ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

Αλληλεπίδραση ανθρώπου - υπολογιστή

Ενότητα 6: Εργαστήριο: Σχεδίαση Διεπαφών και Αλληλεπίδρασης με τα εργαλεία MS Visual Studio, Blend, XAML, C#

Παναγιώτης Κουτσαμπάσης &
Σπυριδών Βοσινάκης
Τμήμα Μηχανικών Σχεδίασης
Προϊόντων και Συστημάτων

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αιγαίου**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Αλληλεπίδραση Ανθρώπου-Υπολογιστή (Human-Computer Interaction, HCI)

ΠΑΡΟΥΣΙΑΣΕΙΣ ΕΡΓΑΣΤΗΡΙΟΥ: Θέματα Σχεδίασης Διεπαφών Χρήστη

Εργαλεία: MS Visual Studio, Blend, XAML, JSON, C#.

Παναγιώτης Κουτσαμπάσης, επίκουρος καθηγητής, http://www.syros.aegean.gr/users/kgp

Σπύρος Βοσινάκης, επίκουρος καθηγητής, http://www.syros.aegean.gr/users/spyrosv

Εργαστήριο HCI

- Εισαγωγή στο MS Visual Studio, Blend, XAML (1E)
- Σχήματα, χρώματα, κινηματική (shapes, colour, animation) (1E)
- Διατάξεις (layouts) (1E)
- Συστατικά διεπαφής (user interface components) (2E)
- Στυλ και περιγράμματα (styles and templates) (2E)
- Δέσμευση δεδομένων (data binding) (2E)
- Συμβάντα και χειριστές συμβάντων (events, event handlers) (2E)
- Δεδομένα κατά το χρόνο της σχεδίασης (design-time data) (2E)

Εισαγωγή στο MS Visual Studio, Blend, XAML

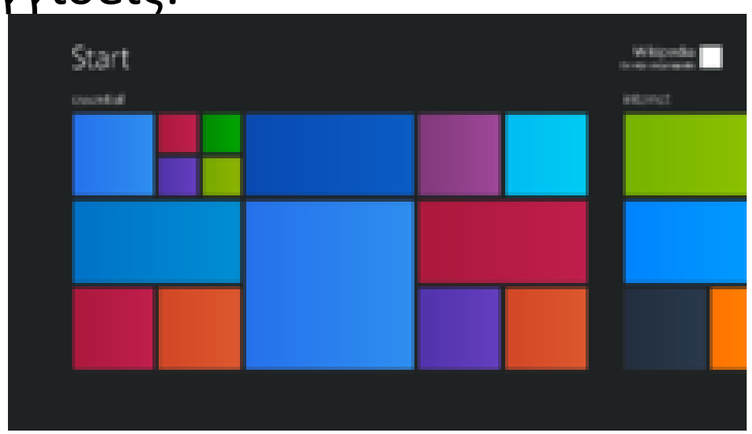
- MS Visual Studio 2013 + Blend 2013, MS Windows 8.1
  - Συνεργάζονται για να ολοκληρωθεί η σχεδίαση με την ανάπτυξη λογισμικού…
  - Μέχρι και το 2012, ξεχωριστές εκδόσεις.
  - Στα εργαστήρια χρησιμοποιούμε Remote Desktop!
  - Για πρόσβαση από το σπίτι σας:
    - VPN Connection: https://ype.aegean.gr/yphresiesdiktioudedomenwn/vpn/genikesodigievpn
    - Remote desktop to: SYROS-RDS.AEGEAN.GR
- **C#,** VB, HTML5, C, C++, …
- Προσπάθεια ανάπτυξης για κάθε πλατφόρμα και το Web
  - Σε αντίθεση με τις παλαιότερες τακτικές της Microsoft…

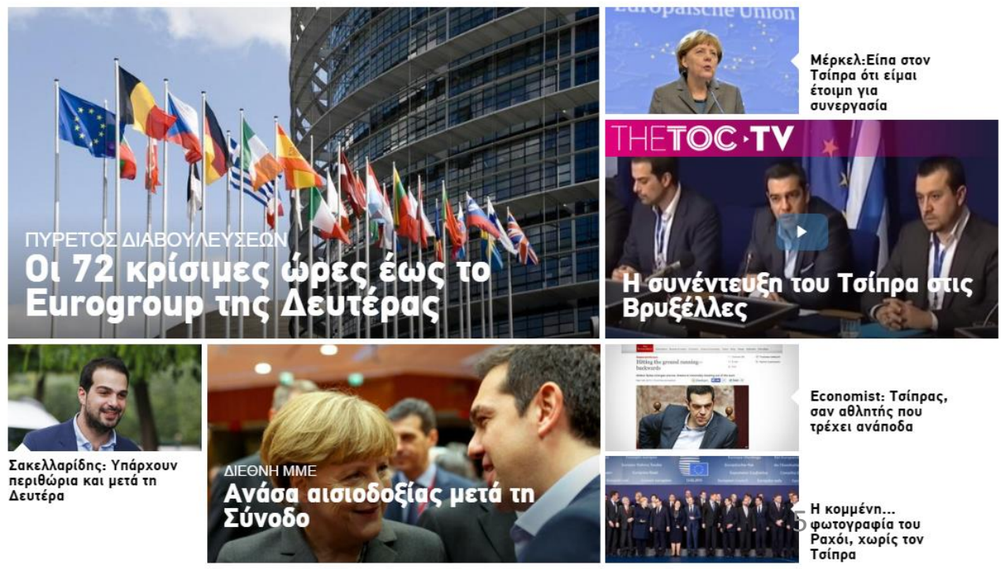
Εισαγωγή στο MS Visual Studio, Blend, XAML

- Γιατί μόνο C#, και όχι π.χ. HTML5 (HTML, CSS, Javascript)?
  - Δεν θέλουμε να δούμε μόνο για Web (θα δείτε σε άλλα μαθήματα)…
    - + Mobile app design & development (either MS Phone, but also cross platform – universal apps, Cordoba, etc.),
    - + MS Windows (ver. 8.0, and higher) design & development.
- Τα εργαλεία είναι μακράν πιο ώριμα από οτιδήποτε άλλο κυκλοφορεί σήμερα (π.χ. Online prototyping tools, Dreamweaver, κλπ.)…
  - Πολλές δυνατότητες, με κάποια πολυπλοκότητα… απαιτείται να αποκτήσετε εξοικείωση…

Εισαγωγή στο MS Visual Studio, Blend, XAML

- Metro design, μια σχεδιαστική προσέγγιση (‘γλώσσα’) με «έμφαση στο περιεχόμενο αντί των γραφικών»
  - Win8, Xbox, Windows Store, επίσης σε αρκετά Web sites (ιδιαίτερα WordPress-based).
  - Tiles (underlying grid), support for interactions for multiple devices, animations, responsive UI (adapts to different screens and supports gestures), based on dynamic content (content lists, maps, etc), rests heavily on photographs instead of icons.
  - Αντίστοιχες προσεγγίσεις: flat design (older term, generic), material design (Google, Android)





XAML (eXtensible Application Markup Language)

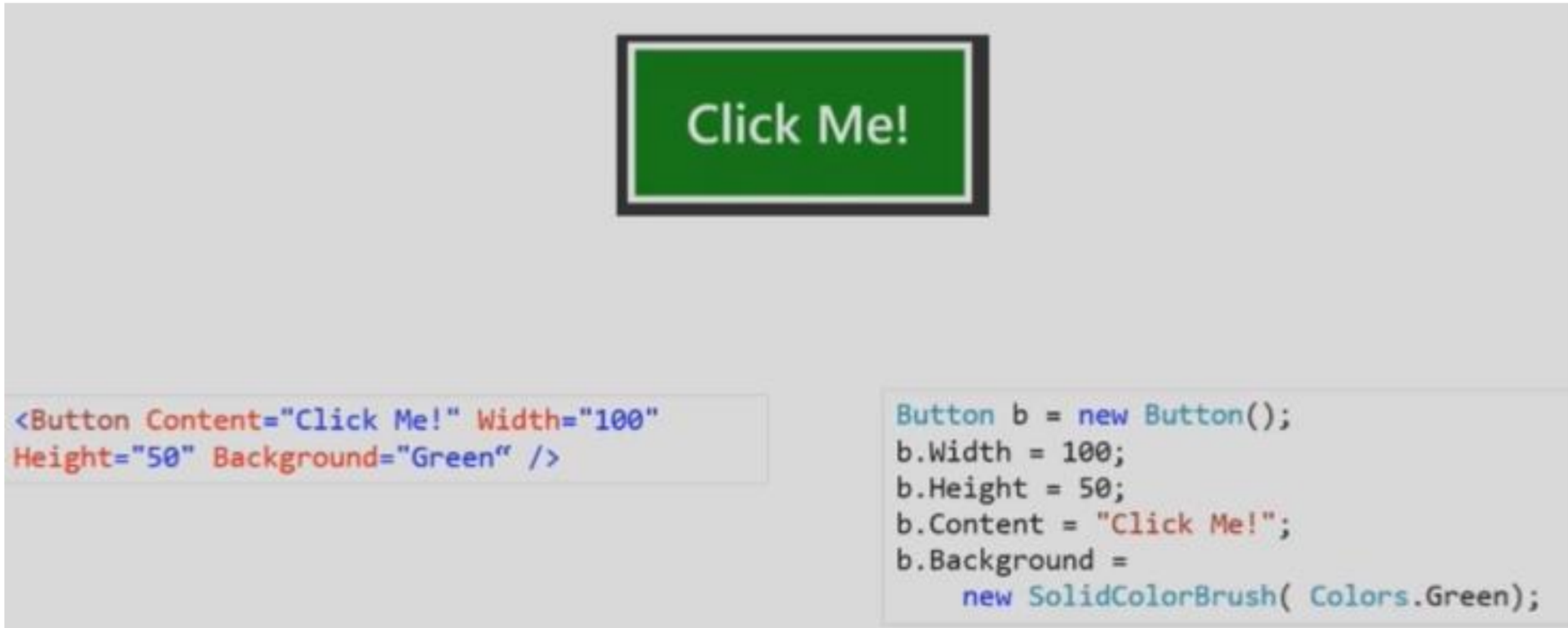
- Declarative language
- Initialization
- Serialization

- XAML Syntax In Detail:
- https://msdn.microsoft.com/en-us/library/ms788723(v=vs.110).aspx

```
<Page
  x:Class="App12.MainPage"…>
  <Grid>
    <Grid.Resources>
      <Style x:Key="PinkButton" TargetType="Button">
        <Setter Property="Background" Value="Pink" />
      </Style>
    </Grid.Resources>

    <Button
            x:Name="myButton"
            Style="{StaticResource PinkButton}"
            Content="{Binding data.buttonName}"
            Click="OnButtonClick"
            Width="300"
            Margin="250"
            VerticalAlignment="Stretch">
        </Button>
    </Grid>

</Page>
```

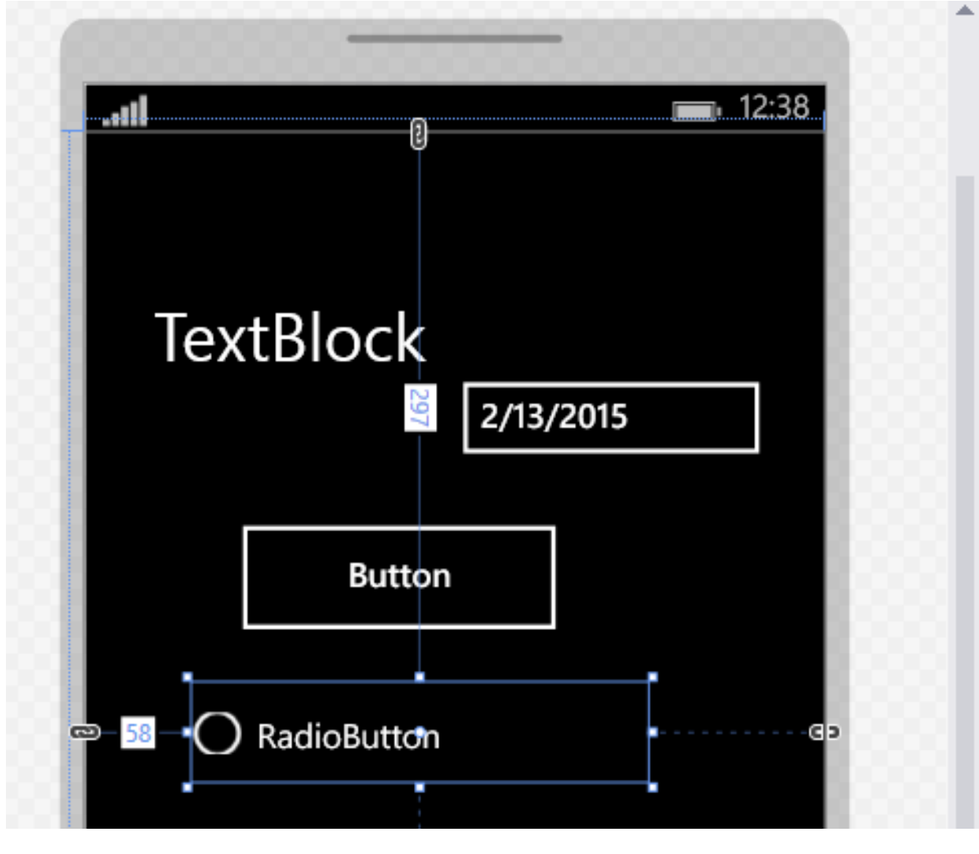
XAML (eXtensible Application Markup Language)

- Κάθε XAML file, έχει ένα code-behind file (C#, ή άλλη)
- XAML vs. C#
- η XAML έχει πολλούς converters για να είναι human-readable



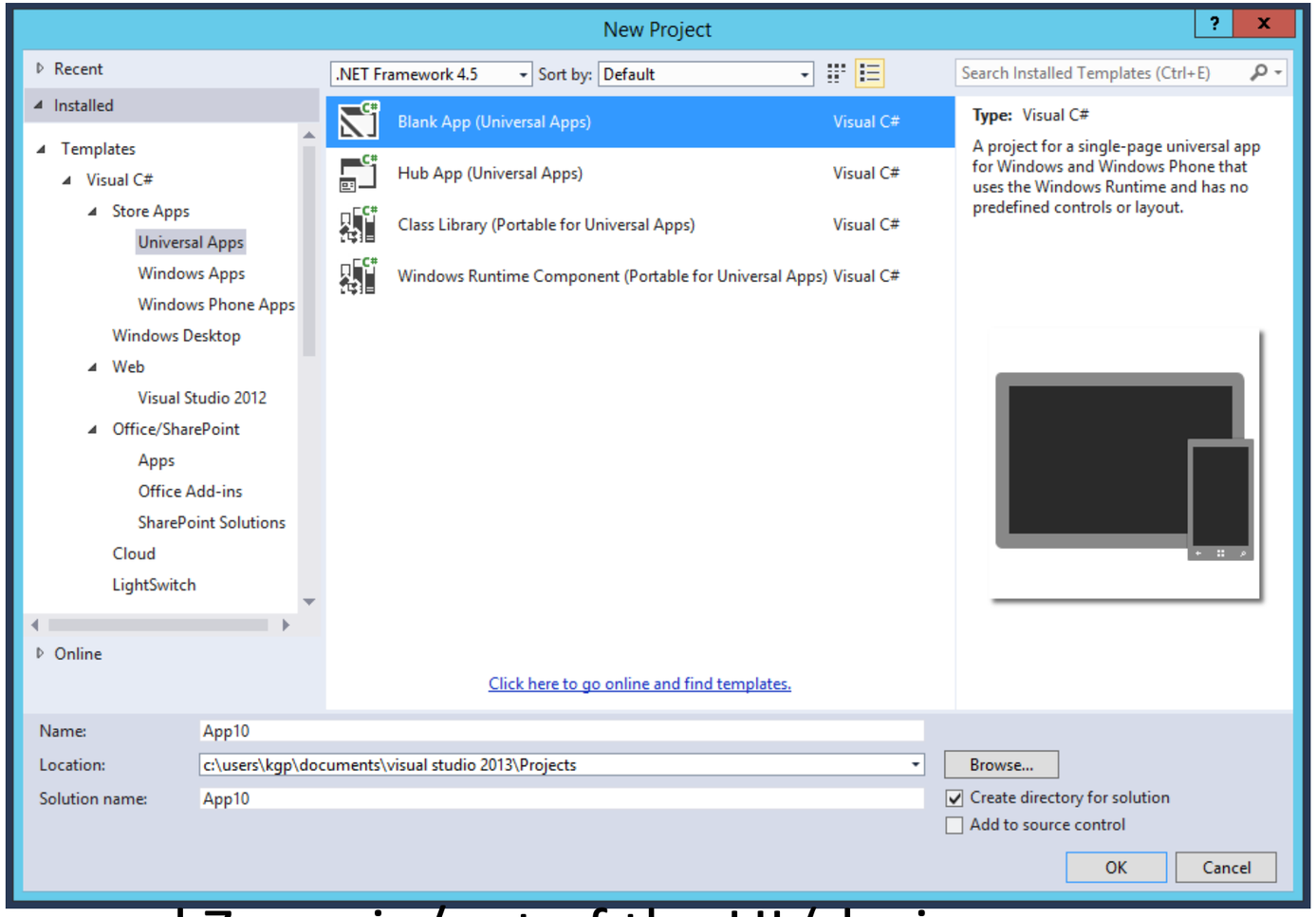
XAML + Visual Studio

- Εργασία σε 3 επίπεδα: XAML, Design View (Blend), C#.
- Δεν θα μάθουμε να γράφουμε XAML, αλλά να τη κατανοούμε.



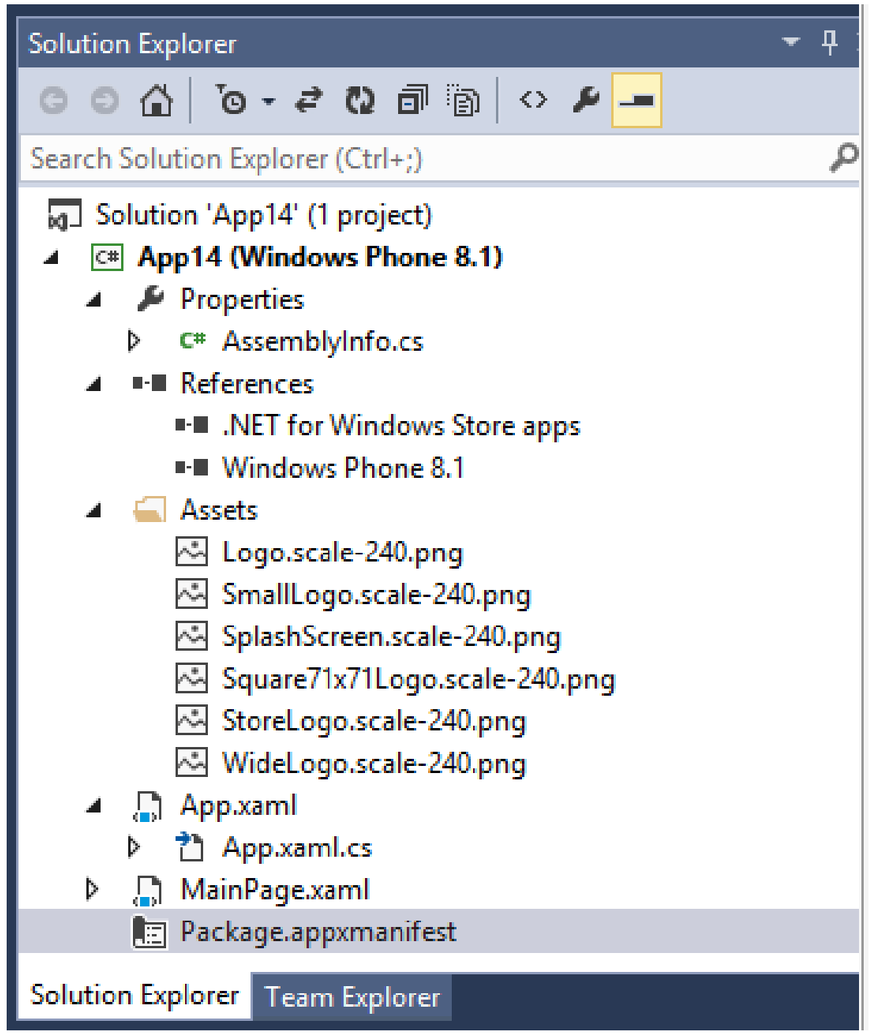
Visual studio

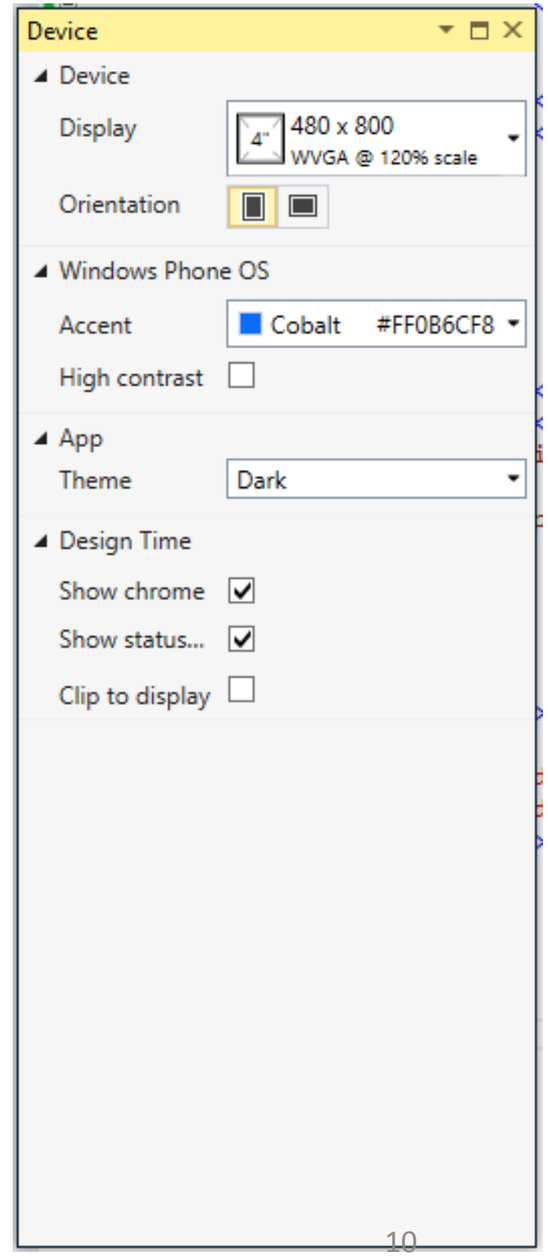
- **Projects, Types of projects**
- Code View / Design View
- **Solution Explorer**
- Properties Window
- Toolbox
- Document Outline
- **Device**
- Μερικοί χρήσιμοι χειρισμοί: Move and Zoom in/out of the UI (design view), Switch design view with code (windows), Intellisense (auto search and completion suggestions, when writing code), switch between tabs, right click on a UI component to View Source(C#)/Code(XAML), view definition, …



Visual studio

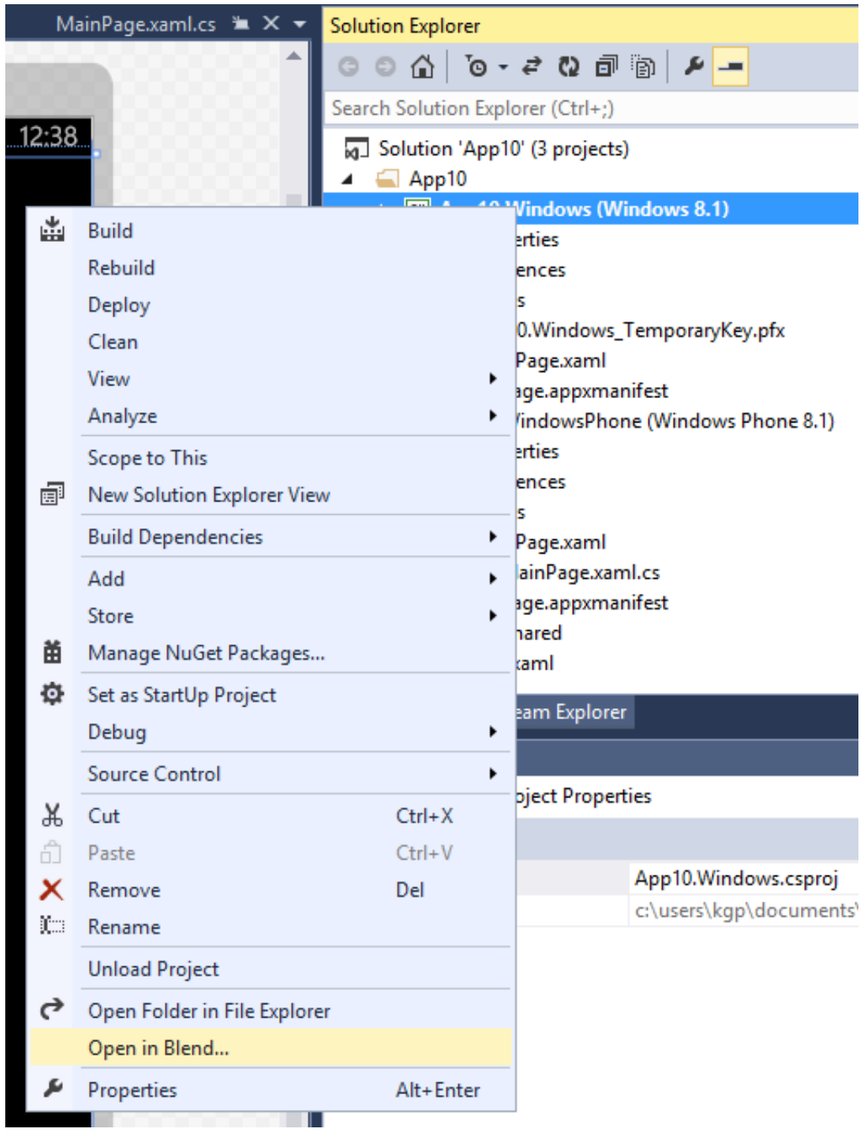
- Go through Solution Explorer, Device…



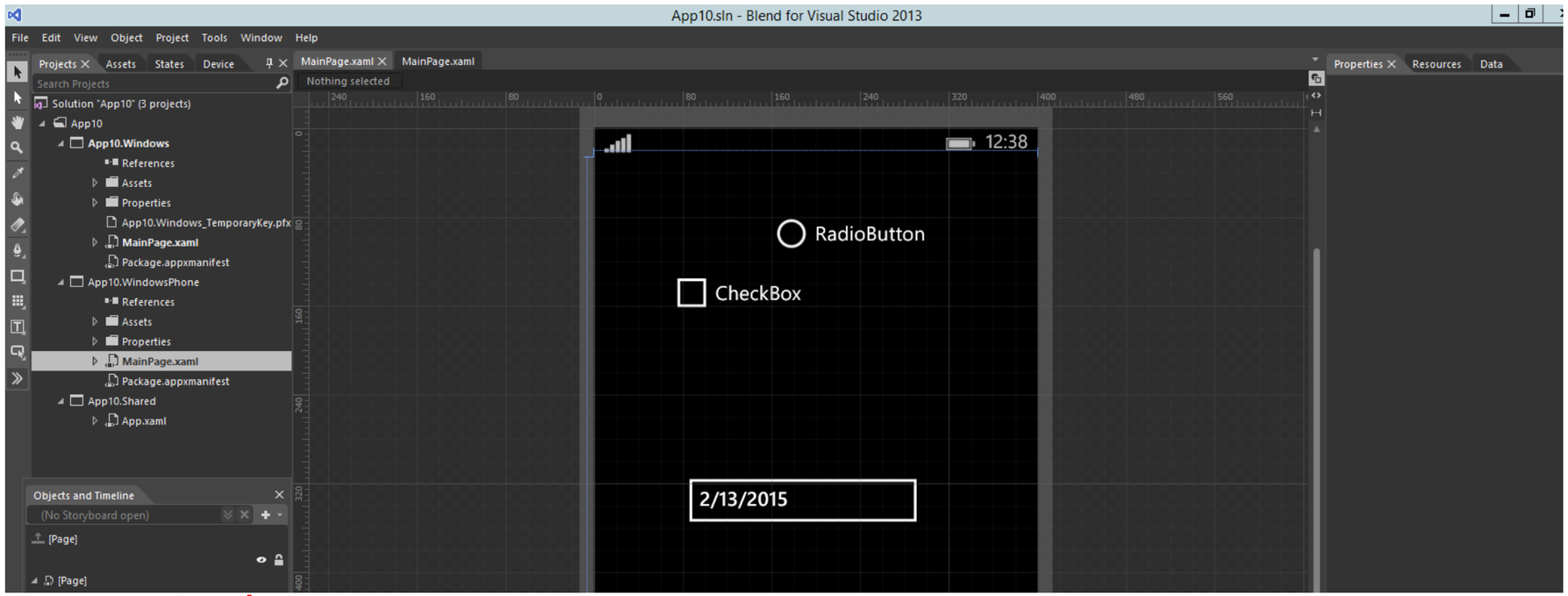


Blend

- Είναι δυνατή η παράλληλη εργασία με το Visual Studio
- Μπορείτε να ανοίξετε απευθείας ένα project από το VS -> Blend.
- Μεγάλη αντιστοιχία των εργαλείων, αν και όχι πλήρης (ακόμα).



Blend

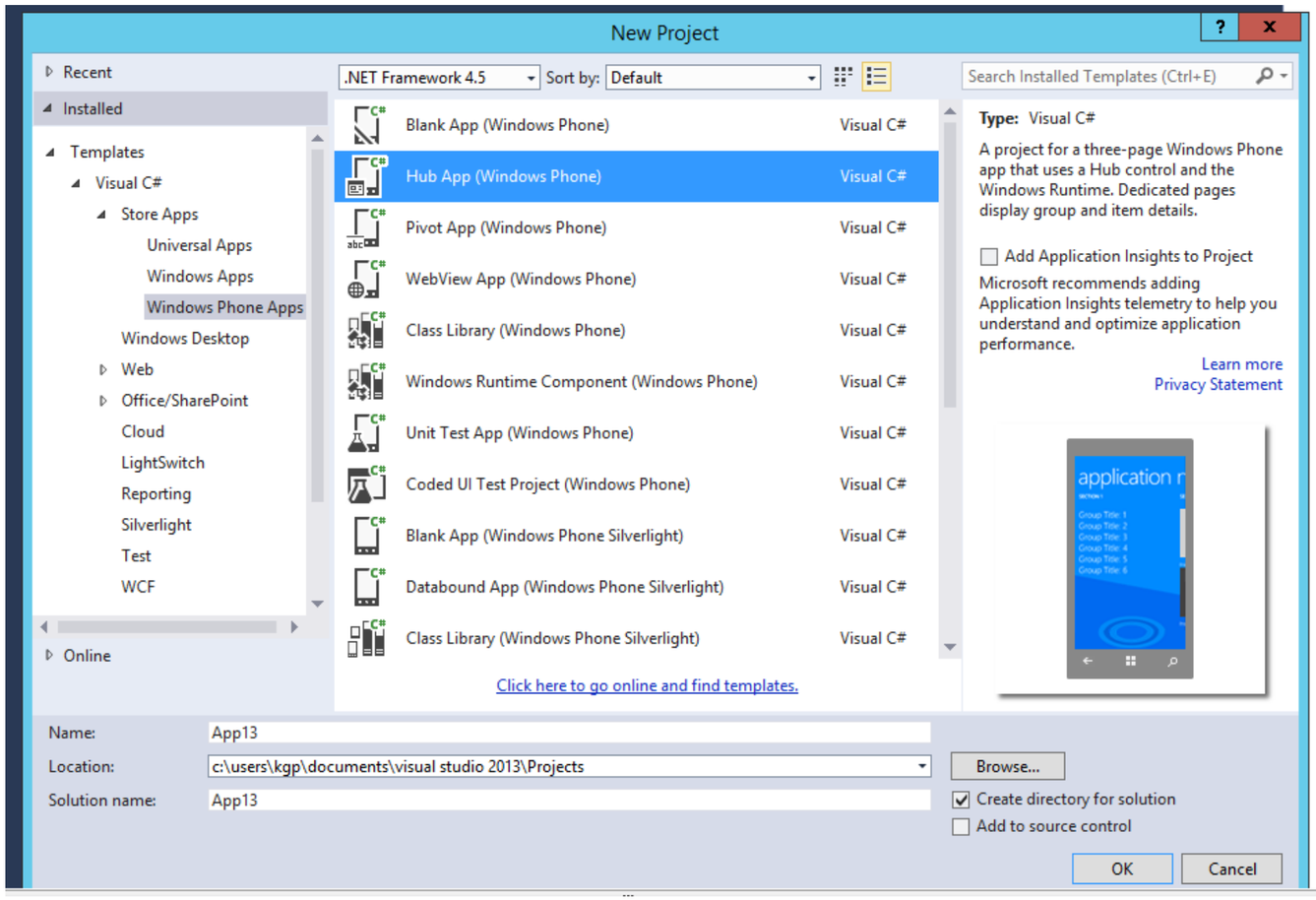


- .
- .
- .

- Device (Visual Studio: Device)
- Projects (VS: Projects)
- Assets (VS: Toolbox)
- Properties (VS: Properties)
- Objects and Timeline (Document Tree, no timeline)
- Θα χρησιμοποιούμε το Visual Studio, εκτός κι αν κάνουμε κάτι που με το blend γίνεται ευκολότερα (ιδιαίτερα: shapes + animations)

XAML, Visual Studio, Blend…

- Δημιουργία Project με Template
  - Hub app, Pivot App.
  - Χρήση του Emulator.
- Θα δούμε τη διαμόρφωση (δεδομένα) του template αργότερα (πρέπει να γράψουμε κώδικα C#).



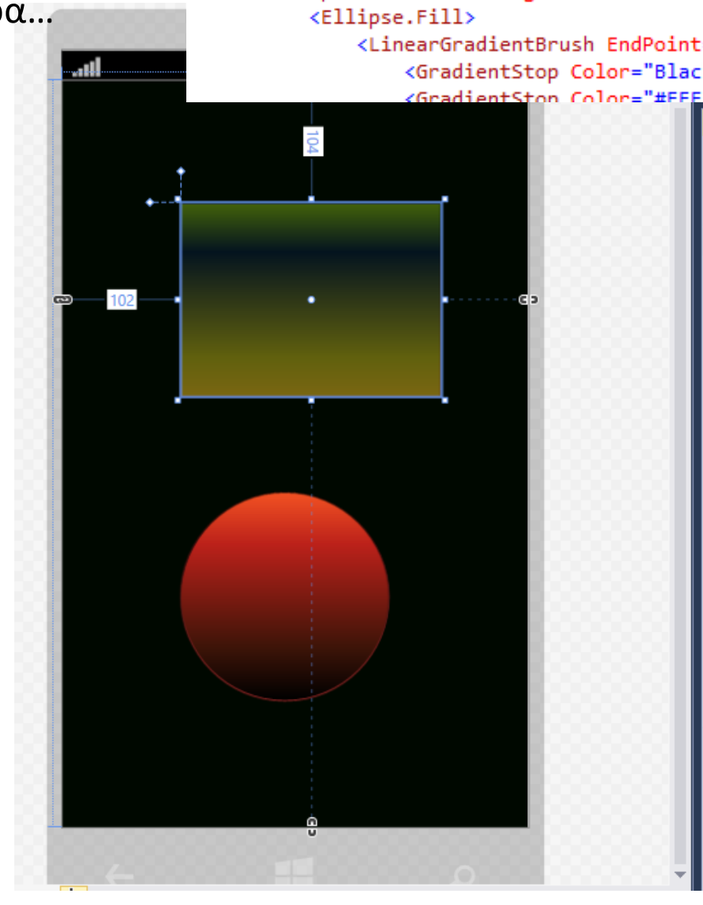
XAML, Visual Studio, Blend: Σχήματα, χρώματα, μετασχηματισμοί, κινηματική.

- Properties σε μεγαλύτερο βάθος
  - Κοινές κατηγορίες ιδιοτήτων για κάθε component.
  - Προφανώς δεν είναι απολύτως ίδιες (μεταξύ components)
  - Κάθε property μπορεί να γραφτεί/διαβαστεί σε XAML.
    - Και σε C#
  - Οι πλέον βασικές properties δημιουργούνται αυτόματα στην XAML.
- Σχήματα, χρώματα, μετασχηματισμοί, κινηματική

Σχήματα και χρώματα

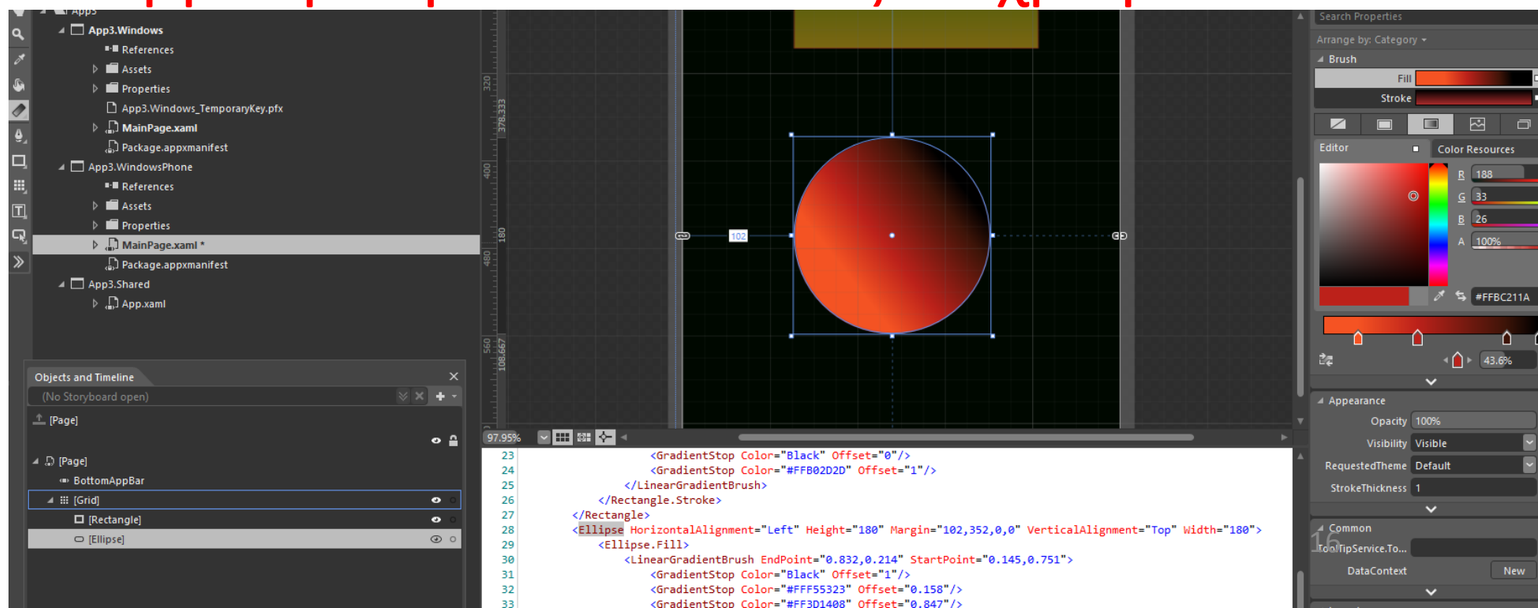
- Δημιουργήστε νέο project (windows phone). Ζωγραφίστε ένα rectangle + ellipse. Εξερεύνηση properties: Brush, Appearance, Transform.
  - Επίσης υπάρχουν interactions, events, τα οποία θα δούμε αργότερα…
- Properties Window | Brushes.
  - Color, Opacity, Gradient, Fill/Stroke.
    - Fill = no brush, the item is transparent not only to the eye but to clicks as well.
  - Eye dropper (you can select color from everywhere)
  - Color Resources - Reset (to start from scratch)
  - Gradient, gradient stops (horizontal bar)
- Από εδώ καθορίζετε τα χρώματα σας με μεγάλη ακρίβεια.
  - Many color formats supported, we will work with 32 bit R, G, B, and A (transparency).
  - You can change to work with other color formats…
- Η XAML ενημερώνεται αυτόματα.

```
<Grid Background="#FF000800">
    <Rectangle HorizontalAlignment="Left" Height="167" Margin="102,104,0,0" V
        <Rectangle.Fill>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="#FF87BD14"/>
                <GradientStop Color="#FFF5C523" Offset="1"/>
                <GradientStop Color="#FF08203D" Offset="0.253"/>
                <GradientStop Color="#FFC1B91B" Offset="0.796"/>
            </LinearGradientBrush>
        </Rectangle.Fill>
        <Rectangle.Stroke>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="Black" Offset="0"/>
                <GradientStop Color="#FFB02D2D" Offset="1"/>
            </LinearGradientBrush>
        </Rectangle.Stroke>
    </Rectangle>
    <Ellipse HorizontalAlignment="Left" Height="180" Margin="102,352,0,0" Ver
        <Ellipse.Fill>
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                <GradientStop Color="Black" Offset="1"/>
                <GradientStop Color="#FFF55323"/>
```



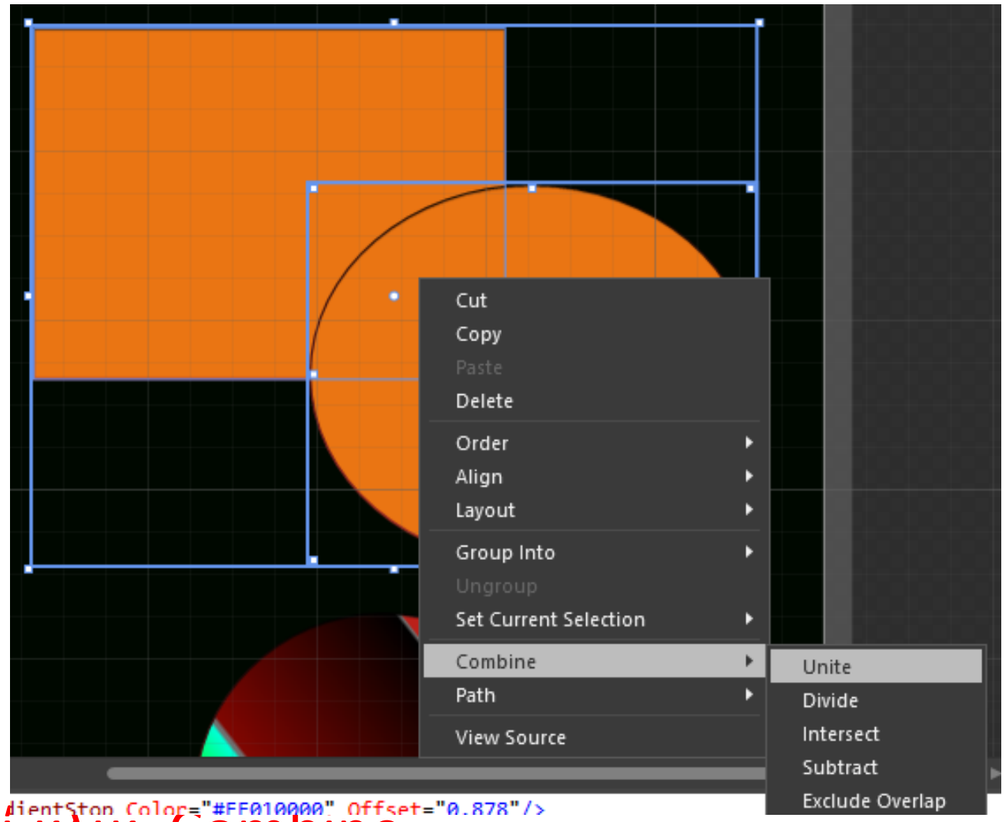
Σχήματα και χρώματα

- Δημιουργήστε έλλειψη με χρωματική κλίση (gradient) στο Visual Studio.
- Ανοίξτε το project από το Blend (From Solution Explorer, Open in Blend) και εργαστείτε με το gradient tool.
- Επίσης, με το Gradient eye dropper μπορείτε να επιλέξετε χρώματα από οπουδήποτε…
- Επίσης, μπορείτε να εισάγετε εικόνες από Photoshop, Illustrator στο Blend (File, Import).



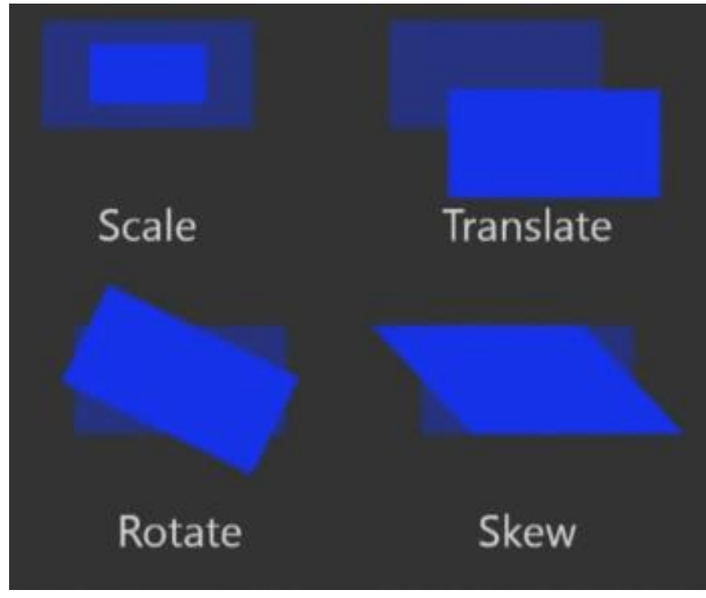
Σχήματα και χρώματα

- Σχήματα, βασικοί συνδυασμοί:
  - Unite
  - Divide
  - Intersect
  - Subtract
  - Exclude Overlap
- Για βασικές ενέργειες, στο Blend:
  επιλέγετε τα σχήματα μαζί (με Ctrl) και κάνετε δεξί κλικ, Combine…
- Το σχήμα μετατρέπεται σε μονοπάτι (path), δηλαδή προσδιορίζεται από τις ακμές του και τους τρόπους μετάβασης (γραμμή, τόξο).
  - Βλ. XAML
- Η XAML ενημερώνεται αυτόματα… το πρότυπο του path είναι SVG (Scaleable Vector Graphics) (XML-based), άρα μπορεί να μεγεθυνθεί/σμικρυνθεί gracefully.



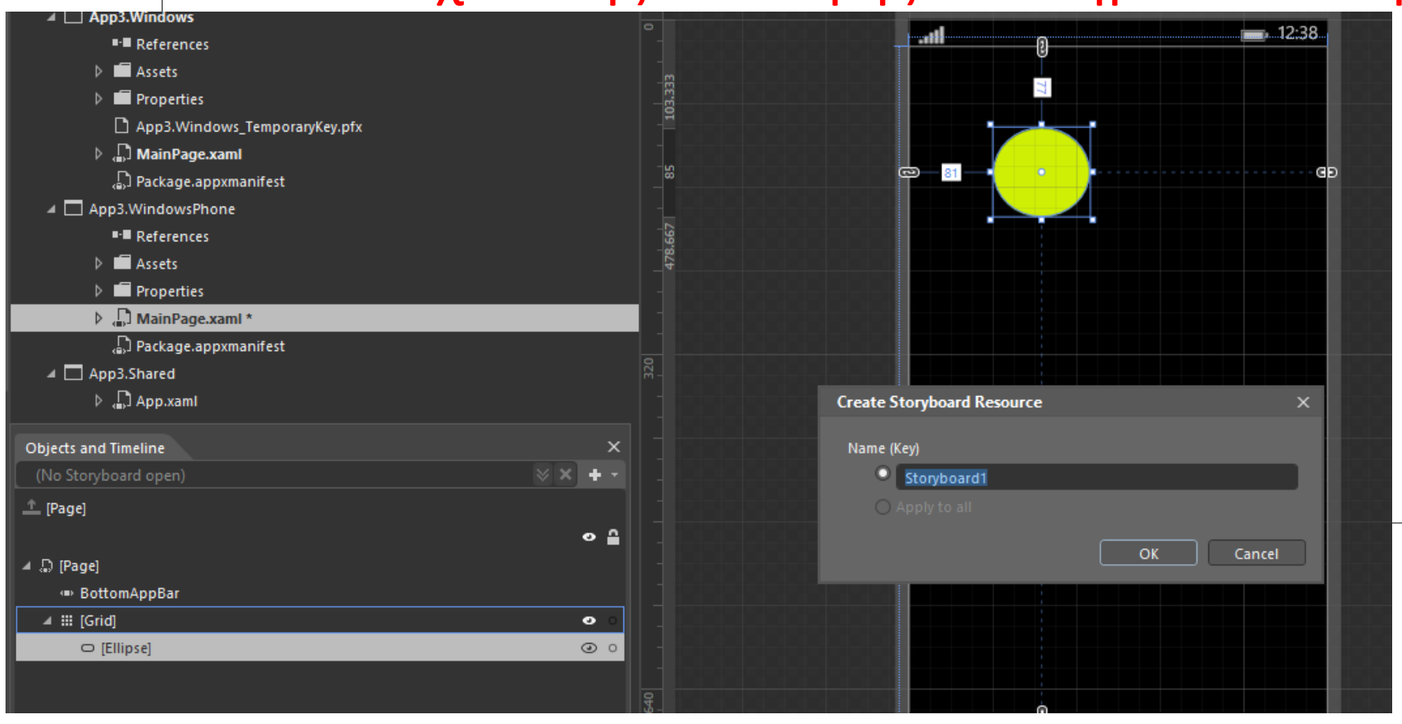
Μετασχηματισμοί

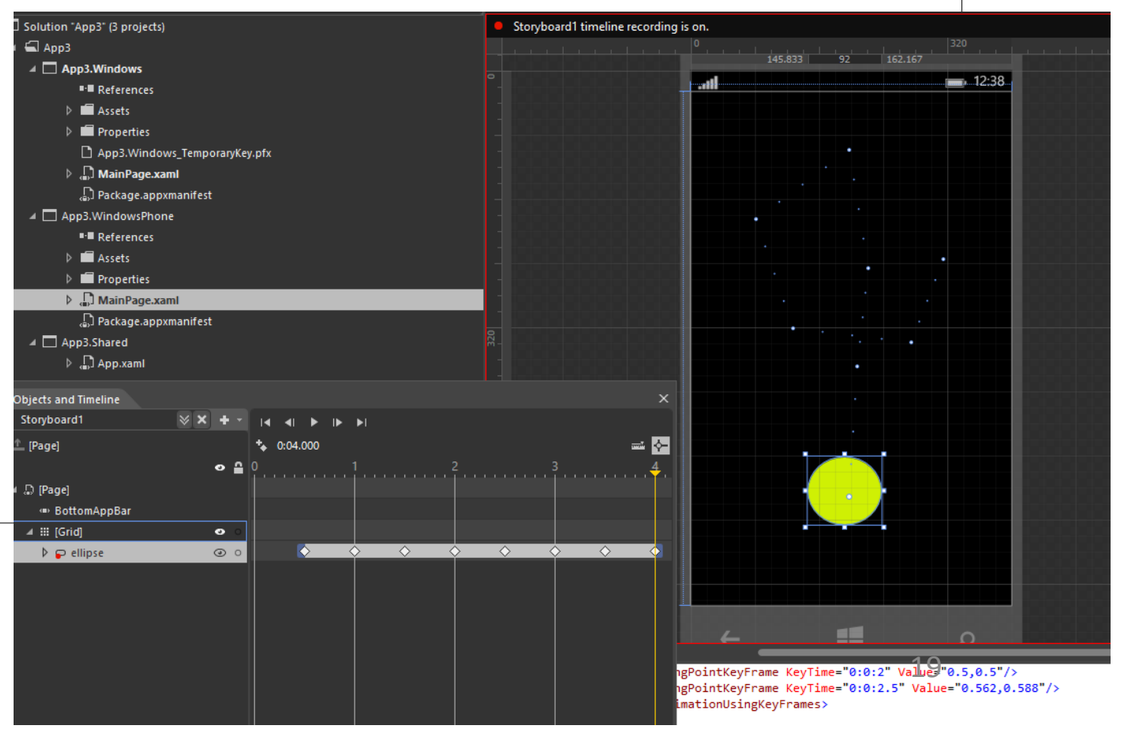
- Μετασχηματισμοί και κινηματική.
  - Συσχετισμένοι (αντιστρέψιμοι στο επίπεδο)
    - Scale
    - Translate
    - Rotate
    - Skew
  - Μη συσχετισμένοι (γίνονται σε σχέση με σημείο εκκίνησης)
    - Projection
- Χρησιμοποιούνται για κινηματική (animations), στο Blend.
  - Properties, Transform.



Κινηματική (animations)

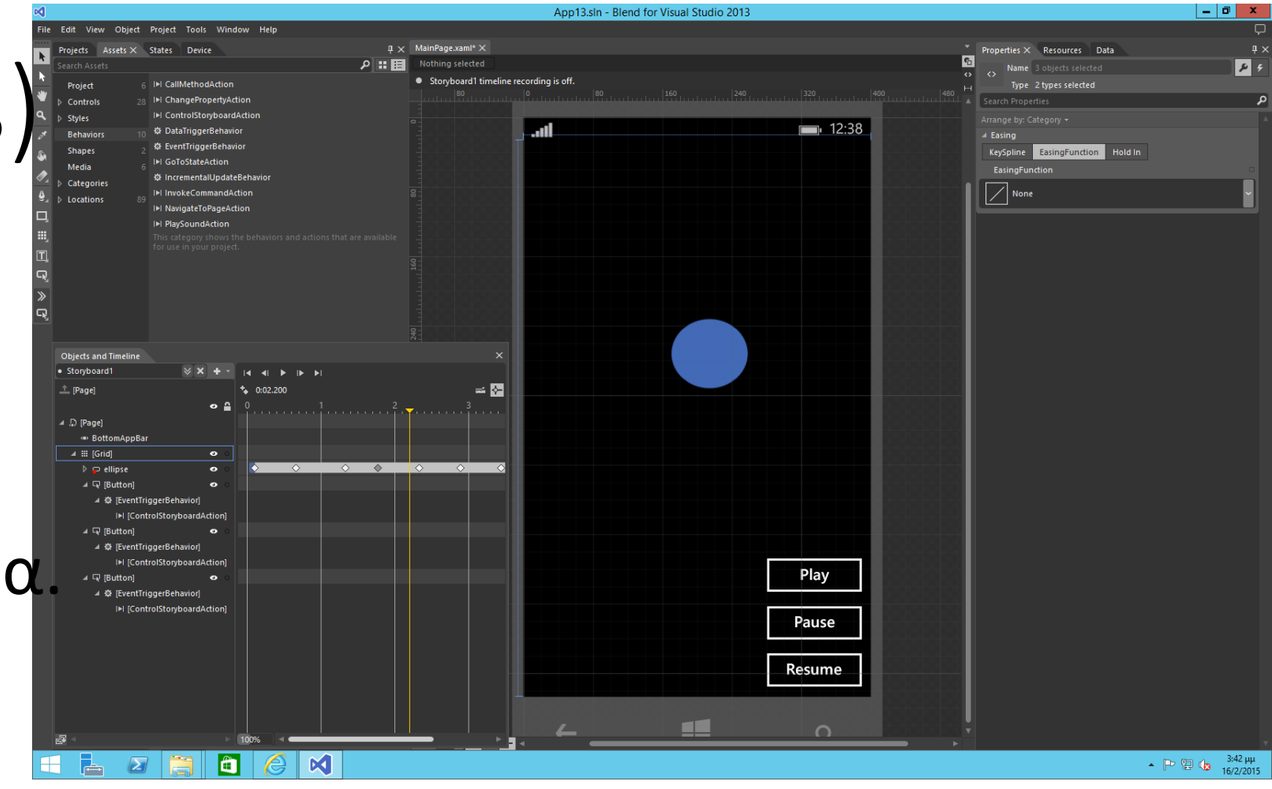
- Η σχεδίαση animations υποστηρίζεται από το Blend (όχι Visual Studio – εκτός φυσικά αν γράψουμε κώδικα).
  - Window | Objects and Timeline | + | Storyboard.
  - Εντός του timeline, σημειώνουμε χρονικές στιγμές και τοποθετούμε τα στοιχεία της διεπαφής στο σημείο που πρέπει.





Κινηματική (animations)

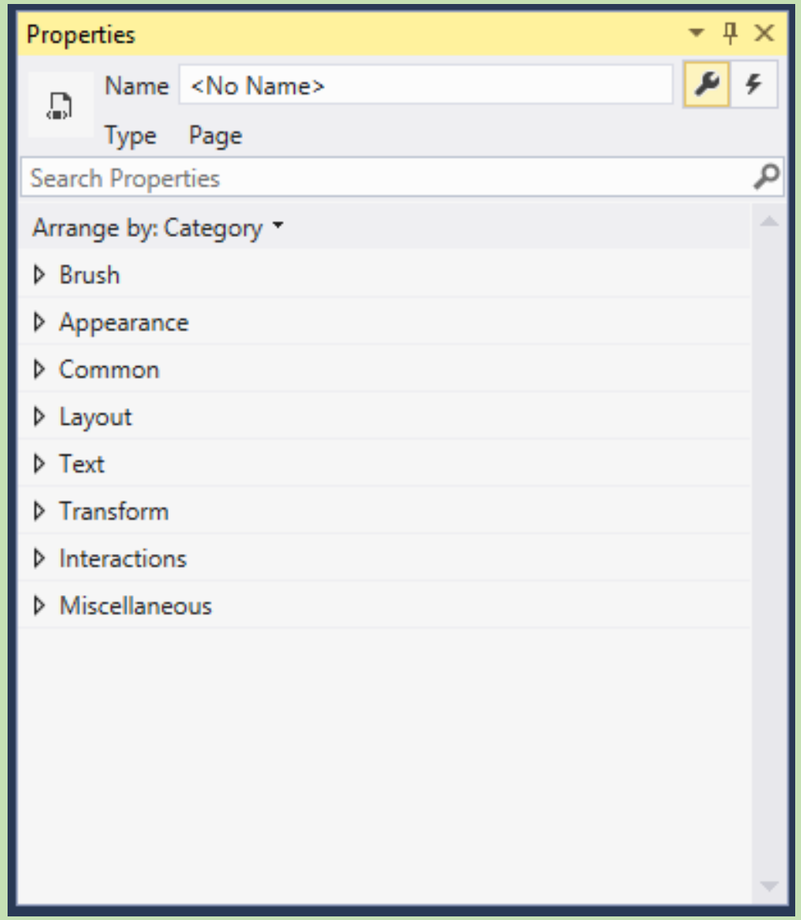
- Κατά την κινηματική μπορούμε να αλλάζουμε όχι μόνο τη **θέση** του αντικειμένου, αλλά το **χρώμα** του, να μετασχηματίζουμε το **σχήμα**, κ.α.



- Επίσης, μπορούμε να φτιάξουμε απλές εφαρμογές κινηματικής χωρίς κώδικα στο Blend, απλά βάζοντας κάποια κουμπιά χειρισμού.
  - Προσθέστε κουμπιά Start, Stop, Resume στη διεπαφή (Assets|Controls).
  - Προσθέστε συμπεριφορά στα κουμπιά (Assets | Behaviours | ControlStoryboardAction), και προσδιορίστε την για κάθε κουμπί από τα Objects and Timeline + Properties.

Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Panels
  - Canvas
  - StackPanel
  - Grid
- Properties for layout and appearance...

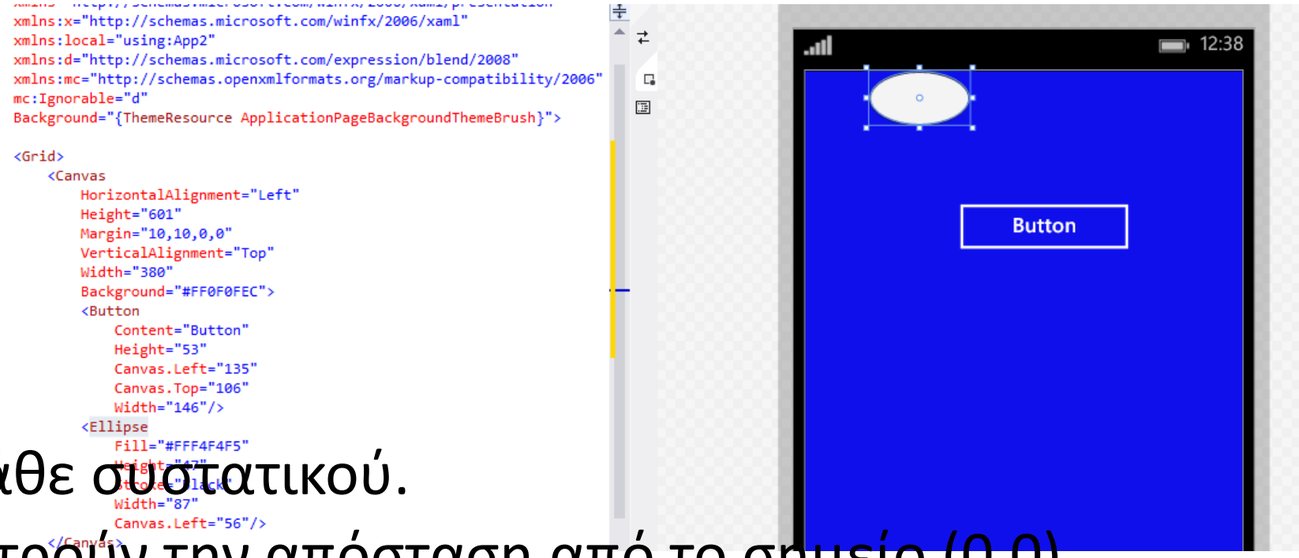


Διάταξη και συστατικά διεπαφής (Layout and UI components)

```
public class Panel : FrameworkElement
 {

        protected Panel();
        public Brush Background { get; set; }
        public UIElementCollection Children { get; }
        public TransitionCollection ChildrenTransitions { get; set; }
        public bool IsItemsHost { get; }


        // from FrameworkElement
        protected virtual Size ArrangeOverride(Size finalSize);
        protected virtual Size MeasureOverride(Size availableSize);

}
```

- Panel (‘πλαίσιο’)
  - Ένα διάφανο (transparent) πλαίσιο που περιέχει συστατικά διεπαφής.
- Πρόκειται για container, χρησιμοποιείται για να διαταχθούν τα στοιχεία.
- Τα συστατικά της διεπαφής τοποθετούνται εσωτερικά ενός panel.
- Στη C#, XAML, δεν φτιάχνουμε απευθείας ένα panel, αλλά κάποια κλάση αυτού. Βασικές κλάσεις panel:
  - Canvas, StackPanel, Grid.
  - Άλλα panels: VirtualizingStackPanel (Win apps), ItemsWrapGrid (MS Phone apps), VariableSizedAppGrid (MS Phone apps).

Διάταξη και συστατικά διεπαφής (Layout and UI components)

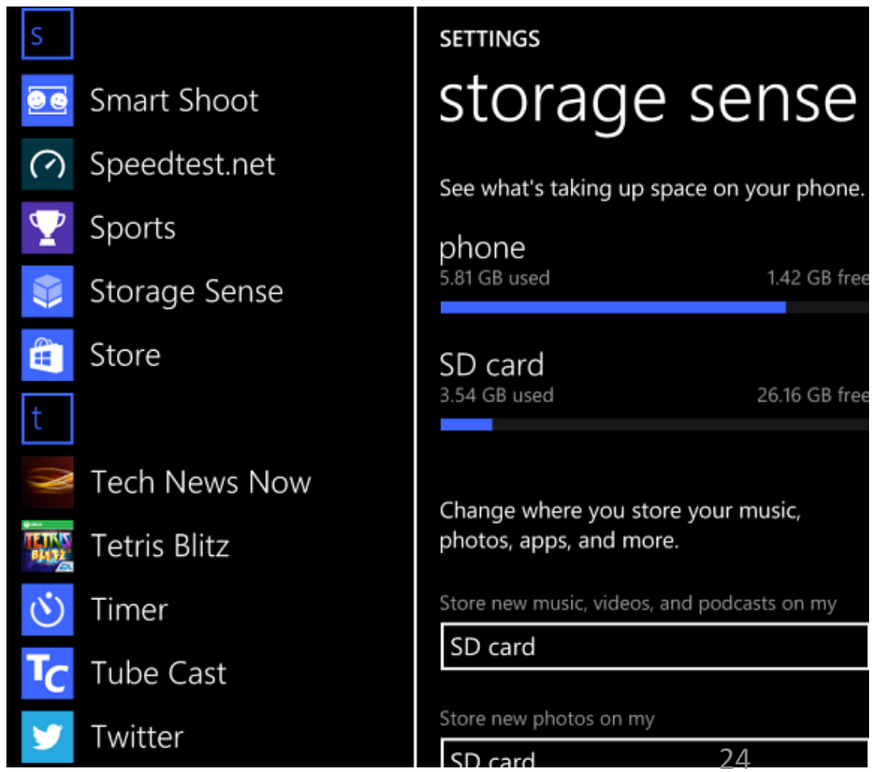


- **Καμβάς – Canvas.**
- Τοποθέτηση συστατικών με προσδιορισμό των συντεταγμένων τους.
- Εξετάζεται το πάνω αριστερά σημείο κάθε συστατικού.
- Οι ιδιότητες Canvas.Left, Canvas.Top μετρούν την απόσταση από το σημείο (0,0) του καμβά (επίσης το πάνω αριστερά) σε Pixels.
- Φτιάχνουμε έναν καμβά και τοποθετούμε 1-2 συστατικά. Παίζουμε με τη θέση τους και βλέπουμε το αποτέλεσμα στην XAML.
- Οι διαστάσεις είναι οριστικές (absolute positioning)! Ο καμβάς δεν προσαρμόζεται σε διαφορετικά μεγέθη οθόνης αυτόματα!
  - Χρησιμοποιείται σε παιχνίδια με γραφικά (π.χ. «η πίστα»), ή όταν η εφαρμογή μας κάνει ζωγραφική…

Διάταξη και συστατικά διεπαφής (Layout and UI components)

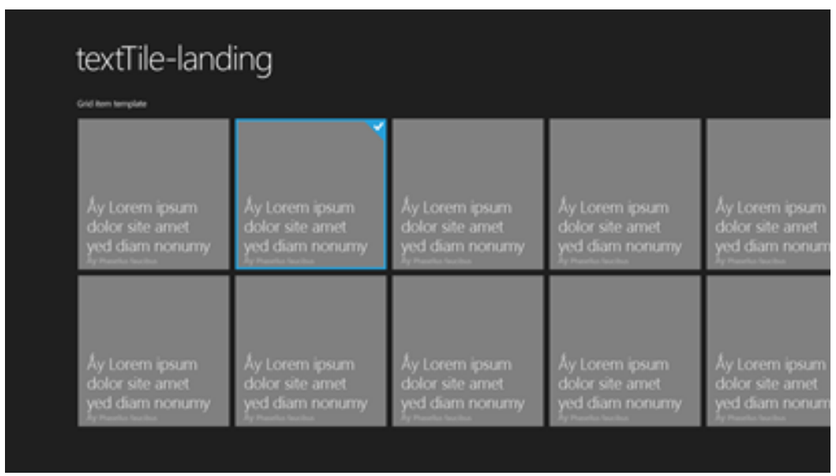
- StackPanel (‘πλαίσιο-στοίβα’).
  - Τοποθετεί τα στοιχεία το ένα μετά το άλλο σε προσανατολισμό.
    - Orientation (προσανατολισμός):
      - Vertical
      - Horizontal
- Φτιάχνουμε ένα StackPanel με buttons και του αλλάζουμε το orientation.

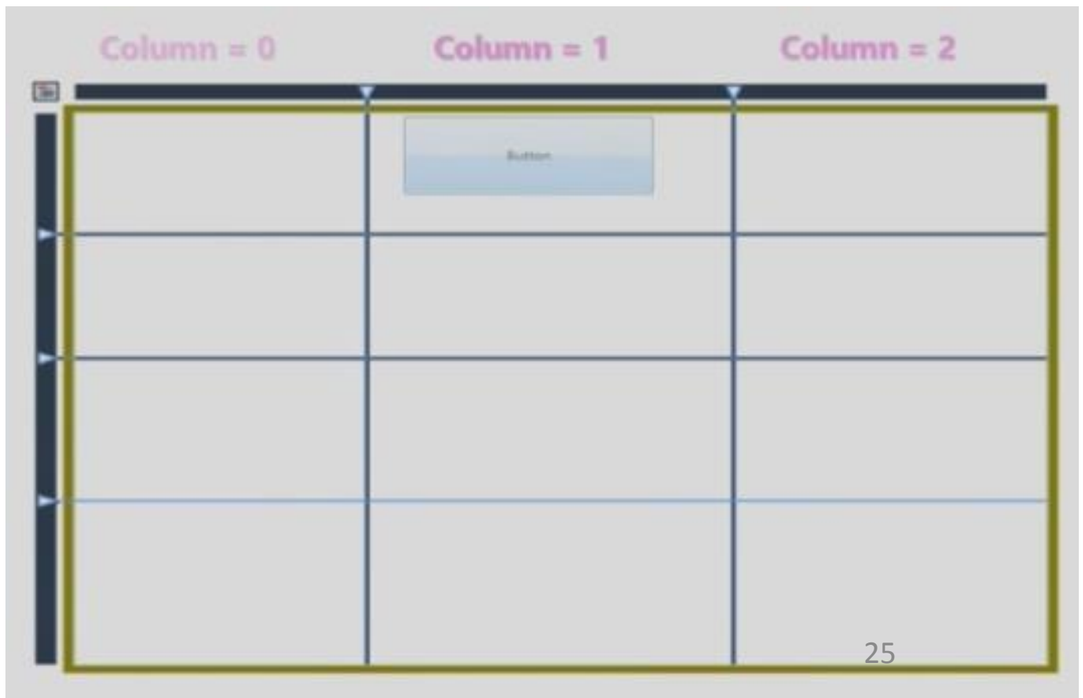
```
<StackPanel
    Height="207"
    Canvas.Left="126"
    Canvas.Top="244"
    Width="175"
    Orientation="Vertical">
    <Button Content="Button" Horizo
    <Button Content="Button" Horizo
    <Button Content="Button" Horizo
</StackPanel>
```



Διάταξη και συστατικά διεπαφής (Layout and UI components)

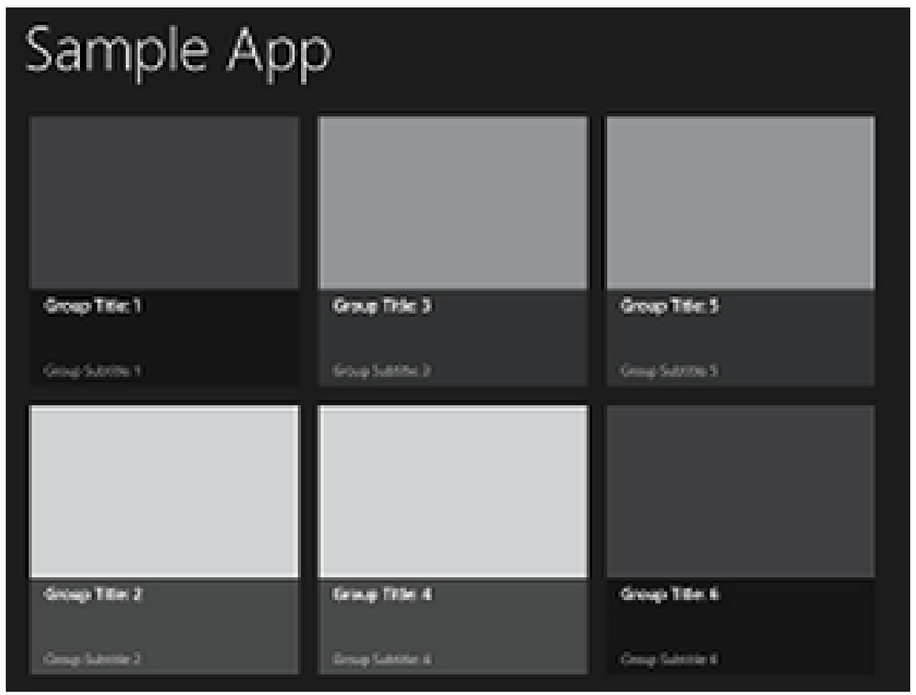
- Grid (πλέγμα)
  - Με Γραμμές (GridRows) και στήλες (GridColumns)
    - Προσοχή! Η αρίθμηση τους ξεκινάει από το 0.
    - Κάθε στοιχείο διεπαφής μπορεί να ανήκει σε 1 ή περισσότερα ‘κελιά’ GridRowSpan, GridColumnSpan.
- Η βάση των περισσότερων διεπαφών..

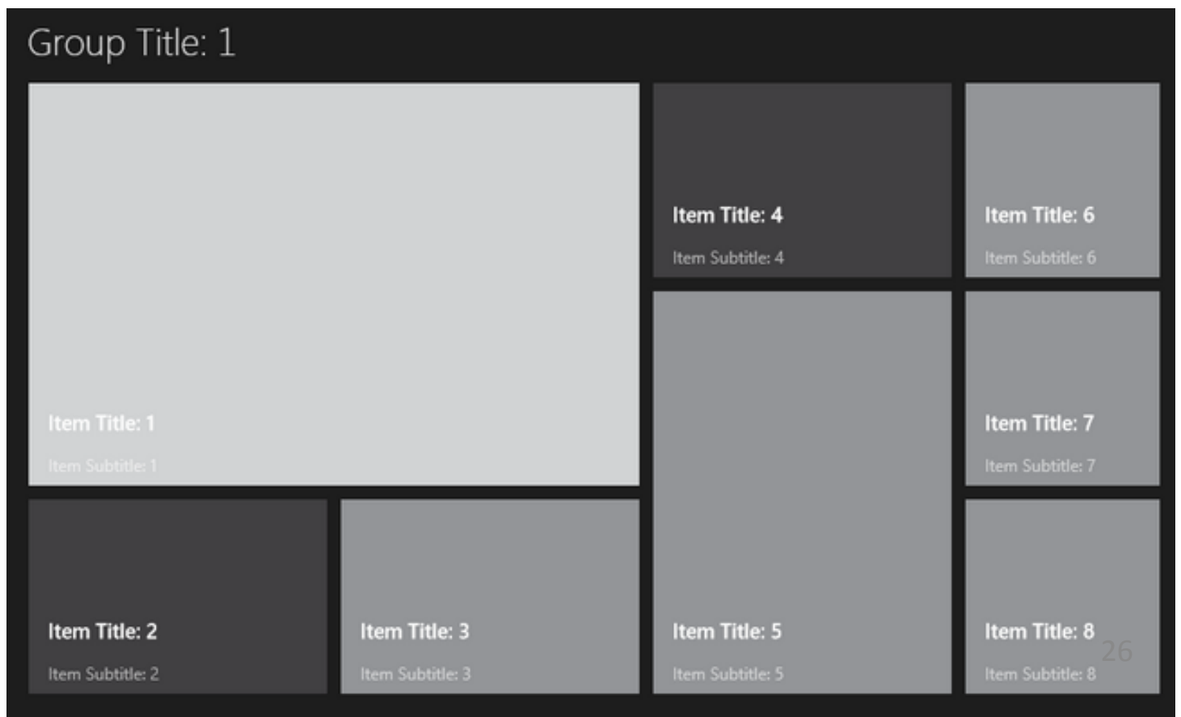




Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Grid (πλέγμα) - Η βάση των περισσότερων διεπαφών..
  - Ένα κλασικό pattern είναι να ορίζουμε ένα Grid, και εντός των κελιών του να περιλαμβάνουμε StackPanel.
  - Σημειώστε: το StackPanel μπορεί να περιλαμβάνει και άλλο Grid, κ.ο.κ.

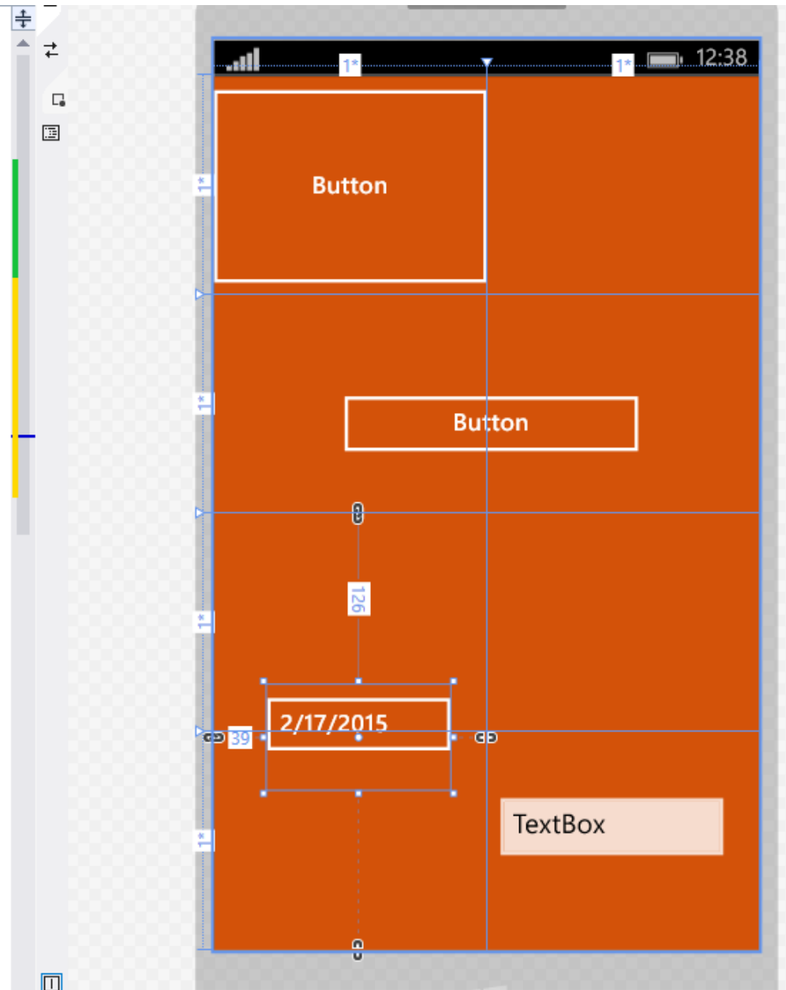




Διάταξη και συστατικά διεπαφής (Layout and UI components)

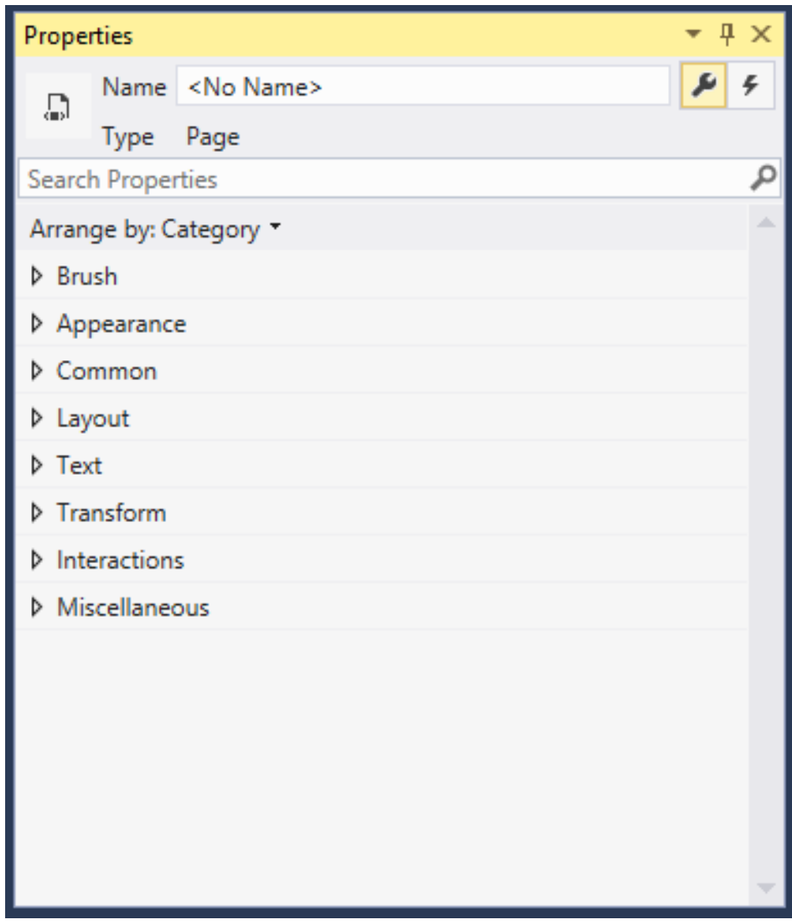
- Νέο project.
- Φτιάξτε ένα Grid με 4 γραμμές, 2 στήλες
  - Properties | RowDefinitions, ColumnDefinitions
  - Με το ποντίκι στο χάρακα γύρω από το Grid… (παρατηρείστε την XAML ότι σε αυτήν την περίπτωση τα μεγέθη είναι αναλογικά)
- Τοποθετήστε συστατικά διεπαφής.
  - Ή και stackpanel
- Δείτε στην XAML τις ιδιότητες τους, κάθε φορά που αλλάζετε το σχήμα, τη θέση τους, κ.α.

```
<Page
    x:Class="App14.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:App14"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid Background="#FFD35209">
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Button
            Content="Button"
            HorizontalAlignment="Stretch" VerticalAlignment="Stretch"/>
        <Button
            Grid.ColumnSpan="2" Content="Button"
            HorizontalAlignment="Left" Height="59"
            Margin="96,65,0,0" Grid.Row="1"
            VerticalAlignment="Top" Width="215"/>
        <TextBox
            Grid.Column="1"
            HorizontalAlignment="Left" Height="42"
            Margin="10,48,0,0" Grid.Row="3"
            TextWrapping="Wrap" Text="TextBox"
            VerticalAlignment="Top" Width="163"/>
        <DatePicker
            HorizontalAlignment="Left" Height="77"
            Margin="39,126,0,0" Grid.Row="2"
            Grid.RowSpan="2" VerticalAlignment="Top"
            Width="134"/>
    </Grid>
</Page>
```

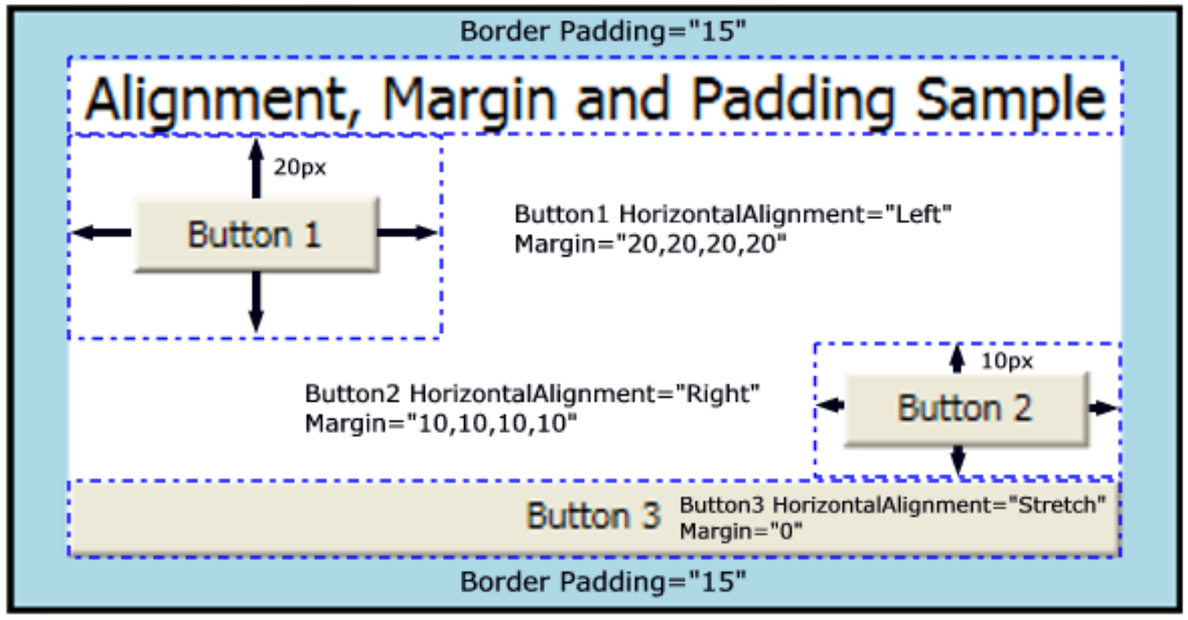


Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Properties Appearance, Common, Layout, Text σε μεγαλύτερο βάθος
  - Κοινές κατηγορίες για κάθε component.
- Εντός των κατηγοριών, όχι απολύτως ίδιες (μεταξύ components)
- Κάθε property μπορεί να γραφτεί/διαβαστεί σε XAML.
  - Και σε C#
- Οι πλέον βασικές properties δημιουργούνται αυτόματα στην XAML.



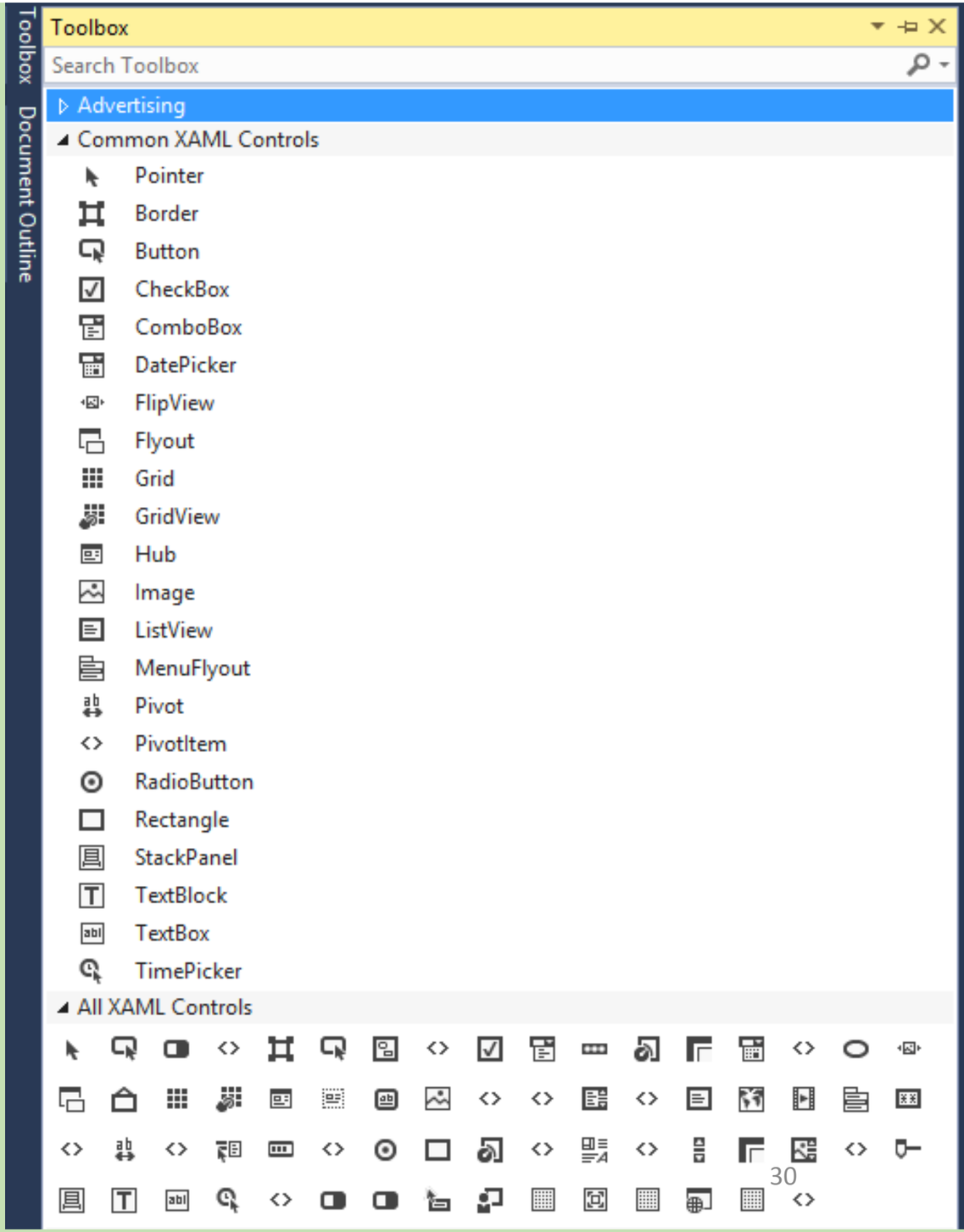
Διάταξη και συστατικά διεπαφής (Layout and UI components)



- Εξερεύνηση-εξήγηση των Properties μία-προς-μια: Appearance, Common, Layout, Text.
- Βασικά properties για μορφή και διάταξη:
  - Height, Width (*pixels*)
  - Text/Content (*string*)
  - HorizontalAllignment, VerticalAlignment (*“Top”, “Bottom”, “Left”, “Right”, “Stretch”*)
  - Margin: η <u>εξωτερική</u> απόσταση του στοιχείου από τα γειτονικά του.
    - Η τιμή του είναι 4 αριθμοί, που με τη σειρά προσδιορίζουν τα *left, top, right, bottom. Αν υπάρχει μόνιο μία τιμή, τότε εξ ορισμού αφορά το left, αν υπάρχουν μόνο δύο τιμές, τα left, top, κ.ο.κ.*
    - *Μπορούν να οριστούν και αρνητικά margins! (επικάλυψη με άλλα αντικείμενα, ή εκτός ορατότητας της οθόνης)*
  - Padding: ο κενός χώρος <u>εσωτερικά</u> του αντικειμένου.
    - Αν είναι μεγάλος, τα αντικείμενα «σπρώχνονται» προς τα κάτω.

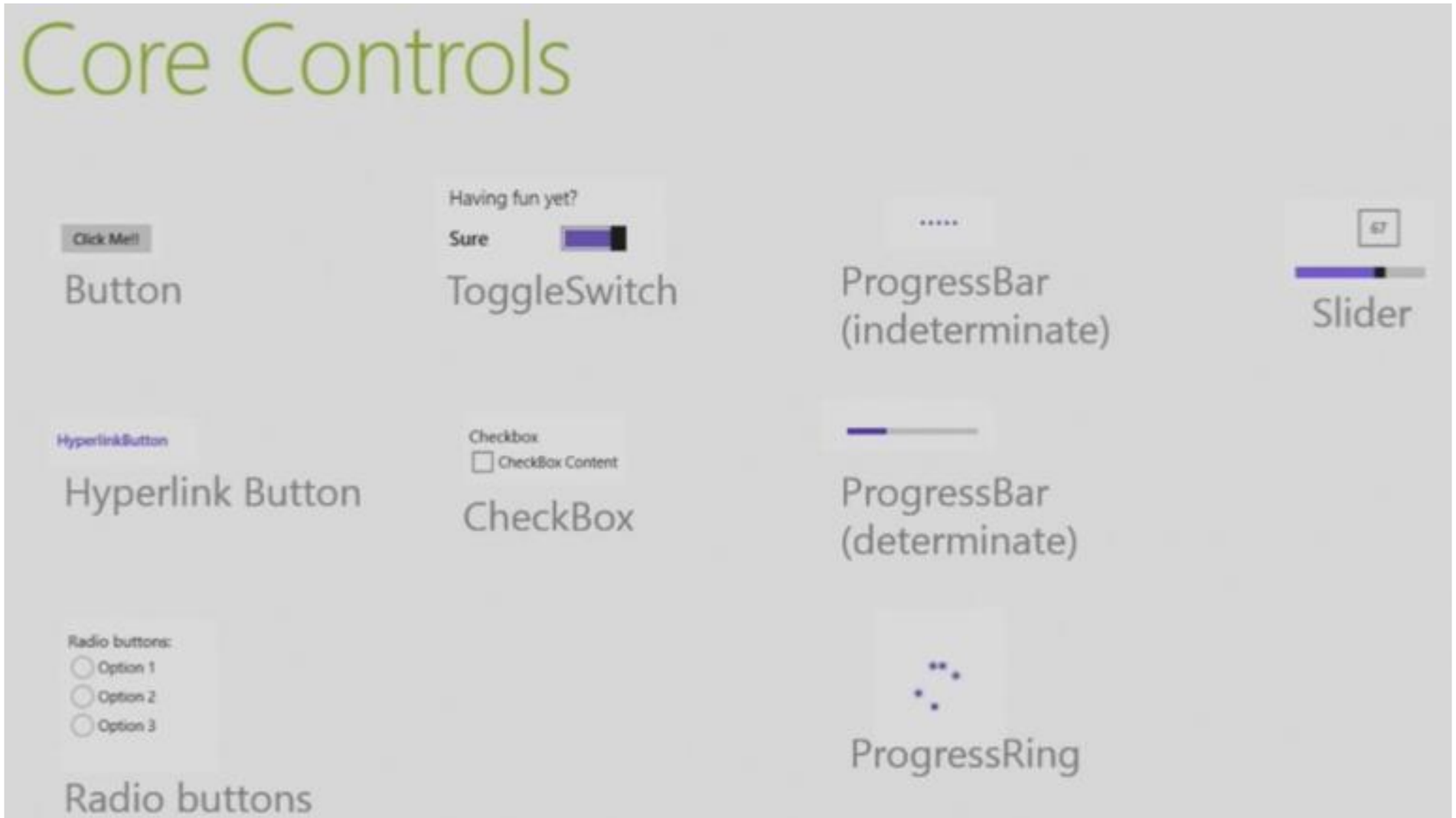
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Τα συστατικά διεπαφής σε μεγαλύτερο βάθος…
  - Συνδυασμός, διάταξη, επαναχρησιμοποίηση
- Αλληλεπίδραση:
  - Είσοδος, επιλογή, ανάδραση, έξοδος, κ.α.
- Τυποποιημένη Συμπεριφορά
- Σύνδεση με:
  - ‘κύριο πρόγραμμα’
  - Δεδομένα
  - Γεγονότα (events)



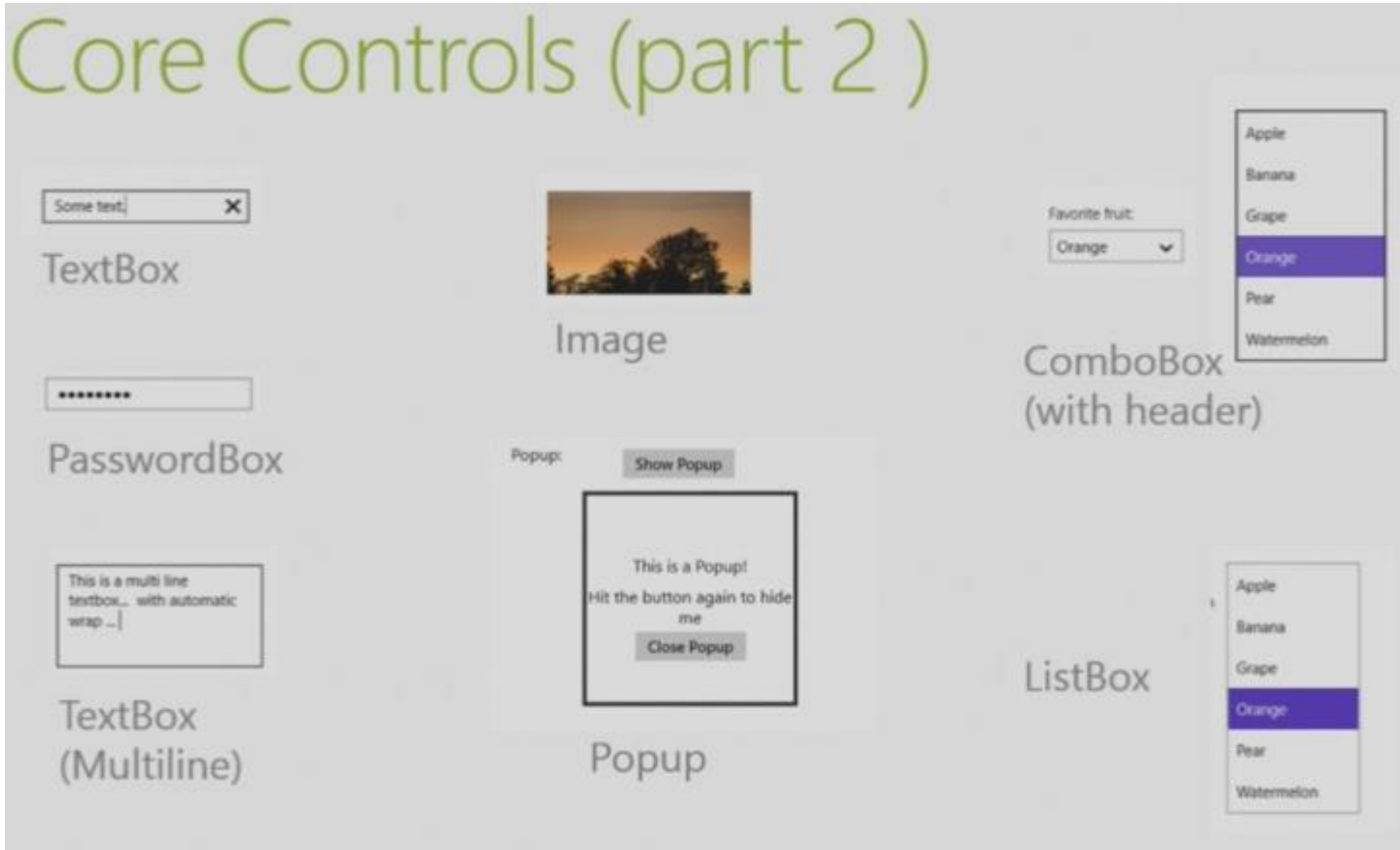
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- .



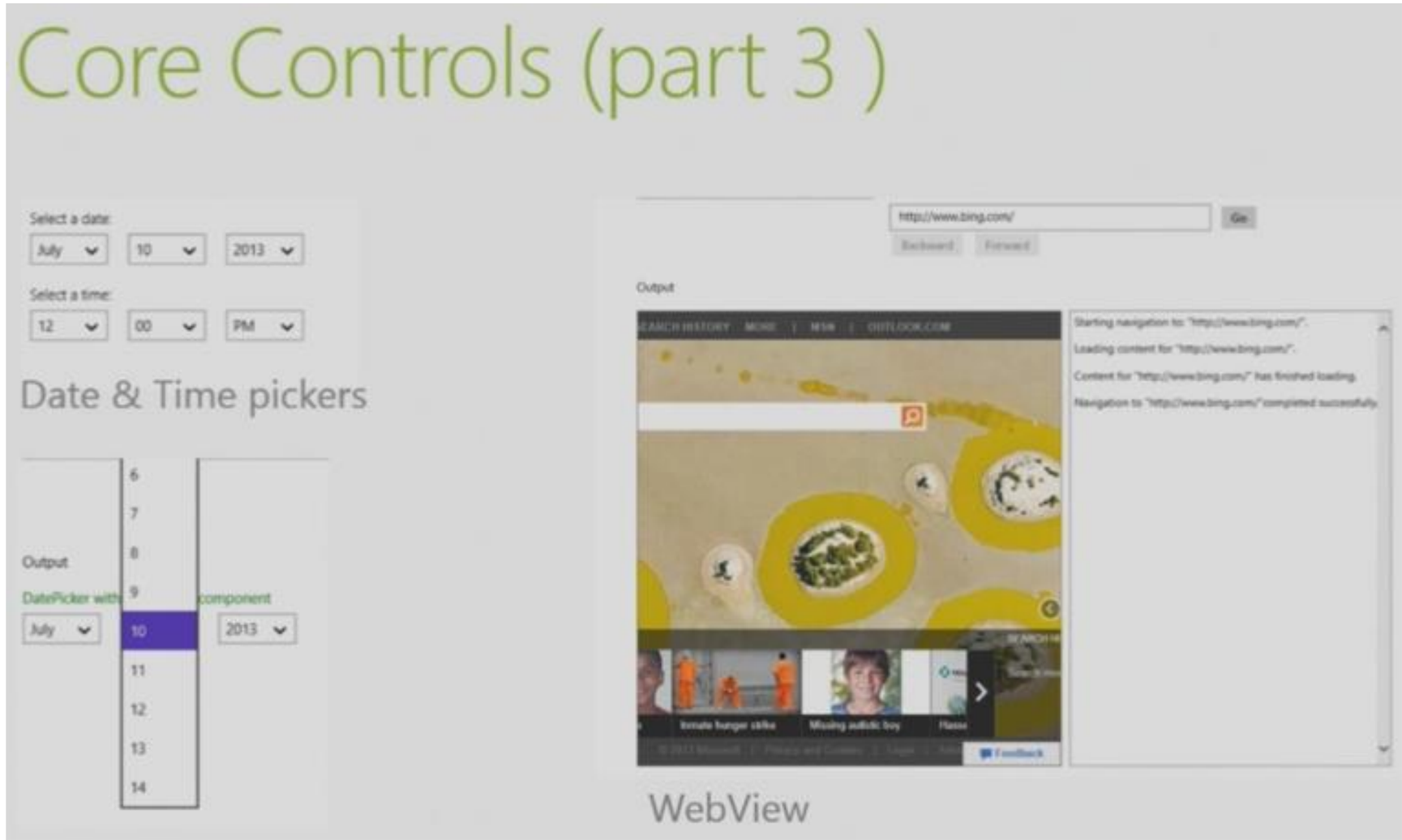
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- .



Διάταξη και συστατικά διεπαφής (Layout and UI components)

- .



Διάταξη και συστατικά διεπαφής (Layout and UI components)

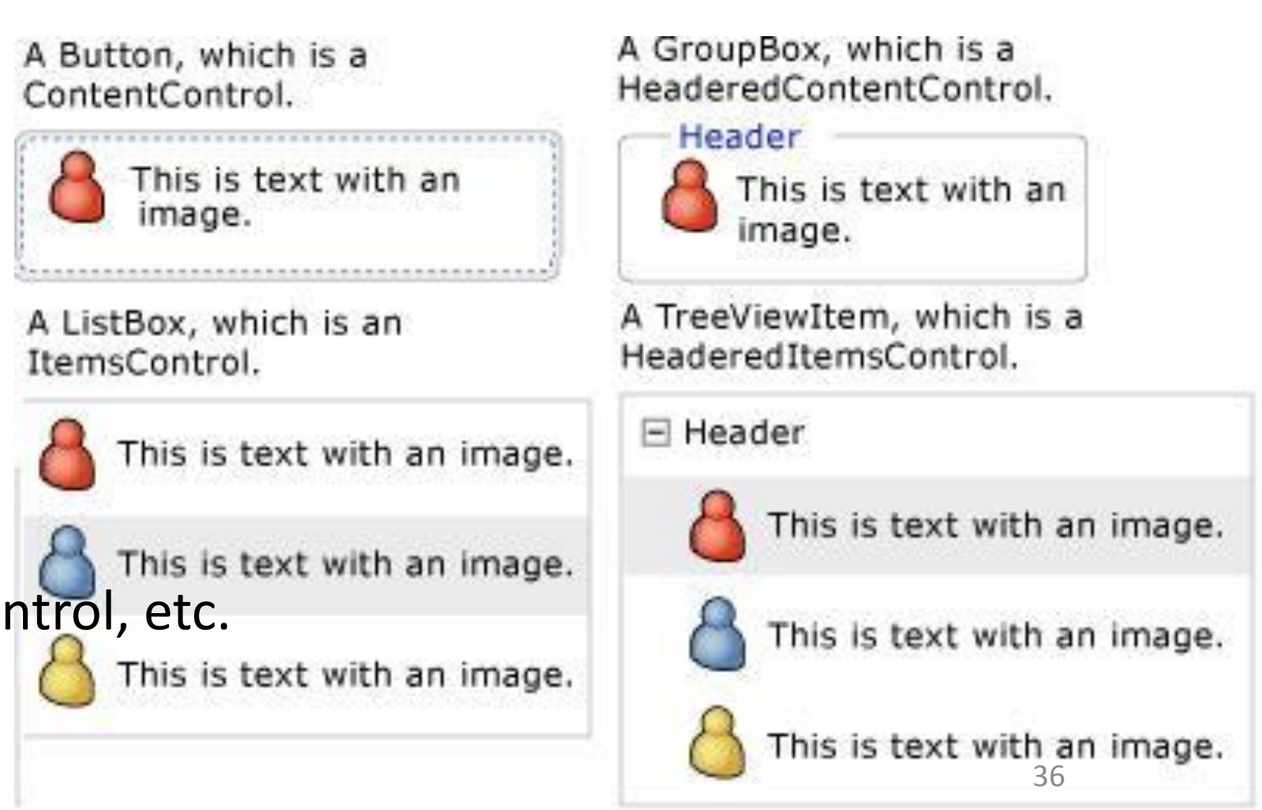
- Αλληλεπιδράσεις:
  - Εντολή (επιλογή): Button, RadioButton, Menu, HyperLinkButton (Windows Phone).
  - Είσοδος αλφαριθμητικών δεδομένων (πληκτρολόγιο): TextField, PasswordBox, Slider (!), RichEditBox.
  - Επιλογή δεδομένων (από λίστα): ComboBox, ListBox, ListView, ToggleButton.
  - Παρουσίαση δεδομένων: Textblock, Label, Image, MediaElement, WebView.
  - Ανάδραση (και μηνύματα προς το χρήστη): ProgressBar, Tooltip, PopUp.
  - Βοήθεια για τη διάταξη και πλοήγηση: Border, ScrollBar, ScrollView.
  - Ρύθμιση ημερομηνίας ή ώρας: DatePicker, TimePicker.
- Lookless UI controls!
  - Πολλά UI controls είναι εξ αρχής αόρατα στο Design View / Blend!
  - Για να τα δούμε (και να αλληλοεπιδράσουμε με αυτά) θα πρέπει να τα συνδέσουμε (προγραμματιστικά) με δεδομένα…
    - … τα οποία διαβάζονται δυναμικά από τη διεπαφή κατά την αρχικοποίηση της.

Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Τα συστατικά διεπαφής είναι πολλά…
- Υπάρχουν αρκετά έτοιμα στο Web, ελεύθερα (free), π.χ.
  - http://timheuer.github.io/callisto/
  - http://www.blendrocks.com/
- Επίσης, υπάρχουν εταιρίες που διαθέτουν όγκο συστατικών εμπορικά.
  - http://www.infoq.com/research/dotnet-web-components
- Κάθε σχεδιαστής, καθώς εργάζεται και αποκτά πείρα, ‘χτίζει’ τη δική του ‘βάση’ με UI controls.
  - Κυρίως όπως αποκτά πείρα για το πώς να φτιάξει τα δικά του UI controls.
  - Ένας έμπειρος σχεδιαστής, φτιάχνει πολύ γρήγορα ένα νέο UI control…

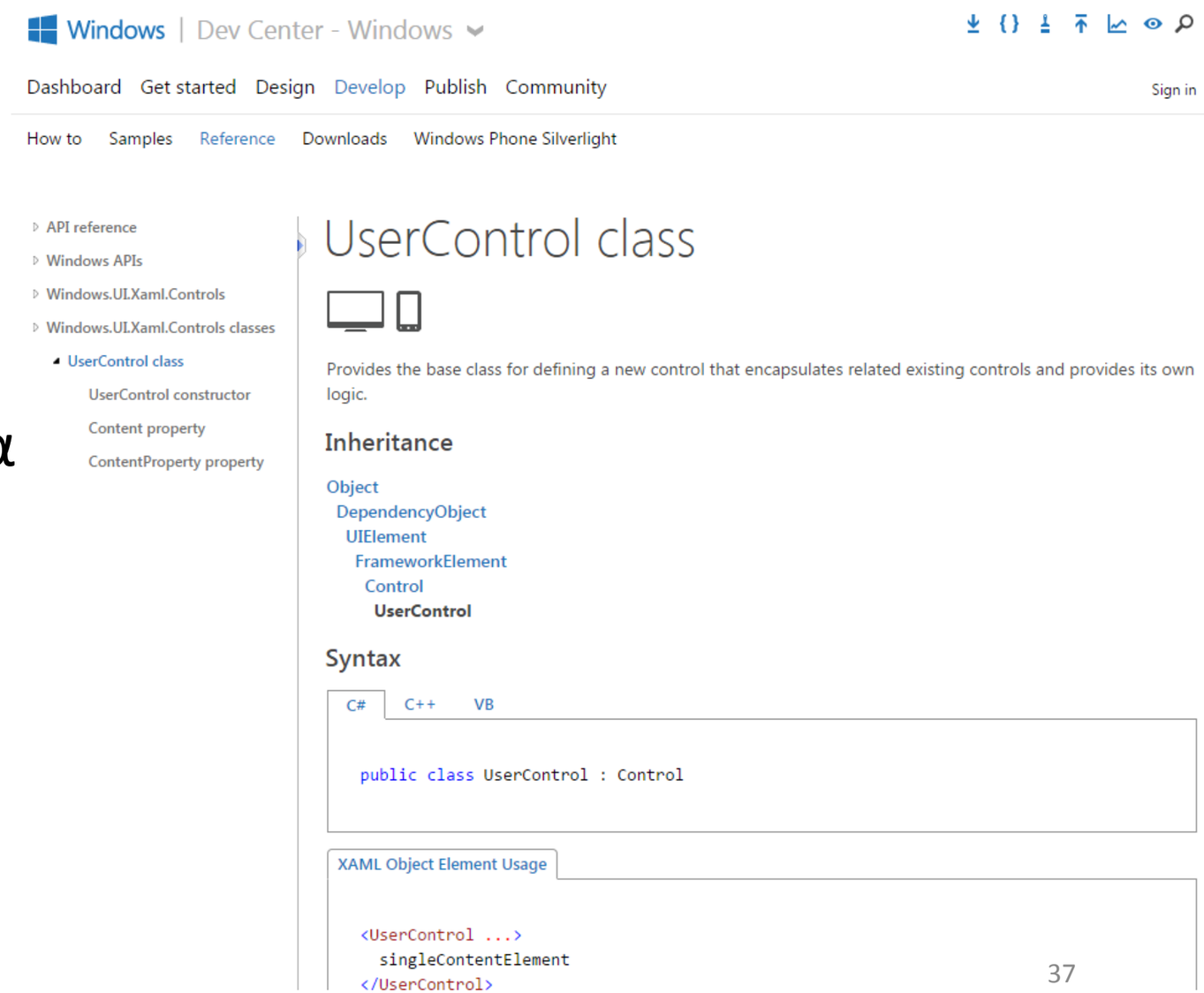
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Κάθε UI Control είναι (κληρονομεί από) ContentControl ή ItemsControl!
- ContentControl Class – **has a Content property with any type of content:** from a string to a panel (grid, stackpanel, etc.) with elements.
  - ButtonBase Class
    - Button, RerpeatButton, ToggleButton, RadioButton, Checkbox
  - UserControl Class
    - A logical group of elements into one reusable control.
    - This is defined by the designer.
- ItemsControl Class – has **a list of elements (items)**.
  - An item is a ContentControl.
  - Π.χ. ListBox, ComboBox, TreeView, Tabcontrol, etc.



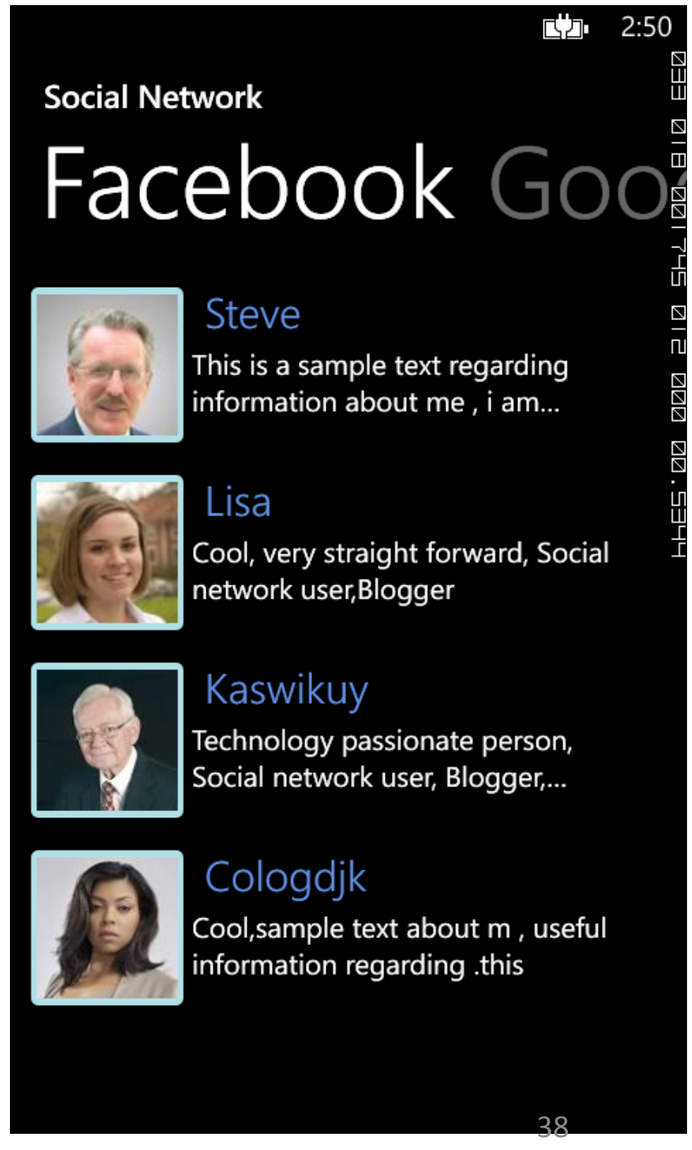
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Τεκμηρίωση – που μπορείτε να διαβάσετε περισσότερα; Στην online τεκμηρίωση της Microsoft για XAML
- Απλά αναζητήστε στο Web για π.χ.: “XAML, UserControl” ή ”UserControl class”.



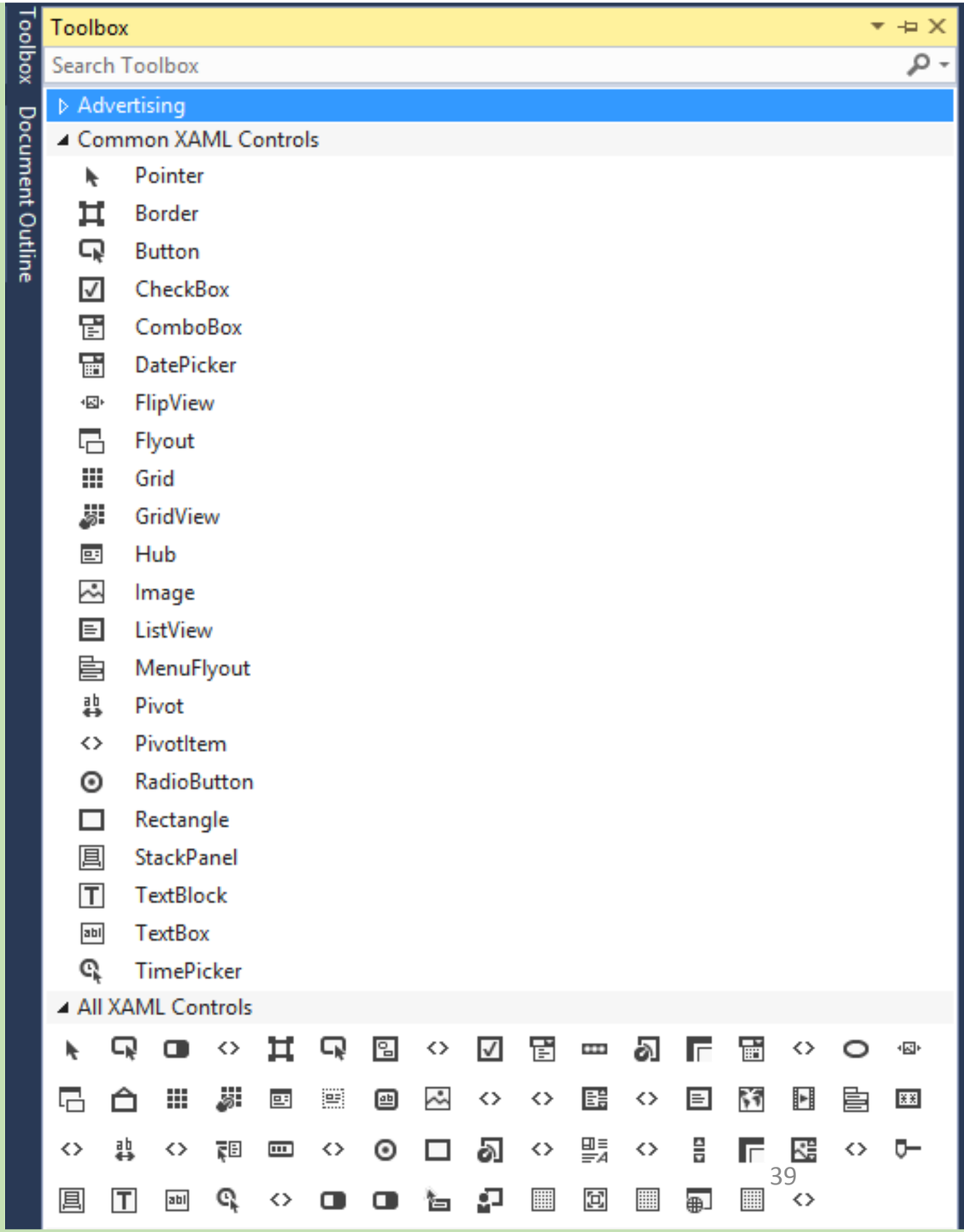
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Φτιάξτε στατική διεπαφή χρήστη με χρήση των Grid, StackPanel, Image, Textblock η οποία να δείχνει σε μια λίστα στοιχεία ανθρώπων, αντίστοιχη της εικόνας.
  - Θα χρειαστεί να βρείτε κάποιες εικόνες από το Web…
  - Για να χρησιμοποιήσετε την εικόνα, πρέπει να την κατεβάσετε τοπικά και να τη μεταφέρετε στα assets (solution explorer)
- Φτιάξτε animation ώστε όταν ο χρήστης επιλέγει την εικόνα, αυτή να περιστρέφεται και να δείχνει τον εικονιζόμενο σε νεότερη ηλικία (παλιά/σήμερα).



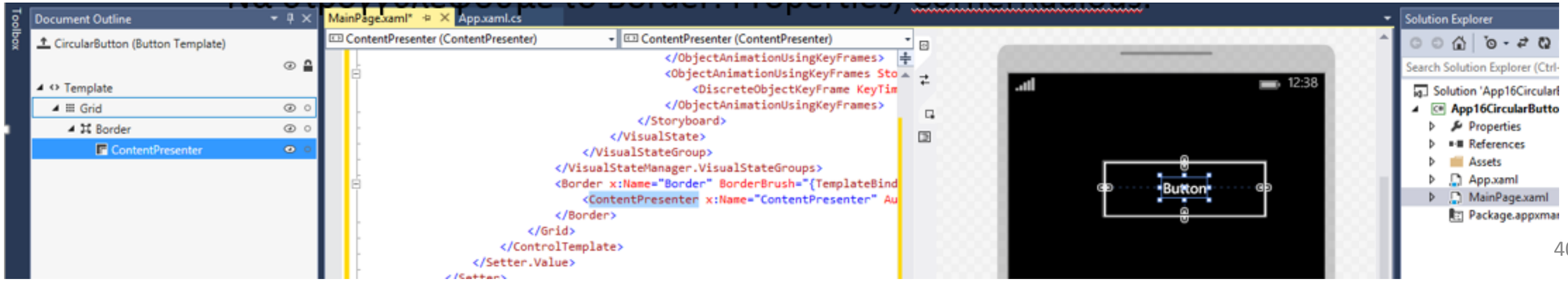
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Custom UI controls …
  - (γιατί;)
- Styles and Templates
  - Μορφοποίηση του στυλ των συστατικών διεπαφής.
- Εργασία με έτοιμα templates
  - Pivot
  - Hub



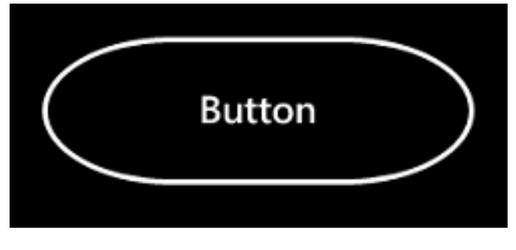
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Custom UI Control – Circular Button!
  - New Project. Add a Button. Right Click | Edit Template | Edit a Copy.
  - Name: CircularButton, for this page (For the App, into App.xaml file)
- Από το Document Outline, βλέπουμε την ‘οπτική δομή’ του Button
  - Grid + Border + ContentPresenter.
- Για να κάνουμε το κουμπί «κυκλικό» μπορούμε:
  - Να στρογγυλέψουμε το Border: Properties, CornerRadious.



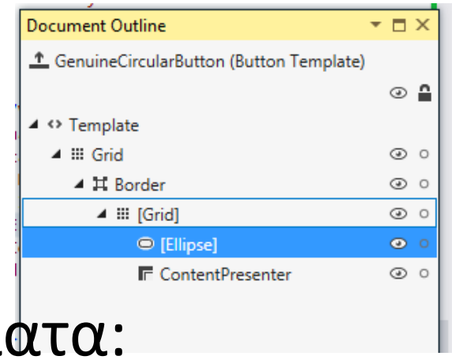
Διάταξη και συστατικά διεπαφής (Layout and UI components)

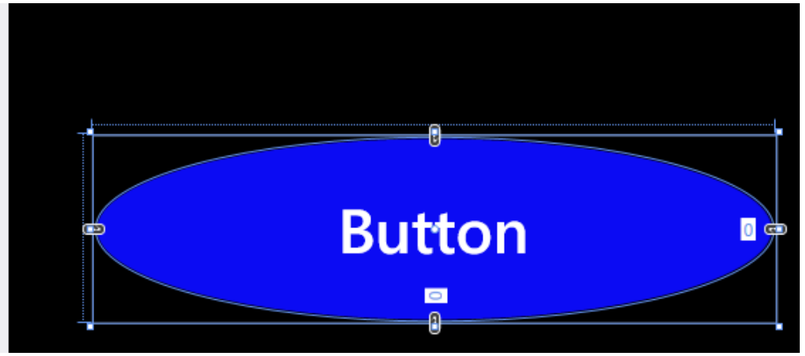
- Με τα παραπάνω βήματα αλλάζουμε το στυλ του Button (αντίγραφο).
- Αν δούμε στην XAML, έχει ενσωματωθεί αρκετός κώδικας (ένα αντίγραφο του Button Template, με όποιες αλλαγές κάνουμε).
- Το κουμπί μας παραμένει ως τέτοιο, άλλα έχει άλλο στυλ από το εξ’ ορισμού (default).




```
<Grid>
    <Button
        Content="Button"
        HorizontalAlignment="Left"
        Height="89"
        Margin="121,76,0,0"
        VerticalAlignment="Top"
        Width="205"
        Style="{StaticResource CircularButton}"/>
</Grid>
```

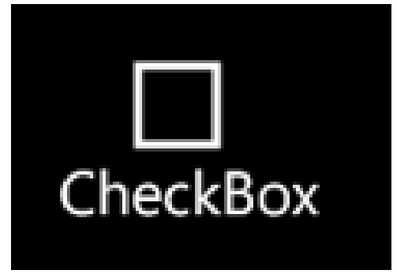
Διάταξη και συστατικά διεπαφής (Layout and UI components)





- Το κουμπί όμως δεν είναι εντελώς ελλειπτικό… Για να το πετύχουμε, κάνουμε τη διαδικασία με άλλα βήματα:
- Νέο button, edit Template (copy)
- Διώχνουμε το Border Background & BorderBrush.
  - Brush | Background → reset, BorderBrush → Reset
- Προσθέτουμε ένα Grid πριν τον ContentPresenter
  - Επειδή εντός του Border μπορεί να υπάρχει μόνο ένα στοιχείο…
  - Document Outline | Right Click on Border | Group Into | Grid.
- Εντός του Grid, προσθέτουμε μια έλλειψη.
  - Καλύτερα από το Document Outline
- Ορίζουμε την έλλειψη ώστε να πιάνει όλο το χώρο του border.
  - Properties: Reset Margin, Reset Width, Height.

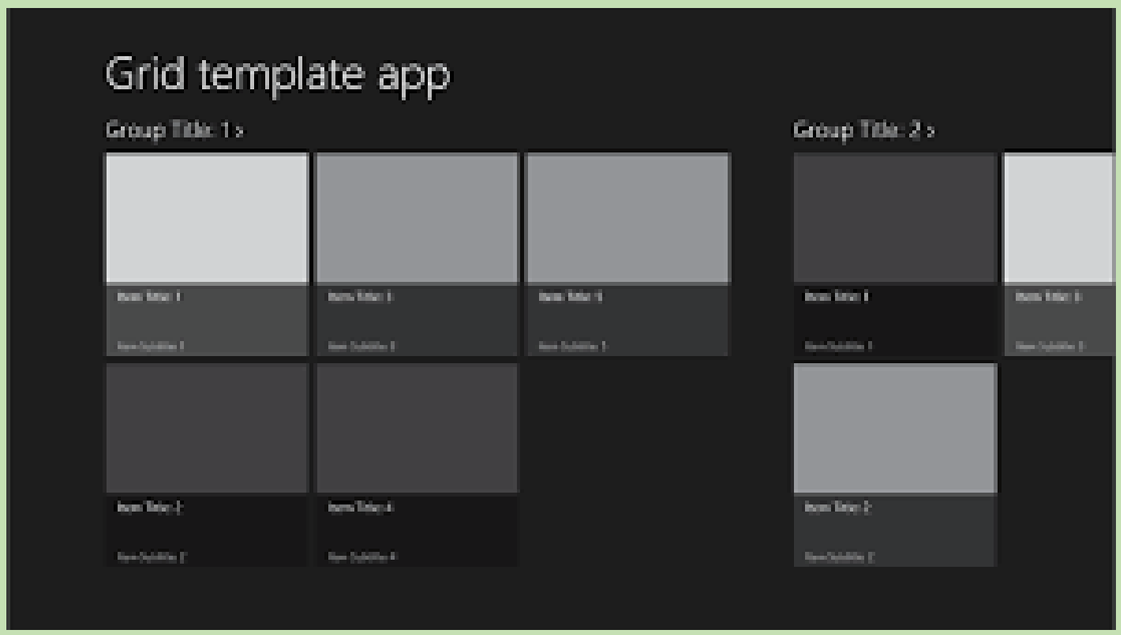
Διάταξη και συστατικά διεπαφής (Layout and components)



- Άσκηση: Αλλάξτε το στυλ ενός CheckBox ώστε να βρίσκεται το κουτί επιλογής πάνω από το κείμενο.
  - Βοήθεια: μπορείτε εργαστείτε αποκλειστικά στην XAML…
- Άσκηση: Φτιάξτε ένα ImageButton
  - Βοήθεια: κατεβάστε ένα icon από το web, αποθηκεύστε το τοπικά και εισάγετε το στο project (σύρτε το στα Assets).
  - Προσθέστε StackPanel εντός του Border.
- Άσκηση: Αλλάξτε το στυλ ενός Slider, ώστε το handle να είναι κυκλικό.
- Η εκτεταμένη διαμόρφωση των Templates, απαιτεί καλή γνώση της XAML + C#…
  - Επίσης, είναι δυνατόν να φτιαχτούν Custom Controls από μηδενική βάση (Add | Item | Custom Control).

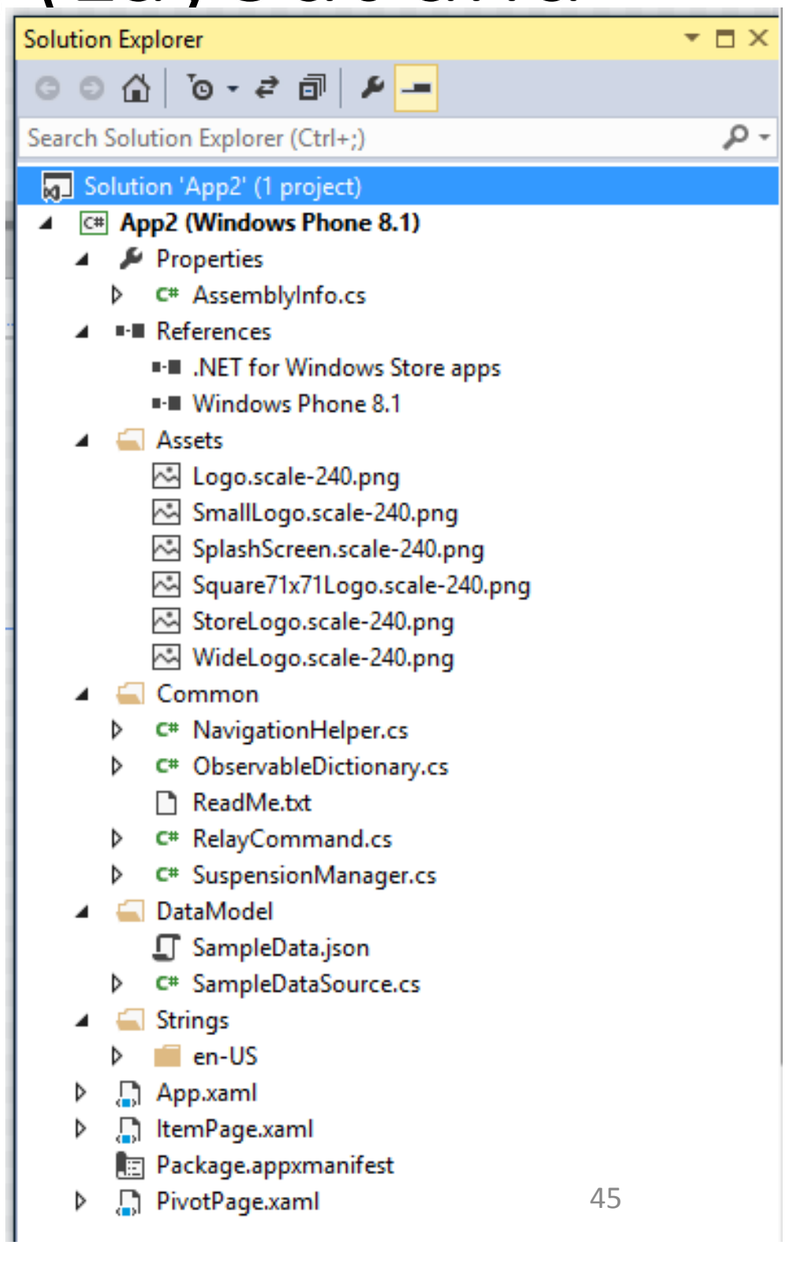
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Εργασία με project templates
  - Pivot, Hub (Phone), Grid (WPF)



Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Εργασία με ένα Pivot Template…
  - Τα στοιχεία που απαρτίζουν ένα Pivot Project.
    - Solution Explorer
    - XAML
    - Binding to styles
  - Τα δεδομένα ενός Pivot Project.
    - JSON: Javascript Object Notation
- Επιθεωρήστε τα XAML αρχεία ItemPage.xaml, PivotPage.xaml.
  - Από πού παίρνουν τα δεδομένα τους;
- Επιθεωρήστε τις κλάσεις εντός του SampleDataSources.cs.
  - Βρείτε την αναφορά σε αρχείο .json.



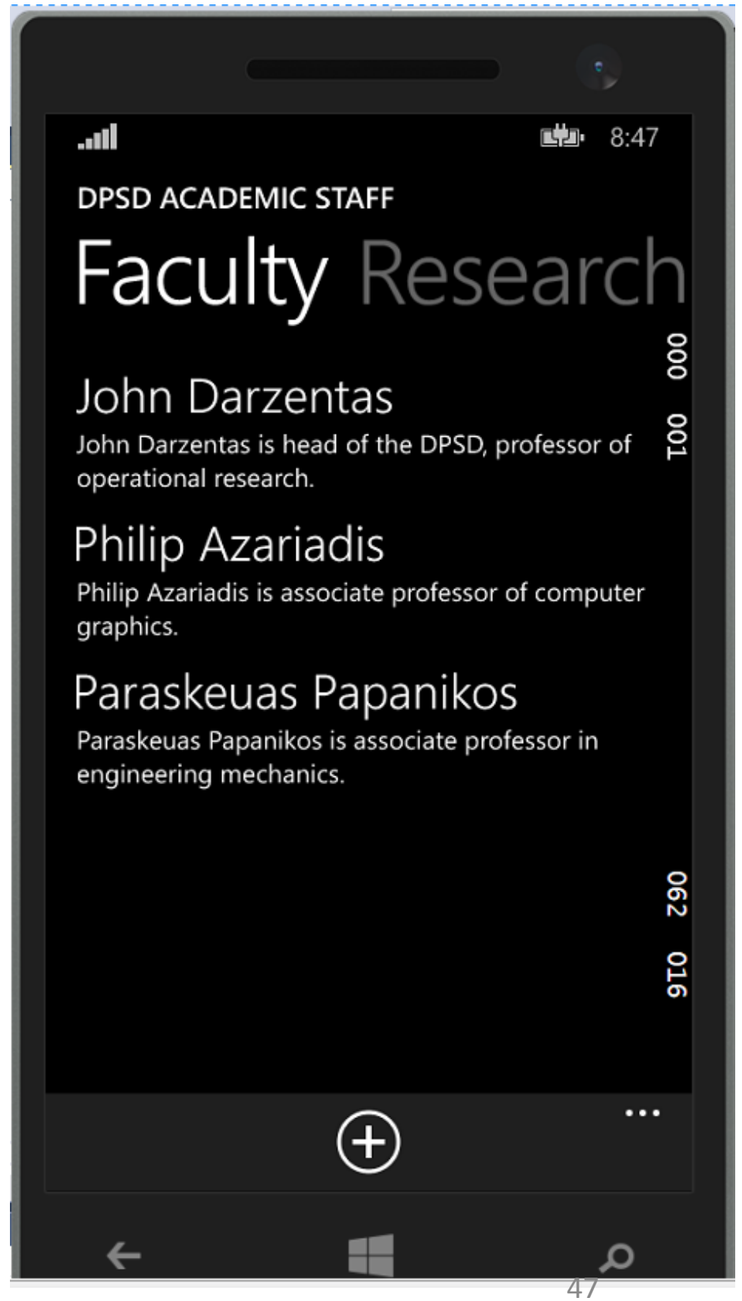
Διάταξη και συστατικά διεπαφής (Layout and UI components)

- JSON: Java Object Notation
  - Human-readable format για αποθήκευση και ανταλλαγή δεδομένων μεταξύ εφαρμογών.
- Βασικοί τύποι δεδομένων
  - Number,
  - String,
  - Boolean,
  - Array,
  - Object,
  - null.

```
{"Groups":[
  {
    "UniqueId": "Group-1",
    "Title": "Faculty",
    "Subtitle": "Permanent",
    "ImagePath": "Assets/Logo.scale-240.png",
    "Description" : "Permanent faculty are professors and assistant professors.",
    "Items":
    [
      {
        "UniqueId": "Group-1-Item-1",
        "Title": "John Darzentas",
        "Subtitle": "Professor, Head of Department",
        "ImagePath": "Assets/Logo.scale-240.png",
        "Description" : "John Darzentas is head of the DPSD, professor of operational research.",
        "Content" : "Short cv: His research interests include systems thinking..."
      },
      {
        "UniqueId": "Group-1-Item-2",
        "Title": "Philip Azariadis",
        "Subtitle": "Associate Professor",
        "ImagePath": "Assets/blue-home-icon.png",
        "Description" : "Philip Azariadis is associate professor of computer graphics.",
        "Content" : "Short cv: His research interests include computer aided design, ..."
      },
      {
        "UniqueId": "Group-1-Item-3",
        "Title": "Paraskeuas Papanikos",
        "Subtitle": "Associate Professor",
        "ImagePath": "Assets/Logo.scale-240.png",
```

Διάταξη και συστατικά διεπαφής (Layout and UI components)

- Εργασία με ένα Pivot Template…
  - Προσαρμόζοντας το pivot project για μια εφαρμογή.
  - Διεπαφή:
    - Αλλαγές σε χρώματα και τυπογραφία.
    - Προσθήκες στη διάταξη (UI components).
    - Επιπλέον σχεδίαση εντός των επιλεγμένων στοιχείων (Item).
- Δεδομένα:
  - Ενημέρωση του json file «χειρωνακτικά» (για ενδεικτικά δεδομένα).
  - Μετατροπή δεδομένων από άλλα formats (από βάση δεδομένων μέχρι Excel).
- Αντίστοιχα και για τα άλλα templates…
- Δημιουργήστε νέο Project (Pivot Template, ή άλλο) και προσαρμόστε το για εφαρμογή DPSD ACADEMIC STAFF.
  - Headers: Faculty, Researchers.
  - Items: ‘Persons’
  - Item content: a photo and a short cv.



Διάταξη και συστατικά διεπαφής (Layout and UI components)

Εργασία: 2-3 άτομα, 20% του τελικού βαθμού

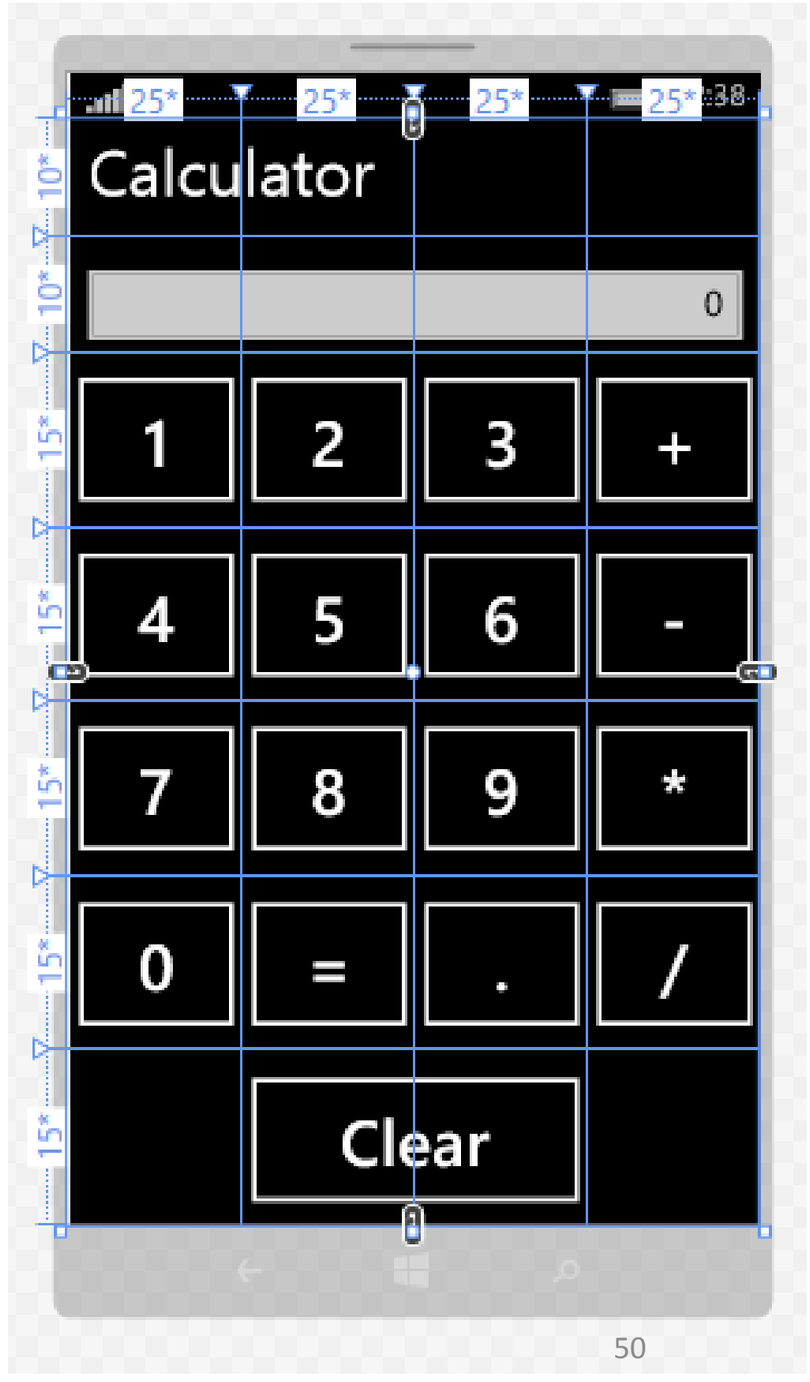
- Project: Αναπτύξτε ένα ολοκληρωμένο **Pivot**/Hub/Grid Project, για:
  - 1. COURSES (Headers: 1st semester, 2nd semester; Items: each course; Item content: related photo + description)
  - 2. MY PORTFOLIO (Headers: Course projects, Funded projects; Items: each project ; Item content: related photo + description)
  - 3. DPSD BUILDINGS & ROOMS (Headers: A' Gymnasio, Studios; Items: rooms ; Item content: related photo + description)
  - 4. A' GYMNASIO HISTORY (Headers: Periods, People; Items: certain periods, certain people; Item content: related photo + description)
  - 5. Άλλο θέμα της επιλογής σας… Ή προσαρμόστε τα παραπάνω θέματα.
- Να εργαστείτε στα εξής:
  - Διεπαφή:
    - Αλλαγές σε χρώματα και τυπογραφία.
    - Προσθήκες στη διάταξη (UI components).
    - Επιπλέον σχεδίαση εντός των επιλεγμένων στοιχείων (Item).
  - Δεδομένα:
    - Ενημέρωση του json file «χειρωνακτικά» (για ενδεικτικά δεδομένα).
    - Εύρεση φωτογραφιών από το Web, με πιθανή επεξεργασία τους.

Σύνδεση δεδομένων, συμβάντα και χειριστές (data binding, events, event handlers)

- Άσκηση: Φτιάξτε ένα υπολογιστή όπως της εικόνας.

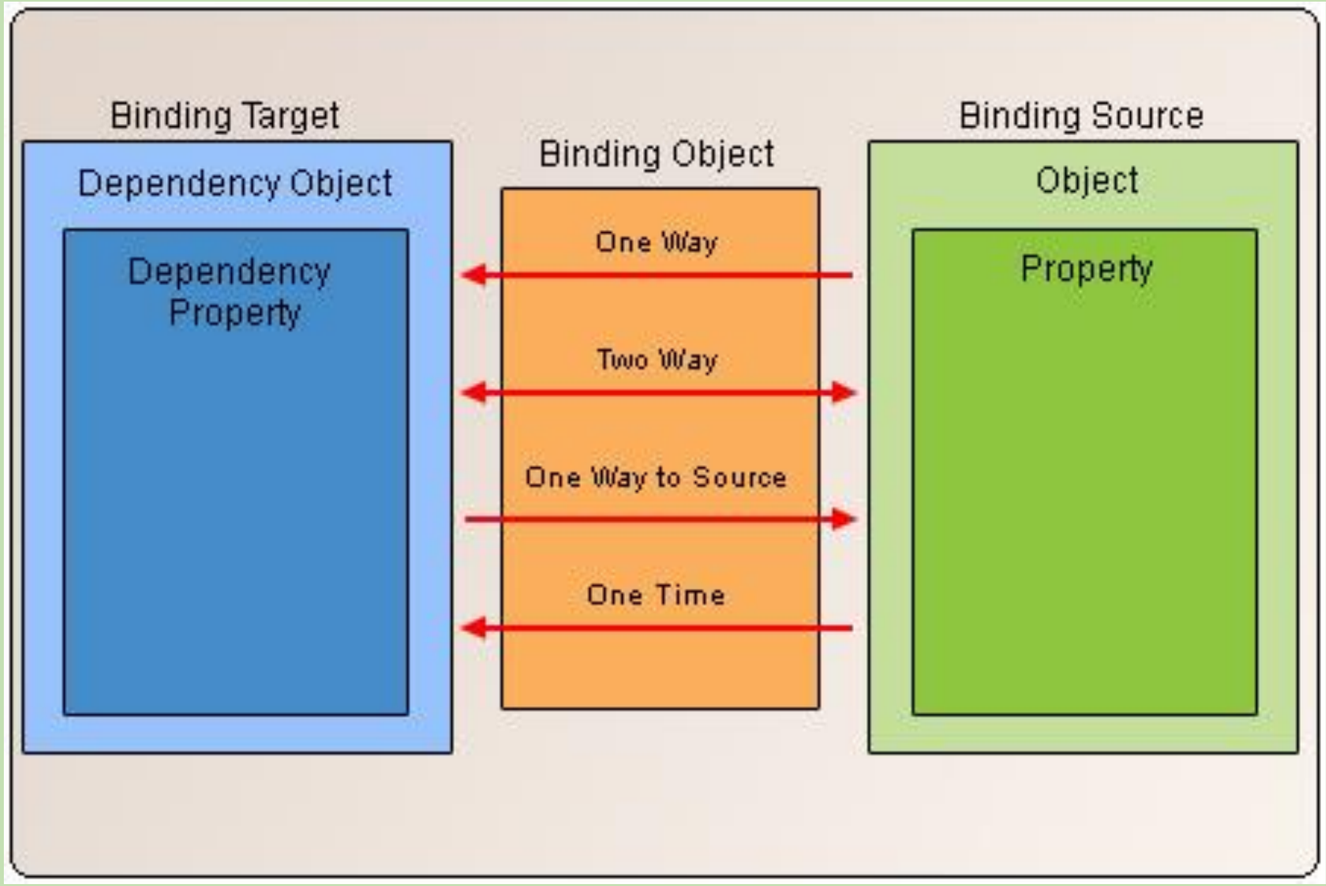
Σύνδεση δεδομένων, συμβάντα και χειριστές (data binding, events, event handlers)

- Άσκηση: Αναπτύξτε ένα υπολογιστή όπως της εικόνας.
- Βοήθεια για τη διεπαφή
  - Τοποθετείστε τα στοιχεία εντός Grid, reset size, margins, stretch.
  - Δουλέψτε στην XAML με Copy & Paste, ανά γραμμή.
- Βοήθεια C#: θα χρειαστείτε 5 event handlers:
  - Αριθμητικά κουμπιά → εμφάνιση στην οθόνη (TextField) του νέου αριθμού
  - Κουμπιά πράξεων → υπολογισμός, και εμφάνιση στην οθόνη (TextField)
  - Κουμπί Clear → Καθαρισμός οθόνης (TextField).
  - Κουμπί ‘.’
  - Κουμπί ‘=‘



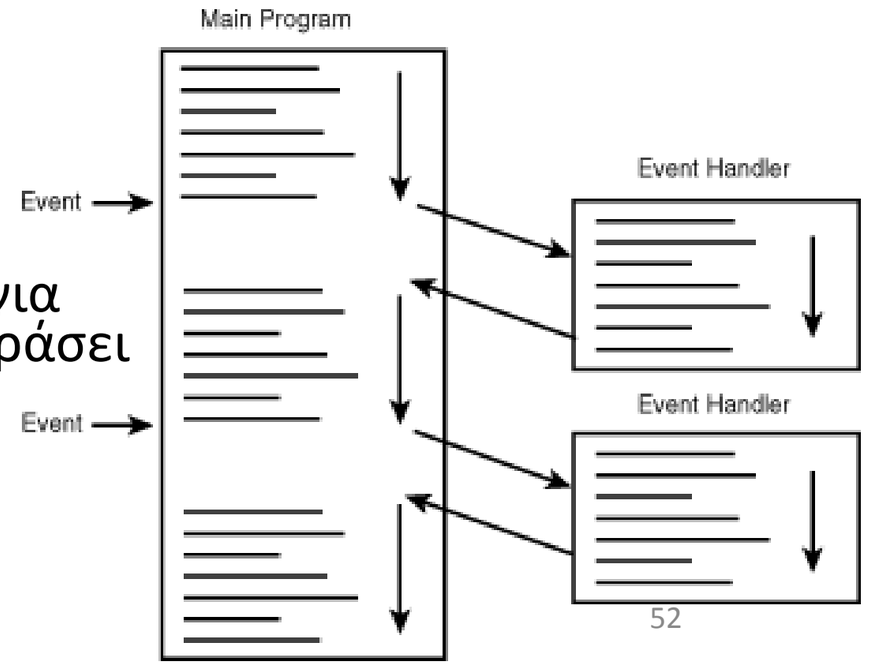
Συμβάντα (Events) και Σύνδεση Δεδομένων (Data Binding)

- Συμβάντα – τι είναι;
- Σύνδεση δεδομένων - Τι είναι;
- Data binding on the UI
- Χειριστές συμβάντων – τι είναι;
- Χειρισμός απλών συμβάντων.



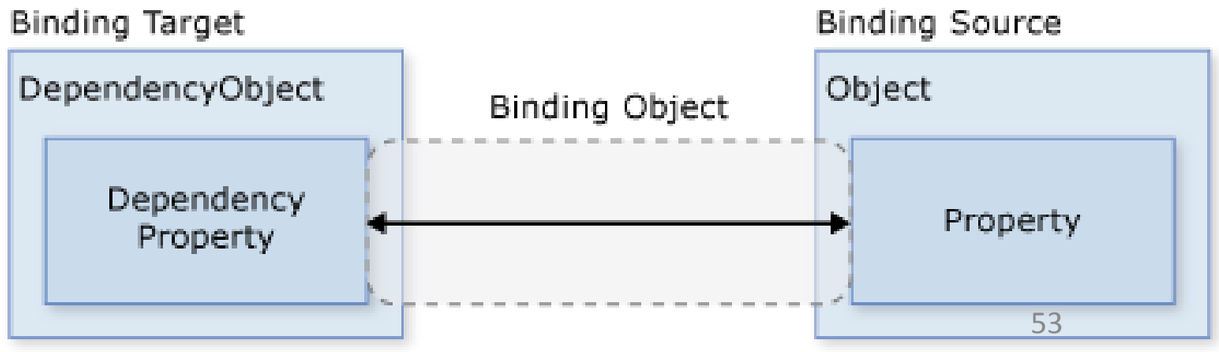
Συμβάντα και χειριστές (events, event handlers)

- Παρότι ένα πρόγραμμα συνήθως εννοείται ως ένα σύνολο εντολών και της ροής εκτέλεσης τους (που ελέγχεται με συνθήκες, επαναλήψεις, κ.α.) …
- … μεγάλο μέρος κάθε προγράμματος είναι **καθοδηγούμενο από συμβάντα (event-driven)**:
  - Η ροή εκτέλεσης καθορίζεται από **εξωτερικά του προγράμματος** συμβάντα, στα οποία το πρόγραμμα καλείται να ανταποκριθεί.
- Ένα συμβάν είναι ένα **σήμα** για κάτι σημαντικό έχει συμβεί. Π.χ.
  - Όταν ο χρήστης πατάει το Button, αυτό δίνει σήμα ότι πατήθηκε (click event).
  - Αν ο προγραμματιστής έχει υλοποιήσει κάποια μέθοδο για να διαχειριστεί το συμβάν, τότε το πρόγραμμα θα αντιδράσει
  - Η μέθοδος διαχείρισης ενός συμβάντος ονομάζεται **χειριστής συμβάντος (event handler)**.



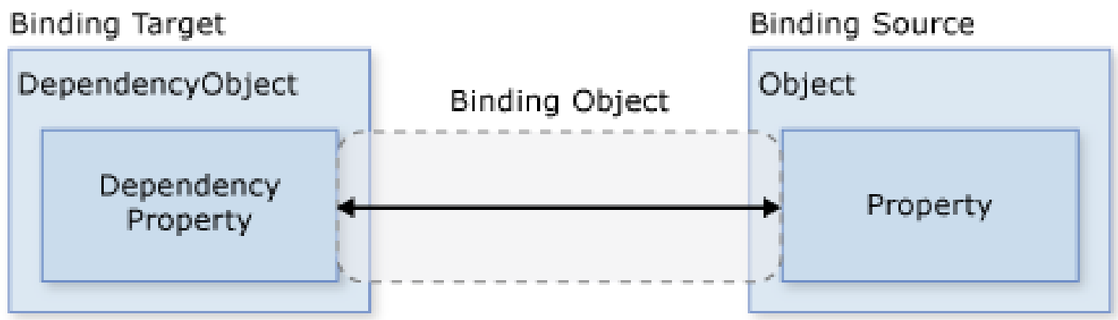
Σύνδεση Δεδομένων (Data Binding) και Συμβάντα (Events)

- Όταν δημιουργούνται συμβάντα (events) από/σε:
  - δράσεις του χρήστη (ενέργειες στη διεπαφή)
  - στιγμές της εκτέλεσης του προγράμματος όπου απαιτείται να ενημερωθεί η διεπαφή με δεδομένα.
- Απαιτείται η **αυτοματοποιημένη, δυναμική σύνδεση των δεδομένων με τη διεπαφή χρήστη (και αντίστροφα)**.
  - Σύνδεση δεδομένων (data binding)
- Η σύνδεση δεδομένων πρέπει να γίνεται άμεσα, γι αυτό είναι καθοδηγούμενη από συμβάντα.
- Διαχωρισμός της διεπαφής από το ‘κύριο πρόγραμμα’.



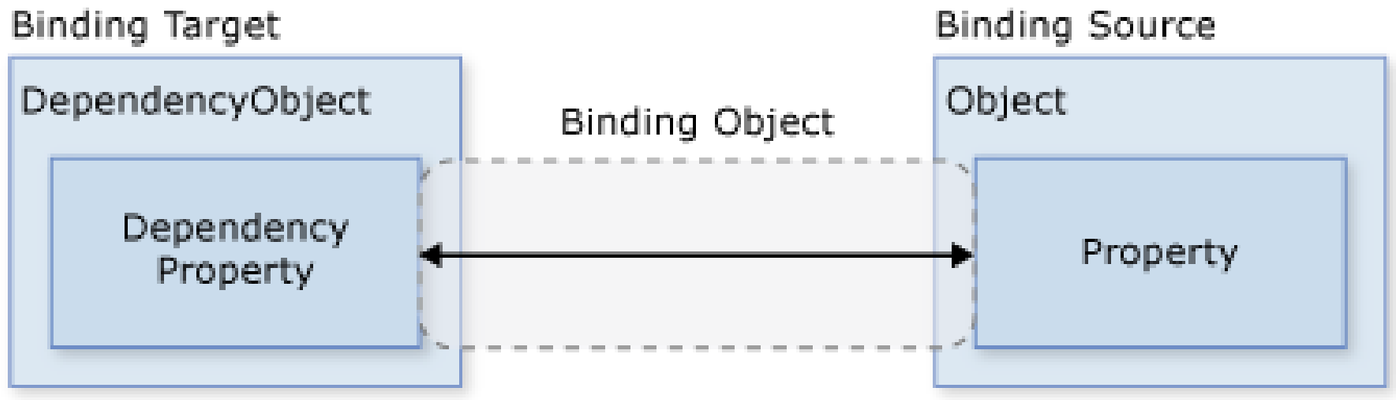
Σύνδεση Δεδομένων (Data Binding) και Συμβάντα (Events)

- Binding **source object**: An object from the ‘main program’
- Binding **source property (path)**: A property from the ‘main program’
- Binding **target object**: (Dependency Object =) XAML object
- Binding **target property**: (Dependency Object =) XAML property
- **Binding object**: an object in between, that populates the event (and may also hold the data values)
- `<TextBox IsEnabled="{Binding ElementName=MyCheckBox, Path=IsChecked}"/>`



Σύνδεση Δεδομένων (Data Binding) και Συμβάντα (Events)

- `<TextBox IsEnabled="{Binding ElementName=MyCheckBox, Path=IsChecked}"/>`
- Binding **source object**: MyCheckBox
- Binding **source property (path)**: IsChecked
- Binding **target object**: (Dependency Object =) TextBox
- Binding **target property**: (Dependency Object =) IsEnabled
- **Binding object**: this is called with the Binding keyword



Σύνδεση Δεδομένων (Data Binding) και Συμβάντα (Events)

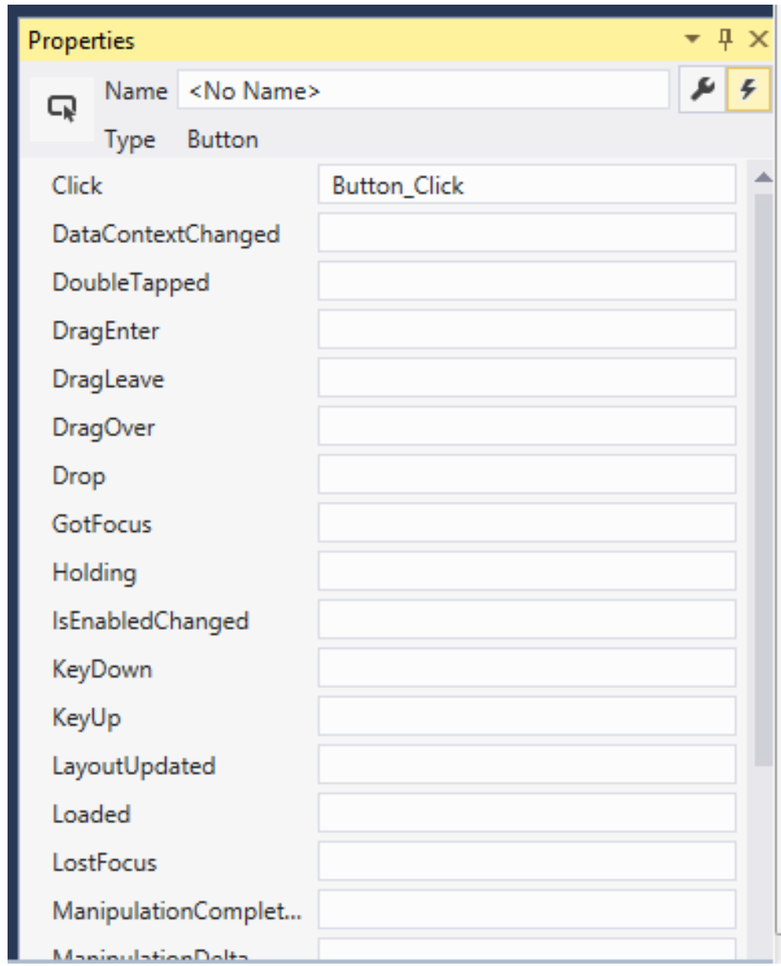
- Data Binding on the UI
  - Γιατί; - Για όποιο υπολογισμό μπορούμε να κάνουμε στο επίπεδο της διεπαφής.
    - Αυτοματοποιημένοι έλεγχοι τιμών, αυτοματοποιημένη συμπλήρωση φόρμας, κ.α.
    - Πρώτα ενημερώνουμε τη διεπαφή, έπειτα το κύριο πρόγραμμα.
    - Responsive apps!
  - Πρέπει να προσδιορίσετε το ElementName (Binding Source) και το Path (Binding Path, i.e. Property).
  - Μπορείτε και με τη χρήση της κλάσης DataContext.

```
<ListBox x:Name="myListBox" /> ...
<TextBlock Text="{Binding Path=SelectedItem,
                          ElementName=myListBox}" />
```

```
DataContext="{Binding ElementName=myListBox,
                      Path=SelectedItem}" ...>
```

Συμβάντα και χειριστές(events and handlers)

- Προκαθορισμένα συμβάντα ανά συστατικό διεπαφής χρήστη.
  - Properties, events icon
  - Για κάθε συμβάν που θέλουμε να χειριστούμε, απαιτείται να γράψουμε κώδικα C# στο κύριο πρόγραμμα μας (.cs)
    - Κάνουμε υπολογισμούς…
    - Αναφερόμαστε στα στοιχεία της διεπαφής με τα ονόματα τους (Properties: Name, XAML: x:Name).
    - Επεξεργαζόμαστε δεδομένα από άλλες πηγές (εκτός της διεπαφής)
- Περιήγηση των συμβάντων για μερικά συστατικά όπως Button, Textblock, ComboBox, …
- Οι χειριστές συμβάντων (event handlers) είναι οι μέθοδοι του προγράμματος μας που κάνουν τις παραπάνω επεξεργασίες.



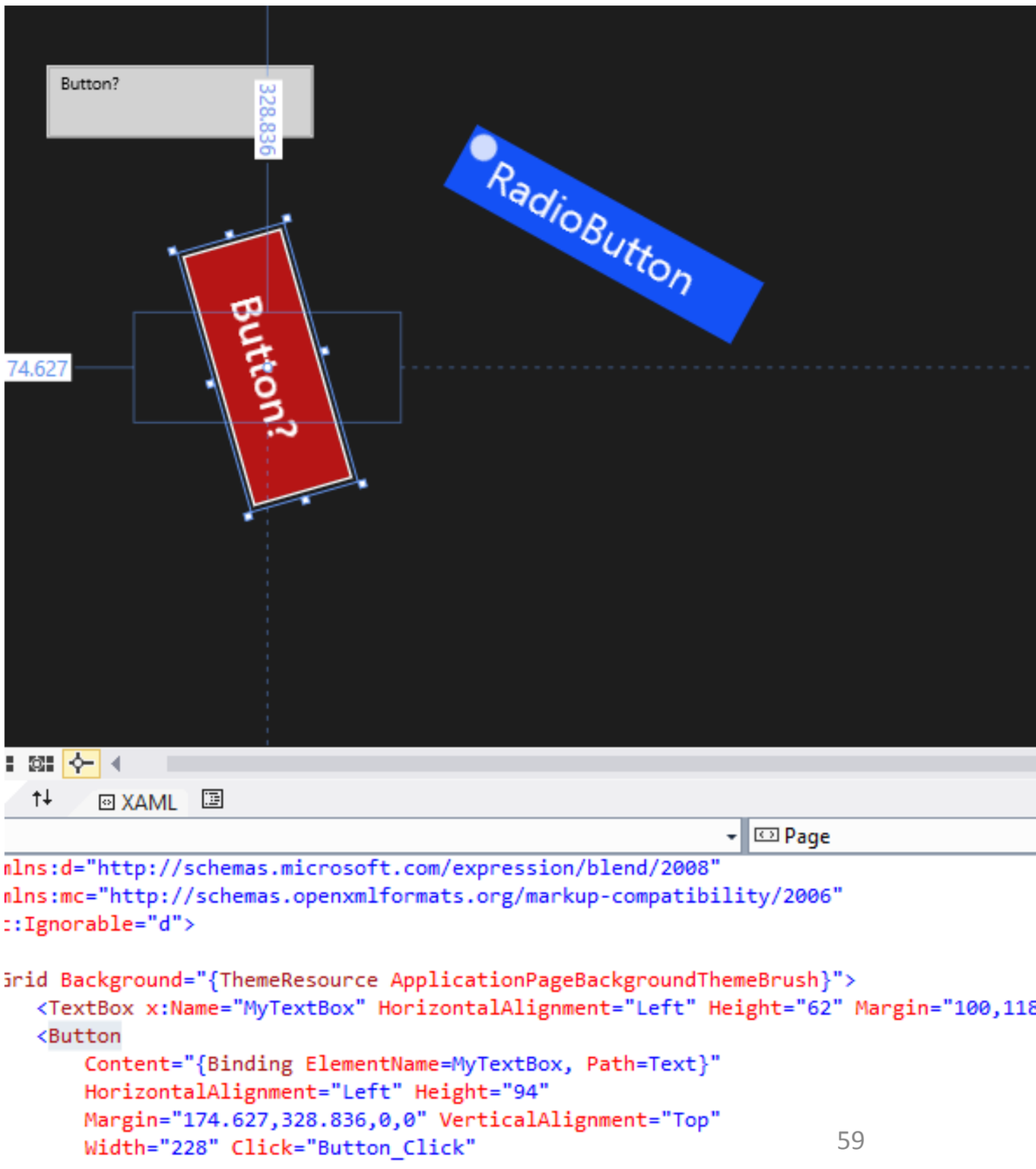
Συμβάντα και χειριστές (events and handlers)

- Ο σκελετός κώδικα για κάθε χειριστή συμβάντων ενός συστατικού της διεπαφής δημιουργείται αυτόματα, με διπλό κλικ εντός του κάθε συμβάντος από το Properties, events.
- Ο κώδικας C# πίσω από τη διεπαφή (code-behind) είναι μια κλάση με το ίδιο όνομα με αυτό του XAML.
- Μπορεί να είναι Page (Phone App) ή Window (WPF App).
- Η μέθοδος InitializeComponent αρχικοποιεί και φορτώνει τη διεπαφή.
- Οι χειριστές συμβάντων καλούνται αυτόματα, όταν ο χρήστης επιλέξει το αντίστοιχο UI component.

```
namespace App9
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            MyTextBox.Text = "Start again!";
        }

        private void RadioButton_Checked(object sender, RoutedEventArgs e)
        {
            MyTextBox.Text = "Radio";
        }
    }
}
```

Σύνδεση δεδομένων (Data binding)

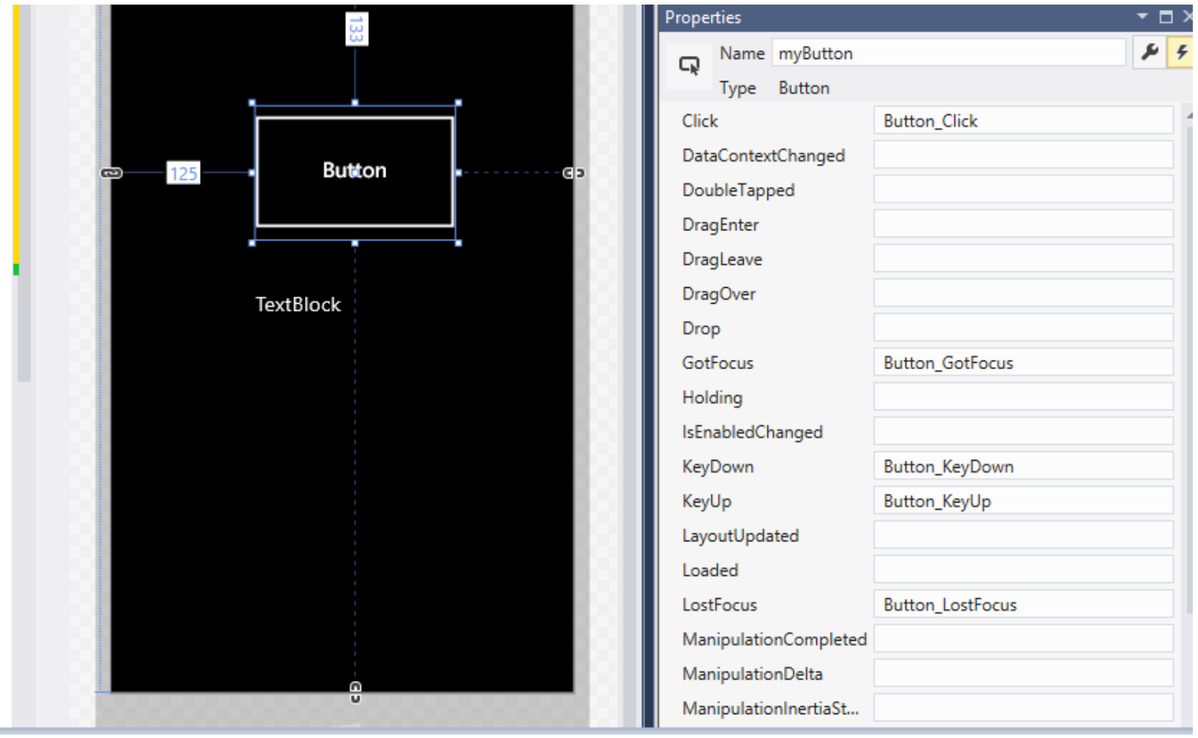
- Άσκηση: Ας φτιάξουμε μερικά 3-4 UI components και ας δεσμεύσουμε τα δεδομένα μεταξύ τους.
  - Έστω ένα Button που θα παίρνει το Content του από ένα TextField.
  - Έστω ότι το TextField παίρνει συγκεκριμένη τιμή όταν ένα RadioButton είναι επιλεγμένο.



Σύνδεση δεδομένων, συμβάντα και χειριστές

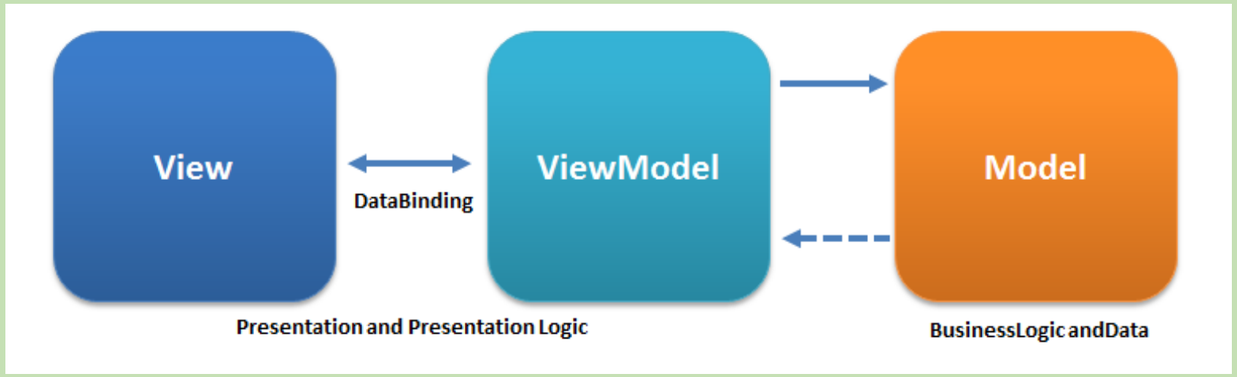
- Άσκηση: Φτιάξτε ένα Button που να διαχειρίζεται κάποια συμβάντα (με μεθόδους-χειριστές συμβάντων) τυπώνοντας τα σε ένα TextBox.

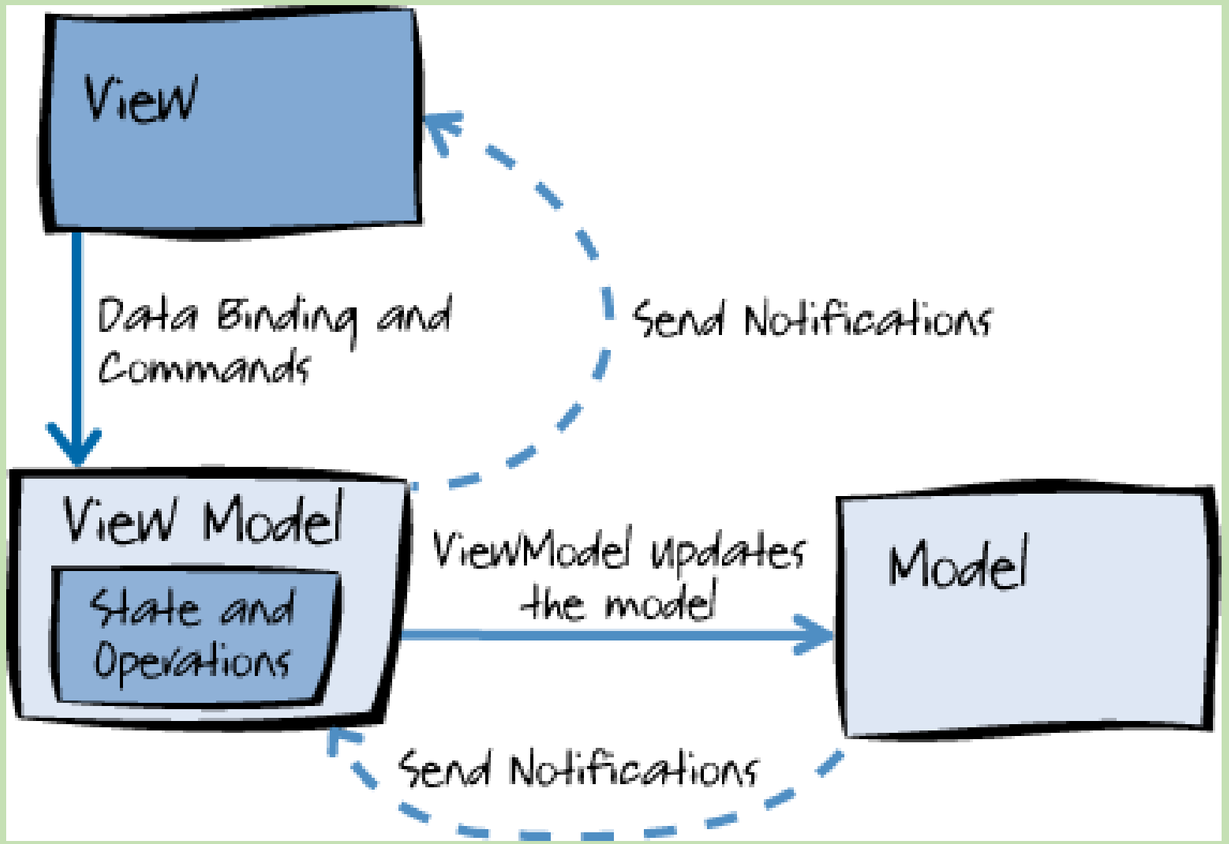
```
    <Grid>
        <Button
            x:Name="myButton"
            Content="Button"
            HorizontalAlignment="Left"
            Margin="125,133,0,0"
            VerticalAlignment="Top"
            Height="115" Width="172"
            Click="Button_Click"
            KeyDown="Button_KeyDown"
            KeyUp="Button_KeyUp"
            GotFocus="Button_GotFocus"
            LostFocus="Button_LostFocus"
            PointerPressed="Button_PointerPressed"
            PointerReleased="Button_PointerReleased"
            RightTapped="Button_RightTapped"
            Tapped="Button_Tapped"/>
        <TextBlock
            x:Name="myTextBlock"
            HorizontalAlignment="Left"
            Height="46"
            Margin="125,295,0,0"
            TextWrapping="Wrap"
            Text="TextBlock"
            VerticalAlignment="Top"
            Width="166"
            FontSize="18"/>
    </Grid>
/Page>
```



Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

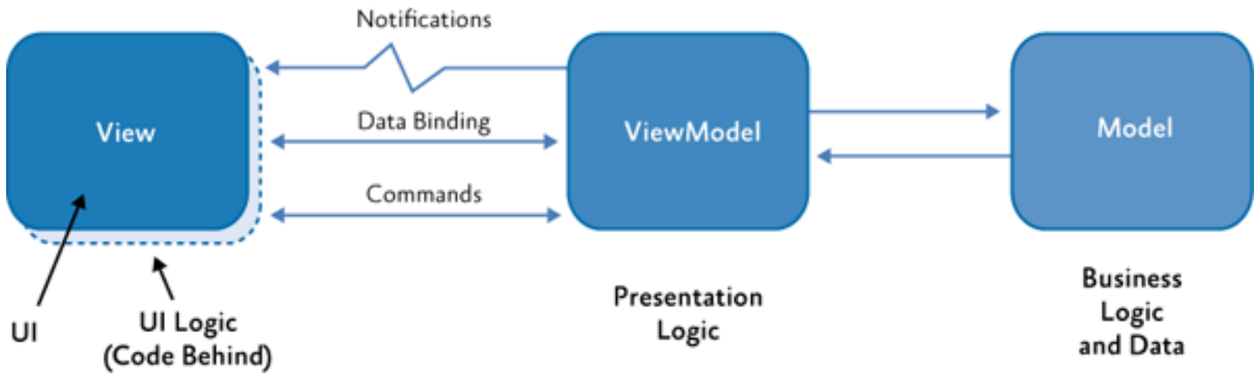
- MVVM (Model, View, ViewModel)
- INotifyPropertyChanged





Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

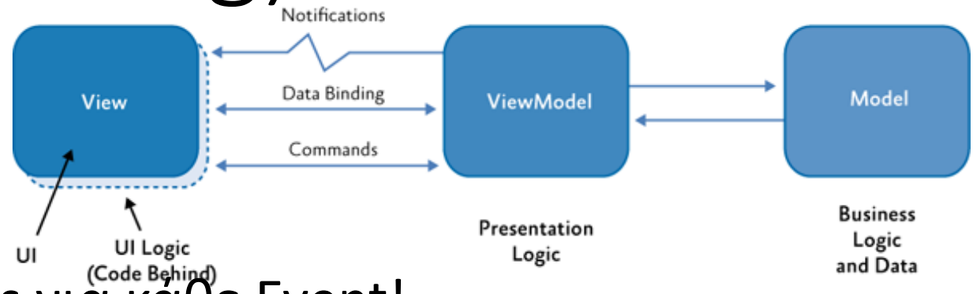
- MVVM (Model-View, View-Model) - Software architecture pattern
  - Model. Η **λογική** της εφαρμογής.
    - Μια ή περισσότερες κλάσεις, αλγόριθμοι, υπολογισμοί.
    - Μπορούν να είναι σε ένα ή περισσότερα .cs (C#) αρχεία, εντός ενός project.
  - View. Η **διεπαφή** (όψη) της εφαρμογής.
    - Μια ή περισσότερες σελίδες (Page, phone app) ή παράθυρα (Window, WPF app)
    - Κάθε σελίδα ή παράθυρο:
      - Ξεχωριστό XAML αρχείο.
      - Έχει code-behind file (.cs for C#)
  - ViewModel. Η **σύνδεση** μοντέλου-όψης.
    - Μια κλάση (.cs, for C#).
    - Περιλαμβάνει όλες τις ιδιότητες (Properties) που συνδέονται με Data Binding.
      - Δηλαδή, όλες τις ιδιότητες του View που πρέπει να συνδεθούν με ιδιότητες του Model.
    - Πρέπει να υλοποιεί την διεπαφή InotifyPropertyChanged.



Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- INotifyPropertyChanged - Διεπαφή λογισμικού (Software Interface)
  - **Υλοποιείται από το ViewModel (ξεχωριστή κλάση)** σε projects μεσαίου ή μεγάλου μεγέθους, ή το Model για μικρότερα projects.
    - Η κλάση ViewModel περιλαμβάνει όλα τα δεδομένα που ζουν στη διεπαφή και το ‘κύριο’ πρόγραμμα.
- Όταν η τιμή μιας ιδιότητας της εφαρμογής αλλάξει από το χρήστη (στην Όψη), τότε **ενημερώνεται αυτόματα το ‘κύριο’ πρόγραμμα (Model)**.
  - Π.χ. ο χρήστης συμπληρώνει σε TextField.
- Όταν η τιμή μιας ιδιότητας της εφαρμογής αλλάξει στο ‘κύριο’ πρόγραμμα (Model), τότε **ενημερώνεται αυτόματα η Όψη (ώστε να ενημερωθεί ο χρήστης)**.
  - Π.χ. αυτόματη συμπλήρωση TextField από βάση δεδομένων.
- Η εναλλακτική της χρήσης της INotifyPropertychanged είναι να υλοποιήσουμε event handlers…
  - Το κάναμε ήδη στα προηγούμενα με το calculator.

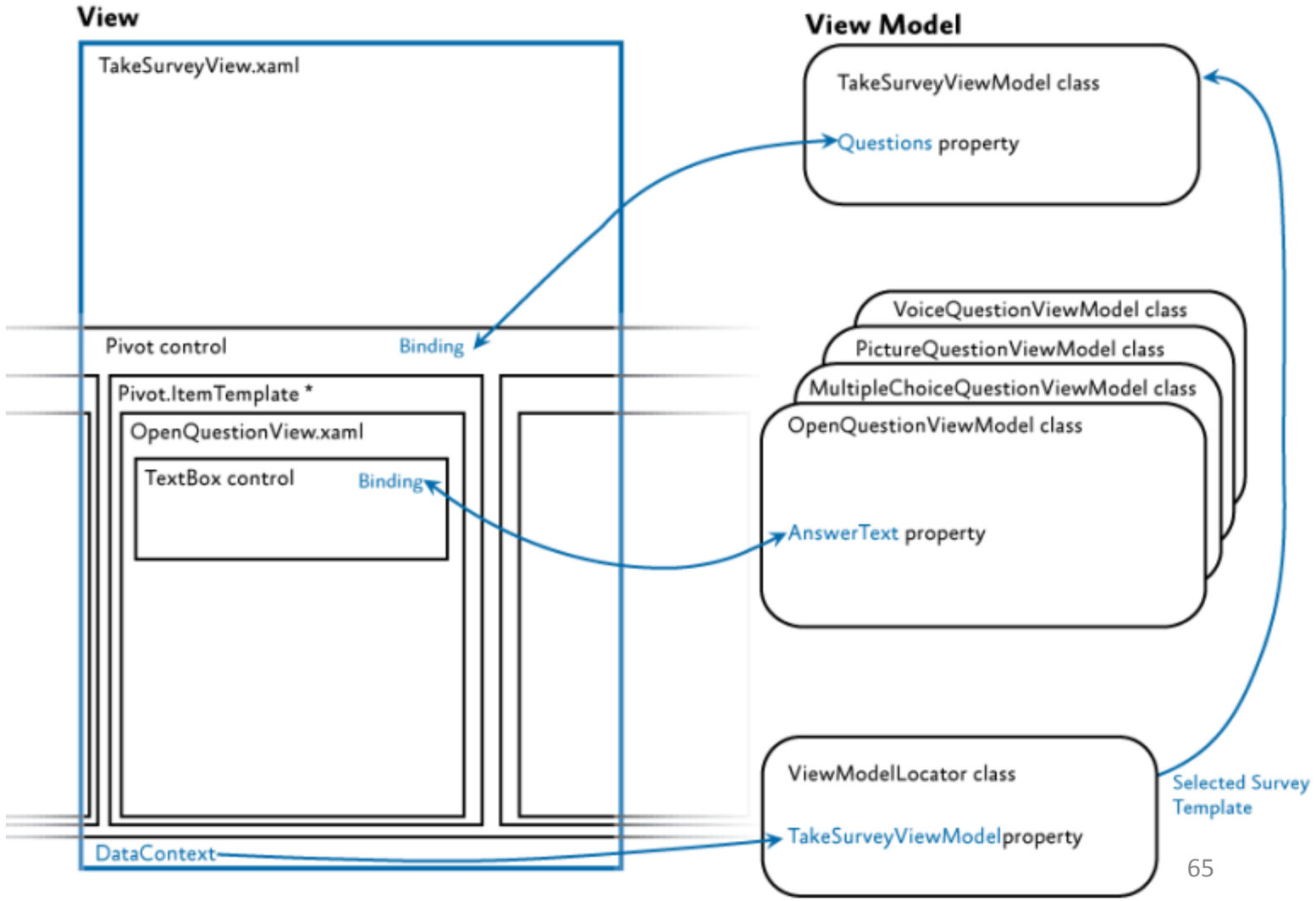
Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος



- MVVM (Model-View, View-Model)
  - Γιατί; Αφού μπορούμε να γράψουμε Event Handlers για κάθε Event!
  - Επειδή:
    - Σε μια μέτριου μεγέθους εφαρμογή, απαιτείται να γράψουμε 10δες ίσως και 100δες event handlers!!
    - Όταν έχουμε πολλούς event handlers…
      - Ο κώδικας δεν είναι εύκολα ελέγξιμος – η ροή ελέγχου διαμορφώνεται σε μεγάλο βαθμό από την αλληλεπίδραση του χρήστη …
      - Μικρές αλλαγές στη διεπαφή, μπορεί να δημιουργήσουν πολύ κόπο στο προγραμματιστή για να προσαρμόσει το κώδικα της ‘λογικής’ του προγράμματος …
- Θυμηθείτε το calculator…
  - Πολύ απλή εφαρμογή.
  - Γράψαμε 5 event handlers!
  - Δεν γράψαμε event handlers για είσοδο από το πληκτρολόγιο, ή για άλλες δράσεις του χρήστη (π.χ. “mouse over”).

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Παράδειγμα View, ViewModel και data bindings για εφαρμογή ερωτηματολογίου.
- https://msdn.microsoft.com/en-us/library/hh821028.aspx



Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Η σύνδεση δεδομένων μπορεί να αφορά σε οποιοδήποτε είδος δεδομένων.
  - Δεδομένα από βάσεις δεδομένων, κ.α. - πρέπει να μετατραπούν σε κάποιο format με το οποίο θα τα διαβάσετε, π.χ. JSON.
- Δεδομένα στυλ – πρέπει να έχετε προηγουμένως φτιάξει style template.
- Δεδομένα που βρίσκονται στη διεπαφή σας ήδη (εντός κάποιου UI component, π.χ. στις ιδιότητες Text, Content) – είδαμε τέτοιο παράδειγμα προηγουμένως.
- **Δεδομένα από κάποια κλάση του προγράμματος σας.**
  - Θα πρέπει να φτιάξετε τη ‘λογική’ του προγράμματος σας.
  - Ή, επειδή ως σχεδιαστές δεν θα πρέπει να απαιτείται να κάνετε κάτι τέτοιο, μπορείτε απλά **να φτιάξετε μια μέθοδο που να δίνει κάποια αρχικά/δοκιμαστικά δεδομένα.**

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Απλό παράδειγμα MVVM. Σύνδεση δεδομένων String → TextBlock.
  - Για την περίπτωση WPF Project…
  - Έστω ότι φτιάχνετε ένα Grid-based UI και σε κάθε Κελί θέλετε να έχετε
    - 1 εικόνα υποβάθρου
    - 1 κείμενο τίτλου
    - 1 κείμενο σύντομης περιγραφής.
- Πως θα συνδέσετε τα παραπάνω με τη διαδικασία Data Binding?

- Για windows Phone, η διαδικασία είναι λίγο διαφορετική, βλ.
  - https://msdn.microsoft.com/en-us/library/windows/apps/gg521153(v=vs.105).aspx

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- 3 βασικά βήματα:
  - Βήμα 1. Σκαρίφημα της διεπαφής σε Visual Studio+XAML (View)
  - Βήμα 2. Κώδικας C# του ViewModel
  - Βήμα 3. Κώδικας C# του Model.

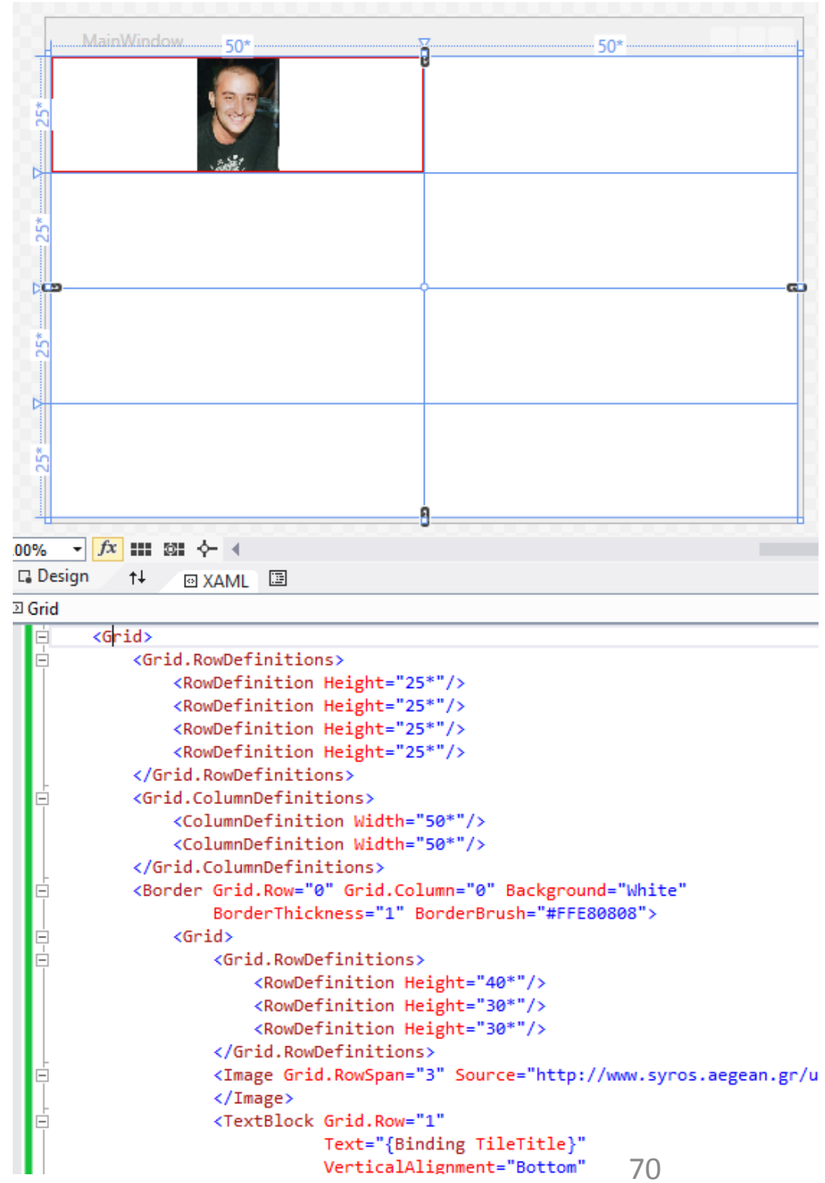
Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Βήμα 1. Σκαρίφημα της διεπαφής σε Visual Studio+XAML (View)
  - Grid, Grid.Row, Grid.Column
  - Εντός ενός κελιού του Grid:
    - Border, Grid with Image, TextBlock, TextBlock
    - Bind TextBlocks to PropertyNames
- Βήμα 2. Κώδικας C# του ViewModel
  - Implement INotifyPropertyChanged.
  - Define properties (and fire PropertyChanged event within set{}!)
- Βήμα 3. Κώδικας C# του Model.
  - Create the ViewModel and assign this to the DataModel
  - Create method for initial values.

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Βήμα 1. Για κάποια δοκιμαστικά δεδομένα, η XAML…

```
<Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="25*"/> <RowDefinition Height="25*"/>
            <RowDefinition Height="25*"/> <RowDefinition Height="25*"/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="50*"/> <ColumnDefinition Width="50*"/>
        </Grid.ColumnDefinitions>
        <Border Grid.Row="0" Grid.Column="0" Background="White"
                BorderThickness="1" BorderBrush="#FFE80808">
            <Grid>
                <Grid.RowDefinitions>
                    <RowDefinition Height="40*"/>  <RowDefinition Height="30*"/>
                    <RowDefinition Height="30*"/>
                </Grid.RowDefinitions>
                <Image Grid.RowSpan="3"
Source="http://www.syros.aegean.gr/users/kgp/images/kgp_photo_small.jpg">
                </Image>
                <TextBlock Grid.Row="1" Text="{Binding TileTitle}"
                           VerticalAlignment="Bottom" Foreground="#FFF10505"
                           FontSize="16">
                </TextBlock>
                <TextBlock Grid.Row="2" Text="{Binding TileParagraph}"
                           Foreground="#FFFF6E00">
                </TextBlock>
            </Grid>
        </Border>
    </Grid>
```



Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Βήμα 2.
  - Solution Explorer | Add Item | Class, δίνουμε όνομα MainWindowViewModel
  - Εδώ **θα ορίσουμε τις ιδιότητες που θα δεσμευτούν** τόσο με τη διεπαφή …
    - (αυτό έχει ήδη γίνει στην XAML)
- … όσο και με το πρόγραμμα μας (MainWindow.cs)
  - Αυτό θα το κάνουμε σε λίγο.
- Επίσης, **θα υλοποιήσουμε τη διεπαφή INotifyPropertyChanged** …
  - … ώστε όταν γίνεται κάποια αλλαγή (από το χρήστη στη διεπαφή / ή το πρόγραμμα μας) να ενημερώνεται (αντίστοιχα: το πρόγραμμα μας / ή η διεπαφή χρήστη)

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Βήμα 2. ViewModel
  - Implement INotifyPropertyChanged.
  - Define properties (and fire PropertyChanged event within set{}!)
- Ο κώδικας C# για το MainWindowViewModel.

```
using System.ComponentModel;

namespace MVVMSimple {
    class MainWindowViewModel: INotifyPropertyChanged {
        public event PropertyChangedEventHandler PropertyChanged;

        public void OnNotifyPropertyChange(string propertyName) {
            if (PropertyChanged != null) {
                PropertyChanged.Invoke(this, new PropertyChangedEventArgs(propertyName));
            }
        }

        private String tileTitle;
        public String TileTitle {
            get { return tileTitle; }
            set {
                if (tileTitle != value) {
                    tileTitle = value;
                    OnNotifyPropertyChange("TileTitle");
                }
            }
        }

        private String tileParagraph;
        public String TileParagraph {
            get { return tileParagraph; }
            set {
                if (tileParagraph != value) {
                    tileParagraph = value;
                    OnNotifyPropertyChange("TileParagraph");
                }
            }
        }
    }
}
```

Σύνδεση δεδομένων (Data binding) σε μεγαλύτερο βάθος

- Βήμα 3. Κώδικας C# του Model.
  - Create the ViewModel and assign this to the DataModel
  - Create method for initial values
- Ο κώδικας C# για το Model.

```
public partial class MainWindow : Window {
    MainWindowViewModel viewModel;

    public MainWindow() {
        InitializeComponent();
        viewModel = new MainWindowViewModel();
        DataContext = viewModel;
        InitialValues();
    }

    public void InitialValues() {
        viewModel.TileTitle = "An old photo";
        viewModel.TileParagraph = "this photo was taken
a long time ago, ...";
    }
}
```