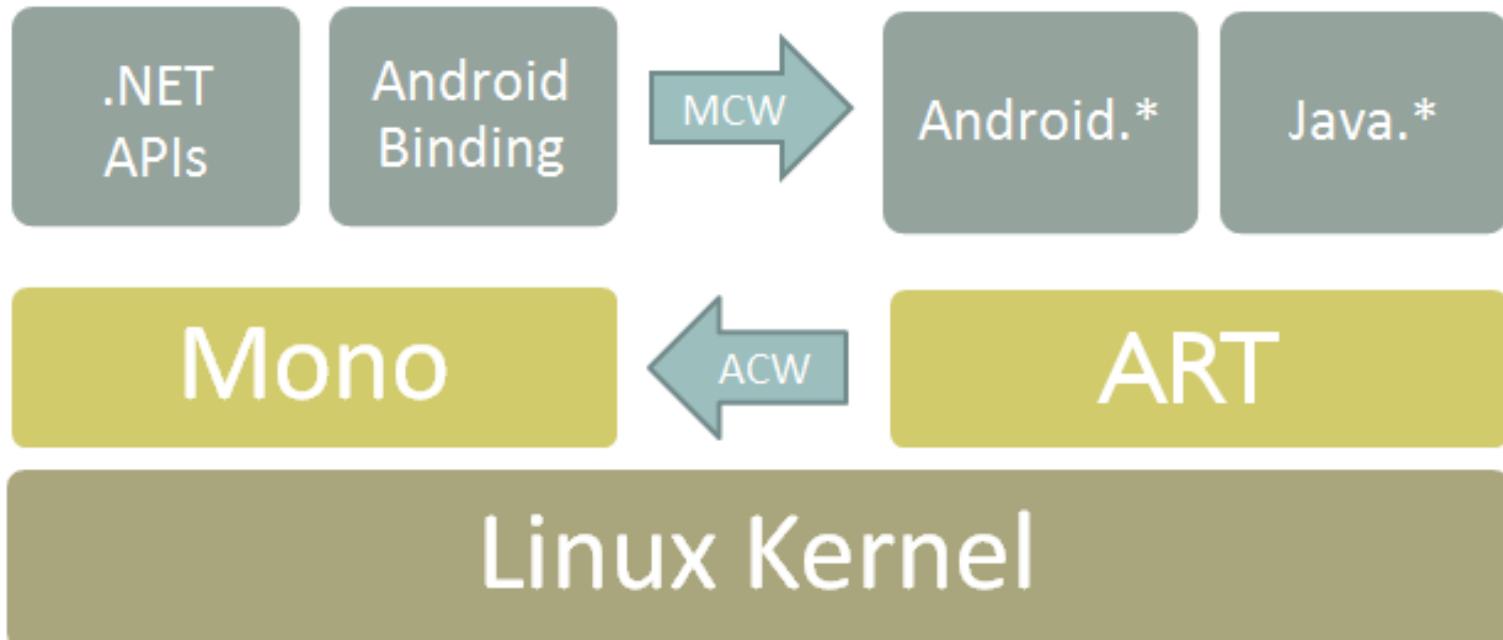
- Xamarin:

```
LocationManager locMgr;
...
locMgr = GetSystemService (Context.LocationService) as LocationManager;
```

- Csak nyelvi különbségek látszanak...

# Xamarin . Android

- Egy alkalmazást két futtatókörnyezet szolgál ki
- Mono
  - > A CLR és HAL C-ben van, a kernelt hívja
  - > Az osztálykönyvtár C#, így érhető el a fájlrendszer stb.
  - > A platform funkciók jelentős része csak az Android Runtime Java API-n érhető el – csomagoló osztályok



# Android Callable Wrapper

- Android/Java oldal hív bele a .NET-es kódba
- JNI-n alapul: Java Native Interface
  - > C jellegű hívásokat tesz lehetővé a javas kódból
- A felülírt virtuális metódusok (override) hívásai is ezen keresztül történnek
  - > Így támogatja .NET-es öröklést is
- A fordítási folyamat részeként automatikusan generálódnak minden .NET-es osztályhoz, ami a **Java.Lang.Object**-ből származik

# Managed Callable Wrapper

- .NET-es oszálykönyvtár, ami teljes androidos névteret lefedi
- JNI kommunikáció, másik irány
- minden .NET-es objektum hivatkozik egy Java-s objektumra
  - > A .NET-es megszűnésekor (Dispose) engedi el a javas referenciát, így a JVM GC tudja törölni

# Egy példa: MCW, ACW

- Egy Android alkalmazás központi elemei az Activityk
- **MCW:** A javas Activity osztály (leszármazottaival, œseivel, interfészeivel) megjelenik a Mono.Android.dll-ben, .NET-es osztályként
  - > Előre generált, a Xamarin része
- Leszármazunk az Activityből és felülírjuk a megfelelő virtuális metódusokat .NET-ben
- **ACW:** a build folyamat során az osztályunkból a Xamarin generál egy Java kódot, ami a felülírt metódusban áthív a .NET-es kódunkba

# Gyűjtemények leképezése

- Inkább a javas megvalósításokat és ne a .NET-es gyűjteményeket használjuk!
  - > minden más gyűjtemény implementáció másolást fog használni, amikor átkerül a Java oldalra!

Java Type	System Type	Helper Class
<a href="#">java.util.Set&lt;E&gt;</a>	<a href="#">ICollection&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaSet&lt;T&gt;</a>
<a href="#">java.util.List&lt;E&gt;</a>	<a href="#">IList&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaList&lt;T&gt;</a>
<a href="#">java.util.Map&lt;K,V&gt;</a>	<a href="#">IDictionary&lt;TKey,TValue&gt;</a>	<a href="#">Android.Runtime.JavaDictionary&lt;K,V&gt;</a>
<a href="#">java.util.Collection&lt;E&gt;</a>	<a href="#">ICollection&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaCollection&lt;T&gt;</a>

# Két irányú gyűjtemény kezelés

- A .NET-es gyűjtemény másolódik, így a java osztályon való módosítás nem az eredeti példányon dolgozik!

```
// This fails:
var badSource = new List<int> { 1, 2, 3 };
var badAdapter = new ArrayAdapter<int>(context, textViewResourceId, badSource);
badAdapter.Add (4);
if (badSource.Count != 4) // true
 throw new InvalidOperationException ("this is thrown");

// this works:
var goodSource = new JavaList<int> { 1, 2, 3 };
var goodAdapter = new ArrayAdapter<int> (context, textViewResourceId, goodSource);
goodAdapter.Add (4);
if (goodSource.Count != 4) // false
 throw new InvalidOperationException ("should not be reached.");
```

# Tulajdonságok, események, ...

- A get... / set... metódusok automatikusan .NET-es tulajdonságokká képeződnek
- Eseményekre feliratkozás működik a .NET-es metódus referenciaikkal (delegate)
- Egyéb leképezések

# Külső komponensek használata

- Java Bindings Library készítése
  - > Automatikus, becsomagolja a hivatalos jar fájlban lévő osztályokat
    - Hasonlóan a Java rendszer osztályokhoz
  - > Ez is JNI-vel dolgozik mélyen...
- JNI alapú megoldás
  - > Nagyon jól finomhangolható de sok munka
- Forráskód szintű portolás
  - > Nagyon sok munka de van automatikus nyelvi konverter

# Android build folyamat

- Debug
  - > Az alaposztály könyvtár és a futtató környezet az Android kötéssel együtt nem a csomag része
  - > Kisebbek a telepítendő szerelvények, gyorsabb a fejlesztési-tesztelési ciklus
- Release
  - > minden, a BCL és a futtató környezet is része a telepítő csomagnak

# Mono

- A .NET-es kód nyílt forráskódú futtatókörnyezete
- Garbage Collection
  - > A .NET-es osztályokat a Mono GC takarítja
  - > A javas osztályokat a JVM takarítja
  - > „Peer objects”: IJavaObject interfész – akkor szűnik meg, amikor minden oldalon elengedik az összes hivatkozást (explicit: Dispose hívás)
  - > Teljesítmény kritikus!
- JIT, ...

# iOS / Objective-C specialitások

- Target-action minta – .NET-es események
- Protokoll – interfész opcionális metódusokkal
- Message – „metódus hívás”
- Delegate – „delegáció” tervezési minta, NEM a .NET-es delegate-re utal (metódus referencia)

# Tulajdonságok, események, ...

- Eseményekre (event) feliratkozás működik a .NET-es metódus referenciaikkal (delegate)
- Protokollok
  - > Absztrakt osztályokká fordulnak
  - > A szükséges (virtuális) metódust felül tudjuk írni

# Protokollhoz tartozó osztály példa

```
[Register ("MKAnnotation"), Model]
public abstract class MKAnnotation : NSObject
{
 public abstract CLLocationCoordinate2D Coordinate
 {
 [Export ("coordinate")]
 get;
 [Export ("setCoordinate:")]
 set;
 }

 public virtual string Title
 {
 [Export ("title")]
 get
 {
 throw new ModelNotImplementedException ();
 }
 }
}
```



# Delegate (iOS nomenklatúra)

- Egy objektum egy feladatot egy másik objektumra bíz
  - > Például a felhasználó kattintás kezelését az iOS vezérlő rábízza egy általunk implementált osztályra
- A kezelhető feladatokat a protokoll (interfész) adja meg, azt implementáljuk
- A vezérlő egy Delegate mezőjén hivatkozunk a protokollt megvalósító objektumra

# Delegate minta, belső osztályval

```
public partial class Protocols_Delegates_EventsViewController : UIViewController
{
 SampleMapDelegate _mapDelegate;
 ...
 public override void ViewDidLoad ()
 {
 base.ViewDidLoad ();

 //set the map's delegate
 _mapDelegate = new SampleMapDelegate ();
 map.Delegate = _mapDelegate;
 ...
 }
 class SampleMapDelegate : MKMapViewDelegate
 {
 ...
 }
}
```

```
public class SampleMapDelegate : MKMapViewDelegate
{
 public override void DidSelectAnnotationView
 (MKMapView mapView, MKAnnotationView ann
 {
 var sampleAnnotation =
 annotationView.Annotation as Sample
 if (sampleAnnotation != null) {
 //demo accessing the coordinate of
 //zoom in on it
 mapView.Region = MKCoordinateRegion(
 sampleAnnotation.Coordinate, 50
);
 //demo accessing the title of the s
 Console.WriteLine ("{0} was tapped"
 }
 }
}
```

# WeakDelegate

- Nem kell a protokollt (interfészt) követni
- Tetszőleges osztály megvalósíthatja a protokoll tetszőleges metódusát

```
public partial class Protocols_Delegates_EventsViewController : UIViewController
{
 ...
 public override void ViewDidLoad ()
 {
 base.ViewDidLoad ();
 //assign the controller directly to the weak delegate
 map.WeakDelegate = this;
 }
 //bind to the Objective-C selector mapView:didSelectAnnotationView:
 [Export("mapView:didSelectAnnotationView:")]
 public void DidSelectAnnotationView (MKMapView mapView,
 MKAnnotationView annotationView)
 {
 ...
 }
}
```

# Delegate-ek eseményekként

- A .NET-es oldal eventekként is látja az eseményeket, hagyományosan fel lehet

```
map.DidSelectAnnotationView += (s,e) => {
 var sampleAnnotation = e.View.Annotation as SampleMapAnnotation;
 if (sampleAnnotation != null) {
 //demo accessing the coordinate of the selected annotation to
 //zoom in on it
 mapView.Region = MKCoordinateRegion.FromDistance (
 sampleAnnotation.Coordinate, 500, 500);

 //demo accessing the title of the selected annotation
 Console.WriteLine ("{0} was tapped", sampleAnnotation.Title);
 }
};
```

# Xamarin - iOS

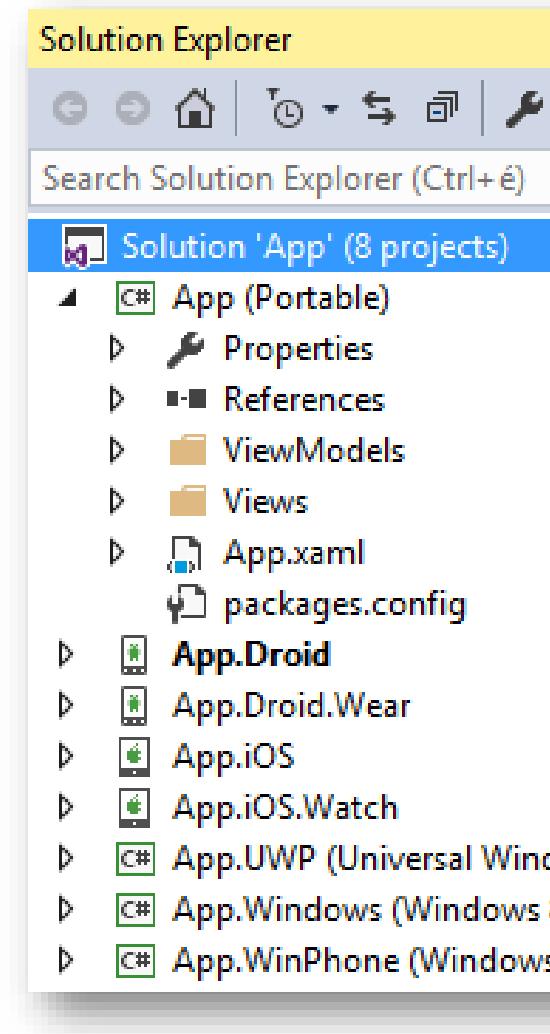
- ObjCRuntime
  - > Leképezés a .NET-es és iOS világ között
  - > Foundation: alaptípusok az interophoz
  - > Általában az Objective-C-s típusokat követi, kivétel:
    - string NSString helyett
    - Típusos tömb NSArray helyett
    - RectangleF, ... - CGRect, ... helyett
- UIKit - vezérlők
- OpenGL ES - grafika

# GC

- minden natív (nem felügyelt) típus elérhető a Handle tulajdonságon
- Dispose hívás rögtön megszünteti a natív objektumot, nem kell várni a takarításra

# Xamarin projekstruktúra

- .NET Standard / Shared / PCL
  - > Teljes egészében platform független kód
  - > Az alkalmazás core része
- Platform specifikus projektek
  - > Android, iOS, UWP
  - > Jellegzetes platform specifikus kód megosztások
    - UWP
    - iOS, Droid



# Alkalmazás építése

- Akár üres solution fájlból is indulhatunk
- Közös kód: shared project, PCL, Standard Lib
  - > Shared project: másolja a fájlokat, tipikusan #if-eket fogunk használni
  - > PCL/.NET Standard: speciális projekt típus, a fordított dll minden .NET-es platformon betölthető: Xamarin.iOS, Xamarin.Android, WPF, Windows Phone, Xbox, ...

# Több platform – ami közös

- Vannak közös koncepciók, amik minden platformra igazak
  - > Tabok, menük használata
  - > Listák, görgetés
  - > Részletes nézet
  - > Szerkesztés
  - > Vissza navigáció
  - > Élet ciklus követés

# Több platform - eltérések

- Platform specifikus területek
  - > Képernyő méretek
  - > Navigációs „metafórák”
    - Manapság erős a konvergencia
  - > Klaviatúra, érintés, gesztus kezelés
  - > Push notification kezelés
    - Különböző lehetőségek, például élő csempék stb.

# Több platform – eszköz specifikus

- Eszköz specifikus funkciók
  - > Egy funkció elérhető-e az adott eszközön
- Például
  - > Kamera
  - > Geo-pozíció
  - > Gyorsulásmérő, giroszkóp, iránytű
  - > Twitter, Facebook integráció
    - Beépített a platformba vagy külön API kell hozzá
  - > Near Field Communications (NFC)
  - > Fizetés API

# Platform absztrakció

- Alaposztályok a közös kódban
  - > Alap funkció, logika közös
- Xamarin.Forms
  - > Közös UI – ahol lehet
- A Xamarin BCL lehetővé teszi bizonyos funkciók elérését közös kódból
  - > Media Picker: fénykép készítése, média elem kiválasztása
  - > Geolokáció kezelés
  - > Névjegyek kezelése
  - > Tetszőlegesen bővíthető

# Amikor nem elég az absztrakció

- Feltételes fordítás
  - > MOBILE : iOS, Android
  - > IOS : iOS eszközök
  - > ANDROID : Android eszközök
  - > ANDROID\_XX specifikus Android verzió
    - Például: ANDROID\_11

# Fordítási direktíva példa

```
public static string DatabaseFilePath {
 get {
 var filename = "MwcDB.db3";

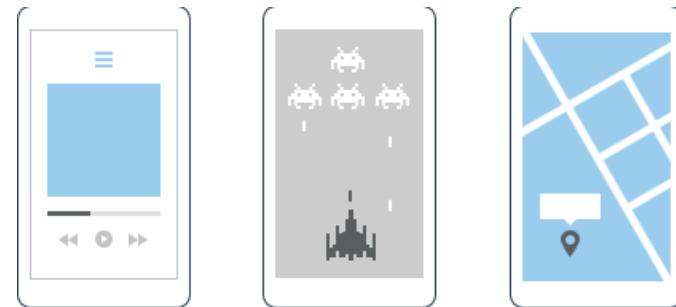
#if __ANDROID__
 string libraryPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal); ;
#else
 // we need to put in /Library/ on iOS5.1 to meet Apple's iCloud terms
 // (they don't want non-user-generated data in Documents)
 string documentsPath = Environment.GetFolderPath (Environment.SpecialFolder.Personal);
 // Documents folder
 string libraryPath = Path.Combine (documentsPath, "..", "Library");
#endif
 var path = Path.Combine (libraryPath, filename);
 return path;
 }
}
```

# Xamarin.Forms

- Nyílt forráskód
- Elsősorban platform független UI megvalósítására
- UI definiálása XAML-lel vagy C#-ban
- Kiterjeszthető, az egyedi igények szerint
- Mikor válasszuk?



Xamarin.Forms

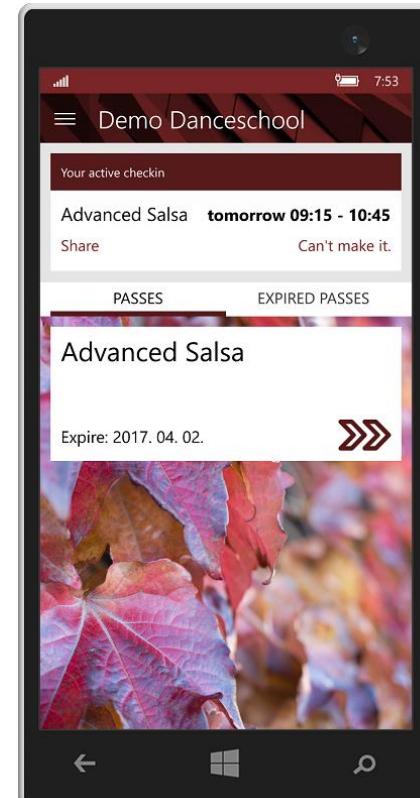
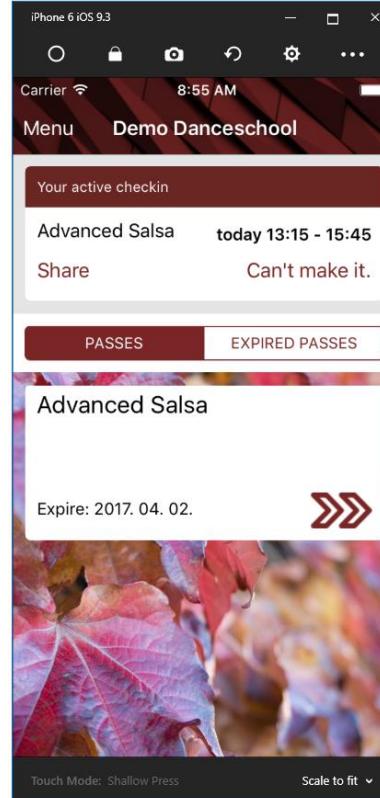
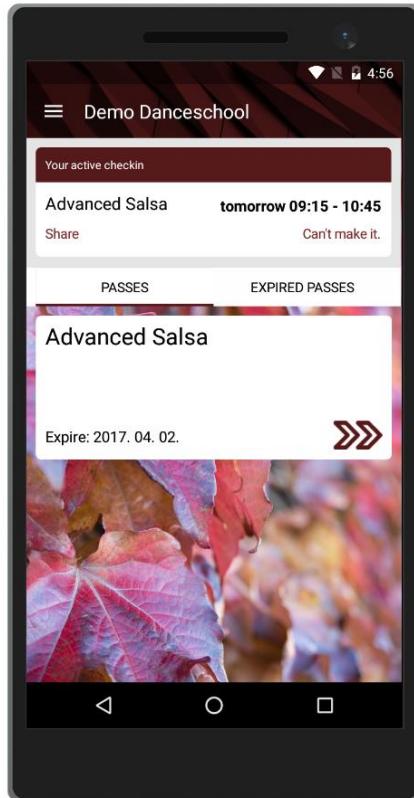


Xamarin.iOS, Xamarin.Android

# Valós példa

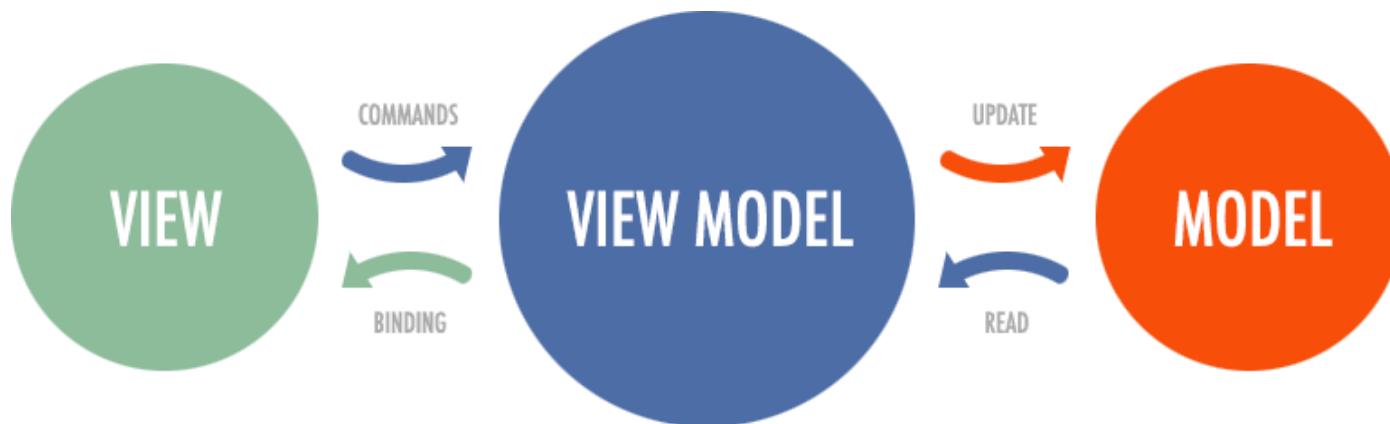
- CheckIn Swift

CheckinSwift	14586	86.36%
CheckinSwift.Droid	713	
CheckinSwift.iOS	927	13.64%
CheckinSwift.UWP	663	
16889		



# Alkalmazás architektúra

- Platform specifikus és platform független komponensek
- MVVM architektúra
  - > Valamennyi réteg platform független
  - > De kiterjeszthető platform specifikus kóddal is

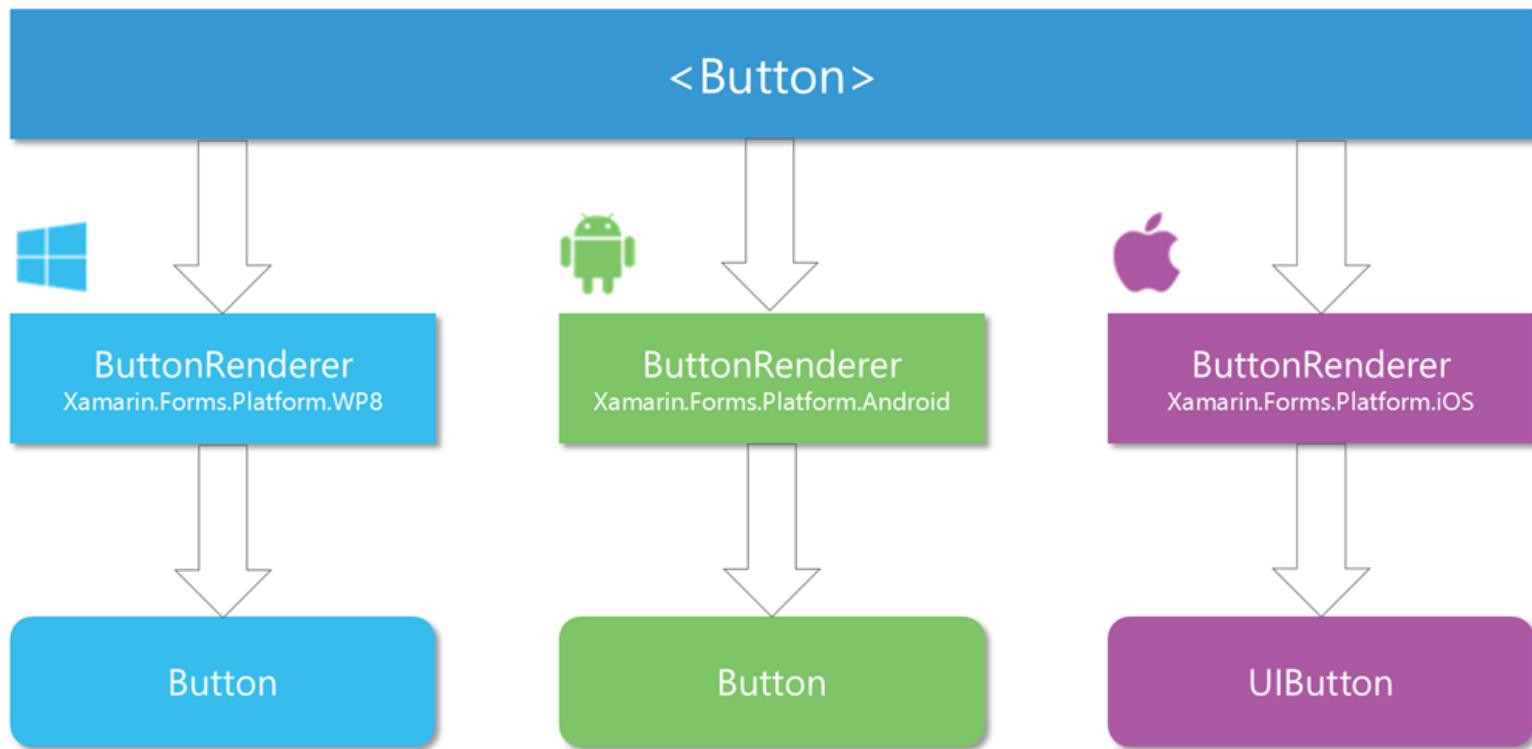


# Teljesítmény

- Mennyire hatékony a natív nézetek ilyen módon történő kezelése?
- Teljesítmény növelés módjai
  - > XAML fordítás
  - > Megfelelő, egyszerűbb layout választása
  - > Kevesebb adatkötés
  - > ListView-k megfelelő kezelése, virtualizációja
  - > Kevesebb globális Resource
  - > Renderer pattern betartása

# ViewRendererek

- XAML elemekből natív nézet
- Ez már platform specifikus logika



# ViewRenderer példa

```
[assembly: ExportRenderer(typeof(Label), typeof(Common.Droid.Controls.LabelRenderer))]
namespace Common.Droid.Controls
{
 1 reference | Balázs Ádám, 7 days ago | 1 author, 1 change
 public class LabelRenderer : ViewRenderer<Label, TextView>
 {
 19 references | Balázs Ádám, 7 days ago | 1 author, 1 change
 protected override void OnElementChanged(ElementChangedEventArgs<Label> e)
 {
 base.OnElementChanged(e);

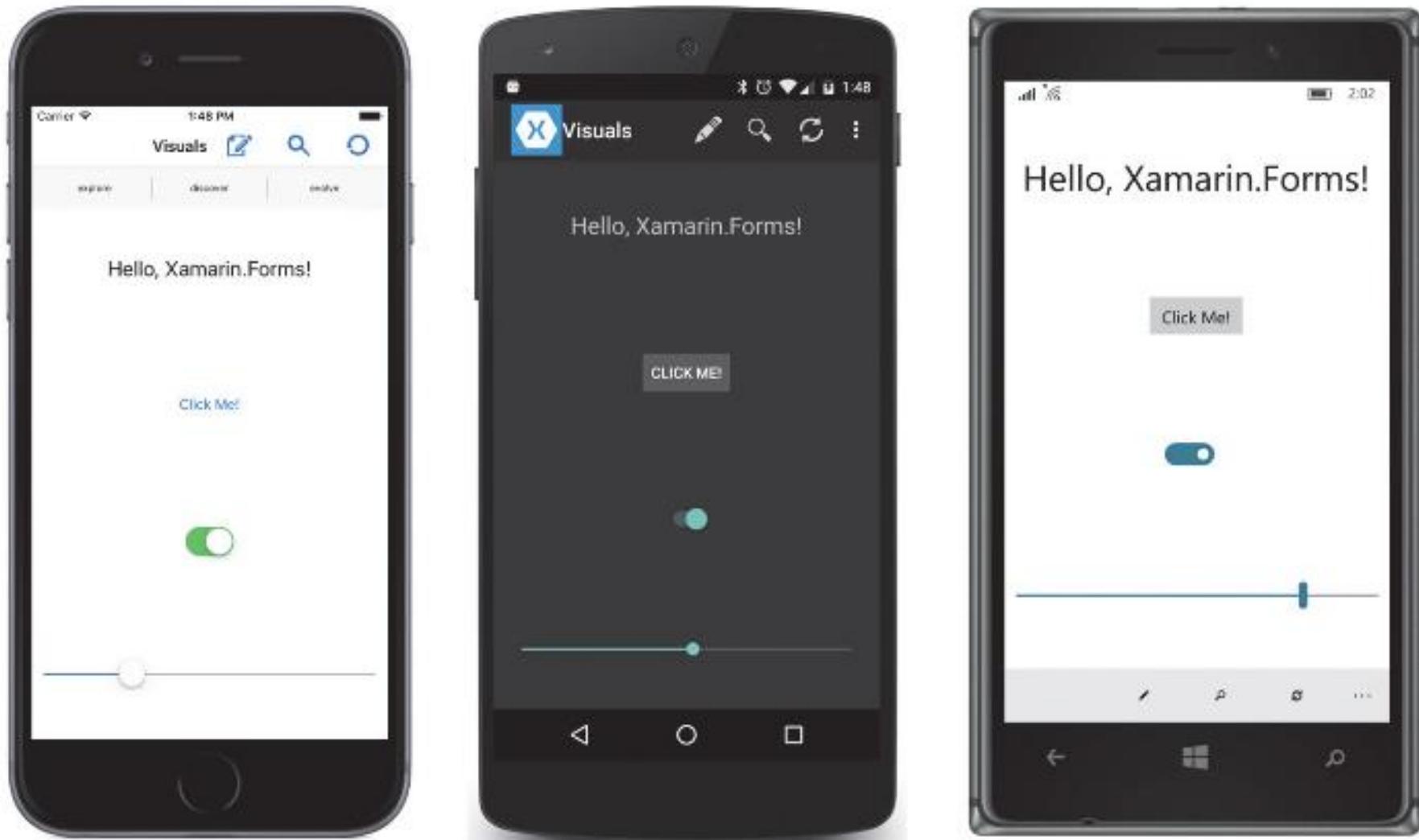
 if(e.OldElement!=null)
 {
 //Takarítás
 }

 if (e.NewElement != null)
 {
 SetNativeControl(new TextView(Context));
 }
 }

 19 references | Balázs Ádám, 7 days ago | 1 author, 1 change
 protected override void OnElementPropertyChanged(object sender, PropertyChangedEventArgs e)
 {
 base.OnElementPropertyChanged(sender, e);

 if (e.PropertyName == Label.TextProperty.PropertyName)
 Control.Text = Element.Text;
 }
 }
}
```

# Különböző vezérlők platformonként



# XAML jellemzők, egyediségek

- Kicsit más

```
<StackLayout HorizontalOptions="Center" IsVisible="{Binding IsPanelVisible}">
 <Label Text="Hello from Xamarin.Forms" />
</StackLayout>
```

- Data Binding

- > Dependency Property helyett Bindable Property
  - > Attached propertyk

- Resource-ok

- > Style-ok, Control Template-ek, Data Template-ek, színek vagy konverterek
  - > StaticResource vagy DynamicResource

# XAML jellemzők, egyediségek

- Könnyen kiterjeszthető saját Markup Extensionnel
  - > Például a Translate egy bővítés

```
<Label Text="{controls:Translate Text=Email_LoginLabel}"
 LineBreakMode="WordWrap"
 TextColor="{StaticResource DefaultAccentColor}"
 Style="{StaticResource SubtitleLabelStyle}"/>
```

- Akár natív nézetek is beágazhatóak a XAML kódba
  - > Adatkötés is támogatott

```
<StackLayout Orientation="Vertical" >
 <iOS:UILabel Text="Native Text" View.HorizontalOptions="Start"/>
 <Android:TextView Text="Native Text" x:Arguments="{x:Static formsAndroid:Forms.Context}" />
 <Windows:TextBlock Text="Native Text"/>
</StackLayout>
```

# Platformonként eltérő értékek

- **OnPlatform** xml tag
- **TypeArgument:** az érték típusa

```
<Grid.Margin>
 <OnPlatform x:TypeArguments="Thickness" iOS="10" Android="15" WinPhone="15" />
</Grid.Margin>
```

```
<ToolbarItem Text="edit" Order="Primary">
 <ToolbarItem.Icon>
 <OnPlatform x:TypeArguments="FileImageSource"
 iOS="edit.png"
 Android="ic_action_edit.png"
 WinPhone="Images/edit.png" />
 </ToolbarItem.Icon>
</ToolbarItem>
```

# INotifyPropertyChanged interfész

- A XAML-ben szokásos INPC interfész
- Itt is MVVM-et használunk
- Itt is vannak keretrendszerök!

# BindableObject osztály

- Hasonló a DependencyObject-hez
  - > DependencyProperty helyett BindableProperty
- Megvalósítja az INPC-t!
  - > Szemben a DependencyObjecttel

```
class MockBindableObject : BindableObject
{
 public static readonly BindableProperty NameProperty =
 BindableProperty.Create("Name", typeof(string),
 typeof(MockBindableObject),
 null);

 0 references
 public string Name
 {
 get { return (string)GetValue(NameProperty); }
 set { SetValue(NameProperty, value); }
 }
}
```

# Adatkötés Xamarinban 1.

- Adatforrás: BindableObject . **BindingContext**
  - > Hasonlóan a DataContext-hez, a gyerekek öröklik a szülőtől
- Binding osztály
  - > BindingMode
    - Default: a vezérlő propertyjén megadott alapértelmezett
    - TwoWay: frissíti az adatforrást
    - OneWay: csak a vezérlőt frissíti
    - OneWayToSource: csak az adatforrást frissíti
  - > StringFormat - formázás

# Adatkötés Xamarinban 2.

- Source
  - > A forrás adat osztály
- Path
  - > A property a forrás objektumon, amihez kötünk
- Converter, ConverterParameter
  - > IValueConverter interfész

```
this.SetBinding(
 Page1.TitleProperty,
 Binding.Create<BaseViewModel>(m => Title));
```

The image shows a screenshot of a Xamarin code editor. On the left, there is C# code:this.SetBinding(  
 Page1.TitleProperty,  
 Binding.Create<BaseViewModel>(m => Title));On the right, there is XAML code:<ContentPage xmlns="http://xamarin.com/  
 xmlns:x="http://schemas.mi  
 x:Class="Parkolunk.Views.A  
 xmlns:vm="clr-namespace:Pa  
 Title="{Binding Title}">A tooltip or callout box is visible over the XAML code, pointing to the `Title` property. The tooltip contains the text "Title={Binding Title}" and "Title".

# Csatolt tulajdonságok

- Hasonlóan más XAML technológiákhoz itt is van
- Tetszőleges objektumra rátehető egy másik osztály által kezelt tulajdonság
  - > Például: AbsoluteLayout . LayoutBounds

```
<AbsoluteLayout VerticalOptions="FillAndExpand"
 HorizontalOptions="FillAndExpand">
 <BoxView AbsoluteLayout.LayoutBounds="0.25, 0.25, 0.5, 0.5"
 Color="Blue"
 AbsoluteLayout.LayoutFlags="All" />
</AbsoluteLayout>
```

# Compiled bindings

- Hasonló az x:Bindhoz -> érdemes használni!
  - > 5-20-szor gyorsabb a reflection alapú adatkötésnél
- XAML fordítás bekapcsolása

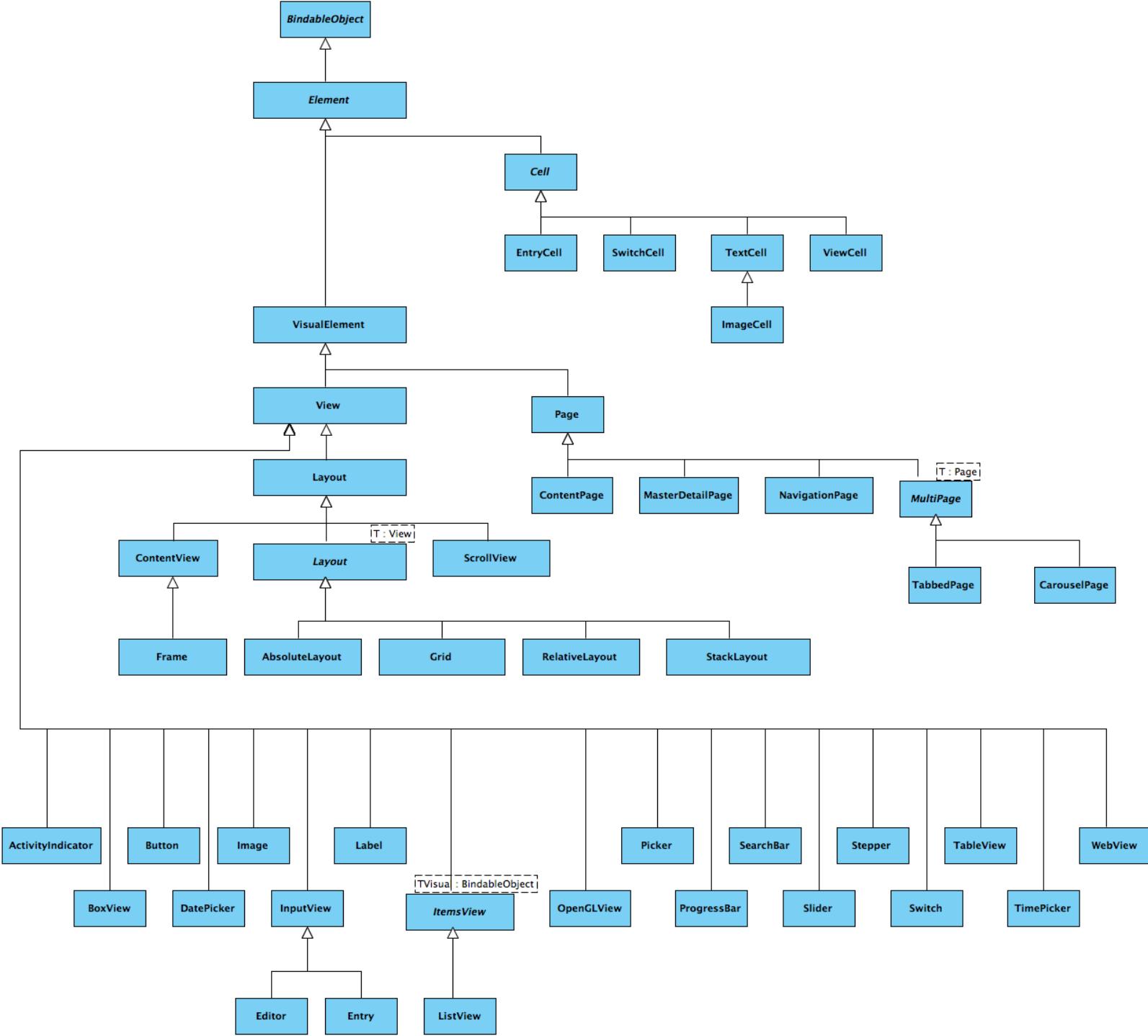
```
using Xamarin.Forms.Xaml;
...
[assembly: XamlCompilation (XamlCompilationOptions.Compile)]
```

- Az x:DataType-pal adjuk meg az adatforrás típusát
  - > Vezérlőkön és adatsablonokon is!

```
<DataTemplate x:DataType="viewModels:ListItemViewModel">
...
</DataTemplate>
```

# Element osztály : BindableObject

- Az objektum hierarchiában résztvevő elemek alaposztálya
  - > Parent: közvetlen szülő
  - > ParentView (readonly): látható szülő
  - > ChildAdded/Removed események
    - Közvetlen gyerek
  - > DescendantAdded/Removed események
    - Az új gyerek elem gyerekeire is elsül



# VisualElement osztály : Element

- A képernyőn megjelenő vezérlő elem
  - > Kiterjedés: Bounds, Height, Width
  - > X, Y: lekérdezhető aktuális pozíció a szülőhöz képest
- Transformáció
  - > RotationX,Y,Z: perspektívikus is
  - > Scale: aránytartó
  - > TranslationX/Y: eltolás
- Opacity, BackgroundColor (egyszerű szín)
- Style, StyleClass
- IsEnabled, IsVisible, IsFocused
- Navigation

# ELÁGAZÁS

- VisualElement alaposztály
- Page irány
  - > Teljes képernyőt elfoglaló szülő elem
  - > Egyelten, View típusú tartalommal
- View osztály
  - > Vezérlők
  - > Elrendező (layout) elemek

# Page osztály : VisualElement

- A teljes képernyőt elfoglaló elem
  - > Title: szöveges cím
  - > ToolbarItems: „menü”
  - > Padding: térköz
  - > Icon, BackgroundImage
- IsBusy: platform specifikus várakozó animáció
- DisplayAlert: modális dialógus ablak
- ForceLayout() / LayoutChanged
- Appearing / Disappearing

# View osztály : VisualElement

- Klasszikus vezérlők és elrendező elemek őse
- Horizontal/VerticalOptions: elrendezés
  - > LayoutOptions osztály
    - Statikus tagok a gyakran használt opciókhöz
  - > Alignment tag
    - Start, End, Center, Fill: bal/fenn, jobb/lenn, közép, kitölt
  - > Expands tag
    - Kitölți-e a vezérlő a rendelkezésre álló helyet
- Margin

# Layout osztály : View

- Alaposztály, a rajta lévő vezérlőket helyezi el
- Padding: a gyerek elemek távolsága a kerettől
- IsClippedToBounds: vágás
- LowerChild / RaiseChild: vizuális Z index állítás
- ForceLayout: elrendezés újraszámolása
- GetSizeRequest: mekkora helyre van szüksége

# ScrollView osztály : Layout

- A benne lévő tartalom görgethető
- Content tulajdonság: tetszőleges View leszármazott
- Orientation: görgetés iránya
- ScrollX/Y, Scrolled esemény: pozíció lekérdezése
- ScrollToAsync: görgetés adott helyre
- ScrollView vezérlőket nem szabad egymásba ágyazni!

# Button

- Egyszerű gomb
- Text: megjelenítendő szöveg
  - > Betűkészlet, -méret, -szín + félkövér, italic
- Image: megjelenítendő kép
- Testreszabható keret
- Clicked esemény, ICommand interfész



# Label

- Egy vagy több sornyi szöveg megjelenítése
- Szöveg tulajdonsága
  - > Betűkészlet, -méret, -szín + félkövér, *italic*
  - > Formázás akár karakterenként
- Szöveg tördelés
  - > Karakterenként vagy szavanként, "... helye
- Szöveg elrendezése horizontálisan, vertikálisan
  - > Nincs sorkizárt

# Xamarin 1

< Albert István >



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Tartalom

## Mi a Xamarin?

- Technológia, amivel minden nagyobb mobil platform elérhető

- Nativ felhasználói felület
- Nativ teljesítmény
- Közös kód a platformok között
- C# & .NET Framework



AU/UT

Xamarin 1

## Fejlesztő környezet

- Mac: Xamarin Studio
- Windows: Visual Studio



Development Environment	MACOS	WINDOWS	XAMARIN STUDIO
Xamarin.iOS	Yes	Yes (with Mac computer)	No
Xamarin.Android	Yes	Yes	Yes
Xamarin.Forms	iOS & Android only	Android, Windows, Windows Phone (iOS with Mac computer)	Android only
Xamarin.Mac	Yes	Open project & compile only	No

AU/UT 27 Xamarin 1

## Pozíció meghatározás

- A Xamarin dokumentáció egy-az-egyen lehívhatkoza az Android doksit  
-> Ott van az „autentikus” információ
- Android:

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

• Xamarin:

LocationManager locMgr;
...
locMgr = GetSystemService(Context.LocationService) as LocationManager;

• Csak nyelvi különbségek látszanak...
```

AU/UT

Xamarin 1

## iOS / Objective-C specialitások

- Target-action minta – .NET-es események
- Protokoll – interfész opcionális metódusokkal
- Message – „metódus hívás”
- Delegate – „delegáció” tervezési minta, NEM a .NET-es delegate-re utal (metódus referencia)

AU/UT Xamarin 1

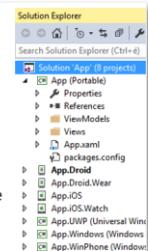
## Alapfogalmak

- Xamarin
- Mono
- Xamarin.Android
- Xamarin.iOS
- Xamarin.Mac
- C# de a szokásos Android és iOS architektúra
  - > MVP és MVC
  - > minden platform specifikus API elérhető
- Xamarin.Forms

AU/UT 39 Xamarin 1

## Projekstruktúra

- PCL / .NET Standard / Shared
  - > Teljes egészében platform független kód
  - > Az alkalmazás core része
- Platform specifikus projektek
  - > Android, iOS, UWP, Windows Phone
  - > Jellegzetes platform specifikus kód megosztások
    - UWP, Windows 8.1, Windows Phone 8.1
    - iOS, Droid



AU/UT

65

Xamarin 1

# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

# Mi a Xamarin?

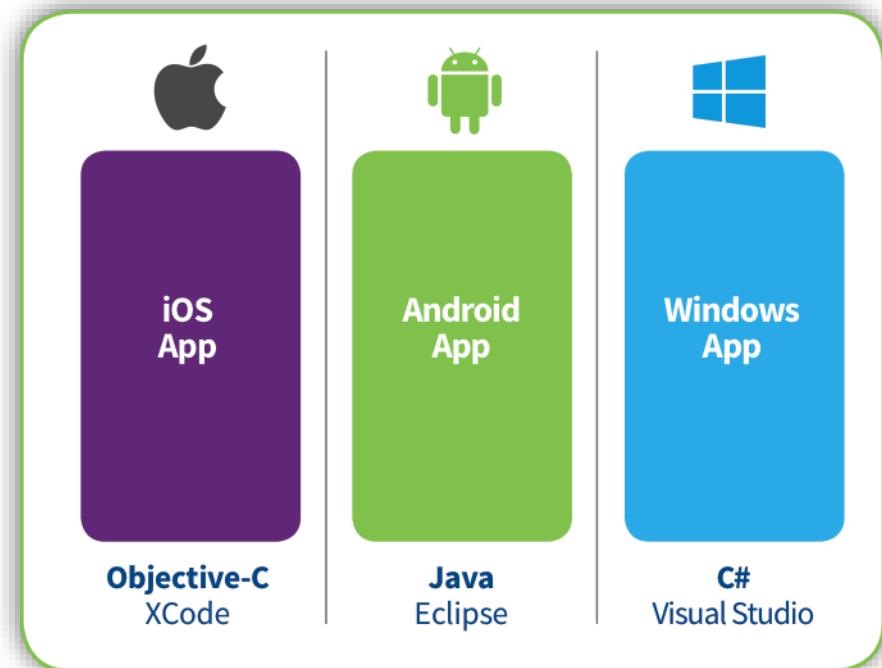
- Technológia, amivel minden nagyobb mobil platform elérhető
  - > Natív felhasználói felület
  - > Natív teljesítmény
  - > Közös kód a platformok között
  - > C# & .NET Framework



# Silós megközelítés

## Több natív alkalmazást készítünk

- Több csapat
- Több kód bázis
- Különböző eszközök
- Fejlesztés
- Karbantartás
- HR menedzsment



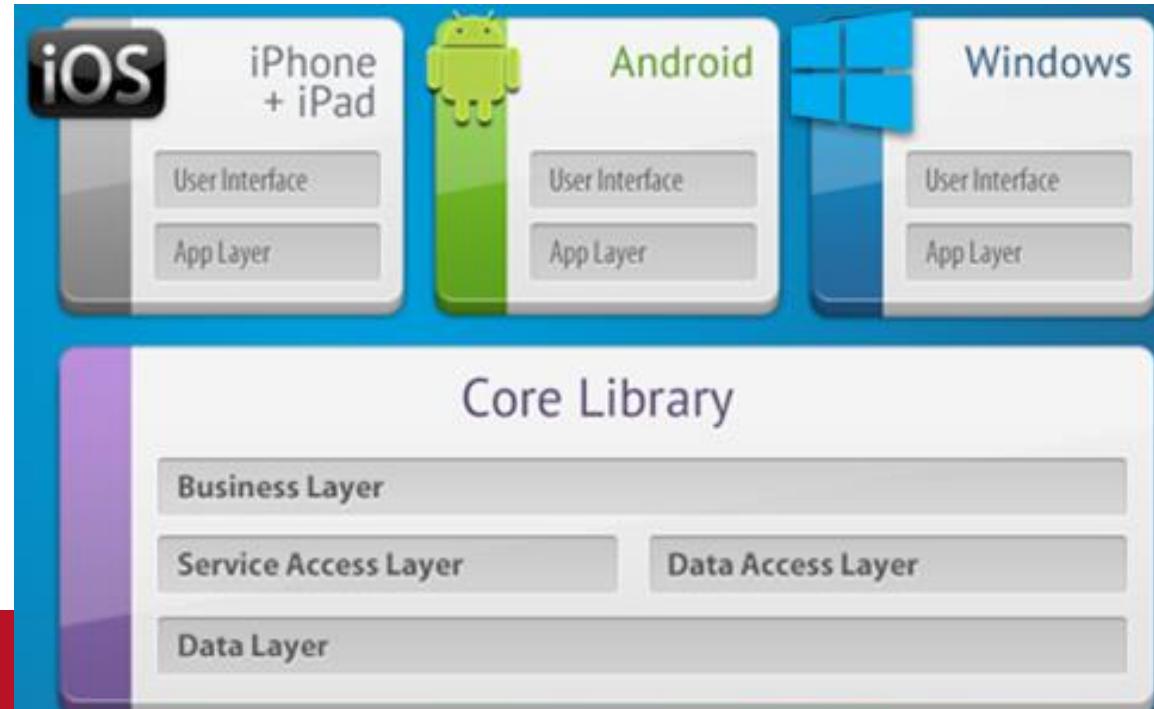
# Írjuk meg egyszer, futtassuk mindenütt

- Legkisebb közös többszörös
- Böngésző töredézettség
- Valójában egy platformra tervezük és végül több platformon futtatjuk
- Elégedetlen felhasználók
- Elégedetlen fejlesztők
- Korlátosztott alaprendszer



# Xamarin megközelítés

- Natív felhasználói felület
- Natív teljesítmény
- Közös kód a platformok között
- C# & .NET Framework
- Teljes API elérés



# Xamarin megközelítés



iOS C# UI

Android C# UI

Windows C# UI

Azure

Linux/Mono  
CoreCLR

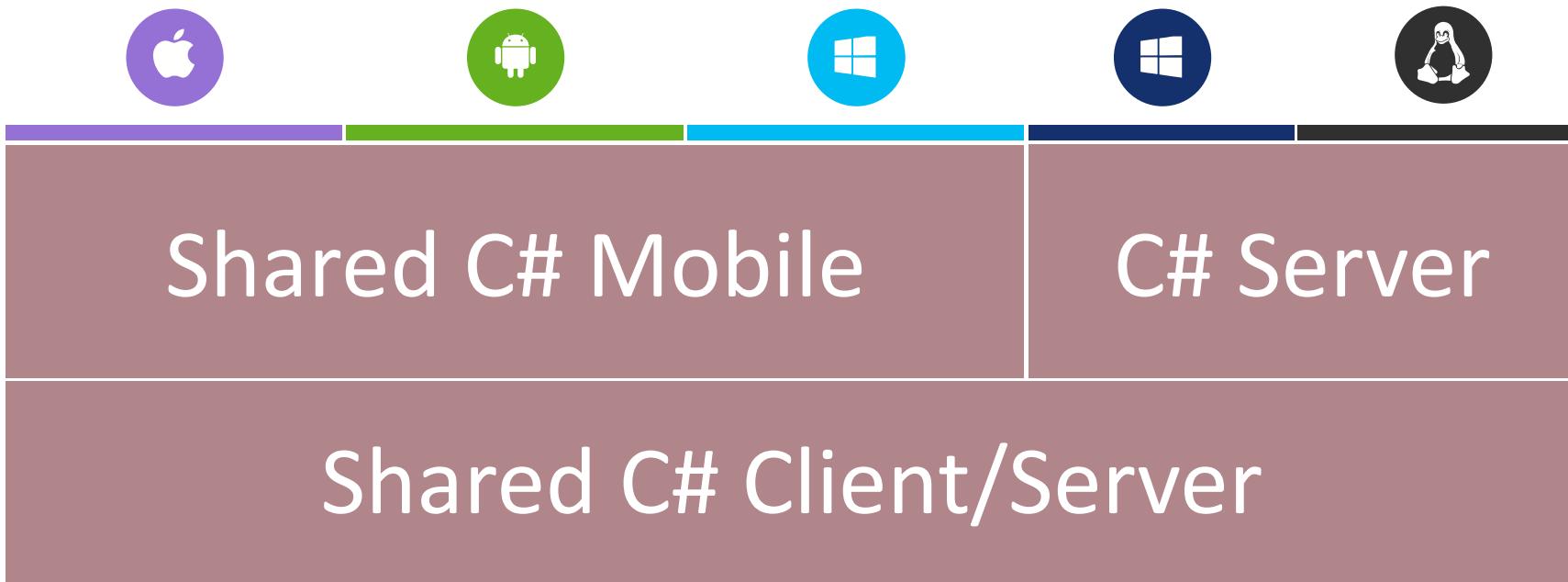
Shared C# Mobile

C# Server

Shared C# Client/Server

Shared C# codebase • 100% native API access • High performance

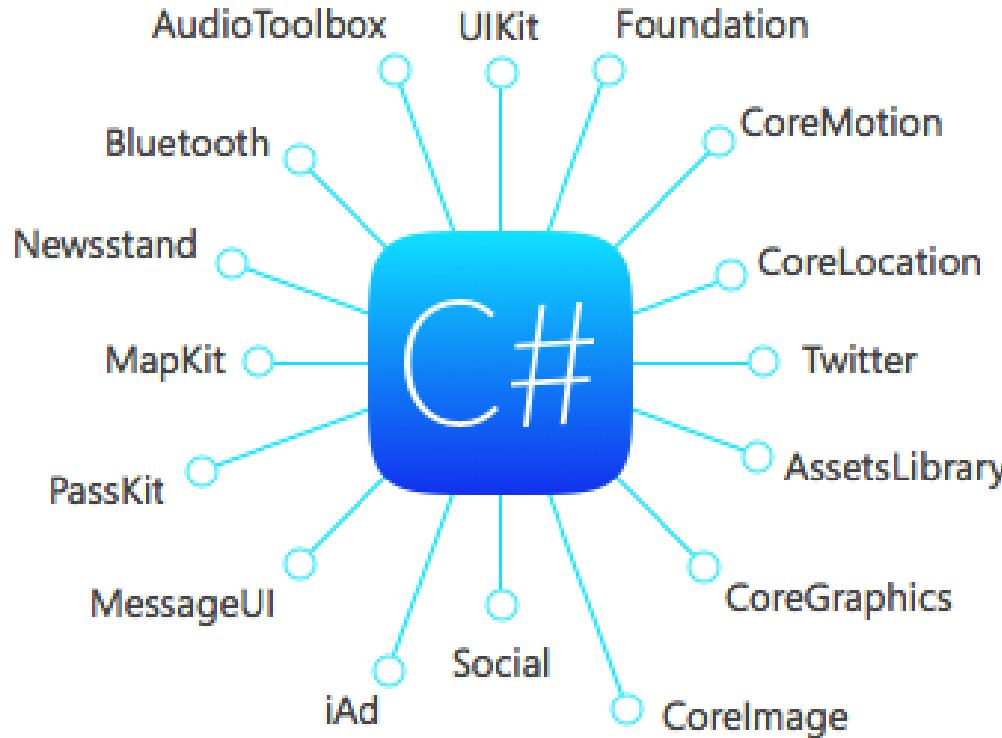
# Xamarin.Forms-szal még több közös kód



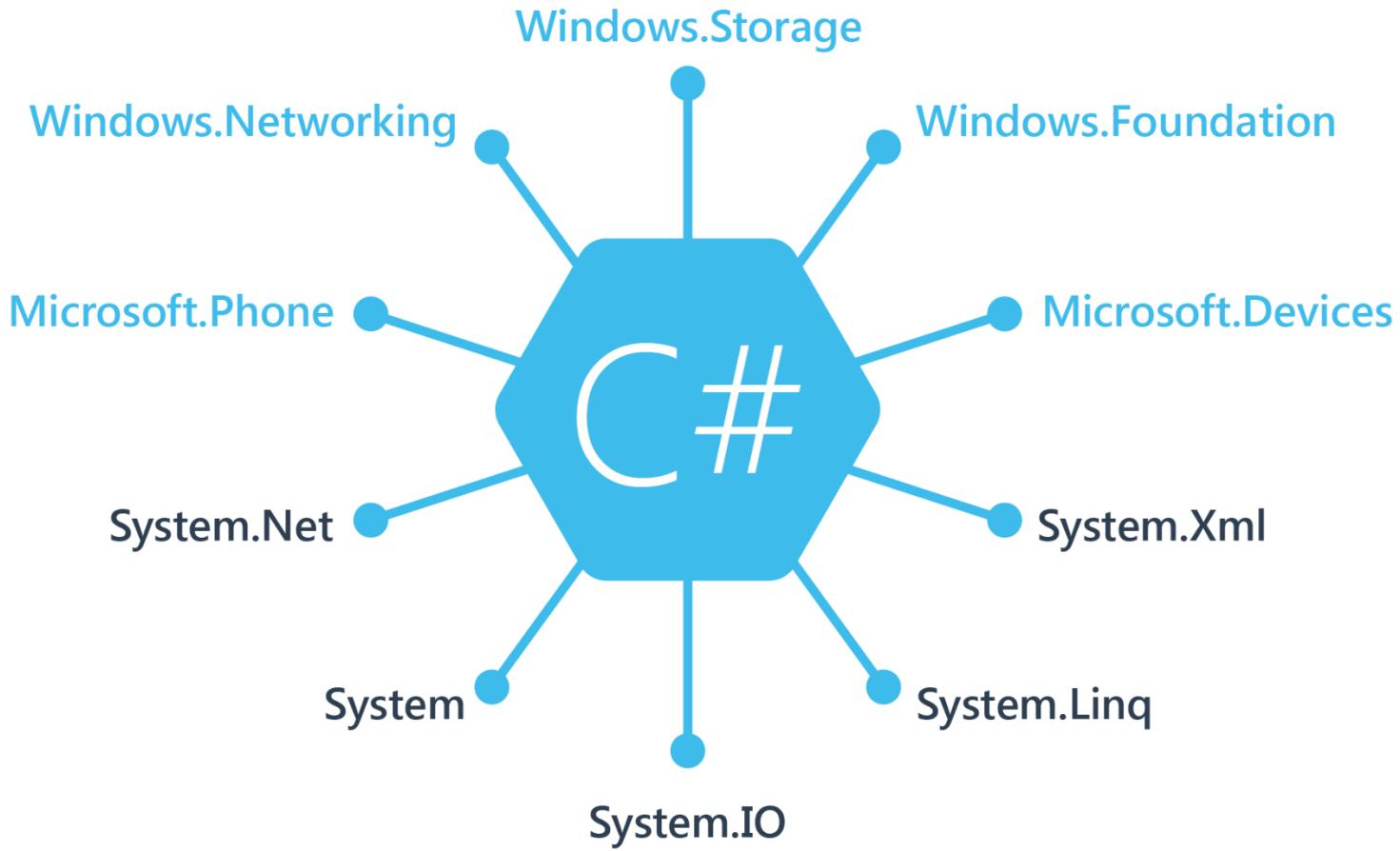
Shared C# codebase • 100% native API access • High performance

# 100%-os API lefedettség

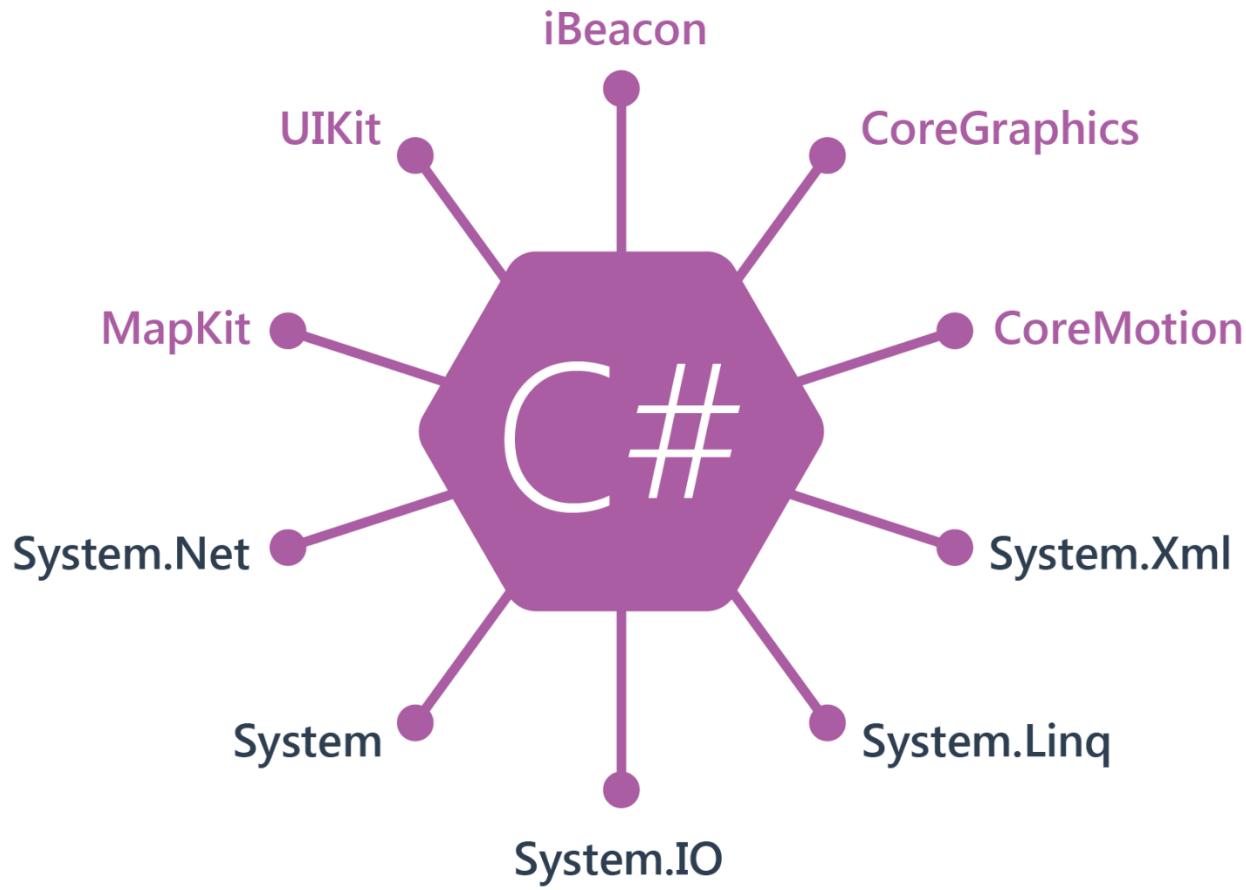
- Bármi, amit meg tudsz írni Objective-C-ben (Swiftben), Java-ban, megírhatod C#-ban, Visual Studio-val és Xamarin-nal!



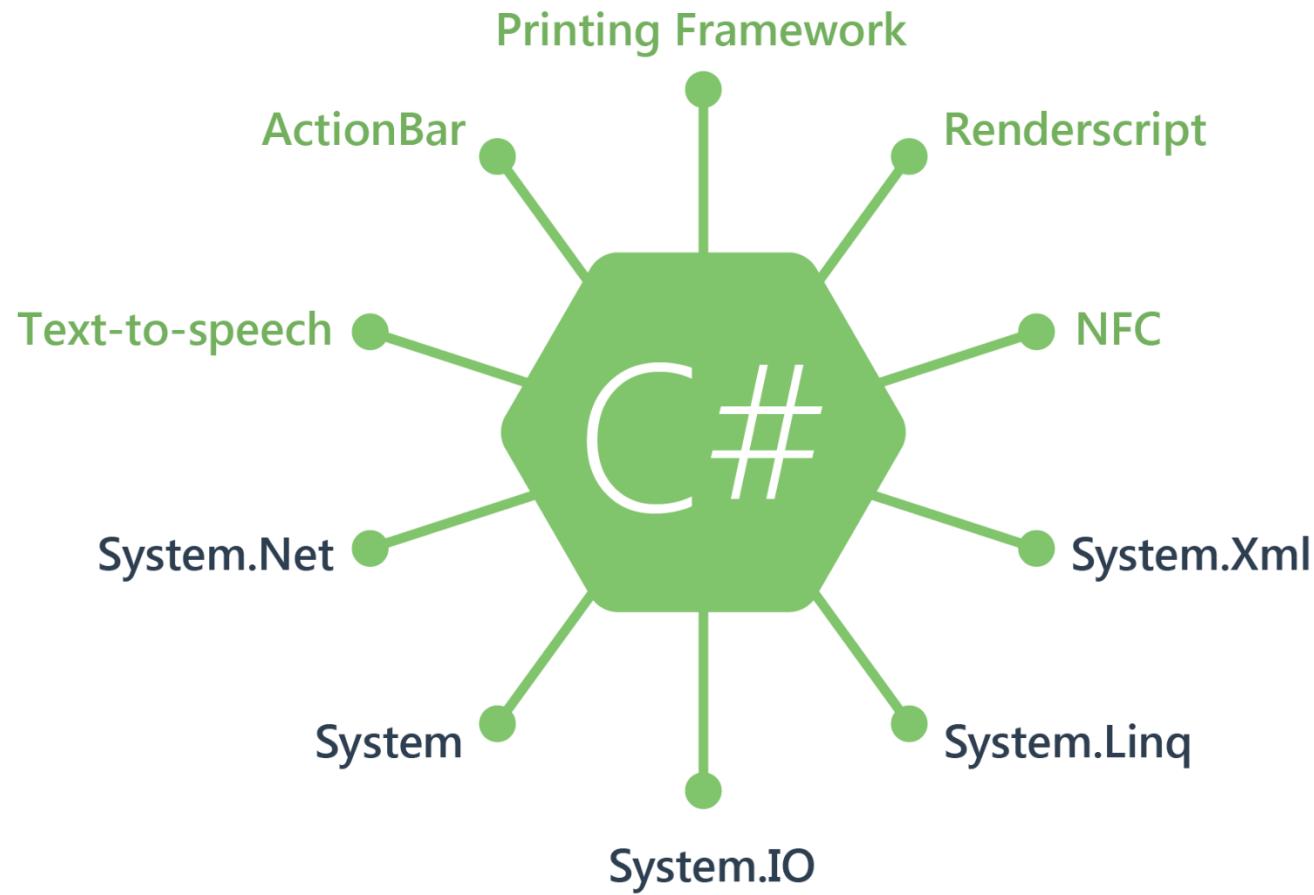
# Windows APIs



# iOS APIs



# Android APIs

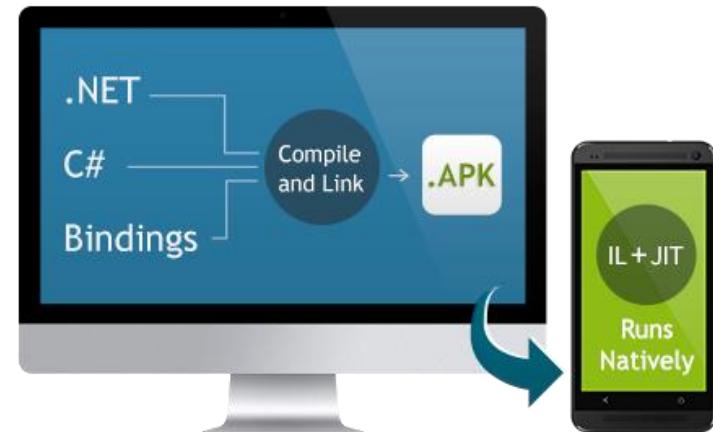
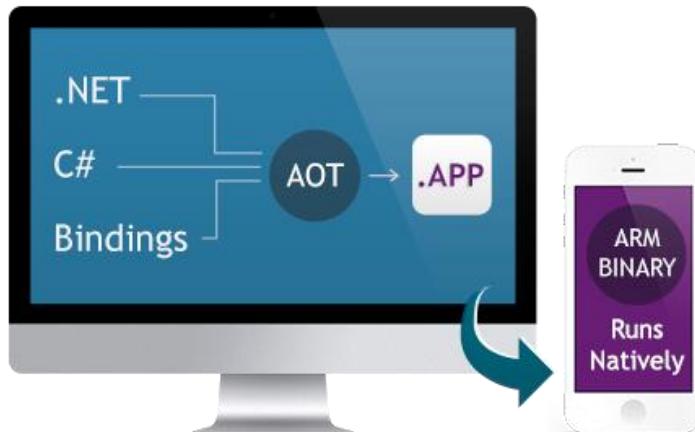


# APIs

- A .NET Framework mindenütt elérhető
- Minden platform API elérhető tetszőleges .NET-es nyelvből csomagolók segítségével
- minden API-t elérünk

# Teljesítmény

- iOS: Ahead-Of-Time fordítás ARM-ra
- Android: Just-In-Time bytecode fordítás
- A teljesítmény a natívvval közel azonos
  - > minden áthívás a natív platformba tartalmaz overheadet





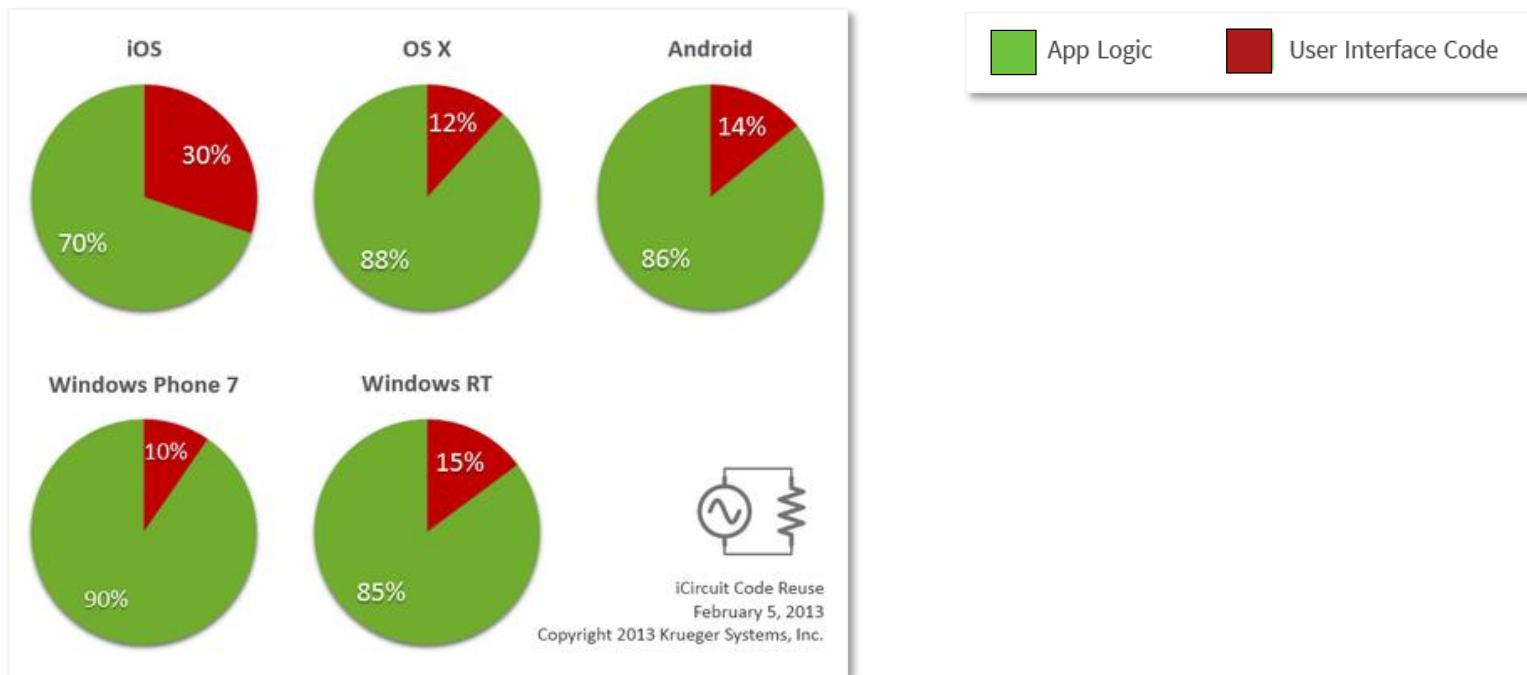
A mítikus kód-újrafelhasználás...

# Kód megosztás általában

- Tipikusan megosztható
  - > Adat elérés
  - > Szolgáltatás elérés
  - > Kliens oldali logikák
  - > View-modellek
- Nem megosztható natív Xamarinnal
  - > Felhasználói felület
    - Ha nem használunk Xamarin.Forms-ot
  - > Esemény kezelők
  - > Platformhoz kötött szolgáltatások

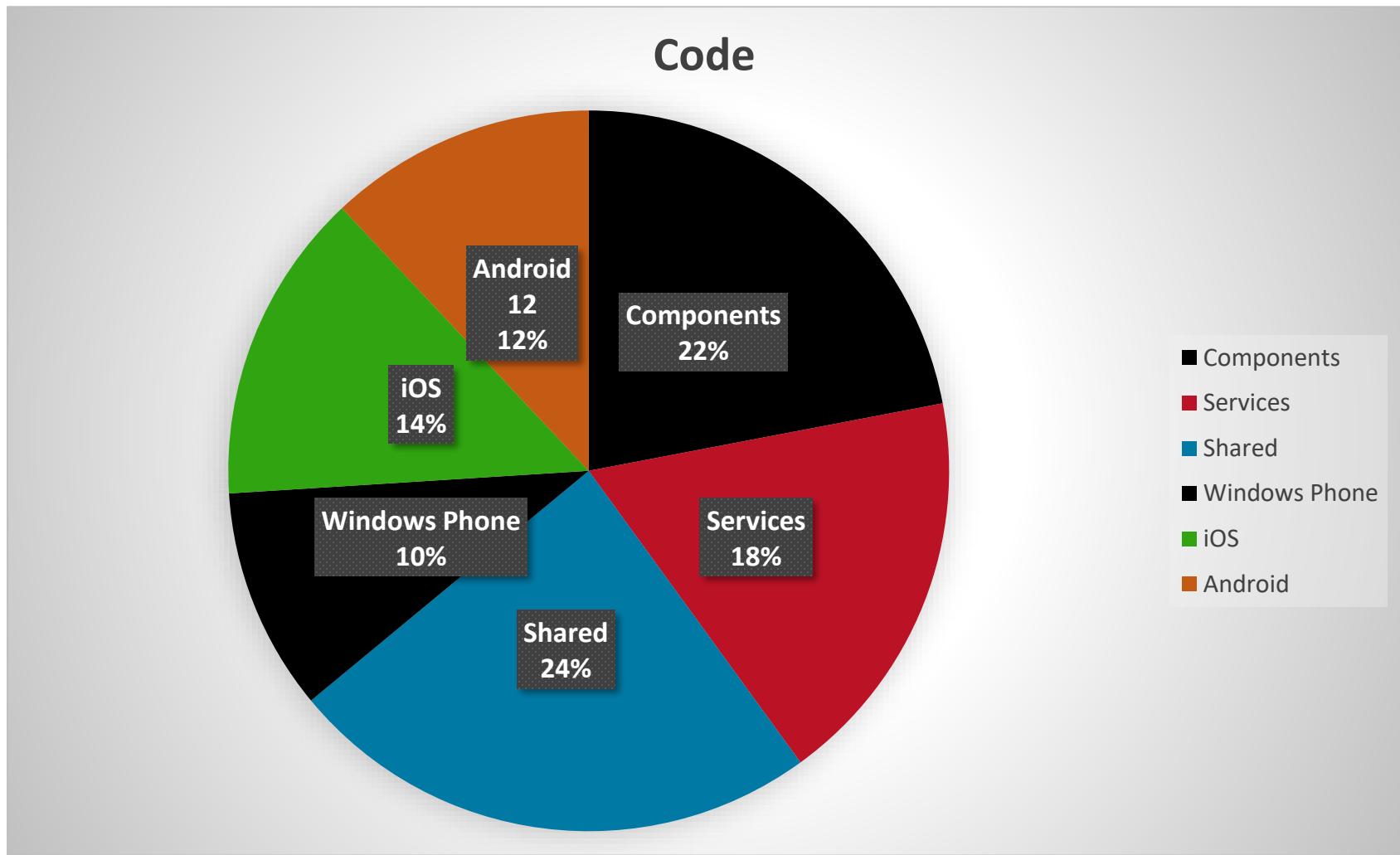
# Code Sharing: Accelerate Development

- Akár 90%-os közös kód
- Meglévő könyvtárak felhasználása
  - > NuGet Support



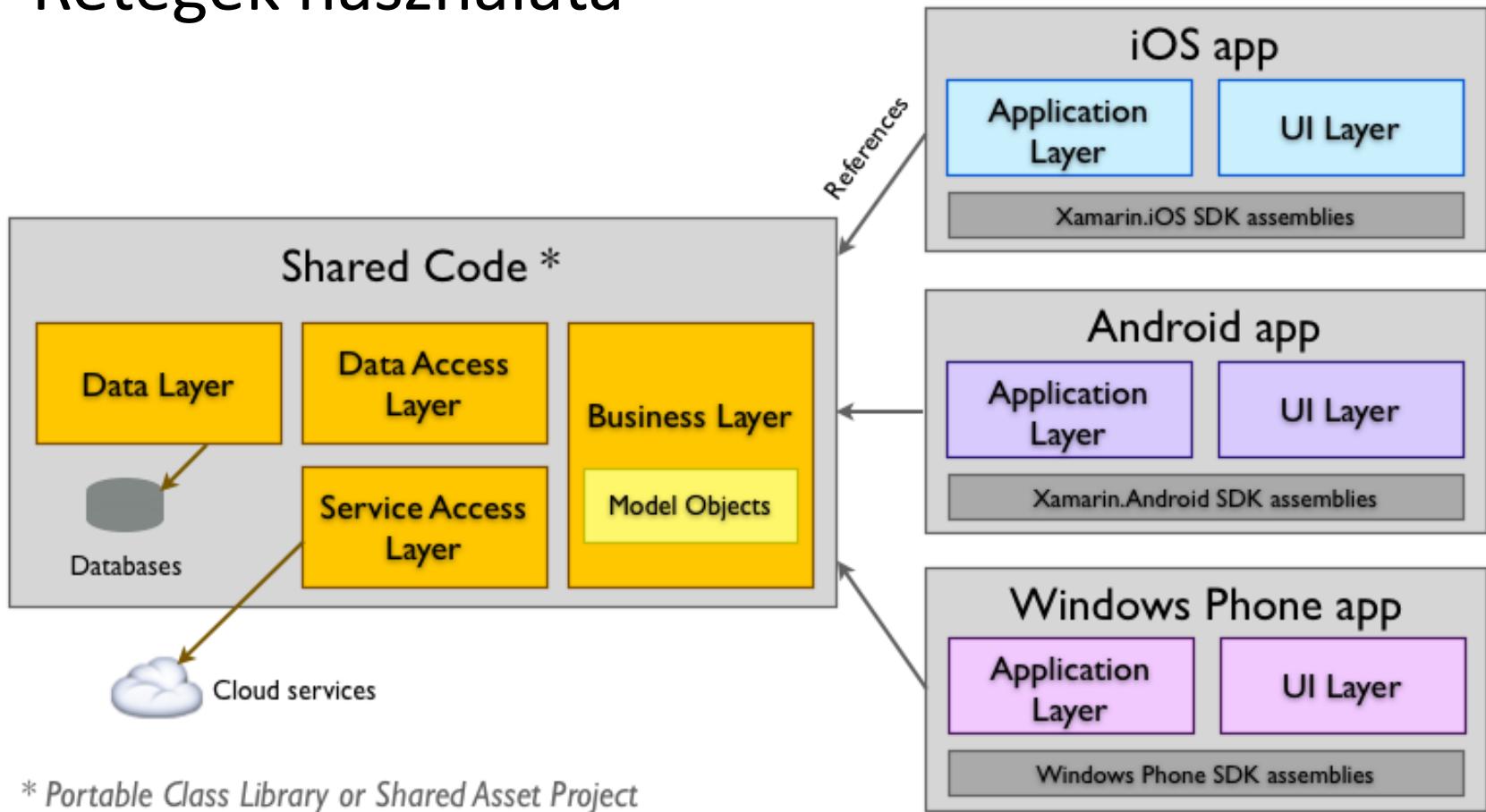
Statisztika: iCircuit

# Statisztikák a Xamarin-tól



# Projekt architektúra

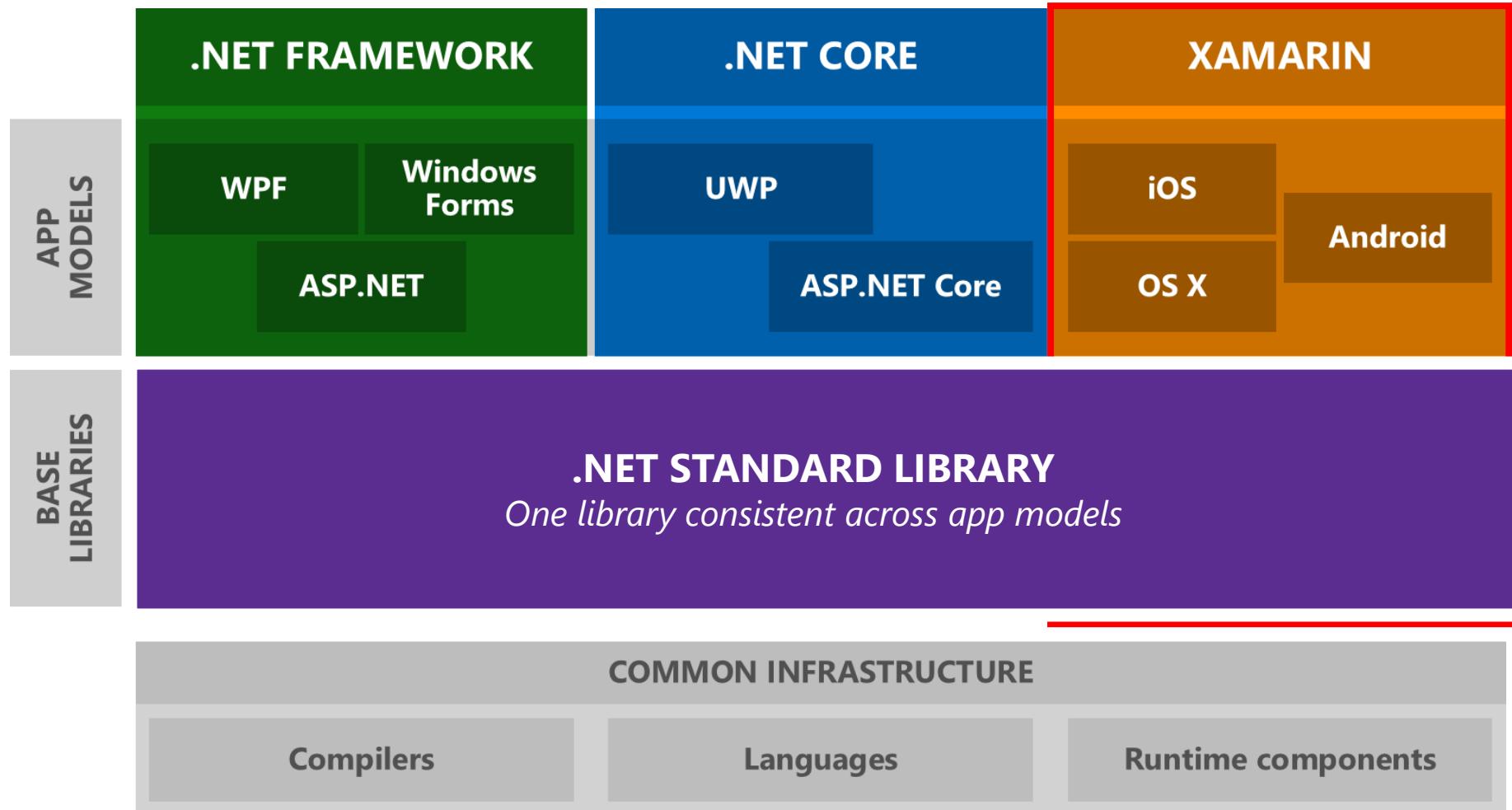
- Rétegek használata



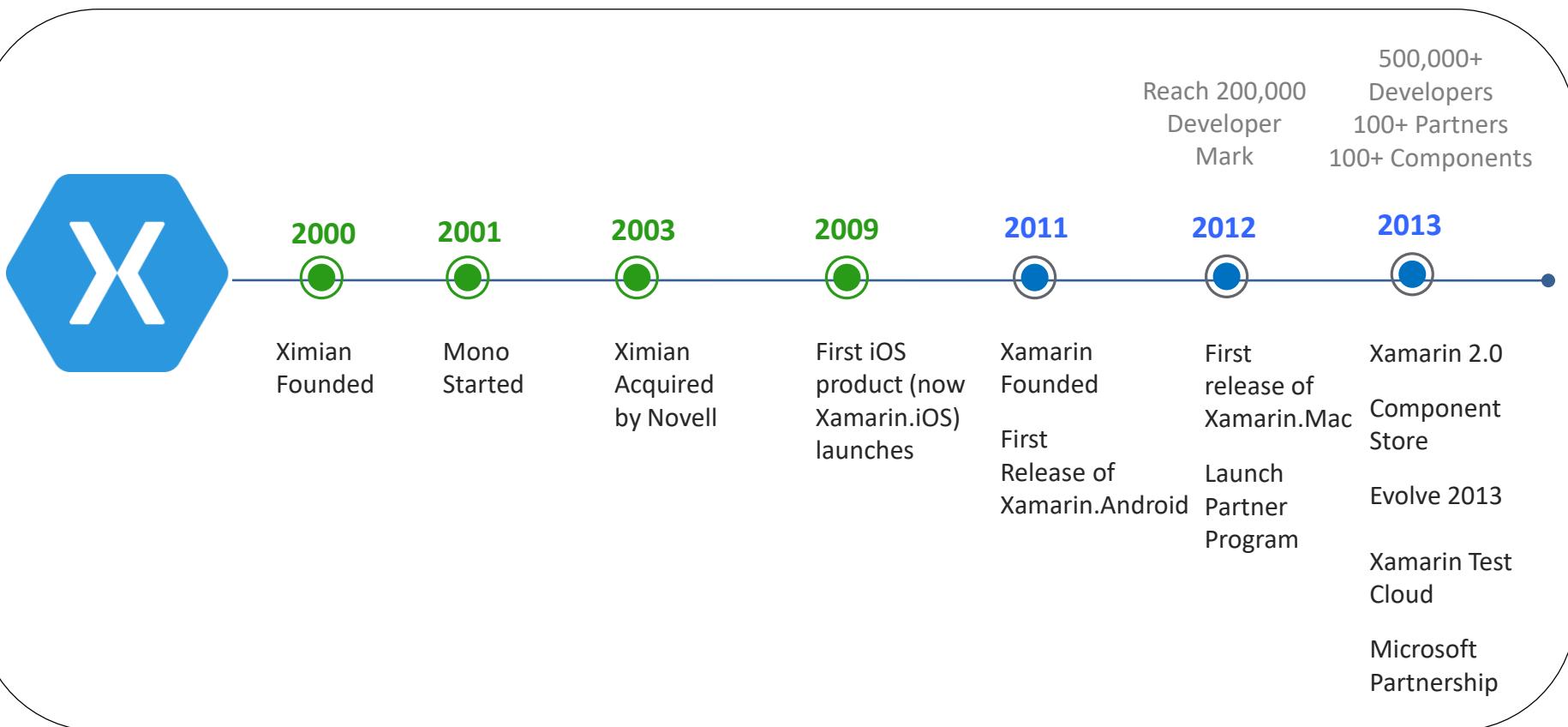
# Megosztott kód formája

- „Shared projects”:
  - > A kód „másolása” a platform projektekbe
  - > #if direktívák használata platform specifikus kódhoz
- .NET Standard Library Project:
  - > Javasolt módszer
  - > Beállítható a .NET Standard verziószám
  - > Készítsünk interféseket, ahol platform specifikus hívásokra van szükség
- PCL: Portable Class Libraries
  - > Régi megoldás, .NET Standard előtt

# .NET az egyes platformokon



# Xamarin történelem



# Kik használják?



# Fejlesztő környezet

- Mac: Xamarin Studio
- Windows: Visual Studio

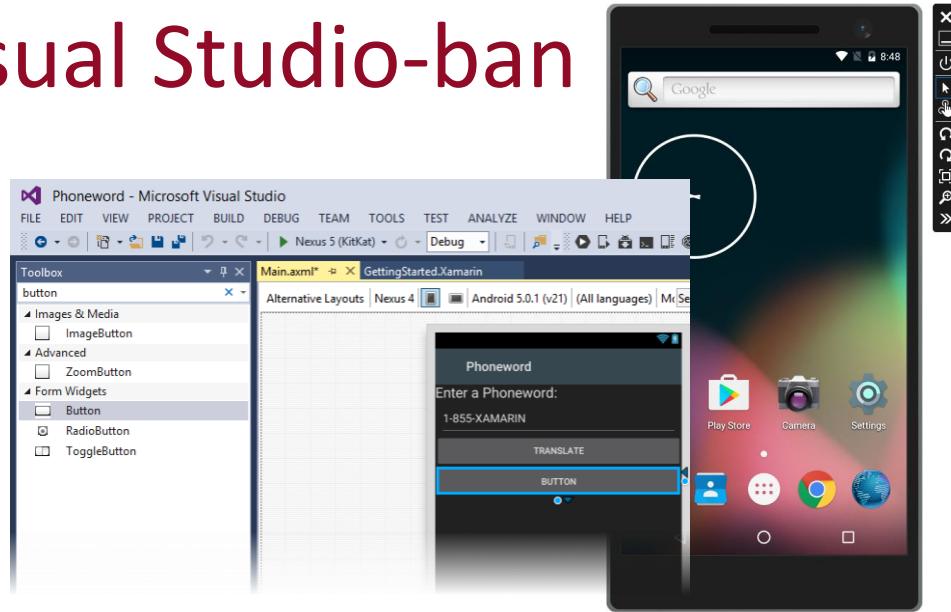


	MACOS	WINDOWS	
Development Environment	XAMARIN STUDIO	VISUAL STUDIO	XAMARIN STUDIO
Xamarin.iOS	Yes	Yes (with Mac computer)	No
Xamarin.Android	Yes	Yes	Yes
Xamarin.Forms	iOS & Android only	Android, Windows, Windows Phone (iOS with Mac computer)	Android only
Xamarin.Mac	Yes	Open project & compile only <a href="#">^</a>	No

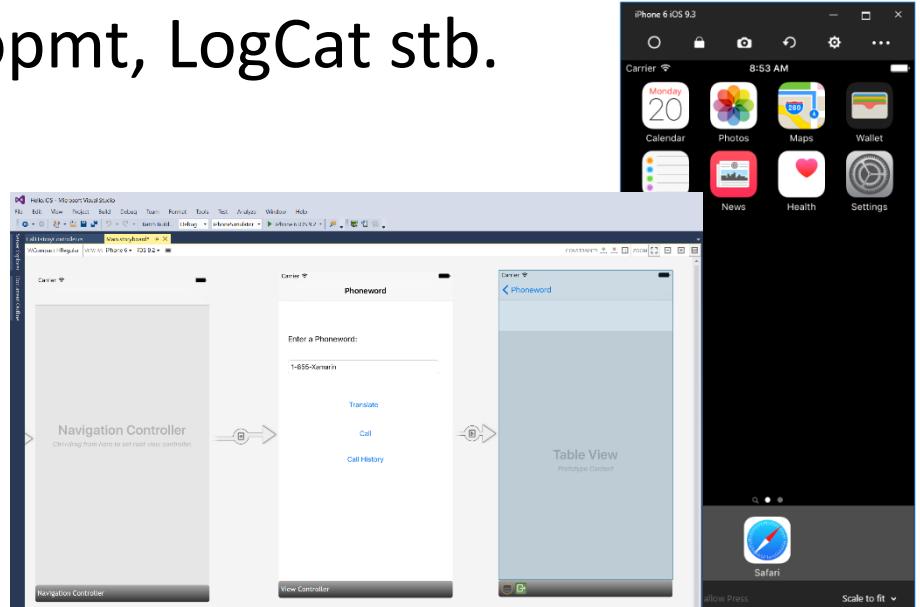
# Xamarin fejlesztés Visual Studio-ban

- Legalább VS 2013

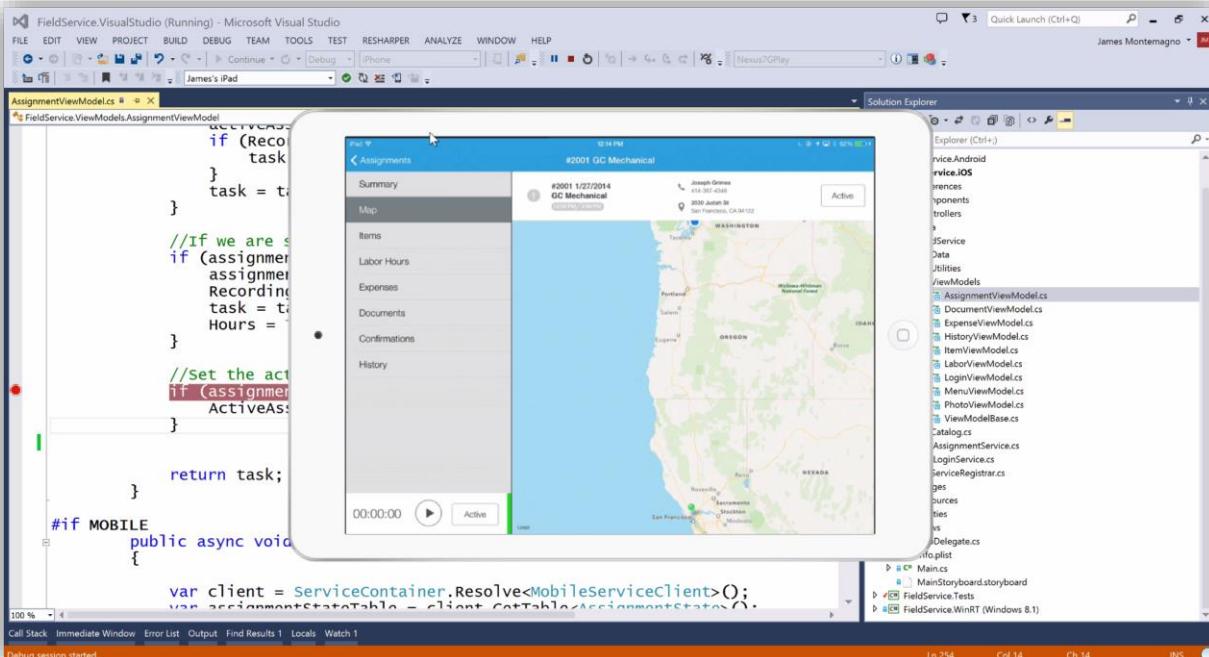
- Android
  - > Emulátor támogatás
  - > deploy eszközre
  - > Android Command Propmt, LogCat stb.



- iOS
  - > Nehézkesebb...
  - > Build csak Mac-en
  - > iOS Simulator



# Visual Studio integráció

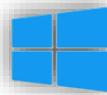


## Egy solution:

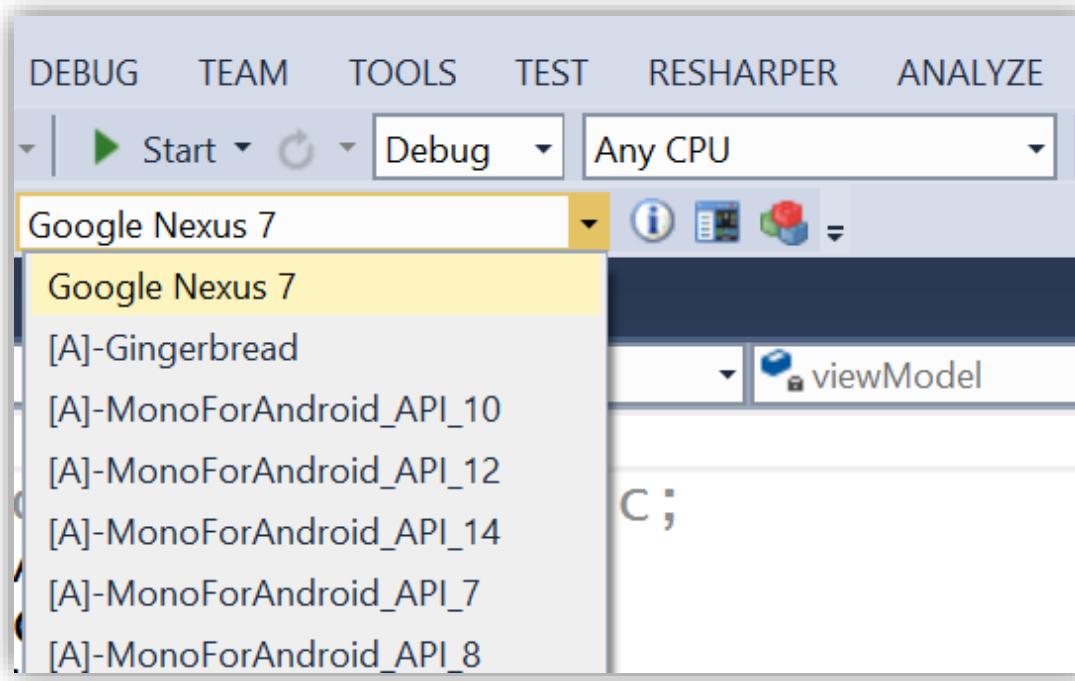
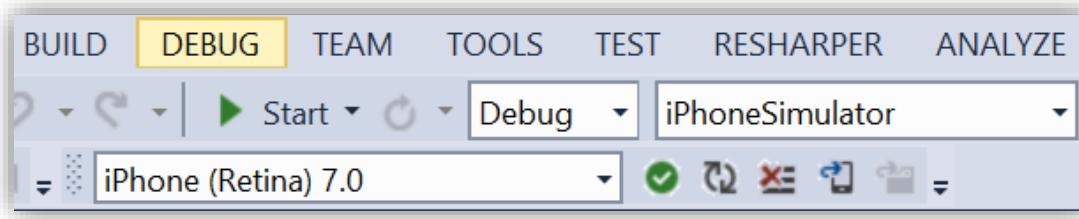
- iOS
- Android
- Windows Phone
- Windows Store

## Microsoft környezet:

- ReSharper
- Team Foundation Server
- code coverage, profiling és egyéb eszközök



# Visual Studio Integration



## Debug:

- Emulátor
- Fizikai eszköz

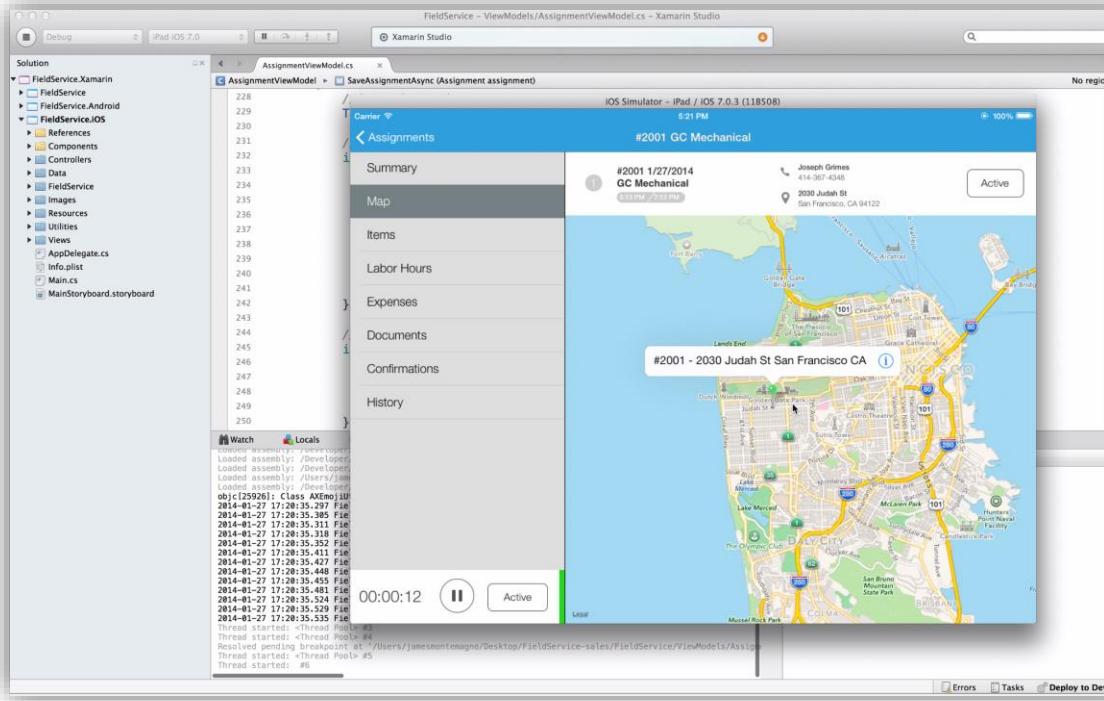
## Eszköztár integráció:

- Státusz
- Napló
- Eszköz lista

**Egy gombos debug!**

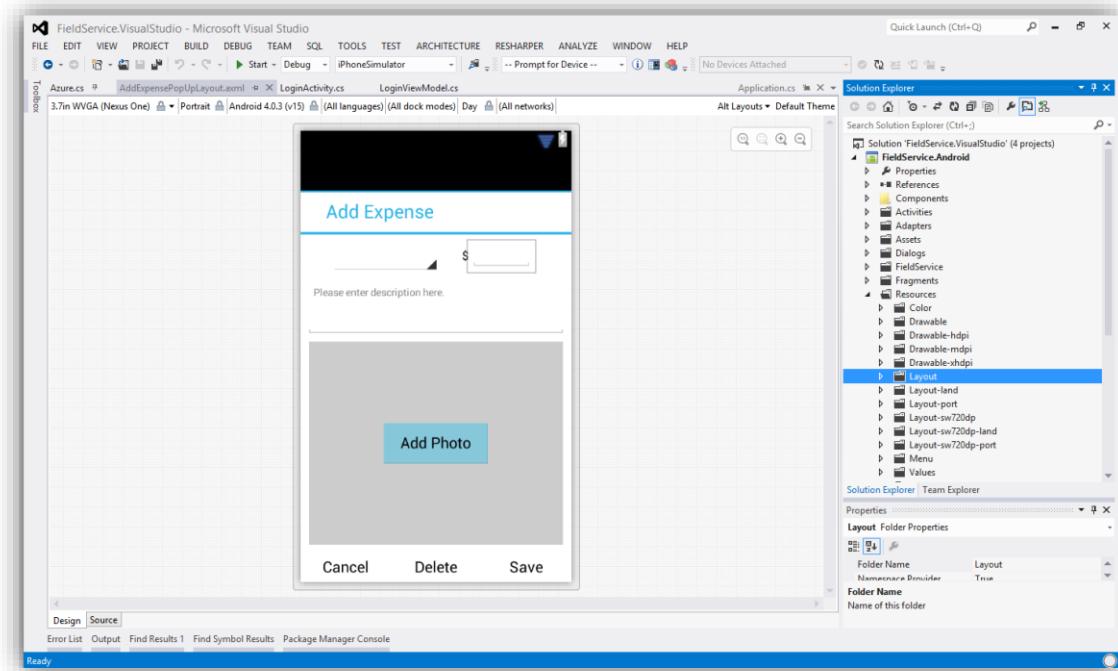
# Xamarin Studio

- Cross platform fejlesztésre optimalizált
- IntelliSense, navigáció
- Git/Subversion integráció
- Android és iOS tervező eszközök
- Debugging:
  - Szimulátor
  - Eszköz

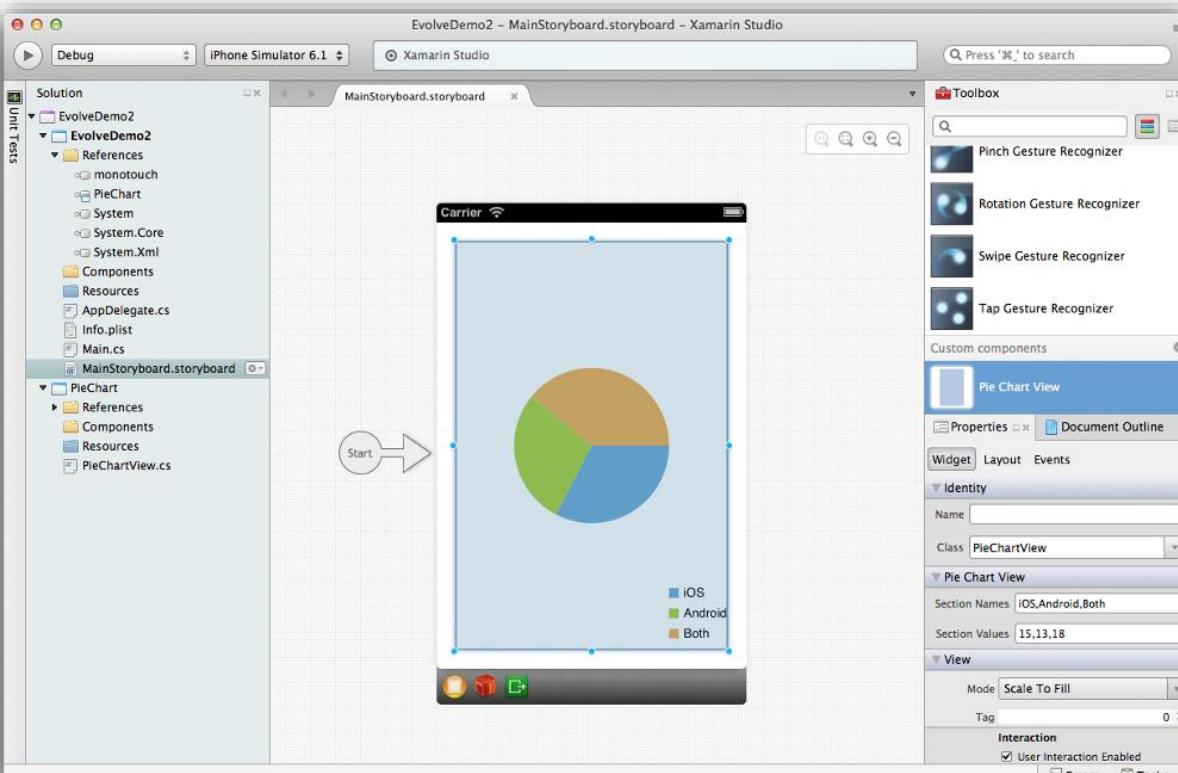


# Android Designer

- Nagyon jó tervező
  - Xamarin Studio
  - Visual Studio
- Drag & drop
- Több képernyő méret, felbontás és Android verzió támogatása
- Mindez standard Android XML fájlokkal



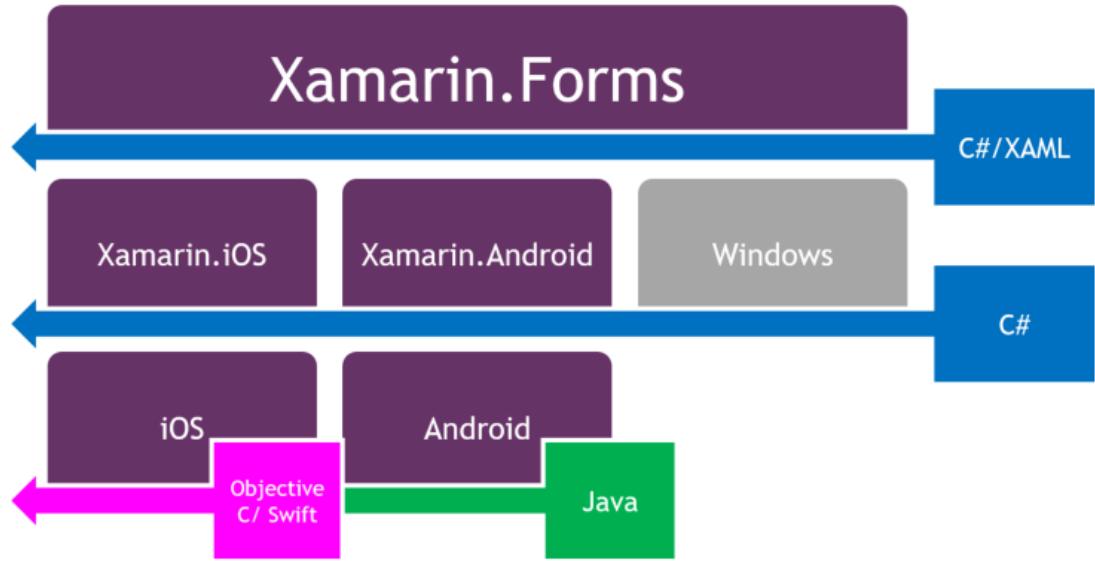
# iOS Designer



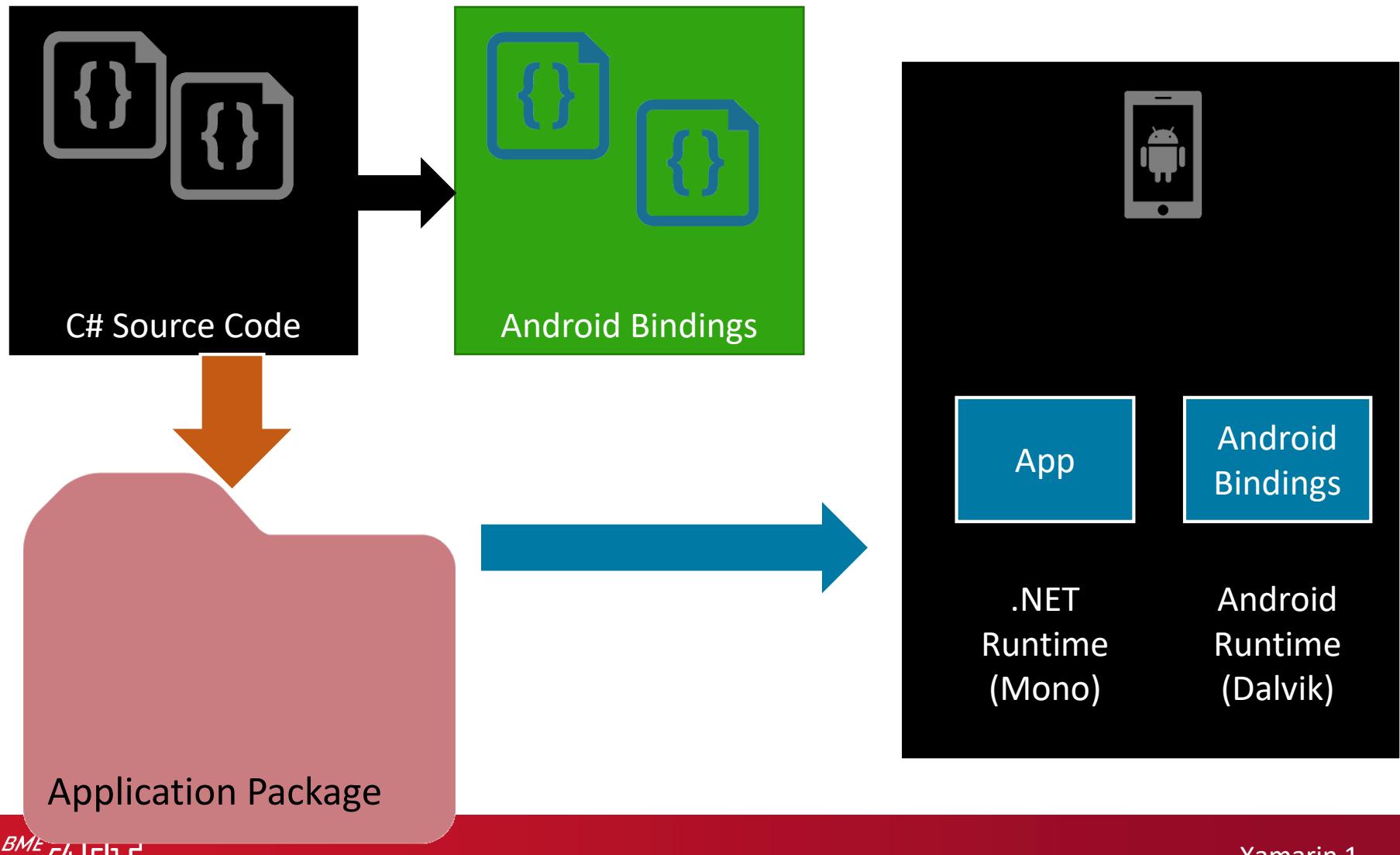
- iOS Designer
  - Xamarin Studio
  - Visual Studio
- Hasonló a Visual Studio-hoz
- Támogatja az összes UIKit elemet
- Saját és 3<sup>rd</sup> party komponensek támogatása
- Elő előnézet

# Alapfogalmak

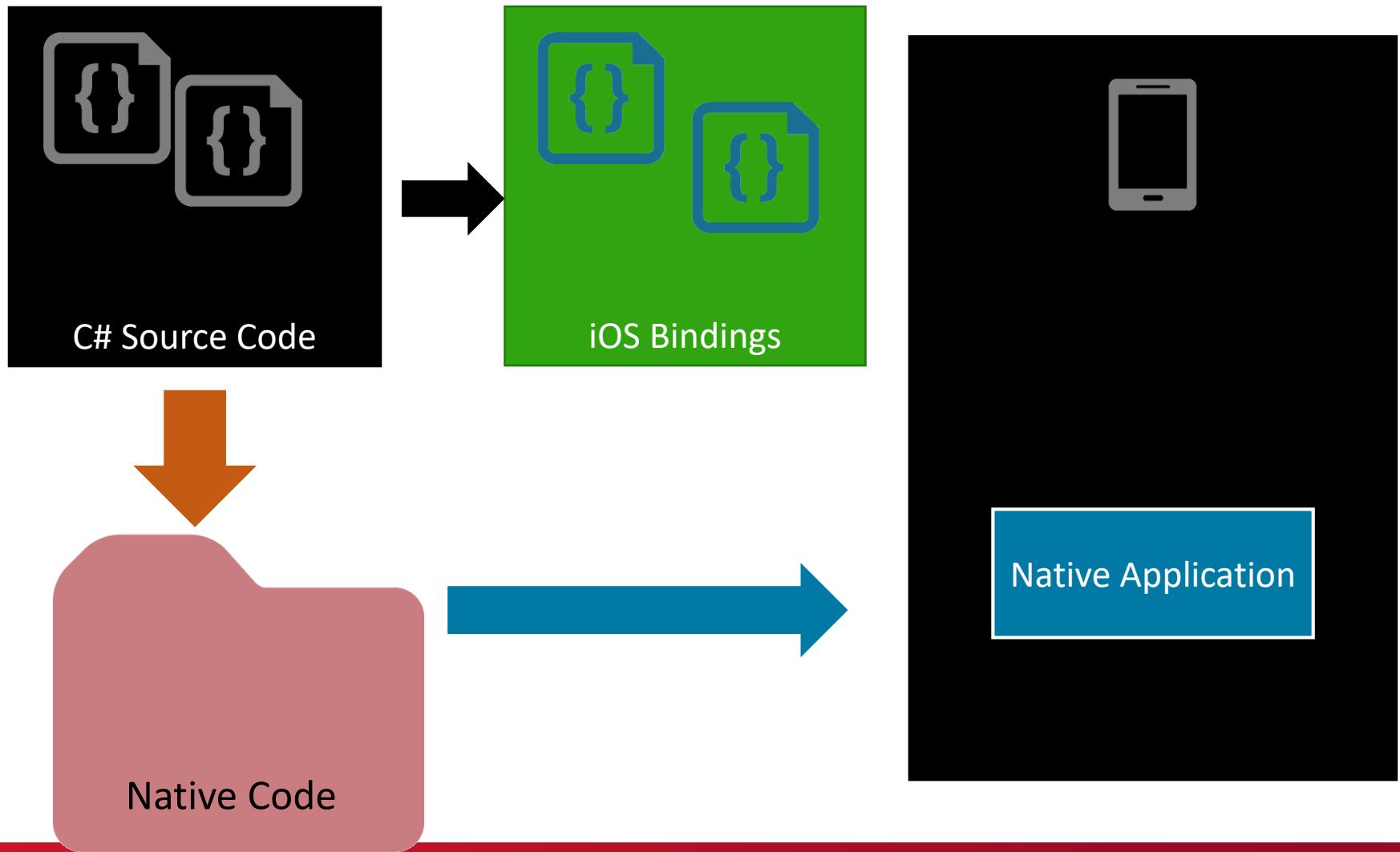
- Xamarin
- Mono
- Xamarin.Android
- Xamarin.iOS
- Xamarin.Mac
- C# de a szokásos Android és iOS architektúra
  - > MVP és MVC
  - > minden platform specifikus API elérhető
- Xamarin.Forms



# C# on Android



# C# on iOS



# Platform virtualizáció (language binding)

- Xamarin.iOS, Xamarin.Android: ezek végzik el a hívást a natív platform API-ba a .NET-es hívásból
  - > .NET-es kód: IL kód (már nem C#)
- iOS: az IL kódot Mac-en fordítja natív ARM kóddá úgy, mint az Objective-C fordító
- Android: az IL kódot a Mono CLR futtatja

# Xamarin tervezési szempontok

- Kövesse a .NET Framework Design Guideline-okat
- Származhassunk az Android / Objective-C osztályokból
- Objective-C / JavaBean tulajdonságok C#-ban is tulajdonságokként jelenjenek meg
- Használjunk C# metódus referenciákat egy-metódusos interfészek és protokollok helyett
- Típusos API-t használjunk ami biztonságot és IntelliSense-t ad
- IDE-ben is működjön a dokumentáció
- A gyakori Java / Objective-C feladatok legyenek könnyűek, a nehezek legyenek lehetségesek
- Tetszőleges Java / Objective-C könyvtár meghívható legyen

# Pozíció meghatározás

- A Xamarin dokumentáció egy-az-egyben lehivatkozza az Android doksit
  - > Ott van az „autentikus” információ
- Android:

```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
```

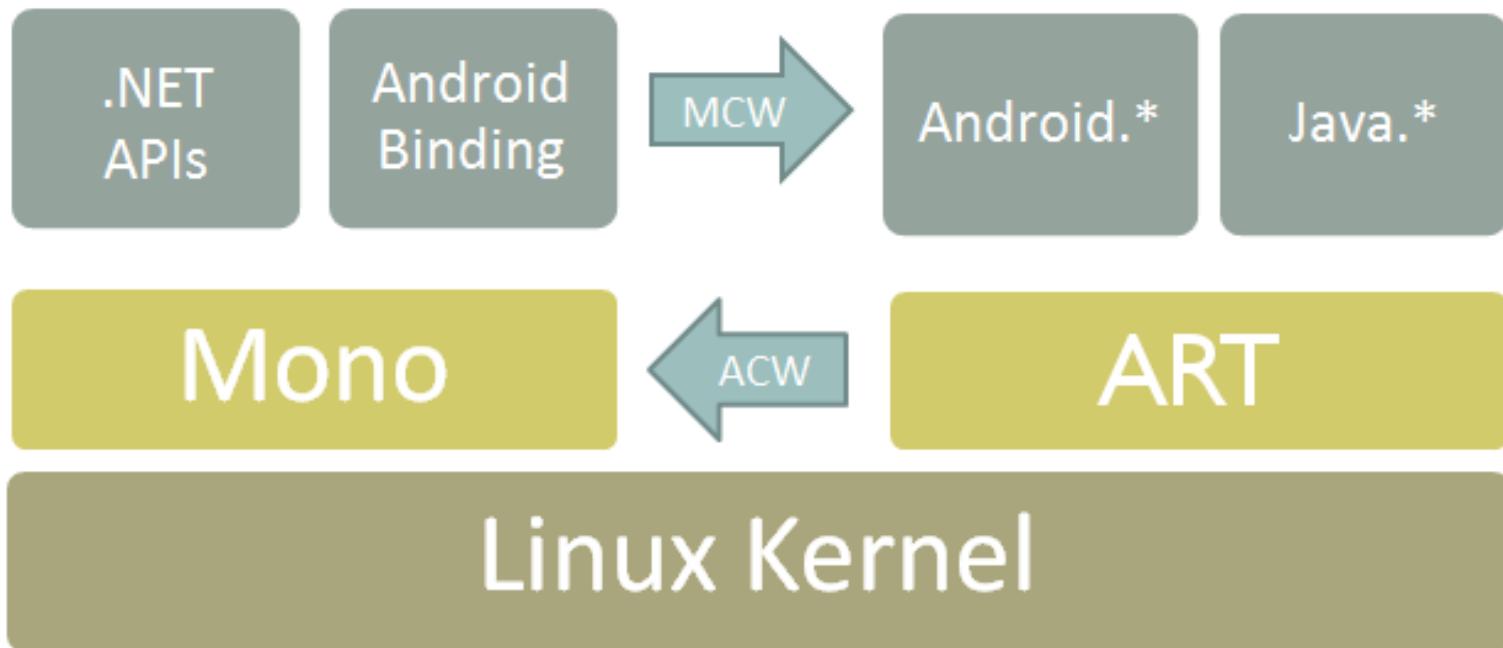
- Xamarin:

```
LocationManager locMgr;
...
locMgr = GetSystemService (Context.LocationService) as LocationManager;
```

- Csak nyelvi különbségek látszanak...

# Xamarin . Android

- Egy alkalmazást két futtatókörnyezet szolgál ki
- Mono
  - > A CLR és HAL C-ben van, a kernelt hívja
  - > Az osztálykönyvtár C#, így érhető el a fájlrendszer stb.
  - > A platform funkciók jelentős része csak az Android Runtime Java API-n érhető el – csomagoló osztályok



# Android Callable Wrapper

- Android/Java oldal hív bele a .NET-es kódba
- JNI-n alapul: Java Native Interface
  - > C jellegű hívásokat tesz lehetővé a javas kódból
- A felülírt virtuális metódusok (override) hívásai is ezen keresztül történnek
  - > Így támogatja .NET-es öröklést is
- A fordítási folyamat részeként automatikusan generálódnak minden .NET-es osztályhoz, ami a **Java.Lang.Object**-ből származik

# Managed Callable Wrapper

- .NET-es oszálykönyvtár, ami teljes androidos névteret lefedi
- JNI kommunikáció, másik irány
- minden .NET-es objektum hivatkozik egy Java-s objektumra
  - > A .NET-es megszűnésekor (Dispose) engedi el a javas referenciát, így a JVM GC tudja törölni

# Egy példa: MCW, ACW

- Egy Android alkalmazás központi elemei az Activityk
- **MCW:** A javas Activity osztály (leszármazottaival, œseivel, interfészeivel) megjelenik a Mono.Android.dll-ben, .NET-es osztályként
  - > Előre generált, a Xamarin része
- Leszármazunk az Activityből és felülírjuk a megfelelő virtuális metódusokat .NET-ben
- **ACW:** a build folyamat során az osztályunkból a Xamarin generál egy Java kódot, ami a felülírt metódusban áthív a .NET-es kódunkba

# Gyűjtemények leképezése

- Inkább a javas megvalósításokat és ne a .NET-es gyűjteményeket használjuk!
  - > minden más gyűjtemény implementáció másolást fog használni, amikor átkerül a Java oldalra!

Java Type	System Type	Helper Class
<a href="#">java.util.Set&lt;E&gt;</a>	<a href="#">ICollection&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaSet&lt;T&gt;</a>
<a href="#">java.util.List&lt;E&gt;</a>	<a href="#">IList&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaList&lt;T&gt;</a>
<a href="#">java.util.Map&lt;K,V&gt;</a>	<a href="#">IDictionary&lt;TKey,TValue&gt;</a>	<a href="#">Android.Runtime.JavaDictionary&lt;K,V&gt;</a>
<a href="#">java.util.Collection&lt;E&gt;</a>	<a href="#">ICollection&lt;T&gt;</a>	<a href="#">Android.Runtime.JavaCollection&lt;T&gt;</a>

# Két irányú gyűjtemény kezelés

- A .NET-es gyűjtemény másolódik, így a java osztályon való módosítás nem az eredeti példányon dolgozik!

```
// This fails:
var badSource = new List<int> { 1, 2, 3 };
var badAdapter = new ArrayAdapter<int>(context, textViewResourceId, badSource);
badAdapter.Add (4);
if (badSource.Count != 4) // true
 throw new InvalidOperationException ("this is thrown");

// this works:
var goodSource = new JavaList<int> { 1, 2, 3 };
var goodAdapter = new ArrayAdapter<int> (context, textViewResourceId, goodSource);
goodAdapter.Add (4);
if (goodSource.Count != 4) // false
 throw new InvalidOperationException ("should not be reached.");
```

# Tulajdonságok, események, ...

- A get... / set... metódusok automatikusan .NET-es tulajdonságokká képeződnek
- Eseményekre feliratkozás működik a .NET-es metódus referenciaikkal (delegate)
- Egyéb leképezések

# Külső komponensek használata

- Java Bindings Library készítése
  - > Automatikus, becsomagolja a hivatalos jar fájlban lévő osztályokat
    - Hasonlóan a Java rendszer osztályokhoz
  - > Ez is JNI-vel dolgozik mélyen...
- JNI alapú megoldás
  - > Nagyon jól finomhangolható de sok munka
- Forráskód szintű portolás
  - > Nagyon sok munka de van automatikus nyelvi konverter

# Android build folyamat

- Debug
  - > Az alaposztály könyvtár és a futtató környezet az Android kötéssel együtt nem a csomag része
  - > Kisebbek a telepítendő szerelvények, gyorsabb a fejlesztési-tesztelési ciklus
- Release
  - > minden, a BCL és a futtató környezet is része a telepítő csomagnak

# Mono

- A .NET-es kód nyílt forráskódú futtatókörnyezete
- Garbage Collection
  - > A .NET-es osztályokat a Mono GC takarítja
  - > A javas osztályokat a JVM takarítja
  - > „Peer objects”: IJavaObject interfész – akkor szűnik meg, amikor minden oldalon elengedik az összes hivatkozást (explicit: Dispose hívás)
  - > Teljesítmény kritikus!
- JIT, ...

# iOS / Objective-C specialitások

- Target-action minta – .NET-es események
- Protokoll – interfész opcionális metódusokkal
- Message – „metódus hívás”
- Delegate – „delegáció” tervezési minta, NEM a .NET-es delegate-re utal (metódus referencia)

# Tulajdonságok, események, ...

- Eseményekre (event) feliratkozás működik a .NET-es metódus referenciaikkal (delegate)
- Protokollok
  - > Absztrakt osztályokká fordulnak
  - > A szükséges (virtuális) metódust felül tudjuk írni

# Protokollhoz tartozó osztály példa

```
[Register ("MKAnnotation"), Model]
public abstract class MKAnnotation : NSObject
{
 public abstract CLLocationCoordinate2D Coordinate
 {
 [Export ("coordinate")]
 get;
 [Export ("setCoordinate:")]
 set;
 }

 public virtual string Title
 {
 [Export ("title")]
 get
 {
 throw new ModelNotImplementedException ();
 }
 }
}
```



# Delegate (iOS nomenklatúra)

- Egy objektum egy feladatot egy másik objektumra bíz
  - > Például a felhasználó kattintás kezelését az iOS vezérlő rábízza egy általunk implementált osztályra
- A kezelhető feladatokat a protokoll (interfész) adja meg, azt implementáljuk
- A vezérlő egy Delegate mezőjén hivatkozunk a protokollt megvalósító objektumra

# Delegate minta, belső osztályval

```
public partial class Protocols_Delegates_EventsViewController : UIViewController
{
 SampleMapDelegate _mapDelegate;
 ...
 public override void ViewDidLoad ()
 {
 base.ViewDidLoad ();

 //set the map's delegate
 _mapDelegate = new SampleMapDelegate ();
 map.Delegate = _mapDelegate;
 ...
 }
 class SampleMapDelegate : MKMapViewDelegate
 {
 ...
 }
}
```

```
public class SampleMapDelegate : MKMapViewDelegate
{
 public override void DidSelectAnnotationView
 (MKMapView mapView, MKAnnotationView ann
 {
 var sampleAnnotation =
 annotationView.Annotation as Sample
 if (sampleAnnotation != null) {
 //demo accessing the coordinate of
 //zoom in on it
 mapView.Region = MKCoordinateRegion(
 sampleAnnotation.Coordinate, 50
);
 //demo accessing the title of the s
 Console.WriteLine ("{0} was tapped"
 }
 }
}
```

# WeakDelegate

- Nem kell a protokollt (interfészt) követni
- Tetszőleges osztály megvalósíthatja a protokoll tetszőleges metódusát

```
public partial class Protocols_Delegates_EventsViewController : UIViewController
{
 ...
 public override void ViewDidLoad ()
 {
 base.ViewDidLoad ();
 //assign the controller directly to the weak delegate
 map.WeakDelegate = this;
 }
 //bind to the Objective-C selector mapView:didSelectAnnotationView:
 [Export("mapView:didSelectAnnotationView:")]
 public void DidSelectAnnotationView (MKMapView mapView,
 MKAnnotationView annotationView)
 {
 ...
 }
}
```

# Delegate-ek eseményekként

- A .NET-es oldal eventekként is látja az eseményeket, hagyományosan fel lehet

```
map.DidSelectAnnotationView += (s,e) => {
 var sampleAnnotation = e.View.Annotation as SampleMapAnnotation;
 if (sampleAnnotation != null) {
 //demo accessing the coordinate of the selected annotation to
 //zoom in on it
 mapView.Region = MKCoordinateRegion.FromDistance (
 sampleAnnotation.Coordinate, 500, 500);

 //demo accessing the title of the selected annotation
 Console.WriteLine ("{0} was tapped", sampleAnnotation.Title);
 }
};
```

# Xamarin - iOS

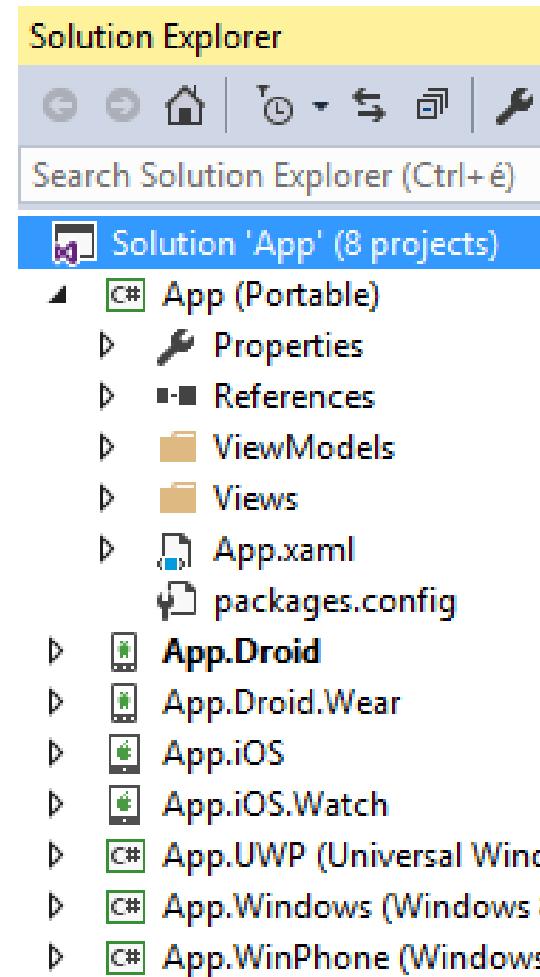
- ObjCRuntime
  - > Leképezés a .NET-es és iOS világ között
  - > Foundation: alaptípusok az interophoz
  - > Általában az Objective-C-s típusokat követi, kivétel:
    - string NSString helyett
    - Típusos tömb NSArray helyett
    - RectangleF, ... - CGRect, ... helyett
- UIKit - vezérlők
- OpenGL ES - grafika

# GC

- minden natív (nem felügyelt) típus elérhető a Handle tulajdonságon
- Dispose hívás rögtön megszünteti a natív objektumot, nem kell várni a takarításra

# Projekstruktúra

- PCL / .NET Standard / Shared
  - > Teljes egészében platform független kód
  - > Az alkalmazás core része
- Platform specifikus projektek
  - > Android, iOS, UWP, Windows Phone
  - > Jellegzetes platform specifikus kód megosztások
    - UWP, Windows 8.1, Windows Phone 8.1
    - iOS, Droid



# Alkalmazás építése

- Akár üres solution fájlból is indulhatunk
- Közös kód: shared project, PCL, Standard Lib
  - > Shared project: másolja a fájlokat, tipikusan #if-eket fogunk használni
  - > PCL/.NET Standard: speciális projekt típus, a fordított dll minden .NET-es platformon betölthető: Xamarin.iOS, Xamarin.Android, WPF, Windows Phone, Xbox, ...

# Több platform

- Vannak közös koncepciók, amik minden platformra igazak
  - > Tabok, menük használata
  - > Listák, görgetés
  - > Részletes nézet
  - > Szerkesztés
  - > Navigáció vissza
  - > Élet ciklus követés

# Több platform

- Platform specifikus területek
  - > Képernyő méretek
  - > Navigációs „metafórák”
    - Manapság erős a konvergencia
  - > Klaviatúra, érint, gesztus kezelés
  - > Push notification kezelés
    - Különböző lehetőségek, például élő csempék stb.

# Több platform

- Eszköz specifikus funkciók
  - > Egy funkció elérhető-e az adott eszközön
- Például
  - > Kamera
  - > Geo-pozíció
  - > Gyorsulásmérő, giroszkóp, iránytű
  - > Twitter, Facebook integráció
    - Beépített a platformba vagy külön API kell hozzá
  - > Near Field Communications (NFC)
  - > Fizetés API

# Platform absztrakció

- Alaposztályok a közös kódban
  - > Alap funkció, logika közös
- Xamarin.Forms
  - > Közös UI – ahol lehet
- A Xamarin BCL lehetővé teszi bizonyos funkciók elérését közös kódból
  - > Media Picker: fénykép készítése, média elem kiválasztása
  - > Geolokáció kezelés
  - > Névjegyek kezelése
  - > Tetszőlegesen bővíthető

# Amikor nem elég az absztrakció

- Feltételes fordítás
  - > \_\_MOBILE\_\_ : iOS, Android
  - > \_\_IOS\_\_ : iOS eszközök
  - > \_\_ANDROID\_\_ : Android eszközök
  - > \_\_ANDROID\_XX\_\_ specifikus Android verzió
    - Például: \_\_ANDROID\_11\_\_
  - > \_\_WINDOWS\_PHONE\_\_ Windows Phone eszközök

# Fordítási direktíva példa

```
public static string DatabaseFilePath {
 get {
 var filename = "MwcDB.db3";
#if SILVERLIGHT
 var path = filename;
#else

#if __ANDROID__
 string libraryPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal); ;
#else
 // we need to put in /Library/ on iOS5.1 to meet Apple's iCloud terms
 // (they don't want non-user-generated data in Documents)
 string documentsPath = Environment.GetFolderPath (Environment.SpecialFolder.Personal);
 // Documents folder
 string libraryPath = Path.Combine (documentsPath, "..", "Library");
#endif
 var path = Path.Combine (libraryPath, filename);
 #endif
 return path;
 }
}
```

# Xamarin 2

< Albert István >



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Tartalom

## Xamarin Forms



Xamarin 2

Xamarin 2

## Layout<T> osztály : Layout

- `IList<T> Children`: T típusú gyerek vezérlők
  - > T a View-ból származik
  - > A Layout vezérlők egymásba ágyazhatók

Xamarin 2

Xamarin 2

## Navigáció

- Platformonként különbözik a navigációs paradigmá
- A Page osztályok rejtik el a különbségeket vizuálisan



## INotifyPropertyChanged interfész

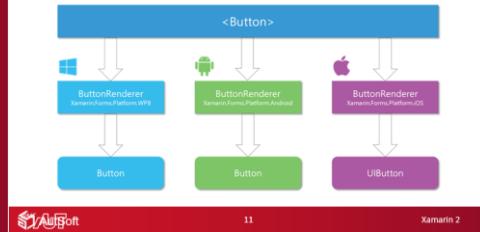
- A XAML-ben szokásos INPC interfész
- Itt is MVVM-et használunk
- Itt is vannak keretrendszerök!

Xamarin 2

Xamarin 2

## ViewRendererek

- XAML elemekből natív nézet
- Ez már platform specifikus logika

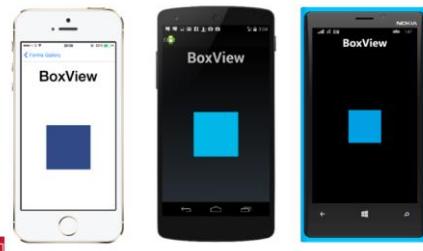


Xamarin 2

Xamarin 2

## BoxView osztály

- Színes téglalap



Xamarin 2

Xamarin 2

## Adat sablonok

- ListView-n használjuk
  - > `ItemTemplate`, `Header`/`Footer`/`GroupHeader`
- A gyökér elem mindenkorban egy `ViewCell`

```
<ListView x:Name="listView">
 <ListView.ItemTemplate>
 <DataTemplate>
 <ViewCell>
 <Grid>
 <Label Text="{Binding Name}" FontAttributes="Bold" />
 <Label Grid.Column="1" Text="{Binding Age}" />
 <Label Grid.Column="2" Text="{Binding Location}" />
 </Grid>
 </ViewCell>
 </DataTemplate>
 </ListView.ItemTemplate>
</ListView>
```

## Animáció

- Elsőször kódóból
  - > Később a kód felhasználható XAML-ben
- ViewExtensions osztály bővíti metódusai
  - > [Rel] TranslateTo: eltolás
  - > [Rel] RotateTo: forgatás + Anchor
  - > [Rel] ScaleTo: skálázás + Anchor
  - > FadeTo: elhalványulás / megjelenés - Opacity
  - > RotateX/YTo
  - > LayoutTo: elrendezés átalakításhoz
  - > CancelAnimations
- Mind aszinkront metódus – több is indítható

Xamarin 2

Xamarin 2

# Xamarin Forms



# Xamarin natív UI megközelítés



iOS C# UI

Android C# UI

Windows C# UI

Azure

Linux/Mono  
CoreCLR

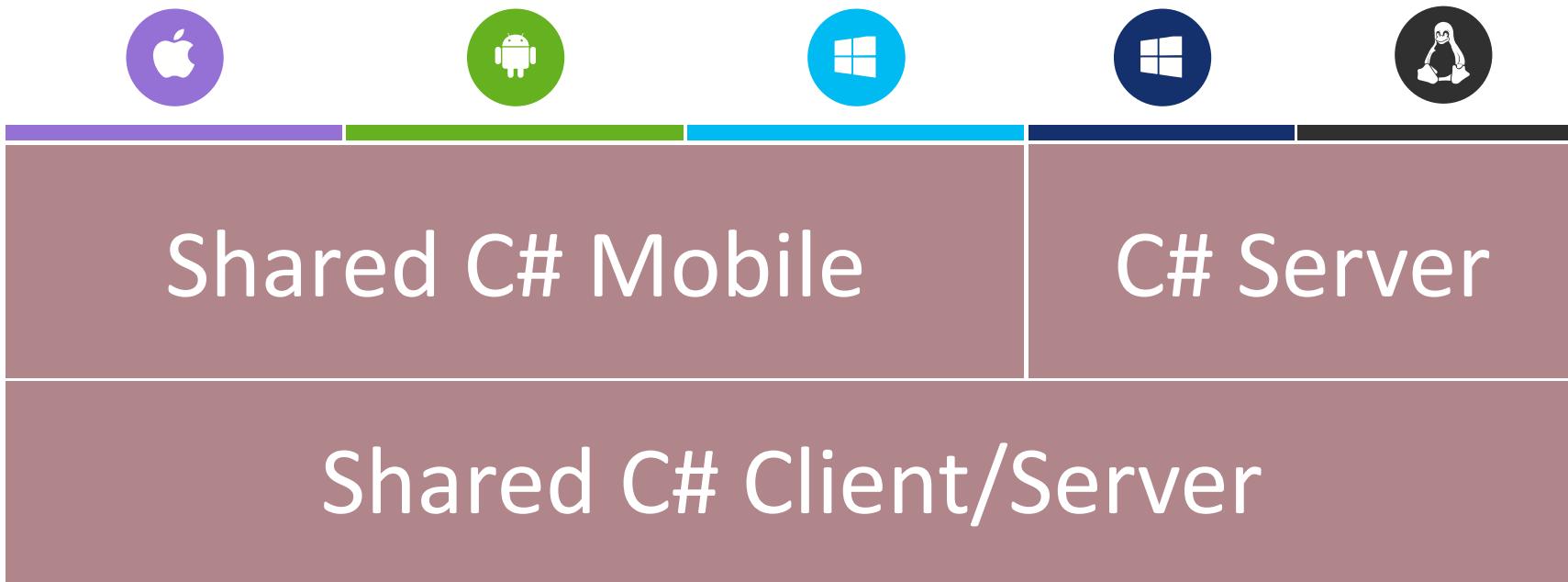
Shared C# Mobile

C# Server

Shared C# Client/Server

Shared C# codebase • 100% native API access • High performance

# Xamarin.Forms-szal még több közös kód



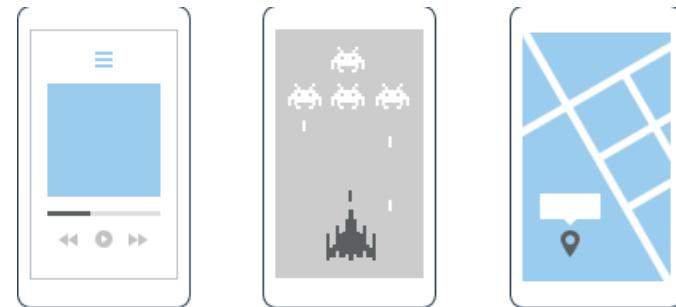
Shared C# codebase • 100% native API access • High performance

# Xamarin.Forms

- Nyílt forráskód
- Elsősorban platform független UI megvalósítására
- UI definiálása XAML-el vagy C#-ban
- Kiterjeszthető, az egyedi igények szerint
- Mikor válasszuk?



Xamarin.Form



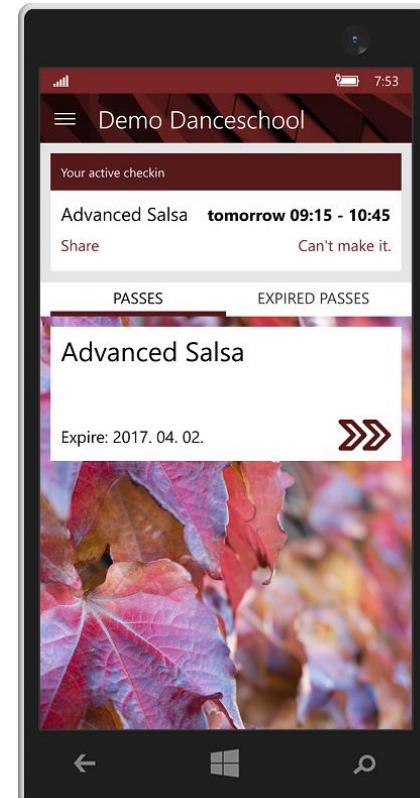
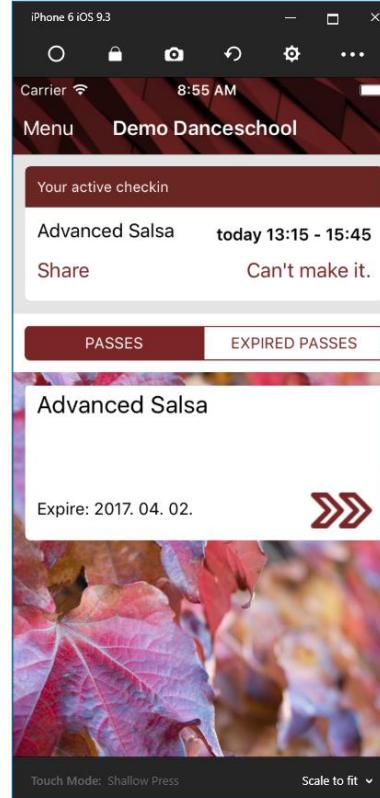
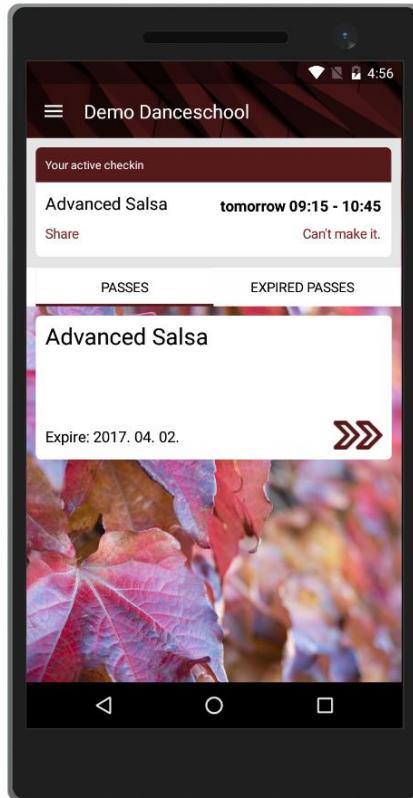
Xamarin.iOS, Xamarin.Android

S

# Valós példa

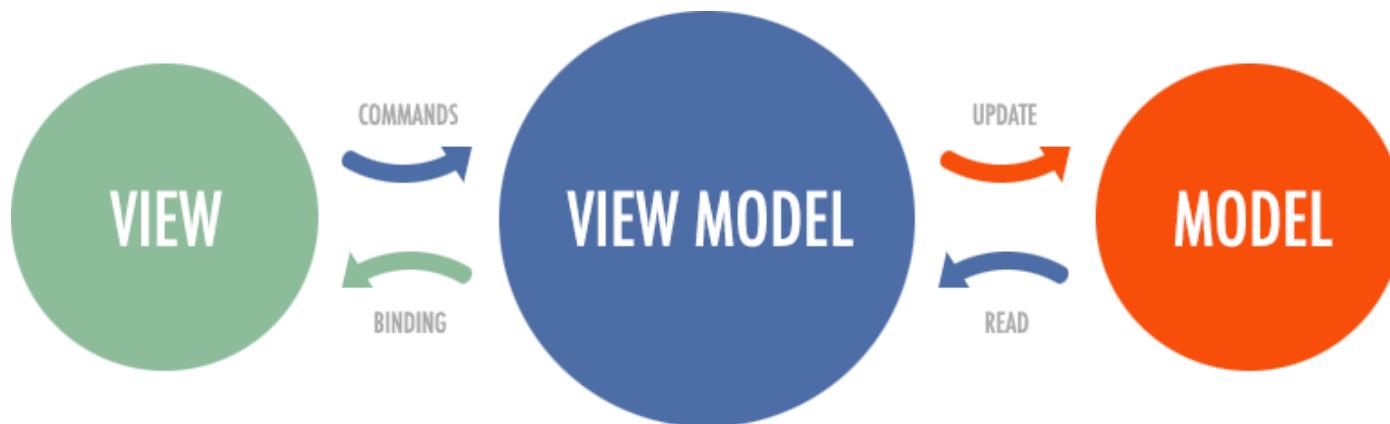
- CheckIn Swift

CheckinSwift	14586	86.36%
CheckinSwift.Droid	713	
CheckinSwift.iOS	927	13.64%
CheckinSwift.UWP	663	
16889		



# Alkalmazás architektúra

- Platform specifikus és platform független komponensek
- MVVM architektúra
  - > Valamennyi réteg platform független
  - > De kiterjeszthető platform specifikus kóddal is

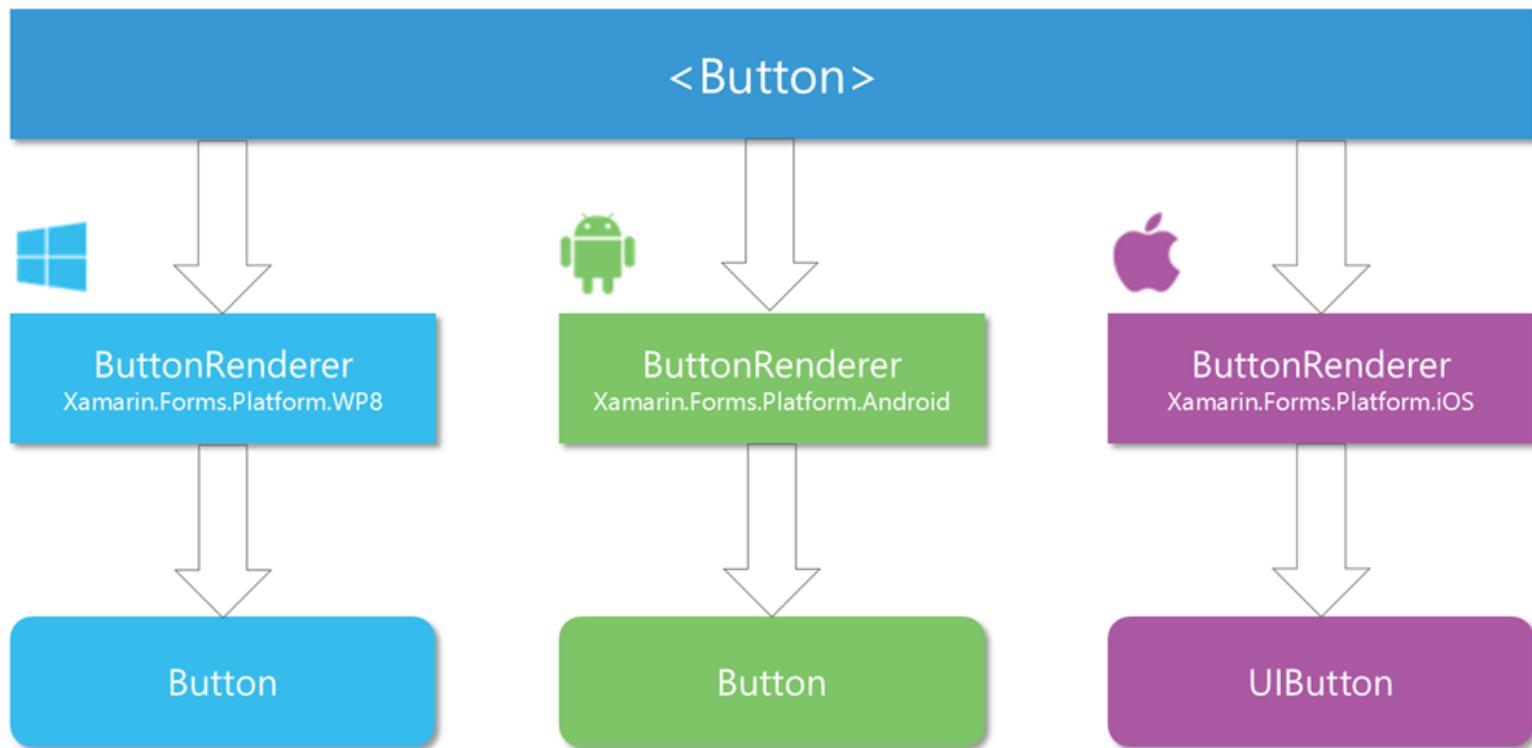


# Teljesítmény

- Mennyire hatékony a natív nézetek ilyen módon történő kezelése?
- Teljesítmény növelés módjai
  - > XAML fordítás
  - > Megfelelő, egyszerűbb layout választása
  - > Kevesebb adatkötés
  - > ListView-k megfelelő kezelése, virtualizációja
  - > Kevesebb globális Resource
  - > Renderer pattern betartása

# ViewRendererek

- XAML elemekből natív nézet
- Ez már platform specifikus logika



# ViewRenderer példa

```
[assembly: ExportRenderer(typeof(Label), typeof(Common.Droid.Controls.LabelRenderer))]
namespace Common.Droid.Controls
{
 1 reference | Balázs Ádám, 7 days ago | 1 author, 1 change
 public class LabelRenderer : ViewRenderer<Label, TextView>
 {
 19 references | Balázs Ádám, 7 days ago | 1 author, 1 change
 protected override void OnElementChanged(ElementChangedEventArgs<Label> e)
 {
 base.OnElementChanged(e);

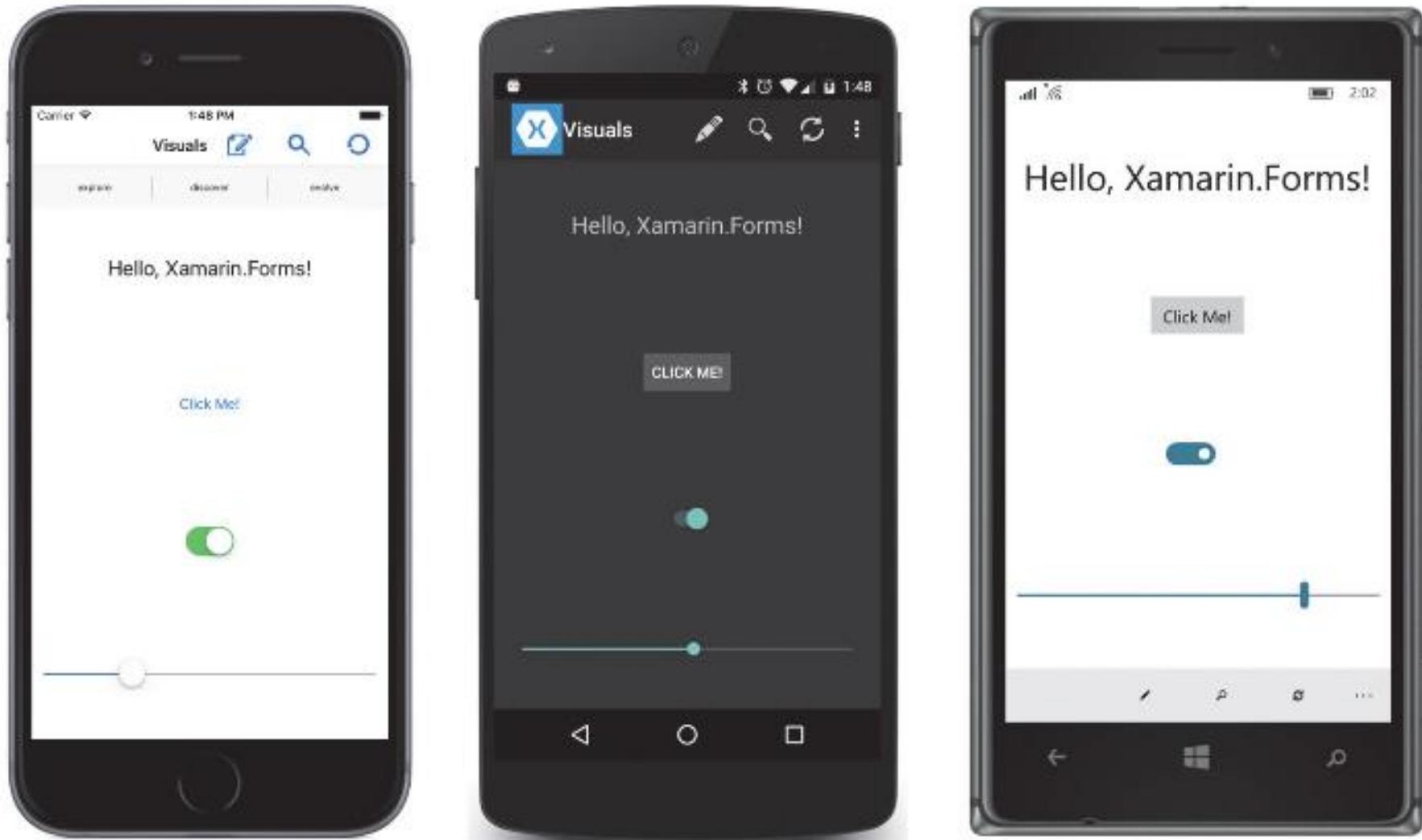
 if(e.OldElement!=null)
 {
 //Takarítás
 }

 if (e.NewElement != null)
 {
 SetNativeControl(new TextView(Context));
 }
 }

 19 references | Balázs Ádám, 7 days ago | 1 author, 1 change
 protected override void OnElementPropertyChanged(object sender, PropertyChangedEventArgs e)
 {
 base.OnElementPropertyChanged(sender, e);

 if (e.PropertyName == Label.TextProperty.PropertyName)
 Control.Text = Element.Text;
 }
 }
}
```

# Különböző vezérlők platformonként



# XAML jellemzők, egyediségek

- Kicsit más

```
<StackLayout HorizontalOptions="Center" IsVisible="{Binding IsPanelVisible}">
 <Label Text="Hello from Xamarin.Forms" />
</StackLayout>
```

- Data Binding

- > Dependency Property helyett Bindable Property
  - > Attached property-k

- Resource-ok

- > Style-ok, Control Template-ek, Data Template-ek, színek vagy konverterek
  - > StaticResource vagy DynamicResource

# XAML jellemzők, egyediségek

- Könnyen kiterjeszthető saját Markup Extension-el
  - > Például a Translate egy bővítés

```
<Label Text="{controls:Translate Text=Email_LoginLabel}"
 LineBreakMode="WordWrap"
 TextColor="{StaticResource DefaultAccentColor}"
 Style="{StaticResource SubtitleLabelStyle}"/>
```

- Akár natív nézetek is beágazhatóak a XAML kódba
  - > Adatkötés is támogatott

```
<StackLayout Orientation="Vertical" >
 <iOS:UILabel Text="Native Text" View.HorizontalOptions="Start"/>
 <Android:TextView Text="Native Text" x:Arguments="{x:Static formsAndroid:Forms.Context}" />
 <Windows:TextBlock Text="Native Text"/>
</StackLayout>
```

# Platformonként eltérő értékek

- **OnPlatform** xml teg
- **TypeArgument:** az érték típusa

```
<Grid.Margin>
 <OnPlatform x:TypeArguments="Thickness" iOS="10" Android="15" WinPhone="15" />
</Grid.Margin>
```

```
<ToolbarItem Text="edit" Order="Primary">
 <ToolbarItem.Icon>
 <OnPlatform x:TypeArguments="FileImageSource"
 iOS="edit.png"
 Android="ic_action_edit.png"
 WinPhone="Images/edit.png" />
 </ToolbarItem.Icon>
</ToolbarItem>
```

# INotifyPropertyChanged interfész

- A XAML-ben szokásos INPC interfész
- Itt is MVVM-et használunk
- Itt is vannak keretrendszerök!

# BindableObject osztály

- Hasonló a DependencyObject-hez
  - > DependencyProperty helyett BindableProperty
- Megvalósítja az INPC-t!
  - > Szemben a DependencyObjecttel

```
class MockBindableObject : BindableObject
{
 public static readonly BindableProperty NameProperty =
 BindableProperty.Create("Name", typeof(string),
 typeof(MockBindableObject),
 null);

 0 references
 public string Name
 {
 get { return (string)GetValue(NameProperty); }
 set { SetValue(NameProperty, value); }
 }
}
```

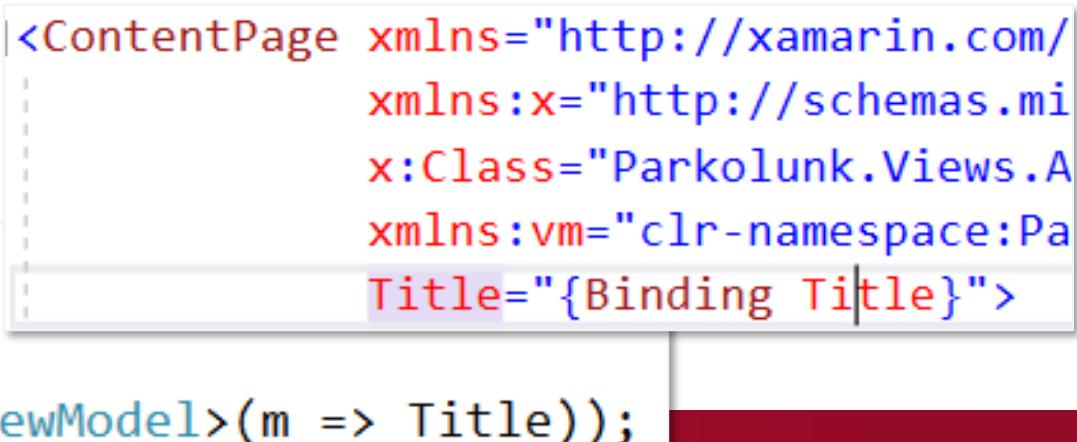
# Adatkötés Xamarinban 1.

- Adatforrás: BindableObject . **BindingContext**
  - > Hasonlóan a DataContext-hez, a gyerekek öröklik a szülőtől
- Binding osztály
  - > BindingMode
    - Default: a vezérlő propertyjén megadott alapértelmezett
    - TwoWay: frissíti az adatforrást
    - OneWay: csak a vezérlőt frissíti
    - OneWayToSource: csak az adatforrást frissíti
  - > StringFormat - formázás

# Adatkötés Xamarinban 2.

- Source
  - > A forrás adat osztály
- Path
  - > A property a forrás objektumon, amihez kötünk
- Converter, ConverterParameter
  - > IValueConverter interfész

```
this.SetBinding(
 Page1.TitleProperty,
 Binding.Create<BaseViewModel>(m => Title));
```



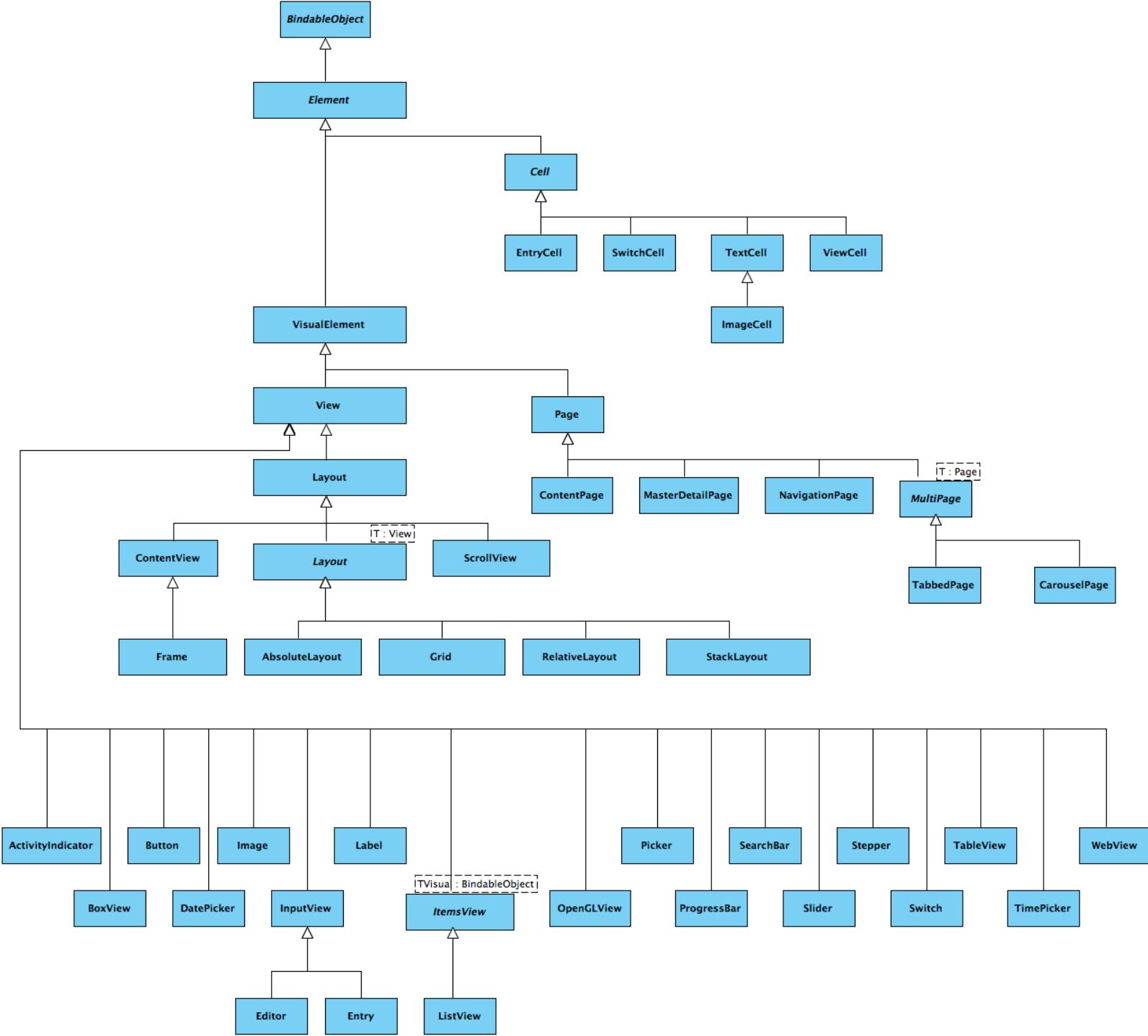
# Csatolt tulajdonságok

- Hasonlóan más XAML technológiákhoz itt is van
- Tetszőleges objektumra rátehető egy másik osztály által kezelt tulajdonság

```
<AbsoluteLayout VerticalOptions="FillAndExpand"
 HorizontalOptions="FillAndExpand">
 <BoxView AbsoluteLayout.LayoutBounds="0.25, 0.25, 0.5, 0.5"
 Color="Blue"
 AbsoluteLayout.LayoutFlags="All" />
</AbsoluteLayout>
```

# Element osztály : BindableObject

- Az objektum hierarchiában résztvevő elemek alaposztálya
  - > Parent: közvetlen szülő
  - > ParentView (readonly): látható szülő
  - > ChildAdded/Removed események
    - Közvetlen gyerek
  - > DescendantAdded/Removed események
    - Az új gyerek elem gyerekeire is elsül



# VisualElement osztály : Element

- A képernyőn megjelenő vezérlő elem
  - > Kiterjedés: Bounds, Height, Width
  - > X, Y: lekérdezhető aktuális pozíció a szülőhöz képest
- Transformáció
  - > RotationX,Y,Z: perspektívikus is
  - > Scale: aránytartó
  - > TranslationX/Y: eltolás
- Opacity, BackgroundColor (egyszerű szín)
- Style, StyleClass
- IsEnabled, IsVisible, IsFocused
- Navigation

# ELÁGAZÁS

- VisualElement alaposztály
- Page irány
  - > Teljes képernyőt elfoglaló szülő elem
  - > Egyelten, View típusú tartalommal
- View osztály
  - > Vezérlők
  - > Elrendező (layout) elemek

# Page osztály : VisualElement

- A teljes képernyőt elfoglaló elem
  - > Title: szöveges cím
  - > ToolbarItems: „menü”
  - > Padding: térköz
  - > Icon, BackgroundImage
- IsBusy: platform specifikus várakozó animáció
- DisplayAlert: modális dialógus ablak
- ForceLayout() / LayoutChanged
- Appearing / Disappearing

# TemplatedPage osztály : Page

- Vezérlő sablonnal rendelkező lap
- ControlTemplate: a megjelenítéshez használt sablon

# ContentPage osztály : TemplatedPage

- Egyetlen tartalmat megjelenítő lap
- A XAML dokumentum gyökér eleme
- Content tulajdonság: hivatkozás a tartalomra
- Típusa: View

# View osztály : VisualElement

- Klasszikus vezérlők és elrendező elemek őse
- Horizontal/VerticalOptions: elrendezés
  - > LayoutOptions osztály
    - Statikus tagok a gyakran használt opciókhöz
  - > Alignment tag
    - Start, End, Center, Fill: bal/fenn, jobb/lenn, közép, kitölt
  - > Expands tag
    - Kitölți-e a vezérlő a rendelkezésre álló helyet
- Margin

# Layout osztály : View

- Alaposztály, a rajta lévő vezérlőket helyezi el
- Padding: a gyerek elemek távolsága a kerettől
- IsClippedToBounds: vágás
- LowerChild / RaiseChild: vizuális Z index állítás
- ForceLayout: elrendezés újraszámolása
- GetSizeRequest: mekkora helyre van szüksége

# ScrollView osztály : Layout

- A benne lévő tartalom görgethető
- Content tulajdonság: tetszőleges View leszármazott
- Orientation: görgetés iránya
- ScrollX/Y, Scrolled esemény: pozíció lekérdezése
- ScrollToAsync: görgetés adott helyre
- ScrollView vezérlőket nem szabad egymásba ágyazni!

# ContentView osztály : Layout

- Olyan Layout vezérlő, ami egyetlen View elemet csomagol a Content tulajdonságban
- Kényelmes: beállítható rajta háttér, margó

# Frame osztály: ContentView

- Keretet rajzol a tartalom köré
- OutlineColor: szín
- HasShadow: árnyék
- CornerRadius: lekerekítés

# Layout<T> osztály : Layout

- IList<T> Children: T típusú gyerek vezérlők
  - > T a View-ból származik
  - > A Layout vezérlők egymásba ágyazhatók

# StackLayout

- Elemek elrendezés horizontálisan vagy vertikálisan
- Orientation
- Spacing: térköz az elemek között
- Használja az Expand bitet: ha van hely, a megjelölt vezérlő foglalja el a maradék helyet

# Grid

- Sorok, oszlopok definiálása megszokott módon
  - > Abszolút méretek eszköz koordinátákban
  - > Automatikus méret számolás
  - > Arányos tér elosztás
- Cella összevonás
- Sor / oszlop térközök

# AbsoluteLayout

- Pozicionálás eszköz vagy relatív [0..1] koordinátákkal
- LayoutBounds: koordináták
- LayoutFlags: melyik koordináta milyen típusú
- Ha a szélesség / magasság relatív módon van megadva, akkor a vezérlő biztosan ki fog férni
  - > Például: „1.0, 1.0, 0.5, 0.5”
  - > a vezérlő a jobb alsó sarokban lesz, nem lóg ki

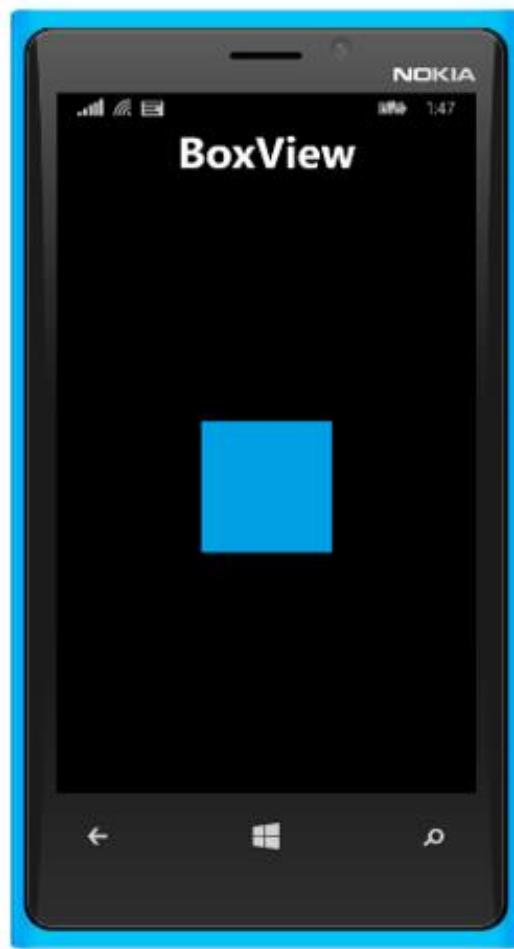
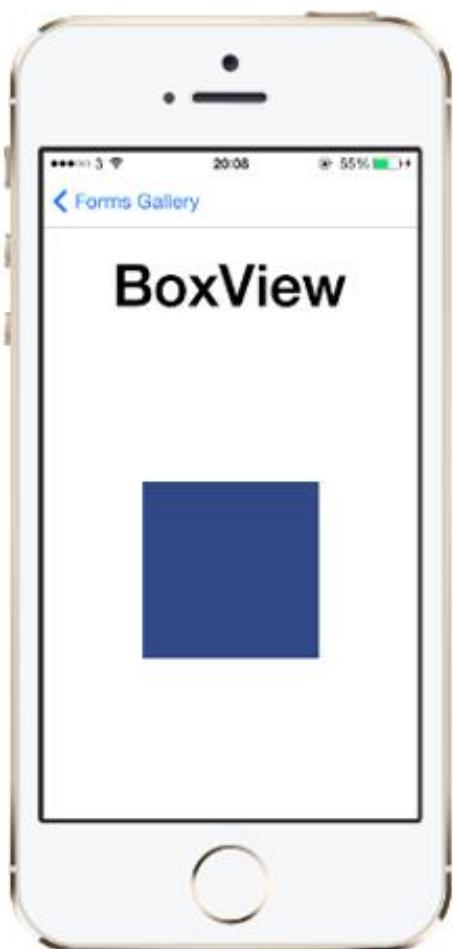
# RelativeLayout

- A vezérlő pozíciója és méretei a szülőhöz vagy más elemhez képest változnak
- Tetszőleges elem tetszőleges tulajdonsága felhasználható
- Alapműveletek

```
relativeLayout.Children.Add (relativelyPositioned,
 Constraint.RelativeToParent ((parent) => {
 return parent.Width / 3;
 }),
 Constraint.RelativeToParent ((parent) => {
 return parent.Height / 2;
 }));
```

# BoxView osztály

- Színes téglalap



# Button

- Egyszerű gomb
- Text: megjelenítendő szöveg
  - > Betűkészlet, -méret, -szín + félkövér, italic
- Image: megjelenítendő kép
- Testreszabható keret
- Clicked esemény, ICommand interfész



# Label

- Egy vagy több sornyi szöveg megjelenítése
- Szöveg tulajdonsága
  - > Betűkészlet, -méret, -szín + félkövér, italic
  - > Formázás akár karakterenként
- Szöveg tördelés
  - > Karakterenként vagy szavanként, "... helye
- Szöveg elrendezése horizontálisan, vertikálisan
  - > Nincs sorkizárt

# Image

- Kép megjelenítése
- IsLoading tulajdonság
- Kitöltés: Aspect tulajdonság
  - > AspectFill: aránytartó, levágja a képet
  - > AspectFit: aránytartó, nem vágja le a képet
  - > Fill: nem aránytartó, kitölți a teret
- ImageSource : Element
  - > Fájlból
  - > Erőforrás azonosító alapján
  - > Folyamból
  - > URI alapján

# Entry osztály: InputView

- Egy sornyi szöveg bekérése
- Szöveg tulajdonsága
  - > Betűkészlet, -méret, -szín + félkövér, italic, alignment
- IsPassword
- Placeholder + Color
- TextChanged esemény
- Completed esemény: a felhasználó végzett (Enter billentyű)

# Editor osztály: InputView

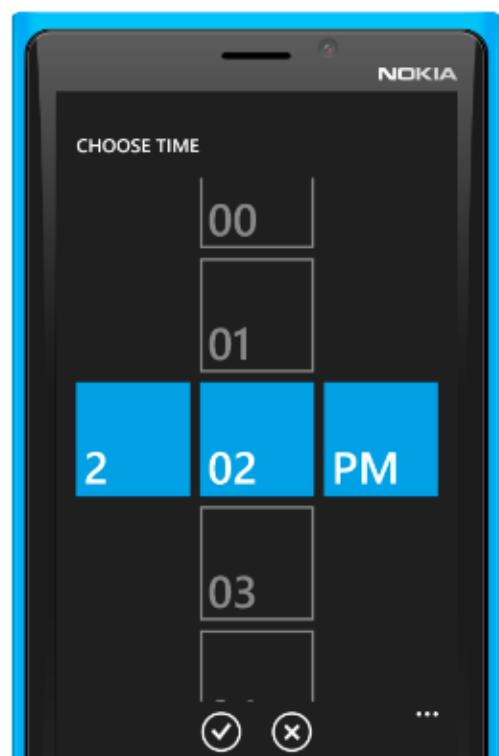
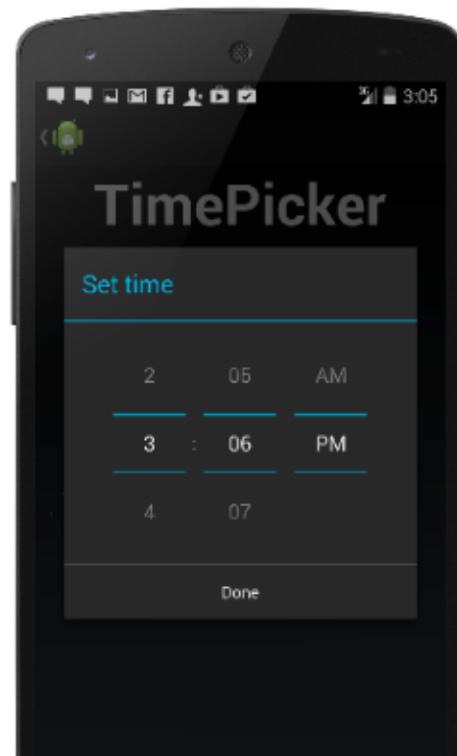
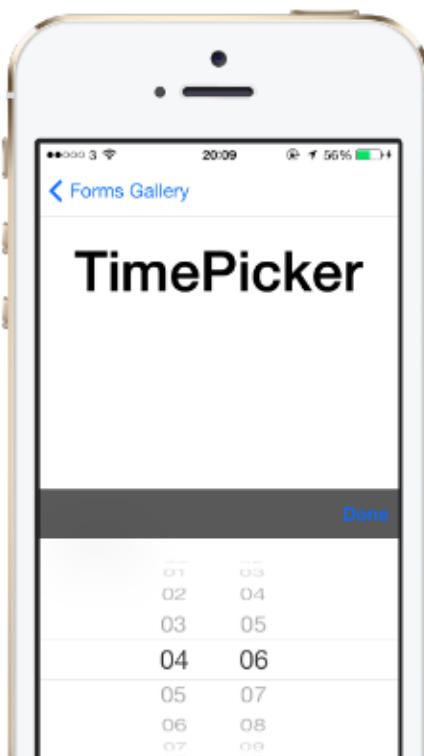
- Több sornyi szöveg bekérése
- Szöveg tulajdonsága
  - > Betűkészlet, -méret, -szín + félkövér, italic
- TextChanged esemény
- Completed esemény: a felhasználó végzett (Enter billentyű)

# Keyboard osztály

- Mindkét szöveg beviteli vezérlőhöz állítható
- Egyéni bevitel tipikus feladatokhoz
  - > Számok, email, url, chat, telefon, ...

# TimePicker

- Idő bekérése
- Time (TimeSpan)
- Szöveg szín és formázás



# DatePicker

- Dátum bekérése
- Date (DateTime)
- Intervallum beállítása
- Szöveg szín és formázás

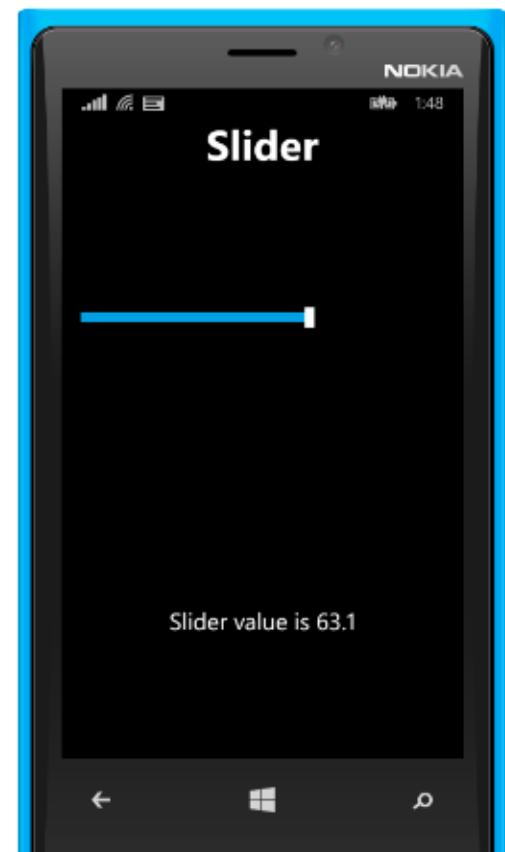
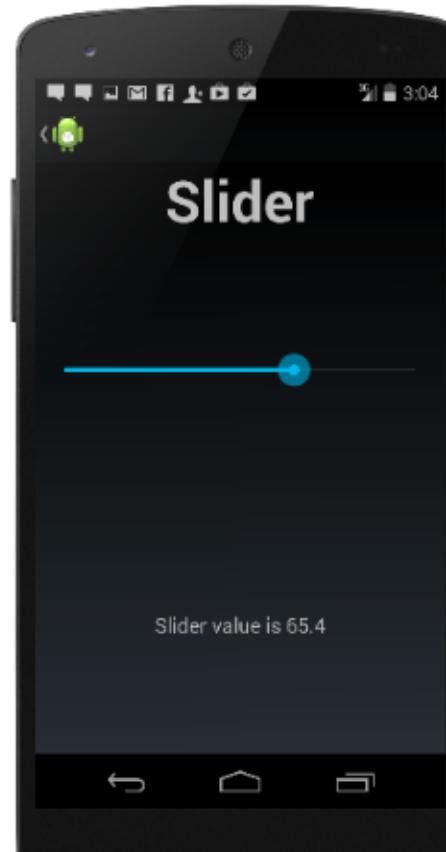


# WebView

- HTML tartalom megjelenítése
- Navigáció
  - > GoBack / GoForward / Navigating / Navigated
- Eval: tetszőleges szkript meghívható
- Source: szöveg vagy URI

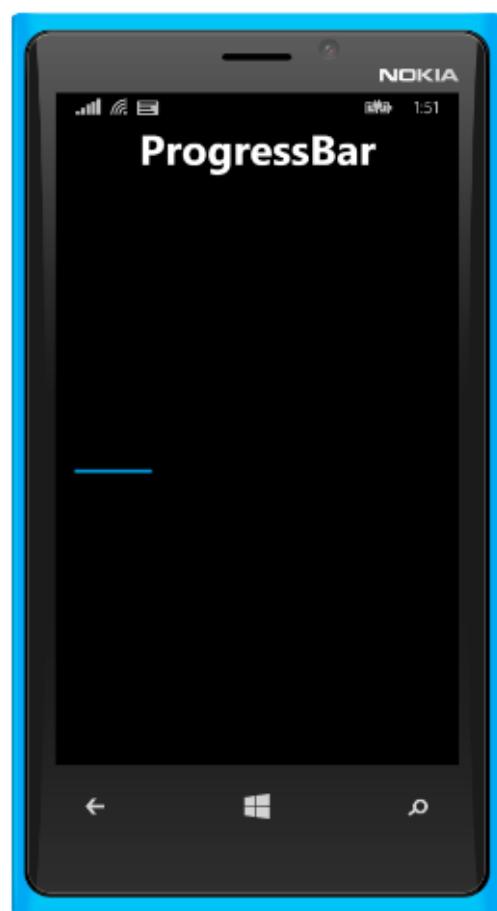
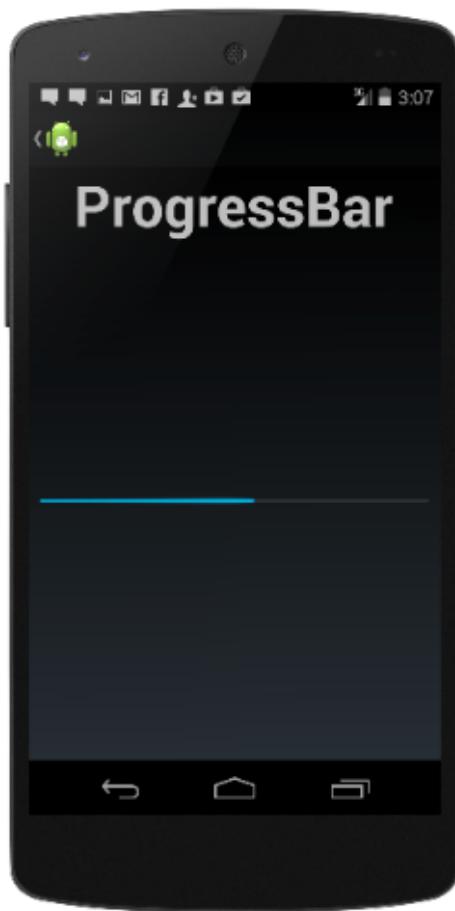
# Slider

- Egy érték kiválasztása
- Value / ValueChanged + intervallum



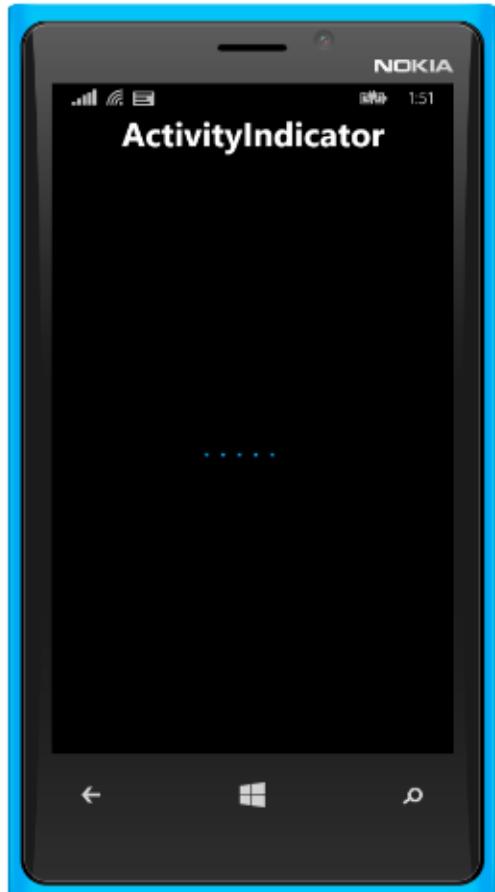
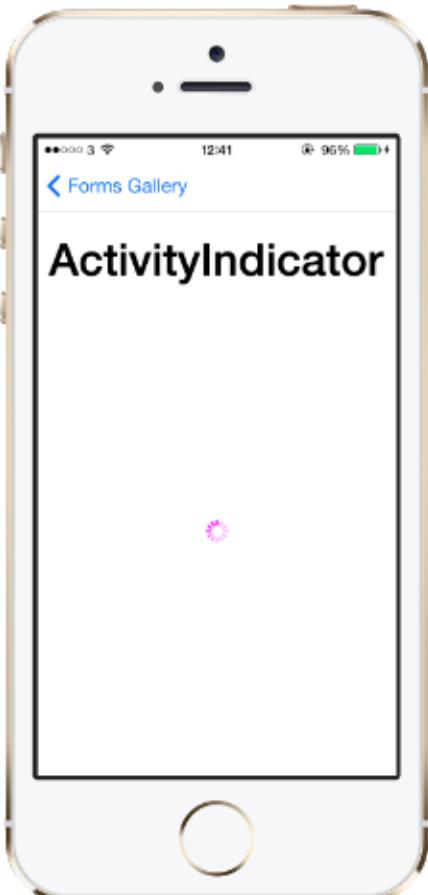
# ProgressBar

- Progress: [0..1]



# ActivityIndicator

- Color: állítható szín
- IsRunnging: bool



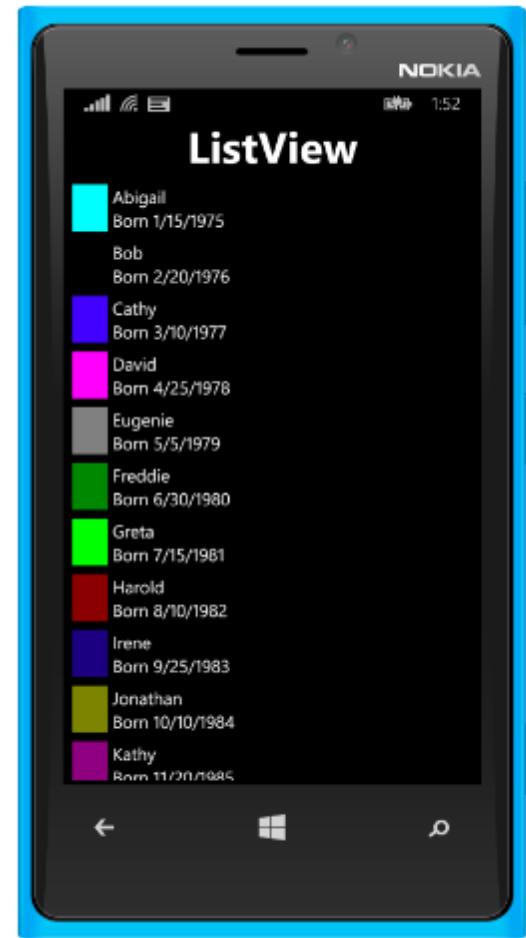
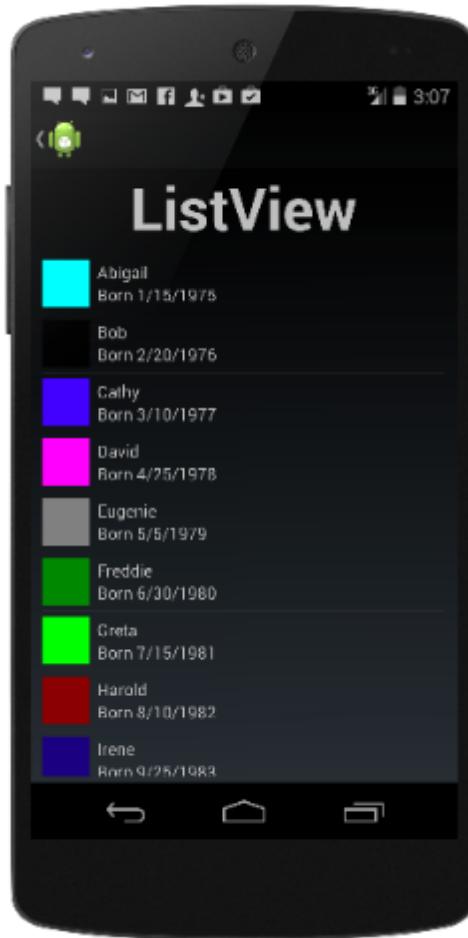
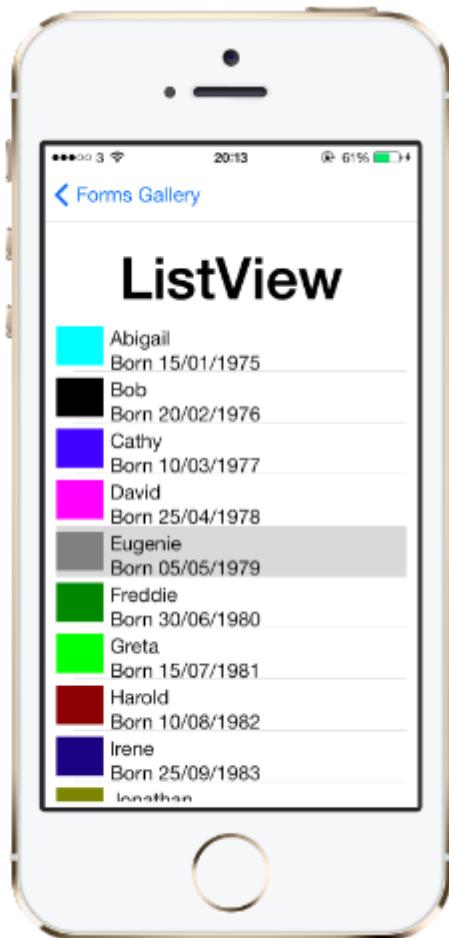
# ItemsView<TView>

- Listás vezérlő alaposztálya
- ItemsSource: adatforrás
- ItemTemplate: megjelenítéshez használt sablon

# ListView osztály: ItemsView<Cell>

- Vertikális lista megjelenítése
- Fejléc, lábléc (akár sablonnal)
- Csoportosítás, csoport fejléc
- Különböző magasságú sorok támogatása
- PullToRefresh támogatás
- Kiválasztott sor támogatás
- Elválasztó vonal, szín

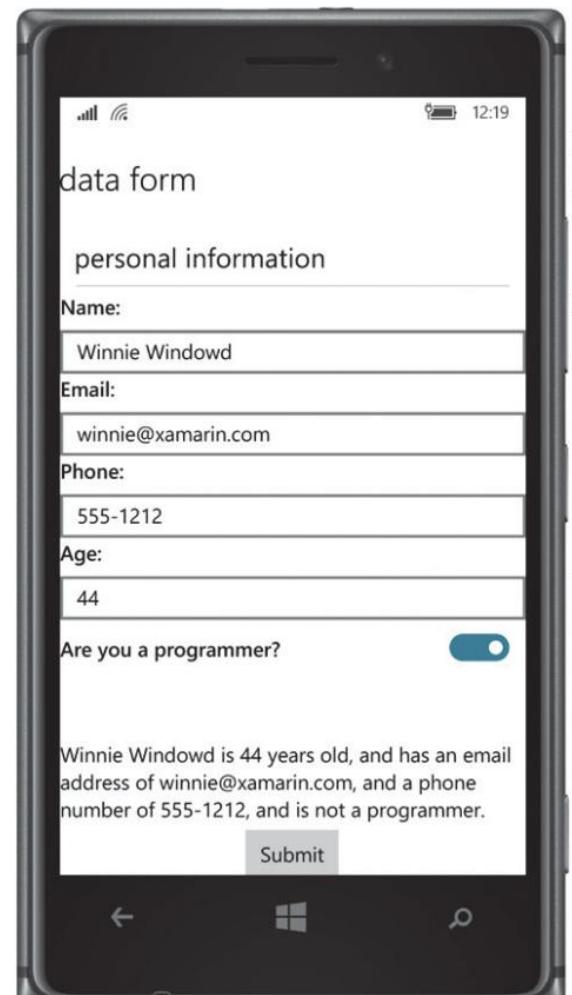
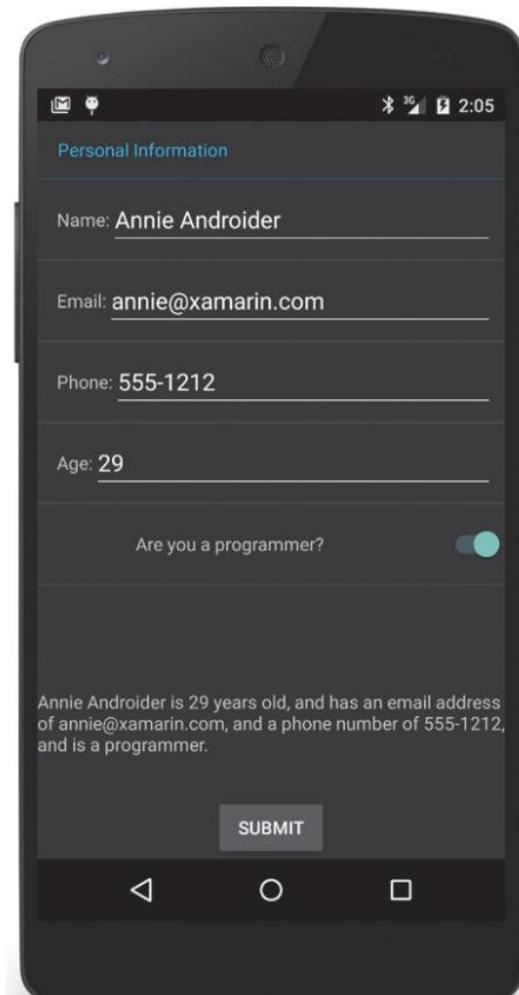
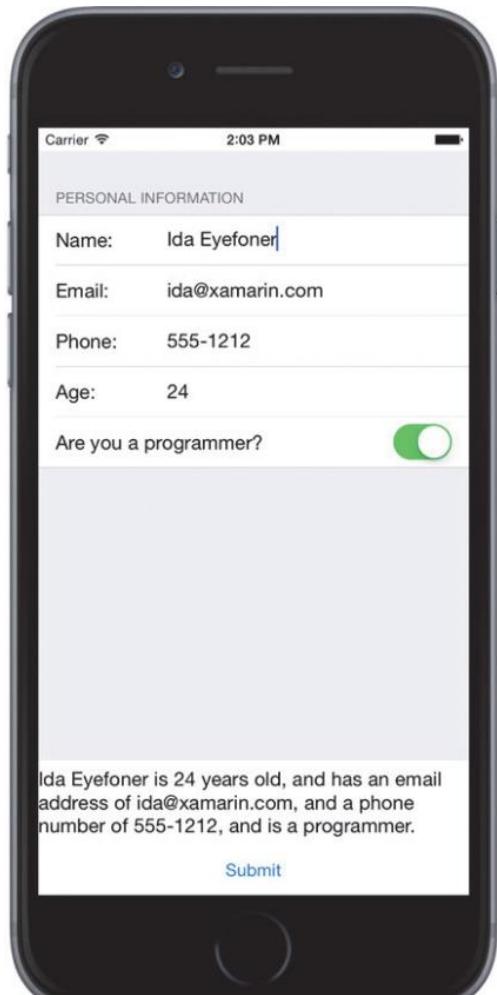
# ListView megjelenése



# TableView osztály: View

- Ez is elemek vertikális listája, nem táblázat!
  - > A ListView tipikusan azonos elemeket jelenít meg
  - > A TableView különbözőket jelenít meg
- Intent tulajdonság: mire használjuk
  - > Menu, Settings, Form, Data
- Két szintű hierarchia
  - > TableSection + Cell objektumok

# TableView - Form



# Cell osztály : Element

- Lista vagy tábla nézet belső elemei lesznek
  - > ListView, TableView
- Magassága állítható
- IsEnabled tulajdonság
- ContextActions: az elemhez tartozó menü
  - > MenuItem: szöveg, ikon + Command

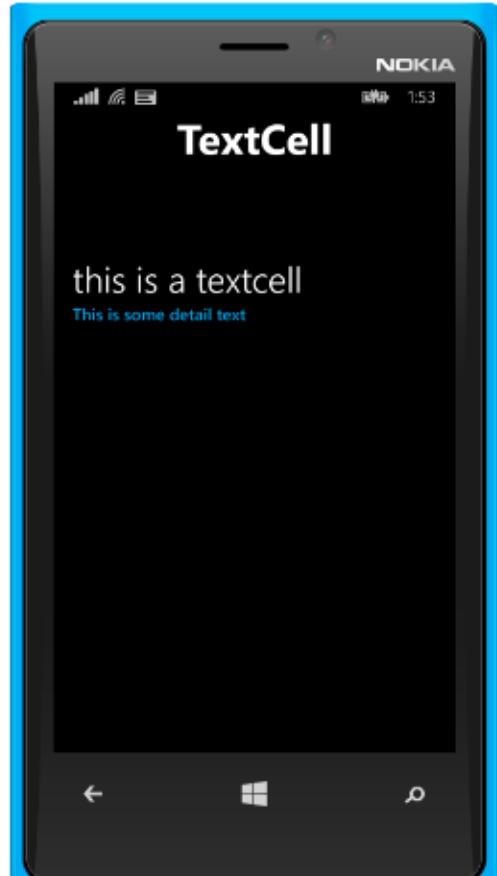
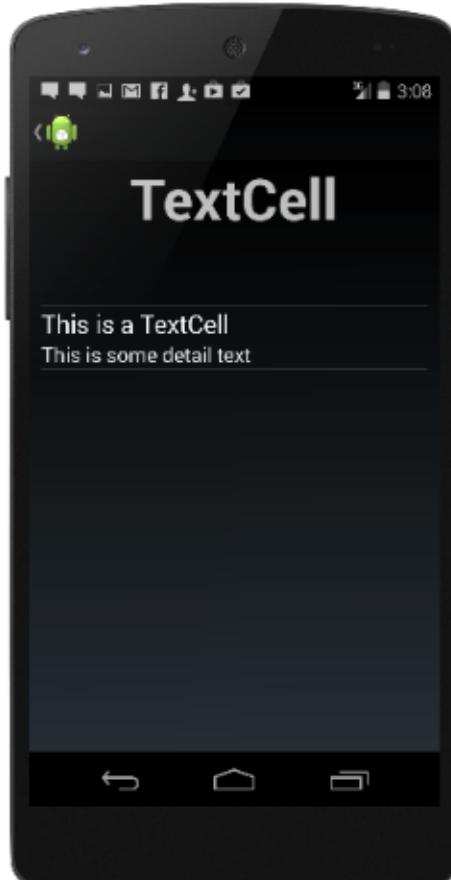
# EntryCell osztály : Cell

- Címke + szöveg beviteli mező
- Minimális testreszabás



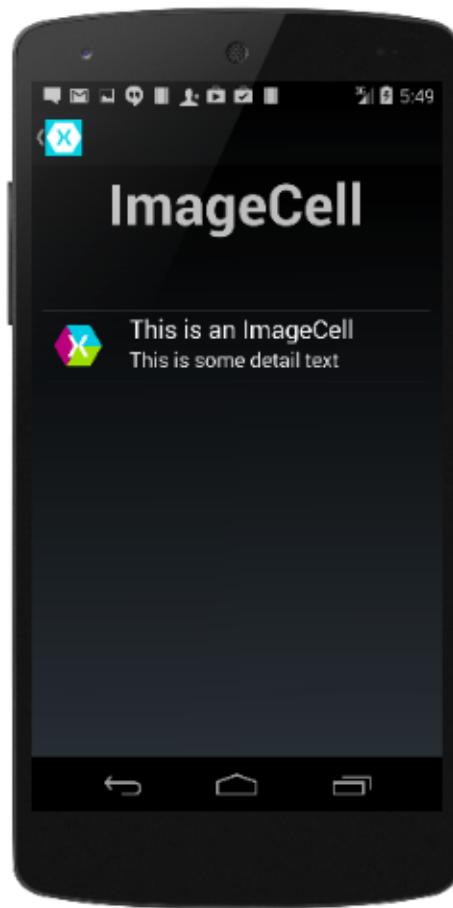
# TextCell osztály: Cell

- Címke, szöveg + Command
- Minimális testreszabás



# ImageCell osztály: TextCell

- A szöveg mellett kép



# SwitchCell osztály: Cell

- Címke + kapcsoló
- Minimális testreszabás

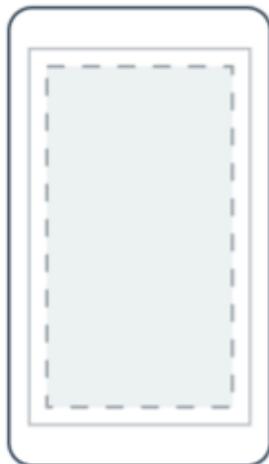


# ViewCell osztály: Cell

- Saját nézet
- View tulajdonság, tetszőleges tartalom

# Navigáció

- Platformonként különbözik a navigációs paradigma
- A Page osztályok rejtik el a különbségeket vizuális



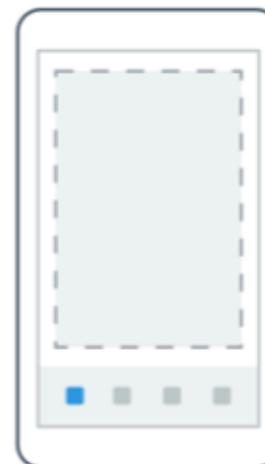
ContentPage



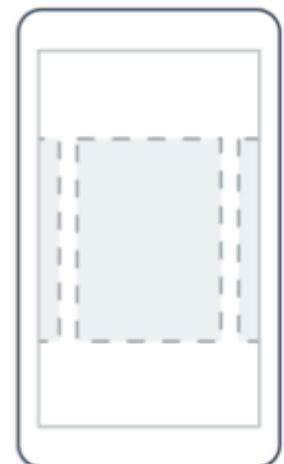
MasterDetailPage



NavigationPage



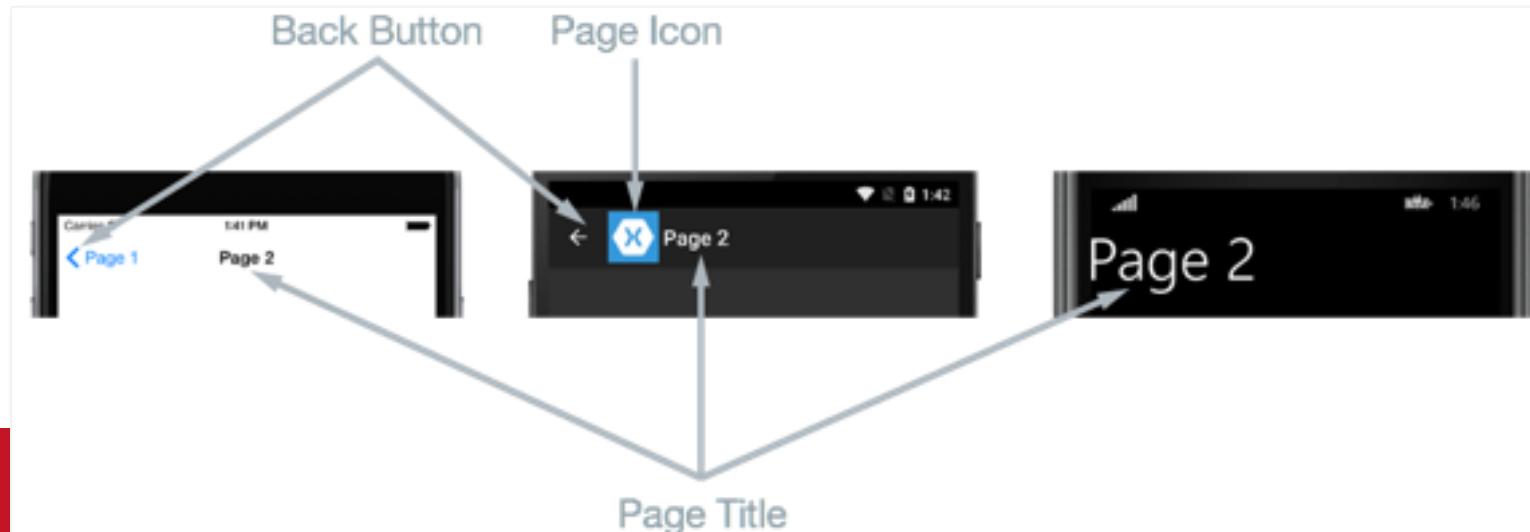
TabbedPage



CarouselPage

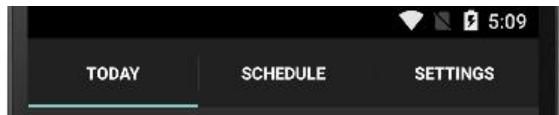
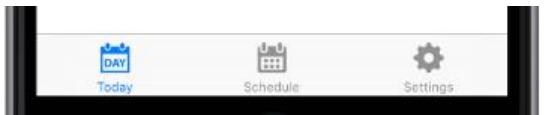
# Hierarchikus navigáció

- NavigationPage lesz a gyökér elem
- A fejléc minimálisan testreszabható
- CurrentPage: az oldal, ami megjelenik benne
- LIFO jellegű: Push / Pop / PopToRoot

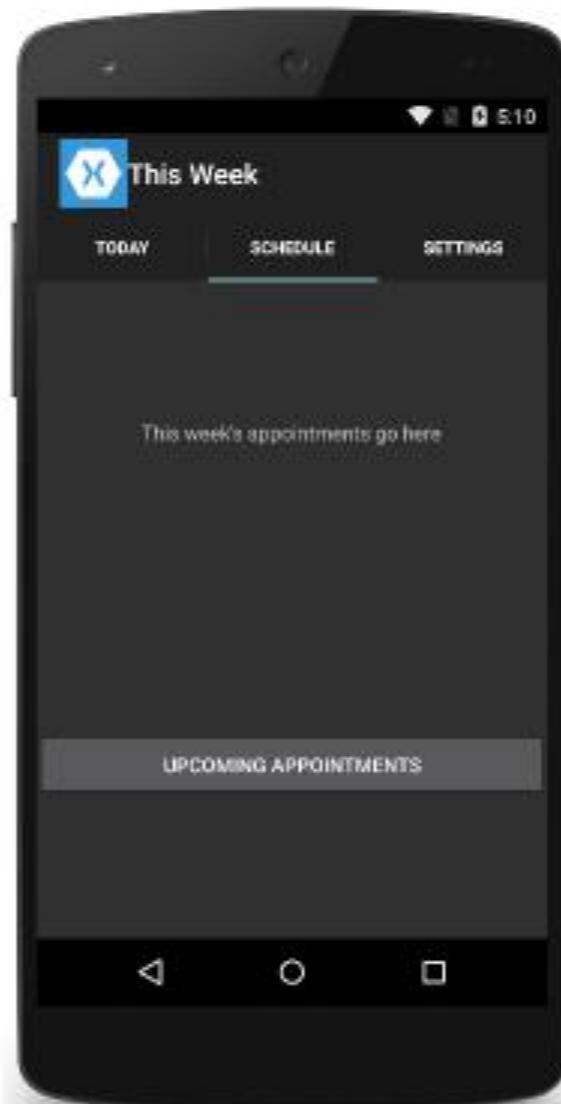
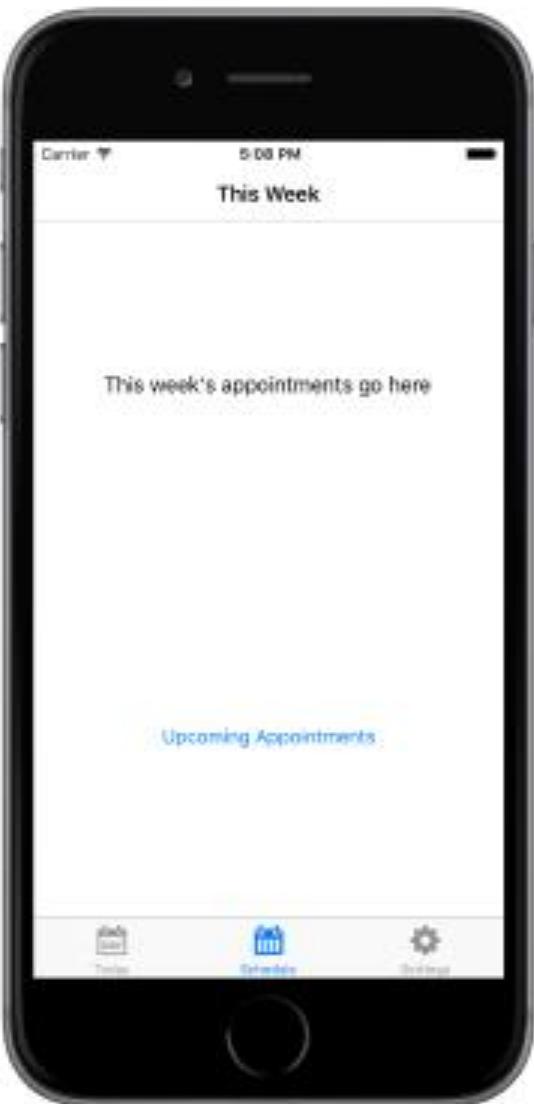


# Tabos navigáció

- TabbedPage vezérlő, máshogy jelenik meg az egyes platformokon
  - > A Page-nek van ikonja és címe
- Tipikusan NavigationPage-ek és ContentPage-ek vannak rajta
  - > Children gyűjteménybe
  - > ItemsSource + ItemTemplate
- A fejléc minimálisan testreszabható

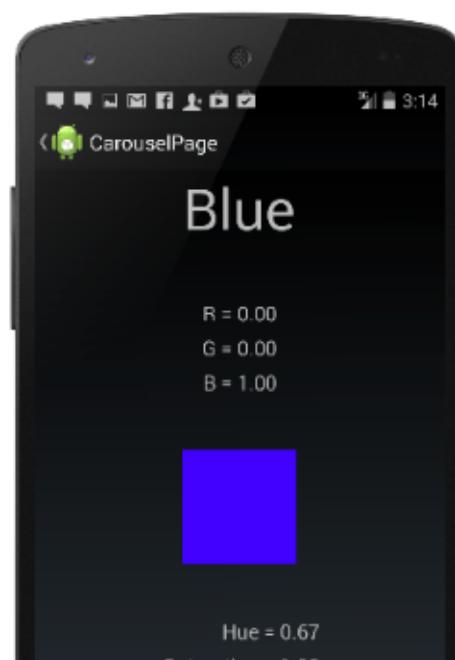


# Hierarchikus navigáció a tabokon belül



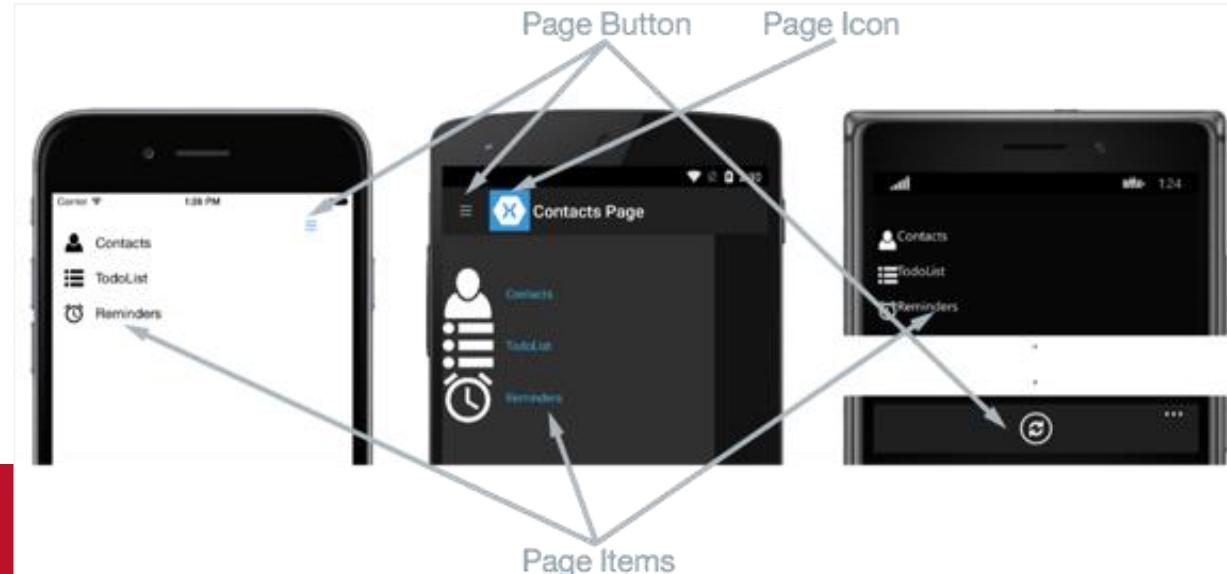
# Carousel navigáció

- Jobbra-balra lehet húzni az oldalakat
- Csak ContentPage használható benne
  - > Children gyűjteménybe
  - > ItemsSource + ItemTemplate



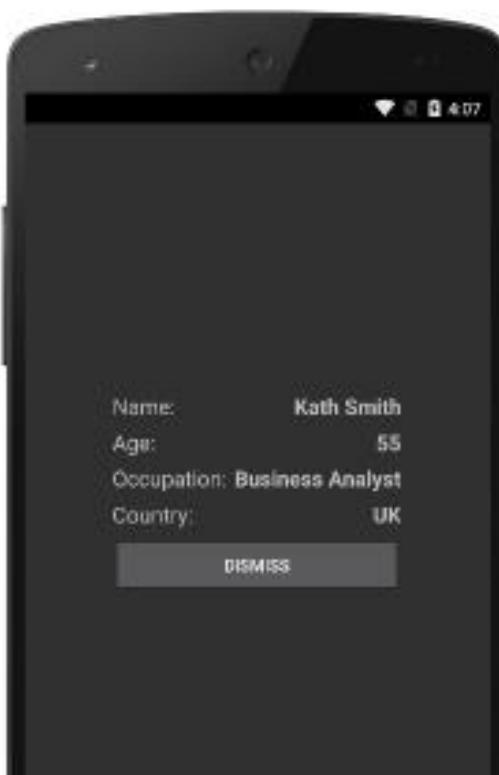
# Master-Detail Page

- Master: egy ContentPage
- Detail: Tabbed/Navigation/ContentPage
- IsPresented: váltás a két oldal között
- MasterBehavior - megjelenítés módja
  - > Default: a platform alapértelmezett megjelenítése
  - > Popover
  - > Split
  - > SplitOn...
    - Landscape
    - Portrait



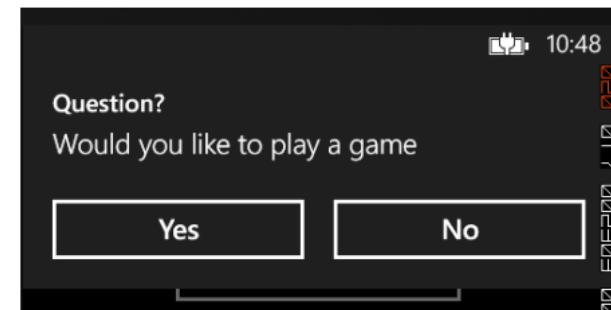
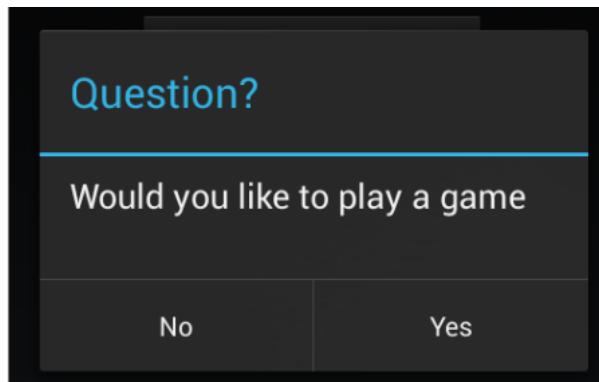
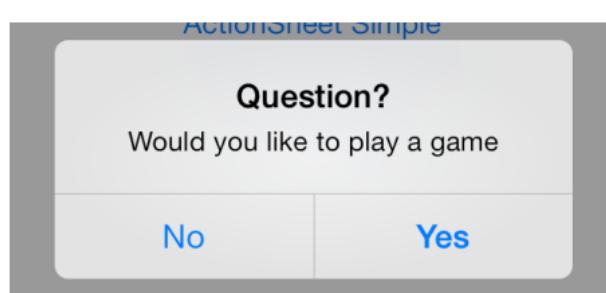
# Modális ablakok

- Bármelyik típusú oldalon használhatunk modális ablakot
- Navigation . PushModal ( Page )

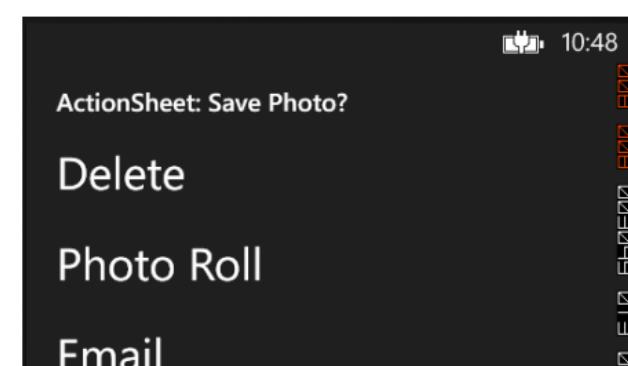
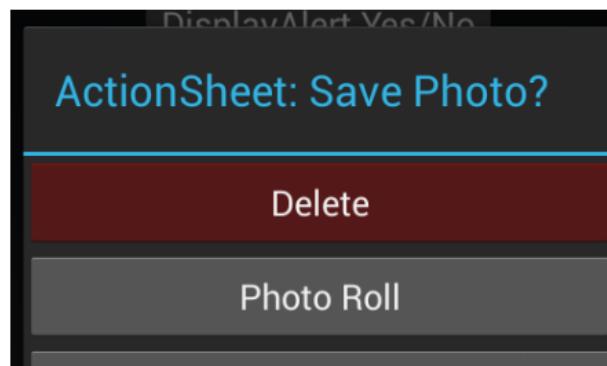
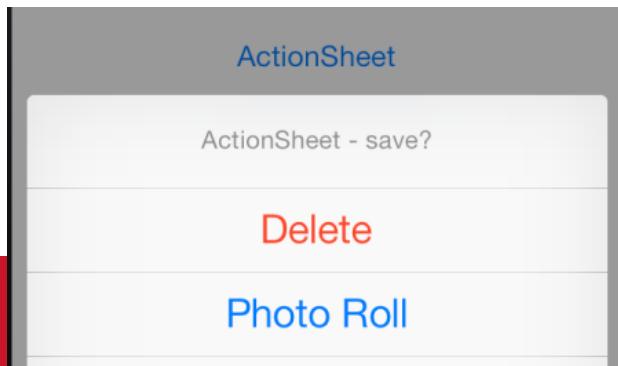


# Popupok

- **DisplayAlert hívás**
  - > Cím, szöveg + oké, mégsem feliratok



- **DisplayActionSheet hívás**
  - > cím, mégsem, destruction szöveg + gombok



# Adat sablonok

- ListView-n használjuk
  - > **ItemTemplate**, Header/Footer/GroupHeader
- A gyökér elem mindenkorban egy ViewCell

```
<ListView x:Name="listView">
 <ListView.ItemTemplate>
 <DataTemplate>
 <ViewCell>
 <Grid>
 <Label Text="{Binding Name}" FontAttributes="Bold" />
 <Label Grid.Column="1" Text="{Binding Age}" />
 <Label Grid.Column="2" Text="{Binding Location}" />
 </Grid>
 </ViewCell>
 </DataTemplate>
 </ListView.ItemTemplate>
</ListView>
```

# DataTemplateSelector : DataTemplate

- Select metódus, DataTemplate-et ad vissza
- Adat példányonként választ sablont
- Közvetlenül adatsablonként használható

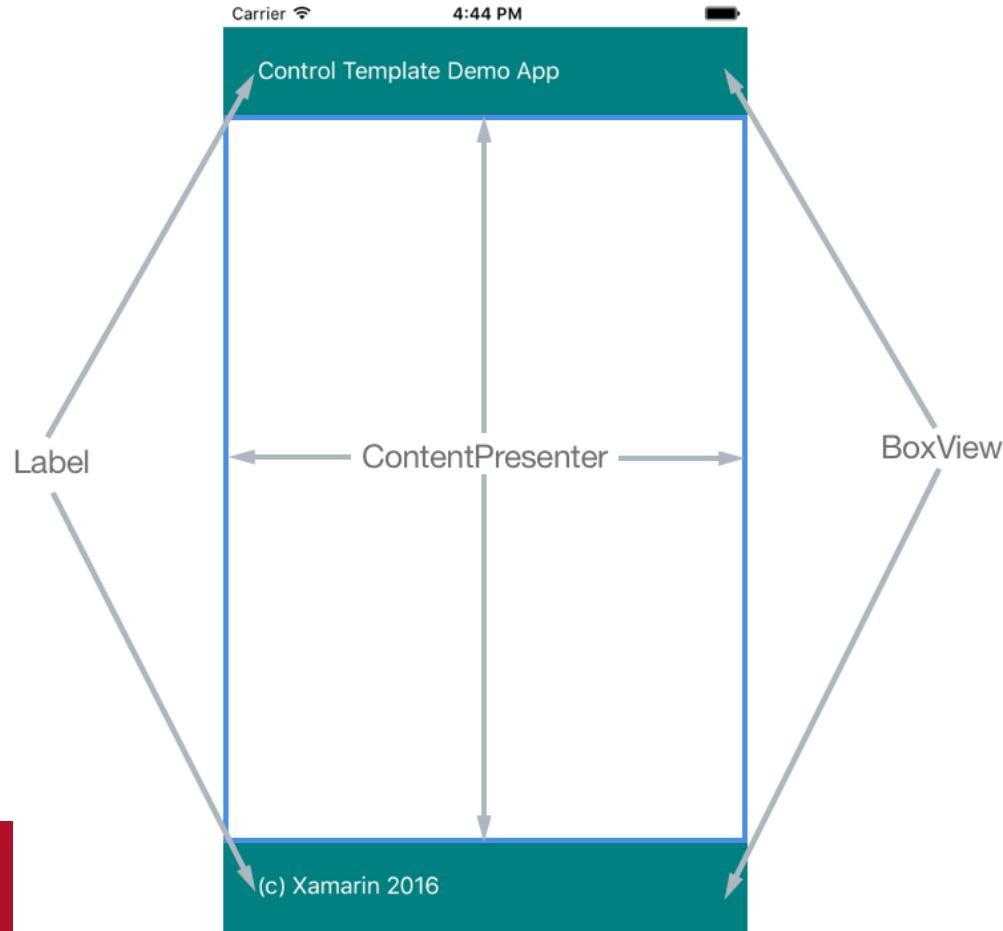
1 reference | 0 changes | 0 authors, 0 changes

```
class MyDataTemplateSelector : DataTemplateSelector
{
 0 references | 0 changes | 0 authors, 0 changes
 protected override DataTemplate OnSelectTemplate(object item, BindableObject container)
 {
 if (item is double)
 return this.templateOne;
 return this.templateTwo;
 }

 0 references | 0 changes | 0 authors, 0 changes
 public MyDataTemplateSelector()
 {
 // Retain instances
 this.templateOne = new DataTemplate(typeof(ViewA));
 this.templateTwo = new DataTemplate(typeof(ViewB));
 }
}
```

# ControlTemplate

- Nem általános vezérlő sablon
- ContentPage-ek testreszabása
  - > ContentPresenter
  - > Layout



# Kérdések?

Albert István  
[ialbert@aut.bme.hu](mailto:ialbert@aut.bme.hu)

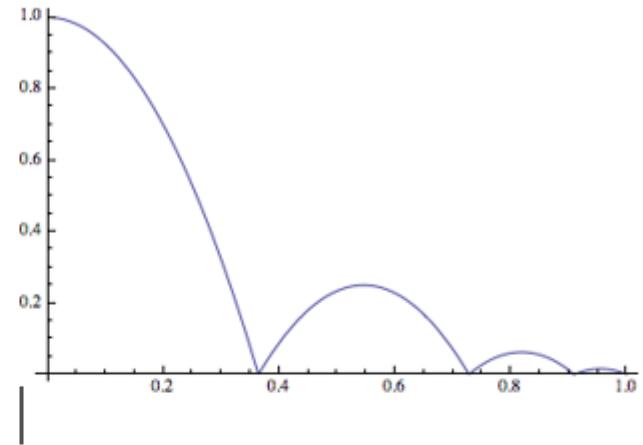
# Animáció

- Elsősorban kódból
  - > Később a kód felhasználható XAML-ben
- ViewExtensions osztály bővítő metódusai
  - > [Rel] TranslateTo: eltolás
  - > [Rel] RotateTo: forgatás + Anchor
  - > [Rel] ScaleTo: skálázás + Anchor
  - > FadeTo: elhalványulás / megjelenés – Opacity
  - > RotateX/YTo
  - > LayoutTo: elrendezés átalakításhoz
  - > CancelAnimations
- Mind aszinkront metódus – több is indítható

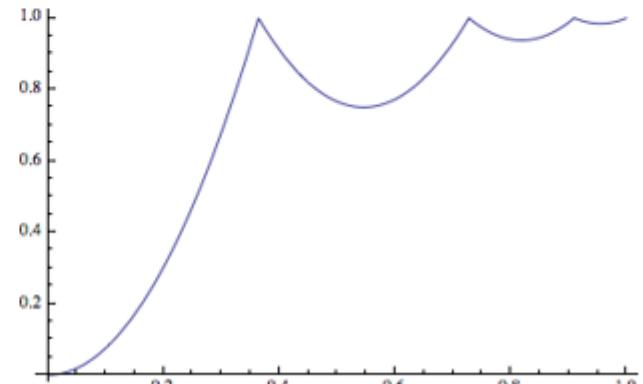
# Easing funkciók

- Beépített és saját funkciók

Easing.BounceIn



Easing.BounceOut



# Forgatás példa

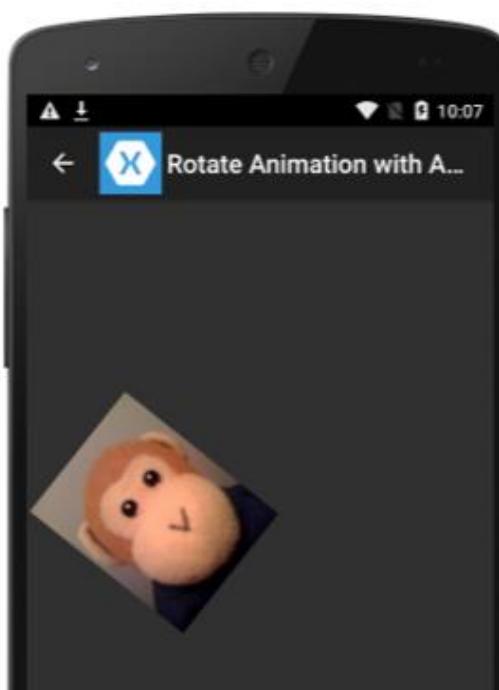
```
img.RotateTo(360, 1000);
```

⌚ (awaitable, extension) System.Threading

Returns a task that performs the rotation t

Usage:

```
bool x = await RotateTo(...);
```



# Animáció osztály

- Animation paraméterezése

```
new Animation(d => img.Rotation = d * 360, 0, 1);
▲ 2 of 2 ▼ Animation(System.Action<double> callback, [double start = 0],
```

- Animáció indítása

```
animation.Commit(img, "rot", | length: 1000);
void Animation.Commit(IAnimatable owner, string name, [uint rate = 16], |
Runs the owner animation with the supplied parameters.
length: To be added.
```

# Animáció paraméterei

- Az osztály csak double értéket animál
- A property módosítása saját kód -> konvertálható
- Callback metódus animáció végén
- Ismétlés: saját funkció ami boollal tér vissza
- Az animáció név alapján lekérdezhető és megszakítható
- Rate: módosítás gyakorisága

# Több animáció összefogása

- Animációk egymásba ágyazása

```
new Animation {
 { 0, 0.5, new Animation (v => image.Scale = v, 1, 2) },
 { 0, 1, new Animation (v => image.Rotation = v, 0, 360) },
 { 0.5, 1, new Animation (v => image.Scale = v, 2, 1) }
}.Commit (this, "ChildAnimations", 16, 4000, null, (v, c) => SetIsEna
```

- Add metódus: relatív eltolás, hossz

# Stílusok

- Több tulajdonság egységes beállítása

```
<Style x:Key="labelStyle" TargetType="View">
 <Setter Property="HorizontalOptions" Value="Center" />
 <Setter Property="VerticalOptions" Value="CenterAndExpand" />
 <Setter Property="FontSize" Value="Large" />
</Style>
```

```
<Label Text="Sample using style." Style="{StaticResource labelStyle}">
```

- Explicit: Style propertyn beállítva kulcs alapján
- Implicit: TargetType-pal beállítva
- A stílusok örökölhetnek egymásból

# A felhasználói élmény alapjai – Fejlesztőknek

Kis-Nagy Dániel      [kis-nagy.daniel@aut.bme.hu](mailto:kis-nagy.daniel@aut.bme.hu)



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

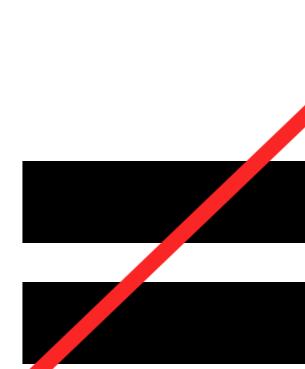
„AZ ÉLMÉNY, AMIT A TERMÉK KIVÁLT A  
FELHASZNÁLÓBAN, AMIKOR VALÓS  
KÖRÜLMÉNYEK KÖZÖTT HASZNÁLJA  
AZT.”

JESSE JAMES GARRETT: THE ELEMENTS OF THE USER EXPERIENCE

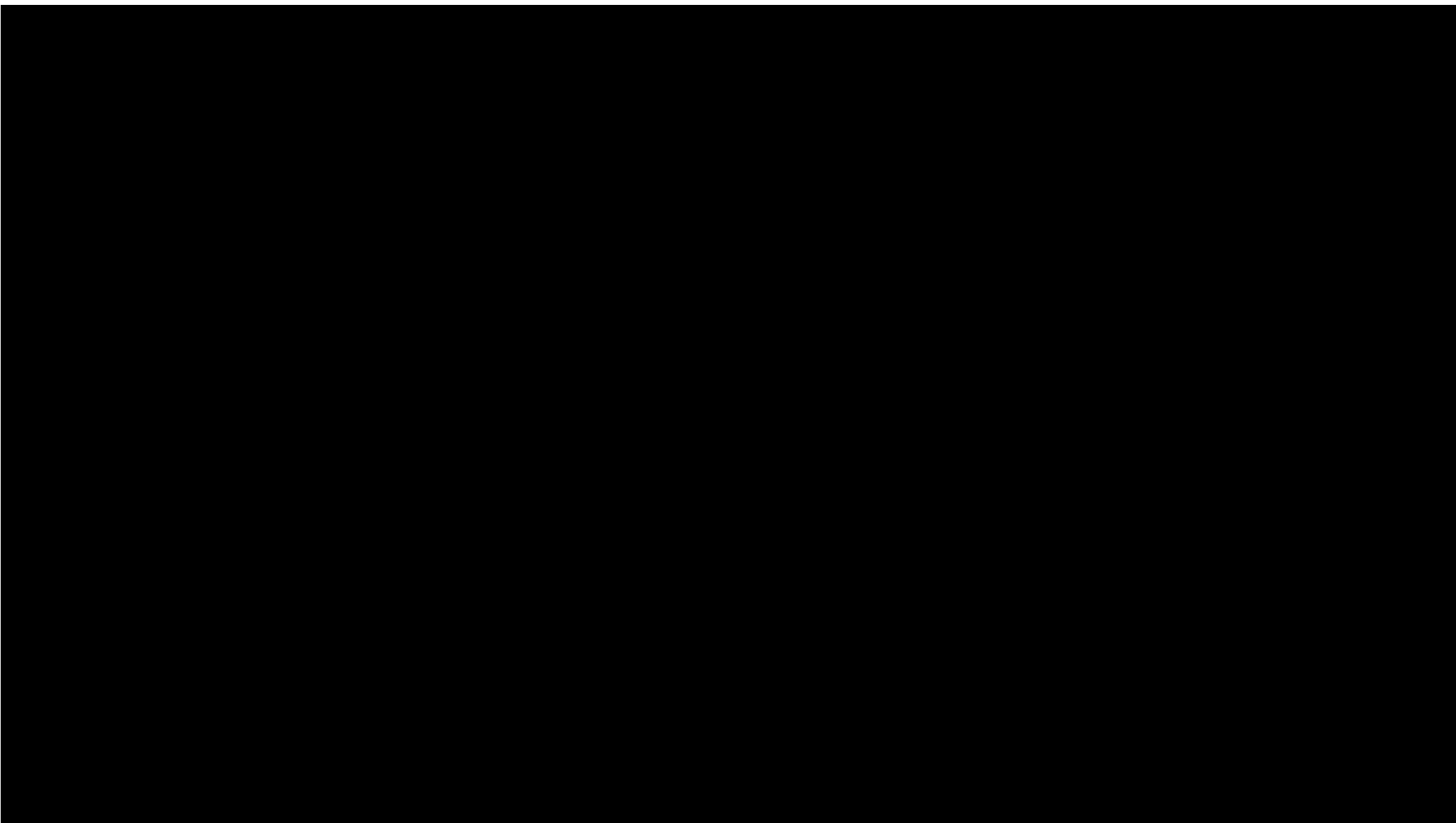
# Felhasználói élmény (User Experience, UX)

- Nem az, **amire** a termék használható, hanem **ahogy** használható.
- A termék megítélésének meghatározó eleme
- „Na milyen az új kávégőződ (ébresztőórád, telefonod, széked...)?”
  - Az élmény, és nem a specifikáció / funkciólista alapján szoktunk válaszolni.  
(Legalábbis a többség.)

# Élmény



# Élmény



A FELHASZNÁLÓI ÉLMÉNY NEM AZT JELENTI, HOGY  
A TERMÉK HASZNÁLATÁ „ÉLMÉNY”, ABBAN AZ  
ÉRTELEMBEN, AHOGY ÉLMÉNY MEGMÁSZNI EGY  
HEGYET VAGY ELOLVASNI EGY KÖNYVET.

A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
TUDOM ÁLLÍTANI AZ ÉBRESZTŐRÁT ANÉLKÜL,  
HOGY FELIDEGESÍTENE, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
TUDOM ÁLLÍTANI AZ ÉBRESZTŐRÁT ANÉLKÜL,  
HOGY FELIDEGESÍTENE, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

- A lényeg az, hogy kényelmesen tudjam használni, nem kell, hogy „élmény” legyen!
- Persze még jobb, ha jól is néz ki, kellemes a hangja stb., de minden másodlagos

# A FELHASZNÁLÓI ÉLMÉNY ALAPJAI

A FELHASZNÁLÓI ÉLMÉNY  
ALAPJAI KÉZZEL FOGHATÓ  
TERMÉKEKNÉL

A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
**TUDOM ÁLLÍTANI AZ ÉBRESZTŐRÁT** ANÉLKÜL,  
HOGY FELIDEGESÍTENE, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

- Rendelkezik a funkcióval, amire szükségem van

# „Klasszikus” Terméktervezés (Product Design)

- Eredeti elképzelés: az a jó termék, ami beteljesíti a funkcióját.
  - Műszakilag legyen rendben, akkor nem lehet baj...
- A külső megjelenés alakulását a funkciók megvalósítása vezérli (és csak az)



„ÉRTELMETLEN AZON VITATKOZNI, HOGY A DESIGN  
SZÜKSÉGES VAGY MEGFIZETHETŐ-E: A DESIGN  
ELKERÜLHETETLEN. A JÓ DIZÁJN ALTERNATÍVÁJA A  
ROSSZ DIZÁJN, ÉS NEM ANNAK HIÁNYA.”

DOUGLAS MARTIN: BOOK DESIGN: A PRACTICAL INTRODUCTION

A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
TUDOM ÁLLÍTANI AZ ÉBRESZTŐÓRÁT **ANÉLKÜL**,  
**HOGY FELIDEGESÍTENE**, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

- Hamar rájövök, hogy működik – adja magát a használata
- „Emberi adottságokra tervezett”

# Formatervezés

- „Kézzel fogható” termékeknél tipikusan ezt szoktuk „dizájnnak” hívni.
  - Megjelenés, tapintás stb.
- Néha a funkció rovására megy...
  - Néha ez nem baj... (de többnyire az)



# A funkciót támogató dizájn

- A helyes használat **adja magát**, szöveges vagy más instrukciók nélkül



# A funkció rovására menő dizájn: Power Mac G4 Cube

- A New Yorki Modern Művészeti Múzeuma büszkén őriz egyet a gyűjteményében
- Melegedési problémákat okozott a tetejére helyezett szellőzőnyílás
  - Adta magát, hogy rápakkoljanak a felhasználók...



A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY  
BE TUDOM ÁLLÍTANI AZ ÉBRESZTŐÓRÁT  
ANÉLKÜL, HOGY FELIDEGESÍTENE, ÉS NEM  
MÁSNAP DÉLBEN DERÜL KI, HOGY VALAMIT  
ELRONTOTTAM.

- Egyértelmű visszajelzésekkel szolgál

# Egyértelmű visszajelzések

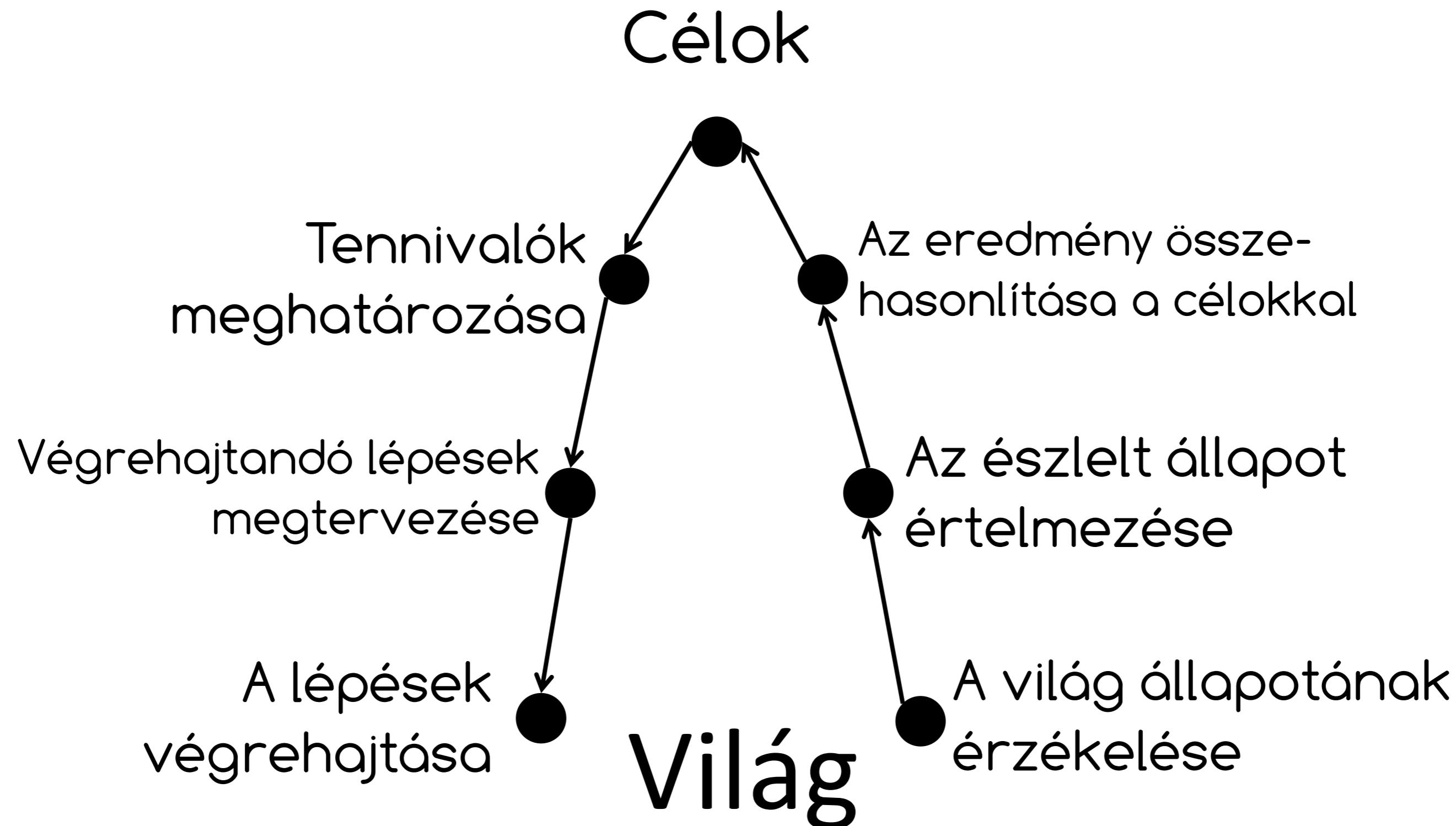
- Azonnali válasz az interakcióra
  - Hanggal
  - Érintésre (pl. gomboknál)
  - Vizuálisan (pl. kijelzők)



A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
TUDOM ÁLLÍTANI AZ ÉBRESZTŐRÁT ANÉLKÜL,  
HOGY FELIDEGESÍTENE, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

- Rendelkezik a funkcióval, amire szükségem van
- Adja magát a használata
- Egyértelmű visszajelzésekkel szolgál

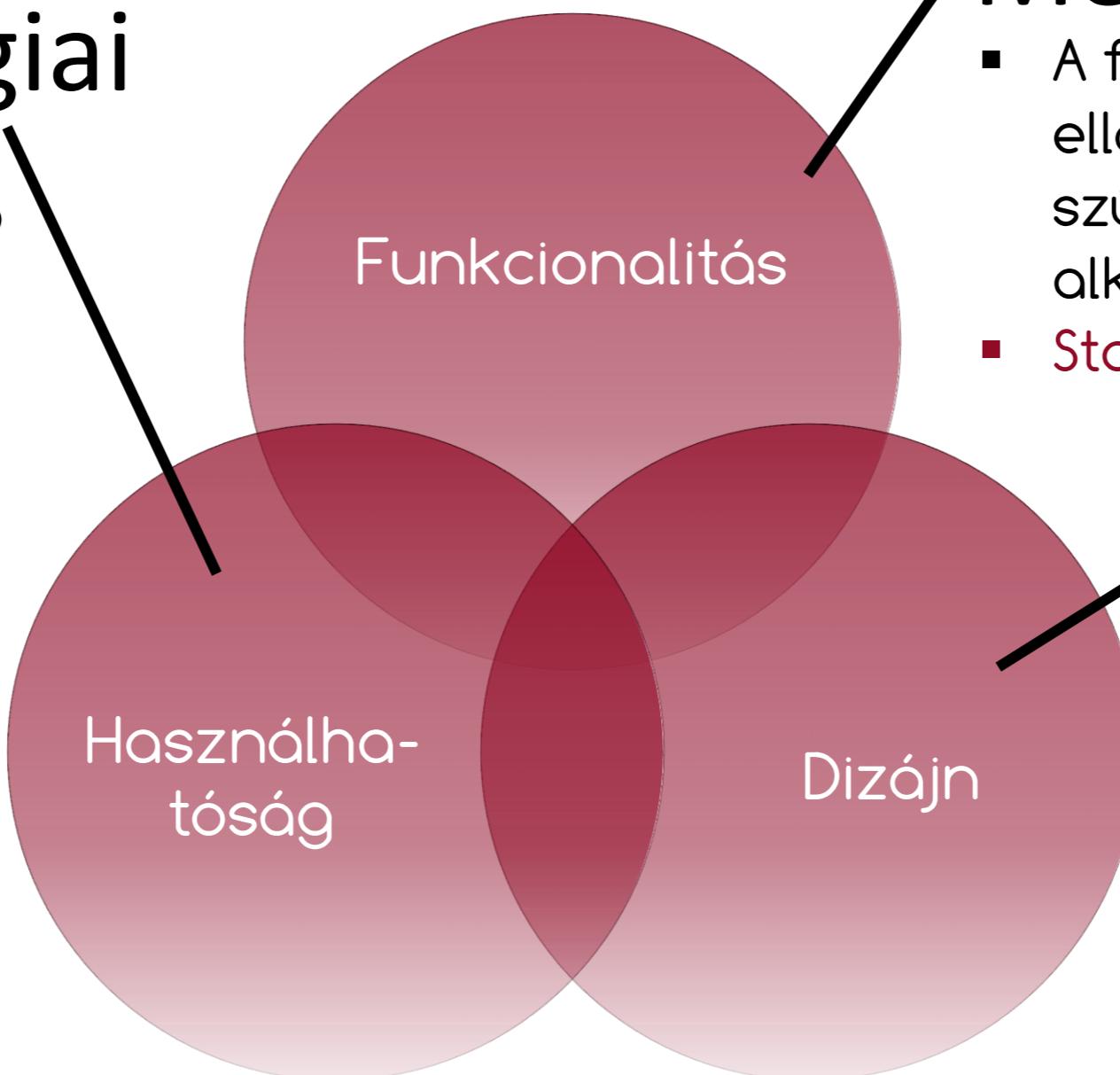
# A dolgok használatának 7 szakasza



# A felhasználói élmény tervezése

## Pszichológiai hozzáállás

- Az emberi viselkedés modelljeit hasznosítja
- Folyamatosan fejlődő tudományos háttér
- Ma még **főleg** tapasztalati tudásra épít (bevált módszereket keresünk)



## Mérnöki hozzáállás

- A funkcionális ellátásához szükséges mérnöki tudás alkalmazása.
- Stabil tudományos háttér

## Művészeti hozzáállás

- Kreativitást, intuíciót igényel
- Állandóan változó divatokat kell kövessen

# Webalkalmazások használhatósága

- A felhasználói élmény fontosabb, mint valaha
  - Nincs se használati utasítás (ha lenne, se olvasná el senki), se tanfolyam
  - „Ha nehéz használni, nem fogom”
    - Senki nem kényszerít (vö. munkahely)
    - Nem veszítek semmit (vö. már megvásárolt dobozos szoftver)
    - A konkurencia csak egy kattintásnyira van
  - De legfőképpen...

HA VALAMI NEM MEGY,  
MAGUNKAT OKOLJUK, ÉS  
HÜLYÉNEK ÉREZZÜK  
MAGUNKAT

# Kinek a hibája, hogy nem megy?

- Mit érzünk?
  - „Biztos rosszul csináltam valamit...”
  - „Biztos nem figyeltem...”
  - „Biztos már eleve rosszul indultam neki....”
- Mi a valóság?
  - A felhasználó úgy cselekszik, ahogy számára az logikus
  - A problémát az okozza, hogy az alkalmazás nem úgy működik, ahogy az a felhasználó számára logikus lenne

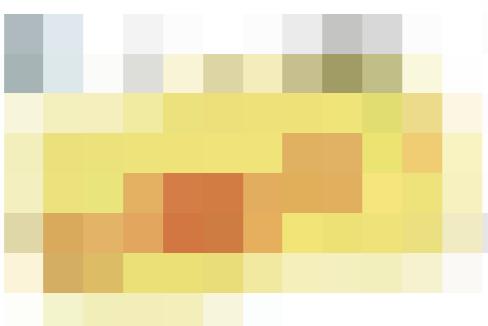
**AKKOR MI A LOGIKUS?**

**AMIN NEM KELL**

**GONDOLKODNI!**

PÉLDA

SZERETNÉK MISKOLCON  
KÖNYVELŐI ÁLLÁST  
TALÁLNI



Keressen Magyarország legnagyobb állásportáljainak kínálatában!

kulcsszavak

kategóriák

Összes

magyar hirdetések  külföldi hirdetések  mindegy

Keresés

**MINŐSÉGBIZTOSÍTÁSI MÉRNÖK  
(Szekszárd)**

Trenkwalder Személyzeti Szolgáltató Kft.  
Baranya/Somogy/Tolna

[bövebben](#)

**Végösszeszerelésért felelős folyamatmérnök (Szombathely)**

Trenkwalder Személyzeti Szolgáltató Kft.  
VasZala

[bövebben](#)

**Termelési kontroller  
(referenciaszám: FE-532)**

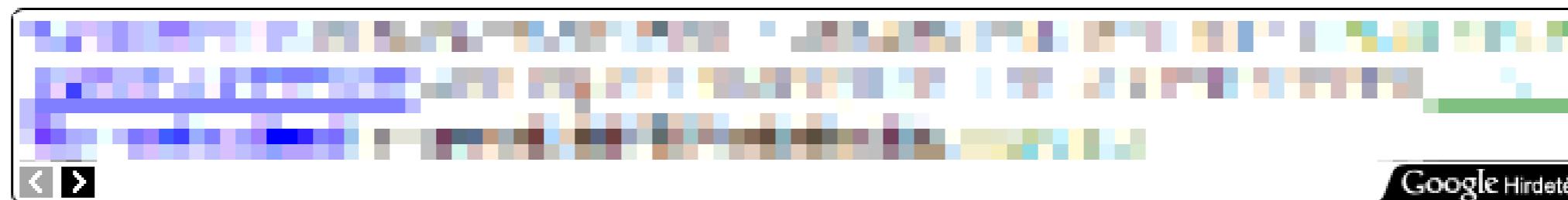
Man at Work Humánszolgáltató és Személyzeti Tanácsadó Kft.  
Tatabánya-Környe/Komárom-Esztergom megye

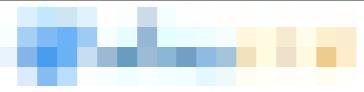
[bövebben](#)

**Gyártástámogató mérnök  
(referenciaszám: BV-427)**

Man at Work Humánszolgáltató és Személyzeti Tanácsadó Kft.  
Veszprém/Veszprém megye

[bövebben](#)





Gyártástervező technológus (Szol...  
Gyártási asszisztens-német nyelv...  
Emelőgép karbantartó - Vác  
Villamosmérnök - Vác  
Informatikus (Vas megye)  
Forrasztási specialista - Tatabá...  
Telefonos értékesítő - áprilisba...  
Customer Service Agent - German  
Folyamatkontroller (működőtöké m...  
Moszaki értékesítő

Összesen 2369 állás  
[részletes keresés >>](#)  
[Önéletrajz feltöltés >>](#)

**Állások**



- Milliónyi állás egy helyen  
IT/Telekommunikáció  
Mérnök  
Tudomány  
Egészségügy  
Emberi erőforrások  
Oktatás/Képzés  
Ellátólánc/Logisztika  
Adminisztráció  
Kiadás/Nyomtatás  
Kreatív / Design  
Ügyvezető igazgatóság  
Értékesítés/Üzleti fejlesztés  
Minőségbiztosítás  
Marketing  
Egyéb  
Könyvelés/Pénzügy  
Állások világszerte

Keressen Magyarország legnagyobb állásportáljainak kínálatában!

kulcsszavak      kategóriák

Összes

magyar hirdetések  külföldi hirdetések  minden

**Keresés**





## Keresés kulcsszóra

Megye kiválasztása ▾

**KERES**

» Részletes keresés

[facebook](#)[twitter](#)[RSS](#)

my

[Regisztráció](#) [Belépés](#) **KARRIEREM**[CV-feltöltés](#)  
[Álláshirdetések mentése](#)  
[Saját profiloldal](#)  
[Állásposta](#) **JÖVŐM**[Karriertippek](#)  
[Munkaerőpiaci hírek](#)

- » [Adminisztráció / Asszisztencia \(105\)](#)
- » [Bank / Biztosítás \(26\)](#)
- » [Cég-, és felső vezetés / Menedzsment \(24\)](#)
- » [Értékesítés / Kereskedelem \(86\)](#)
- » [Humán Erőforrások / Munkaügy \(51\)](#)
- » [Jog / Közigazgatás \(10\)](#)
- » [Kreatív / Művészletek \(0\)](#)
- » [Marketing / PR / Média \(27\)](#)
- » [Minőségbiztosítás \(100\)](#)
- » [Pénzügy / Számvitel \(90\)](#)
- » [Újságírás / Szerkesztőség \(0\)](#)
- » [Vendéglátóipar / Idegenforgalom \(17\)](#)
- » [Agrár / Mezőgazdaság \(3\)](#)
- » [Beszerzés / Logisztika \(178\)](#)
- » [Egészségügy \(16\)](#)
- » [Fordítás / Tolmácsolás \(0\)](#)
- » [IT / Informatika \(553\)](#)
- » [Képzés/Oktatás \(11\)](#)
- » [Kutatás / Fejlesztés \(21\)](#)
- » [Mérnök / Műszaki \(244\)](#)
- » [Pályakezdő / Szakmai gyakorlat \(46\)](#)
- » [Szakmunka / Fizikai munka \(152\)](#)
- » [Ügyfélszolgálat \(101\)](#)
- » [Egyéb \(14\)](#)

Adatbázisunkban jelenleg 1875 álláshirdetés található.

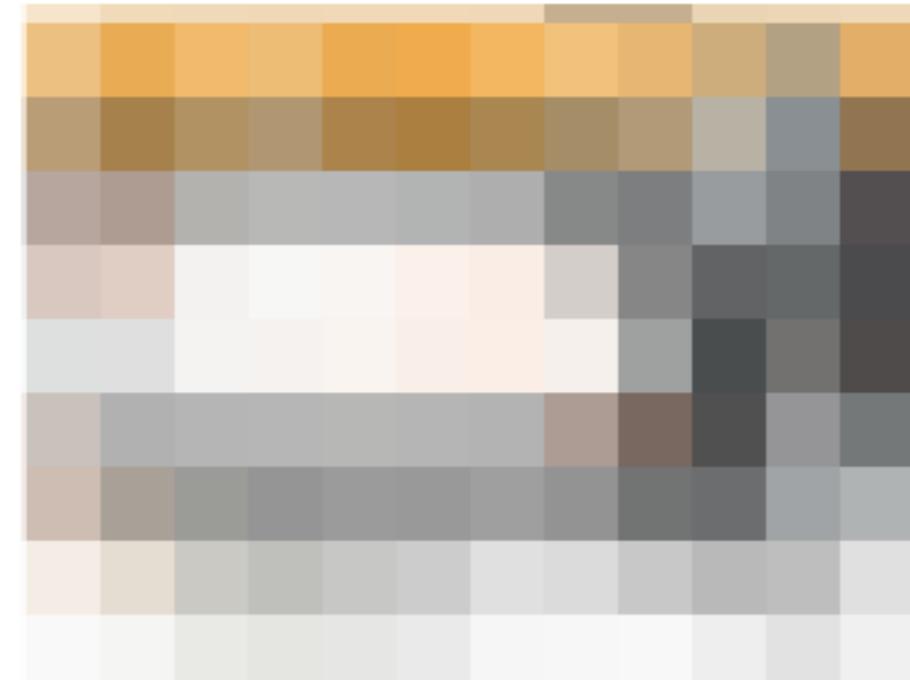
**» Kiemelt álláshirdetések**

**Test Engineers and Senior Test Engineers**  
Scientific Games Kft.

**C/C++ Developers**  
Scientific Games Kft.

**PL/SQL Developers**  
Scientific Games Kft.

**Oracle Forms/ Reports Developers**  
Scientific Games Kft.

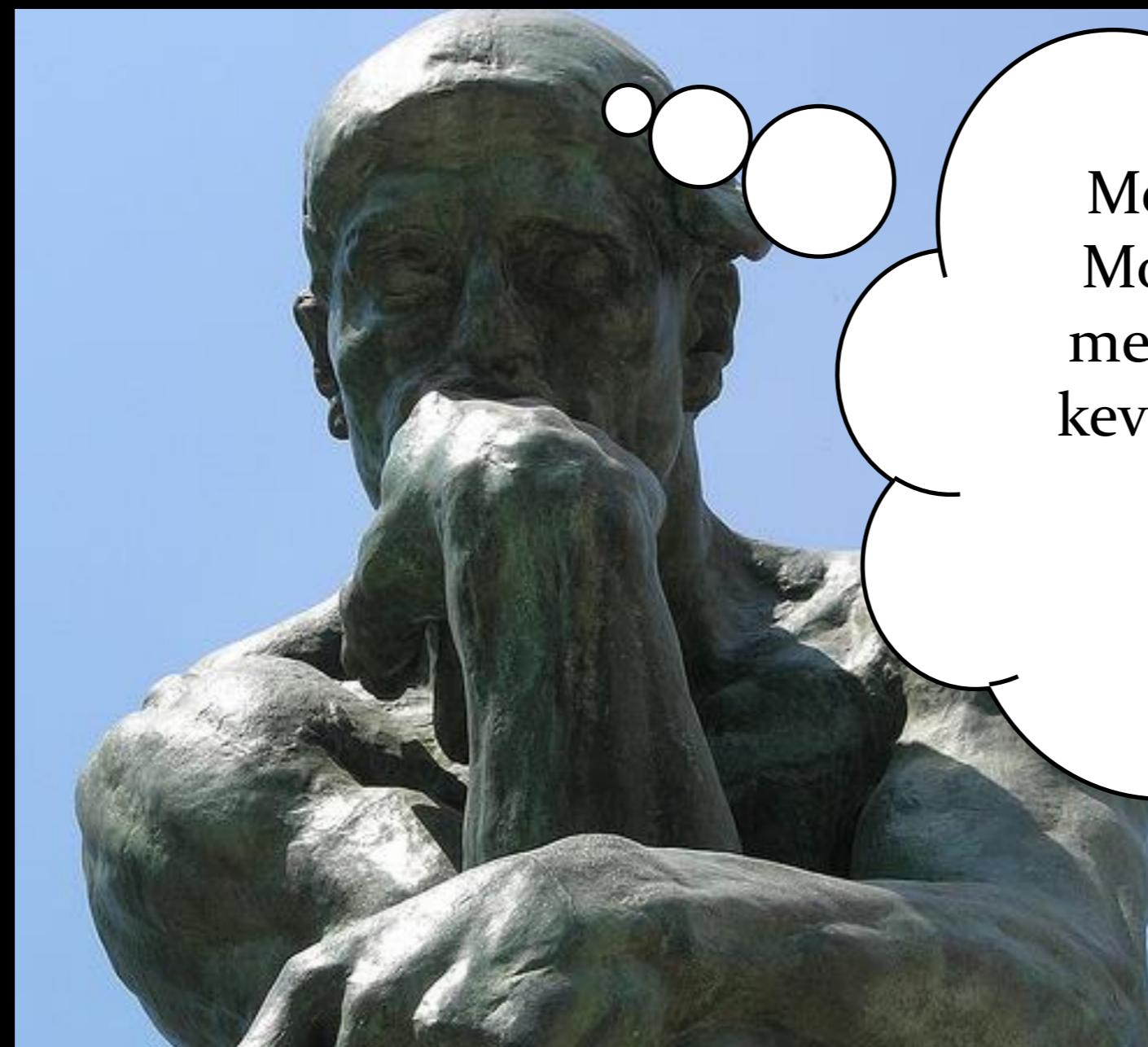
[Hirdessen itt!](#)

Keresés kulcsszóra

Megye kiválasztása

KERES

» Részletes keresés



Megint kulcsszavak?  
Mondjuk itt legalább  
megye van... persze az  
kevés. Talán a részletes  
keresést kéne  
kipróbálnom...

- Főoldal
- Keresés
- Állásaim
- Kereséseim
- Hírlevélem
- Önéletrajzaim
- Álláspiaci hírek

Álláskeresőknek

- + Belépés
- + Regisztráció

Munkaadóknak

- + Ajánlataink
- + Bejelentkezés
- + Regisztráció
- + Hirdetésfeladás

mit:

hol:

**Keresés**

**+ Részletes keresés**

Keresett munka: asszisztens, mérnök, orvoslátogató, programozó, területi képviselő    Még több állás kulcs >>    Összesen 3815 állás, munka 1165 hirdetőtől

**Kategóriák**

- [Adminisztráció / Irodai munka \(290\)](#)
- [Bank / Biztosítás / Tőzsde \(195\)](#)
- [Egészségügy / Szépség \(89\)](#)
- [Építőipar / Ingatlan \(68\)](#)
- [Értékesítés / Kereskedelelem / Üzlet \(698\)](#)
- [Fizikai munka / Segédmunka \(52\)](#)
- [Gyártás / Termelés / Mérnök \(955\)](#)

**Területek**

- [HR / Emberi erőforrás / Munkaügy \(160\)](#)
- [IT / Telekommunikáció \(712\)](#)
- [Jog / Közigazgatás \(64\)](#)
- [Környezet / Mezőgazdaság \(34\)](#)
- [Marketing / Média / Művészet \(151\)](#)
- [Oktatás / Tudomány \(30\)](#)
- [Pénzügy / Számvitel / Menedzsment \(534\)](#)

**+ Részwmunkaidős állások**   **+ Pályakezdőknek**   **+ Legfrissebb állások**

- [Szakmai-Gyakornoki programok \(65\)](#)
- [Szakmunka \(149\)](#)
- [Szállítás / Logisztika \(296\)](#)
- [Személy- és vagyonvédelem \(2\)](#)
- [Ügyfélszolgálat / Ügyfélkapcsolat \(419\)](#)
- [Vendéglátás / Idegenforgalom \(43\)](#)

[Részletes álláskeresés, munkakeresés](#)

#### Kiemelt állásajánlataink



#### Pénzügyi munkatárs

Pénzügyi és management szolgáltatást nyújtó cég

#### Experienced Java developer

ChemAxon Kft.

#### Rendszergazda

Kék Vonal  
Gyermekkritikus  
Alapítvány

#### Junior Kontroller

villamos energia  
kereskedelemmel  
foglalkozó leánycég

#### ▲ Önéletrajz minták

#### ✉ Kísérőlevél minták

#### 💡 Ellenőrizze az önéletrajzát! TESZT

#### ❓ Mit írunk az önéletrajzba? TESZT

#### ❗ Álláskeresési tippek

#### ❓ Kérdezze a karrier-tanácsadókat!

#### 🌐 Munkaügyi központok

#### ☎ Szabadság kalkulátor

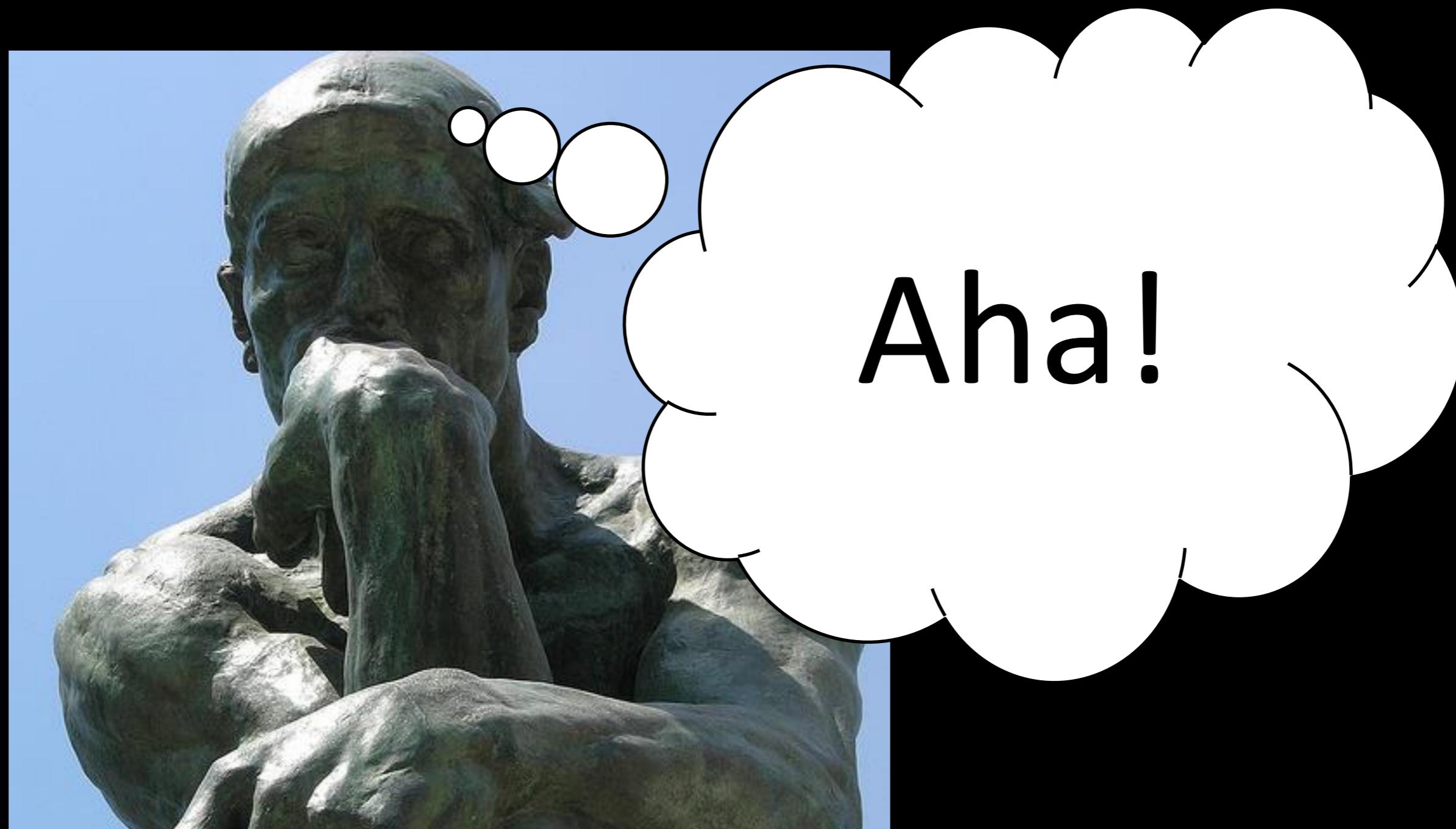
#### 💰 Bériránytű

**TIPP!** A munkaadók most ezt az állást, munkát keresik leginkább:  
villamosmérnök, fejlesztő, projektmanager, könyvelő, minőségbiztosítás, SSC (Shared Service Center)

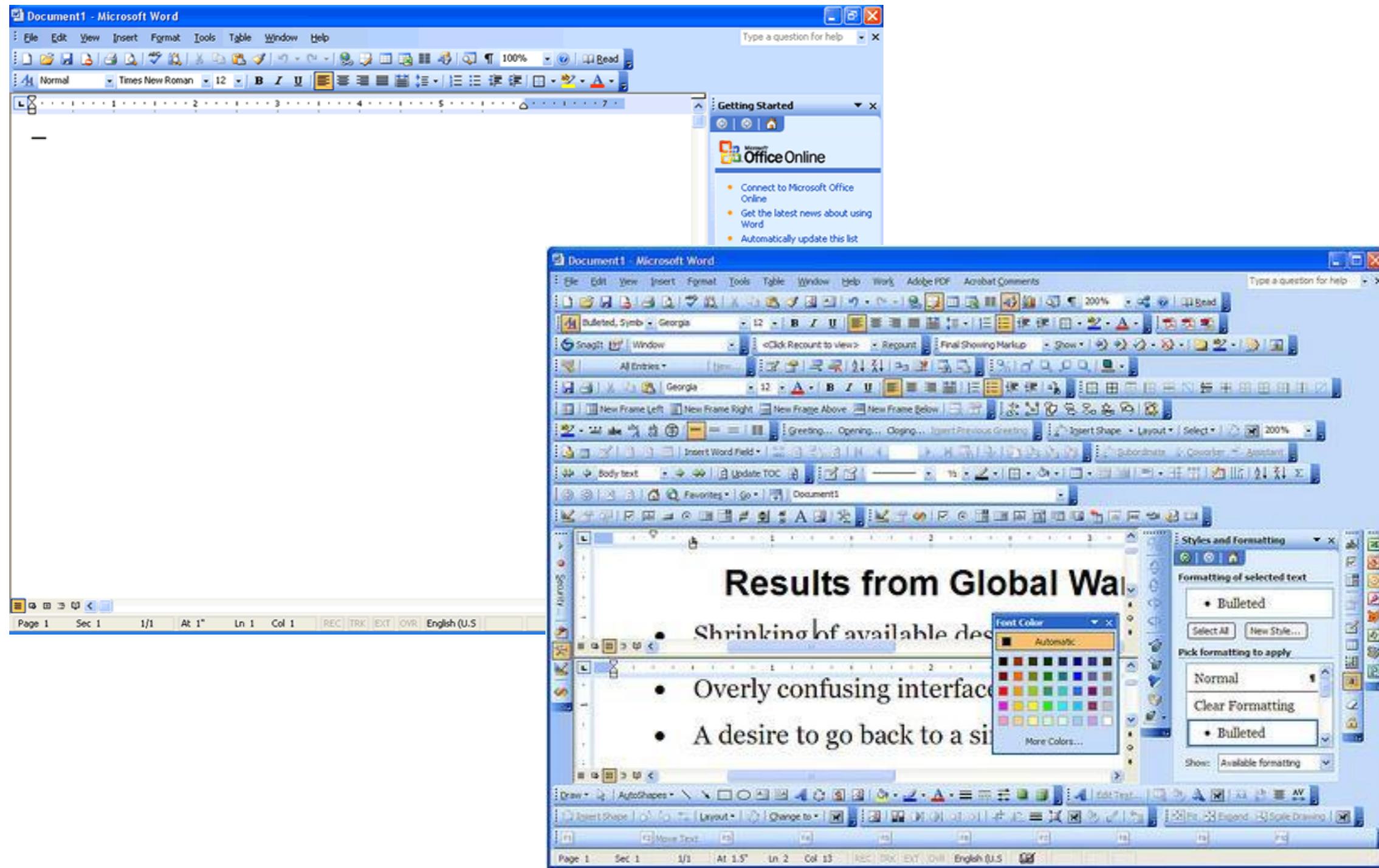
**mit:** állás, kulcsszó vagy vállalat neve

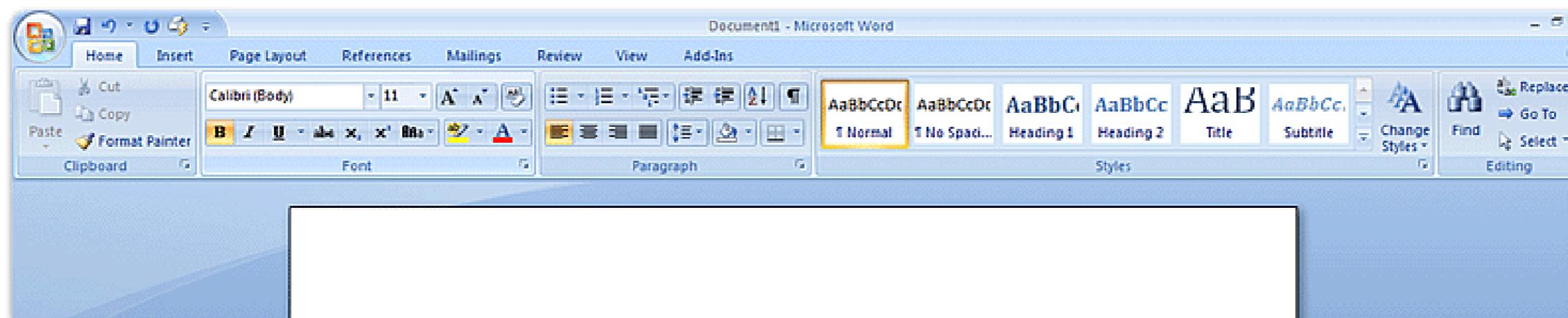
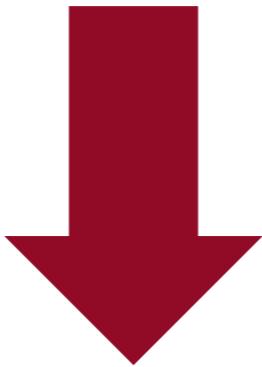
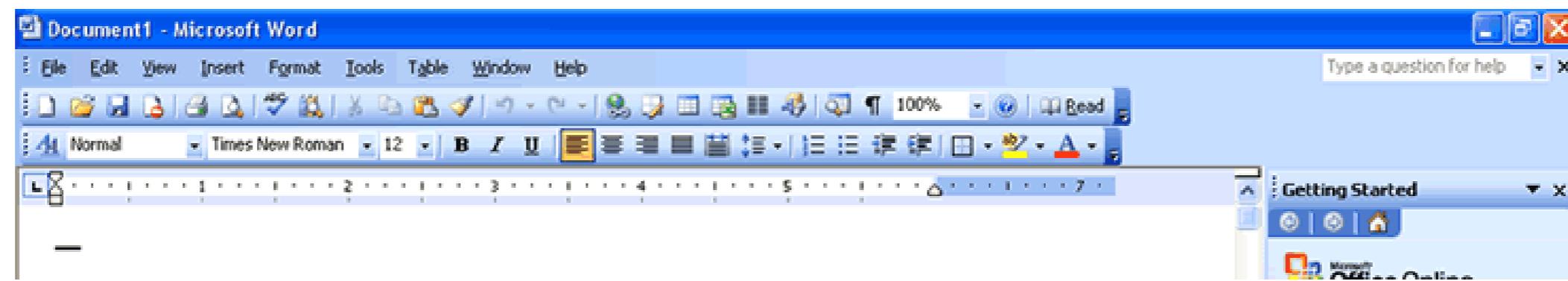
**hol:** város vagy megye

Keresés



# MI A HELYZET AZ IRODÁBAN?





# „Jó volt az úgy!”

- Közhiedelem
  - Az Office jó úgy, ahogy van
  - Úgyis mindenki csak töredékét használja a funkcióknak
  - Az Office 6.0/97/2000 –ben minden benne volt, ami kellett
- Valóság
  - „Ezt biztos, hogy meg lehet csinálni valahogy, de nem tudom, melyik menüben keresse...”
  - „Sokat tud az Office, biztos sokat segítene, ha profibban tudnám használni...”

# Az Office UI kudarai

- Évről-évre új funkciókkal bővült...  
...de senki nem találta meg őket
- Az Office egyre bonyolultabb lett...  
...és a helyzet évről-évre romlott
- A felhasználók szeretnének hatékonyabban dolgozni...  
...de beletörődnek, hogy semmi nem változik

Font: Tms Rmn Pts: 10

Style: Normal

0. 1. 2. 3. 4. 5. 6.

|  
—

# Word for Windows 1.0 (1989)

Common screen resolution: 640x480

Number of toolbars: 2

File Edit View Insert Format Tools Table Window Help



Normal Tms Rmn 10 B I u

0 1 2 3 4 5 6

|

# Word for Windows 2.0 (1992)

Common screen resolution: 640x480

Number of toolbars: 2

User Interface Additions:

Nested Dialog Boxes

Microsoft Word - Document2

File Edit View Insert Format Tools Table Window Help

RBC Spelling and Grammar... F7

Language Set Language... Word Count... Thesaurus... Shift+F7

AutoSummarize... Hyphenation...

Normal Garamond

AutoCorrect... Look Up Reference...

Track Changes Merge Documents...

Protect Document...

Mail Merge...

Envelopes and Labels...

Letter Wizard...

Macro Templates and Add-Ins...

Customize...

Options...

100% ?

Set Language...

Thesaurus... Shift+F7

Hyphenation...

1 2 3 4

Tables and Borders

RICH ANDREW

Click here and type objective]

EXPERIENCE

1990–1994	Arbor Shoe	Southridge, SC
<i>National Sales Manager</i>		
■ Increased sales from \$50 million to \$100 million.		
■ Doubled sales per representative from \$5 million to \$10 million.		
■ Suggested new products that increased earnings by 23%.		
1985–1990	Ferguson and Bardell	Southridge, SC
<i>District Sales Manager</i>		
■ Increased regional sales from \$25 million to \$350 million.		
■ Managed 250 sales representatives in 10 Western states.		
■ Implemented training course for new recruits — speeding profitability.		
1980–1984	Duffy Vineyards	Southridge, SC
<i>Senior Sales Representative</i>		

Draw AutoShapes

Click and drag to create table and to draw rows, columns and borders.

Start Microsoft Word - Doc...

3:01 AM

Document1 - Microsoft Word

File Edit View Insert Format Tools Table Window Help

Type a question for help

Table Columns Times New Roman 10 B I U Table Columns

100% Read

1/2 Recount <Click Recount to view> Recount

1 2 3 4

Value1 Value2 Value3 Value4

Search Results

30 results from Office Online

- Update an index, table of contents, table of figures, or table of authorities
- Change the appearance of a table of contents, index, table of authorities, or table of figures
- Delete an index, table of authorities, or table of figures
- Split a table
- Table of Contents II: Advanced TOCs, long documents, and other tables

Search

Microsoft Office Online

tables

Can't find it?

Draw AutoShapes

Page 1 Sec 1 1/1 At 1.1" Ln 2 Col 1 REC TRK EXT OVR

Microsoft Office Word ...

Change the appearance of a table of contents, index, table of authorities, or table of figures

Show All

- On the Insert menu, point to Reference, click Index and Tables, and then click the tab you want.
- In the Formats box, click From template, and then click Modify.
- In the Styles box, click the style you want to change, and then click Modify.
- To add the new style definition to your template, select the Add to template check box.
- Under Formatting, select the options you want.

Notes

- To change the appearance of a table of contents in a Web frame, you must first position the insertion point inside the frame that contains the table of contents. To change the appearance of a table of contents in a Web frame that uses hyperlinks, change the properties of the hyperlink style.
- To edit the text of an individual entry, locate the source of the entry, modify it, and then update the entire table of contents, index, table of authorities, or table of figures.

Please let us know if this content was helpful.

Rate this content:

★★★★★

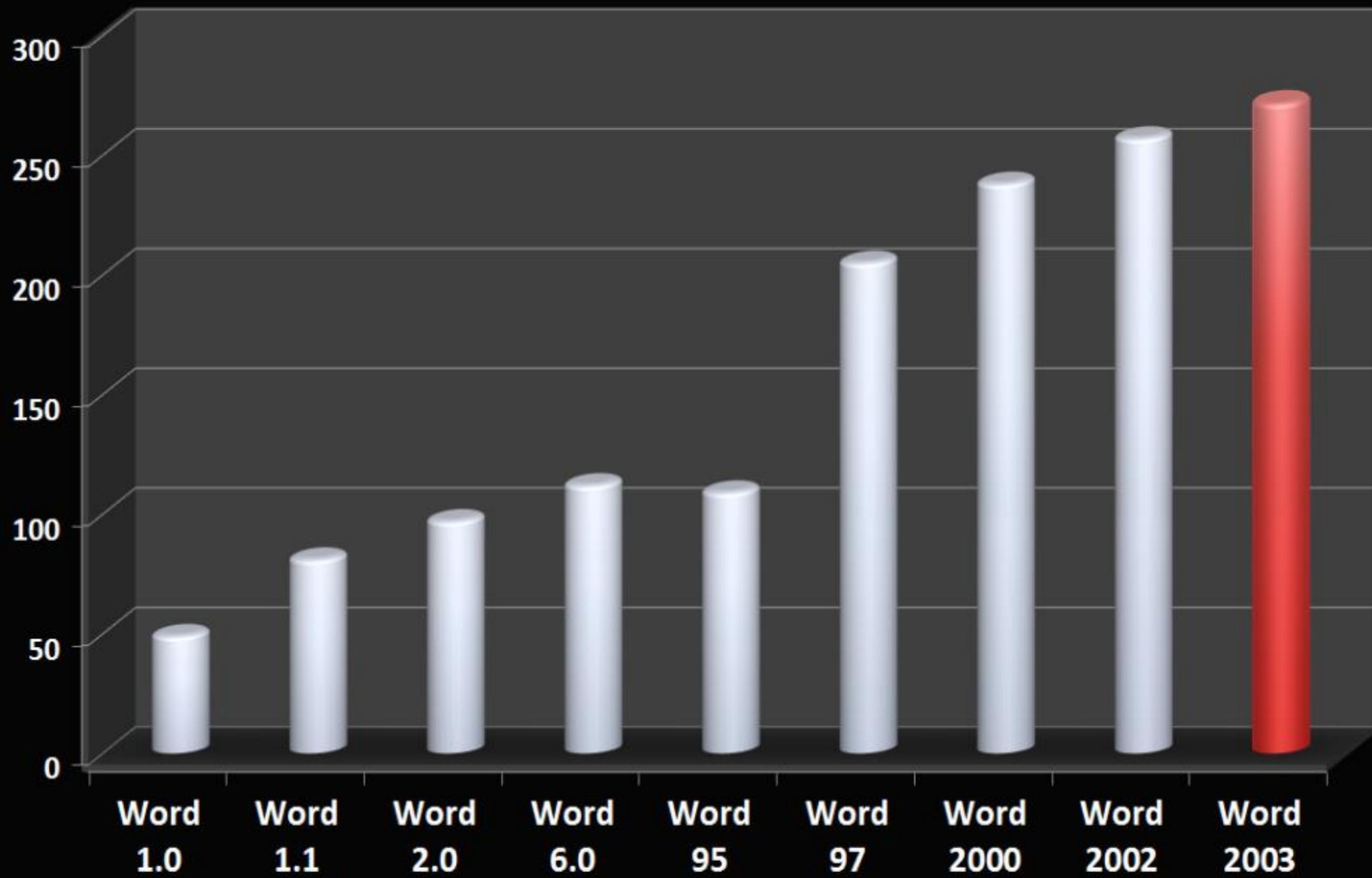
Tell us why you rated the content this way (optional):

Remaining characters: 650

Contact Us Privacy Statement

3:45 AM

# Menu Items in Microsoft Word



# Hogy jutottunk el idáig?

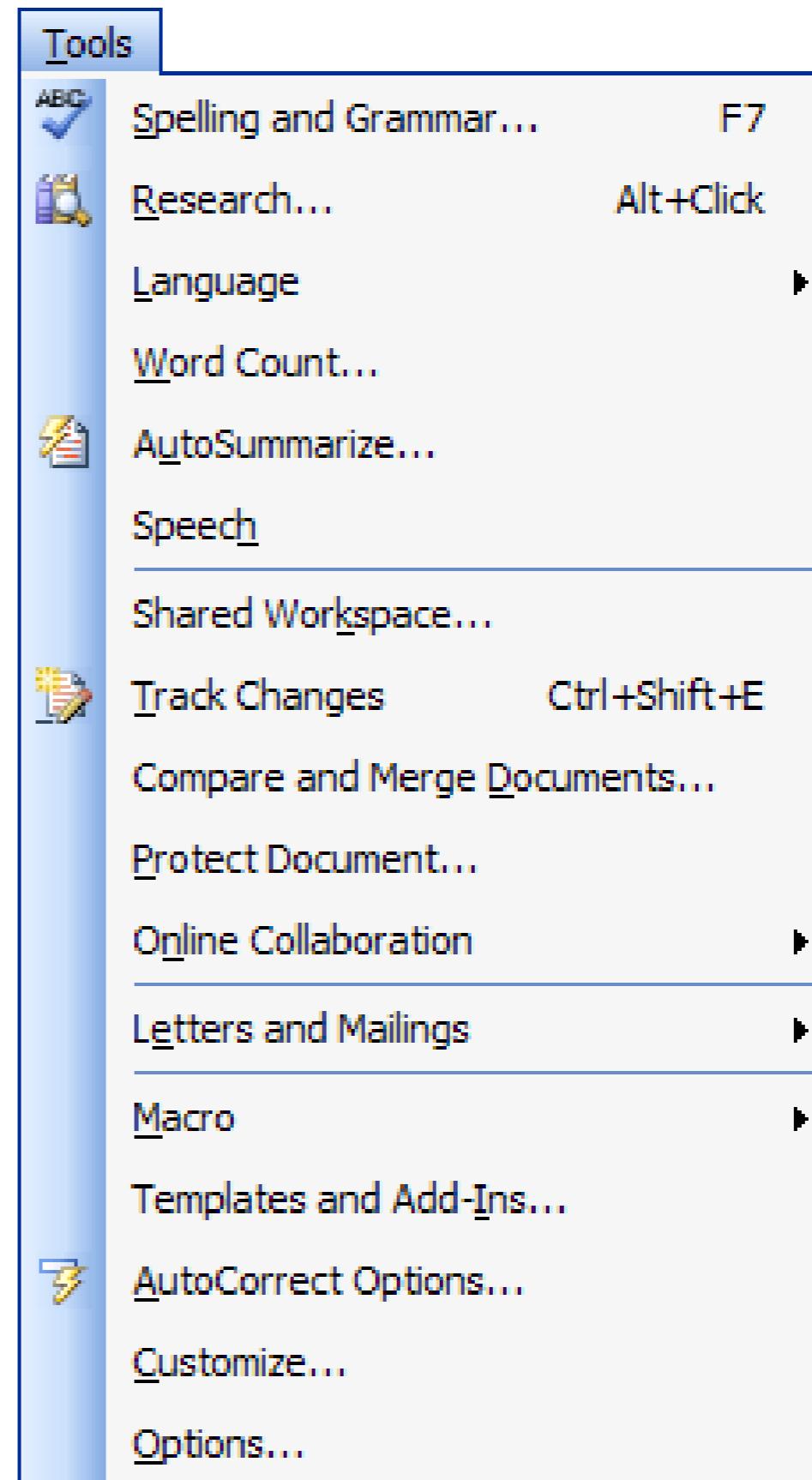
- A menüket és toolbarokat sokkal kevésbé gazdag funkcionáláshoz találták ki
- Az Office funkcionálása túlmutat azon, amiket a régi UI megoldások használhatóan reprezentálni tudnak
- Egy adott funkciót nehezebb ma megtalálni, mint a '90-es években volt
  - „Valahogy biztos meg lehet csinálni, de azt sem tudom, hol keressem...”



# Mi a közös az alábbi parancsokban?

- Find out the current number of words
- Turn on speech command and control
- Create a SharePoint Document Workspace
- Print Envelopes
- Open the Visual Basic Editor
- Turn on hyphenation
- Merge the contents of multiple documents
- Start a web conference
- Tweak AutoCorrect settings

# MIND A TOOLS MENÜBEN VANNAK!

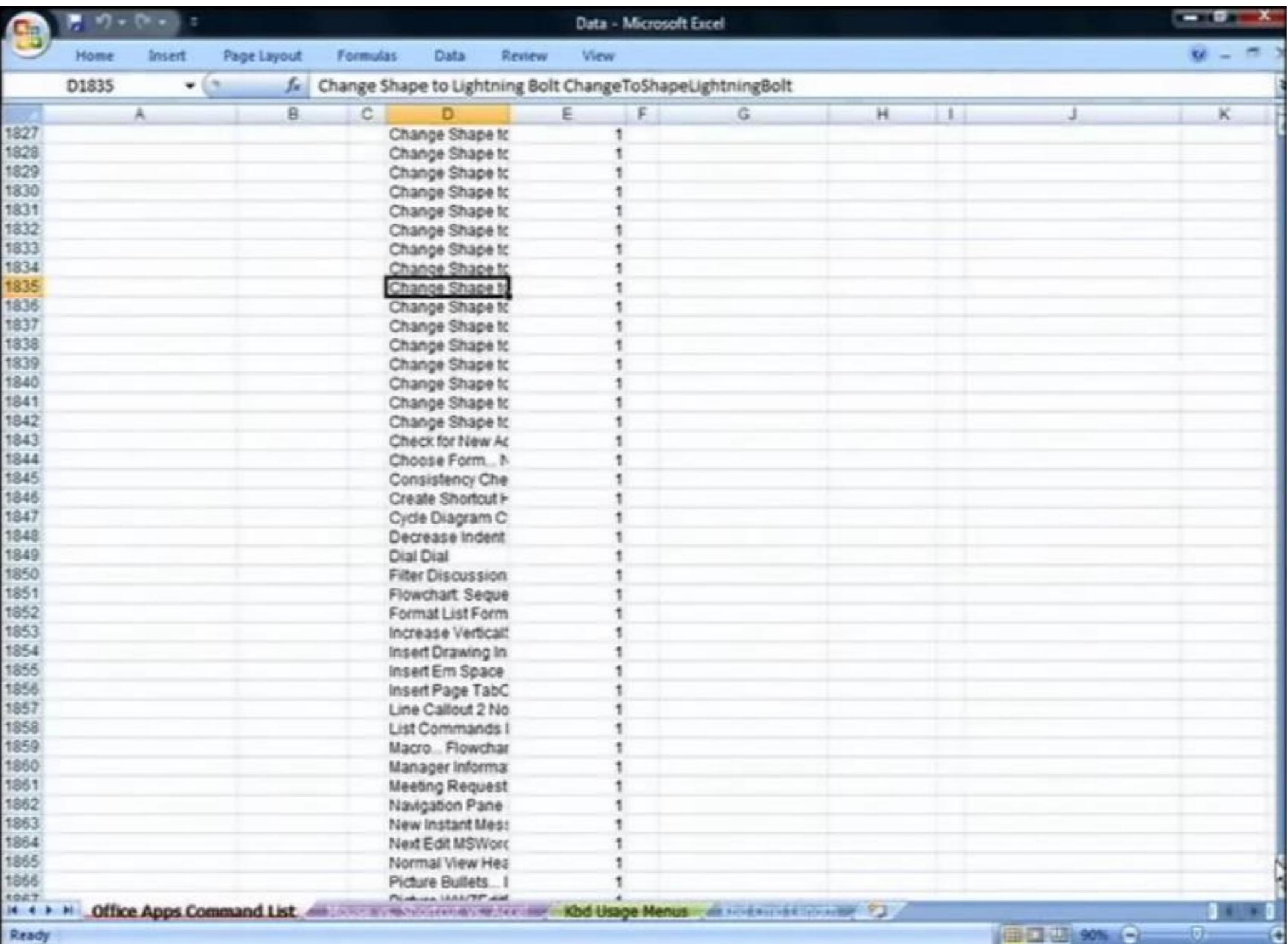


HOGYAN LEHETNE EZEN  
JAVÍTANI?  
MÉRJÜNK!

# Telemetria: parancsok gyakorisága

Data - Microsoft Excel								
A1	B	C	D	E	F	G	H	I
Excel	1439 Controls	Word	1891 Controls	PowerPoint	1317 Controls	Outlook	946 Controls	J
Control	Occurrences	Control	Occurrences	Control	Occurrences	Control	Occurrences	
Paste Paste	2,251,065	Char Right MSWc	23,364,292	Next SlideShowNextSlide	1,603,763	Delete Delete	14,015,442	
Copy Copy	1,816,524	Char Left MSWon	19,991,706	Clear Clear	1,483,437	Send SendDefault	1,844,122	
Save Save	1,202,393	Line Down MSWl	17,486,297	Next Slide NextSlide	1,070,947	Reply Reply	1,559,767	
Clear Clear	1,106,419	Contents WordCl	14,405,107	Down NudgeDown	562,466	SendReceive DELIVERMAIL	1,477,220	
File File	961,441	Line Up MSWordl	11,316,036	Right NudgeRight	537,643	Unknown Control - 32828	1,326,397	
Close CloseWindow	564,326	Paste Paste	2,519,281	Previous Slide Previous	517,671	Unknown Control - 32768	1,135,041	
Undo Undo	487,420	Page Down Page	2,261,499	Up NudgeUp	500,949	New Item NewItemSplitMenu	1,030,391	
Cells... FormatCells	443,194	Word Right MSWi	2,210,148	Left NudgeLeft	458,660	Close CloseSDI	928,995	
Delete Rows DeleteRows	417,272	Send SendDefault	2,115,238	Paste Paste	391,538	Save and Close SaveAndClose	731,195	
Edit Edit	403,886	Char Right Extend	1,999,060	Previous SlideShowPre	283,937	Tools Tools	703,530	
Cut Cut	354,724	Word Left MSWor	1,969,683	Copy Copy	238,400	Forward Forward	609,496	
Insert InsertCells3NoEllipsi	327,284	Line Down Extend	1,797,182	Save Save	225,491	File File	565,071	
Print Preview PrintPreview	321,470	Char Left Extend	1,516,201	File File	223,538	Reply to All ReplyAll	517,458	
Insert Insert	299,136	File File	1,495,725	Undo Undo	203,689	Mark as Read MarkAsRead	496,120	
Bold Bold	294,936	Save Save	1,469,669	End Show EndSlideSho	179,209	Empty !<0#25>! Folder Empt	358,023	
Data Data	280,051	End of Line EndC	1,366,884	Font Size: FontSizeCom	135,500	Add Sender to Blocked Sender	334,463	
Format Format	274,343	Page Up PageUp	1,346,090	Insert Insert	133,066	View View	317,498	
Fill Color FillColorSplitDrop	268,992	Copy Copy	1,273,135	Close CloseWindow	98,038	Edit Edit	314,120	
Tools Tools	217,823	Undo Undo	1,226,805	Edit Edit	94,359	Compose Compose	244,224	
Open... Open	214,511	Close CloseWin	1,167,329	New Slide NewSlide	80,050	Custom Custom	243,179	
Save As... SaveAs	203,908	Word Left Extend	801,768	View View	77,883	Move to Folder MoveToFolderB	216,856	
Merge and Center CenterAc	183,815	Word Right Extend	702,308	Format Format	73,382	Move to Folder... MoveToFolder	216,396	
Window Window	172,612	Bold Bold	694,519	Bold Bold	67,017	ExpanderButton	193,583	
AutoSum AutoSum	169,031	Next Cell MSWon	619,752	Cut Cut	65,995	Select All SelectedAll	188,896	
Borders Borders	160,838	Font Size: FontSu	590,456	Decrease Font Size Shr	64,082	Copy Copy	187,418	
Print... Print	158,398	no spelling suggi	585,780	Slide Show SlideShow	55,248	All Groups SyncGroups	173,242	
Font Size: FontSizeCombo	156,930	Start of Line Start	513,435	Format Object... Format	50,825	Download Pictures InfoBarSho	163,516	
Border BorderSwatch	148,244	Insert Insert	478,324	Increase Font Size Gro	50,715	Previous Item PreviousItemSpl	151,937	
Redo Redo	147,598	Edit Edit	473,309	Print... Print	50,566	Insert Insert	133,034	
Paste Special... PasteSpec	144,552	Delete Back Wor	427,743	Slide Show Slideshow	49,076	Go Web Go	127,775	
Rows InsertRows	144,154	Print... Print	417,314	Save As... SaveAs	48,545	Look For StripSearchWhat	120,752	
Print PrintDefault	143,809	Format Format	416,894	Close CmdBarClose	47,302	Check Names MailCheckNam	120,678	
Center Centered	143,258	Tools Tools	410,668	Entrance CustomAnima	46,213	Mark as Unread MarkAsUnrea	115,655	
Find... Find_Excel	141,578	Line Up Extend M	405,893	Font Color FontColor	45,855	Options... Options	113,980	
Color Scheme FillColorSch	139,992	View View	372,515	Tools Tools	45,074	Next Item NextItemSplitMen	111,282	
Close CmdBarClose	136,166	Cut Cut	371,685	Picture Picture	42,584	Print... Print	110,337	
View View	130,496	Save As... SaveAs	335,371	CustomAnimationEffect	41,856	Save Save	108,871	
Unknown Control - 32768	128,435	Print PrintDefault	299,446	Font FontCombo	40,868	Mail Message NewMessage	106,189	

# Telemetria: parancsok gyakorisága



A screenshot of a Microsoft Excel spreadsheet titled "Data - Microsoft Excel". The spreadsheet contains a single column of data, with rows numbered from 1827 to 1866. The data consists of two columns: a date/time column (A) and a command name/usage count column (D). The command names are repeated frequently, such as "Change Shape To" and "Check for New Ac". The row number 1835 is highlighted with a yellow background.

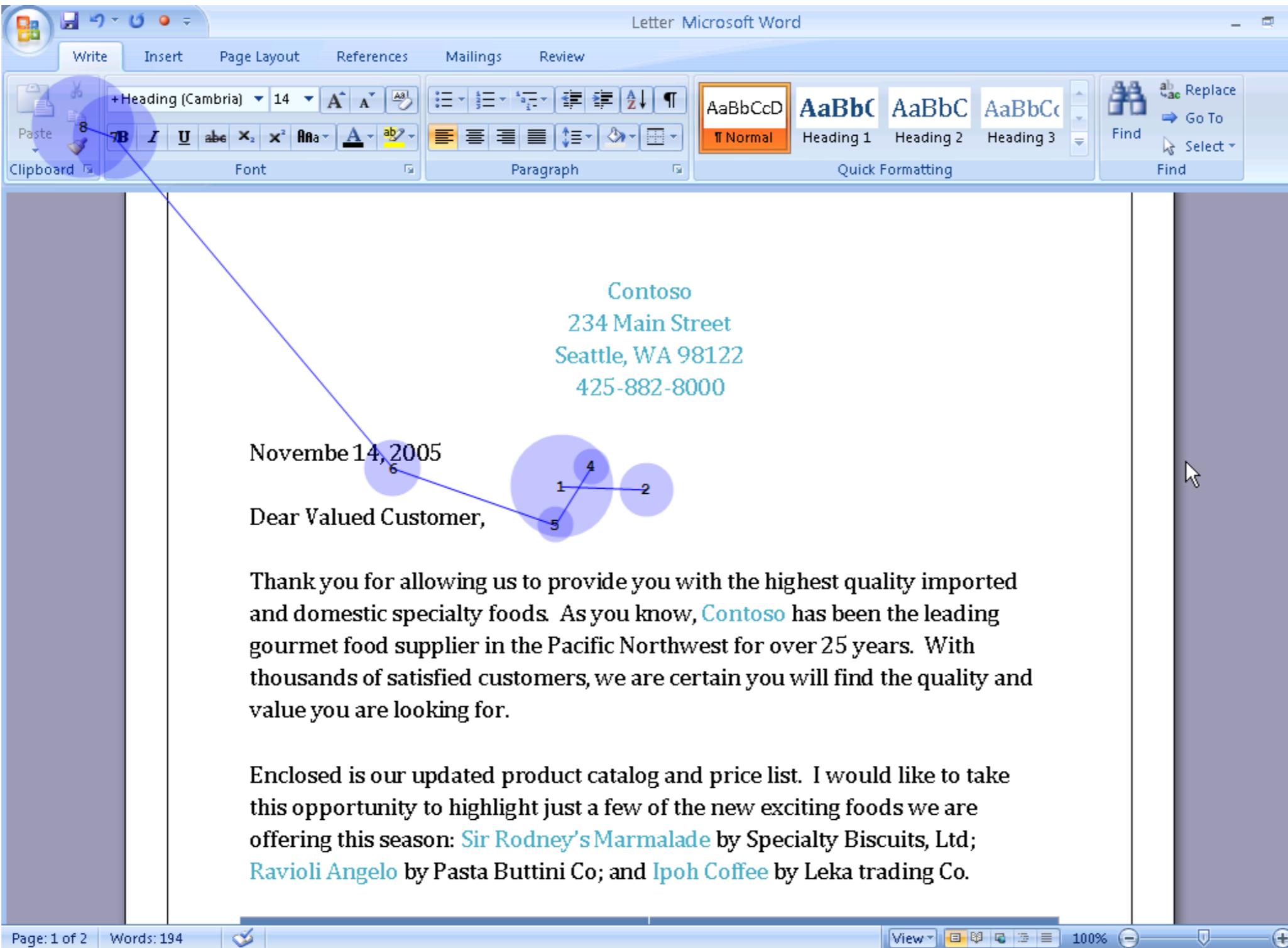
1827		Change Shape To	1
1828		Change Shape To	1
1829		Change Shape To	1
1830		Change Shape To	1
1831		Change Shape To	1
1832		Change Shape To	1
1833		Change Shape To	1
1834		Change Shape To	1
1835		Change Shape To	1
1836		Change Shape To	1
1837		Change Shape To	1
1838		Change Shape To	1
1839		Change Shape To	1
1840		Change Shape To	1
1841		Change Shape To	1
1842		Change Shape To	1
1843		Check for New Ac	1
1844		Choose Form... I	1
1845		Consistency Che	1
1846		Create Shortcut F	1
1847		Cycle Diagram C	1
1848		Decrease Indent	1
1849		Dial Dial	1
1850		Filter Discussion	1
1851		Flowchart: Seque	1
1852		Format List Form	1
1853		Increase Vertical	1
1854		Insert Drawing In	1
1855		Insert Em Space	1
1856		Insert Page TabC	1
1857		Line Callout 2 No	1
1858		List Commands I	1
1859		Macro... Flowchar	1
1860		Manager Informa	1
1861		Meeting Request	1
1862		Navigation Pane	1
1863		New Instant Mess	1
1864		Next Edit MSWord	1
1865		Normal View Hea	1
1866		Picture Bullets... I	1
1867		PrintArea MAST	1

# Telemetria: gyorsbillentyűk

The screenshot shows a Microsoft Excel spreadsheet titled "Data - Microsoft Excel". The table has columns labeled A through M. Column A contains numerical row identifiers from 1 to 35. Column B lists control names. Columns C through H show percentages of usage for different input types: Total Clicks, Mouse, Short, Accel, and Total. The data indicates that most controls are used via mouse clicks, with "Save" being the most frequent overall.

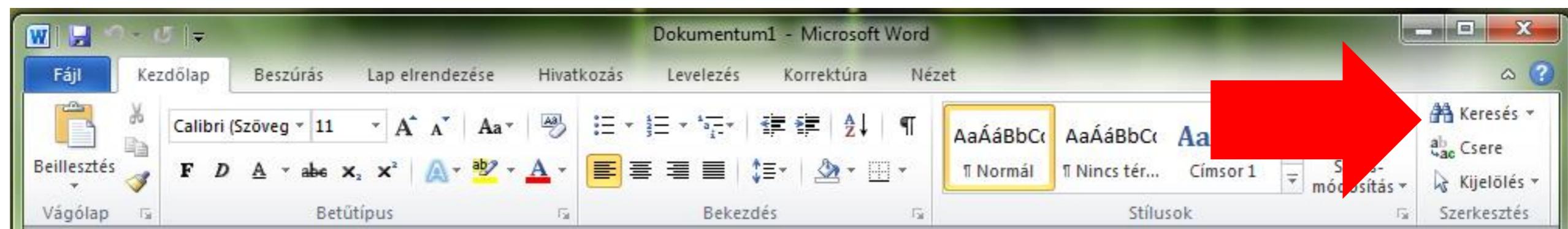
	ControlName	Total Clicks	Mouse	Short	Accel	Total
1	MSWordCharRight	35471659	0.0%	100.0%	0.0%	100.0%
2	MSWordCharLeft	26192205	0.0%	100.0%	0.0%	100.0%
3	WordClearContents	24876003	0.0%	100.0%	0.0%	100.0%
4	MSWordLineDown	22252751	0.0%	100.0%	0.0%	100.0%
5	MSWordLineUp	12744927	0.0%	100.0%	0.0%	100.0%
6	Paste	3255378	50.1%	49.8%	0.1%	100.0%
7	Save	2312797	79.4%	20.2%	0.4%	100.0%
8	Undo	2030009	70.1%	29.9%	0.0%	100.0%
9	File	2023537	98.0%	0.0%	2.0%	100.0%
10	MSWordCharRightExtend	2017326	0.0%	100.0%	0.0%	100.0%
11	PageDown	1983611	0.0%	100.0%	0.0%	100.0%
12	Copy	1740940	51.8%	48.2%	0.1%	100.0%
13	MSWordLineDownExtend	1631436	0.0%	100.0%	0.0%	100.0%
14	CloseWindow	1587094	98.9%	1.0%	0.0%	100.0%
15	MSWordCharLeftExtend	1413983	0.0%	100.0%	0.0%	100.0%
16	SendDefault	1255674	100.0%	0.0%	0.0%	100.0%
17	PageUp	1213169	0.0%	100.0%	0.0%	100.0%
18	Bold	1154736	82.5%	17.5%	0.0%	100.0%
19	FontSizeCombo	1088667	99.9%	0.1%	0.0%	100.0%
20	MSWordWordRight	1086383	0.0%	100.0%	0.0%	100.0%
21	MSWordWordLeft	820988	0.0%	100.0%	0.0%	100.0%
22	EndOfLine	816204	0.0%	100.0%	0.0%	100.0%
23	MSWordNextCell	757848	0.0%	100.0%	0.0%	100.0%
24	Print	706135	84.0%	15.6%	0.4%	100.0%
25	NoSpellingSuggestions	703628	100.0%	0.0%	0.0%	100.0%
26	Edit	586686	96.8%	0.0%	3.2%	100.0%
27	Cut	558827	68.9%	30.9%	0.2%	100.0%
28	Format	556778	99.2%	0.0%	0.8%	100.0%
29	PrintDefault	495869	100.0%	0.0%	0.0%	100.0%
30	SaveAs	491518	94.3%	5.0%	0.8%	100.0%
31	Insert	490117	97.8%	0.0%	2.2%	100.0%
32	FontCombo	442462	99.9%	0.1%	0.0%	100.0%
33	Open	372150	97.2%	2.7%	0.2%	100.0%
34	Tools	365240	99.1%	0.0%	0.9%	100.0%

# Videó: Gaze tracking

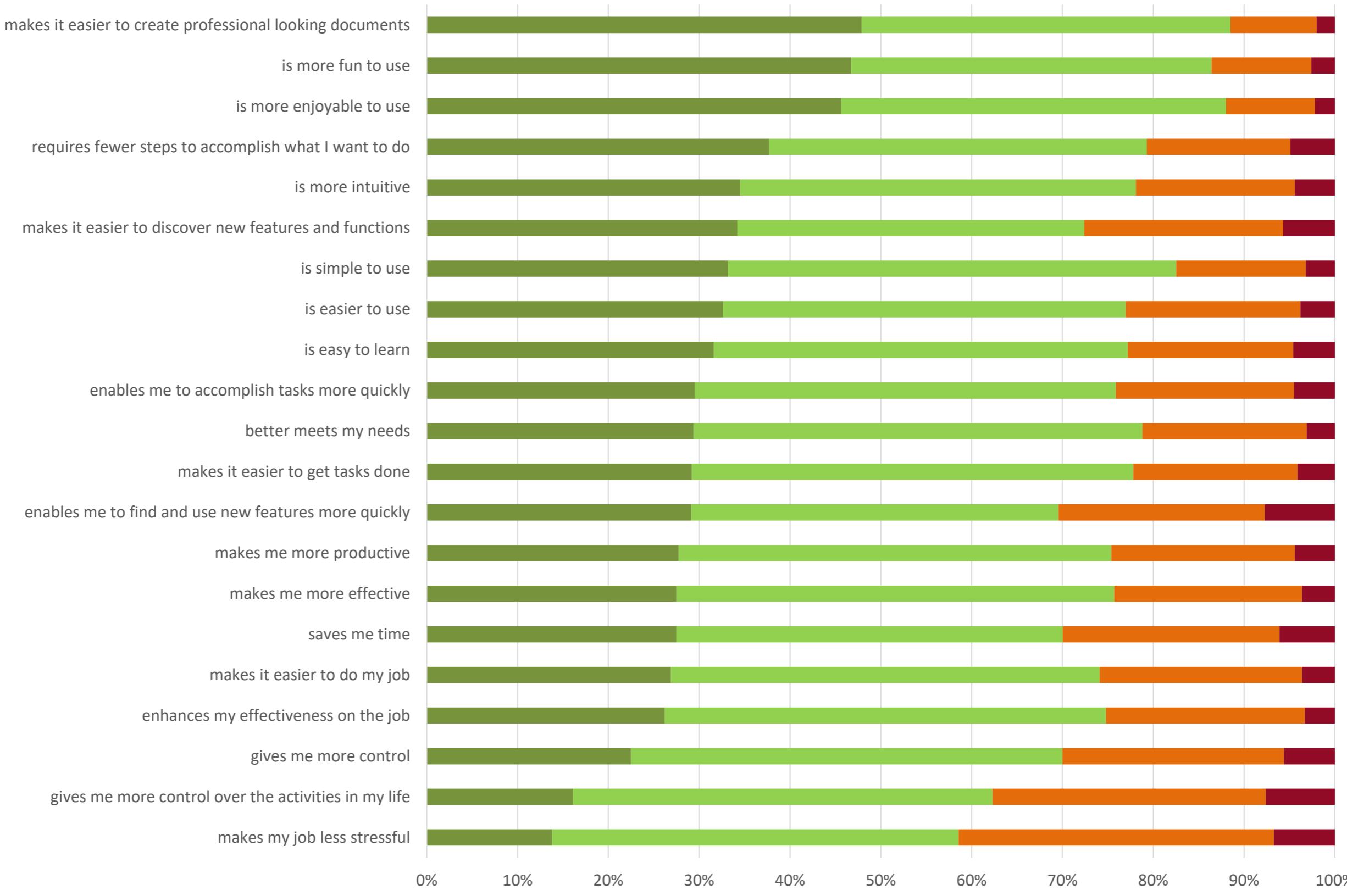


# Videó: Gaze tracking





# Az Office 2007 értékelése



# **ÉS MI A HELYZET A „FELHASZNÁLÓI HIBÁKKAL”?**

# Kezdetben vala...



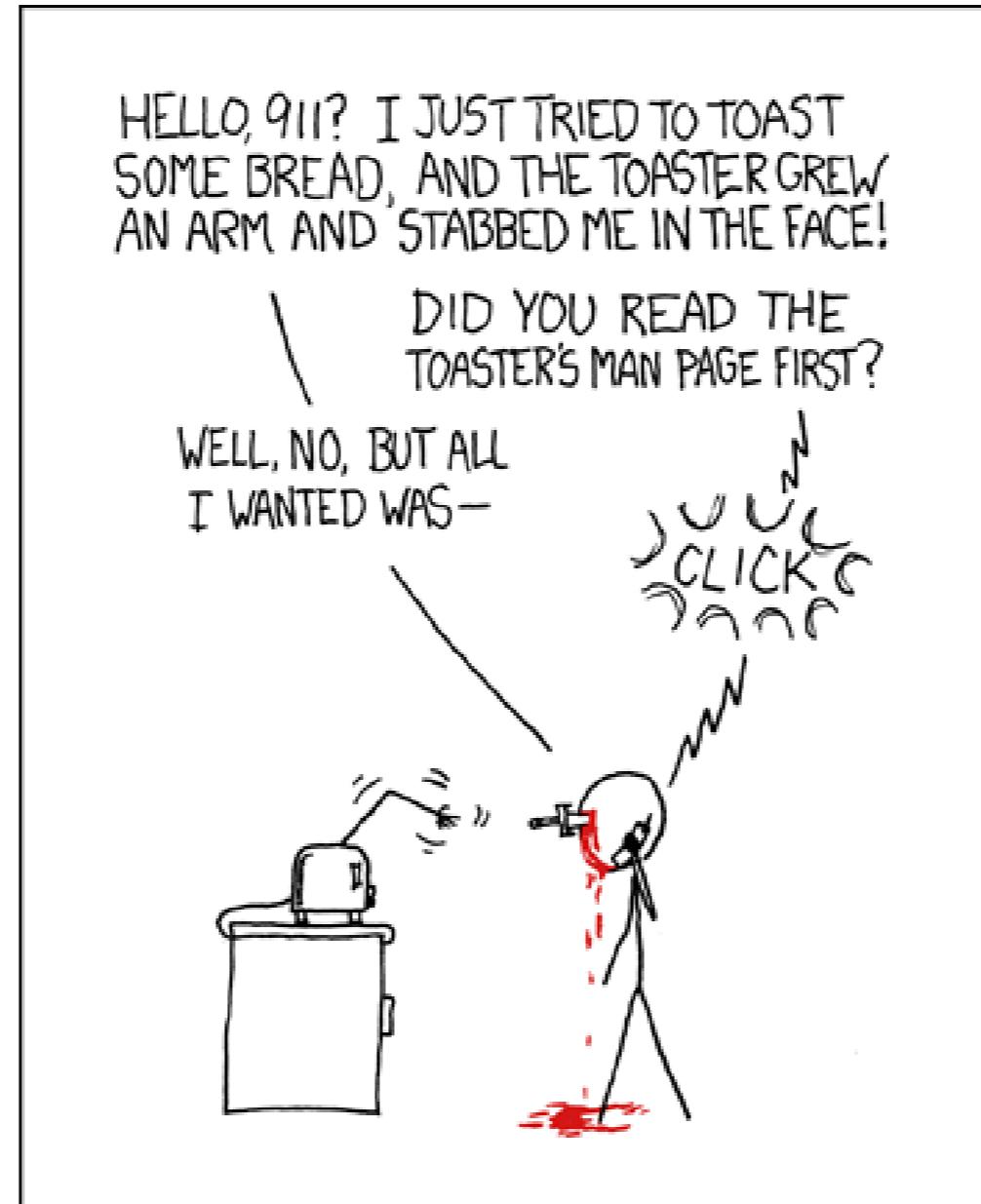
# Kezdetben vala...

- Drága gépek, alacsony teljesítmény
  - Olcsóbb a munkaerőt hosszasan betanítani, mint validációs logikát írni / futtatni
  - A grafikus felhasználói felület fel sem merül!
  - Az emberi hibák költségesek, de nem annyira, mint a gépidő...

# De Kinek a hibája az „emberi hiba”?

- Azért szoktuk emberi hibának hívni, mivel a problémát végső soron egy elmulasztott vagy rosszul véghezvitt emberi cselekedet okozta.
- Tudnia kellett volna, mit tegyen, hiszen:
  - Kapott képzést
  - Benne volt a használati útmutatóban
  - A dokumentáció mindenki számára elérhető az A12 épület C szárnyának 2-es pincéjében.

# RTFM



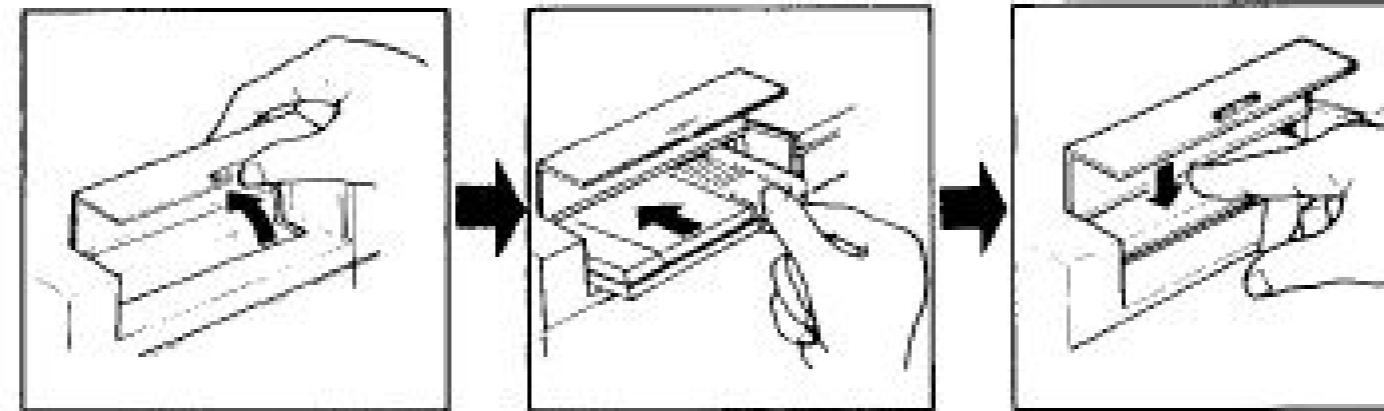
# Olvasnak-e a gyerekek használati útmutatót?

**CAUTION !! ALWAYS MAKE SURE THAT THE POWER SWITCH ON THE CONTROL DECK IS OFF BEFORE INSERTING OR REMOVING A GAME PAK !!**

4. Open the Chamber Lid on the Control Deck.

Insert a Game Pak into the Chamber (Label Facing up) and Push it all the way in.

Press Down on the Game Pak until it locks into place and close the Chamber Lid.



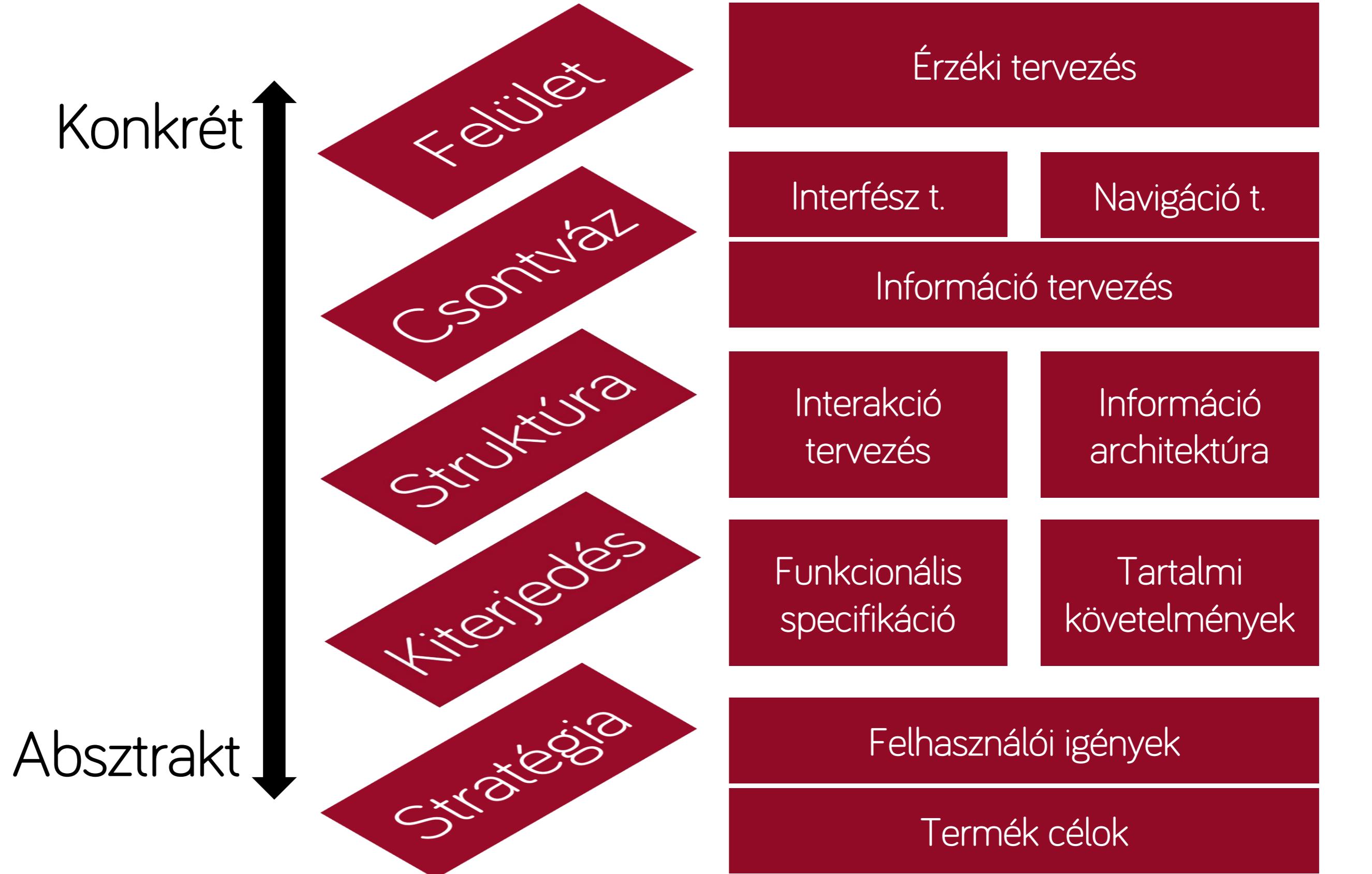
# Kinek a hibája az „emberi hiba”? (folyt.)

- Az emberi viselkedésnek és gondolkodásnak léteznek ismert, majdnem mindenkinél előforduló „hibái” – ez pontosan ugyanannyira adottság, mint az, hogy két kezünk van.
- Figyelmen kívül hagyni ezeket a tervezésnél olyan, mint autót gyártani három kezűeknek.

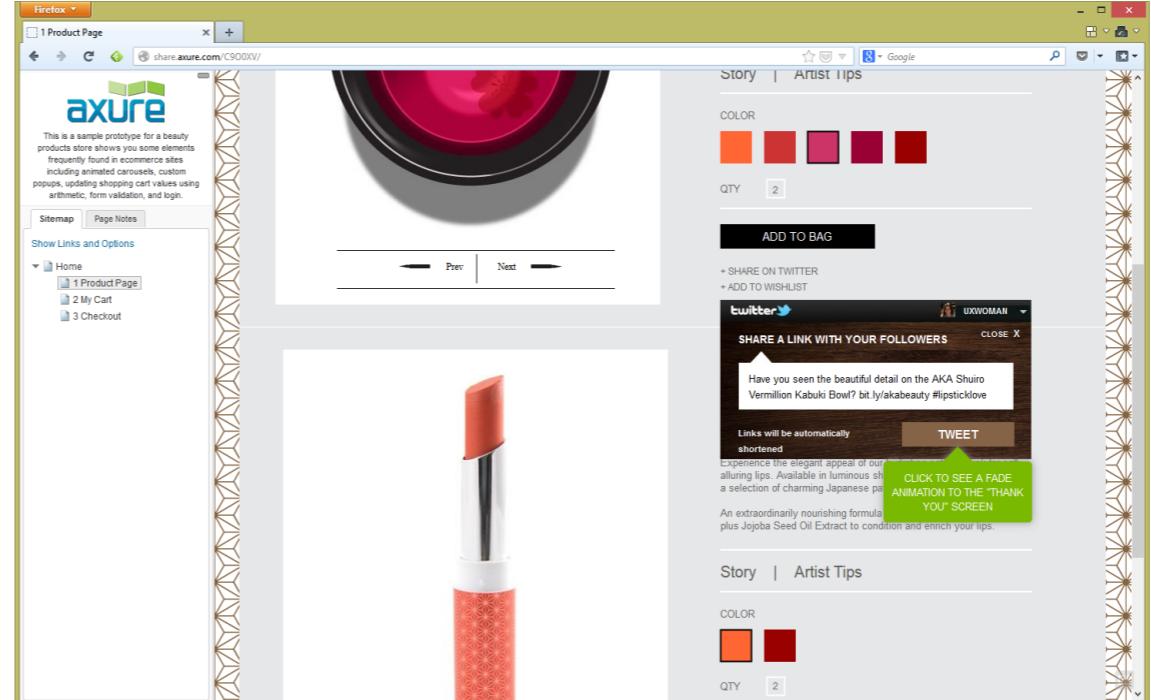
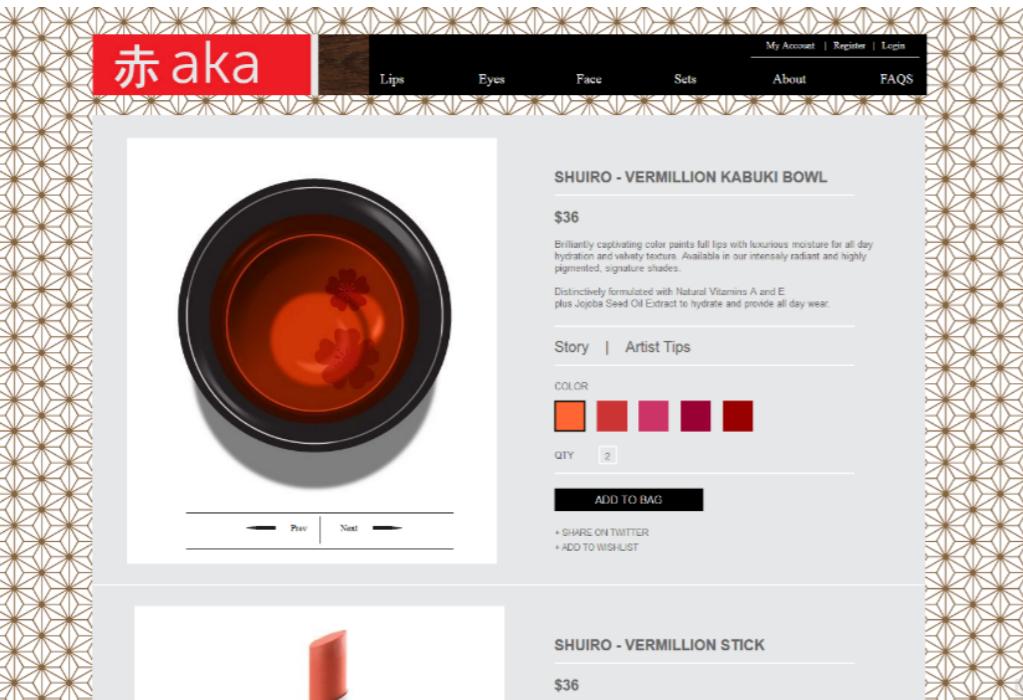
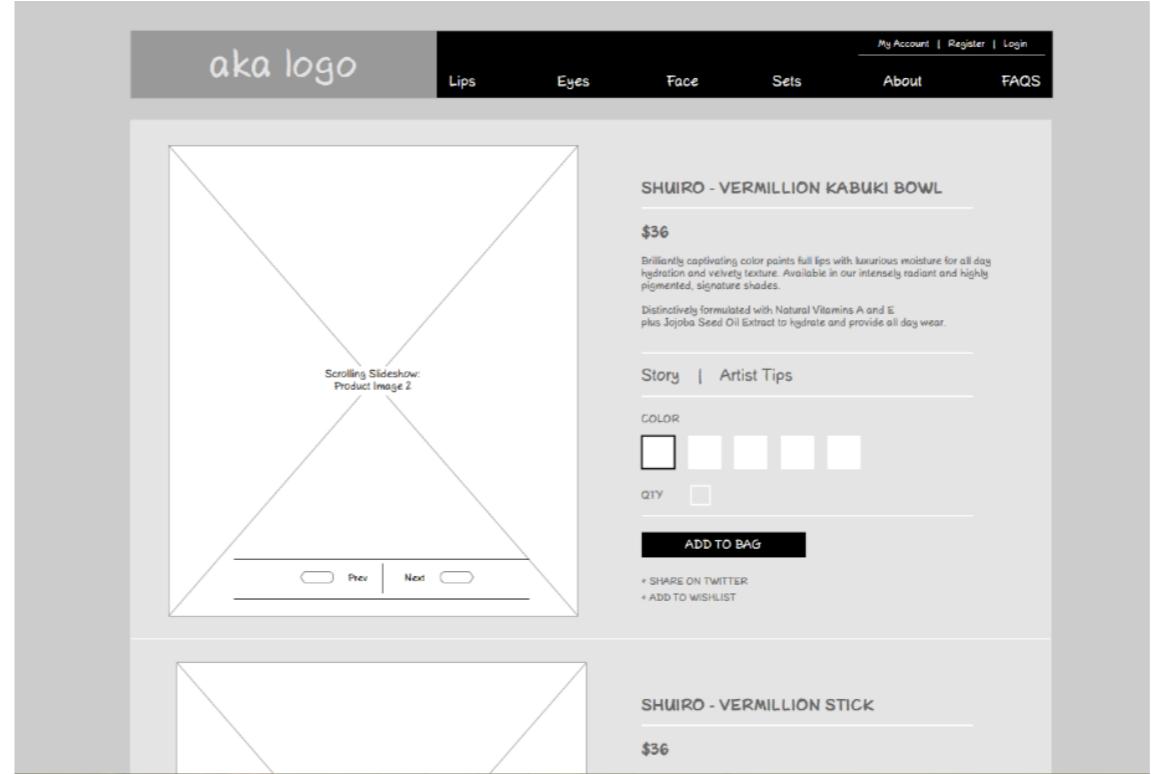
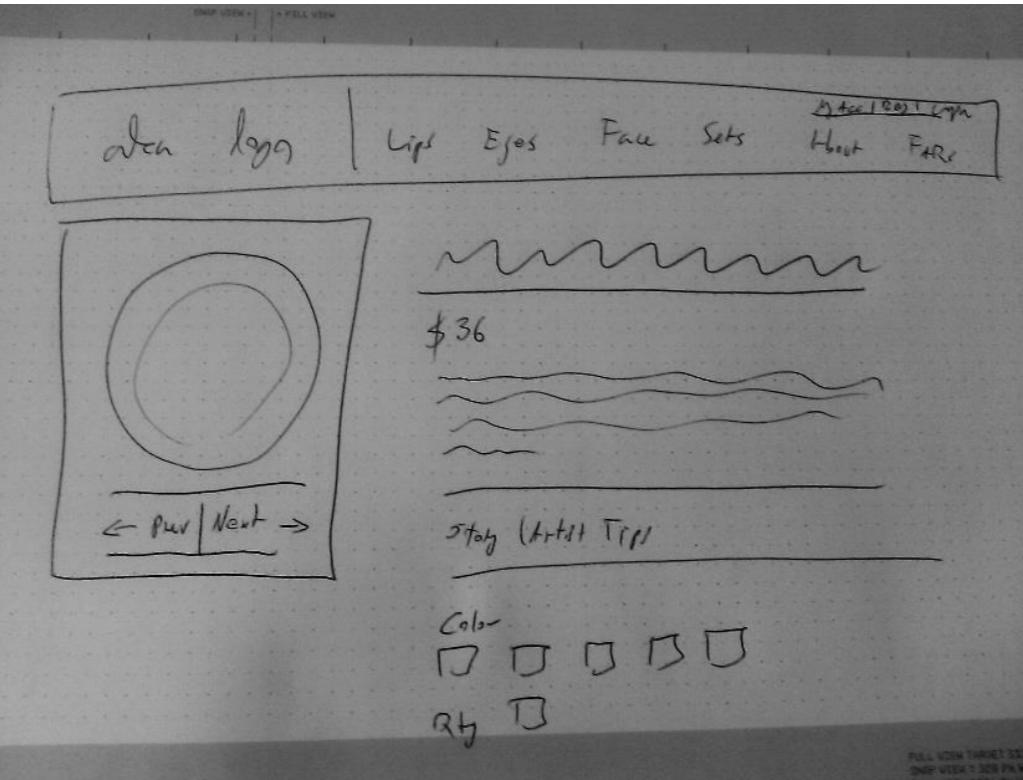
„A LEGTÖBB BALESETET  
EMBERI HIBÁNAK  
TULAJDONÍTJÁK, DE AZ  
EMBERI HIBA MAJDNEM  
MINDIG A ROSSZ TERVEZÉS  
KÖVETKEZMÉNYE.”

DON NORMAN: THE DESIGN OF EVERYDAY THINGS

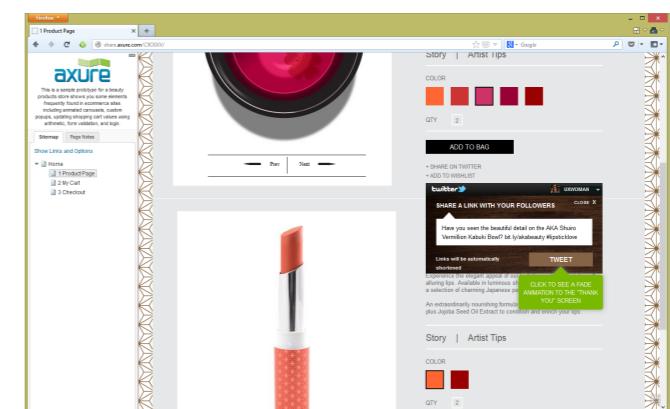
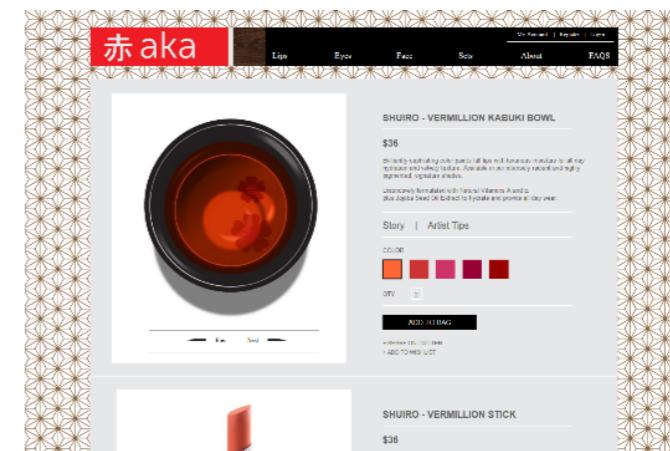
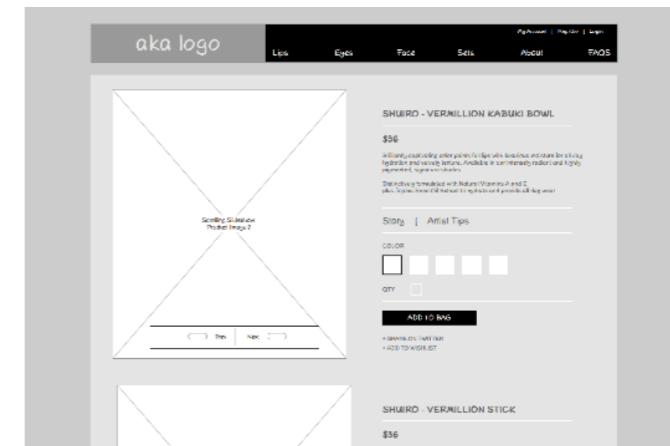
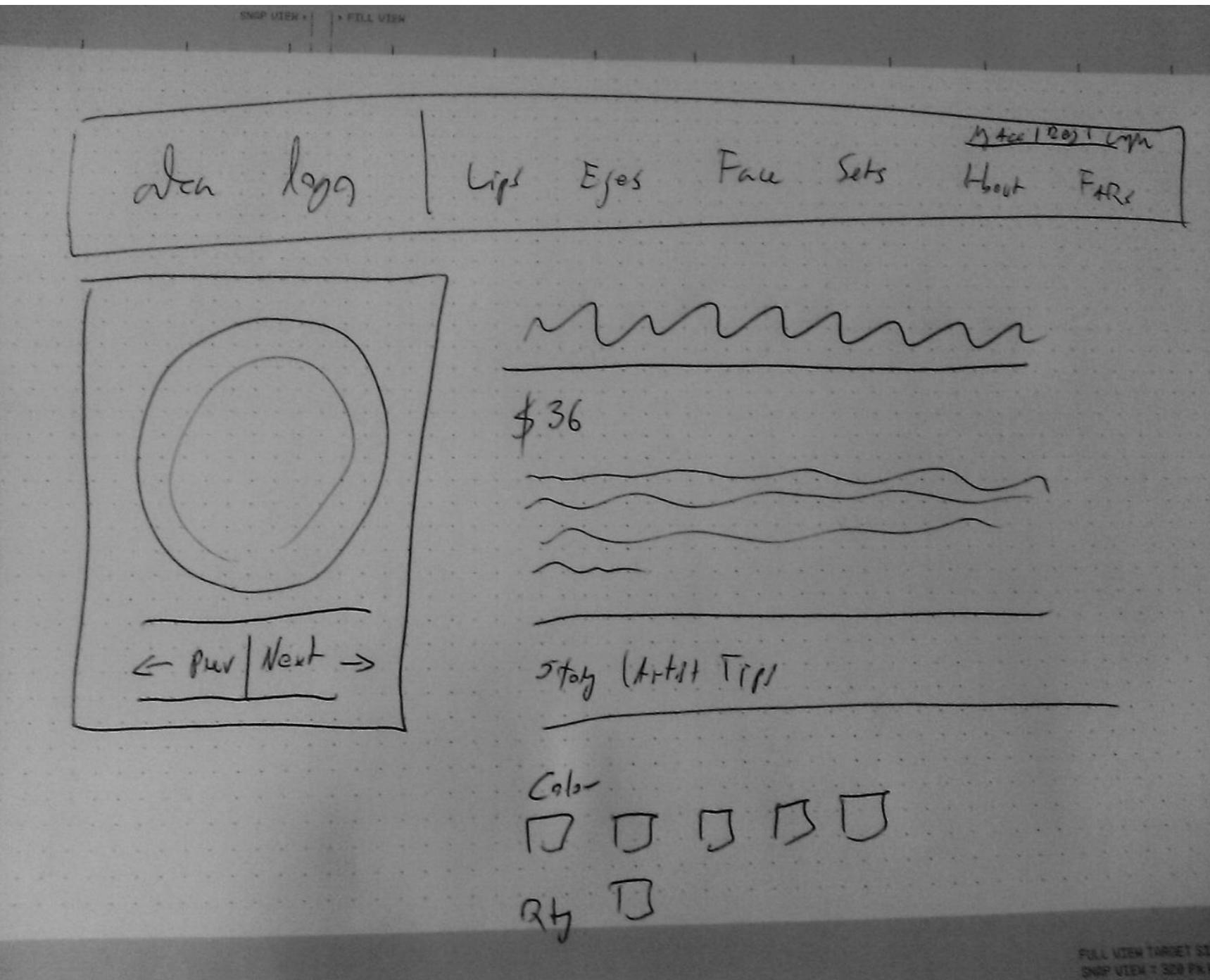
# HOGYAN?



# Egy képernyő megannyi formája...

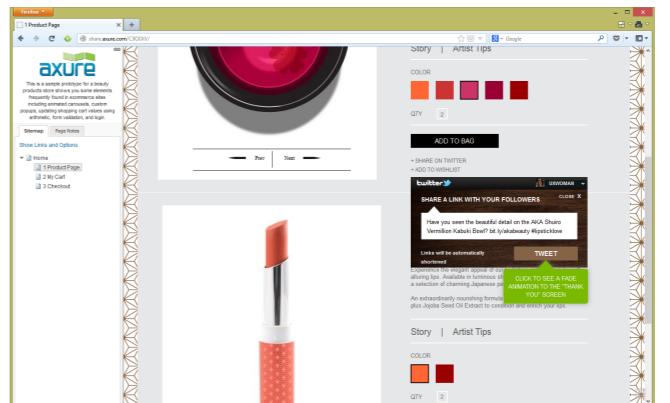
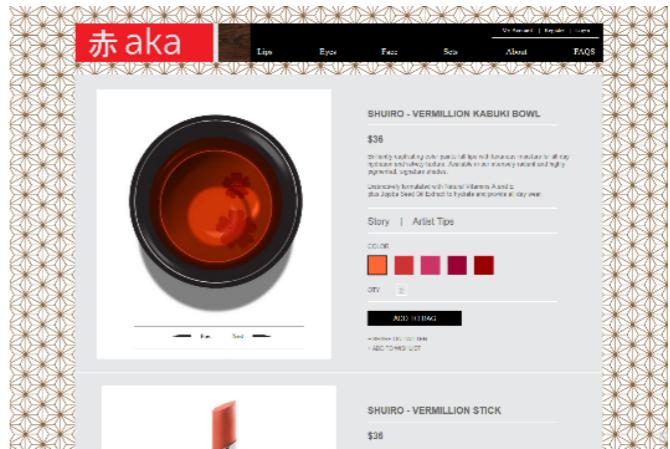
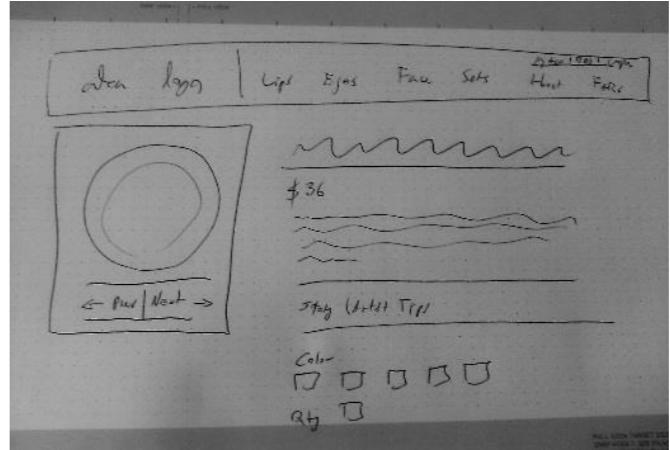


# Sketch: ötletelés



# Wireframe: Tervrajz

The wireframe shows a product page for the SHUIRO - VERMILLION KABUKI BOWL. At the top, there's a navigation bar with links for My Account, Register, Login, Lips, Eyes, Face, Sets, About, and FAQS. Below the navigation is a large triangular placeholder for a product image, with the text "Scrolling Slideshow: Product Image 2" inside it. To the right of the placeholder is the product title "SHUIRO - VERMILLION KABUKI BOWL" and its price "\$36". A detailed description follows, mentioning "Brilliantly capturing color paints full lips with luxurious moisture for all day hydration and velvety texture. Available in our intensely radiant and highly pigmented, signature shades." It also notes "Distinctively formulated with Natural Vitamins A and E plus Jojoba Seed Oil Extract to hydrate and provide all day wear." Below the description are sections for "Story" and "Artist Tips". Further down are color swatches, quantity selection, and an "ADD TO BAG" button. At the bottom of the page is another product section for the "SHUIRO - VERMILLION STICK" at \$36.



# Vizuális terv: minden a helyén

My Account | Register | Login

Lips Eyes Face Sets About FAQS

SHUIRO - VERMILLION KABUKI BOWL

\$36

Brilliantly captivating color paints full lips with luxurious moisture for all day hydration and velvety texture. Available in our intensely radiant and highly pigmented, signature shades.

Distinctively formulated with Natural Vitamins A and E plus Jojoba Seed Oil Extract to hydrate and provide all day wear.

Story | Artist Tips

COLOR

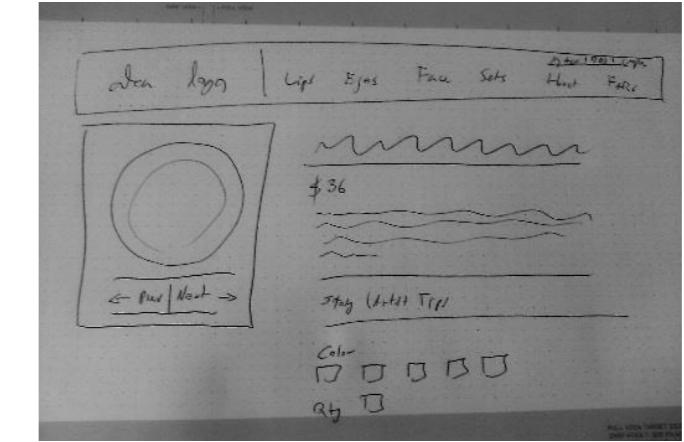
QTY 2

ADD TO BAG

+ SHARE ON TWITTER + ADD TO WISHLIST

SHUIRO - VERMILLION STICK

\$36



aka logo Lips Eyes Face Sets About FAQS

SHUIRO - VERMILLION KABUKI BOWL

\$36

Brilliantly captivating color paints full lips with luxurious moisture for all day hydration and velvety texture. Available in our intensely radiant and highly pigmented, signature shades.

Distinctively formulated with Natural Vitamins A and E plus Jojoba Seed Oil Extract to hydrate and provide all day wear.

Story | Artist Tips

Color

QTY

ADD TO BAG

+ SHARE ON TWITTER + ADD TO WISHLIST

SHUIRO - VERMILLION STICK

\$36

axure

This is a sample prototype for a beauty e-commerce website. It's a great example of how frequently used e-commerce sites are designed. It's a clean, modern design with a classic, elegant feel.

Share Links and Options

Header

Product Page

Prev | Next

SHUIRO - VERMILLION KABUKI BOWL

\$36

Brilliantly capturing color paints full lips with luxurious moisture for all day hydration and velvety texture. Available in our intensely radiant and highly pigmented, signature shades.

Distinctively formulated with Natural Vitamins A and E plus Jojoba Seed Oil Extract to hydrate and provide all day wear.

Story | Artist Tips

Color

QTY

ADD TO BAG

+ SHARE ON TWITTER + ADD TO WISHLIST

SHARE A LINK WITH YOUR FOLLOWERS

Links will be automatically generated for each page. You can copy and paste them into your social media profiles or email newsletters.

TWEET

Have you seen the beautiful detail on the AKA Share Vermillion Kabuki Bowl? It's absolutely delicious!

Links will be automatically generated for each page. You can copy and paste them into your social media profiles or email newsletters.

CLOSE TO SEE A PREVIEW

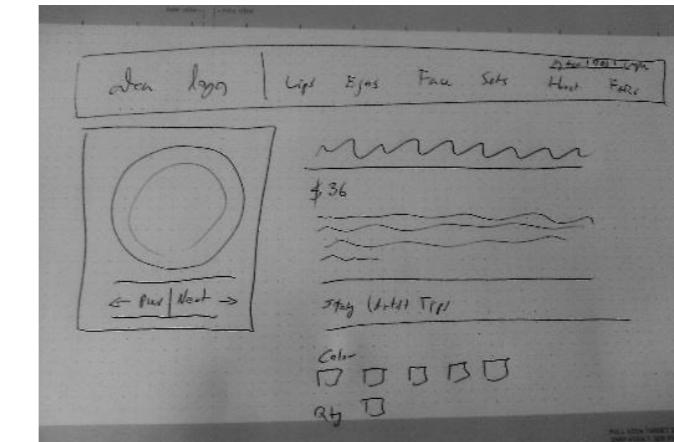
OPEN TO THE THANK YOU PAGE

SHUIRO - VERMILLION STICK

\$36

# Prototípus: Bármelyik eddigi + interakció

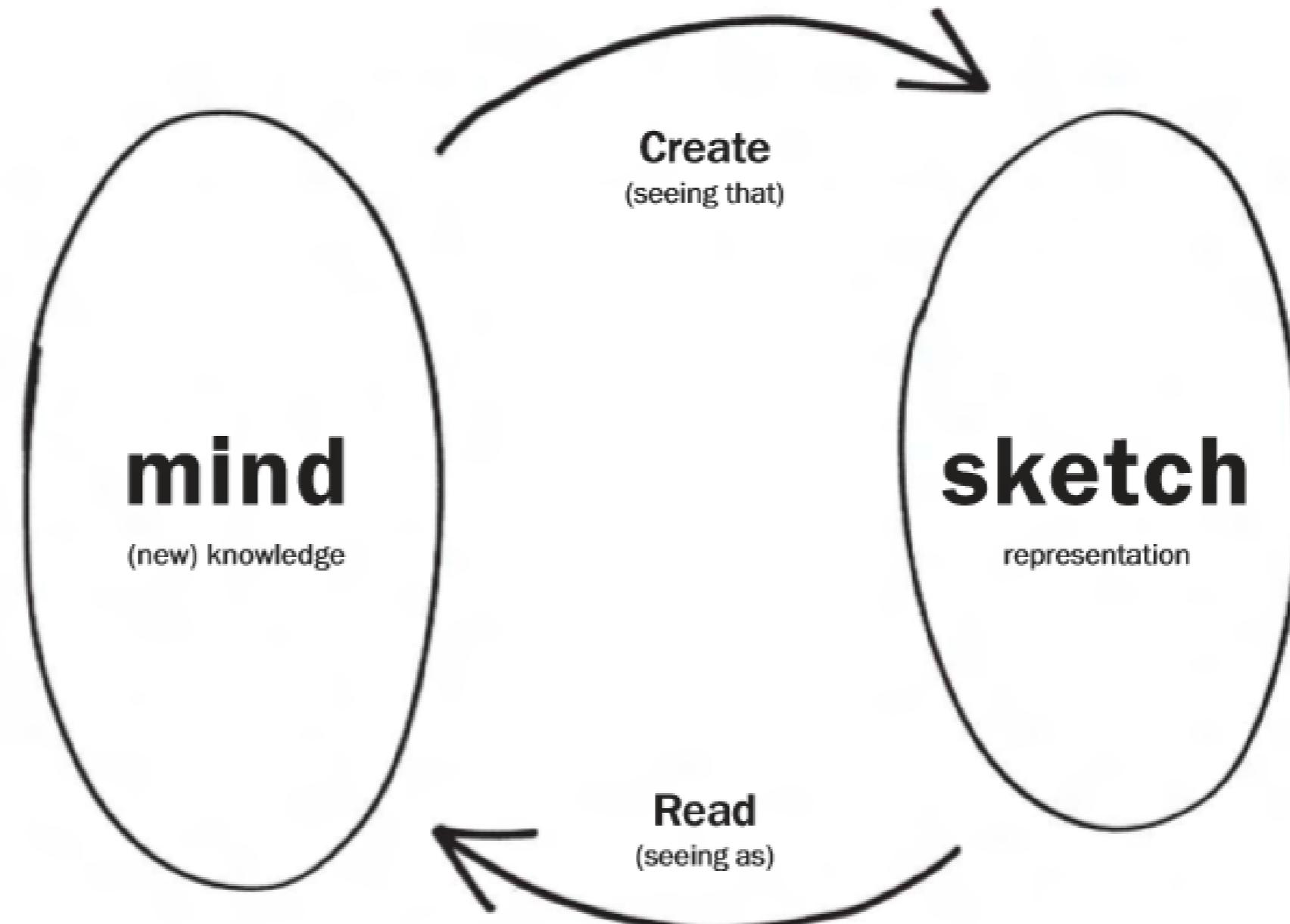
The screenshot shows a product page for a cosmetic item. At the top, there's a large image of a red bowl with a floral design. Below it is a smaller image of a red lipstick tube. The main content area includes a color palette with five swatches (orange, red-orange, pink, dark red, black), a quantity selector set to 2, and a large "ADD TO BAG" button. A "Story | Artist Tips" section follows. A Twitter sharing modal is overlaid on the page, prompting users to share a link with their followers. The URL provided is [#lipsticklove](http://bit.ly/akabeauty). The modal also features a "TWEET" button and a green call-to-action button that says "CLICK TO SEE A FADE ANIMATION TO THE 'THANK YOU' SCREEN". The bottom of the page has another "Story | Artist Tips" section and a "COLOR" section with two swatches.



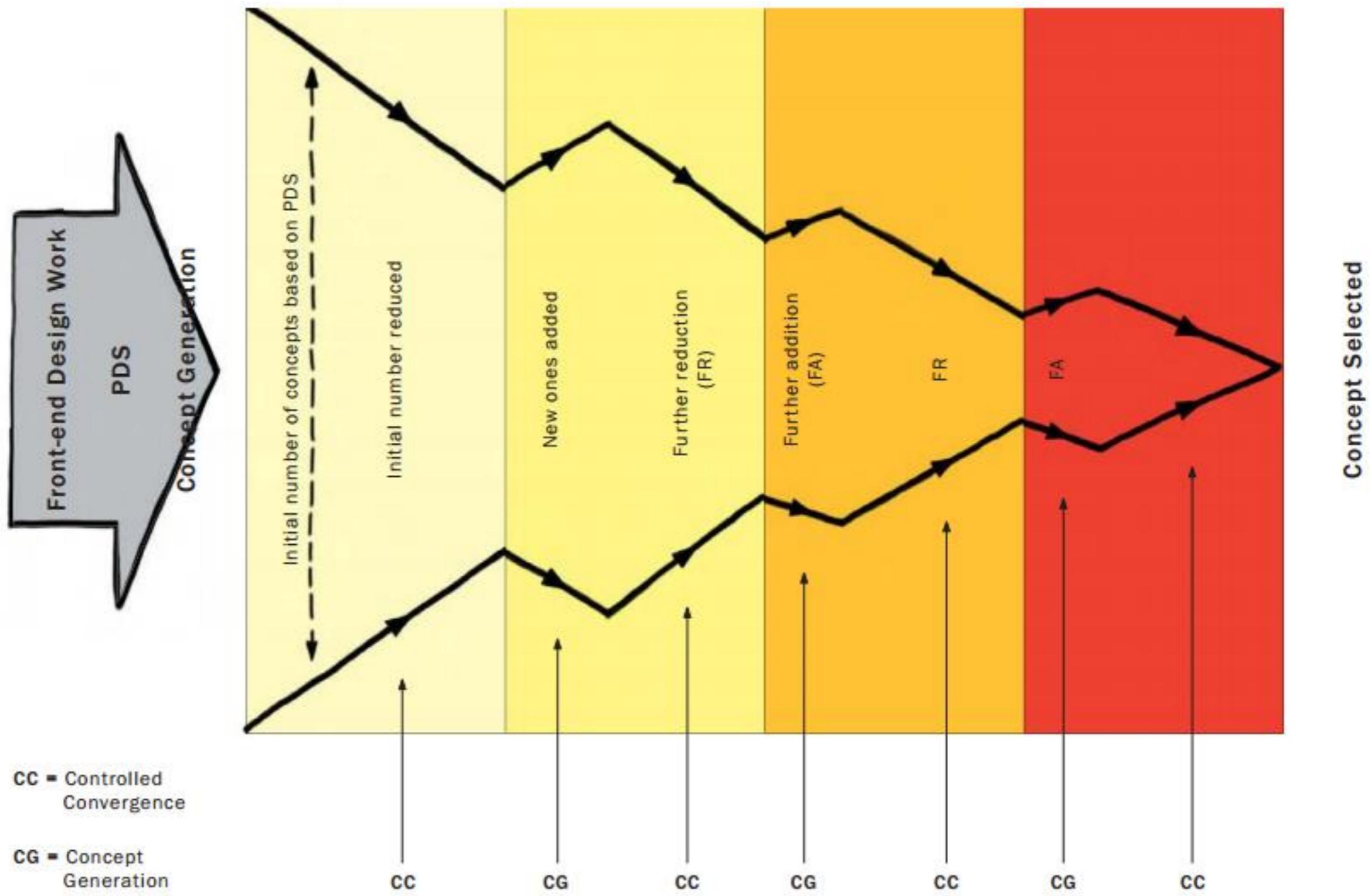
This screenshot shows a mobile-optimized version of the product page. The layout is similar to the desktop version but adapted for a smaller screen. It includes a large product image at the top, followed by a color palette, quantity selector, and "ADD TO BAG" button. Below these are sections for "Story | Artist Tips" and a "Twitter" sharing modal. The "SHIRO - VERNILLION KABUKI BOWL" product details are visible on the right.

This screenshot shows the same mobile product page as above, but with a decorative gold patterned border on the left and right sides. The product image, color palette, and "ADD TO BAG" button are prominent. The "SHIRO - VERNILLION KABUKI BOWL" product details are visible on the right.

# A folyamat konrántsem lineáris



# A folyamat konrántsem lineáris



# A PRECÍZ PIXELEK

**„A DIZÁJN DIADALA:  
A BONYOLULT EGYSZERŰNEK TŰNIK”**

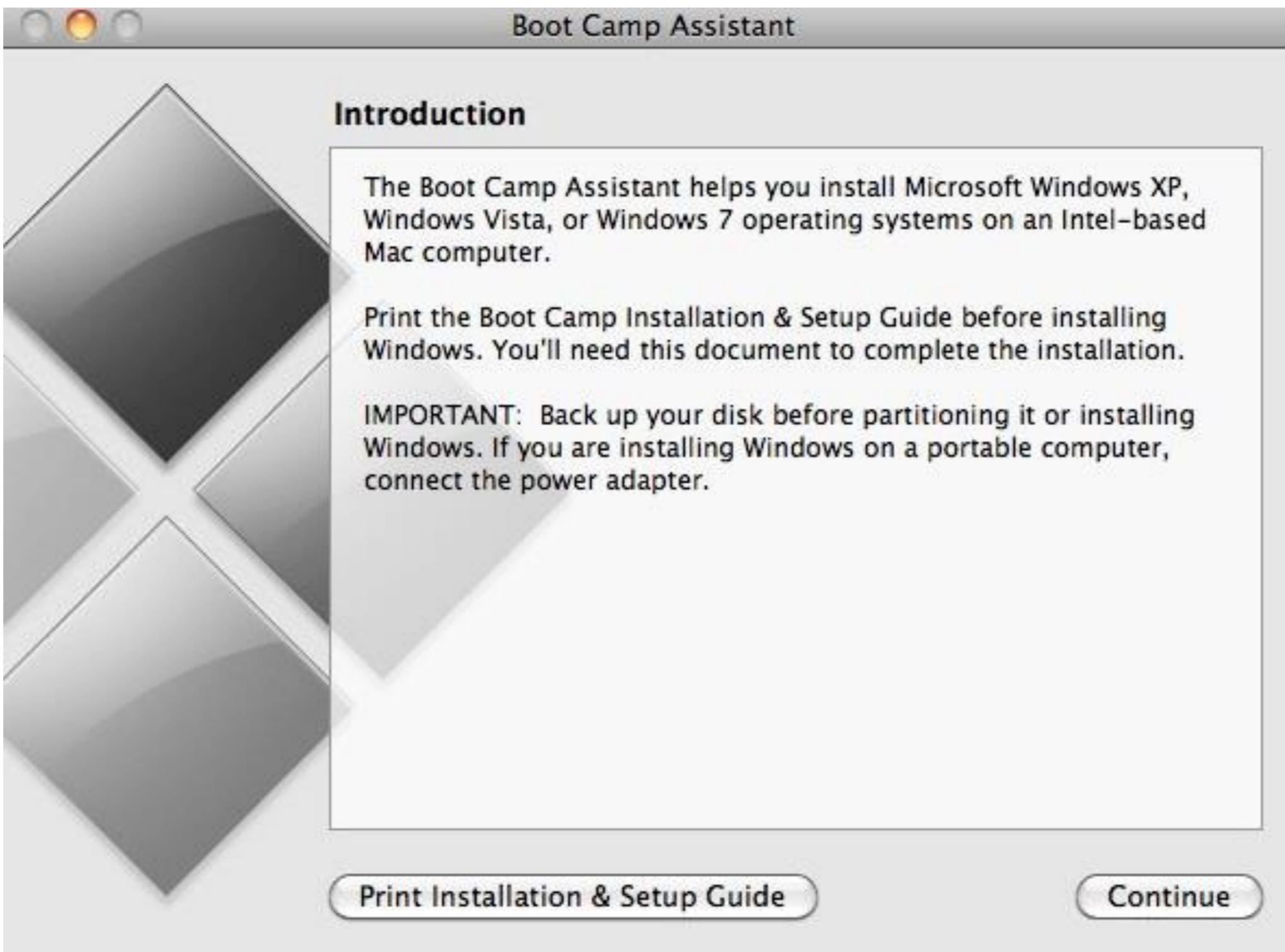
ROBBY INGEBRETSEN

# „A DIZÁJN DIADALA: A BONYOLULT EGYSZERŰNEK TŰNIK”

ROBBY INGEBRETSEN

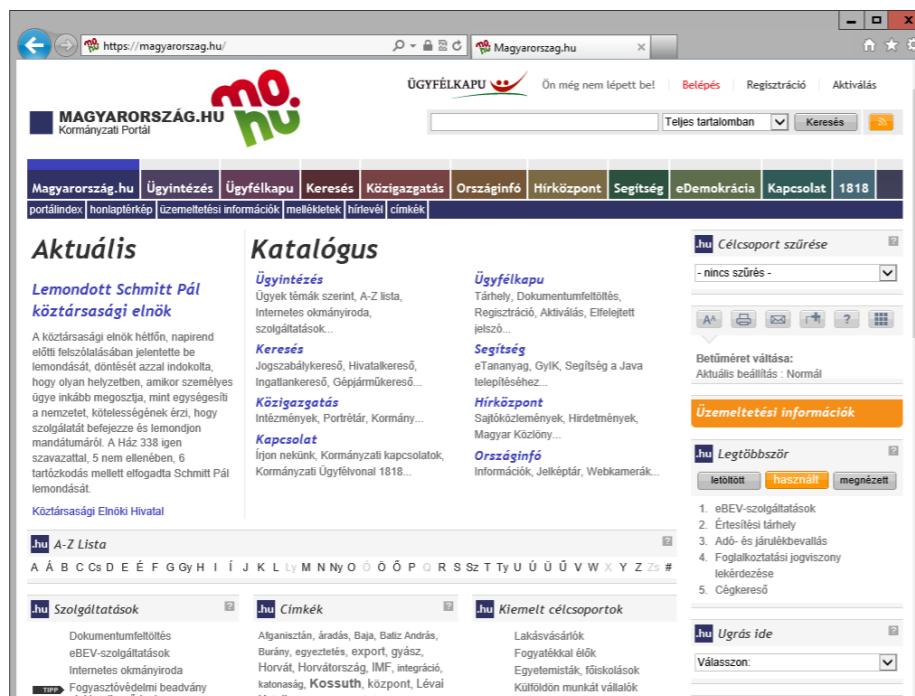
- Nem az a cél, hogy egyszerű legyen
  - Az a cél, hogy egyszerűnek érződjön
- Nem az a baj, ha sok funkció van
  - Az a baj, ha belefulladunk a funkciókba

# Egyszerűsítés ≠ Butítás

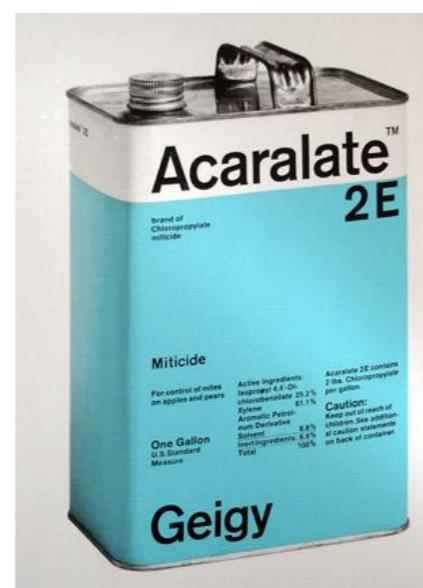
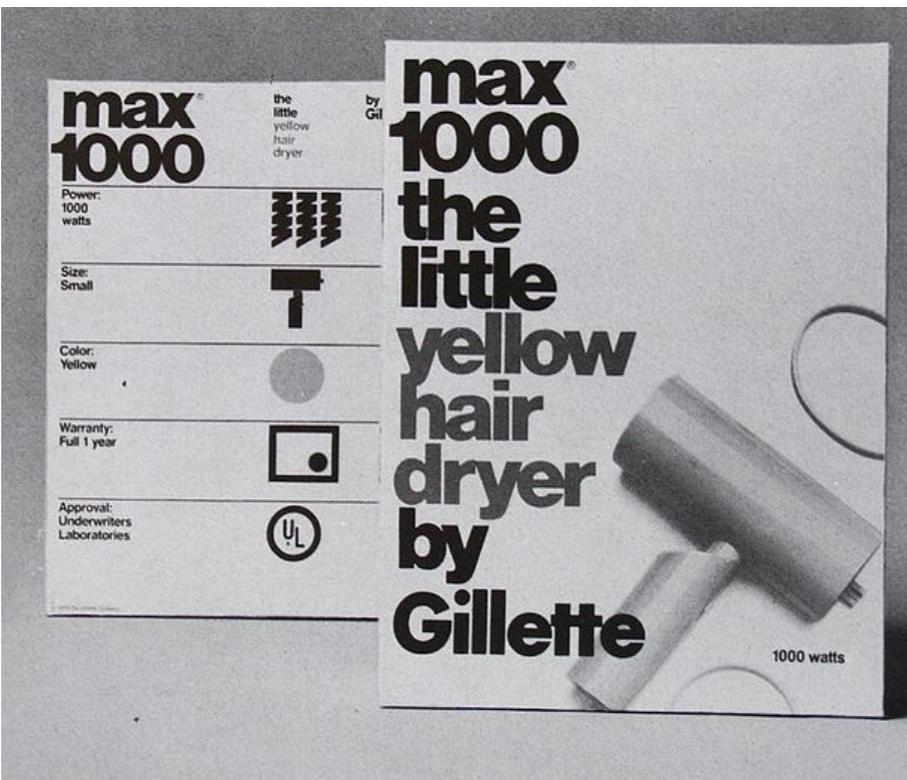


# Kapcsolatok kifejezése

- Önálló elemek helyett csoportokat „kezelünk”
  - Ha sok funkcióink van is, kevés csoportunk még lehet
  - Egyszerűbbnek tűnik, mint ömlesztve



# Átfuthatóság és Hierarchia



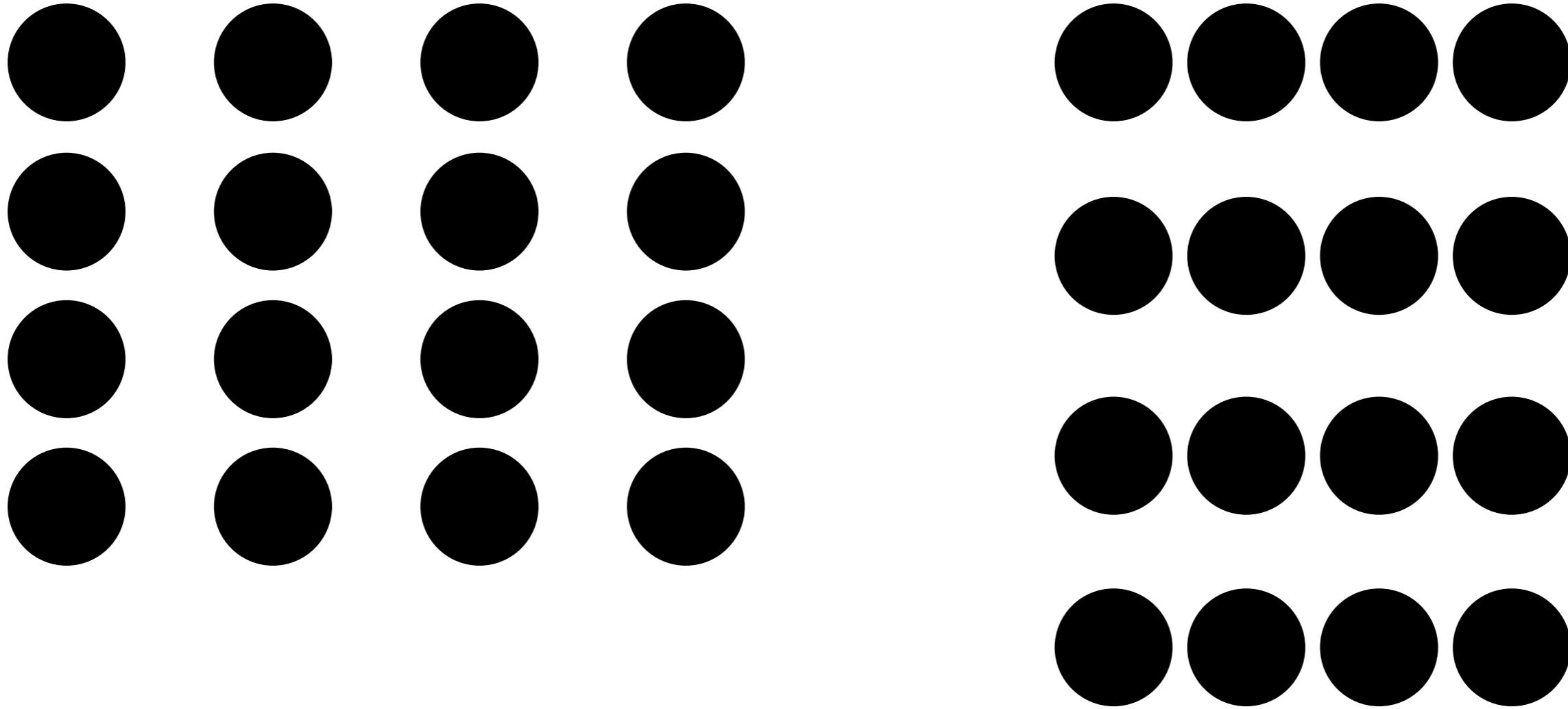
# ELRENDEZÉS

# Gestalt törvények

- Gestalt: valami, amit egyetlen objektumnak / egységnak érzékelünk
  - Szó szerint „formát” jelent (németül)
- A gestalt törvények mondják meg, miket látunk koherens egységek
  - Előképzés vagy tudatos erőfeszítés nélkül
- A felületelemek elrendezésekor építünk rájuk

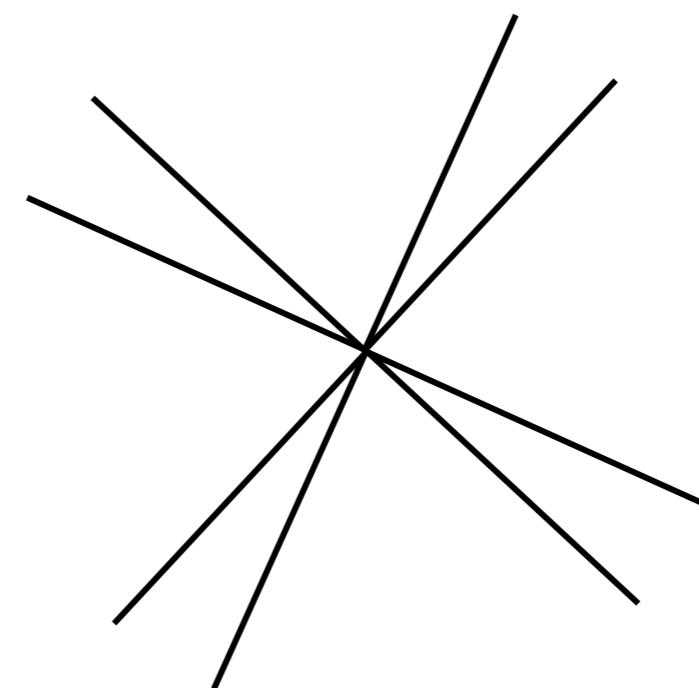
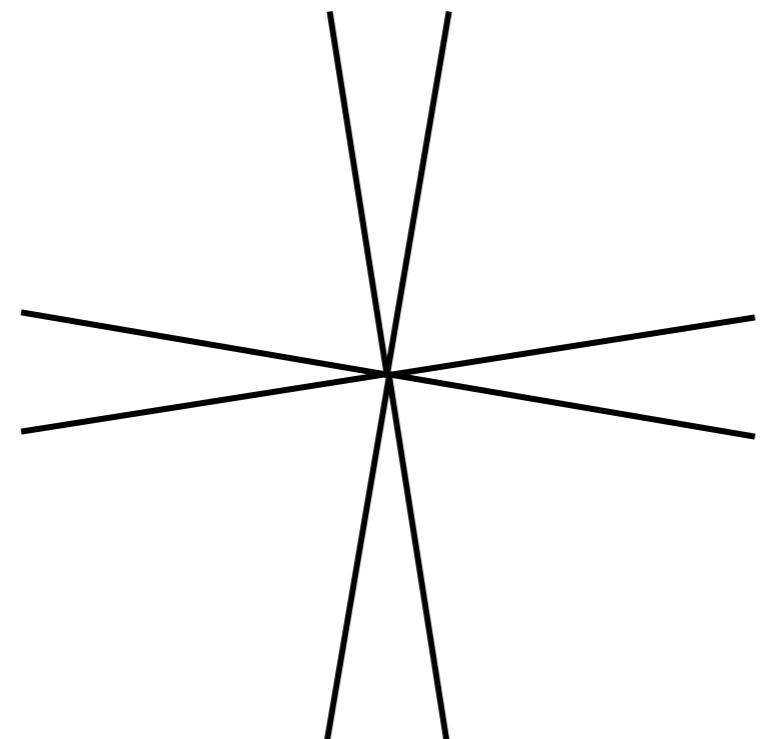
# A közelség törvénye

- Az egymáshoz közelálló elemeket összetartozónak érzékeljük



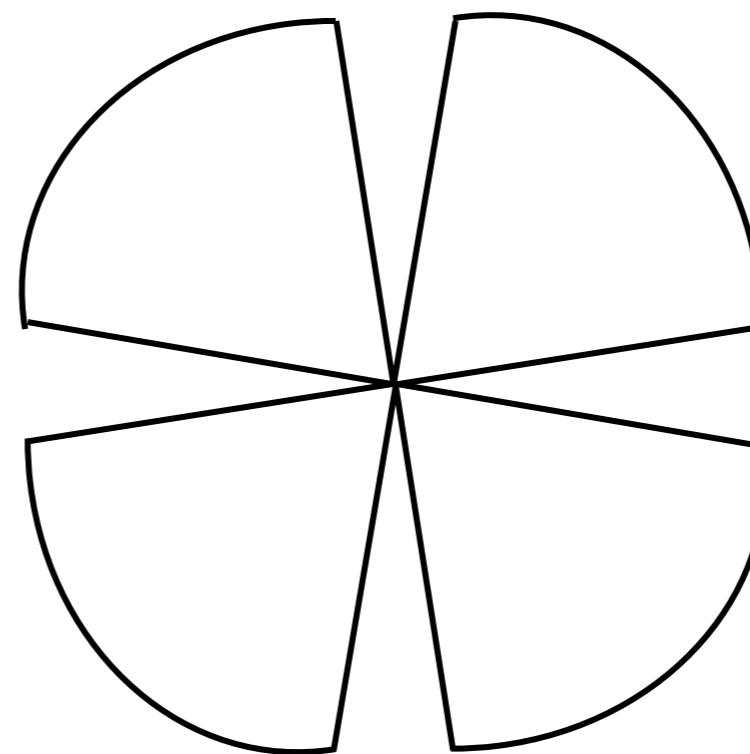
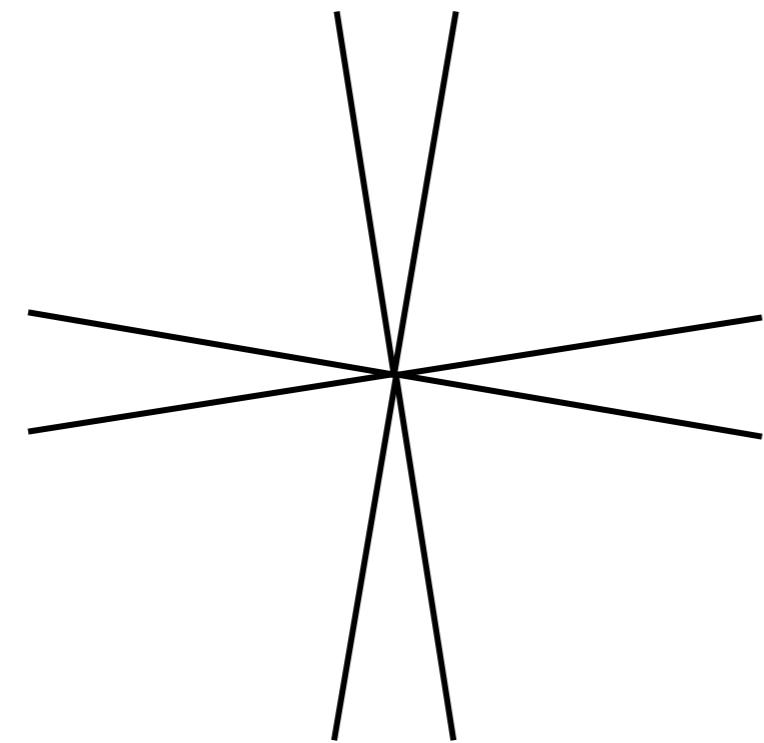
# A közelség törvénye

- Az egymáshoz közelálló elemeket összetartozónak érzékeljük



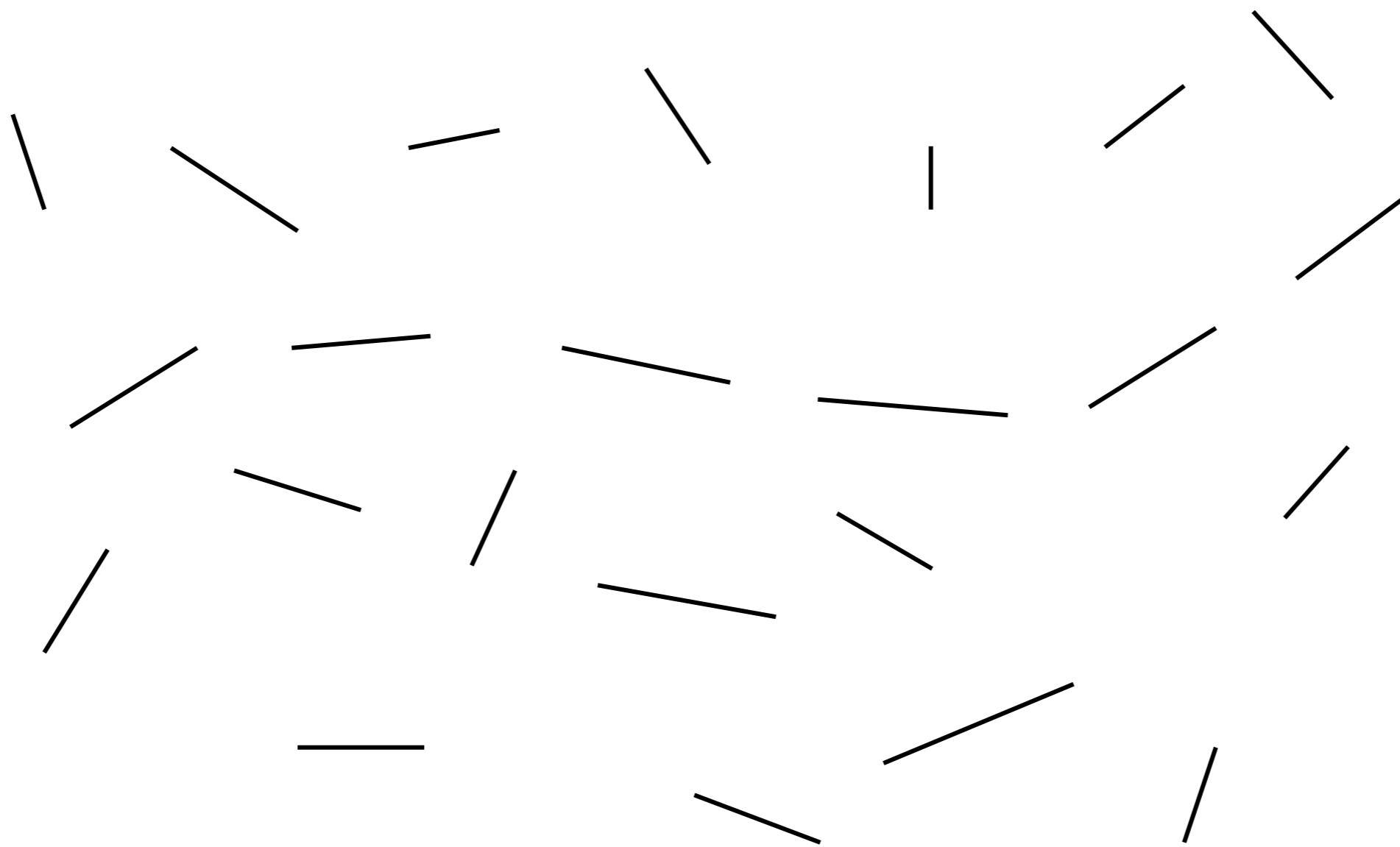
# A Lezárás törvénye

- A vonallal körbezárt területet egy alakzatnak érzékeljük
- Erősebb, mint a közelség törvénye



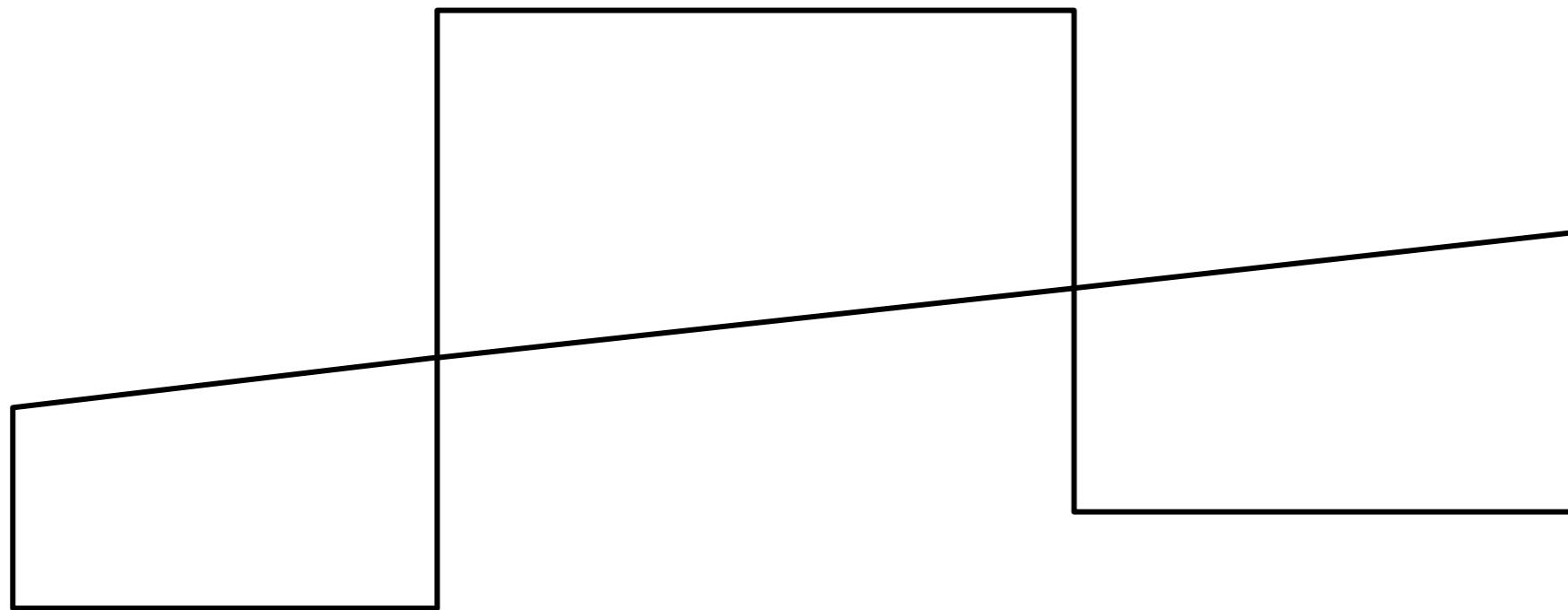
# A helyes folytatás törvénye

- Az egy vonalra eső elemeket összetartozónak érzékeljük



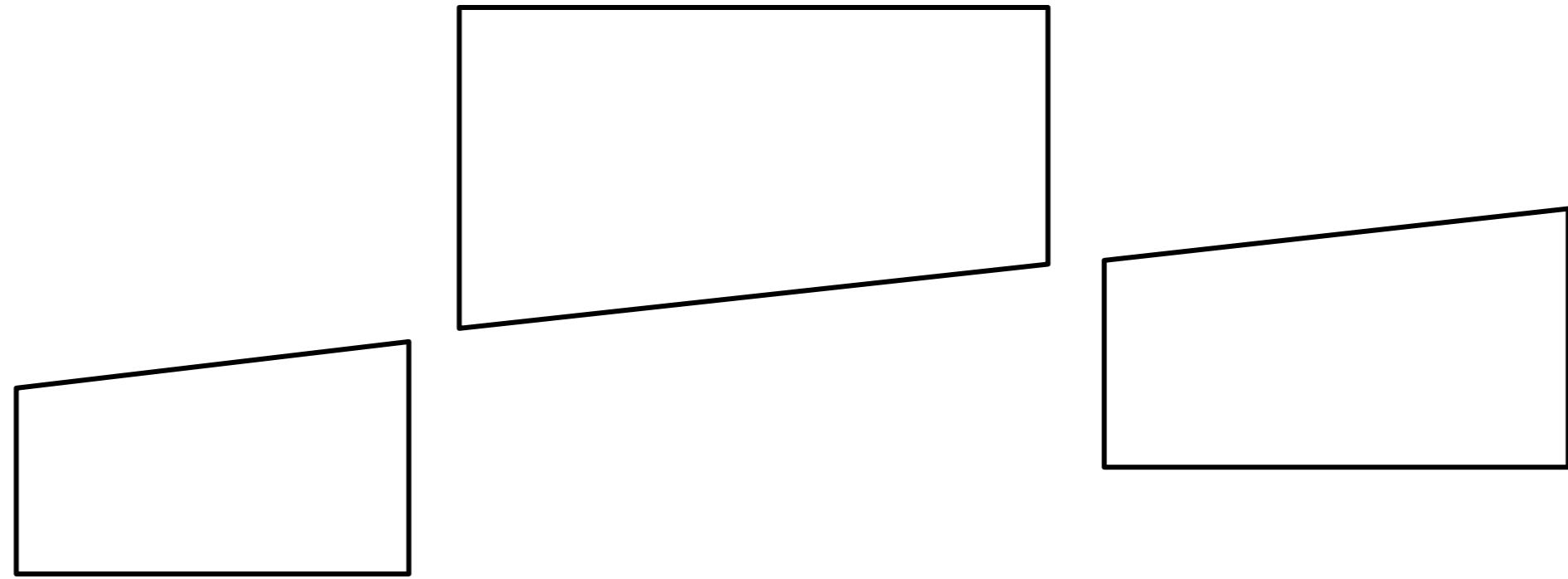
# Lezárás kontra folytatás

- Hosszú egyenes vonal vagy három trapéz?



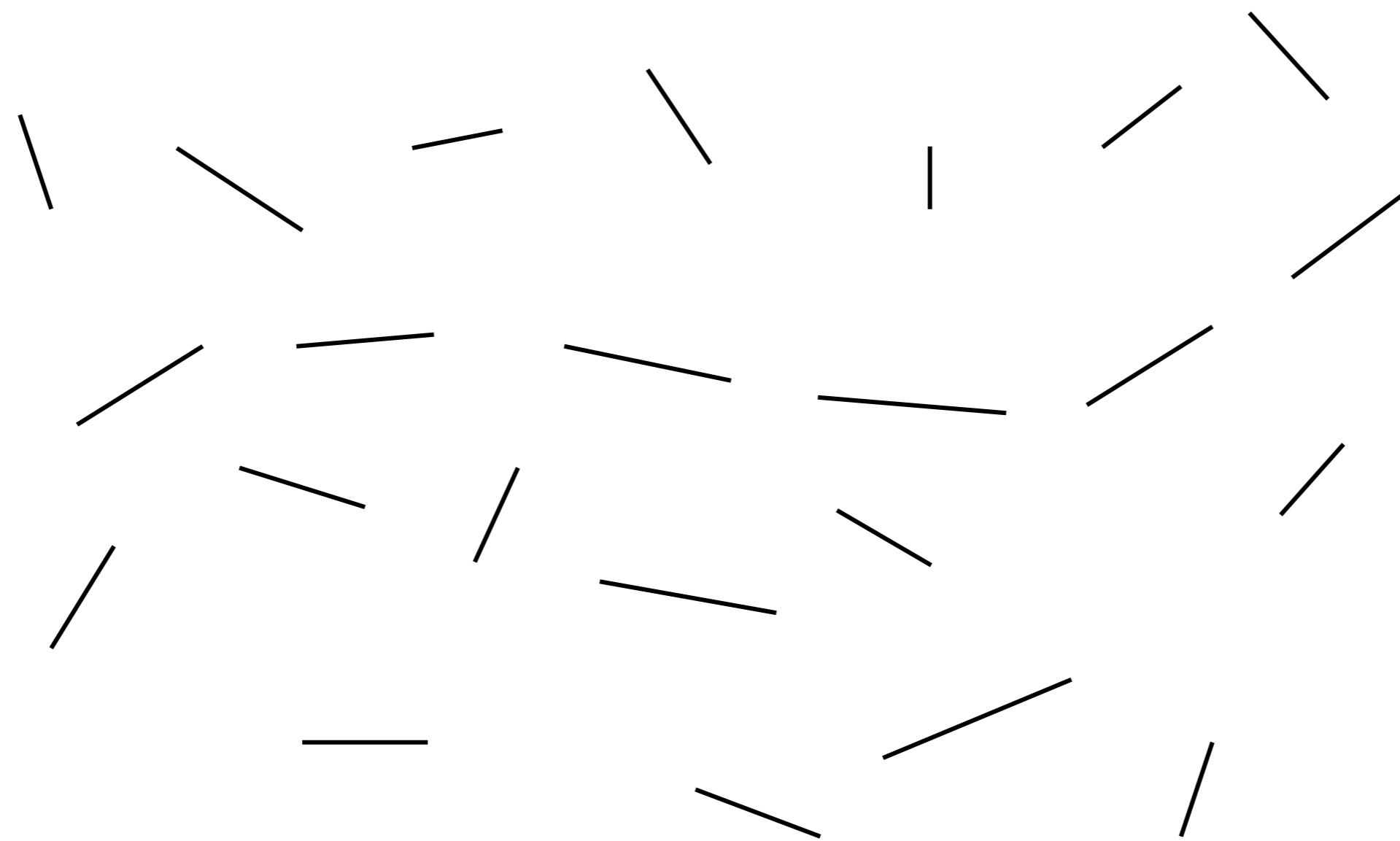
# Lezárás konrta folytatás

- Hosszú egyenes vonal vagy három trapéz?



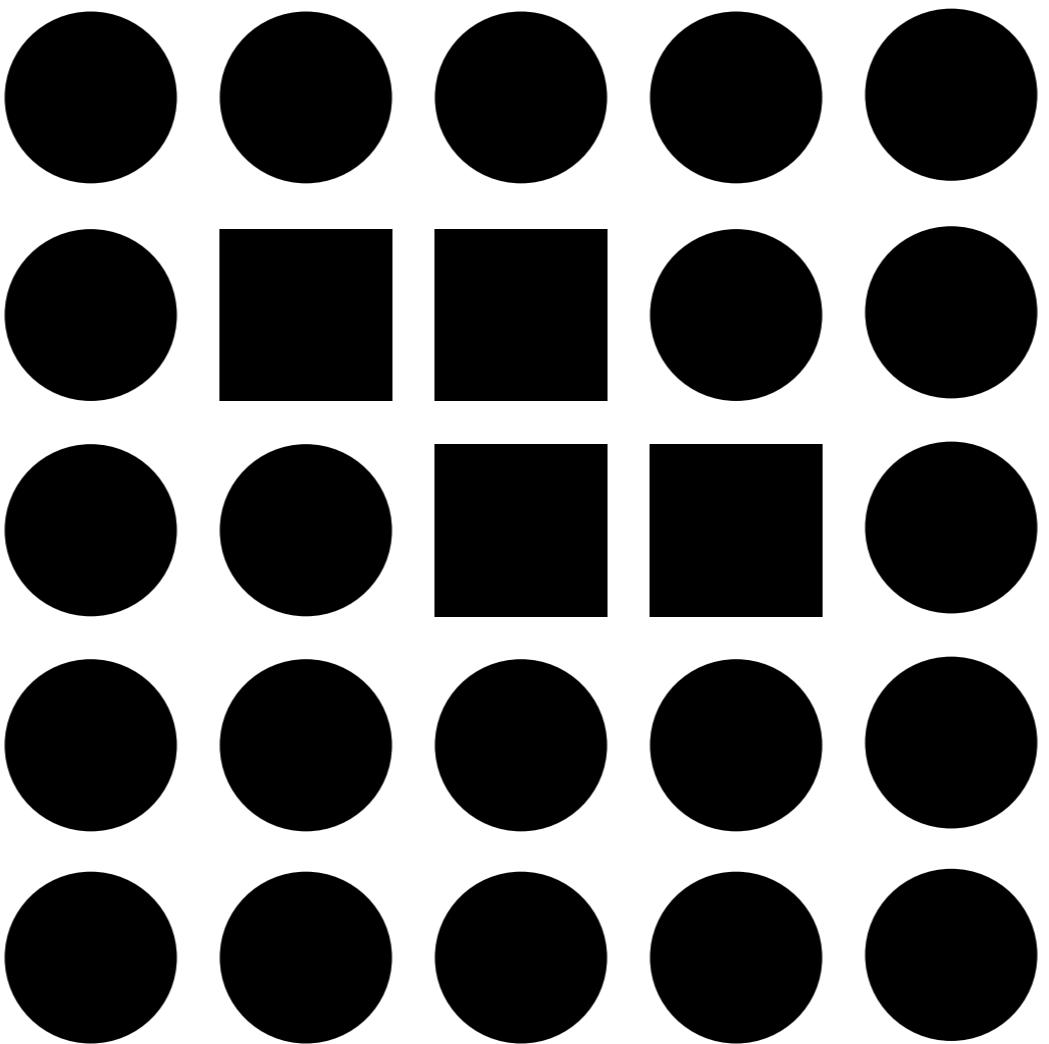
# A párhuzamos mozgás törvénye

- Az együtt mozgó elemeket összetartozónak érzékeljük



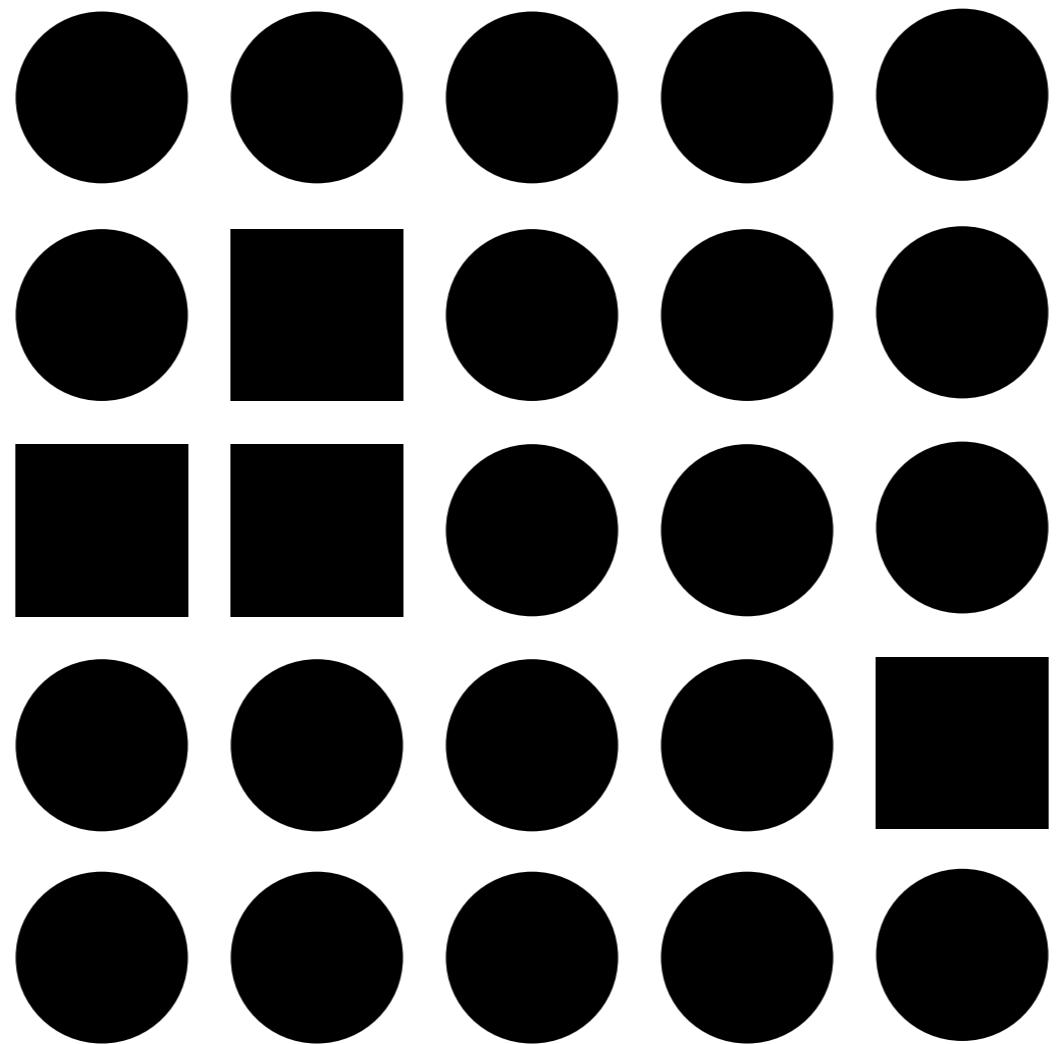
# A hasonlóság törvénye

- A hasonló kinézetű elemeket összetartozónak érzékeljük



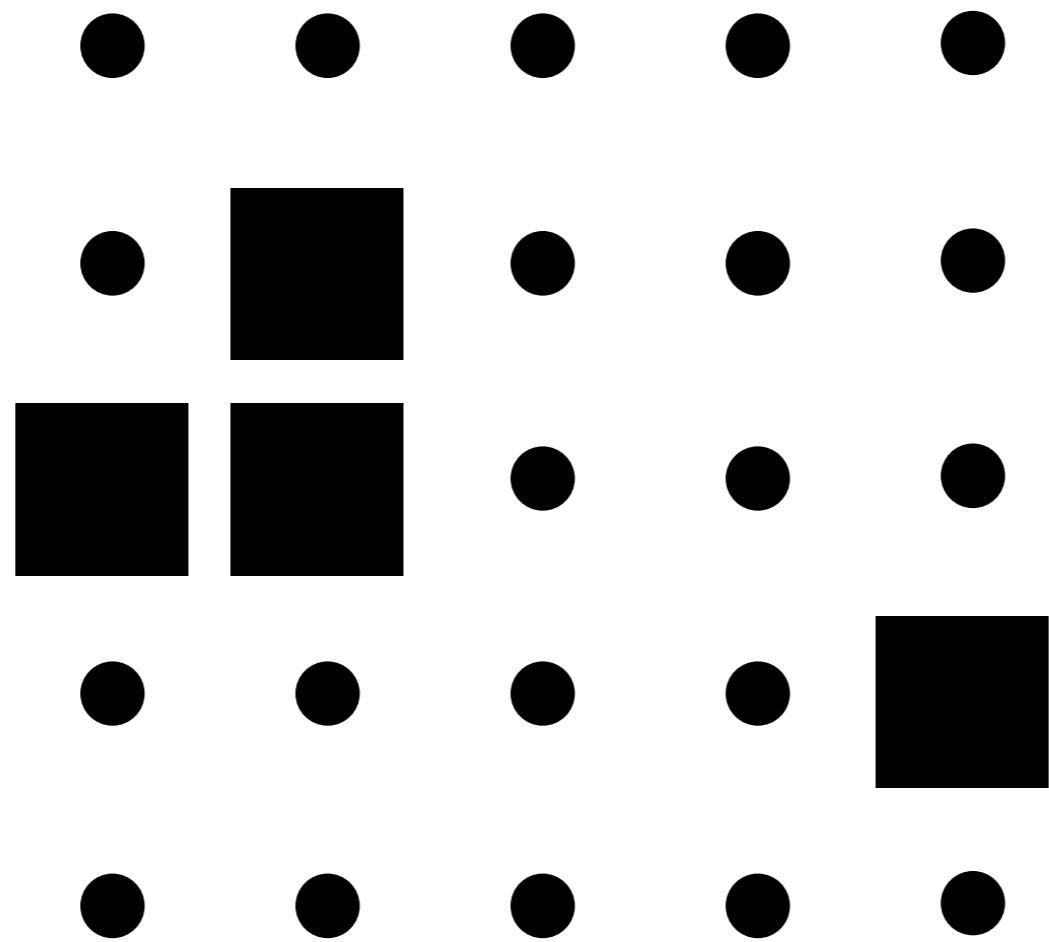
# A hasonlóság törvénye

- A hasonló kinézetű elemeket összetartozónak érzékeljük



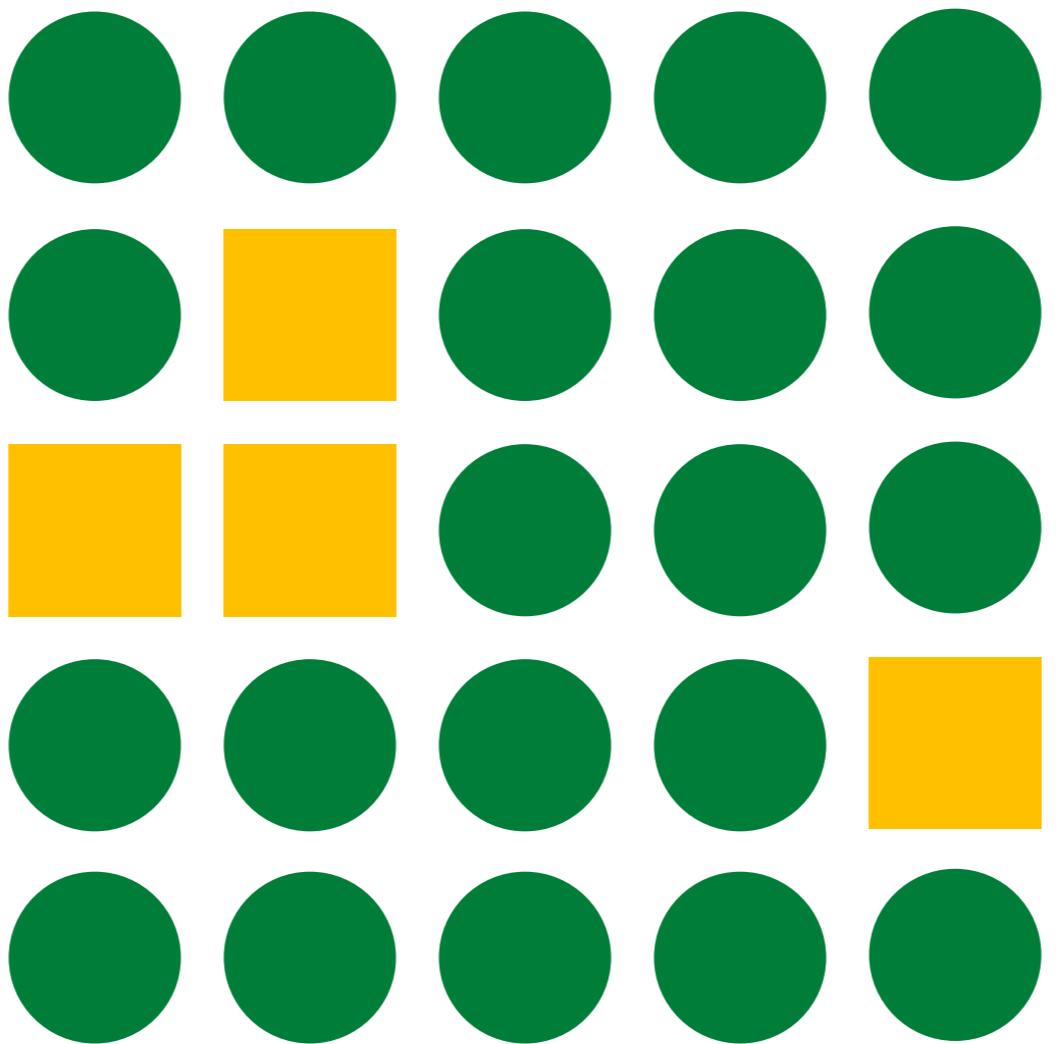
# A hasonlóság törvénye

- A hasonló kinézetű elemeket összetartozónak érzékeljük

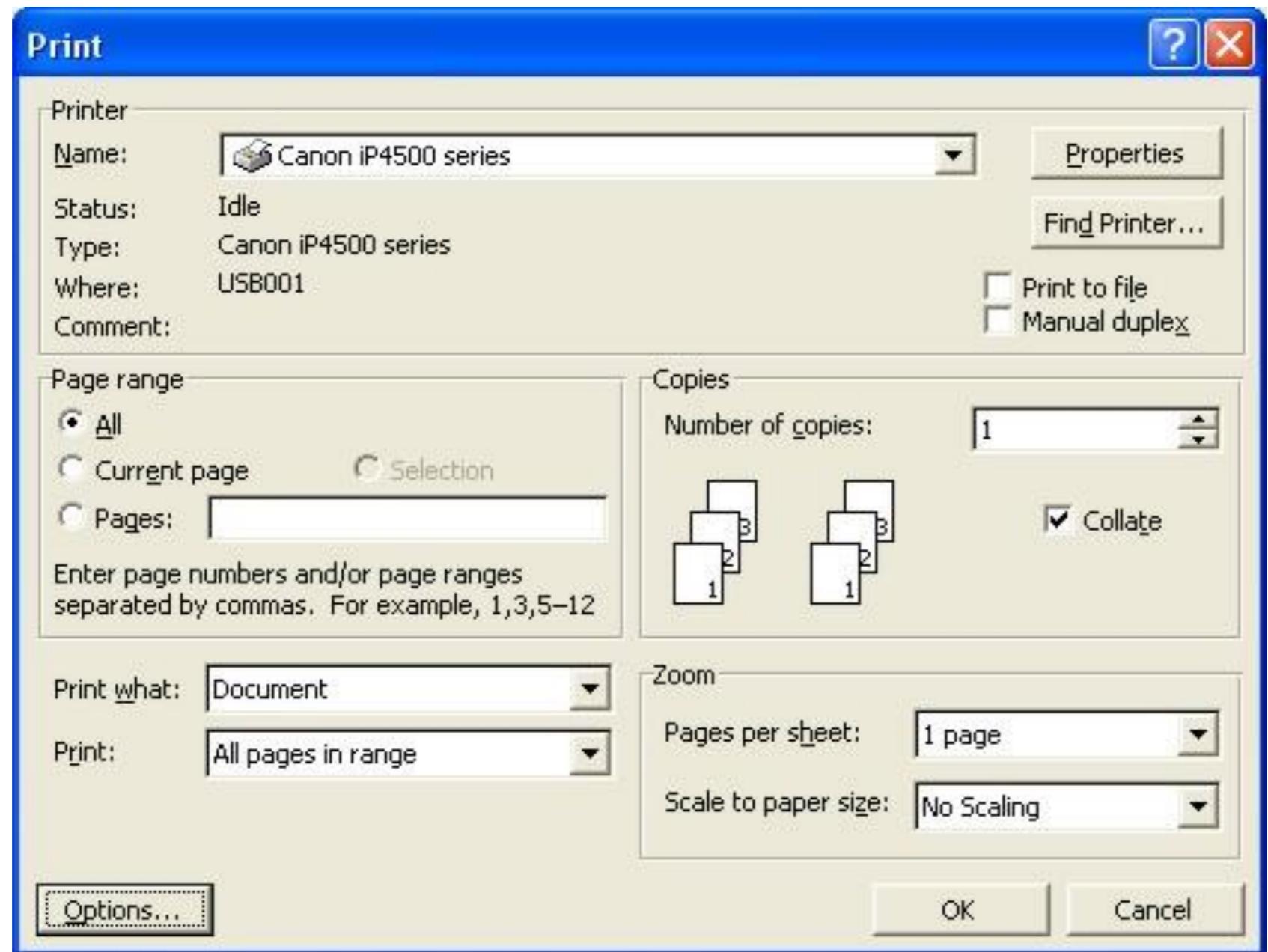


# A hasonlóság törvénye

- A hasonló kinézetű elemeket összetartozónak érzékeljük



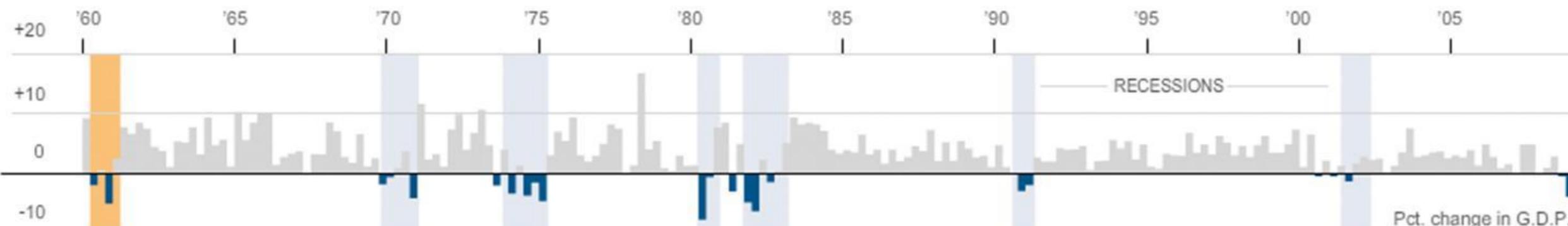
# Gestaltok az interfészeken



# ELRENDEZÉS IGAZÍTÁS ÉS RÁCSOK

# How the Government Dealt With Past Recessions

Since the Great Depression, presidents have frequently experimented with Keynesian economics to combat recessions. Three economists chronicle the history of government policy during past recessions and explain what worked and what didn't.



## 1960

April 1960-Feb. 1961

### The 'best and brightest'

Jeff Frankel  
Harvard University

John F. Kennedy was the first president to deliberately try Keynesian economics, using deficit spending in a recession to stimulate economic growth. But his tax cuts took too much time to take effect, convincing some economists that Keynesian policies were unrealistic.

## 1969 & 1973

Dec. 1969-Nov. '70; Nov. '73-March '75

### Broken models

Mark L. Gertler  
New York University

Economists were caught by surprise in the late 1960s and early 1970s. No models predicted increases in inflation, but it kept rising, possibly because of military buildup. Although President Richard M. Nixon tried wage and price controls, fiscal policy was not key in getting out of the recession.

## 1980 & 1981

Jan.-July 1980; July '81-Nov. '82

### Unexpected stimulus

Jeff Frankel  
Harvard University

Both fiscal and monetary policy played major roles in the recessions of the early 1980s. President Jimmy Carter's Fed chairman, Paul A. Volcker, raised the federal funds rate so much that it may have caused the 1981-82 recession. The increased spending and tax cuts under President Ronald Reagan – Keynesian in consequence, if not in design – led to economic recovery.

## 1990

July 1990-March 1991

### An active chairman

R. Glenn Hubbard  
Columbia University

The Persian Gulf War and rising inflation combined to cause the 1991 recession, which in turn caused a credit crunch. The federal government increased unemployment insurance benefits, but most of the recovery came from the Federal Reserve. Alan Greenspan cut the federal funds rate several times, which secured his reputation as a significant chairman.

## 2001

March-Nov. 2001

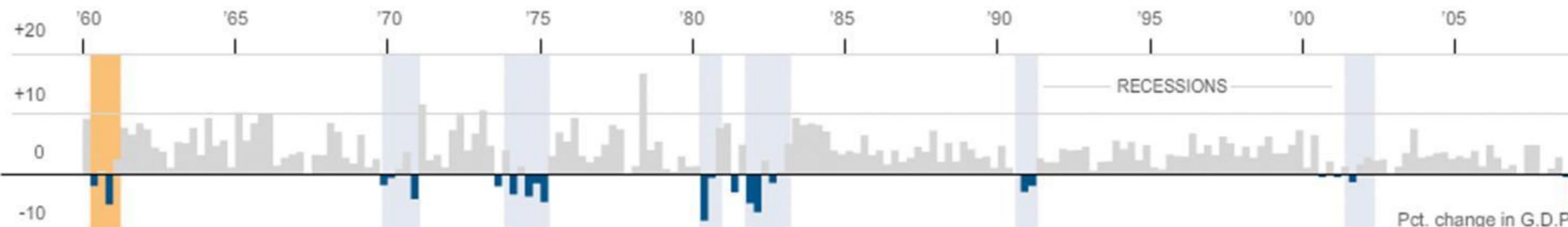
### Tax cuts, terror attacks

R. Glenn Hubbard  
Columbia University

The 2001 recession was a result of the tech bubble's bursting, which drained wealth around the country, and the Sept. 11 terror attacks. President George W. Bush's tax cuts provided a stimulus, and the Fed cut the federal funds rate several times and provided discount lending to combat the recession. The low interest rates, however, may have contributed to asset bubbles.

# How the Government Dealt With Past Recessions

Since the Great Depression, presidents have frequently experimented with Keynesian economics to combat recessions. Three economists chronicle the history of government policy during past recessions and explain what worked and what didn't.



## 1960

April 1960-Feb. 1961

### The 'best and brightest'

Jeff Frankel  
Harvard University

John F. Kennedy was the first president to deliberately try Keynesian economics, using deficit spending in a recession to stimulate economic growth. But his tax cuts took too much time to take effect, convincing some economists that Keynesian policies were unrealistic.

## 1969 & 1973

Dec. 1969-Nov. '70; Nov. '73-March '75

### Broken models

Mark L. Gertler  
New York University

Economists were caught by surprise in the late 1960s and early 1970s. No models predicted increases in inflation, but it kept rising, possibly because of military buildup. Although President Richard M. Nixon tried wage and price controls, fiscal policy was not key in getting out of the recession.

## 1980 & 1981

Jan.-July 1980; July '81-Nov. '82

### Unexpected stimulus

Jeff Frankel  
Harvard University

Both fiscal and monetary policy played major roles in the recessions of the early 1980s. President Jimmy Carter's Fed chairman, Paul A. Volcker, raised the federal funds rate so much that it may have caused the 1981-82 recession. The increased spending and tax cuts under President Ronald Reagan – Keynesian in consequence, if not in design – led to economic recovery.

## 1990

July 1990-March 1991

### An active chairman

R. Glenn Hubbard  
Columbia University

The Persian Gulf War and rising inflation combined to cause the 1991 recession, which in turn caused a credit crunch. The federal government increased unemployment insurance benefits, but most of the recovery came from the Federal Reserve. Alan Greenspan cut the federal funds rate several times, which secured his reputation as a significant chairman.

## 2001

March-Nov. 2001

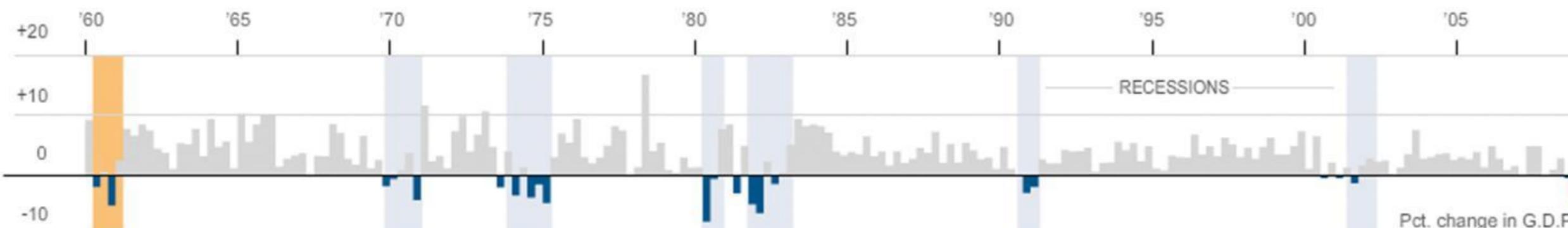
### Tax cuts, terror attacks

R. Glenn Hubbard  
Columbia University

The 2001 recession was a result of the tech bubble's bursting, which drained wealth around the country, and the Sept. 11 terror attacks. President George W. Bush's tax cuts provided a stimulus, and the Fed cut the federal funds rate several times and provided discount lending to combat the recession. The low interest rates, however, may have contributed to asset bubbles.

# How the Government Dealt With Past Recessions

Since the Great Depression, presidents have frequently experimented with Keynesian economics to combat recessions. Three economists chronicle the history of government policy during past recessions and explain what worked and what didn't.



1960	1969 & 1973	1980 & 1981	1990	2001
April 1960-Feb. 1961	Dec. 1969-Nov. '70; Nov. '73-March '75	Jan.-July 1980; July '81-Nov. 82	July 1990-March 1991	March-Nov. 2001
<b>The 'best and brightest'</b>	Broken models	Unexpected stimulus	An active chairman	Tax cuts, terror attacks
Jeff Frankel Harvard University	Mark L. Gertler New York University	Jeff Frankel Harvard University	R. Glenn Hubbard Columbia University	R. Glenn Hubbard Columbia University
John F. Kennedy was the first president to deliberately try Keynesian economics, using deficit spending in a recession to stimulate economic growth. But his tax cuts took too much time to take effect, convincing some economists that Keynesian policies were unrealistic.	Economists were caught by surprise in the late 1960s and early 1970s. No models predicted increases in inflation, but it kept rising, possibly because of military buildup. Although President Richard M. Nixon tried wage and price controls, fiscal policy was not key in getting out of the recession.	Both fiscal and monetary policy played major roles in the recessions of the early 1980s. President Jimmy Carter's Fed chairman, Paul A. Volcker, raised the federal funds rate so much that it may have caused the 1981-82 recession. The increased spending and tax cuts under President Ronald Reagan – Keynesian in consequence, if not in design – led to economic recovery.	The Persian Gulf War and rising inflation combined to cause the 1991 recession, which in turn caused a credit crunch. The federal government increased unemployment insurance benefits, but most of the recovery came from the Federal Reserve. Alan Greenspan cut the federal funds rate several times, which secured his reputation as a significant chairman.	The 2001 recession was a result of the tech bubble's bursting, which drained wealth around the country, and the Sept. 11 terror attacks. President George W. Bush's tax cuts provided a stimulus, and the Fed cut the federal funds rate several times and provided discount lending to combat the recession. The low interest rates, however, may have contributed to asset bubbles.



Search



Settings | Help \_ □ ×

collection marketplace social  
music videos pictures podcasts channels



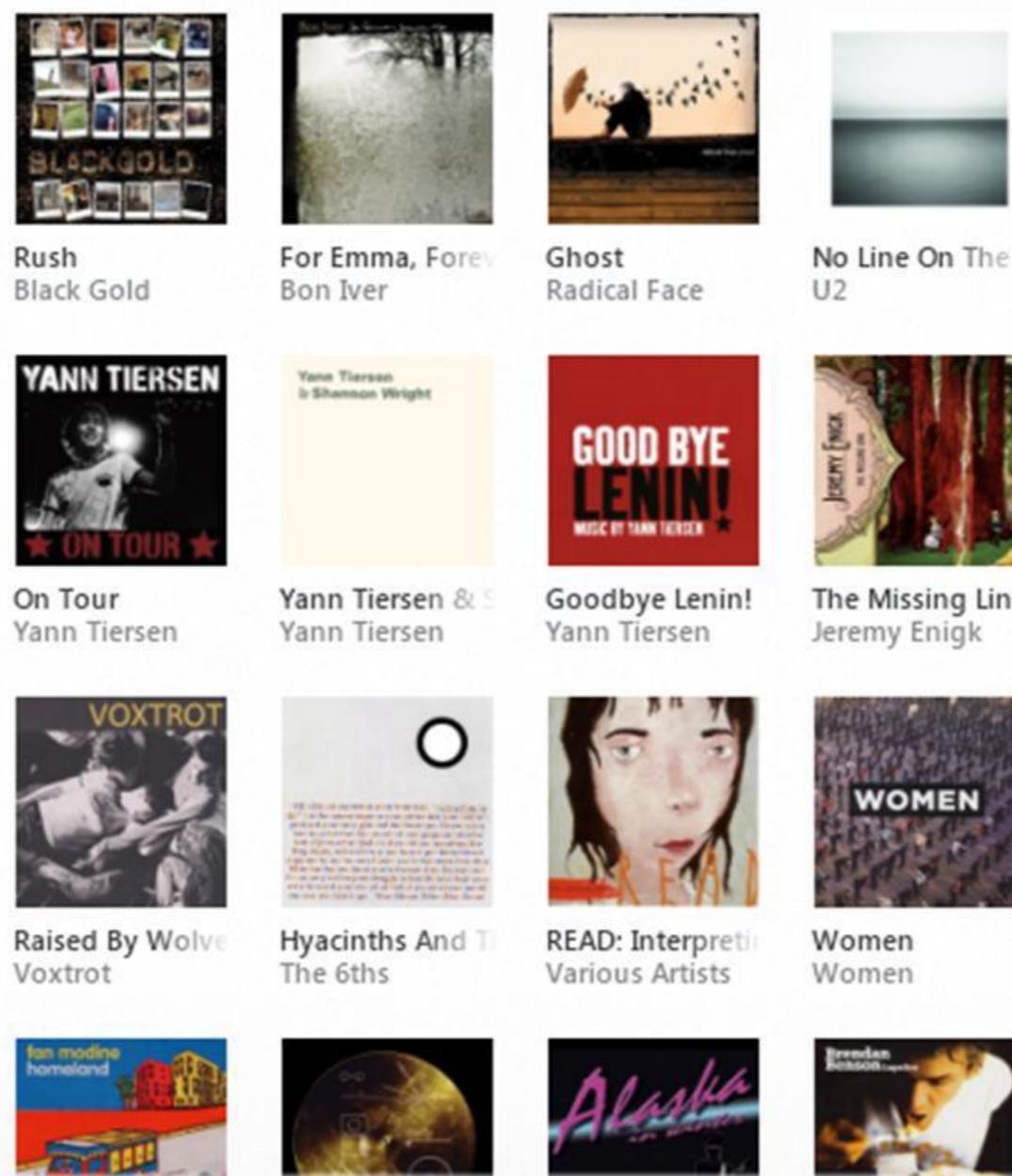
woodnote  
3,085 plays

artists genres albums songs playlists

29 artists a-z

Coldplay  
The Concretes  
Daniel Martin Moore  
Fan Modine  
Figurine  
Headlights  
Jeremy Enigk  
Juana Molina  
Loney Dear  
Melpo Mene  
Nick Drake  
Radical Face  
Signal Hill  
Tunng  
U2  
Unknown Artist  
Various Artists  
Voxtrot  
Women  
Yann Tiersen

47 albums by date added



439 songs a-z

01 - Ready Lets GO  
03 - Beware the Friend  
11 - Energy Warning  
15 - I Saw Drones  
1969  
1er Réveil Par Temps  
42  
After The Flood  
Airport Surroundings  
Along The Road  
Alpha and Omega  
Also Frightened  
Anonanimal  
Are You Prepared  
Arms  
As a Dream from Above  
As Four  
As You Turn To Go  
Asleep On A Train  
Babes And Darlings

		Search			Settings	Help	×
		Collection	marketplace	social			
		music	videos	podcasts	channels		
29 artists a-z		47 albums by date added				439 songs a-z	
Coldplay		BLACK GOLD		THE CONCRETES		01 - Ready Lets GO	
The Concretes						03 - Beware the Friend	
Daniel Martin Moore				DANIEL MARTIN MOORE		11 - Energy Warning	
San Morine						15 - I Saw Drones	
Figurine		Rush	Black Gold	For Emma, Forever Ago	Bon Iver	Ghost	Radical Face
Headlights						No Line On The Horizon	U2
Jeremy Enigk		YANN TIERSEN		<small>Yann Tiersen &amp; Matthew Wright</small>			
Juana Molina						1969	
Loney Dear						1er Réveil Par Temps D	
Helpo Mene						42	
Nick Drake		On Tour	Yann Tiersen	Yann Tiersen & Matthew Wright		After The Flood	
Radical Face						Airport Surroundings	
Signal Hill		VOXTROT				Along The Road	
Tunng							
J2							
Unknown Artist							
Various Artists		Raised By Wolves	Voxtrot	Hyacinths And The 6ths		Alpha and Omega	
Voxtrot						Also Frightened	
Nomen						Anonanimal	
Yann Tiersen		NOMEN		HYACINTHS AND THE 6THS		Are You Prepared	
						Arms	
						As a Dream from Above	
						As Four	
						As You Turn To Go	
						Asleep On A Train	
						Babes And Darlings	

←

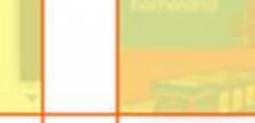
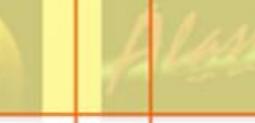
Search  P

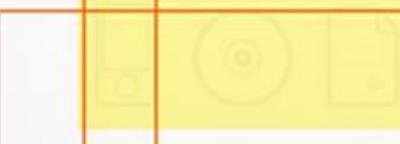
Settings | Help 

Collection marketplace social  
music videos pictures podcasts channels

woodnote 3,085 plays 

artists genres albums songs playlists

29 artists a-z	47 albums by date added			439 songs a-z
Coldplay				01 - Ready Lets GO
The Concretes				03 - Beware the Friend
Daniel Martin Moore				11 - Energy Warning
Fan Modine				15 - I Saw Drones
Figurine	Rush Black Gold	For Emma, Forever Bon Iver	Ghost Radical Face	1969
Headlights			No Line On The U2	1er Réveil Par Temps T
Jeremy Enigk		<small>Yann Tiersen &amp; Silvana Mazzoni</small>		42
Juana Molina				After The Flood
Loney Dear				Airport Surroundings
Melpo Mene				Along The Road
Nick Drake	On Tour Yann Tiersen	Yann Tiersen & Yann Tiersen	Goodbye Lenin! Yann Tiersen	Alpha and Omega
Radical Face			The Missing Link Jeremy Enigk	Also Frightened
Signal Hill				Anonanimal
Tunng				Are You Prepared
J2				Arms
Unknown Artist				As a Dream from Above
Various Artists	Raised By Wolves Voxtrot	Hyacinths And The 6ths	READ: Interpretations Various Artists	As Four
Voxtrot			Women Women	As You Turn To Go
Nomen				Asleep On A Train
Yann Tiersen				Babes And Darlings

 VOLUME 35



Search



Settings | Help \_ □ ×

collection marketplace social  
music videos pictures podcasts channels



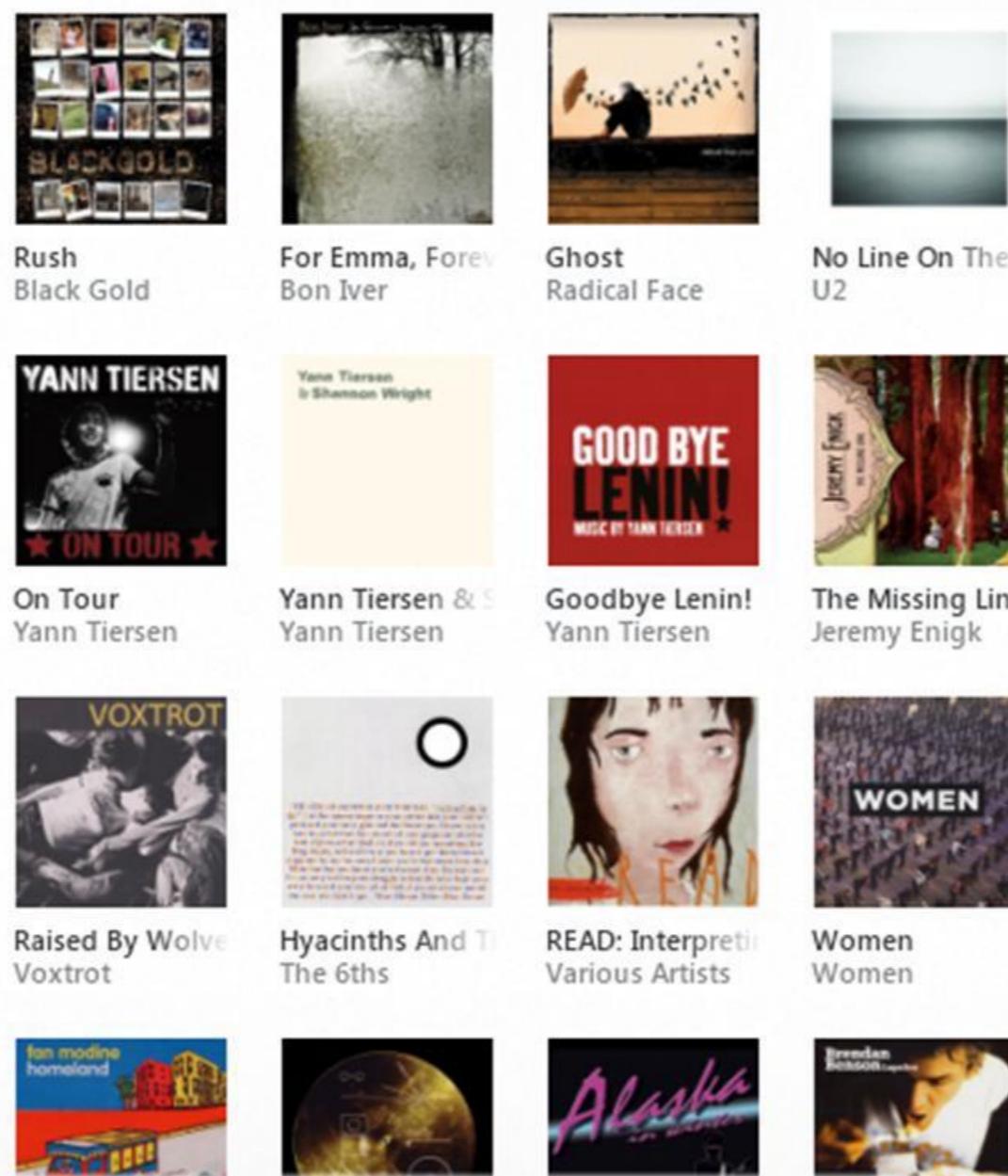
woodnote  
3,085 plays

artists genres albums songs playlists

29 artists a-z

Coldplay  
The Concretes  
Daniel Martin Moore  
Fan Modine  
Figurine  
Headlights  
Jeremy Enigk  
Juana Molina  
Loney Dear  
Melpo Mene  
Nick Drake  
Radical Face  
Signal Hill  
Tunng  
U2  
Unknown Artist  
Various Artists  
Voxtrot  
Women  
Yann Tiersen

47 albums by date added



439 songs a-z

01 - Ready Lets GO  
03 - Beware the Friend  
11 - Energy Warning  
15 - I Saw Drones  
1969  
1er Réveil Par Temps  
42  
After The Flood  
Airport Surroundings  
Along The Road  
Alpha and Omega  
Also Frightened  
Anonanimal  
Are You Prepared  
Arms  
As a Dream from Above  
As Four  
As You Turn To Go  
Asleep On A Train  
Babes And Darlings





Search



Settings | Help \_ □ ×

collection marketplace social  
music videos pictures podcasts channels



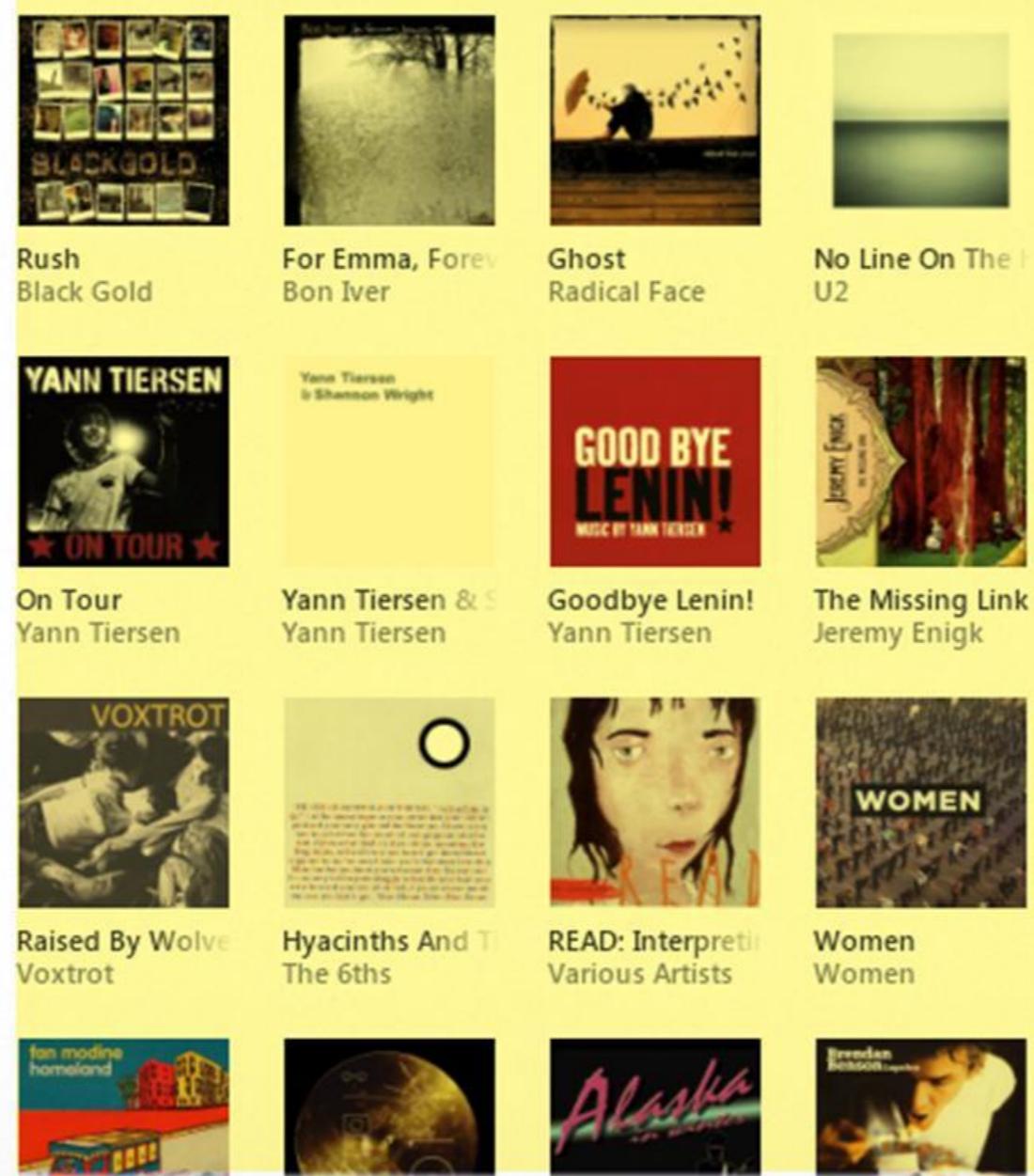
woodnote  
3,085 plays

artists genres albums songs playlists

29 artists a-z

Coldplay  
The Concretes  
Daniel Martin Moore  
Fan Modine  
Figurine  
Headlights  
Jeremy Enigk  
Juana Molina  
Loney Dear  
Melpo Mene  
Nick Drake  
Radical Face  
Signal Hill  
Tunng  
U2  
Unknown Artist  
Various Artists  
Voxtrot  
Women  
Yann Tiersen

47 albums by date added



439 songs a-z

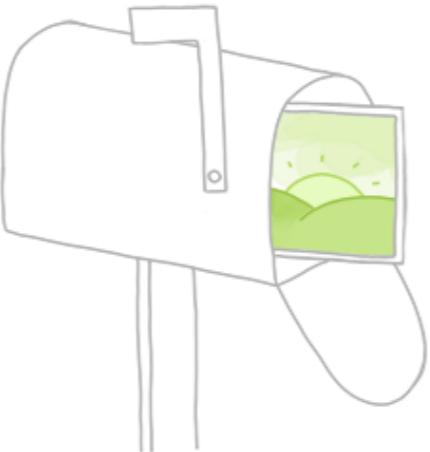
01 - Ready Lets GO  
03 - Beware the Friend  
11 - Energy Warning  
15 - I Saw Drones  
1969  
1er Réveil Par Temps  
42  
After The Flood  
Airport Surroundings  
Along The Road  
Alpha and Omega  
Also Frightened  
Anonanimal  
Are You Prepared  
Arms  
As a Dream from Above  
As Four  
As You Turn To Go  
Asleep On A Train  
Babes And Darlings





## Wherever you are

Put your stuff in Dropbox and get to it from your computers, phones, or tablets. Edit docs, automatically add photos, and show off videos from anywhere.



## Share with confidence

Share photos with friends. Work with your team like you're using a single computer. Everything's automatically private, so you control who sees what.



## Safe and secure

Even if your phone goes for a swim, your stuff is always safe in Dropbox and can be restored in a snap. Dropbox secures your files with 256-bit AES encryption and two-step verification.

BME / UT

The logo for Fry's Electronics, featuring the word "Fry's" in a stylized, italicized font where the letters are interconnected, followed by "ELECTRONICS" in a smaller, sans-serif font.

(818) 526-8100 • FAX (818) 526-8118  
FOUNTAIN VALLEY 10000 Santa Rita A

(310) 364-FRYS (3797) • FAX (310) 364-3718  
**ANAHUAC 3337 E. La Palma**

**CITY OF INDUSTRY 13401 Crossroads Parkway North  
(562) 463-2400 • FAX (562) 463-2418**

**Prices Good Today Only, Fri May 18, 2007.**

Offer valid only in metropolitan circulation area of newspaper in which advertisement appears. Limit one coupon per household. Not responsible for typographical errors.

**STORE HOURS**  
**M-F 8-9, Sat 9-9, Sun 9-7**

www.foopack.com

**THE COMPLETE  
LOW PRICE GUARANTEE**  
"We Will Match Any  
Competitive Price."

11. Valid on selected items only. Required minimum monthly payments greater of \$10 or 1% of balance plus listed finance charges plus any late fees (if applicable). Total minimum monthly payment (not all interest) of the Protection Plus minimum payments plus the Regular Plan.

1

ELRENDEZÉS  
**DOMINANCIA**

# Kiemelés a sokaságból

- „Itt van ez a sok minden, mi ebből a fontos”
  - Belépési pontot kell adnunk a felületen („Hol kezdjem? Honnan induljak?”)
  - Ki kell fejeznünk az elemek relatív fontosságát
    - Ettől tudjuk „átfutni” a képernyőt
- Kontraszttal kiemeljük a domináns elemeket

# Átfuthatóság és hierarchia – tartalmakhoz

The New York Times  
"All the News That's Fit to Print"  
VOL. CCLXII No. 54,546  
Price Two for Four  
NEW YORK, MONDAY, JANUARY 5, 2009  
B1 \$1

**Israeli Attack Splits Gaza; Truce Calls Are Rebuffed**

**Hospital Fills Up, Mainly With Civilians**

**Death Toll Passes 500 — City Is Surrounded**

**By STEPHEN BROOKNER**  
Gaza — A missile hit their car at 11 a.m., which was made of concrete and so, the local family had thought, is bullet-proof. Refugees, older than their wives, fled by sea. Five hours later, it was not clear where it struck, but her mother, Mrs. Farhat, 68, lay dead.

On Tuesday, the day after Israel began its ground incursion of Gaza, Rafah General Hospital at 8600 Hospital St. in Gaza City, one of the city's eight new civilian medical facilities, was under fire again.

"I heard that big boom!" cried Mrs. Farhat, who has been staying in her son's house since the conflict began. "Then I got the first news."

Her son, 26, left home in the morning, while his wife, 24, stayed at the other end of the city. Only half of the body of the 16-year-old daughter of the family emerged. "Was that your daughter?" asked a doctor. "Yes, that's our daughter," Farhat said.

She remained in a room nearby for three days. In the confusion, many Palestinians panic. Hospital officials, however, insisted that the group has prepared against possible injuries by fighting in and around populated areas.

The hospital, like others in Gaza, is run by the United Nations Relief and Works Agency (UNRWA). It has been under fire since the beginning of the conflict, and its staff have been forced to leave the building.

On Tuesday, the hospital was hit by a missile, killing two people. The hospital has been hit twice since the beginning of the conflict, and its staff have been forced to leave the building.

The hospital has been hit twice since the beginning of the conflict, and its staff have been forced to leave the building.

**Richardson Won't Pursue Cabinet Post**

**By MICHAEL KATZ STERNBERG**  
Washington — Once Mitt Romney's team was sure that Richard Cheney chose to remain in his cabinet, they considered the possibility of a political compromise that would have "forced an untenable choice" in his confirmation.

The president-elect and the governors, close friends as well as political allies, announced the compromise in joint statements. Mr. Richardson, one of the most senior members of the administration, will decline to vote on his confirmation, while Mr. Cheney will remain in his position.

Mr. Obama said he accepted "with deep regret" Mr. Bush's decision to leave out, though he had been invited to do so.

**Obama Plan Includes \$300 Billion in Tax Cuts**

**Breaks for Business and Workers Are Nod to Critics**

**By PETER BAKER and CRAIG KOHLER**  
Washington — President Barack Obama plans to propose about \$300 billion in tax cuts for workers and businesses in his economic recovery program, although not Sunday, as he took steps to veto some legislation previously passed that was too focused on general tax cuts.

The legislation Mr. Obama is negotiating with congressional Democrats will devote about 40 percent of the cost to tax cuts, in keeping with his campaign promise to provide credits to help spur job creation, cutting through his conservative critics' argument that what they describe as an overhauls approach to spending.

Although some tax cuts were proposed in his budget, Mr. Obama's economic package, his team disclosed, will focus on more details of the plan or the cost of a time when Republicans have begun raising criticisms of what they describe as an overhauls approach to spending.

The proposal, which will also include more than \$30 billion in tax incentives for business to create jobs and invest in equipment or factories.

The overall economic package, of \$787 billion to \$789 billion, is taking shape as Mr. Obama prepares in Washington and prepares in Congress to build support on Congress and among the broader public for his approach to stimulating the economy. Mr. Obama also has to negotiate a deal to get the Senate to pass his legislation, including the \$80 billion in tax cuts.

**Kidnapping, Long Feared in Mexico, Send Shivers Across Border**

**By RICHARD BROWN**  
Mexico City — Mexico's long-festering fear exploded in the winter hours of an unexpected terrorist strike in Mexico City, sending shock waves across the border.

https://magyarorszag.hu/

**MAGYARORSZÁG.HU**  
Kormányzati Portál

**ÜGYFÉLKAPU**

**Aktuális**

**Lemondott Schmitt Pál köztársasági elnök**

A köztársasági elnök hétfőn, napirend előtti felszólalásában jelentette be lemondását, döntését azzal indokolta, hogy olyan helyzetben, amikor személyes ügye inkább megosztja, mint egységesít a nemzetet, kötelességeinek érzi, hogy szolgálatát befejezze és lemondjon mandátumáról. A Ház 338 igen szavazattal, 5 nem ellenében, 6 tartózkodás mellett elfogadta Schmitt Pál lemondását.

**Katalógus**

**Ügyintézés**  
Ügyek témák szerint, A-Z lista, Internetes okmányiroda, szolgáltatások...

**Keresés**  
Jogsabálykereső, Hivatalkereső, Ingatlankereső, Gépjárműkereső...

**Közigazgatás**  
Intézmények, Portréta, Kormány...

**Kapcsolat**  
Írjon nekünk, Kormányzati kapcsolatok, Kormányzati Ügyfélvonal 1818...

**.hu A-Z Lista**

A Á B C Cs D E É F G Gy H I Í J K L Ly M N Ny O Ó Ö P Q R RSS

**.hu Szolgáltatások**

Dokumentumfeltöltés  
eBEV-szolgáltatások  
Internetes okmányiroda  
Fogyasztóvédelmi beadvány

**TIPP**

**.hu Címkek**

Afganisztán, áradás, Baja, Batiz András, Burány, egyeztetés, export, gyász, Horvát, Horvátorzág, IMF, integráció, katonáság, Kossuth, központ, Lévai

# Belépési pont

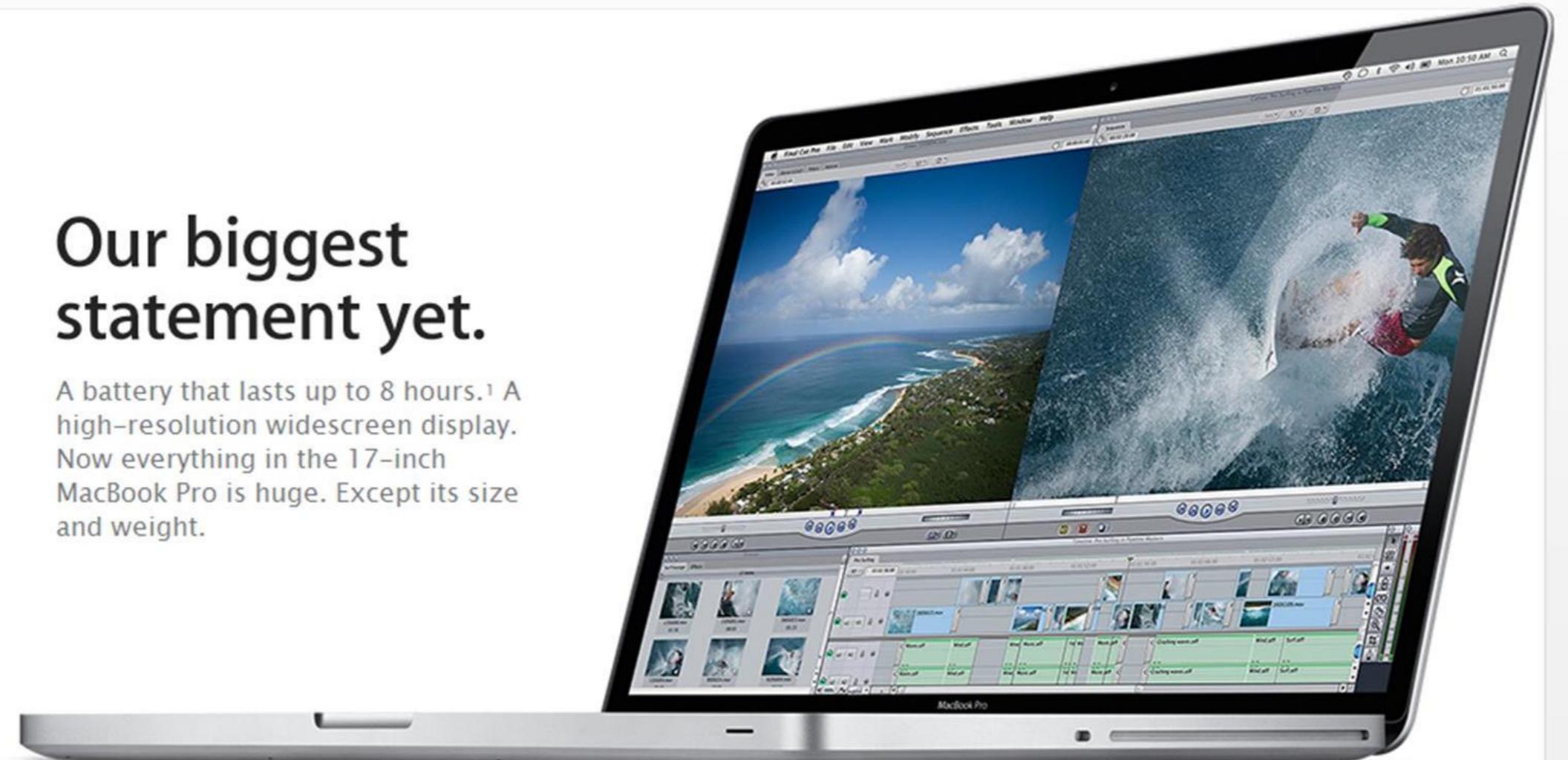
Store Mac iPod + iTunes iPhone Downloads Support  Search

## MacBook Pro

Design 15-inch Features 17-inch Features Graphics Mac OS X + iLife Environment Tech Specs [Buy Now](#)

**Our biggest statement yet.**

A battery that lasts up to 8 hours.<sup>1</sup> A high-resolution widescreen display. Now everything in the 17-inch MacBook Pro is huge. Except its size and weight.



**The longest-lasting Mac notebook battery ever.**

The battery in the new 17-inch MacBook Pro lasts up to 8 hours on a single charge<sup>1</sup> and can be recharged up to 1000 times<sup>2</sup> — compared with only 200 to 300 times for typical notebooks. To do this, Apple engineers custom-designed



**A breakthrough in battery design.**  
Only in the 17-inch



# Túl sok kiemelés = nincs kiemelés

The image is a screenshot of the Havenworks.com website. At the top left is a logo with the letters 'TT' and a small graphic. The main header is 'Havenworks.com' in large blue and black letters. Below it is a sub-header 'News Reference Facts Information Sources Intelligence Haven Works !-' in yellow. To the right of the sub-header are links for 'Wednesday, 11 March 2009', 'CALENDAR', 'Search: Google', 'GO', 'HavenWorks Web', '+2008 News / Blog', and a link to 'TV Online Television' which shows a grid of TV channel logos. A search bar is also present. The page features several sections: 'Democratic News' with a grid of Democratic politicians' photos, 'Republican News' with a grid of Republican politicians' photos, and a central section titled 'Atwater Politics'. There are also sections for 'DEM 2010 TV' (featuring Obama TV, White House TV, and ChangeDotGov TV), 'News' Media Politics (featuring Jay Rosen and Glenn Greenwald), and 'Weblog' (with an opinion section). The sidebar on the left contains links for 'News Topics', 'A-Z', and various categories like 'Secret', 'Terrorism', 'Human', 'Rights', 'Law', 'US', 'Guantánamo', 'Cuba', 'Former Gitmo Guard Tells All.', and a detailed paragraph about Brandon Neely's story. The right sidebar has sections for 'U. S. A.' (with links to Barack Obama, Corporate, Labor, Makers, US, Global, Legislation, Secret, Terrorism, and Detainees), and a large text block about Obama's approach to trade and labor.

# A részleteken múlik

- Ha nincsenek konzisztens betűméretek  
...elvész a hierarchia



MAGYARORSZÁG.HU  
Kormányzati Portál



Magyarország.hu | Ügyintézés | Ügyfélkapu | Keresés | Közigazgatás | portáliindex | honlapterkép | üzemeltetési információk | mellékletek | hírlevél | címletek

## Aktuális

### Lemondott Schmitt Pál köztársasági elnök

A köztársasági elnök hétfőn, napirend előtti felszólalásában jelentette be lemondását, döntését azzal indokolta, hogy olyan helyzetben, amikor személyes ügye inkább megosztja, mint egységesít a nemzetet, kötelességének érzi, hogy szolgálatát befejezzé és lemondjon mandátumáról. A Ház 338 igen szavazattal, 5 nem ellenében, 6 tartózkodás mellett elfogadta Schmitt Pál lemondását.

## Katalógus

### Ügyintézés

Ügyek témák szerint, A-Z lista  
Internetes okmányiroda,  
szolgáltatások...

### Keresés

Jogsabálykereső, Hivatalker  
Ingatlankereső, Gépjárműker

### Közigazgatás

Intézmények, Portrétár, Kormányzati Ügyfélvonal 181  
Írjon nekünk, Kormányzati kaj

1960

April 1960-Feb. 1961

The 'best and brightest'



Jeff Frankel  
Harvard University

John F. Kennedy was the first president to deliberately try Keynesian economics, using deficit spending in a recession to stimulate economic growth. But his tax cuts took too much time to take effect, convincing some economists that Keynesian policies were unrealistic.

1969 & 1973

Dec. 1969-Nov. '70; Nov. '73-March '75

Broken models

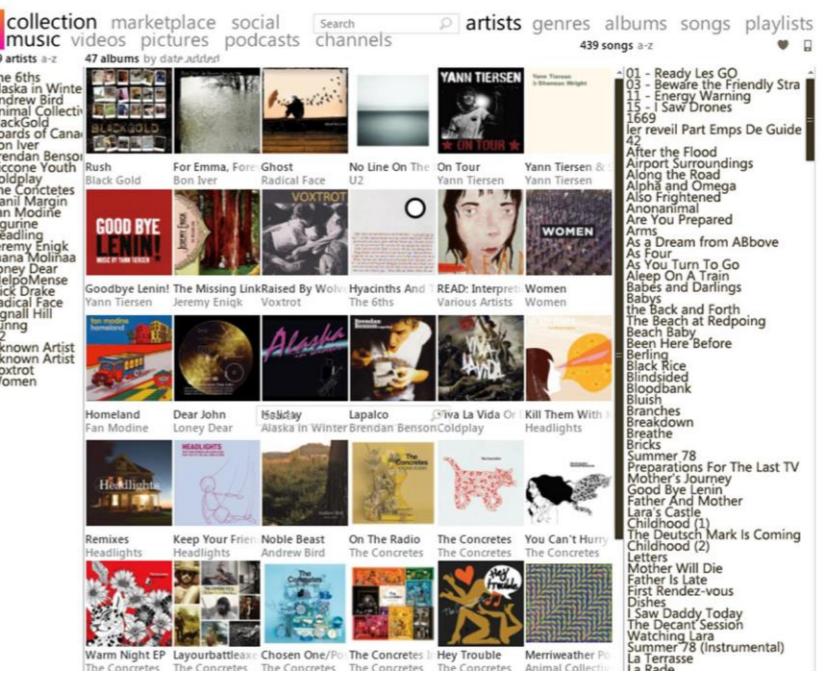


Mark L. Gertler  
New York University

Economists were caught by surprise in the late 1960s and early 1970s. No models predicted increases in inflation, but it kept rising, possibly because of military buildup. Although President Richard M. Nixon tried wage and price controls, fiscal policy was not key in getting out of the recession.

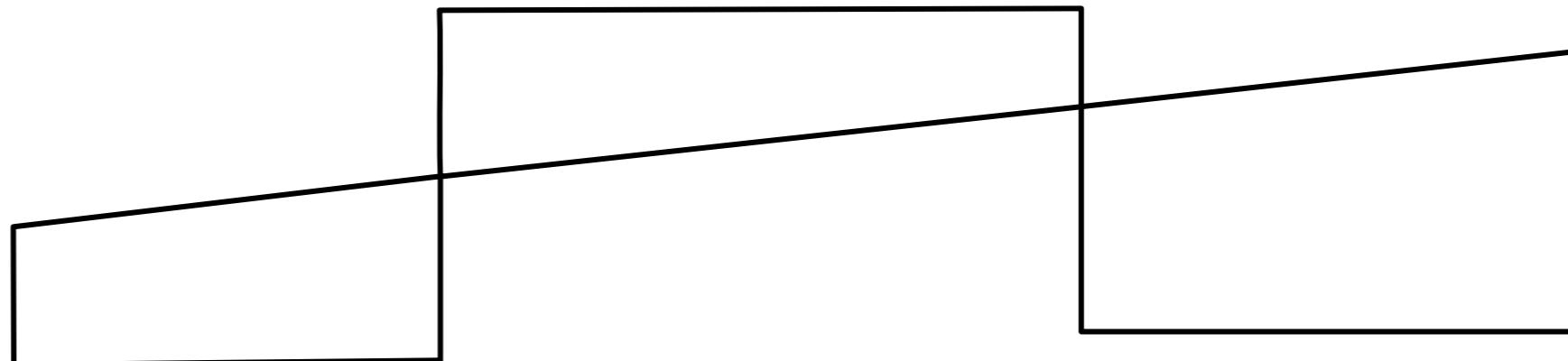
# A részleteken múlik

- Ha nincsenek konzisztens betűméretek  
...elvész a hierarchia
- Ha nincsenek rendben a térközök  
...elvész az átláthatóság

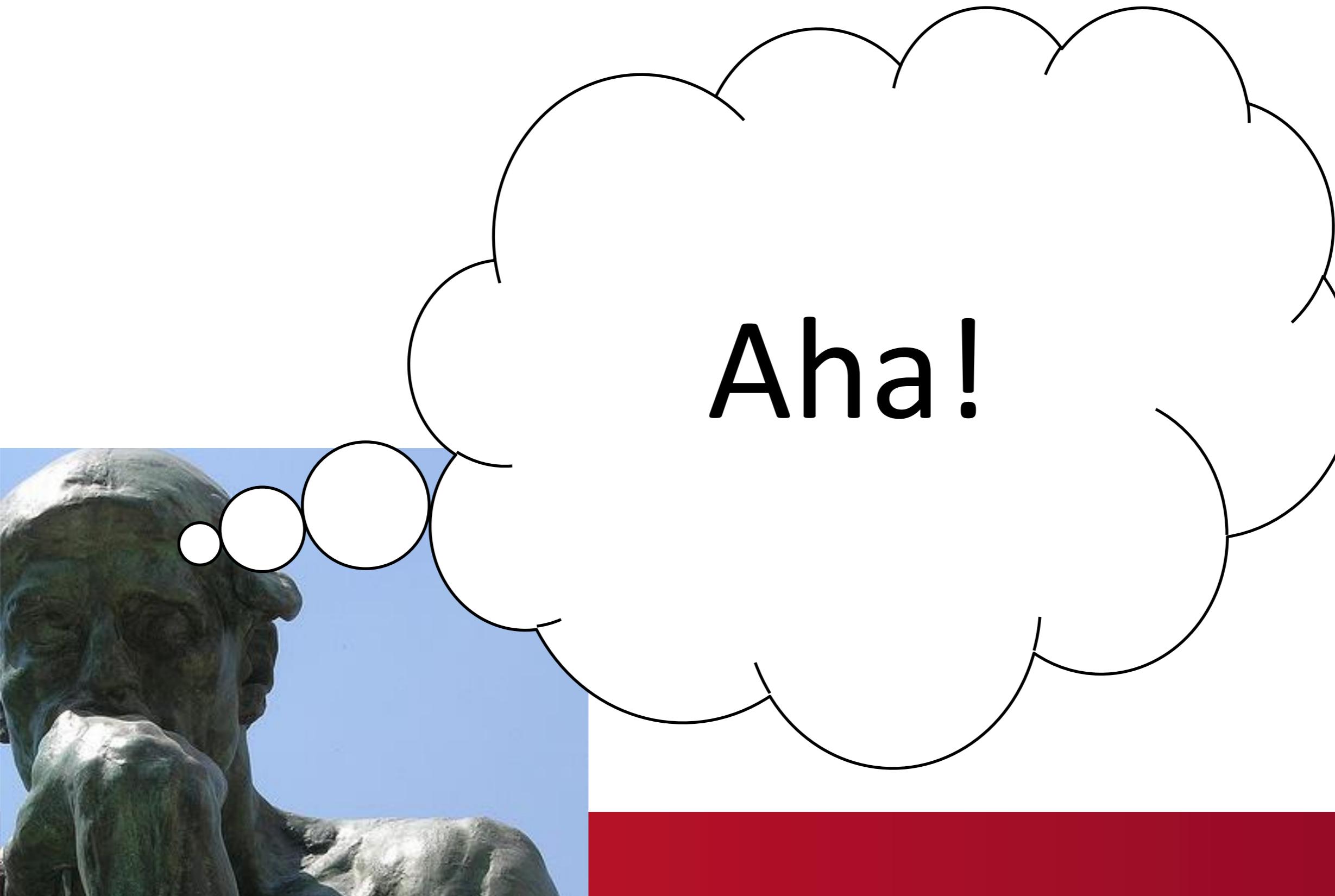


# A részleteken múlik

- Ha nincsenek konzisztens betűméretek  
...elvész a hierarchia
- Ha nincsenek rendben a térközök  
...elvész az átláthatóság
- Ha nincsenek egy vonalban az elemek  
...nem alakul ki az összetartozáság



# Ha minden jól csináltunk



A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI, HOGY BE  
TUDOM ÁLLÍTANI AZ ÉBRESZTŐRÁT ANÉLKÜL,  
HOGY FELIDEGESÍTENE, ÉS NEM MÁSNAP DÉLBEN  
DERÜL KI, HOGY VALAMIT ELRONTOTTAM.

- A lényeg az, hogy kényelmesen tudjam használni, nem kell, hogy „élmény” legyen!
- Persze még jobb, ha jól is néz ki, kellemes a hangja stb., de minden másodlagos

A FELHASZNÁLÓI ÉLMÉNY AZT JELENTI,  
HOGY EL TUDOK VÉGEZNI EGY FELADATOT  
KÜLÖNÖSEBB NEHÉZSÉGEK NÉLKÜL.

ÚGY ÉRZEM, ELÉRTEM A CÉLOM, A  
HASZNÁLAT KÉNYELMES VOLT,  
SIKERÉLMÉNYEM VAN.

# Kliensoldali technológiák

## TypeScript

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki  
és Gazdaságtudományi Egyetem oktatója által  
kidolgozott szerzői mű.**

**Kifejezetten felhasználási engedély nélküli  
felhasználása szerzői jogi jogosításnak minősül.**

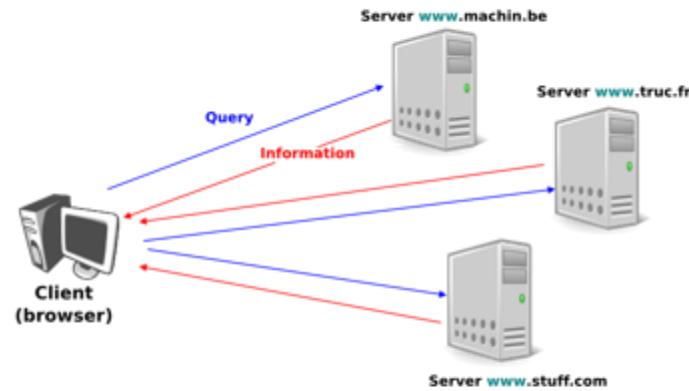
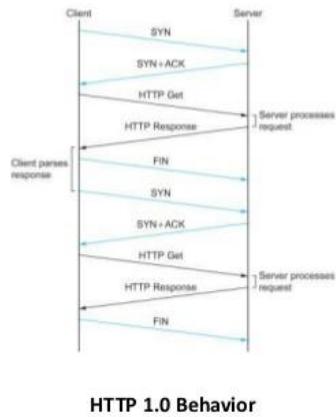
A szerző elérhetősége:  
[Szabo.Gabor@vik.bme.hu](mailto:Szabo.Gabor@vik.bme.hu)

# A webfejlesztés alapjai

Ismétlés címszavakban

# Webes alaptechnológiák: HTTP

## HyperText Transfer Protocol



# Webes alaptechnológiák: HTTP

- URI/URL, header, body
- GET, PUT, POST, DELETE
- Státuszkódok
- Kérés-válasz
- Állapotmentesség, állapotkezelés
  - > Sütik, kliens-/szerveroldali memória
- Statikus/dinamikus kiszolgálás
- Biztonság (HTTPS)

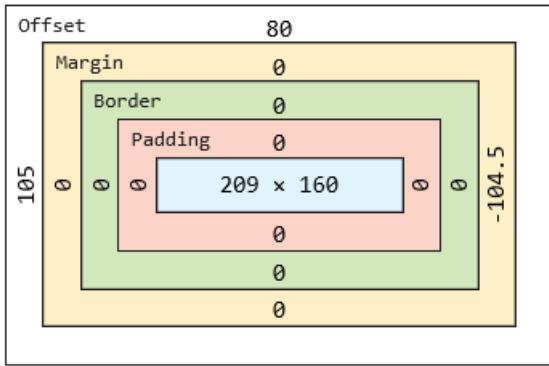
# Webes alaptechnológiák: HTML

```
<n-WW ltr" data-priority="2" onfocus="" onload="if(_w.lb)lb();_ge('SD_TUI___')>
 tyles" data-bm="25">...</div>
 "text/javascript">//<![CDATA[_G.AppVer="8_1_2_6...</script>
 >_table" role="none">...</table>
 = "text/javascript">//<![CDATA[(function(n,t){onl...</script>
 msDefer">...</div>
 >="text/javascript">//<![CDATA[_G.HT=new Date; //...</script>
 c="/rb/5p/cj,nj/3e6a7d75/9a358300.js?bu=EvQekx-8Hr8e3gTNHs8enx_RHtge4B6LH4kf_R7vHfoc_RzyHQ" type="text/javascript" data-rms="1">
 src="/rs/2T/j0/cj,nj/64c6b209/6c8d22b8.js" type="text/javascript" data-rms="1"></script>
 type="text/javascript" data-rms="1"/>//<![CDATA[(function(n,t,i){f...</script>
 src="/rs/3T/i1/cj,nj/1beceeda/3baa9af7.js" type="text/javascript" data-rms="1"></script>
 !="sbicom_loader" style="display:none" data-evt="postRBCComplete" data-id="SERP.5083" data-ptn="Homepage" data-load="1">...</div>
 it type="text/javascript" data-rms="1">//<![CDATA[_w.sj_evt&&sj_evt...</script>
 pt src="/rs/2T/fN/cj,nj/d83a28bc/699c87d7.js" type="text/javascript" data-rms="1"></script>
 ipt src="/rs/30/1H/cj,nj/5983aa50/f8c6dd44.js" type="text/javascript" data-rms="1"></script>
 ipt src="/rs/30/1X/cj,nj/4c7364c5/40e1b425.js" type="text/javascript" data-rms="1"></script>
 ript type="text/javascript" data-rms="1">//<![CDATA[var wlc=function(n...</script>
 ript src="/rs/30/2f/cj,nj/bf587ad6/f1d86b5a.js" type="text/javascript" data-rms="1"></script>
 script type="text/javascript" data-rms="1">//<![CDATA[(function(){functi...</script>
 script src="/rs/6n/kb/cj,nj/6240f061/6fb5e8ee.js" type="text/javascript" data-rms="1"></script>
 <form class="sbi_form_ph" id="sbi_form" action="https://www.bing.com/images/search?view=detai...&iss=%SBIPAGENAME%&FORM=%FORMCODE%&n' u=https%3A%2F%2Fwww.bing.com%2F&sbsrc=%SOURCE%&q=%IMAGEURL%&idpbck=1" enctype="multipart/form-data" method="POST">...</form>
 <script type="text/javascript" data-rms="1">//<![CDATA[var sj_ajax=functi...</script>
 <script src="/rb/16/cj,nj/1b7dfb88/cc8437ad.js?bu=DikuXGxwdGhgZKwBsAEuoAEu" type="text/javascript" data-rms="1"></script>
 <script src="/rb/G/cj,nj/724ba06b/778522c1.js?bu=Gb0BIAWXBZoFLCwsLCydbZkELCws5wP2A_kDLcwskgQsLKcF_AM" type="text/javascript" data- ~~~~.cianin1.0&rpsnv=11&ct=1554624816&rver=6.0.5286.0&wp=MBI_SSL&wreply=https%3F%2 ~~~~.ctvle="display: none;" data-priority="2">...</iframe>
```

# Webes alaptechnológiák: HTML

- XML-szerű, deklaratív, struktúra leírására való jelölőnyelv
- Elemek, attribútumok, osztályok
- HTML5: storage API, canvas, A/V, Web workers, geolocation, WebSocket
- Elemek: `<head>`, `<body>`, `<script>`, `<style>`, `<link>`, `<meta>`, `<title>`, `<a>`, `<p>`, `<h2>`, `<div>`, `<form>`, `<section>`, `<article>`, `<label>`, `<input>`, `<button>`, `<select>`, ...
- Attribútumok: `id`, `class`, `href`, `data-*`, `type`, `name`, `value`, `required`, ~~`bcolor`~~, ~~`link`~~, ...

# Webes alaptechnológiák: CSS



```
92
93 .blue2#miniheader #sb_form_go {
94 background-color: □#fff;
95 border-color: □#fff
96 }
97
98 #miniheader #sb_form_q.focus, .exp #miniheader_searchbox #sb_form_q {
99 width: 465px
100 }
101
102 #miniheader_searchbox {
103 min-width: 273px
104 }
105
106 #miniheader_searchbox #sbi_p {
107 display: none
108 }
109
110 #miniheader .relatedGroup {
111 display: none;
112 opacity: 0;
113 position: absolute;
114 top: 2px
115 }
116
117 #miniheader_searchbox {
118 width: 1px;
119 white-space: nowrap
120 }
121
122 #miniheader td {
123 padding: 14px 0 0
124 }
125
126 #miniheader .itm {
```

# Webes alaptechnológiák: CSS

- Deklaratív, mintaillesztés alapú stílusozó nyelv
- Selectorok: elem, tartalmazás, attribútum, osztály, összefűzés, pszeudo-osztály, pszeudo-elem, univerzális, testvér, következő...
- Tulajdonságok: margin, padding, border, width, height, position, display, color, background, font-size, !important
- Box model
- Blokk, inline, inline-block
- Specificitás
- Margin collapsing
- Elrendezés: float, flexbox, grid

# Webes alaptechnológiák: JavaScript

If it looks like a duck, swims like a duck, and quacks like a duck, then it probably *is* a duck.

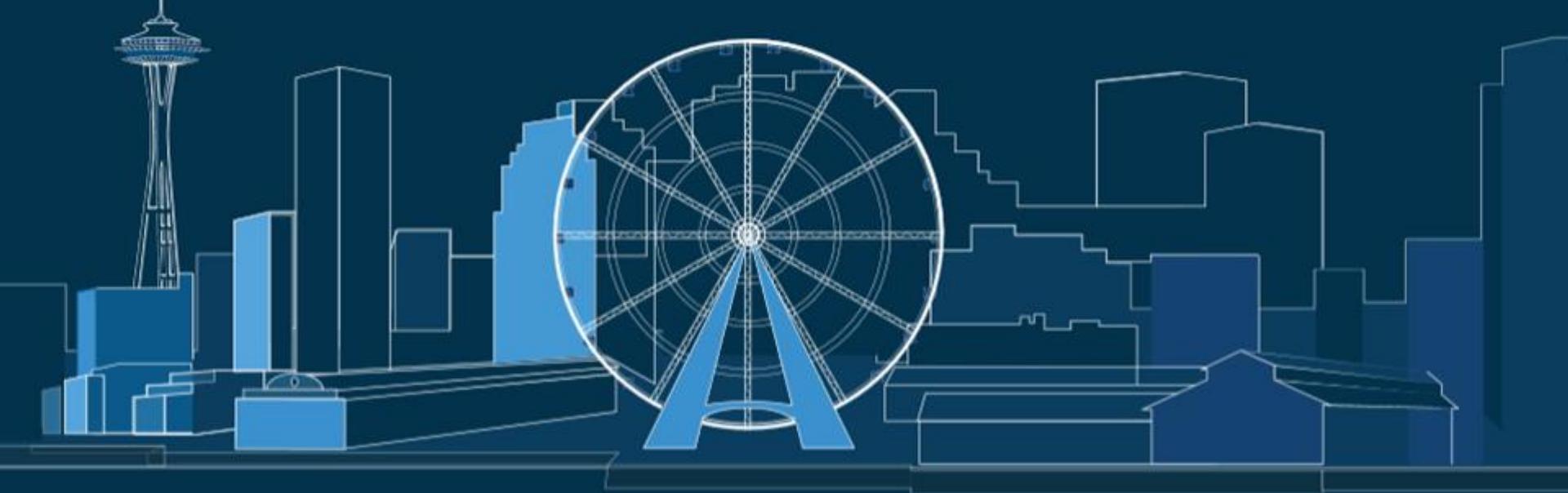


# Webes alaptechnológiák: JavaScript

- Általános szkriptnyelv, interpretáltan/JIT futtatott
- Boolean, Undefined, Number, String, Symbol, Object (Null, Function, ~~Array~~)
  - > minden objektum asszociatív tömb
- Gyengén és dinamikusan típusus (`==`, `===`), truthy/falsey értékek
- Prototípus (objektum) alapú, nem klasszikus objektum-orientált
- Futtatása böngészőben egyszálú
  - > többszálú használatra nincsen nyelvi támogatás
- Láthatóság, változódéklaráció: `var`, `let`, `const`
- DOM, `document`, `window`
- Eseménykezelés, aszinkronitás, AJAX

# Források

- <https://github.com/obonaventure/cnp3/tree/master/book-2nd>
- [https://www.slideshare.net/selvakumar\\_b1985/hftp-43534953](https://www.slideshare.net/selvakumar_b1985/hftp-43534953)



# TypeScript

JavaScript that scales.

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

Any browser. Any host. Any OS. Open source.

# Tartalom

“Történelem”

Áttekintés

TypeScript vs JavaScript

TypeScript alternatívák

Fejlesztéstámogatás

Fordítás

Statikus típusosság

Szintaxis

# “Történelem”

# “Történelem” (1)

## ECMAScript verziók

ES1	1997	
ES2	1998	
ES3	1999	Reguláris kifejezések, try/catch
ES4	-	
ES5	2009	"strict mode", getter/setter, JSON API, reflection
ES5.1	2011	
ES6 (ES2015)	2015	class, module, for/of, arrow function, Promise API, ...
ES7 (ES2016)	2016	** operátor (hatványozás), Array.prototype.includes
ES8	2017	Javaslatok: async/await, megfigyelhető folyamok, class/instance properties, operator overloading, records, tuples, ...
ES9	2018	Rest/spread tulajdonságok, aszinkron iteráció, Promise.prototype.finally()
ES10	2019	Array.flat()/.flatMap(), String.trim*(), Symbol.description, ...
ES11	2020	?? és ?. operátorok , BigInt, globalThis
ES.Next	++	Gyakorlatilag minden éven bővül a funkcionálitás

# “Történelem” (2)

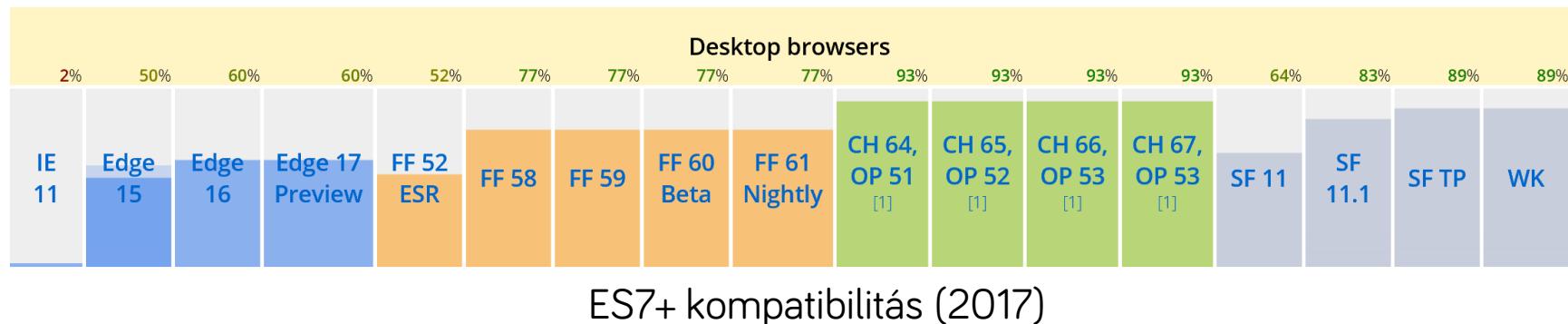
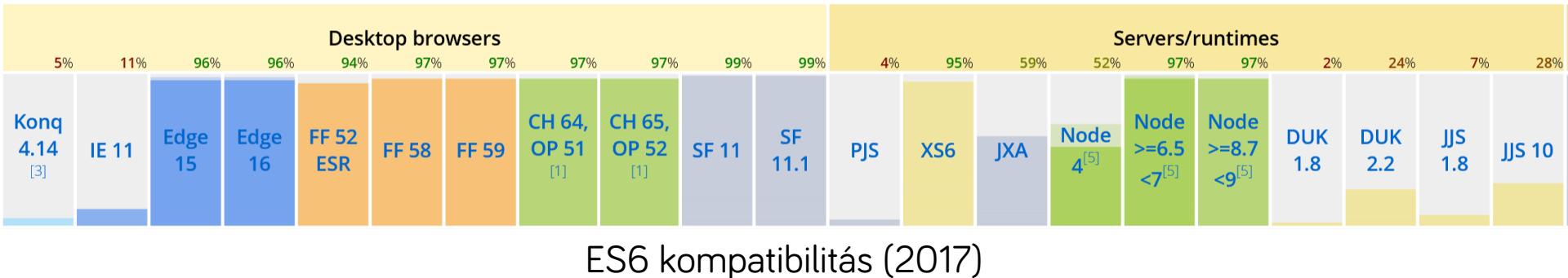
- A JavaScript implementációk túl lassan követték az ECMAScript szabványt (~2017-ig)

Szkript motor	ES5	ES6	ES7+
SpiderMonkey (Firefox)	100%	97%	77%
Chrome V8 (Chrome, Opera)	100%	97%	93%
JavaScriptCore/Nitro (Safari)	97%	99%	64%
Chakra (Edge)	100%	96%	60%

- Az egységesítési törekvések erősödtek

# “Történelem” (3)

- Nehéz a böngészőkompatibilitás támogatása
- A felmerülő problémákat eddig is megoldottuk valahogy...



# “Történelem” (4)

“You can write large programs in JavaScript.  
You just can’t maintain them.”

Anders Hejlsberg

# “Történelem” (5)



C# TypeScript

Anders Hejlsberg

# “Történelem” (6)

- A nagy JavaScript kódbázisok nehezen voltak karbantarthatók
  - Az ECMAScript szabványosítás jó irányba ment, de nem követték elég gyorsan az implementációk
- Készítsünk egy új programozási nyelvet, ami a JavaScript programozóknak szól és megoldást nyújt az egységes kezelésre!

# Áttekintés

Mire jó a TypeScript?

# Áttekintés (1)

- A TypeScript a JavaScript típusos szupersetje
  - > minden JavaScript egyben Typescript is
  - > A dinamikus működés **kiegészítése** statikus típusinformációkkal
- A TypeScript 2012-ben került kiadásra
- A Microsoft ingyenes, open source terméke
  - > <https://github.com/Microsoft/TypeScript>
  - > <https://www.typescriptlang.org/>



# Áttekintés (2)

- A JavaScript programozóknak könnyű belevágni
- Objektumorientált paradigmákat erősen követi
- JavaScriptre fordul, így mindenhol fut, ahol JavaScript is fut
  - > Konfigurálható, hogy melyik ECMAScript verzióval legyen kompatibilis a lefordult kód

# Áttekintés (3)

- Rohamosan fejlődik, gyorsan elterjedt
- Többszáz cég logója szerepel a hivatalos oldalon



<https://www.typescriptlang.org/community/friends.html>

# Áttekintés (4)

- Jövőbe tekintő fejlesztés
  - > Már most támogat funkciókat, amik még a szabványosítás 0. lépését sem érték el
  - > A TypeScript fejlesztése erősen visszahatott az ECMAScript szabványosításra
    - A TypeScript nyelvi elemek használhatók TypeScriptben, de lefordul JavaScriptre → ha a JavaScript támogatja ezeket beépítetten, kevésbé fognak divergálni egymástól

# Áttekintés (5)

- A TypeScript fordító (TypeScript Compiler – tsc) egy **source-to-source compiler**, vagy **transpiler**
- A TypeScriptet TypeScriptben írják
  - > Mivel a JavaScript egyben TypeScript is, már az első verziót is TypeScriptben írták
- A TypeScriptet tudjuk futtatni böngészőben
  - > Hogyan?
  - > A compiler JavaScriptre fordul, így böngészőben futtatható, a bemeneti Typescript fájlból JavaScriptet generál...

# Áttekintés (6)



<https://www.youtube.com/watch?v=NrdrpKE1Ls4>

# TypeScript vs JavaScript

# TypeScript vs JavaScript (1)

- Mivel tud **kevesebbet** a TypeScript?
  - > Gyakorlatilag semmivel, minden tud, amit a JavaScript is, mivel **magába foglalja** azt
  - > Viszont plusz egy karbantartási lépés a fordítás miatt (önmagában nem futtatható)

# TypeScript vs JavaScript (2)

- Mivel tud **többet** a TypeScript?
  - > A fordítási mechanizmus miatt **fordítási** idejű hibákat kapunk **futási** idejű hibák helyett
  - > A statikus típusosság miatt igazi IntelliSense lehetséges
  - > Elrejti a JavaScript “furcsaságait”
  - > Kihasználhatjuk a statikus típusellenőrzést
  - > Használhatunk még nem szabványos JavaScript elemeket, amikből fordítás után sima JavaScript lesz
  - > Jelentősen javul a kód kihasználtsága, karbantarthatósága, olvashatósága
    - Ezáltal nagy kódbázis karbantartását is lehetővé teszi

# TypeScript alternatívák

Mit használhatunk TypeScript helyett?

# TypeScript alternatívák (1)

- Google Closure Tools
  - > JavaScript-elemző – és optimalizáló eszköz, gyakori JavaScript hibákra is felhívja a figyelmet
- Dart (Google)
  - > Lehetséges JS-re fordítani, de a forrás futhat a Dartium böngészőben, önmagában (VM), vagy natívan előre fordítva (Ahead-of-Time compilation)
- Bridge.NET
  - > C# kód fordítása JavaScriptre



# TypeScript alternatívák (2)

- CoffeeScript
  - > Hasonló cél, kevésbé egyértelmű szintaxis (típusok nélkül)
- Elm
  - > Funktionális nyelv, ami JavaScripttel együtt tud működni
- Babel
  - > ECMAScript 6+ → ECMAScript 5 fordító



BABEL

# TypeScript alternatívák (3)

	CoffeeScript	JavaScript	TypeScript	Dart	ECMA Script 6
Better Structure	✓	✗	✓	✓	✓
Editor Friendly	✗	✗	✓	✓	✗
Better Speed	✗	✗	✗	✓	✗
Large Applications	✗	✗	✓	✓	✓
Brevity of Code	✓	✗	✗	✗	✗
Ease of Learning	✗	✓	✓	✗	✓
Easy Debugging	✗	✓	✓	✓	✓

[https://smthngsmwhr.wordpress.com/2013/02/25/  
javascript-and-friends-coffeescript-dart-and-typescript/](https://smthngsmwhr.wordpress.com/2013/02/25/javascript-and-friends-coffeescript-dart-and-typescript/)

# TypeScript alternatívák (4)



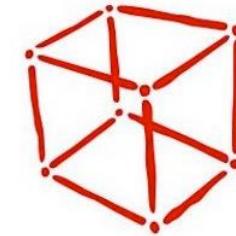
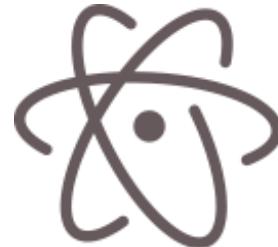
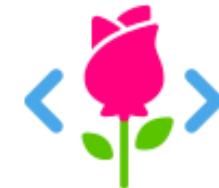
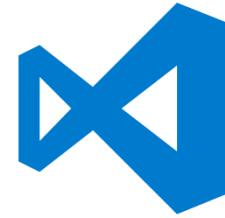
## WEBASSEMBLY

# Fejlesztéstámogatás

A fejlesztés eszközei

# Fejlesztéstámogatás (1)

- A TypeScript széles körben támogatott fejlesztőeszközök terén
- IDE támogatás:



# Fejlesztéstámogatás (2)

- Típusdefiníciók, fordító-kiegészítők, osztálykönyvtárak beszerzése
  - > NPM (NodeJS Package Manager), Yarn
- Csomagolás, disztribúció-előállítás
  - > Webpack, MSBuild, NuGet, Browserify, JSPM
- Automatizálási szkriptek
  - > Gulp, Grunt
- TSC command-line interface (CLI)

# Fordítás

Hogyan lesz a *JavaScript*?

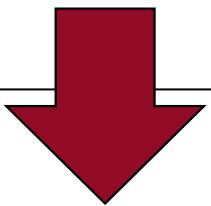
# Fordítás (1)

```
class Kutya {
 constructor(private nev: string, private ev: number) { }
 ugat() {
 console.log(`Vau! ${this.nev} vagyok, ${this.ev} éves.`);
 }
}

var vilmos = new Kutya("Vilmos", 5);
vilmos.ugat();

var Kutya = (function () {
 function Kutya(nev, ev) {
 this.nev = nev;
 this.ev = ev;
 }
 Kutya.prototype.ugat = function () {
 console.log("Vau! " + this.nev + " vagyok, " + this.ev + " \u00E9ves.");
 };
 return Kutya;
})();
var vilmos = new Kutya("Vilmos", 5);
vilmos.ugat();
```

kutya.ts



kutya.js

# Fordítás (2)

- A fordítás nagyon részletesen konfigurálható
  - > Eldönthető, hogy mit vegyünk szigorú fordítási hibának és mit nézzünk el
  - > Nem végleges funkciók engedélyezhetők
  - > Generálhatunk source map fájlokat (analóg a .NET .pdb fájlokkal, a hibakeresési szimbólumokat írja le)
  - > A kimenet egy fájlba csomagolható
  - > JavaScript forráson is futtathatjuk a fordítót, ami észreveszi a típusos hibákat
  - > ...

# Fordítás (3)

- Fordításkor kihasználjuk a JavaScript adta lehetőségeket
  - > Immediately self-invoking function
  - > Module pattern
  - > Closure
  - > Prototypal inheritance
  - > Constructor function
  - > ...

# JavaScript sajátosságok elfedése (1)

- A fordítónak megadható, hogy milyen ECMAScript verzióra fordítson (3, 5, 6, 2016, 2017, ES.Next...)
  - Ennek függvényében különböző kódot generál
- Kihasználjuk, hogy a nyelvi elemek különböző verziókon különbözőképpen valósíthatók meg
  - Nem minden nyelvi funkció használható régebbi ECMAScript fordítással

# JavaScript sajátosságok elfedése (2)

Csak típusinformáció,  
így nem jelenik meg a  
generált JS-ben!

```
my-class.ts
class MyClass {
 num: number;
}
```

ES3/ES5

```
var MyClass = (function () {
 function MyClass() {}
 return MyClass;
}());
```

Immediately self-  
invoking function

Privát closure

```
class MyClass {
}
```

ES6 class

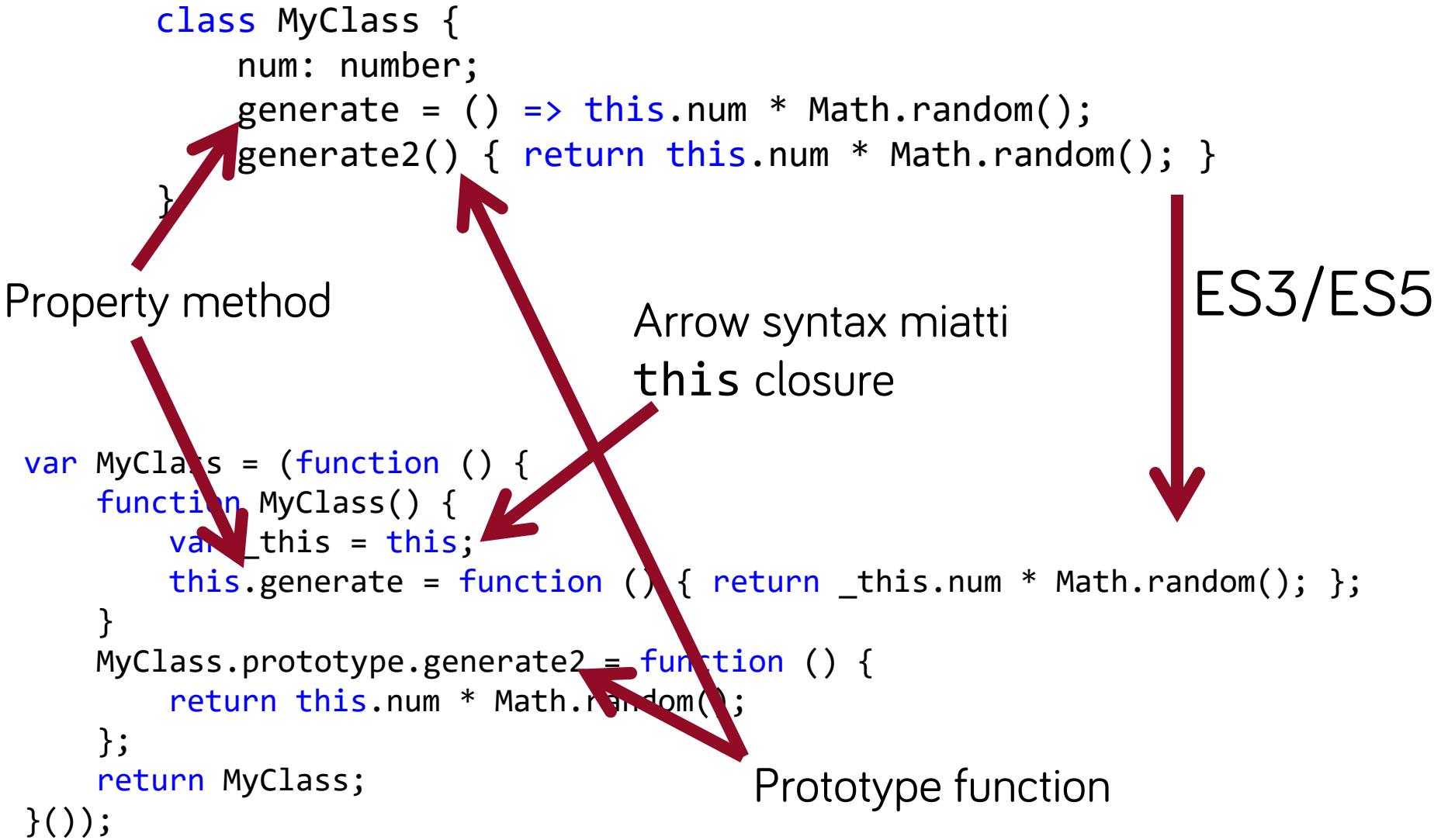
# JavaScript sajátosságok elfedése (3)

```
class MyClass {
 num: number;
 generate = () => this.num * Math.random();
 generate2() { return this.num * Math.random(); }
}
```

ES3/ES5

```
var MyClass = (function () {
 function MyClass() {
 var _this = this;
 this.generate = function () { return _this.num * Math.random(); };
 }
 MyClass.prototype.generate2 = function () {
 return this.num * Math.random();
 };
 return MyClass;
}());
```

# JavaScript sajátosságok elfedése (3)



# JavaScript sajátosságok elfedése (4)

```
class MyClass {
 num: number;
 generate = () => this.num * Math.random();
 generate2() { return this.num * Math.random(); }
}
```

Nincs szükség külön this  
closure-re, az ES6 arrow syntax  
nem hoz létre saját this-t

ES6

```
class MyClass {
 constructor() {
 this.generate = () => this.num * Math.random();
 }
 generate2() { return this.num * Math.random(); }
}
```

# JavaScript sajátosságok elfedése (5)

ES6 scoped  
variable



```
let myVar = 4;
for (let myVar in [1, 2, 3]) {
 // ...
}
```

ES3/ES5

ES6

```
var myVar = 4;
for (var myVar_1 in [1, 2, 3]) {
 // ...
}
```



Nincs **let**, változóütközés elkerülése

```
let myVar = 4;
for (let myVar in [1, 2, 3]) {
 // ...
}
```

ES6 scoped variable

# Fordítási hibák (1)

program.js:

```
function Greeter(greeting) {
 this.greeting = greeting;
 this.greet = function () {
 return "<h1>" + this.greeting + "</h1>";
 }
};
```

```
var greeter = new Greeter("Hello, world!");
document.body.innerHTML = greeter.greet;
```



Mi a hiba?

# Fordítási hibák (2)

program.ts:

```
function Greeter(greeting) {
 this.greeting = greeting;
 this.greet = function () {
 return "<h1>" + this.greeting + "</h1>";
 }
};
```

```
var greeter = new Greeter("Hello, world!");
document.body.innerHTML = greeter.greet();
```

```
[ts] Type '() => string' is not assignable to type 'string'.
(property) Element.innerHTML: string
document.body.innerHTML = greeter.greet;
```



# A fordítás eredménye

- A fordítás eredményeképp előálló JavaScript **nem tartalmazza** a TypeScript által használt típusinformációkat
  - > A lefordított JavaScript továbbra is **dinamikusan típusos**
- A meglevő JavaScript forrásainkat a hiányzó típusinformációkkal kiegészítve használhatunk külső JavaScript forrásokat is TypeScript kódban
  - > A statikus típusrendszert megtartva

# Típusdeklarációs fájlok (1)

- A fordítónak megadható, hogy a JavaScript forrás mellett a típusinformációkat is exportálja
  - > **.d.ts** kiterjesztésű fájlok
- Mivel ezek kizárálag típusinformációt hordoznak:
  - > Nem hajthatók végre
  - > Szándékosan JavaScriptre fordítva nem produkálnak kimenetet
  - > A forrás JavaScript fájlokkal együtt használhatók TypeScriptben típusellenőrzésre
- Analóg a *C/C++ header fájlokkal*

# Típusdeklarációs fájlok (2)

- Külső JavaScript forrásokat használhatunk, ha rendelkezünk a .ts fájllal
  - > TypeScriptből generálható
  - > Kézzel is elkészíthető (a legtöbbet kézzel írják)

# Típusdeklarációs fájlok (3)

kutya.ts:

```
class Kutyta {
 constructor(private name: string) {}
 ugat() { console.log(` ${name}: Woof! `) }
}
```

kutya.js

```
var Kutyta = /** @class */ (function () {
 function Kutyta(name) {
 this.name = name;
 }
 Kutyta.prototype.ugat = function () { console.log(name + ": Woof!"); };
 return Kutyta;
}());
```

kutya.d.ts

Csak típusinformáció,  
nincs futtatható kód



```
declare class Kutyta {
 private name;
 constructor(name: string);
 ugat(): void;
}
```

Csak futtatható kód,  
nincs típusinformáció

# Típusdeklarációs fájlok (4)

- Hogyan szerezzük be a típusdeklarációs fájlokat?

› Beépített támogatás van a **@types** npm csomagokra, a fordítónak megadható, mire van szükségünk:

```
// tsconfig.json:
{
 "compilerOptions": { /* ... */ },
 "typeAcquisition": {
 "enable": true,
 "include": ["jquery", "react", "lodash"]
 }
}
```

# Típusdeklarációs fájlok (5)

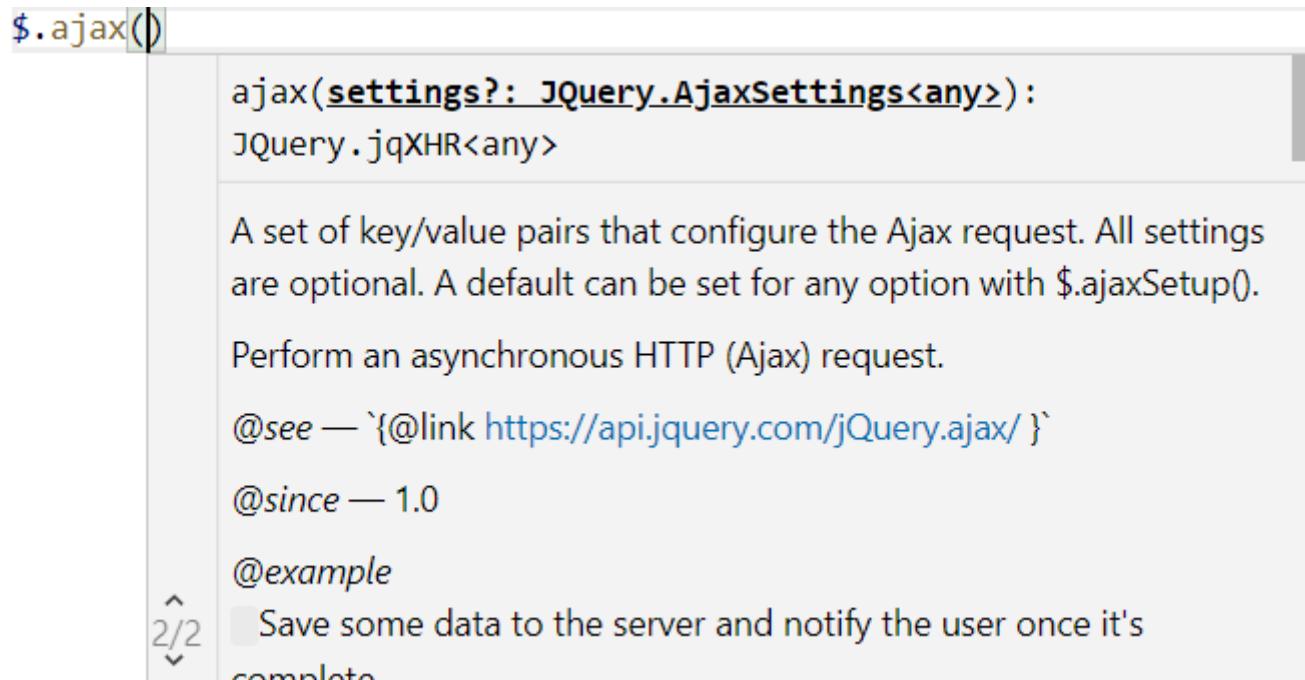
- Hogyan szerezzük be a típusdeklarációs fájlokat?
  - > Letölhetjük npm-ből a megfelelő **@types** scoped package-eket:

```
npm install --save @types/jquery
```

- > Így a fordító a node\_modules/@types/jquery mappában találja a jQuery típusdeklarációt az osztálykönyvtár használatához

# Típusdeklarációs fájlok (6)

- Így tehát a külső, JS-ben íródott (és használt) osztálykönyvtárakhoz is kapunk IntelliSense-t és fordítási idejű hibákat:



# Kliensoldali technológiák

## TypeScript

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Kliensoldali technológiák

## TypeScript

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki  
és Gazdaságtudományi Egyetem oktatója által  
kidolgozott szerzői mű.**

**Kifejezetten felhasználási engedély nélküli  
felhasználása szerzői jogi jogosításnak minősül.**

A szerző elérhetősége:  
[Szabo.Gabor@vik.bme.hu](mailto:Szabo.Gabor@vik.bme.hu)

# Tartalom

“Történelem”

Áttekintés

TypeScript vs JavaScript

TypeScript alternatívák

Fejlesztéstámogatás

Fordítás

Statikus típusosság

Szintaxis

# Statikus típusosság

# Statikus típusosság (1)

- A megszokott magasszintű nyelvekhez hasonlóan statikus típusosságra is tudunk hagyatkozni
- A típusosság lehet **explicit** (konkrétan megadjuk a típust) vagy **implicit** (a kontextusból egyértelműen következik)
- A típushibák csak fordítási/fejlesztési időben derülnek ki, futási idejű hibáink a dinamikus környezet miatt továbbra is lehetnek

# Statikus típusosság (2)

```
var num = 10;
```

```
num = 1.0;
```

```
num = "alma";
```

```
[ts] Type '"alma"' is not assignable to type 'number'.
```

```
var num: number
```

```
var num2: number;
```

```
num2 = "kutya";
```

```
[ts] Type '"kutya"' is not assignable to type 'number'.
```

```
var num2: number
```

# Statikus típusosság (3)

- Megtársíthatjuk a dinamikus típusosságot (**any** típus)

```
var something;
something = "";
something = 4;
```

- Visszatérhetünk dinamikus típusosságra

```
var num = 6;
num = <any>"kutya";
num = "kutya" as any;
```

```
var num: any = 6;
num = "kutya";
```

# Statikus típusosság (4)

- A TypeScript **any** típusa a dinamikus típust reprezentálja
  - > Az **any** bármilyen értéket felvehet és bárminek értékül adható
  - > A jó kód nem tartalmaz **any**-t
  - > Nincs IntelliSense, visszatérünk a futási időben felbukkanó típushibákhoz
  - > A fordítónak megadható, hogy változó ne lehessen implicit **any** típusú (`noImplicitAny` flag)

# Statikus típusosság (5)

- A TypeScript nagyon komplex típusrendszerrel rendelkezik
  - > Osztályok, absztrakt osztályok, interfések, öröklés
  - > Enum típusok
  - > String literálok
  - > Type inference (típusok kikövetkeztetése, strukturális típusosság)
  - > Genericitás, generikus típusok, függvények
  - > Implicit interfészmegvalósítás
  - > Unió- és metszettípusok

# Strukturális típusosság (1)

- **Strukturális típusosság:** egy A objektum a B típussal strukturálisan kompatibilis, ha A megvalósítja a B által leírt **strukturális interfész**t
- **Strukturális interfész:** egy típus publikusan elérhető tagváltozóinak, függvényeinek halmaza

# Strukturális típusosság (2)

```
interface Named {
 name: string;
}

class Person {
 id: string;
 name: string;
}

let john: Named = new Person();
```



# Strukturális típusosság (3)

```
interface Named {
 name: string;
 shortName: string;
}

class Person {
 id: string;
 name: string;
}
```



```
let john: Named = new Person();
```

[ts]

Type 'Person' is not assignable to type 'Named'.  
Property 'shortName' is missing in type 'Person'.

```
let john: Named
```

# Épség (soundness)

- A TypeScript a JavaScript sajátosságaiból adódóan nem teljesíti a programozási nyelvek épségi (soundness) elvárását
  - > Elfogad típushelytelen kódot bizonyos esetekben

```
enum EventType { Mouse, Keyboard }
```

```
interface Event { timestamp: number; }
```

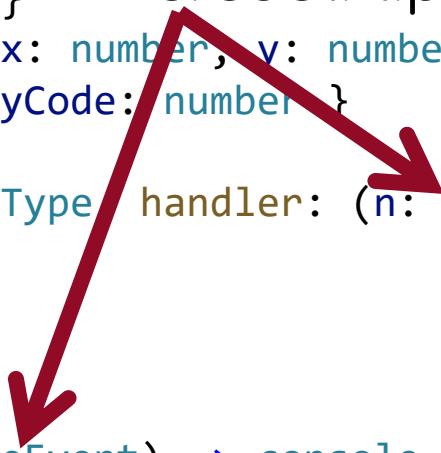
```
interface MouseEvent extends Event { x: number, y: number }
```

```
interface KeyEvent extends Event { keyCode: number }
```

```
function listenEvent(eventType: EventType, handler: (e: Event) => void) {
 /* ... */
}

// Unsound, but useful and common
listenEvent(EventType.Mouse, (e: MouseEvent) => console.log(e.x + "," + e.y));
```

Felüldefiniáltuk az eredeti típusmegkötést



# Szintaxis

Deep dive

# Változódeklarációk (1)

- A **var** kulcsszóval függvényhez (nem blokkhoz) kötött változót hozhatunk létre
  - > A külső függvény a globális névtér
- A **let** és **const** kulcsszavakkal blokkhoz kötött változókat hozhatunk létre, a **const** immutábilis
- A típust explicit megadhatjuk a változó neve után, vagy az értékkedásból egyértelműen kiderül
  - > Különben a típus **any** lesz

# Változódeklarációk (2)

```
let var1 = "Bodri"; // string típusú változó
var1 = 6; // Error: Type '6' is not
 assignable to type 'string'.
let var2: any = "Fülöp"; // explicit any típusú változó
var2 = 8; // OK
const var3; // Error: 'const' declarations
 must be initialized.
const var4 = "Kókusz"; // string típusú konstans
var4 = "Banán"; // Error: Cannot assign to 'var4'
 because it is a constant
 or a read-only property.

if (Math.random() < 0.5) {
 let var5 = "Morzsi";
 var var6 = "Puffancs";
}

console.log(var5); // Error: Cannot find name 'var5'.
console.log(var6); // OK
```

# Változódeklarációk (3)

- A fordító strictNullChecks flag-jének beállításával a null és undefined típusok nem lesznek többé részhalmazai az összes többi típusnak

```
let num: number;
function f(n: number) {
 // ...
}
f(num); // Error: Variable 'num' is used before
 // being assigned.
num = null; // Error: Type 'null' is not assignable
 // to type 'number'.
num = 6; // OK
f(num); // OK
```

# Változódeklarációk (4)

- A fordító strictNullChecks flag-jének beállításával a null és undefined típusok nem lesznek többé részhalmazai az összes többi típusnak

```
let num: number | null;
function f(n: number) {
 // ...
}
f(num); // Argument of type 'number | null' is not
 // assignable to parameter of type 'number'.
 // Type 'null' is not assignable to type 'number'
num = null; // OK
num = 6; // OK
f(num); // OK
```

# Type assertion

- “Típus bizonygatás”: nem ekvivalens a típuskonverzióval (cast), ugyanis valójában csak a compiler-nek szól, nem történik futási idejű típusmódosítás vagy -ellenőrzés
  - > A JS dinamikusságából adódik, hogy szükség van rá
  - > A futási idejű hibákat nem küszöböli ki
  - > Két ekvivalens szintaxis:

```
let someValue: any = "this is a string";
let strLength: number = (<string>someValue).length;
strLength = (someValue as string).length;
```

# Típusok áttekintése

- Támogatottak:
  - > Osztályok
  - > Interfészek (explicit- és implicit megvalósítás)
  - > Absztrakt osztályok
  - > Öröklés
  - > Láthatósági módosítószók
  - > Osztályszintű változók és függvények
  - > Enum típusok, string literálok, unió- és metszettípusok
- Nem támogatottak:
  - > Valódi metódus overloading
  - > Valódi többszörös öröklés
  - > Típusonként több konstruktur

# Enumok, string literálok

- Az Enum sorszámozása 0-tól növekszik, kivéve, ha explicit növeljük

```
enum DogKind { "Pitbull", "Terrier" = 4, "Corgi" };
```

- A string literál egy olyan típus, melyet konkrét stringek részhalmaza reprezentál

```
type DogSize = "tiny" | "small" | "medium" | "large";
```

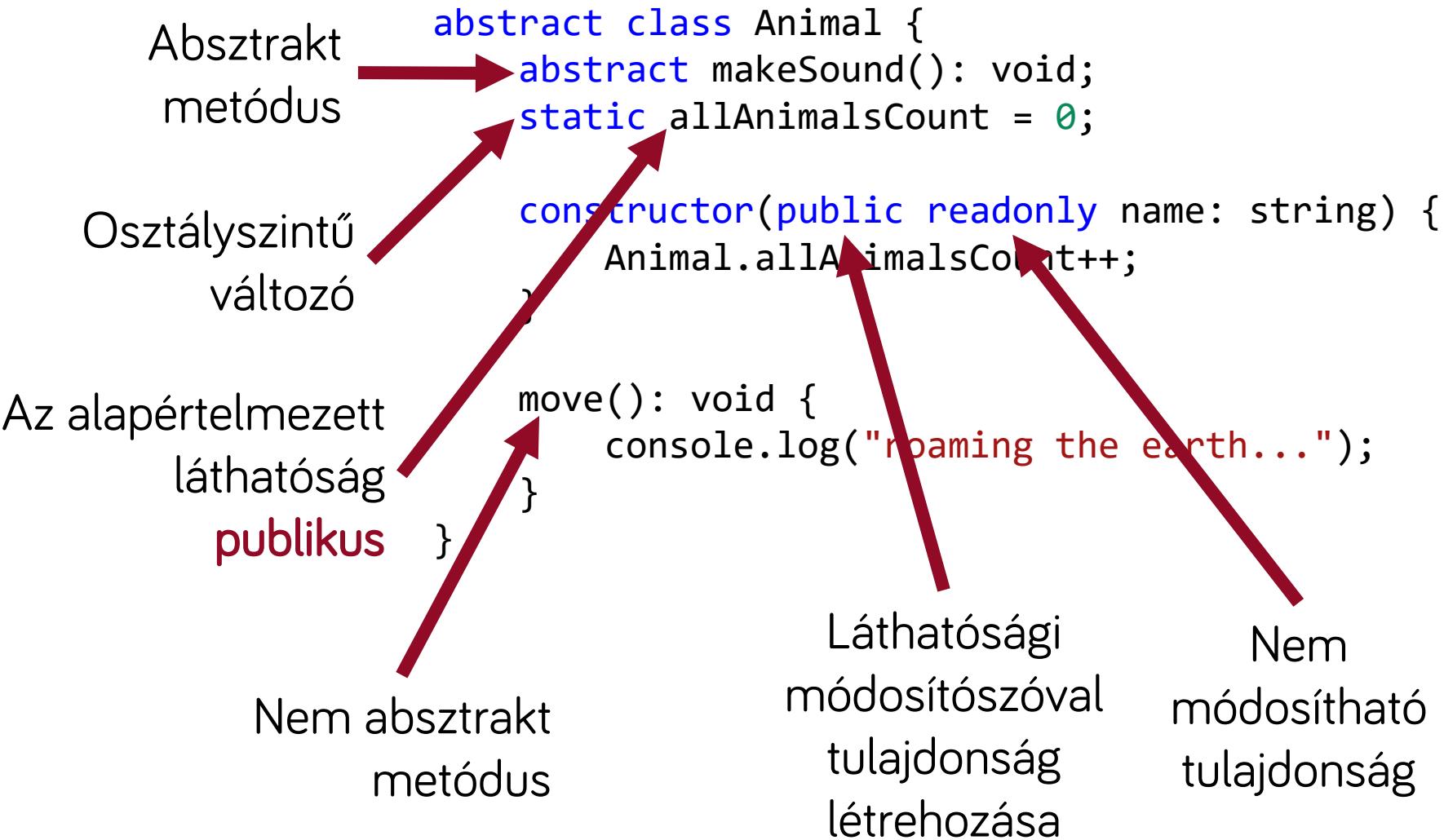
# Osztályok, öröklés, tulajdonságok (1)

```
abstract class Animal {
 abstract makeSound(): void;
 static allAnimalsCount = 0;

 constructor(public readonly name: string) {
 Animal.allAnimalsCount++;
 }

 move(): void {
 console.log("roaming the earth...");
 }
}
```

# Osztályok, öröklés, tulajdonságok (1)



# Osztályok, öröklés, tulajdonságok (2)

```
class Dog extends Animal {
 protected bones: number = 0;
 constructor(name: string, public kind: DogKind) {
 super(name);
 this.bones++;
 }
 makeSound(): void {
 console.log(`Woof! I am ${this.name}.`);
 }
}
```

# Osztályok, öröklés, tulajdonságok (2)

```
class Dog extends Animal {
 protected bones: number = 0;
 constructor(name: string, public kind: DogKind) {
 super(name);
 this.bones++;
 }
 makeSound(): void {
 console.log(`Woof! I am ${this.name}.`);
 }
}
```

Absztrakt metódus megvalósítása

Ós kötelező inicializálása a this használata előtt

Leszármazotti láthatóság

Leszármazás

String interpolation (parametrizált behelyettesítés)

# Accessors (~Java/C# property)

```
let passcode = "secret passcode";
class Employee {
 private _fullName: string;
 get fullName(): string {
 return this._fullName;
 }
 set fullName(newName: string) {
 if (passcode == "secret passcode") {
 this._fullName = newName;
 }
 else {
 console.log("Error: Unauthorized update of employee!");
 }
 }
}

let employee = new Employee();
employee.fullName = "Bob Smith";
console.log(employee.fullName);
```

# Accessors (~Java/C# property)

```
let passcode = "secret passcode";
class Employee {
 private _fullName: string;
 get fullName(): string {
 return this._fullName;
 }
 set fullName(newName: string) {
 if (passcode == "secret passcode") {
 this._fullName = newName;
 } else {
 console.log("Error: Unauthorized update of employee!");
 }
 }
}

let employee = new Employee();
employee.fullName = "Bob Smith";
console.log(employee.fullName);
```

get és set kizárolag azonos láthatósággal, azonos típussal szerepelhet, de elegendő csak az egyik

A használat látszólag megegyezik a sima mező használatával

# Generics (1)

- A korábban tanultakkal analóg a genericitás használata

```
function identity<T>(arg: T): T {
 return arg;
}
```

```
class GenericNumber<T> {
 zeroValue: T;
 add: (x: T, y: T) => T;
}
```

```
let myGenericNumber = new GenericNumber<number>();
myGenericNumber.zeroValue = 0;
myGenericNumber.add = function (x, y) { return x + y; };
```

# Generics (1)

- A korábban tanultakkal analóg a genericitás használata

```
function identity<T>(arg: T): T {
 return arg;
}
```

Típusparaméterezett,  
generikus függvény

```
class GenericNumber<T>
 zeroValue: T;
 add: (x: T, y: T) => T;
}
```

Generikus típus

```
let myGenericNumber = new GenericNumber<number>();
myGenericNumber.zeroValue = 0;
myGenericNumber.add = function (x, y) { return x + y; };
```

Generikus példány  
létrehozása

# Generics (2)

- Generikus paraméternek lehet megkötése

```
interface Lengthwise {
 length: number;
}

function loggingIdentity<T extends Lengthwise>(arg: T): T {
 console.log(arg.length);
 return arg;
}
```

Generikus megkötés

Típusbiztos kezelés

# Interfészek (1)

- Nem csak objektumpéldányok valósítanak meg interfészt, hanem
  - > Függvénypéldányok
  - > Objektum literálok (anonim objektumok),
  - > Osztályok (statikus objektum)
- Interfész definiálhat:
  - > Tagváltozót
  - > Függvényt
  - > Függvényszignatúrát
  - > Konstruktorszignatúrát
  - > Indexert

# Interfészek (2)

```
let myObj = { size: 10, label: "Size 10 Object" };

function printLabel(labelledObj: { label: string }) {
 console.log(labelledObj.label);
}

printLabel(myObj);

interface LabelledValue {
 label: string;
}

function printLabel2(labelledObj: LabelledValue) {
 console.log(labelledObj.label);
}

printLabel2(myObj)← myObj implicit megvalósítja a
LabelledValue interfészt
```

# Interfészek (3)

- Emlékeztető: a JavaScript konstruktorfüggvény analóg más nyelvek típusaival

```
class Dolog { }

let dologCtor: (new () => Dolog) = Dolog;

let dolog : Dolog = new dologCtor();
```

- A TypeScript osztály egyben saját típusának konstruktorfüggvény-példánya is
  - > Tehát minden konstruktorfüggvény valójában statikus (de nem csak TypeScriptben!)

# Interfészek (4)

```
interface ClockConstructor<T extends ClockInterface> {
 new (hour: number, minute: number): T;
}

interface ClockInterface {
 tick: TickInterface;
}

interface TickInterface {
 (): void;
}

function createClock<T extends ClockInterface>(ctor: ClockConstructor<T>,
 hour: number, minute: number): T {
 return new ctor(hour, minute);
}

class DigitalClock implements ClockInterface {
 constructor(h: number, m: number) { }
 tick() {
 console.log("beep beep");
 }
}
let digital = createClock(DigitalClock, 12, 17);
digital.tick();
```

# Interfészek (4)

```
interface ClockConstructor<T extends ClockInterface> {
 new (hour: number, minute: number): T;
}
interface ClockInterface {
 tick: TickInterface;
}
interface TickInterface {
 (): void;
}
```

Generikus konstruktor-interfész

Objektuminterfész

Függvényinterfész

```
function createClock<T extends ClockInterface>(ctor: ClockConstructor<T>,
 hour: number, minute: number): T {
 return new ctor(hour, minute);
}
class DigitalClock implements ClockInterface {
 constructor(h: number, m: number) { }
 tick() {
 console.log("beep beep");
 }
}
let digital = createClock(DigitalClock, 12, 17);
digital.tick();
```

Konstruktur példány

# Uniótípusok (1)

- Az uniótípus egy absztrakt típus, amely arra szolgál, hogy egy érték típusát adott meglevő típushalmazra szűkítsük
  - > Gyakori JavaScript minta alapján készült
  - > Valójában a string literál is egy speciális uniótípus

# Uniótípusok (2)

```
type sizes = 'small' | 'medium' | 'large';

function increaseSome(value: number, param: number | sizes) {
 if (typeof param === "number") {
 return value += param * Math.random();
 } else switch (param) {
 case 'small': return value *= 1 + Math.random() * 5;
 case 'medium': return value *= 1 + Math.random() * 10;
 case 'large': return value *= 1 + Math.random() * 50;
 default: return value;
 }
}
```

# Uniótípusok (2)

Unió típusú paraméter

```
type sizes = 'small' | 'medium' | 'large';

function increaseSome(value: number, param: number | sizes) {
 if (typeof param === "number") {
 return value += param * Math.random();
 } else switch (param) {
 case 'small': return value *= 1 + Math.random() * 5;
 case 'medium': return value *= 1 + Math.random() * 10;
 case 'large': return value *= 1 + Math.random() * 50;
 default: return value;
 }
}
```

Fordítási idejű következtetés: a típust a fordító a kifejezésfából egyértelműen kikövetkezteti (ha nem number, csak sizes lehet)

# Metszettípusok

- Hasonló az uniótípusokhoz, viszont két vagy több típus metszetét képezzük

```
function extend<T extends object, U extends object>(
 first: T, second: U): T & U {
 const result: Partial<T & U> = {};
 for (const prop in first) {
 if (first.hasOwnProperty(prop))
 (result as T)[prop] = first[prop];
 }
 for (const prop in second) {
 if (second.hasOwnProperty(prop))
 (result as U)[prop] = second[prop];
 }
 return result as T & U;
}
```

# Metszettípusok

- Hasonló az uniótípusokhoz, viszont két vagy több típus metszetét képezzük
- Metszet típusú paraméter

```
function extend<T extends object, U extends object>(
 first: T, second: U): T & U {

 const result: Partial<T & U> = {};
 for (const prop in first) {
 if (first.hasOwnProperty(prop))
 (result as T)[prop] = first[prop];
 }
 for (const prop in second) {
 if (second.hasOwnProperty(prop))
 (result as U)[prop] = second[prop];
 }
 return result as T & U;
}
```

Metszet típusú változó (opcionális tulajdonságokkal)

Ezen a ponton a visszatérés minden bemenő paraméter tulajdonságait tartalmazza

# Típusdeklarációk (1)

- Típusdeklarációt .d.ts fájl létrehozásával készíthetünk
  - > Meglevő JavaScript forráshoz
  - > Generálhatjuk TypeScript forrásból
- Nem tartalmazhat futtatható kódot
  - > Fordítási hibát eredményez
- Nincs outputja
  - > JavaScriptben nincsenek meg a TypeScript által kezelt típusinformációk

# Típusdeklarációk (2)

- JQuery – jquery.d.ts típusdeklaráció (részlet):

```
// Type definitions for jQuery 1.10.x / 2.0.x
interface JQueryStatic {
 ajax(settings: JQueryAjaxSettings): JQueryXHR;
 ajax(url: string, settings?: JQueryAjaxSettings): JQueryXHR;
 (selector: string, context?: Element|JQuery): JQuery;
 (): JQuery;
}

interface JQuery {
 addClass(className: string): JQuery;
 attr(attributeName: string): string;
 attr(attributeName: string, value: string|number|null): JQuery;
 attr(attributes: Object): JQuery;
}
declare var jQuery: JQueryStatic;
declare var $: JQueryStatic;
```

# Típusdeklarációk (3)

- Van lehetőségünk kiegészíteni meglevő típusdeklarációt
  - > pl. JQuery plugin típusinformáció JQuery objektumhoz rendelése
  - > Nem szerencsés és nehézkes, inkább modulokat használunk
- Egyszerű újradeklaráció esetén a compiler összefűzi az illeszkedő típusokat

# Modulok (1)

- A modulok az egységbezárást segítik, a logikailag összefüggő osztályok, objektumok, függvények, változók egy *logikai* fájlba helyezhetők
  - > A modulokból ezek kifelé publikálhatók (export), kívülről pedig konzumálhatók (import)
  - > Külső függőségek is ezt a mintát használják, így szeparálhatók a felelősségi körök

# Modulok (2)

```
//components.ts
export class Processor { }
export class Memory { }
export class PC {
 constructor(public memory: Memory, public cpu: Processor) { }
}
export let composePC = (memory: Memory, cpu: Processor) =>
 new PC(memory, cpu);

//office.ts
import * as Components from './components';
import PC from './components';
import { Memory, Processor } from './components';
import Rx from 'rx';

let obs = Rx.Observable.create<Components.PC>((obs) => obs.onNext(
 Components.composePC(new Memory(), new Processor())));
```

# Modulok (2)

```
//components.ts
```

```
export class Processor { }
export class Memory { }
export class PC {
 constructor(public memory: Memory, public cpu: Processor) { }
}
export let composePC = (memory: Memory, cpu: Processor) =>
 new PC(memory, cpu);
```

A modulból exportált tagok

```
//office.ts
```

```
import * as Components from './components';
import PC from './components';
import { Memory, Processor } from './components';
import Rx from 'rx';

let obs = Rx.Observable.create<Components.PC>((obs) => obs.onNext(
 Components.composePC(new Memory(), new Processor())));
```

Saját importok

Külső függőség importja

# Dekorátorok (1)

- A dekorátorok az aspektus-orientált programozás kelléke, más nyelvekben attribútumként vagy **annotációként** ismert
- Használható statikus metaadatok osztályokhoz, tulajdonságokhoz, függvényekhez rendelésére is

# Dekorátorok (2)

Dekorátorfüggvény  
factory definíciója

```
var LogCall = () => 
 (target: any, propertyKey: string) => console.log(
 `${new Date().toTimeString()}: called ${target.constructor.name}
 .${propertyKey}`);
```

```
class Logged {
 @LogCall()
 danger(...args: any[]) {
 for (var i in args)
 console.log(args[i]);
 }
}
```

Dekorátor alkalmazása

```
var logged = new Logged();
logged.danger(1, "2");
logged.danger("kutya");
```

```
//> 15:26:40 GMT+0100 (CESTime): called Logged.danger
//> 1
//> 2
//> kutya
```

# Opcionális láncolás

- A ?. („Elvis”) operátorral opcionálisan érhetünk el tagokat változókon, ha a változó nem null vagy undefined értékű (különben a visszatérés undefined)

```
if (foo && foo.bar && foo.bar.baz) {
 // ...
}
```

```
if (foo?.bar?.baz) {
 // ...
}
```

# Null összefűzés

- Null összefűzéssel (??, *null coalescing*) olyan kifejezéseket gyártunk, amelyek a jobb oldalt adják vissza, ha a bal oldal null vagy undefined

```
let x = (foo !== null && foo !== undefined) ?
```

```
 foo :
```

```
 bar();
```

```
let x = foo ? foo : bar();
```

```
let x = foo ?? bar();
```

# Elkenés

- Az elkenés (...), *spread*) segítségével könnyen bonthatunk szét és építhetünk össze objektumokat tagjaiból és tömböket más tömbökből

```
let defaults = { food: "spicy", price: "$$", ambiance: "noisy" };
let search = { ...defaults, food: "rich" };

let first = [1, 2];
let second = [3, 4];
let bothPlus = [0, ...first, ...second, 5];
```

# Egyéb konstrukciók

- Szimbólumok
- Iterátorok és generátorok
- Névterek
- Mixinek
- Segédtípusok (Partial<T, U>, NonNullable<T>, ...)
- Egyéb speciális típusok (pl. never)
- ... és még (nagyon) sok más

# Kliensoldali technológiák

## TypeScript

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Kliensoldali technológiák

## Angular

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki  
és Gazdaságtudományi Egyetem oktatója által  
kidolgozott szerzői mű.**

**Kifejezetten felhasználási engedély nélküli  
felhasználása szerzői jogi jogosításnak minősül.**

A szerző elérhetősége:  
[Szabo.Gabor@vik.bme.hu](mailto:Szabo.Gabor@vik.bme.hu)



FEATURES

DOCS

RESOURCES

EVENTS

BLOG

Search



One framework.  
Mobile & desktop.

[GET STARTED](#)

## DEVELOP ACROSS ALL PLATFORMS

Learn one way to build applications with Angular and reuse your code and abilities to build apps for any deployment target. For web, mobile web, native mobile and native desktop.

# Tartalom

Áttekintés  
Architektúra  
Metaadatok  
Komponensek  
Modulok  
Direktívák  
Adatkötés  
Szolgáltatások  
Életciklus  
Pipes  
Auszinkronitás  
Routing  
Ürlapok

# Áttekintés

Angular 1-0-1

# Probléma (1)

- 2009-et írunk (2 évvel az iPhone után)
- Elkezdődik az átállás az okos kliens alkalmazásokról (*Silverlight*, *Flash*) vékonykliensekre
- A legelterjedtebb kliensoldali keretrendszer a **JQuery**, ami DOM-manipulációra lett kifejlesztve
  - > A DOM-manipuláció lassú
  - > Az adatot nem illik a DOM-ban tárolni

```
jQuery(document).ready(function($){

 // Prevent the enter key from submitting the form.
 $(".ninja-forms-form input").bind("keypress", function(e) {
 if (e.keyCode == 13) {
 var type = $(this).attr("type");
 if(type != "textarea"){
 // return false;
 }
 }
 });

 /* * * Begin Mask JS * * */

 jQuery("div.label-inside input, div.label-inside textarea").focus(function(){
 var label = jQuery("#" + this.id + "_label_hidden").val();
 if(this.value == label){
 this.value = '';
 }
 });

 jQuery("div.label-inside input, div.label-inside textarea").blur(function(){
 var label = jQuery("#" + this.id + "_label_hidden").val();
 if(this.value == ''){
 this.value = label;
 }
 });

 if(jQuery.fn.mask){
 jQuery(".ninja-forms-mask").each(function(){
 var mask = $(this).data('mask');
 mask = mask.toString();
 $(this).mask(mask);
 });

 var date_format_mask = ninja_forms_settings.date_format;
 date_format_mask = date_format_mask.replace(/m/g, 9);
 date_format_mask = date_format_mask.replace(/d/g, 9);
 date_format_mask = date_format_mask.replace(/y/g, 99);
 date_format_mask = date_format_mask.replace(/\Y/g, 0000);
 }
});
```

# Probléma (2)

- A JavaScriptben írt alkalmazások kódázása jelentősen nőni kezdett
- Nem voltak elterjedt, egységes konvenciók az alkalmazásfejlesztés területén
- A vastagkliens- és szerveralkalmazásokban viszont kellően komplex architekturális mintákat alkalmaztak
  - > MVVM, MVP, MVC

# Probléma (3)

- Az alkalmazás növekedésével lehetetlen volt a JavaScript dinamikussága és a DOM-manipuláció miatt látni egy apró módosítás hatását az alkalmazásra

# Megoldás

- 2010, a Google megalkotja az AngularJS alkalmazásfejlesztő keretrendszerét
  - > Elsősorban (de nem kizárálag) SPA alkalmazások fejlesztésére
  - > Voltak korábban próbálkozások, de a kisebb keretrendszerek sem terjedtek el

# Alapelvek

- Teljes értékű keretrendszer
- Open-source
- Deklaratív UI leírás
- Imperatív üzleti logikai leírás
- DOM-manipulációs logika leválasztása az alkalmazáslogikától
- Kliens- és szerveroldali fejlesztések párhuzamosítása
- Separation of concerns
- Dependency Injection

# Újabb probléma

- 2014, a legelterjedtebb vékony kliens keretrendszer az AngularJS
- Az idő közben bekövetkezett paradigmaváltások és az AngularJS néhány alapkoncepciója kérdésessé teszi a keretrendszer jövőjét

# Újabb megoldás

- 2016, megjelenik az Angular 2.0
  - > Teljes újraírás, nem visszafelé kompatibilis
  - > Paradigmaváltások
    - Nincs többé \$scope
    - Egyszerűbb templating
    - Egyértelműbb architektúra
    - TypeScript alapon
    - Kizárolag modern böngészőket támogat (IE9+)
    - Reaktív felületek készítésének lehetősége RxJS-sel

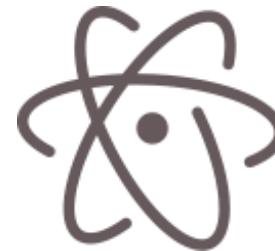
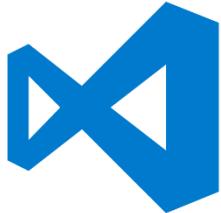
# Elnevezések

- 1.0: AngularJS
- 2.0+: Angular
  - > A 3.0-as verziót a Router komponens verzióugrása miatt kihagyták
  - > Ütemezett kiadási ciklus:
    - Kb. 6 havonta új főverzió
    - 1-3 alverzió főverzióenként
    - Nagyjából heti rendszerességű bugfix kiadás

# Big picture

- A jQuery alapú AngularJS hosszútávú támogatásának vége: 2021. június 30.
- Az AngularJS-t hetente kb. 500e-szer, az @angular/core csomagot kb. 1-3 M-szor töltik le hetente
  - > A React library-t kb. 10M-szor

# Fejlesztőeszközök



...

- npm: csomagkezelés
- @angular/cli: **ng** parancssori eszköz
  - > Kódgenerálás
  - > Kódoptimalizálás (bundling, minification)
  - > Hibakeresés lokális fejlesztői szerveren

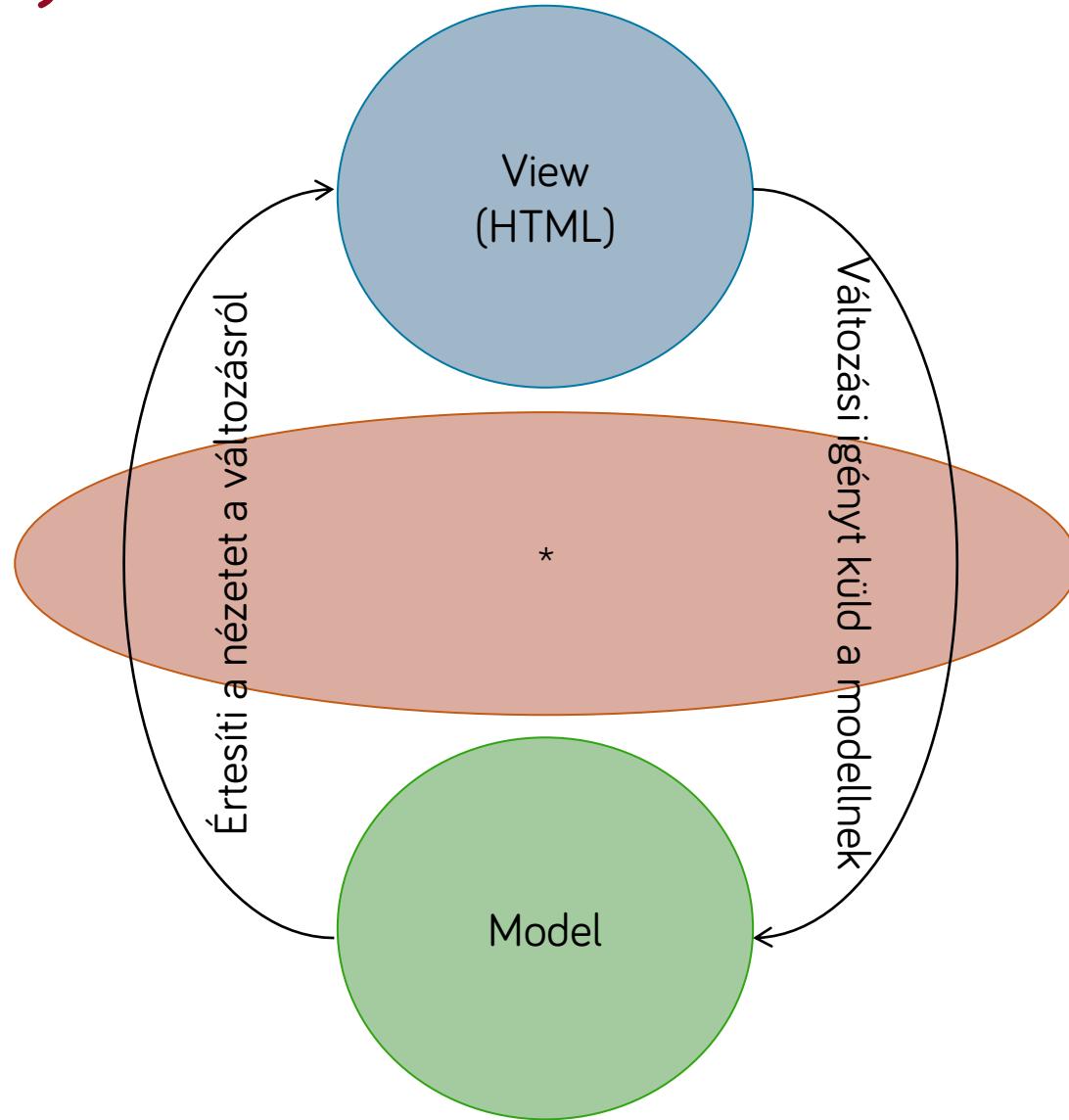
# Architektúra

MV\*

# MV<sup>\*</sup> (1)

- Model
- View
- \*
- > Controller
- > ViewModel
- > Presenter
- > ...

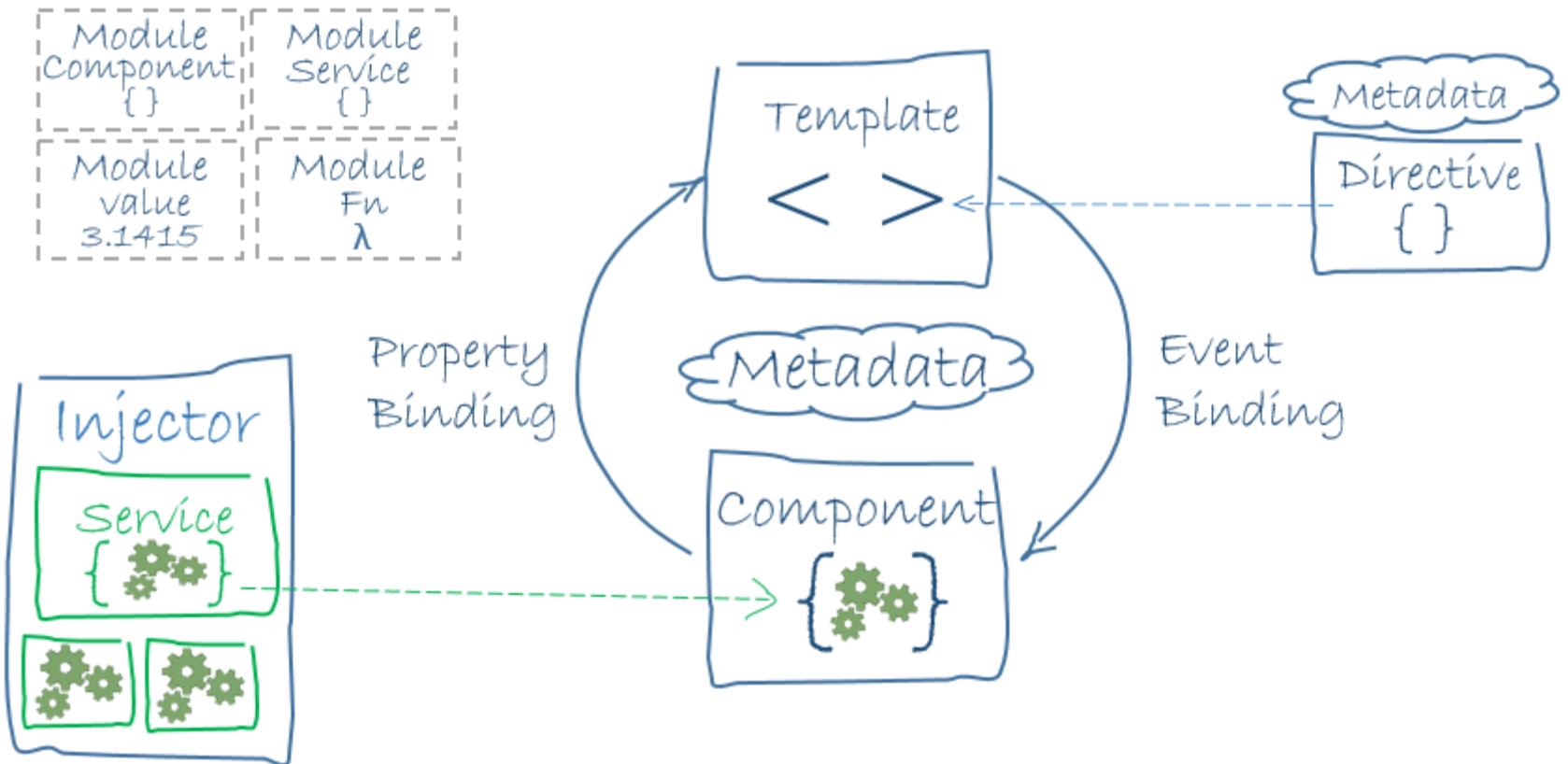
# MV<sup>\*</sup> (2)



# MV\* (3)

- Lehetővé teszi az alkalmazáslogika, a nézet összeállításának és a kettő közötti kommunikációnak a szeparációját
- Az egyes (részben automatizált) kommunikációs lépéseket a Model és a View között minden irányban adatkötésnek nevezzük

# Angular architektúra



<https://angular.io/guide/architecture>

# Metaadatok

# Metaadatok

- Az alkalmazás egyes részeinek plusz információval kell szolgálniuk az Angular felé, ami egyszerű típusinformációból nem következik
  - > Ezeket metaadatként adjuk meg
  - > Jelenleg (még) nem támogatott JavaScriptben a reflexió, ezért szükséges alternatív megoldás
- A TypeScript dekorátorokkal plusz információt rendelhetünk:
  - > Egy osztályhoz
  - > Egy függvényhez vagy konstruktorthoz
  - > Egy függvényparaméterhez
  - > Egy tulajdonsághoz vagy accessorhoz

# Angular dekorátorok

- Az Angular alkalmazásban gyakran használunk dekorátorokat, amik így plusz információt (AOP) hordoznak:
  - > @NgModule
  - > @Component, @Directive
  - > @Input, @Output
  - > @Inject, @Injectable

# Komponensek

# Adatkötés a felületen

- A felület leírását deklaratíván, HTML-szerű sablonokban adjuk meg
- Az adatkötéseket a felület deklarációjakor tudjuk elhelyezni a nézetet reprezentáló sablonban
- A sablon egy komponenshez tartozik, ami egy nézetet és annak állapotát, UI kezelő logikáját tartalmazza

# Komponensek (1)

- A komponensek (Component) jól körülhatárolt funkciót valósítanak meg az alkalmazásban
- minden komponenshez tartozik egy nézet (template)
- A komponens **felhasználja** az alkalmazás modelljét (üzleti logika) **szolgáltatások** formájában, ez **nem** a komponensben van megírva

# Komponensek (2)

- A felhasználói felület egy részletéért felelős
- Újrahasznosítható
- Csak a saját felelősségi körén belül módosít
- Bemeneti adatokat vár, kimenő eseményeket publikál (@Input, @Output)
- Hivatkozhat más komponensekre saját template-jében
  - Ezáltal egy komponens-hierarchia épül fel

# Hello Angular alkalmazás - Component

```
import { Component } from '@angular/core';

@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
export class MyComponent {
 title = 'Angular';
}
```

# Hello Angular alkalmazás - Component

```
import { Component } from '@angular/core';

@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
export class MyComponent {
 title = 'Angular';
}
```

A saját komponens osztályunk állapottárolásra (és eseménykezelésre)

A Component dekorátor, amit az Angular definiál

A komponens dekorátorral metaadat kötése a komponensünkhez

A CSS selector, amire a komponensünk példányosodik

A komponensünk HTML template-je adatkötéssel

# Modulok

# Angular Module / NgModule

- Egy alkalmazás-réshalmaz, az összefüggő alkalmazásrészek összefogására szolgál
- Nem összekeverendő a module loader-ek és különböző modulrendszerök moduljaival
  - > (SystemJS, CommonJS, UMD, AMD, ES6 Modules)
  - > Webpack module

# Angular modulok

- A keretrendszer maga is moduláris, mindenkor csak azokat a modulokat használjuk, amire szükségünk van:
  - > Core
  - > Common
  - > Forms
  - > Http
  - > Browser/Server
  - > Router
  - > Compiler
  - > ...

# Angular osztálykönyvtárak

- A külső Angular alapú függőségeket modulokba szervezik, ezeket a saját modulunk függőségként definiálja

Library Module		
Component	Directive	
{ }	{ }	
Service	value	Fn
{ }	3.1415	λ

# Hello Angular alkalmazás - NgModule

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { HelloComponent } from './hello.component';

@NgModule({
 imports: [BrowserModule],
 declarations: [HelloComponent],
 bootstrap: [HelloComponent]
})
export class HelloModule { }
```

# Hello Angular alkalmazás - NgModule

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { HelloComponent } from './hello.component';

@NgModule({
 imports: [BrowserModule],
 declarations: [HelloComponent],
 bootstrap: [HelloComponent]
})
export class HelloModule { }
```

Maga az NgModule,  
amit az Angular definiál

Böngészőben történő  
futtatást tesz lehetővé

A saját komponensünk

A saját alkalmazás-modulunk

A dekorátorral metaadatokat kötünk a modulhoz:  
függőségek, deklarációk, belépési pont

# Hello Angular – bootstrapping (1)

- Az alkalmazás indításához szükséges

```
import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';
import { HelloModule } from './app/hello.module';

platformBrowserDynamic().bootstrapModule(HelloModule);
```

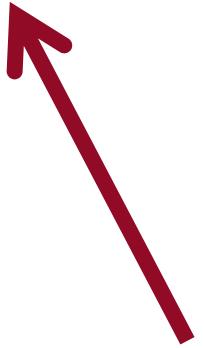


A HelloModule által definiált alkalmazás indítása a böngészőben

# Hello Angular – bootstrapping (2)

```
<!DOCTYPE html>
<html>
 <head>
 <!-- ... -->
 </head>

 <body>
 <hello-component>Loading...</hello-component>
 </body>
</html>
```



Az Angular indulásáig ez egy sima HTML elem, ezért a Loading... felirat jelenik meg a felületen

# Hello Angular – bootstrapping (3)

1. Az Angular függőségeinek betöltése
  1. Polyfill a böngészőkompatibilitás miatt
  2. ZoneJS a végrehajtási kontextusok szeparációja végett
  3. RxJS aszinkron eseménykezeléshez
2. Alkalmazás belépési pontjának betöltése és indítása
3. Az Angular átveszi az alkalmazásindítási folyamatot

# Hello Angular!

```
<body>
 <hello-component>Loading...</hello-component>
</body>
```

```
@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
export class HelloComponent {
 title = 'Angular';
}
```

# Hello Angular!

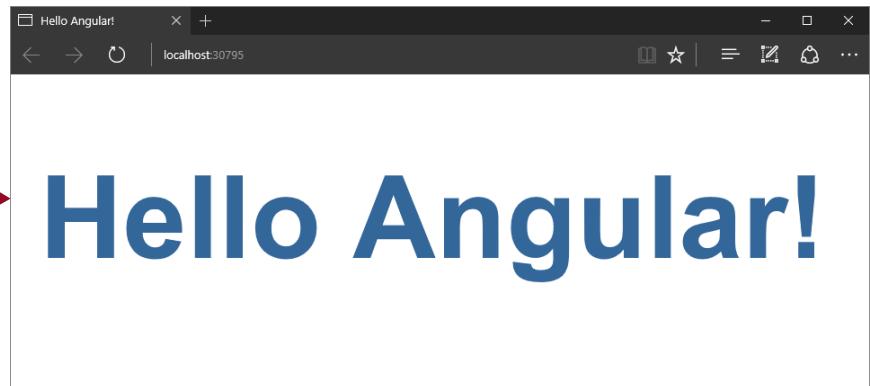
```
<body>
 <hello-component>Loading...</hello-component>
</body>
```

```
@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
export class HelloComponent {
 title = 'Angular';
}
```

1. Az alkalmazás indulása után a bootstrappelt komponens illesztése

2. A kezdeti adatkötések végrehajtása

3. Renderelés a DOM-ba



# Direktívák

# Direktívák

- Egy direktíva:
  - > Adott DOM elemhez kiegészítő működést vagy megjelenést rendel (attribútumdirektíva)
  - > Megváltoztathatja a DOM struktúráját (strukturális direktíva)
- Eseménykezelőket regisztrálhatunk a DOM elemre
- Adatkötéshez használhat **@Input** paramétereket és publikálhat **@Output** eseményeket
- Beépített direktívák segítenek a template tisztán tartásában

# Attribútumdirektívák

- Az attribútumdirektíva nem változtatja meg az elemet a DOM-ban, viszont további, kiegészítő működést rendel hozzá
- Beépített attribútumdirektívák:
  - > **[ngStyle]**: stílus alkalmazása egy elemre
  - > **[ngClass]**: osztályok alkalmazása egy elemre\*
  - > **[(ngModel)]**: modell kétirányú kötése egy (input) elemhez

# Strukturális direktívák

- Olyan direktíva, ami a DOM-ot módosítja
- A \* mikroszintaxis előzi meg az attribútumot
- Beépített strukturális direktívák:
  - > **\*ngIf**: megjeleníti az elemet a DOM-ban, ha a megadott feltétel teljesül
  - > **\*ngSwitchCase**, **\*ngSwitchDefault** : adott kifejezés értékének vizsgálata, és a megfelelő feltétel esetén az adott template renderelése a DOM-ba
    - Az **[ngSwitch]** nem strukturális, mert csak egy logikai egységet alkot a case-ek és default körül
  - > **\*ngFor**: iterál egy adott tömbön, és a megadott DOM-elemet ismétli meg (adatkötésekkel)

# Strukturális direktíva - \*ngIf

```
@Component({
 selector: 'hello-component',
 template: `
 <h1 *ngIf="title === '...'">loading</h1>
 <h1 *ngIf="title !== '...'">Hello {{title}}!</h1>
 `

})
export class HelloComponent {
 title: string = "...";
 constructor() {
 setInterval(() => this.title = Math.random() < 0.5
 ? "..." : "Angular", 1000);
 }
}
```

# Strukturális direktíva - \*ngIf

```
@Component({
 selector: 'hello-component',
 template:
 `<h1 *ngIf="title === '...'">loading</h1>
 <h1 *ngIf="title !== '...'">Hello {{title}}!</h1>
`
})
export class HelloComponent {
 title: string = "...";
 constructor() {
 setInterval(() => this.title = Math.random() < 0.5
 ? "..." : "Angular", 1000);
 }
}
```

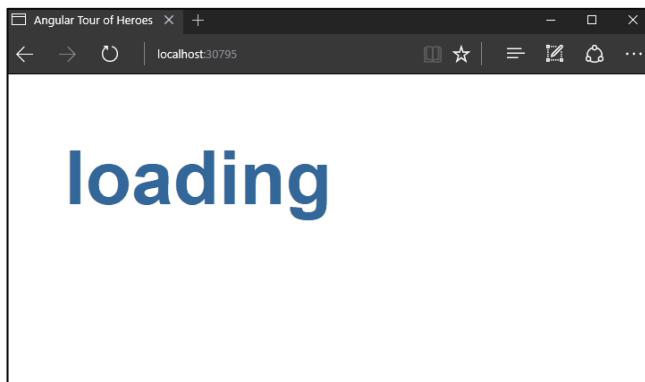
Az \*ngIf teljesül, ha a title értéke **"..."**

Az \*ngIf teljesül, ha a title értéke **nem** **"..."**

Másodpercenként véletlenszerűen beállítjuk az adatkötött title értéket

# Strukturális direktíva - \*ngIf

```
@Component({
 selector: 'hello-component',
 template:
 `<h1 *ngIf="title === '...'">loading</h1>
 <h1 *ngIf="title !== '...'">Hello {{title}}!</h1>
`
})
export class HelloComponent {
 title: string = "...";
 constructor() {
 setInterval(() => this.title = Math.random() < 0.5
 ? "..." : "Angular", 1000);
 }
}
```



Az \*ngIf teljesül, ha a title értéke **"..."**

Az \*ngIf teljesül, ha a title értéke **nem** **"..."**

Másodpercenként véletlenszerűen beállítjuk az adatkötött title értéket

# Komponens ≥ direktíva

- A komponens egy speciális direktíva
  - > Van saját template-je, amit kirajzol
  - > Frissíti a template-ben levő adatkötést
  - > Kezeli a felületről érkező eseményeket

# Direktívák közti kommunikáció

- Direktívák (és komponensek) @Input és @Output paraméterekkel kommunikálhatnak egymással az adatkötésen keresztül
- Használhatnak megosztott szolgáltatásokat

# Saját attribútum direktíva

```
import { Directive, ElementRef, Input, HostListener }
 from '@angular/core';

@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
 constructor(public el: ElementRef) { }

 @HostListener('mouseenter') onMouseEnter() {
 this.highlight('yellow');
 }

 @HostListener('mouseleave') onMouseLeave() {
 this.highlight(null);
 }

 private highlight(color: string) {
 this.el.nativeElement.style.backgroundColor = color;
 }
}
```

# Saját attribútum direktíva

```
import { Directive, ElementRef, Input, HostListener }
 from '@angular/core';
```

```
@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
 constructor(public el: ElementRef) {}
```

A direktíva metaadatai  
és selectora

```
@HostListener('mouseenter') onMouseEnter() {
 this.highlight('yellow');
}
```

A DOM elem  
referenciája, amire a  
direktívát tettük

```
@HostListener('mouseleave') onMouseLeave() {
 this.highlight(null);
}
```

Eseménykezelők regisztrációja  
a @HostListener dekorátorral

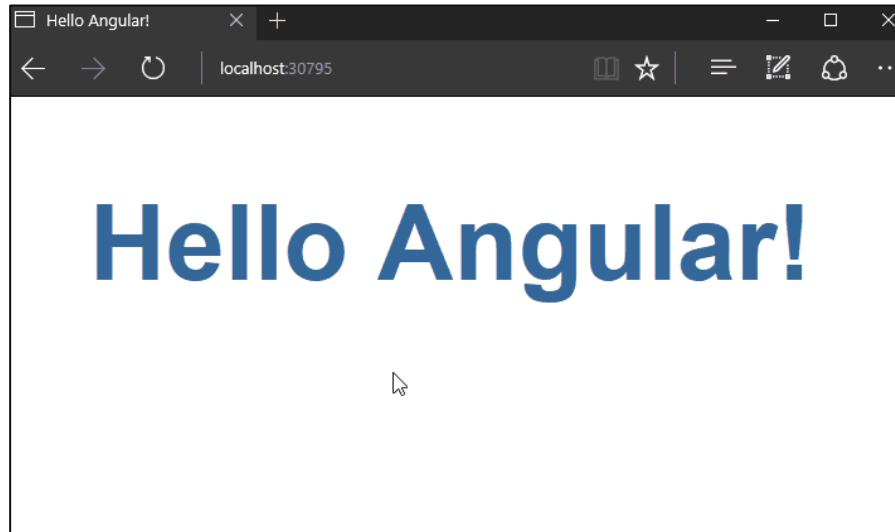
```
private highlight(color: string) {
 this.el.nativeElement.style.backgroundColor = color;
}
```

# Saját attribútum direktíva

```
@Component({
 selector: 'hello-component',
 template: `<h1 myHighlight>Hello {{title}}!</h1>`
})
export class HelloComponent {
 title = 'Angular';
}
```

# Saját attribútum direktíva

```
@Component({
 selector: 'hello-component',
 template: `<h1 myHighlight>Hello {{title}}!</h1>`
})
export class HelloComponent {
 title = 'Angular';
}
```



# Adatkötés

# Adatkötés szintaxis

- Az adatkötést többféle módon érhetjük el a template-ben:
  - > `<elem>{{data}}</elem>`: ún. “mustache” vagy “handlebar” szintaxis, egyirányú ( $M \rightarrow V$ ) adatkötés
  - > `<elem [directive]="data">` (direktíva) @Input adatkötés ( $M_1 \rightarrow V \rightarrow M_2$ )
    - Titokban ekvivalens az `<elem directive="{{data}}>` szintaxissal
  - > `<elem (click)="handler">` eseménykezelő kötése ( $V \rightarrow M$ )
  - > `<elem [(ngModel)]="data">` kétirányú adatkötés ( $M \leftarrow \rightarrow V$ ) az ngModel direktíván keresztül

`[ ( ) ] = BANANA IN A BOX`

Visualize a *banana in a box* to remember that the parentheses go *inside* the brackets.

# Eseménykezelő kötése

- Eseménykezelőt az (event)="handler" szyntaxssal tudunk a felülethez kötni
  - > Az eseménykezelő kiértékelődik a komponens kontextusában, így lehet függvényhívás vagy egyszerű JavaScript kód

```
<input type="radio" name="colors" (click)="color='lightgreen'">Green
<input type="radio" name="colors" (click)="color='yellow'">Yellow
<input type="radio" name="colors" (click)="setColor('cyan')">Cyan
```

# Kanonikus alak

- A kötéseknek létezik egy alternatív, ekvivalens, HTML-kompatibilis alakja is:
  - > `<elem [directive]="data">` helyett  
`<elem bind-directive="data">`
  - > `<elem (click)="handler">` helyett  
`<elem on-click="handler">`
  - > `<input [(ngModel)]="data">` helyett  
`<input bindon-ngModel="data">`

# Adatkötés @Input dekorátorral (1)

- Komponenseken (direktívákon) a mezők vagy tulajdonságok @Input dekorátorral történő ellátása engedélyezi az adott mezőn az adatkötést
- A komponens szülője így dinamikusan paramétert tud átadni a template-ben

# Adatkötés @Input dekorátorral (2)

```
@Component({
 selector: 'hello-component',
 template: `<div>
 <input type="radio" name="colors"
 (click)="color='lightgreen'">Green
 <input type="radio" name="colors"
 (click)="color='yellow'">Yellow
 </div>
 <show-value [value]="color"></show-value>`})
export class HelloComponent {
 color: string;
}
@Component({
 selector: 'show-value',
 template: `<h1>{{value}}</h1>`
})
export class ShowValueComponent {
 @Input()
 value: string;
}
```

Egyirányú adatkötés  
(V → M)

The diagram illustrates a one-way data binding between two components. A red arrow points from the 'value' input property in the HelloComponent's template to the 'value' attribute in the ShowValueComponent's class definition. This visualizes how the component's state is passed to the external component.

# Adatkötés @Input dekorátorral (3)

```
@Component({
 selector: 'hello-component',
 template: `<div>
 <input type="radio" name="colors"
 (click)="color='lightgreen'">Green
 <input type="radio" name="colors"
 (click)="color='yellow'">Yellow
 <input type="radio" name="colors"
 (click)="setColor('cyan')">Cyan
 </div>
 <h1 [myHighlight]="color">Highlight me!</h1>
`)
export class HelloComponent {
 color: string;
 setColor(color: string) {
 this.color = color;
 }
}
```

# Adatkötés @Input dekorátorral (3)

```
@Component({
 selector: 'hello-component',
 template: `<div>
 <input type="radio" name="colors"
 (click)="color='lightgreen'">Green
 <input type="radio" name="colors"
 (click)="color='yellow'">Yellow
 <input type="radio" name="colors"
 (click)="setColor('cyan')">Cyan
 </div>
 <h1 [myHighlight]="color">Highlight me!</h1>
})
export class HelloComponent {
 color: string;
 setColor(color: string) {
 this.color = color;
 }
}
```

Egyirányú adatkötés  
(V → M)

Egyirányú adatkötés  
(M → V)

# Adatkötés @Input dekorátorral (4)

```
@Directive({ selector: '[myHighlight]' })
export class HighlightDirective {
 constructor(public el: ElementRef) { }

 @Input('myHighlight') highlightColor: string;

 @HostListener('mouseenter') onMouseEnter() {
 this.highlight(this.highlightColor);
 }

 @HostListener('mouseleave') onMouseLeave() {
 this.highlight(null);
 }

 private highlight(color: string) {
 this.el.nativeElement.style.backgroundColor = color;
 }
}
```

@Input dekorátor  
opcionális  
megnevezéssel

# Adatkötés @Input dekorátorral (5)



Green  Yellow  Cyan

**Highlight me!**

# Adatkötés @Output dekorátorral (1)

- Az @Output dekorátorral publikálható eseményeket tudunk kifelé elérhetővé tenni

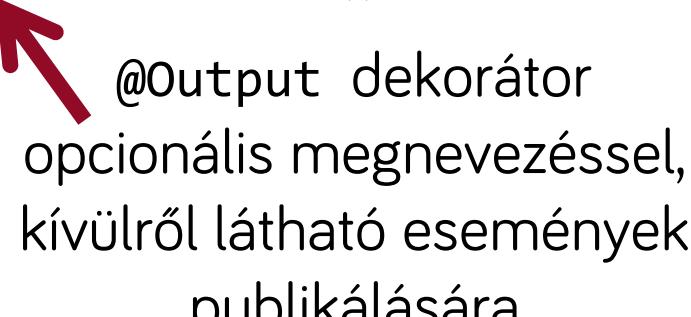
```
@Directive({ selector: '[myClick]' })
export class MyClickDirective {
 @Output('myClick') clicks = new EventEmitter<void>();
 @HostListener('click') onClick() {
 this.clicks.emit();
 }
}
@Component({
 selector: 'hello-component',
 template: `<h1 (myClick)="clicks=clicks+1">
 Clicked {{clicks}} times!
 </h1>`
})
export class HelloComponent {
 clicks = 0;
}
```

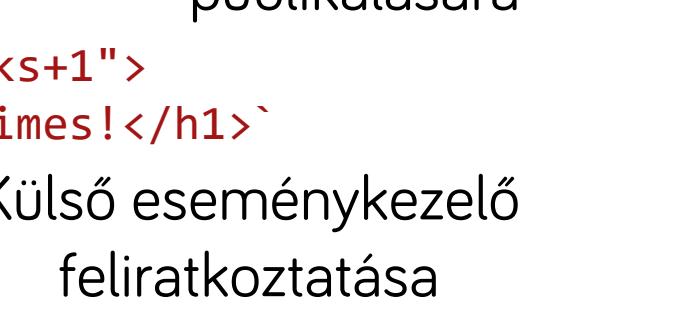
# Adatkötés @Output dekorátorral (1)

- Az @Output dekorátorral publikálható eseményeket tudunk kifelé elérhetővé tenni

```
@Directive({ selector: '[myClick]' })
export class MyClickDirective {
 @Output('myClick') clicks = new EventEmitter<void>();
 @HostListener('click') onClick() {
 this.clicks.emit();
 }
}
@Component({
 selector: 'hello-component',
 template: `<h1 (myClick)="clicks=clicks+1">
 Clicked {{clicks}} times!
 </h1>`
})
export class HelloComponent {
 clicks = 0;
}
```









# Adatkötés @Output dekorátorral (2)

- A template-ből elérhető \$event objektumot átadhatjuk paraméterül az eseménykezelőnek
  - > Nem javasolt, túlságosan erős kötést implikál a nézet és a modell között

```
@Component({
 selector: 'hello-component',
 template: `<h1 (myClick)="myClickHandler($event)">
 Clicked {{clicks}} times!
 </h1>`
})
export class HelloComponent {
 clicks = 0;
 myClickHandler($event: MouseEvent) { this.clicks++; }
}
```

# Kliensoldali technológiák

## Angular

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Kliensoldali technológiák

## Angular

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki  
és Gazdaságtudományi Egyetem oktatója által  
kidolgozott szerzői mű.**

**Kifejezetten felhasználási engedély nélküli  
felhasználása szerzői jogi jogosításnak minősül.**

A szerző elérhetősége:  
[Szabo.Gabor@vik.bme.hu](mailto:Szabo.Gabor@vik.bme.hu)



One framework.  
Mobile & desktop.

<https://angular.io/>

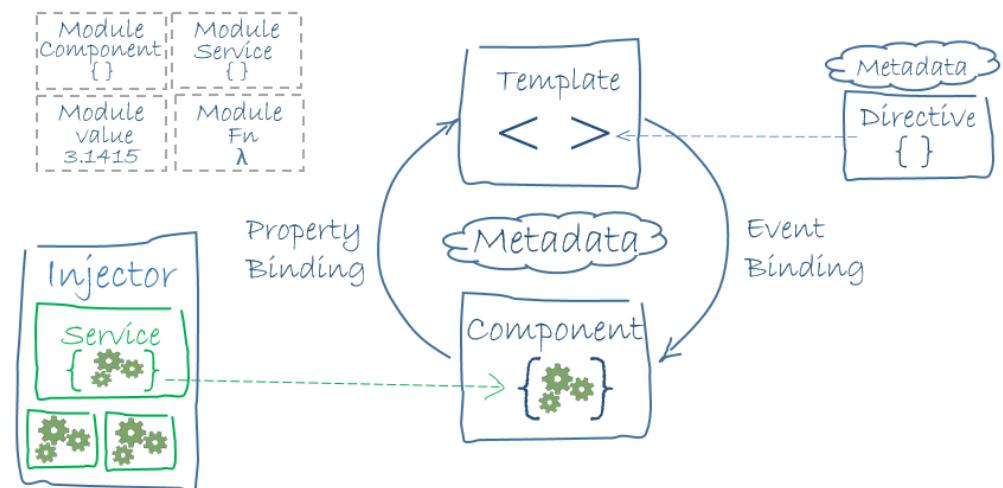
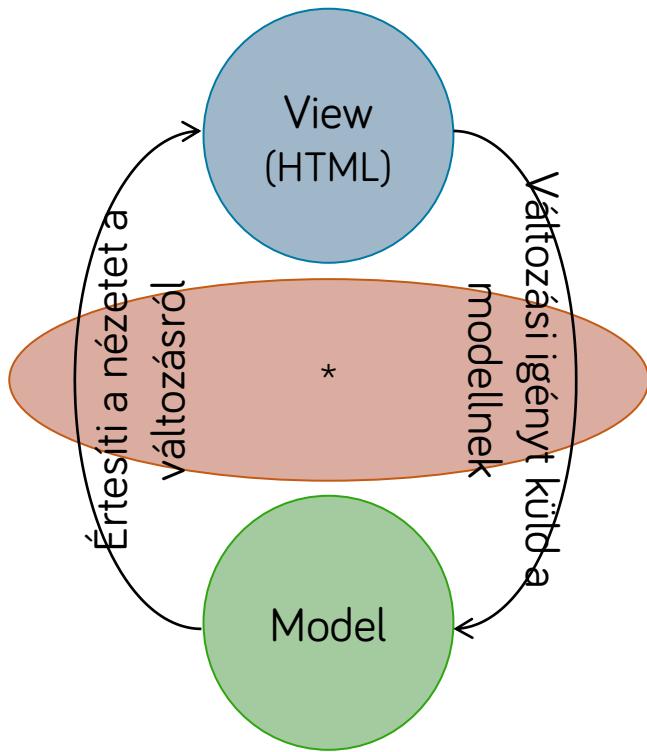


# Áttekintés

- 2009 – Probléma
  - > Túl sok a jQuery, nem karbantartható
  - > Megnő az igény az “okosabb” vékonykliens alkalmazásra
- 2010 – Megoldás: AngularJS
  - > Moduláris, teljes értékű, karbantartható, MV\*, adatkötés, SPA, separation of concerns, DI
  - > Deklaratív UI, imperatív BL
- 2014 – Az AngularJS jövője kérdéses
- 2016 – Angular 2.0
  - > Paradigmaváltások, egyértelműbb architektúra, TypeScript

# Architektúra

- MV\*, adatkötés, dependency injection, single responsibility, separation of concerns



# Metaadatok

- Reflection hiányában: TypeScript dekorátorok
  - > @NgModule, @Component, @Directive, @Input, @Output, @Inject, @Injectable

```
@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
```

```
@NgModule({
 imports: [BrowserModule],
 declarations: [HelloComponent],
 bootstrap: [HelloComponent]
})
```

```
@Directive({ selector: '[myHighlight]' })
```

# Komponensek

- Funkcionális egység saját adatkötést kezelő kóddal (@Input, @Output) és sablonnal, ami a felületen megjelenik
- Hivatkozhat más komponensekre a template-ben (komponens-hierarchia)

```
import { Component } from '@angular/core';
@Component({
 selector: 'hello-component',
 template: `<h1>Hello {{title}}!</h1>`
})
export class MyComponent {
 title = 'Angular';
}
```

# Modulok

- Alkalmazásrészeink (komponensek, direktívák, szolgáltatások) egysége
- A keretrendszer maga is moduláris
- Külső *library module*-okat is használhatunk
- Egy modul belépési pontja megadható

# Direktívák

- Attribútumdirektíva: kiegészítő működés vagy megjelenés DOM elemhez
  - > [ngStyle], [ngClass], [(ngModel)]
- Strukturális direktíva: DOM manipuláció (is)
  - > \*ngIf, \*ngSwitchCase, \*ngFor
- Komponens ≥ direktíva (van template-je, kezeli)

```
<h1 *ngIf="title === '...'">loading</h1>
<h1 *ngIf="title !== '...'">Hello {{title}}!</h1>
```

# Adatkötés

- @Input és @Output
- {{handlebar syntax}}
- [directive attribute binding syntax]="value"
- (event handler binding syntax)="handler"
- [(two-way binding syntax)]="model"

# Tartalom

~~Áttekintés~~  
~~Architektúra~~  
~~Metaadatok~~  
~~Komponensek~~  
~~Modulok~~  
~~Direktívák~~  
~~Adatkötés~~  
Szolgáltatások  
Életciklus  
Pipes  
Auszinkronitás  
Routing  
Ürlapok

# Szolgáltatások

# Szolgáltatások

- Az üzleti logikát, felületfüggetlen módon szolgáltatásokba tudjuk szervezni
- A szolgáltatások jól meghatározott háttérfunkciót látnak el vagy **infrastrukturális funkciót** adnak: HTTP kommunikáció, adattárolás/-visszanyerés, naplázás, felhasználókezelés
  - > Vagy jól leválasztható üzleti logikai funkciókat tesznek elérhetővé

# Dependency Injection (1)

- **Függőséginjektálás:** a függőségek explicit definiálása esetén egy keretrendszer képes a függőségekkel felsorolt entitásokat szolgáltatni a kérőnek
  - > A függőség szolgáltatásának folyamatát injektálásnak nevezzük
  - > A függőségek injektálásáért felelős alkalmazás-komponens az *injector* vagy *inversion of control (IoC) container*

# Dependency Injection (2)

- A függőséginjektálás:
  - > Elősegíti a szerepkörök szétválasztását (*separation of concerns*)
  - > Javítja a tesztelhetőséget
  - > Növeli az újrahasznosíthatóságot
  - > Kifejezőbbé, explicitté teszi az egyes alkalmazáskomponensek közötti függőségeket

# Dependency Injection (3)

- Az Angular az alkalmazás bootstrap folyamatát követően ad nekünk egy injector példányt
- Az injectorba beregisztrált entitások singletonként működnek (az injectorra nézve)
- A DI hierarchikusan szabályozható:
  - > A szükséges függőségeket regisztrálhatjuk az alkalmazásmodulunkba vagy egy komponensbe, utóbbi esetben csak a gyerek komponensek érik el
  - > Így akár “különböző singleton példányaink” is lehetnek

# DI – példa

```
@Injectable()
export class Logger {
 // capture logs for testing
 logs: string[] = [];
 log(message: string) {
 this.logs.push(message);
 console.log(message);
 }
}

@NgModule({
 imports: [BrowserModule],
 declarations: [HelloComponent],
 bootstrap: [HelloComponent],
 providers: [Logger]
})
export class HelloModule { }
```

```
@Component({
 selector: 'hello-component',
 template:
`<h1 (click)="log()">
 Hello {{title}}!
</h1>`
})
export class HelloComponent {
 constructor(public logger: Logger) {}

 title: string = "Angular";

 log() {
 this.logger.log(this.title);
 }
}
```

# DI – példa

```
@Injectable()
export class Logger {
 // capture logs for testing
 logs: string[] = [];
 log(message: string) {
 this.logs.push(message);
 console.log(message);
 }
}

@NgModule({
 imports: [BrowserModule],
 declarations: [HelloComponent],
 bootstrap: [HelloComponent],
 providers: [Logger]
})
export class HelloModule { }
```

Regisztráció a modul  
DI konténerébe

Injektálás engedélyezése az  
osztályon

```
@Component({
 selector: 'hello-component',
 template:
 `<h1 (click)="log()">
 Hello {{title}}!
 </h1>`
})
export class HelloComponent {
 constructor(public logger: Logger) {
 }

 title: string = "Angular"

 log() {
 this.logger.log(this.title);
 }
}
```

Konstruktur-injektálás

# Szolgáltatások

- Az Angular néhány beépített (vagy kapcsolódó) szolgáltatást ad, amiket injektálhatunk szolgáltatásokba, direktívákba, komponensekbe
  - > HttpClient: szabványos, JSON-alapú HTTP kommunikáció
  - > Location: a böngésző URL-jének burkoló szolgáltatása
  - > FormBuilder: Űrlapok összeállításához
  - > Router: alkalmazásnavigáció
  - > ElementRef: az adott direktívát, komponenst burkoló DOM elem referenciája
  - > ...

# HttpClient

- A HttpModule-ban található (@angular/common/http)
  - Nem feltétlenül kommunikál az appunk HTTP-n, ezért külön modul
- A HttpClient szolgáltatás injektálásával XHR (AJAX) kommunikációt folytathatunk
- Ne a komponenseinkben, hanem a szolgáltatásainkban használjuk (separation of concerns)
- Alapértelmezetten JSON alapú HTTP kommunikációt végez

# HttpClient – példa (1)

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/http';
import { Observable } from "rxjs/Rx";

import { Product } from './product';

@Injectable()
export class ProductService {
 constructor(private http: HttpClient) { }

 getProducts(): Observable<Product[]> {
 return this.http.get<Product[]>("/api/products");
 }
}
```

# HttpClient – példa (1)

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/http';
import { Observable } from "rxjs/Rx";
```

HttpClient  
szolgáltatás  
injektálása

```
import { Product } from './product';
```

```
@Injectable()
```

```
export class ProductService {
 constructor(private http: HttpClient) { }
```

Aszinkron működés  
Observable  
használatával

```
getProducts(): Observable<Product[]> {
```

```
 return this.http.get<Product[]>('/api/products');
```

```
}
```

Adat lekérése  
GET-tel URL-ről

# HttpClient – példa (2)

```
import { Component, OnInit } from "@angular/core";
import { Product } from "./product";
import { ProductService } from "./product.service";

@Component({
 selector: "product",
 template: `<div *ngFor='let product in products'>
 {{product.name}} {{product.price}}
 </div>`
})
export class ProductComponent implements OnInit {
 products: Product[];
 constructor(private productService: ProductService) { }
 ngOnInit(): void {
 this.productService.getProducts()
 .subscribe(products => this.products = products);
 }
}
```

# HttpClient – példa (2)

```
import { Component, OnInit } from "@angular/core";
import { Product } from "./product";
import { ProductService } from "./product.service";
```

Adathalmaz  
bejárása (amint elérhetővé válik)

```
@Component({
 selector: "product",
 template: `<div *ngFor='let product in products'>
 {{product.name}} {{product.price}}
 </div>`}
```

```
export class ProductComponent implements OnInit {
 products: Product[];
 constructor(private productService: ProductService) { }
 ngOnInit(): void {
 this.productService.getProducts()
 .subscribe(products => this.products = products);
```

Adat eltárolása  
lokális  
tulajdonságban

# HttpClient – példa (3)

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/http';
import { Observable } from "rxjs/Rx";

import { Product } from './product';

@Injectable()
export class ProductService {
 constructor(private http: HttpClient) { }

 getProducts() : Observable<HttpResponse<Product[]>> {
 return this.http.get<Product[]>("/api/products",
 observe: 'response');
 }
}
```



Az observe paraméter megadásával nem csak az adatot, hanem a teljes HTTP választ elérhetjük (pl. fejlécek, státuszkód)

# Életciklus

# Alkalmazás-életciklus

- Az Angular alkalmazás életciklusát a keretrendszer vezérli
- A DI segítségével a template feldolgozása után példányosítja a komponenseket, azok függőségeit
  - Később aszinkron események hatására új komponenseket, függőségeket gyárt

# Lifecycle hooks (1)

- Egy direktíva/komponens életciklusának különböző eseményeire iratkozunk fel, ha megvalósítjuk az alábbi függvényeket:

constructor

A component has a lifecycle managed by Angular.

ngOnChanges

Angular creates it, renders it, creates and renders its children, checks it when its data-bound properties change, and destroys it before removing it from the DOM.

ngOnInit

ngDoCheck

ngAfterContentInit

Angular offers **lifecycle hooks** that provide visibility into these key life moments and the ability to act when they occur.

ngAfterContentChecked

ngAfterViewInit

ngAfterViewChecked

A directive has the same set of lifecycle hooks, minus the hooks that are specific to component content and views.

ngOnDestroy

# Lifecycle hooks (2)

- Nem érdemes a konstruktorban inicializációs logikát futtatni, mert a konstruktor lefutásakor még nem történt meg az első adatkötés
- Először a minden változáskor is lefutó **ngOnChanges** hívódik meg, az első ilyet követően az **ngOnInit**
- **ngOnInit**: itt írjuk meg az inicializálást, mert már minden adatkötött érték rendelkezésre áll (**@Input**)
  - > Akkor is, ha **még** nem adatkötünk, később lehet, hogy fogunk

# Lifecycle hooks (3)

- Értesülünk továbbá a tartalom és nézet inicializációjáról és változást követő vizsgálatairól, valamint a direktíva megszűnéséről
- A DOM műveleteket érdemes a nézet inicializálásakor elvégezni
  - > Próbáljuk kerülni a kézi DOM műveleteket (**ElementRef** szolgáltatás), inkább használjunk mindenhol adatkötést
  - > Ahol nem elkerülhető, használjuk a **Renderer2** szolgáltatást
  - > Végső esetben nyúljunk csak a DOM-hoz közvetlenül

# Pipes

Formázás csővezetékekkel

# Csővezetékek

- A pipe (csővezeték) a felületen megjelenő adat formázására szolgál
- Néhány beépített pipe lehet segítségünkre:
  - > DatePipe
  - >UpperCasePipe/LowerCasePipe
  - >CurrencyPipe
  - >PercentPipe
  - >AsyncPipe

# ReversePipe – példa

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'reverse' })
export class ReversePipe implements PipeTransform {
 transform(value: string): string {
 return value.split("").reverse().join("");
 }
}
```

# ReversePipe – példa

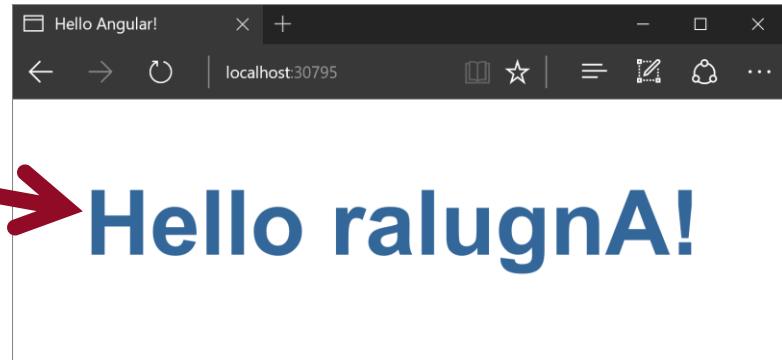
Angular @Pipe dekorátor  
és megnevezés

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({ name: 'reverse' })
export class ReversePipe implements PipeTransform {
 transform(value: string): string {
 return value.split("").reverse().join("");
 }
}
```

Bármilyen  
transzformációs  
függvény

```
<h1>Hello {{title | reverse}}!</h1>
```



# Rendezés, szűrés pipe-okkal

- Rendezési, szűrési csővezetéket **nem** ad az Angular, mert nagyon rossz teljesítményhez vezet
  - > Érdemes csak az alkalmazás logikájában szűrnünk és rendeznünk
  - > A keretrendszer figyeli ugyanis az adatkötött objektumok változásait, hogy szükség esetén újraraajzolja a felületet

# Paraméterezett, kapcsolt pipe

- Csővezeték paraméterezése

```
{{ birthday | date:'fullDate' }}
```

- Csővezetékek egymás után láncolása

```
{{ birthday | date:'fullDate' | uppercase }}
```

# Pure vs impure pipe

- A „tiszta” pipe nem detektálja a változásokat a megadott modellben, kivéve, ha a modellje:
  - > Egyszerű típus (String, Boolean, Number, Symbol)
  - > Referencia típus (Object, Function, Date, Array...) esetén csak akkor, ha a referencia változik!
- Ha minden vizsgálódnunk kell, referencia típus változása esetén, impure pipe-ot használunk: az ilyen „tisztálatlan” pipe-ot jeleznünk kell metaadatban is

```
@Pipe({ name: 'filter', pure: false })
```

  - > Alapértelmezett tiszta, az összehasonlítás gyors – az impure pipe esetén viszont teljes bejárás történik minden felhasználói és egyéb aszinkron eseményre
    - Teljesítménykritikus

# Aszinkronitás

# Szinkronitás és számosság

	Szinkron	Aszinkron
Egy érték	Függvényhívás	Promise
Több érték	Enumeráció	Observable

# Observable

- Az **Observable** (megfigyelhető) egy általános koncepció és tervezési minta, mely szerint időben érkező események egy *adatfolyamot* (stream) írnak le, amire reagálhatunk
- Az Observable-ök:
  - > Transzformálhatók, szűrhetők
  - > Több kezelő iratkoztatható fel rájuk
  - > Jelenleg ECMAScript javaslat
  - > Az RxJS keretrendszer megvalósítja
- Az egyszer bekövetkező Observable visszavezethető egyszerű Promise-ra

# Aszinkronitás + Angular + Observable (1)

## product.component.html

```
<div id="search-component">
 <h4>Product Search</h4>
 <input #searchBox (keyup)="search(searchBox.value)" />

 <div>
 <div *ngFor="let product of products | async"
 (click)="gotoDetail(product)" class="search-result" >
 {{product.name}} {{product.price}}
 </div>
 </div>
</div>
```

# Aszinkronitás + Angular + Observable (1)

product.component.html

```
<div id="search-component">
 <h4>Product Search</h4>
 <input #searchBox (keyup)="search(searchBox.value)" />

 <div>
 <div *ngFor="let product of products | async"
 (click)="gotoDetail(product)" class="search-result" >
 {{product.name}} {{product.price}}
 </div>
 </div>
</div>
```

Template-ben nevezített  
DOM elem

Eseménykezelő billentyű  
felengedésére

async pipe az Observable  
kiburkolására

# Aszinkronitás + Angular + Observable (2)

```
@Component({
 selector: "product",
 templateUrl: "product.component.html"
})
export class ProductComponent implements OnInit {
 products: Observable<Product[]>;
 private searchTerms = new Subject<string>();

 constructor(private productService: ProductService) { }

 search(term: string): void {
 this.searchTerms.next(term);
 }

 ngOnInit(): void {
 this.products = this.searchTerms.asObservable()
 .switchMap(term => this.productService.search(term));
 }
}
```

# Aszinkronitás + Angular + Observable (2)

```
@Component({
 selector: "product",
 templateUrl: "product.component.html"
})
export class ProductComponent implements OnInit {
 products: Observable<Product[]>;
 private searchTerms = new Subject<string>();
 constructor(private productService: ProductService) { }

 search(term: string): void {
 this.searchTerms.next(term);
 }

 ngOnInit(): void {
 this.products = this.searchTerms.asObservable()
 .pipe(switchMap(term => this.productService.search(term)));
 }
}
```

Template külső fájlban

Az adat az Observable-ből jön majd

Saját Observable alany

A termékek a kulcsszavak projekciója

HTTP kéréssé

# Aszinkronitás + Angular + Observable (3)

```
ngOnInit(): void {
 this.products = this.searchTerms
 .pipe(debounceTime(500))

 .pipe(distinctUntilChanged())

 .pipe(switchMap(term => term
 ? this.productService.search(term)
 : Observable.of<Product[]>([])))

 .catch(error => {
 console.log(error);
 return Observable.of<Product[]>([]);
 });
}
```

# Aszinkronitás + Angular + Observable (3)

```
ngOnInit(): void {
 this.products = this.searchTerms
 .pipe(debounceTime(500))
 .pipe(distinctUntilChanged())
 .pipe(switchMap(term => term
 ? this.productService.search(term)
 : Observable.of<Product[]>([])))
 .catch(error => {
 console.log(error);
 return Observable.of<Product[]>([]);
 });
}
```

500ms-enként csak az utolsó eseményt engedjük át

Ha nem változik a kifejezés értéke, nem vesszük új eseménynek

Egy új Observable-t gyártunk feltételtől függően: ha a term üres, akkor üres tömb a válasz

# Routing

# Routing

- A routing (útvonalválasztás) feladata, hogy adott URL-t megfelelő komponens-hierarchiához és alkalmazásállapot-halmazhoz rendelejen
- A RouterModule definiálja (@angular/router)
  - > Direktívák: RouterOutlet, RouterLink, RouterLinkActive
  - > Konfiguráció: Routes
- Lehetséges hierarchikus útvonalválasztást is megvalósítani
- Alapértelmezetten a HTML5 history API-t használja

# Routing konfiguráció

```
@NgModule({
 imports: [
 BrowserModule,
 RouterModule.forRoot([
 { path: 'products', component: ProductsListComponent },
 { path: 'search', component: SearchComponent },
 { path: 'products/:id', component: ProductDetailsComponent }])
],
 declarations: [AppComponent, ProductsListComponent,
 SearchComponent, ProductDetailsComponent],
 bootstrap: [AppComponent]
})
export class AppModule { title = 'products'; }
```

# Routing konfiguráció

```
@NgModule({
 imports: [
 BrowserModule,
 RouterModule.forRoot([
 { path: 'products', component: ProductsListComponent },
 { path: 'search', component: SearchComponent },
 { path: 'products/:id', component: ProductDetailsComponent }])
],
 declarations: [AppComponent, ProductsListComponent,
 SearchComponent, ProductDetailsComponent],
 bootstrap: [AppComponent]
})
export class AppModule { title = 'products'; }
```

Factory metódus a helyben konfigurált RouteModule regisztrációjához

URL paraméter, amit a komponens az ActivatedRoute szolgáltatás segítségével ki tud nyerni

Komponensek útvonalhoz rendelése

# RouterOutlet, RouterLink

- **RouterOutlet**: ez a direktíva végzi maguknak a komponenseknek a példányosítását a megadott helyen a DOM-ban
- **RouterLink**: ez a direktíva egy horgony (`<a>`) elemhez a megfelelő linket és működést rendeli
- **RouterLinkActive**: a link aktivitásának megfelelően CSS classt illeszt az adott DOM elemre

# Linkek

## app.component.html

```
<h1>{{title}}</h1>
<nav>

 Products

 Search products

</nav>
<router-outlet></router-outlet>
```

# Linkek

app.component.html

```
<h1>{{title}}</h1>
<nav>

 Products

 Search products

</nav>
<router-outlet></router-outlet>
```

A Routerbe regisztrált komponensek közül az aktuális a RouterOutlet helyén jelenik meg

A két főbb útvonalat a felhasználó a linkekkel, vagy az URL megadásával választja ki

A stílusozást testeszabjuk az active CSS class segítségével

# Feltételes navigáció: Guard

- A navigáció megakadályozható, ha bizonyos feltételek nem teljesülnek
- Ehhez egy Guard példányt regisztráltunk a DI-ba és az adott útvonalhoz rendeljük

# Feltételek navigáció: Guard

```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';
import { LoginService } from './login.service';

@Injectable()
export class AuthGuardService implements CanActivate {
 constructor(private loginService: LoginService) { }
 canActivate(): boolean {
 return !!this.loginService.user;
 }
}

[
 { path: 'login', component: LoginComponent },
 { path: 'profile', component: ProfileComponent,
 canActivate: [AuthGuardService] }
 ...
]
```

# Feltételek navigáció: Guard

```
import { Injectable } from '@angular/core';
import { CanActivate } from '@angular/router';
import { LoginService } from './login.service';

@Injectable()
export class AuthGuardService implements CanActivate {
 constructor(private loginService: LoginService) { }
 canActivate(): boolean {
 return !!this.loginService.user;
 }
}

[
 { path: 'login', component: LoginComponent },
 { path: 'profile', component: ProfileComponent,
 canActivate: [AuthGuardService] }
 ...
]
```

Guard interfész implementációja

Guarddal védett útvonal

# Ürlapok

# Űrlapok

- Az űrlapok egységes adatbeviteli felületek
- Gyakori feladatok üzleti alkalmazásokban:
  - > Adatbekérő/-módosító felület
  - > Felületi validáció, hibaüzenet megjelenítése
- Két alapvető űrlap-készítési módszer létezik:
  - > Sablon alapú (template-driven)
  - > Modell alapú (reactive)

# Template alapú Űrlap – példa (1)

```
<h1>Hero Form</h1>
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
 <div class="form-group">
 <label for="name">Name</label>
 <input type="text" class="form-control"
 id="name" required [(ngModel)]="model.name",
 name="name" #name="ngModel">
 <div [hidden]="name.valid || name.pristine"
 class="alert alert-danger">
 Name is required
 </div>
 </div>
 <div class="form-group">
 <label for="alterEgo">Alter Ego</label>
 <input type="text" class="form-control" id="alterEgo"
 [(ngModel)]="model.alterEgo" name="alterEgo">
 </div>
 ...
</form>
```

# Template alapú Űrlap – példa (1)

```
<h1>Hero Form</h1>
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
 <div class="form-group">
 <label for="name">Name</label>
 <input type="text" class="form-control"
 id="name" required [(ngModel)]="model.name"
 name="name" #name="ngModel">
 <div [hidden]="name.valid || name.pristine"
 class="alert alert-danger">
 Name is required
 </div>
 </div>
 <div class="form-group">
 <label for="alterEgo">Alter Ego</label>
 <input type="text" class="form-control" id="alterEgo"
 [(ngModel)]="model.alterEgo" name="alterEgo">
 </div>
 ...
</form>
```

Angular ngForm direktíva  
példány eltárolása

Kétirányú adatkötés  
az inputon

Hibaüzenet feltételes  
megjelenítése

# Template alapú Űrlap – példa (2)

```
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
 ...
 <div class="form-group">
 <label for="power">Hero Power</label>
 <select class="form-control" id="power" required
 [(ngModel)]="model.power" name="power" #power="ngModel">
 <option *ngFor="let pow of powers" [value]="pow">{{pow}}</option>
 </select>
 <div [hidden]="power.valid || power.pristine"
 class="alert alert-danger">
 Power is required
 </div>
 </div>

 <button type="submit" class="btn btn-success"
 [disabled]="!heroForm.form.valid">Submit</button>
 <button type="button" class="btn btn-default"
 (click)="newHero(); heroForm.reset()">New Hero</button>
</form>
```

# Template alapú Űrlap – példa (2)

```
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
 ...
 <div class="form-group">
 <label for="power">Hero Power</label>
 <select class="form-control" id="power" required
 [(ngModel)]="model.power" name="power" #power="ngModel">
 <option *ngFor="let pow of powers" [value]="pow">{{pow}}</option>
 </select>
 <div [hidden]="power.valid || power.pristine"
 class="alert alert-danger">
 Power is required
 </div>
 </div>
 <button type="submit" class="btn btn-success"
 [disabled]="!heroForm.form.valid">Submit</button>
 <button type="button" class="btn btn-default"
 (click)="newHero(); heroForm.reset()">New Hero</button>
</form>
```

The diagram illustrates several annotations with red arrows pointing to specific parts of the code:

- An arrow points from the text "Űrlap küldés esemény kötése" to the `(ngSubmit)="onSubmit()"` event binding.
- An arrow points from the text "Legördülő elemválasztó" to the `[(ngModel)]` two-way data binding used for the dropdown.
- An arrow points from the text "Űrlap törlése" to the `(click)="newHero(); heroForm.reset()"` click handler for the "New Hero" button.
- An arrow points from the text "Űrlap küldésének megakadályozása" to the `[disabled]="!heroForm.form.valid"` expression in the "Submit" button's `disabled` attribute.

# Template alapú Űrlap

## Hero Form

**Name**

Batmaaan

**Alter Ego**

Bruce Waaayne

**Hero Power**

Has Gadgets

**Submit** **New Hero**

# Visszatekintés

# Ezkről volt szó

- Architektúra
- Metaadatok
- Komponensek
- Modulok
- Direktívák
- Adatkötés
- Szolgáltatások
- Életciklus
- Pipes
- Aszinkronitás
- Routing
- Űrlapok

Ismétlés, gyors áttekintés: <https://angular.io/guide/cheatsheet>

# Ezkről nem volt szó...

- Komponens stílusok
- Animációk
- AOT fordítás
- Reactive Forms
- Lokalizáció / globalizáció (i18n)
- Változásdetektálás
- Modulszintű lazy-loading
- NgTemplate
- Angular Universal
- Library-k integrálása
- Biztonság
- Tesztelés
- AngularJS
- Fejlesztéstámogatás
- ...

Bővebben a hivatalos dokumentációban: <https://angular.io/docs/>

# Kliensoldali technológiák

## Angular

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Kliensoldali technológiák

Modern webes technológiák

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék

# Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki  
és Gazdaságtudományi Egyetem oktatója által  
kidolgozott szerzői mű.**

**Kifejezetten felhasználási engedély nélküli  
felhasználása szerzői jogi jogosításnak minősül.**

A szerző elérhetősége:  
[Szabo.Gabor@vik.bme.hu](mailto:Szabo.Gabor@vik.bme.hu)

# Tartalom

A modern webfejlesztés  
eszközei

JavaScript keretrendszerek

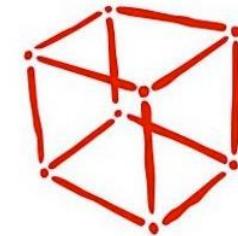
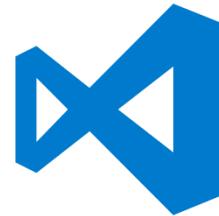
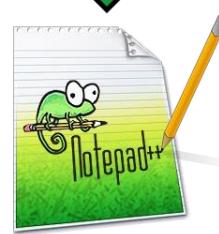
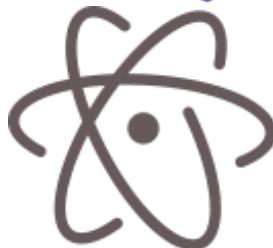
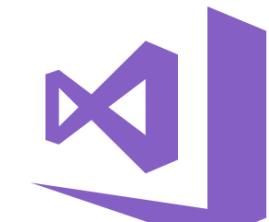
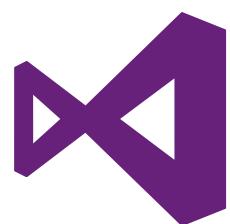
Blazor

Mobil web

Szerverek

# A modern webfejlesztés eszközei

# Kódszerkesztők



...

# Free-for-all

- Az alkalmazásunk értéke nagyon ritkán van a forráskódban
  - > Algoritmusok, tényleg komplex üzleti logikák
  - > A valódi érték szinte mindig az adat
  - > Ha obfuscálnunk kell, akkor az algoritmust védjük, vagy magunkat, hogy rossz kódot írtunk?
- Szinte minden aktívan fejlesztett keretrendszer ingyenesen használható, emellett nyílt forráskódú is
- A most bemutatandó technológiák mindegyike ingyenesen használható, legtöbbük open source is

# Csomagkezelők

- Különböző kliensoldali függőségeink vannak
  - > Angular, Bootstrap, jQuery...
- Fejlesztéshez is használunk függőségeket
  - > Ez nem szükséges az alkalmazás futásához, csak fejlesztéshez (és teszteléshez)
  - > TypeScript, Angular CLI, SASS compiler...
- Ezeket egységesen szeretnénk kezelní



- Az npm a NodeJS szerver csomagkezelője (NodeJS Package Manager), parancssori eszköz
  - > Eredetileg ténylegesen csak szerveroldali csomagok terjesztésére
  - > De semmi nem akadályozza meg a kliensoldali csomagok ugyanilyen jellegű kezelését
  - > 2020 óta a GitHub (Microsoft tulajdonában)
- Az npm a projekteket a **package.json** fájl alapján azonosítja
  - > Verzió, csomag neve, készítő, licensz...
  - > Futási függőségek
  - > Fejlesztői függőségek
- A függőségek a package.json melletti **node\_modules** mappába kerülnek
- A npm alapértelmezetten a saját repository-jában elérhető, publikus csomagokat telepíti ([npmjs.com](https://npmjs.com))



# - package.json példa

```
{
 "name": "project",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 /*...*/
 },
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 /*...*/
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~6.0.0-beta.6",
 "@types/node": "~8.9.4",
 "typescript": "~2.5.3"
 /*...*/
 }
}
```



# - package.json példa

```
{
 "name": "project",
 "version": "0.0.0",
 "license": "MIT",
 "scripts": {
 "ng": "ng",
 "start": "ng serve",
 /*...*/
 },
 "dependencies": {
 "@angular/animations": "^5.2.0",
 "@angular/common": "^5.2.0",
 "@angular/compiler": "^5.2.0",
 /*...*/
 "core-js": "^2.4.1",
 "rxjs": "^5.5.6",
 "zone.js": "0.8.19"
 },
 "devDependencies": {
 "@angular/cli": "~6.0.0-beta.6",
 "@types/node": "~8.9.4",
 "typescript": "~2.5.3"
 /*...*/
 }
}
```

Projektszintű metaadatok

Parancsok (pl. npm start)

Projekt függőségei

Legfrissebb főverzió (2.x.y)

Pontos verzió (0.8.19)

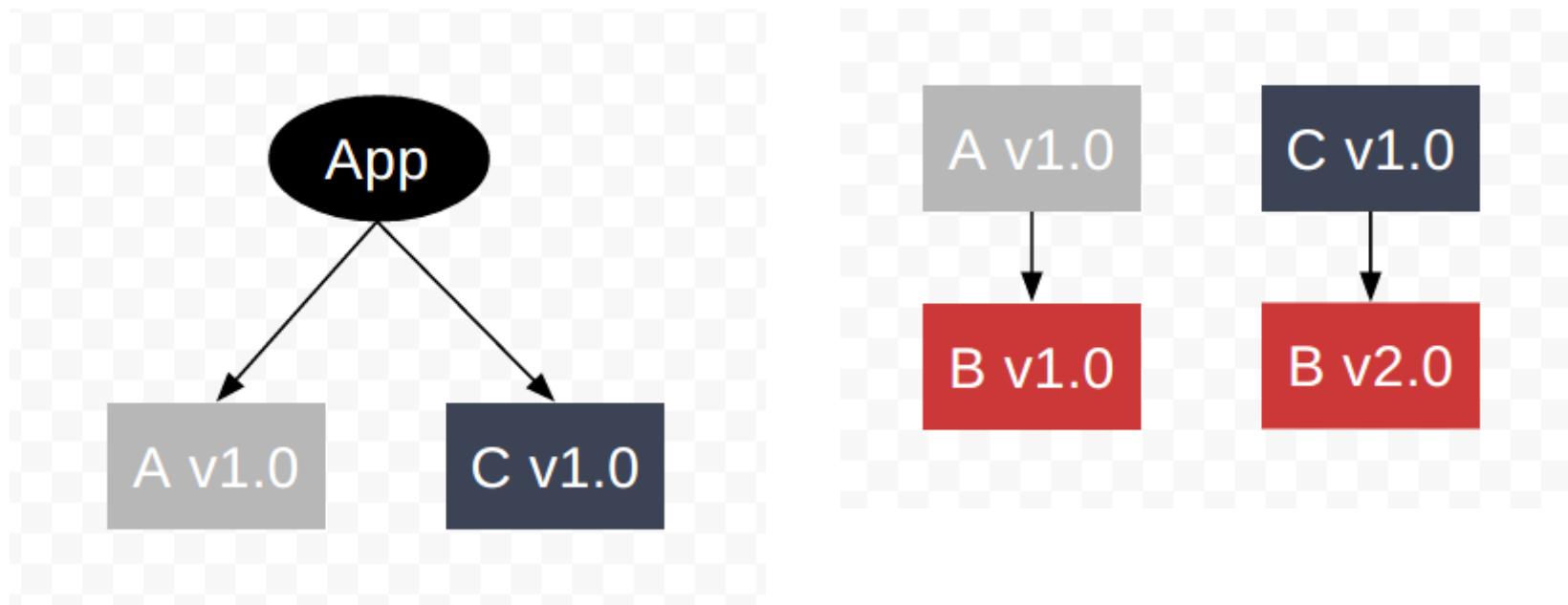
Legfrissebb patch-verzió (6.0.x)

Fejlesztői függőségek



# - függőségi problémák (1)

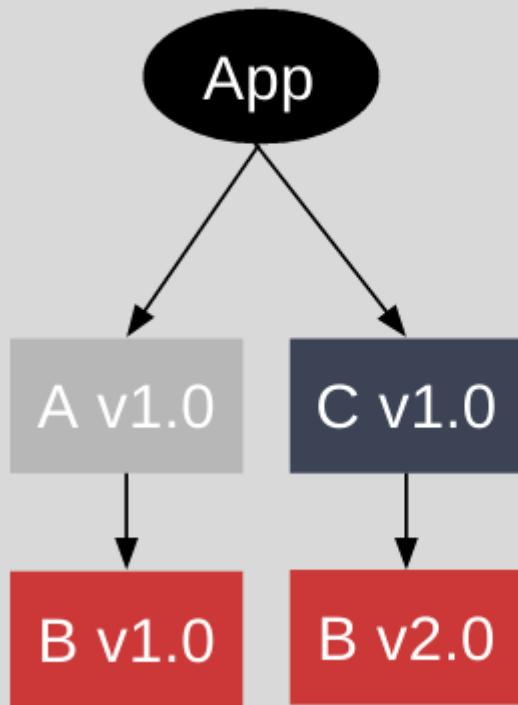
- A tranzitív függőségek nehezen oldhatók fel, ha ütközés van verziók között: *dependency hell*



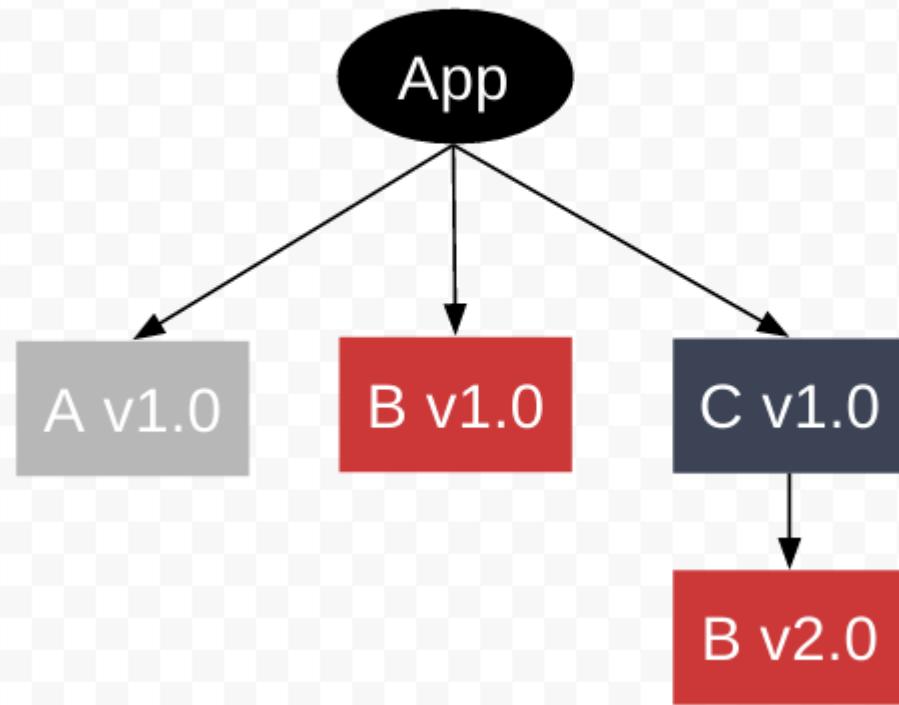


# - függőségi problémák (2)

npm v2



npm v3





# - függőségi problémák (3)

- Megoldás
  - > ?
- package.lock.json
  - > A telepítéskor fennálló teljes függőségi fáról készít egy lenyomatot
  - > Ennek segítségével reprodukálható installációt lesz
  - > Ha az aktuális pillanatnak megfelelő függőségi fát szeretnénk:  
`npm ci`



# - „package hell”

- **leftpad**
  - > Egy szokásos, „egysoros” npm csomag volt
  - > A fejlesztő 2016-ban úgy döntött, hogy nem publikálja többé a csomagot
  - > Gyakorlatilag a teljes JavaScript világ romokba dőlt napokig, ugyanis minden nagyobb keretrendszer támaszkodott erre az egyszerű funkcióra
- Kb. minden éven történik egy hasonló eset

# npm - függőségi problémák (3)

- Az npm-et sok kritika érte a teljesítménye, stabilitása és (túl)konfigurálhatósága miatt a korábbi években
- Elkezdtek előretörni az alternatívák:



libman

# GitHub Package Registry

- Az eszközök maradnak (npm, yarn), a csomagok helyének ez az alternatívájra
- Különféle csomagoknak:
  - > npm
  - > NuGet
  - > Maven
  - > RubyGems
  - > Docker image

<https://github.blog/2019-05-10-introducing-github-package-registry/>



# Bower

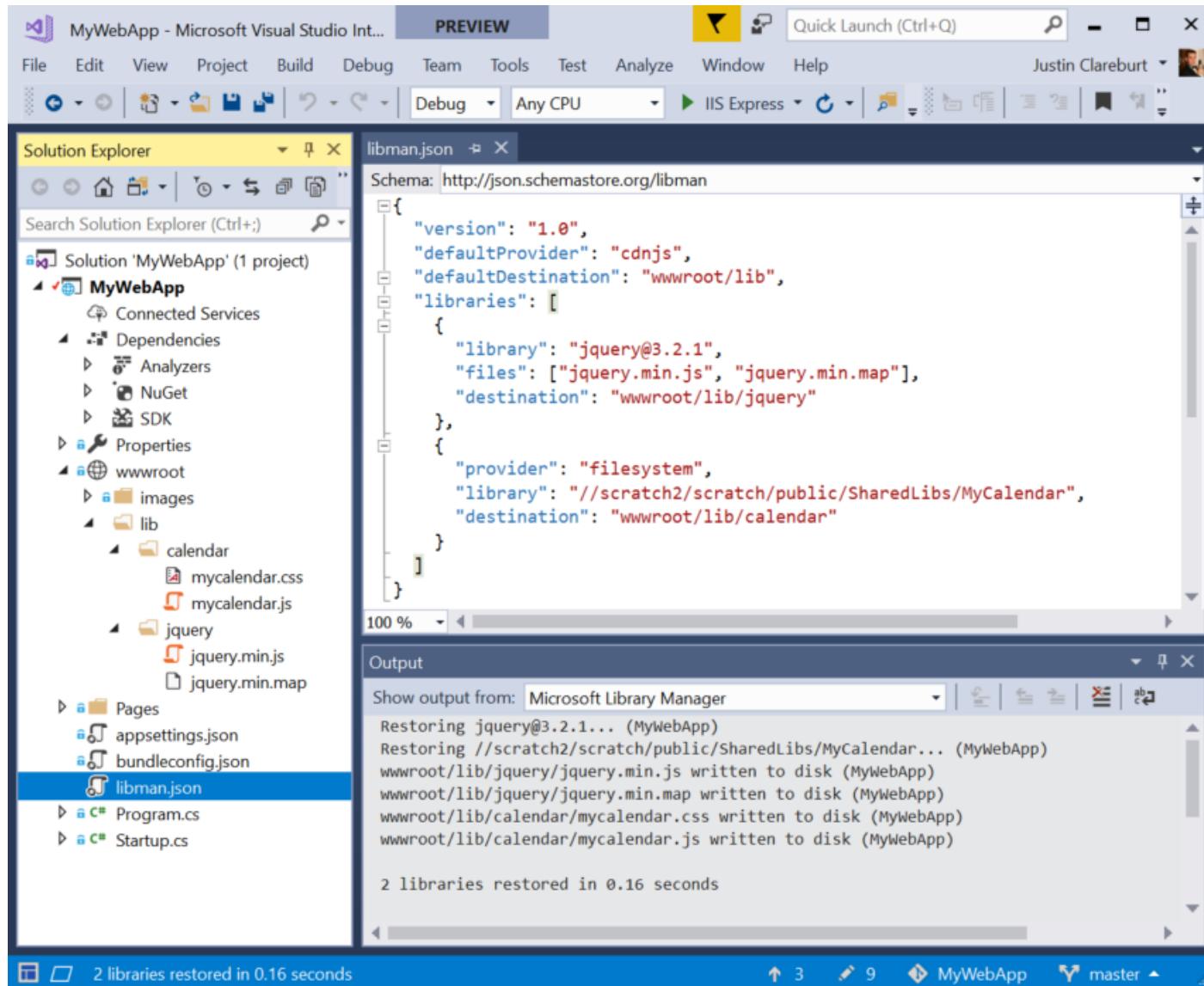
- Kizárálag kliensoldali csomagkezelésre
  - > Fejlesztői függőségeket máshogyan kell beszerezni
- Az npm egyeduralkodóvá vált, már nem ajánlatos használni
  - > A [bower](#) viszont a yarn, webpack és parcel használatára buzdít

# LibMan

- Egy nagyon egyszerű megoldás függőségek fájlrendszerbeni karbantartására
  - > Elsőszorban a szerver oldali renderelést használó projektekben hasznos .NET környezetben

<https://devblogs.microsoft.com/aspnet/library-manager-client-side-content-manager-for-web-apps/>

# LibMan



# Fejlesztői csomagkezelők

- Nem csak kliensoldali csomagokat kell kezelnünk
  - > Természetesen letölthetjük az .exe fájlt vagy .msi installert, de sok fejlesztői függőség esetén körülményes



# Automatizálás

- Gyakran automatizálható feladatokat végzünk fejlesztés közben
  - > Build, csopatos átnevezések, összefűzések, batch jellegű feladatok
- Egyéb automatizálható feladatok segítenek optimalizálni a csomagjainkat
  - > Minifikáció, csomagolás, obfuscálás

# Grunt vs Gulp



# Grunt vs Gulp

- Gyakorlatilag ugyanazt a feladatot oldják meg: minden kettőben futtatható feladatok definiálhatók

gruntfile.js

```
module.exports = function(grunt) {
 grunt.initConfig({
 concat: {
 'dist/all.js': ['src/*.js']
 },
 uglify: {
 'dist/all.min.js': ['dist/all.js']
 },
 jshint: {
 files: ['gruntfile.js', 'src/*.js']
 },
 watch: {
 files: ['gruntfile.js', 'src/*.js'],
 tasks: ['jshint', 'concat', 'uglify']
 }
 });

 // Load Our Plugins
 grunt.loadNpmTasks('grunt-contrib-jshint');
 grunt.loadNpmTasks('grunt-contrib-concat');
 grunt.loadNpmTasks('grunt-contrib-uglify');
 grunt.loadNpmTasks('grunt-contrib-watch');

 // Register Default Task
 grunt.registerTask('default', ['jshint', 'concat', 'uglify']);
};


```

gulpfile.js

```
var gulp = require('gulp');
var jshint = require('gulp-jshint');
var concat = require('gulp-concat');
var rename = require('gulp-rename');
var uglify = require('gulp-uglify');

// Lint JS
gulp.task('lint', function() {
 return gulp.src('src/*.js')
 .pipe(jshint())
 .pipe(jshint.reporter('default'));
});

// Concat & Minify JS
gulp.task('minify', function(){
 return gulp.src('src/*.js')
 .pipe(concat('all.js'))
 .pipe(gulp.dest('dist'))
 .pipe(rename('all.min.js'))
 .pipe(uglify())
 .pipe(gulp.dest('dist'));
});

// Watch Our Files
gulp.task('watch', function() {
 gulp.watch('src/*.js', ['lint', 'minify']);
});

// Default
gulp.task('default', ['lint', 'minify', 'watch']);

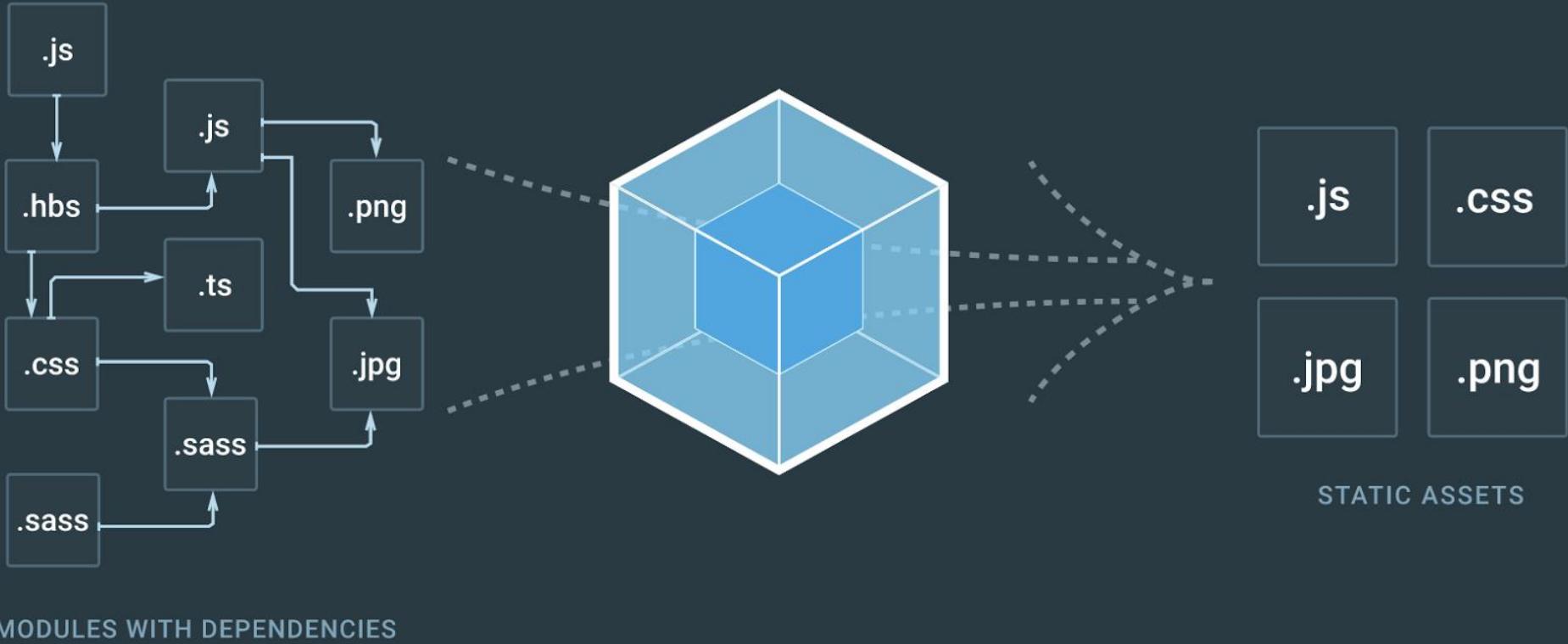

```

- A Gulp idővel átvette a vezetést az átláthatóbb szintaxisa miatt

Forrás: <https://www.accelebrate.com/blog/tale-two-task-runners-grunt-gulp/>



# webpack





# webpack

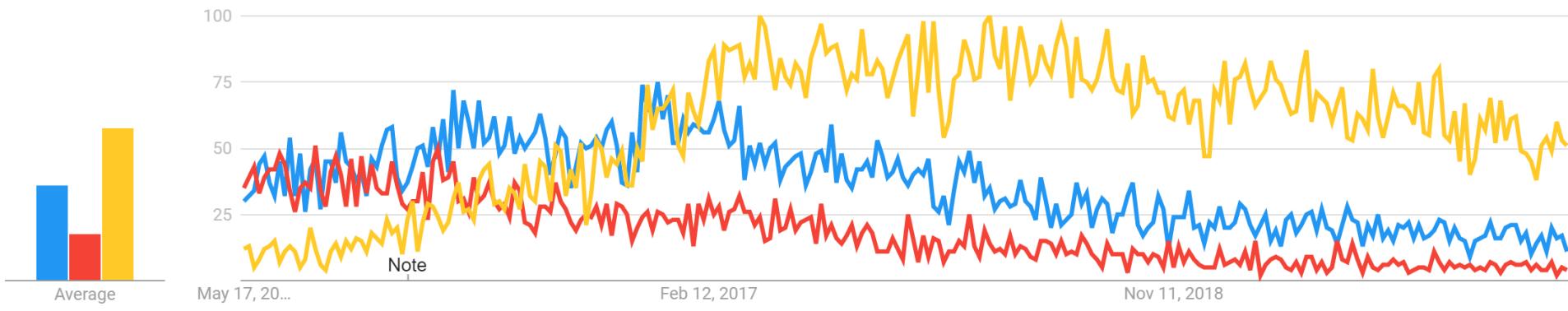
- A leggyakrabban automatizált feladat az alkalmazás fordítása és csomagolása (bundling)
  - > JS, CSS, HTML (!)
  - > TypeScript → JavaScript
  - > SASS → CSS
  - > Képek (!)
- A webpack nem *feladatfuttató*, mint a Gulp és Grunt, hanem egy *modulcsomagoló*
- Ezzel a célzattal speciálisabb pluginek is készíthetők
  - > Pl. Hot Module Replacement: csak a szükséges kódrészletet fordítja újra és frissíti a megnyitott böngészőt
- Az Angular CLI is webpack alapú, a teljes csomagolást a webpack végzi
- Nagyon részletesen konfigurálható, de hamar elbonyolódik



Forrás: <https://blog.qmo.io/javascript-tooling-the-evolution-and-future-of-js-front-end-build-tools/>

# Változó trendek

- Nehéz a frontend fejlesztő dolga, mert folyamatosan változnak az aktuális technológiák
- Alkalmazkodni és tanulni kell



Grunt, Gulp, Webpack

Forrás: [Google Trends](#)

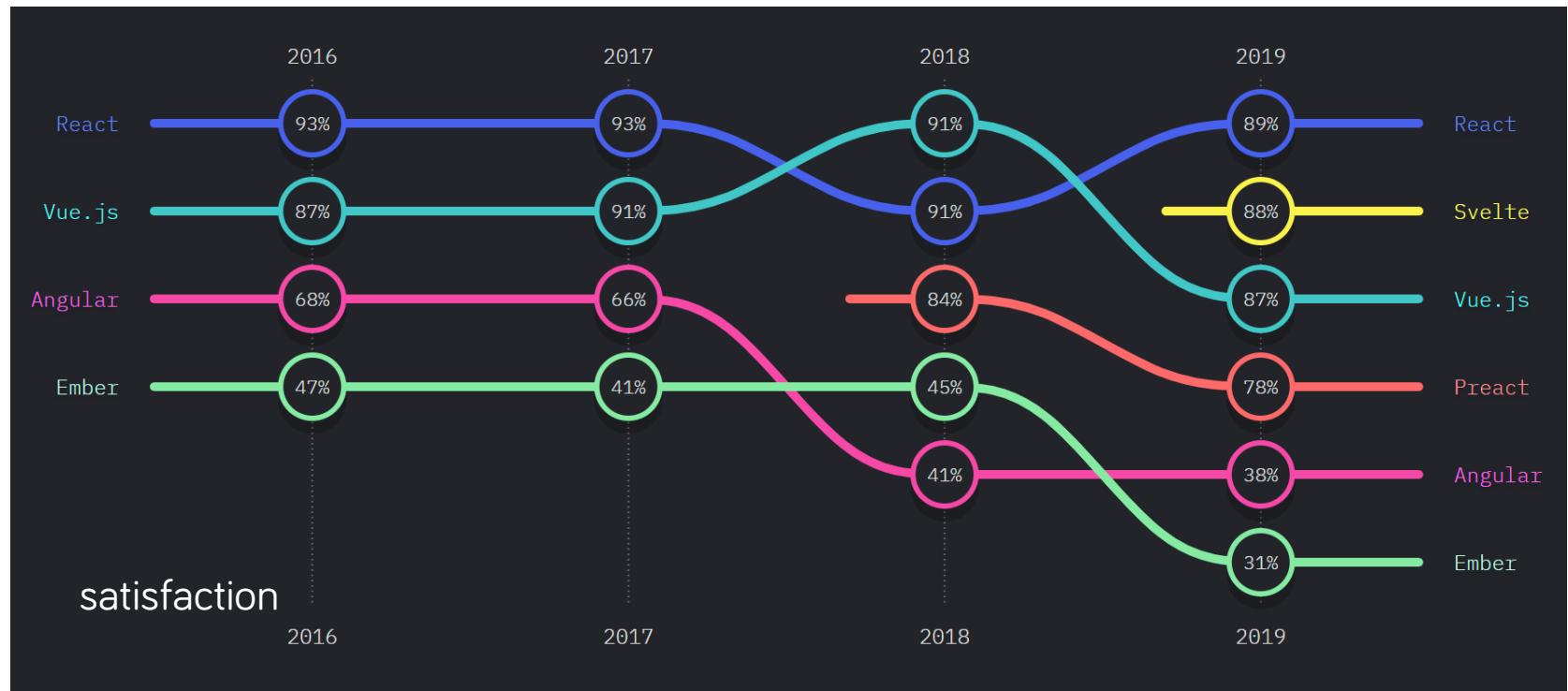
# Változó trendek

- Nagyon széles az eszközkészlet, amit használatba vehetünk
- Jellemzően a „trendi” eszközökről olvasunk jókat, az elődeikről rosszakat
  - > De érdemes mindig ésszerűen mérlegelni, hogy milyen eszközre van szükségünk!
  - > CI/CD eszközök használatával néha szükségtelenné is válnak az automatizáló eszközök, de a háttérben ezek gyakran az ismertetett technológiákra építenek
    - Azure Pipelines, GitHub Actions, Jenkins...

# JavaScript keretrendszerök

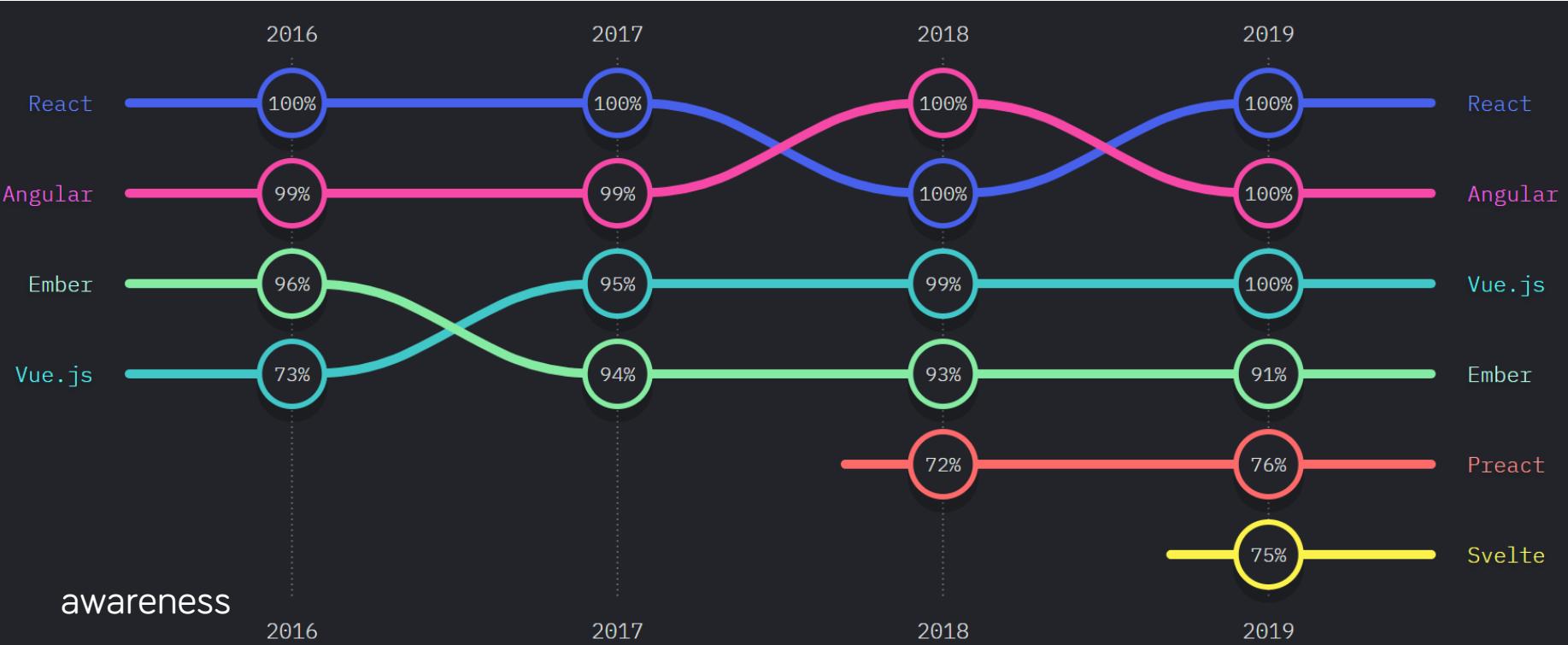
# JS keretrendszerök

- Hasonlóan a toolinghez, a keretrendszerök trendje is változó



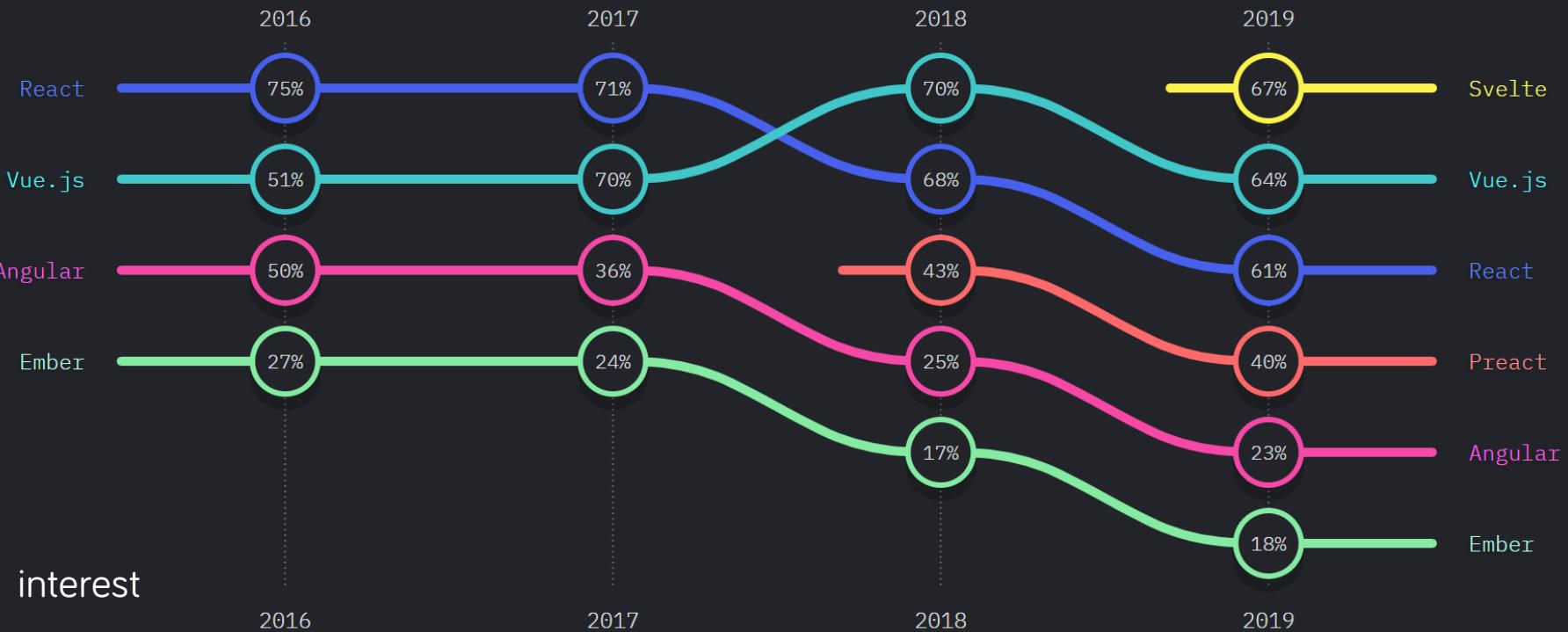
Forrás: <https://2019.stateofjs.com/front-end-frameworks/>

# JS keretrendszerök



Forrás: <https://2019.stateofjs.com/front-end-frameworks/>

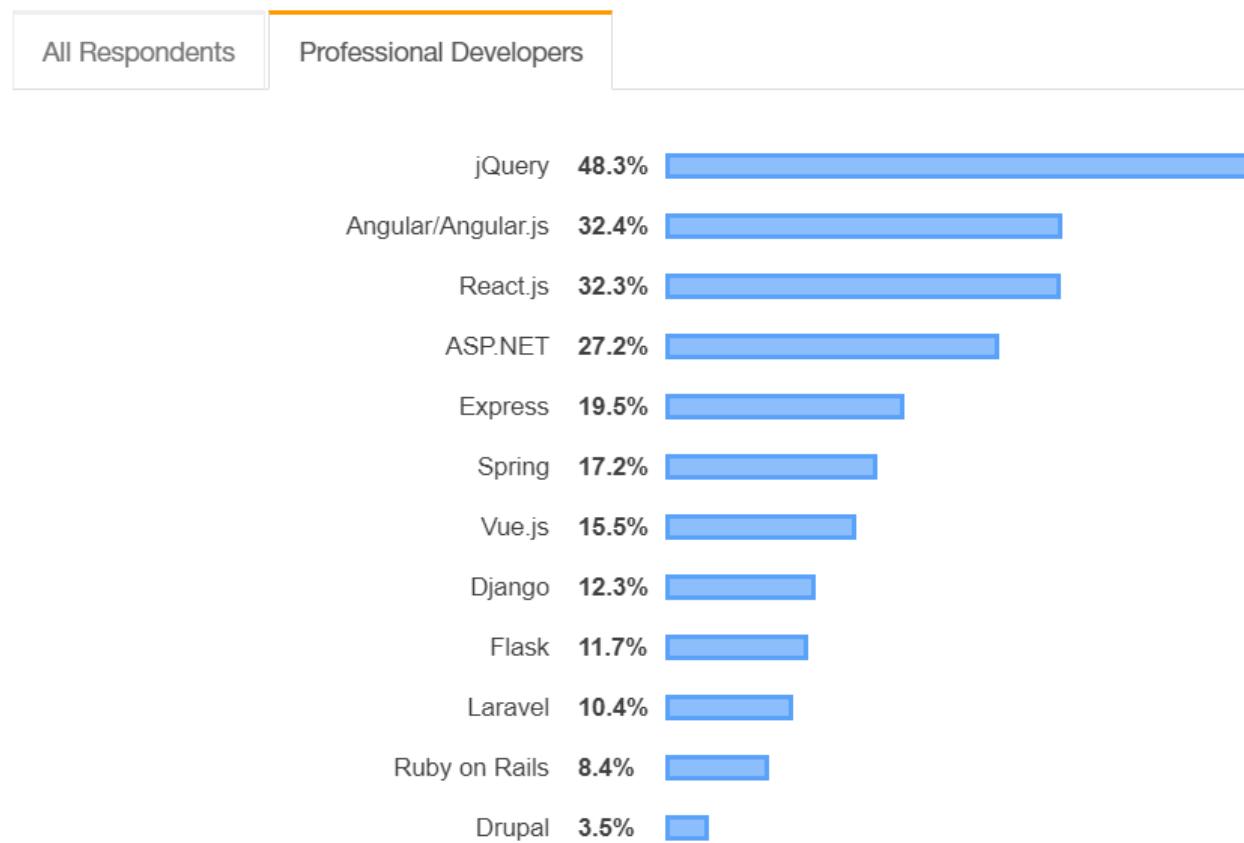
# JS keretrendszerök



Forrás: <https://2019.stateofjs.com/front-end-frameworks/>

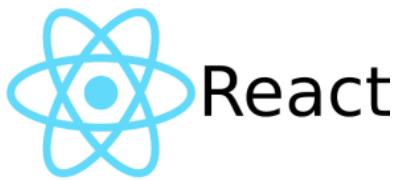
# Webes keretrendszerek

## Web Frameworks



55,079 responses; select all that apply

Forrás: <https://insights.stackoverflow.com/survey/2019#technology--web-frameworks>



- A Facebook fejlesztése, a Facebook és az Instagram is React alapú UI-t használ
- Csak a “V” az “MVC”-ben (nem framework, library!)
- Egyirányú adatáramlás
- JSX: saját kiterjesztett JS szintaxis template-ezésre

```
<div id="myReactApp"></div>
<script type="text/babel">
 class Greeter extends React.Component {
 render() { return <h1>{this.props.greeting}</h1> }
 }
 ReactDOM.render(<Greeter greeting="Hello World!" />,
 document.getElementById('myReactApp'));
</script>
```



- Reaktív sablonok
- Komponensek, direktívák, routing
- Hasonlít az Angularre, de kevésbé teljesértékű
- Támogatja a fokozatos átállást
- TypeScript támogatással:

```
import Vue from 'vue'
import Component from 'vue-class-component'

@Component({
 template: '<button @click="onClick">Hello {{ clicked }}!</button>'
})
export default class MyComponent extends Vue {
 clicked = 0;
 onClick () {
 this.clicked++;
 }
}
```

# A választás jogá

- Sokszor belekényszerülünk egy keretrendszerbe
- Valamikor választanunk kell közülük
  - > A legtöbbször nem egyértelmű a döntés, mozgó célpontra lövünk
- De van, hogy keretrendszer nélkül indulunk el, natív JS kóddal
  - > A szerveroldalon renderelt oldal kiegészítéseképp
  - > Vagy a teljes kliensoldal JS-ben
    - Így születtek a mostaniak is!
    - De alapvető kérdéseket nem válaszolnak meg...

# Melyik a *legjobb* keretrendszer?

- minden keretrendszernek vannak előnyei
- Fontos, hogy egy keretrendszer nem azért jó, mert mindenki azt használja
- A jó keretrendszerek:
  - > Jól használhatók (nincs „fighting the framework”)
  - > Jól támogatottak (pl. hamar reagálnak a talált bugokra)
  - > Jól oldják meg az általuk megoldani kívánt problémákat



blazor



- A Blazor a Microsoft legújabb webalkalmazás-keretrendszeré
- WebAssembly használatával böngészőben fut, .NET alapon
  - > Vagy a Blazor Server segítségével a szerver tartja karban a kliens állapotát
- Razor szintaxis
- Megosztható kód/logika a szerver-kliens között (kódgenerálás nélkül)



<https://github.com/yugabe/BlazorMiner>

Lobby Hello, gabe@test.com! Log out About

## BlazorMiner

Let's Play!

gabe szabo-772 (24 pts)

gabe-fef (-5 pts)

	1					
	1					
	1	1				
	\$	2	1	1	2	
	1	2	\$		2	\$
		1				

4



# WEBASSEMBLY

# Áttekintés

- A WebAssembly (wasm) alacsonyszintű, bináris programozási nyelv, mely hatékony, natív-közeli futtatást tesz lehetővé böngészőkben
  - > Az asm.js tekinthető az elődjének
- Már elérhető, lehet rá fejleszteni ☺



WebAssembly 1.0 has shipped in 4 major browser engines.

- Célja, hogy más, magasszintű nyelvek erre fordítva weben, hatékonyan futtathatók legyenek

# Bináris formátum

- A bináris formátumnak köszönhetően:
  - > Jól tömöríthető
  - > Gyorsan betölthető
- A bináris formátum szövegesen reprezentálható

C++	Binary	Text
<pre>int factorial(int n) {     if (n == 0)         return 1;     else         return n * factorial(n-1); }</pre>	<pre>20 00 42 00 51 04 7e 42 01 05 20 00 20 00 42 01 7d 10 00 7e 0b</pre>	<pre>get_local 0 i64.const 0 i64.eq if i64     i64.const 1 else     get_local 0     get_local 0     i64.const 1     i64.sub     call 0     i64.mul end</pre>

# Egy köztes lépés

- A wasm végső célja, hogy egységes platformként legyenek kezelhetők a webböngészők is natív élményekhez
  - > Nem cél tehát, hogy valaki kézzel wasm kódot írjon
- Elérhető a böngésző JavaScript API-ja is
- Várhatóan új alkalmazásplatformok fognak elterjedni, amik WebAssembly alapúak
  - > Az 1.0 verzió csak magának a platformnak a kiismerésére szolgál
  - > Egyelőre csak C/C++ a fókusz
  - > A Blazor WebAssembly az egyik élenjárója

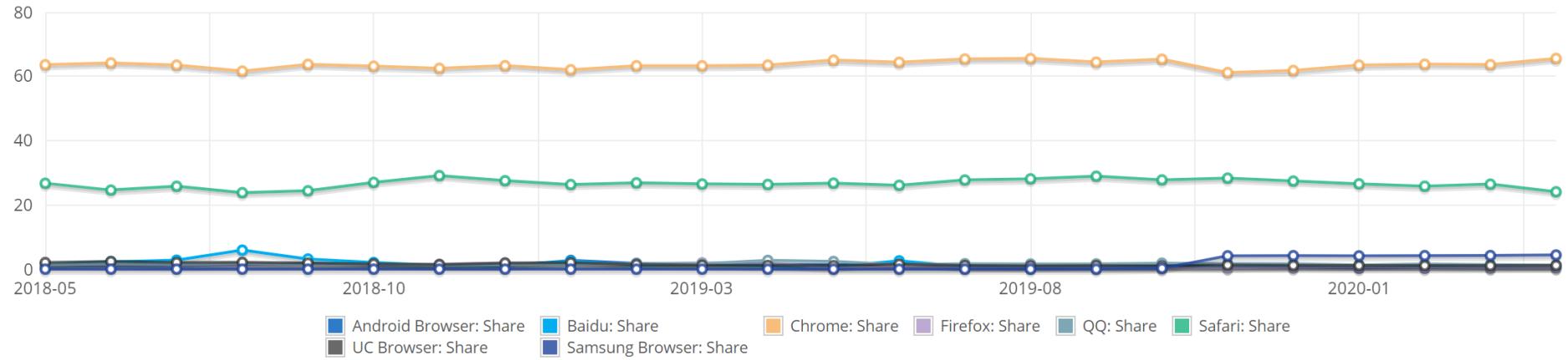
# Mobil web

Web a mobilon

# Webtechnológiák mobil eszközön

- Mobil eszközökön alkalmazásokat futtathatunk a böngészőben:
  - > Ténylegesen a böngészőben
  - > WebView konténerben natív alkalmazáson belül
- A két fő platformot célszerű támogatni
  - > Ezen belül már több lehetőségünk van, hogy melyik böngészőket célozzuk meg

# Mobil böngészők



Show 10 entries

Search:

<input type="checkbox"/> Browser	<input checked="" type="checkbox"/> Share
<input checked="" type="checkbox"/> Chrome	63.67%
<input checked="" type="checkbox"/> Safari	26.62%
<input checked="" type="checkbox"/> QQ	1.61%
<input checked="" type="checkbox"/> Baidu	1.51%
<input checked="" type="checkbox"/> UC Browser	1.43%
<input checked="" type="checkbox"/> Samsung Browser	1.26%
<input checked="" type="checkbox"/> Android Browser	1.16%
<input checked="" type="checkbox"/> Firefox	1.01%

Forrás: [NetMarketShare](#)

# Weboldalak mobilon (1)

- Ha szimplán weboldalt készítünk, a mobilra történő optimalizálás általában “kimerül” az adaptív és reszponzív elvek használatában (ld.: Bootstrap)
  - > **Reszponzív**: az alkalmazás minden felbontáson “nyúlik”, igyekszik kitölteni a rendelkezésre álló helyet
  - > **Adaptív**: töréspontok mentén az alkalmazás kinézete “átalakul”

# Weboldalak mobilon (2)

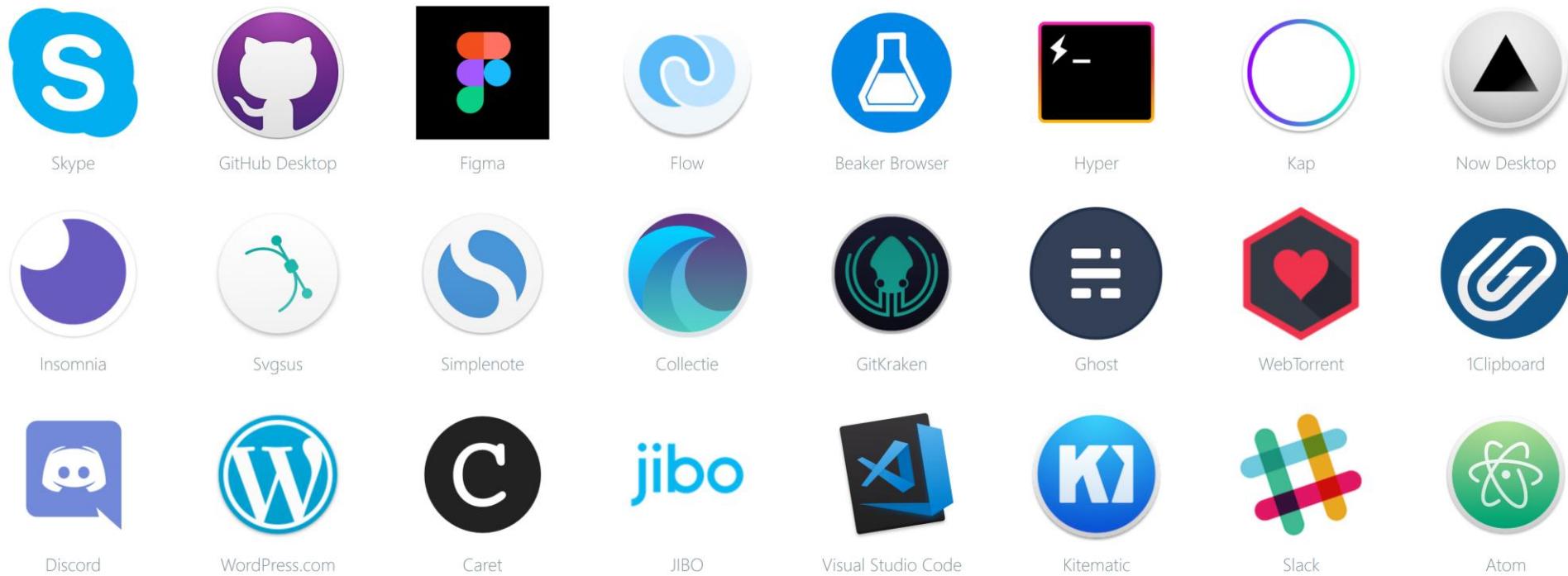
- Alapvetően nem érjük el a natív OS által kínált mobilspecifikus funkciókat
  - > Kamera, akkumulátor töltöttsége, névjegyzék, SMS, mozgásérzékelő, orientáció, hálózat stb...
- Ha mégis szeretnénk *valamiféle* natív funkcionalitást...

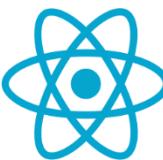
- Az Apache Cordova egy natív appkontérbe burkolja a webalkalmazásunkat, és plugineket nyújt a natív kommunikációhoz
  - > Terjesztés tehát a store-okon keresztül történik
- Készíthetünk saját pluginet is, ekkor a natív implementációt platformspecifikusan el kell készítenünk (pl. iOS-re, Androidra)
- A projektszerkezetért szintén npm-ből letölthető CLI felel
- Támogatott platformok:
  - > Android, iOS, Windows, Ubuntu, OS X (BlackBerry, Windows Phone)
- Bármilyen webalkalmazást készíthetünk, a Cordova csak a csomagolásért, a WebView konténerért és a JS API-n elérhető natív funkciókért felelős



# ELECTRON

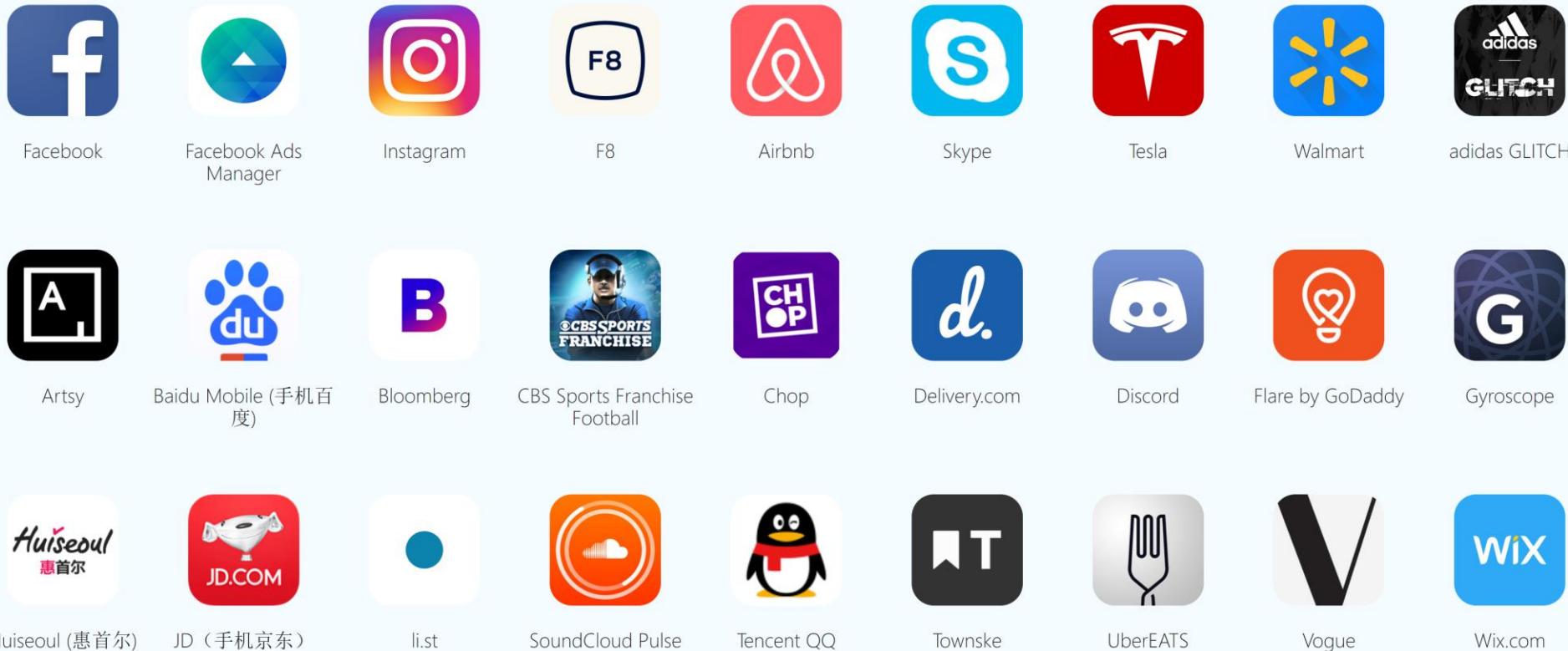
- A GitHub terméke
- Node.js és Chromium alapú “vastagkliens” alkalmazások fejlesztésére

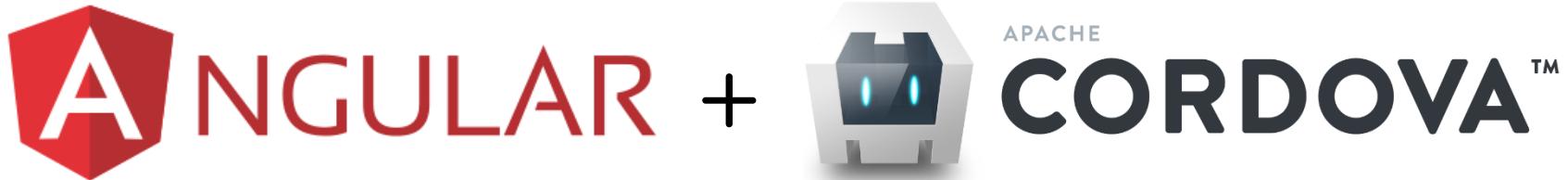




# React Native

- Ténylegesen natív mobilalkalmazások  
JavaScriptben, Reacttal





- Natív kinézetű (iOS, Material, Windows), de webtechnológiákon alapuló alkalmazások fejlesztésére
- Kiváló dokumentáció
- Böngészőben és natív konténerben futtatható
  - > *PWA* támogatással

# Progresszív webalkalmazások

- Böngészőben futtatható webalkalmazások, melyek önálló alkalmazásként **is** futtathatók
  - > Mobil, desktop
- Támogatnak offline működést
- Telepíthetők a lokális eszközre
- A store-okból is telepíthetők/lesznek
  - > A Microsoft Store már listázza őket

# Progresszív webalkalmazások alapelvei

- **Progresszív:** bármilyen böngészőn, bármilyen internetkapcsolattal elérhető (egyre jobb élmény)
- **Reszponzív:** bármilyen elrendezésen
- **Kapcsolatfüggetlen:** a *service worker* kezeli az alkalmazás offline módját
- **Alkalmazás-szerű:** interakciók és navigáció
- **Friss:** minden friss információk
- **Biztonságos:** HTTPS kapcsolaton
- **Megtalálható:** a manifest információi alapján felderíthető
- **Visszahúz:** a felhasználó érdeklődését folyamatosan fenntartja, pl. push értesítések formájában
- **Telepíthető:** alkalmazásként telepíthető, nem kell store-ban megkeresni
- **Hivatkozható:** minden URL-lel azonosítható

Forrás: <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/>

# Szerverek

A kliensek szerverei

# Szerverek a kliens szemszögéből (1)

- A vékonykliens (+hibrid/natív) alkalmazásaink jellemzően csak a megjelenítést és a felhasználói interakciókat kezelik
- A valós érték továbbra is a szervere(ke)n van
  - > Megosztott adattárolás (DB)
  - > Üzleti logika
  - > Szolgáltatások

# Szerverek a kliens szemszögéből (2)

- A kliens alkalmazást el kell juttatnunk a böngészőbe
  - > Ehhez egy egyszerű fájlszerver is elegendő
    - Kivéve talán az SPA navigációt
- Fejlesztéskor plusz funkciókra lehet szükségünk
  - > Hot Module Replacement
  - > Sourcemap fájlok
  - > Timetravel debugging

# Szerverek a kliens szemszögéből (3)

- Az esetek nagy részében a kliens szólítja meg a szervert (client pull)
  - > Leggyakrabban HTTP+JSON kommunikáció (REST, RPC)
- Van viszont szerveroldali push is, ha már a kétirányú kapcsolat felépült
  - > WebSocket, natív push

# Szerveroldali renderelés

- LoB, enterprise alkalmazásokban jellemzően *nem* a UX az elsődleges, hanem:
  - > Biztonság
  - > Megbízhatóság
  - > Bővíthetőség
  - > Gyors fejlesztés
  - > Robosztusság
  - > Párhuzamos fejlesztések (feature teams)
- Sokszor elegendő a szerveren renderelnünk a HTML-t
  - > Esetleg kiegészíteni kliensoldali funkciókkal

# ASP.NET Core

- C#, F# (VB)
  - > Az aszinkron funkcionális kezelése kiemelkedő
- Razor View Engine
  - > “Angular direktívák szerveroldalon”
- Blazor
- A Microsoft nyílt forráskódú, ingyenes terméke
- Middleware pipeline, dependency injection, MVC, statikus fájlkiszolgálás out-of-the-box
- Nagyon robosztus, dinamikusan fejlődik
- Könnyű belevágni, de utána megugrik a tanulási görbe
- Hosszú távú projekteknél hamar megtérül, nagyon jó karbantarhatóságot eredményez
- Kiváló tooling a VS Code és VS formájában
- A de facto .NET szerver keretrendszer



- Java, Kotlin, Groovy
- Dependency injection, auth, adatelérés, MVC, tranzakciókezelés, AOP
- Eclipse, Spring Tool Suite, IntelliJ IDEA



- Kotlin, Java
- Microservice micro-framework, website-okhoz is
- Egyéb Java alapú szerver keretrendszerek:  
<https://www.dalyrazor.com/blog/best-java-web-frameworks/>

```
import static spark.Spark.*;

public class HelloWorld {
 public static void main(String[] args) {
 get("/hello", (req, res) -> "Hello World");
 }
}
```

```
import spark.kotlin.*

fun main(args: Array<String>) {
 val http: Http = ignite()

 http.get("/hello") {
 "Hello Spark Kotlin!"
 }
}
```

# express

- JavaScript
  - > TypeScript, CoffeeScript stb.
- Minimalista, nagyon gyors webszerver
- Routing, 14+ template engine
- Middleware alapú
- Gyors alkalmazásfejlesztéshez
- Kliensoldali JS tapasztalattal a könnyű belevágni

# Python szerverek

## django

- > Magasszintű, gyors fejlesztésre és tiszta tervezéshez



## Flask

- > Microservice-ekhez
- Egyéb Python alapú szerver keretrendszerek:  
<https://wiki.python.org/moin/WebFrameworks>

# Egyéb források

- Szerver keretrendszerkről az MDN-en:  
[https://developer.mozilla.org/en-US/docs/Learn/Server-side/First\\_steps/Web\\_frameworks](https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks)
- Ruby alapú szerverek:  
<https://medium.com/hashdog-blog/top-10-ruby-frameworks-3cc1214caae7>
- PHP alapú szerverek:  
<https://www.hongkiat.com/blog/best-php-frameworks/>

# Kliensoldali technológiák

Modern webes technológiák

Szabo.Gabor@vik.bme.hu



Automatizálási és  
Alkalmazott  
Informatikai Tanszék