# Adaptive UIs

Contents

[Adaptive UIs 1](#_Toc436161236)

[Doelstelling 2](#_Toc436161237)

[UWP & Device families 2](#_Toc436161238)

[UWP 2](#_Toc436161239)

[Device family (zie **Universal Windows Platform** (UWP)) 3](#_Toc436161240)

[Adaptive UI voorbeelden 4](#_Toc436161241)

[Financiën app (Microsoft Money, geldzaken) 4](#_Toc436161242)

[Nieuws app 4](#_Toc436161243)

[Foto’s 4](#_Toc436161244)

[Kalender 4](#_Toc436161245)

[Store application 5](#_Toc436161246)

[Cross device family support 5](#_Toc436161247)

[Adaptive UI support 5](#_Toc436161248)

[Adaptive Scaling 5](#_Toc436161249)

[Adaptive UI: nieuwe controls 5](#_Toc436161250)

[Pivot control 6](#_Toc436161251)

[Adaptive UI: Visual State Triggers 7](#_Toc436161252)

[XAML views 9](#_Toc436161253)

[Werkwijze: Bereid u voor op adaptive UI’s 9](#_Toc436161254)

[Creating teams: intro 9](#_Toc436161255)

[Creating teams: ContentControl 9](#_Toc436161256)

[Creating Teams: DataTemplate basis 10](#_Toc436161257)

[Creating Teams: DataTemplate, VisualStateManager.VisualStateGroups 10](#_Toc436161258)

[Leesmateriaal (niet verplicht, geen examen material) 11](#_Toc436161259)

## Doelstelling

Adaptive UI’s pogen voor applicaties hetzelfde te zijn als [responsive websites](https://en.wikipedia.org/wiki/Responsive_web_design):



## UWP & Device families

### UWP

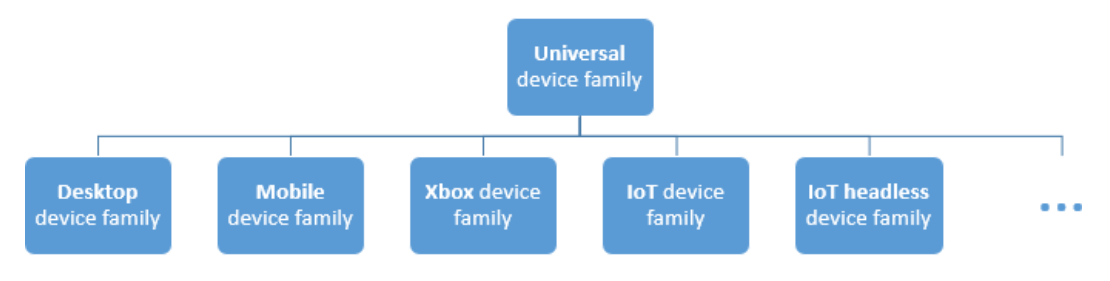
Windows 10 introduceert het [**Universal Windows Platform** (UWP)](https://msdn.microsoft.com/en-us/library/windows/apps/dn894631.aspx), een WinRT- evolutie die op alle Windows 10 devices kan uitgevoerd worden: de **write once, run anywhere** visie (beperkt tot Windows 10 dan wel ☺). Onderstaande figuur illustreert de situatie:



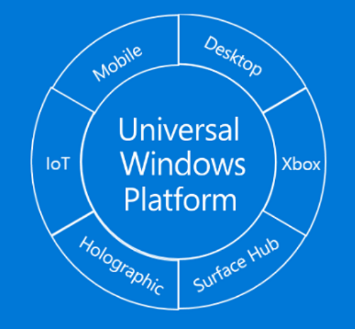
Een belangrijk onderdeel bij het ontwerpen van dergelijke software (die op alle devices uitgevoerd kan worden) is de UI die zich aanpast aan het device: **Adaptive UI’s**. Een belangrijke factor hierbij zijn de afmetingen van het scherm, de device karakteristiek waar de meeste Adaptive UI voorbeelden op gebaseerd zijn.

**Niet UI opmerking**: adaptive code kan de device- familie detecteren en zo device specifieke calls uitvoeren. Zo kan het programma de meerwaarde van een device familie gebruiken indien geïnstalleerd op dergelijk device (een mobiel apparaat kan gebruikt worden om een SMS te versturen, wat we met een Xbox niet het geval is).

### Device family (zie [**Universal Windows Platform** (UWP)](https://msdn.microsoft.com/en-us/library/windows/apps/dn894631.aspx))

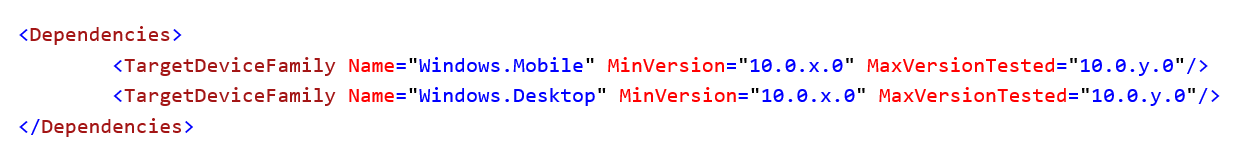


Een device family bundelt een set API’s en vormt zo de basis van het OS.

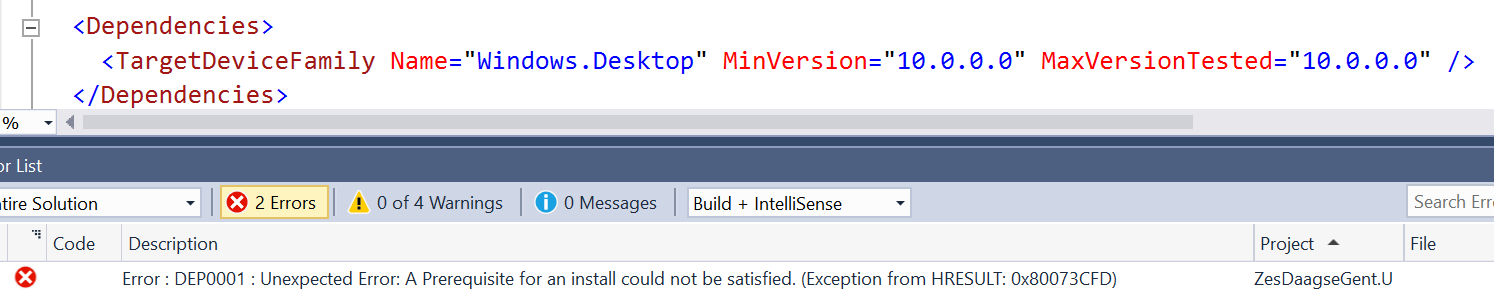
**Universal Device Family** bevat API’s die in elke device family terugkeren. Het plaatje op [Develop for Windows 10 – (1) The Concept of UWP](http://windowscomments.com/?p=1359) is iets duidelijker. Programma’s die enkel deze API’s gebruiken kunnen dan ook op elk Windows-10 device uitgevoerd worden.

1Develop for Windows 10 – (1) The Concept of UWP

U kan in de manifest file via de [TargetDeviceFamily (Windows 10)](https://msdn.microsoft.com/en-us/library/windows/apps/dn986903.aspx) tag instellen op welke device families de app gedeployed kan worden:



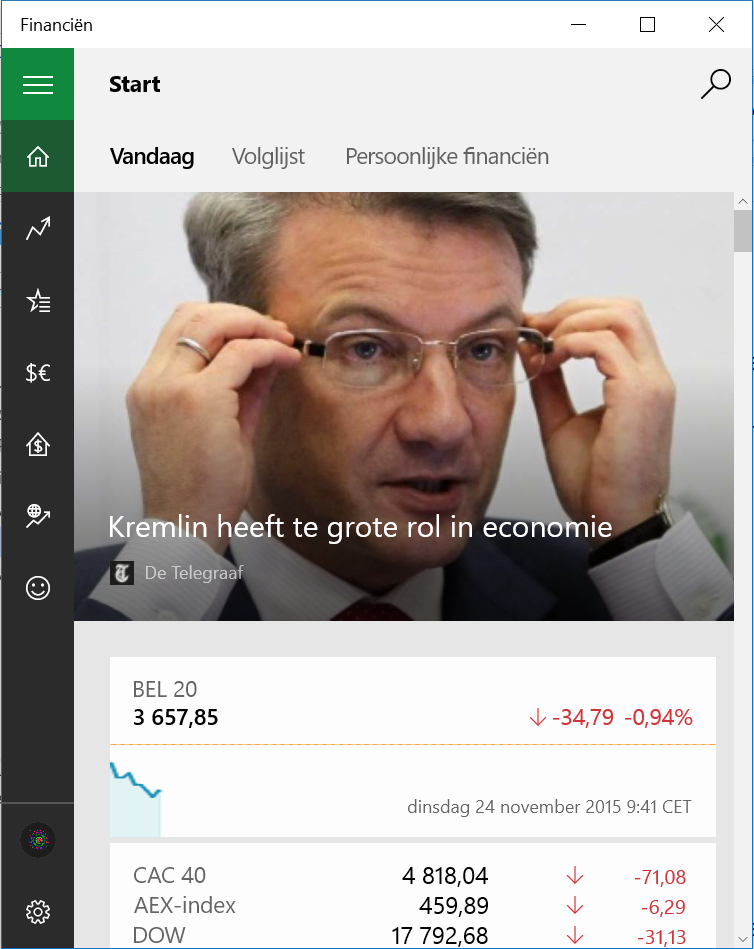
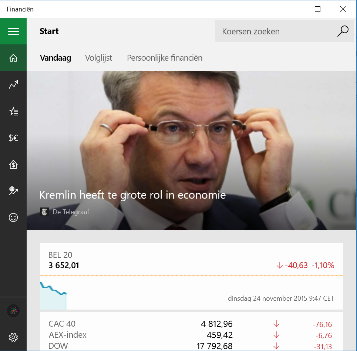
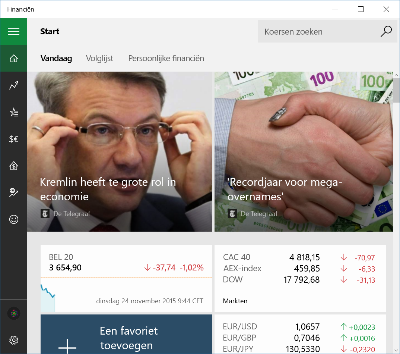
Indien we de TargetDeviceFamily beperken tot Windows.Desktop en de toepassing proberen op te starten op een Mobile device faalt dit met onderstaande uitvoeringsfout:



## Adaptive UI voorbeelden

### Financiën app (Microsoft Money, geldzaken)

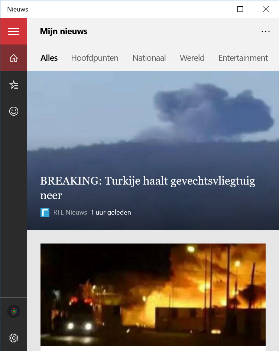
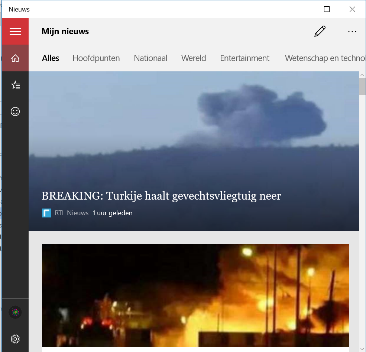
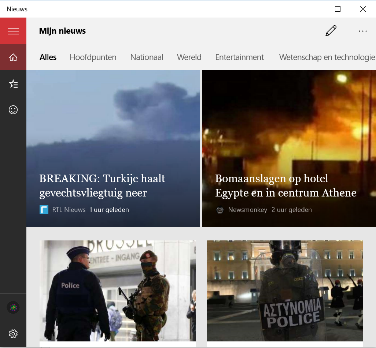
Bemerk het ontbreken van een horizontale scrollbar: het scherm glijdt niet over een gegeven met vaste breedte, de gegevens worden verschillend gepresenteerd afhankelijk van de schermbreedte. In een desktop omgeving kunnen we gemakkelijk verschillende schermbreedtes simuleren, wat op een mobiel toestel (niet windowed) veel minder evident is.



De layout wordt opgedeeld in gehele kolommen afhankelijk van de beschikbare breedte.

Bemerk ook dat de navigatie op de dekstop versie zowel het Hamburger menu als een CommandBar bevat.

### Nieuws app

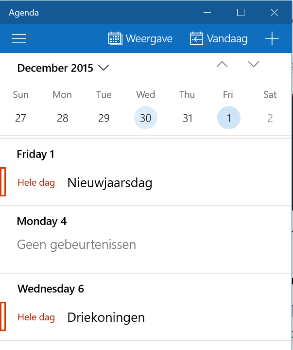
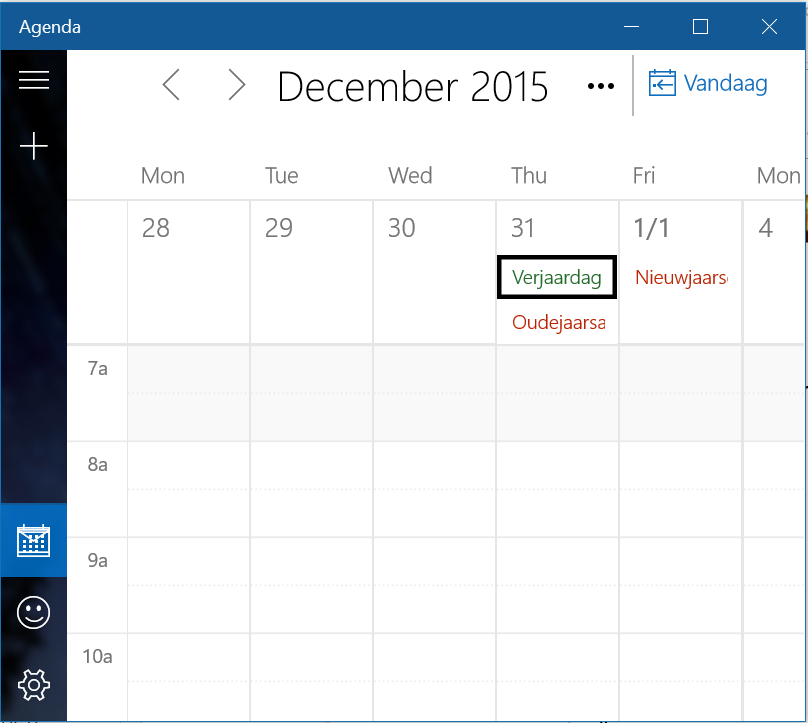


Gelijkaardig aan de Finance app

### Foto’s

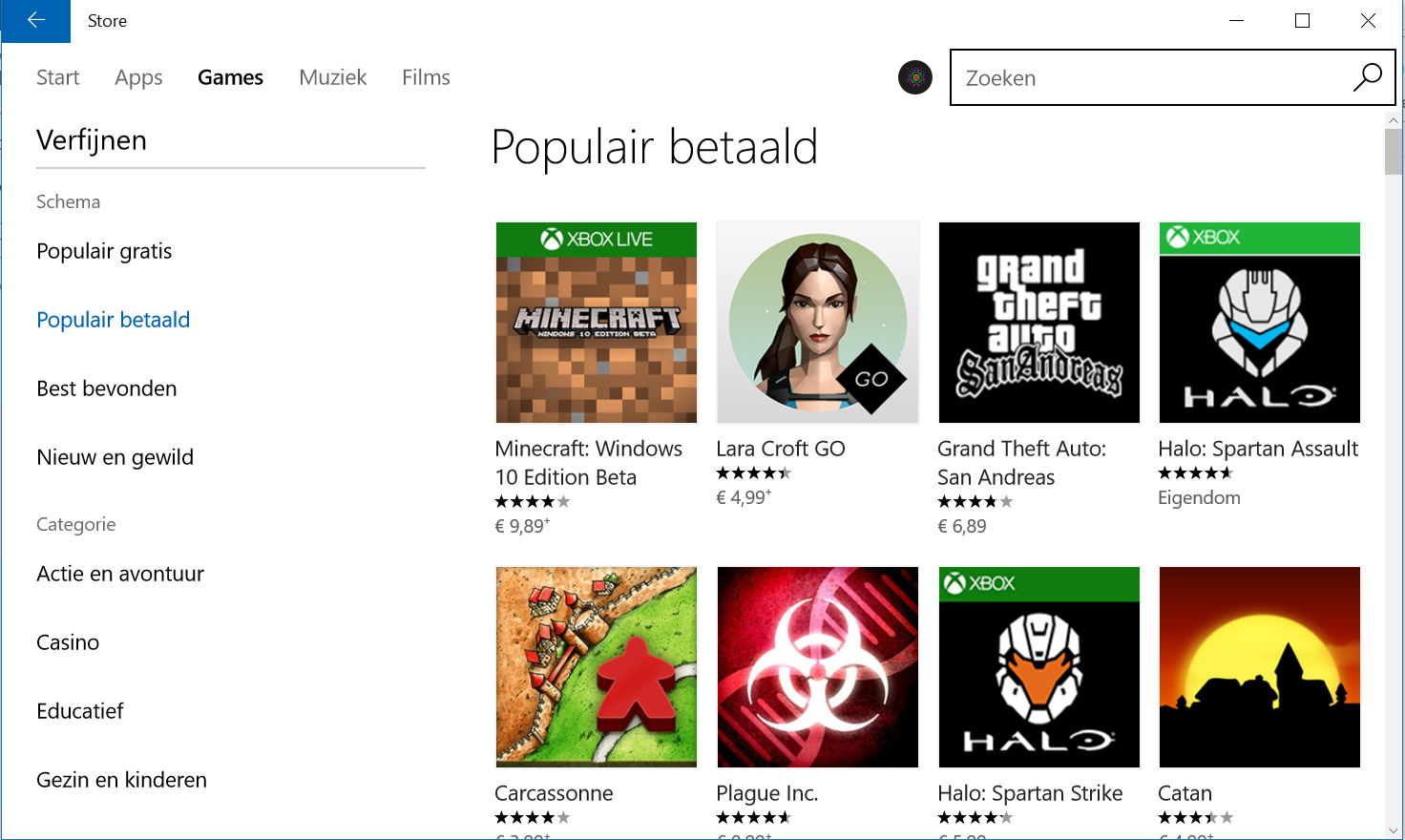
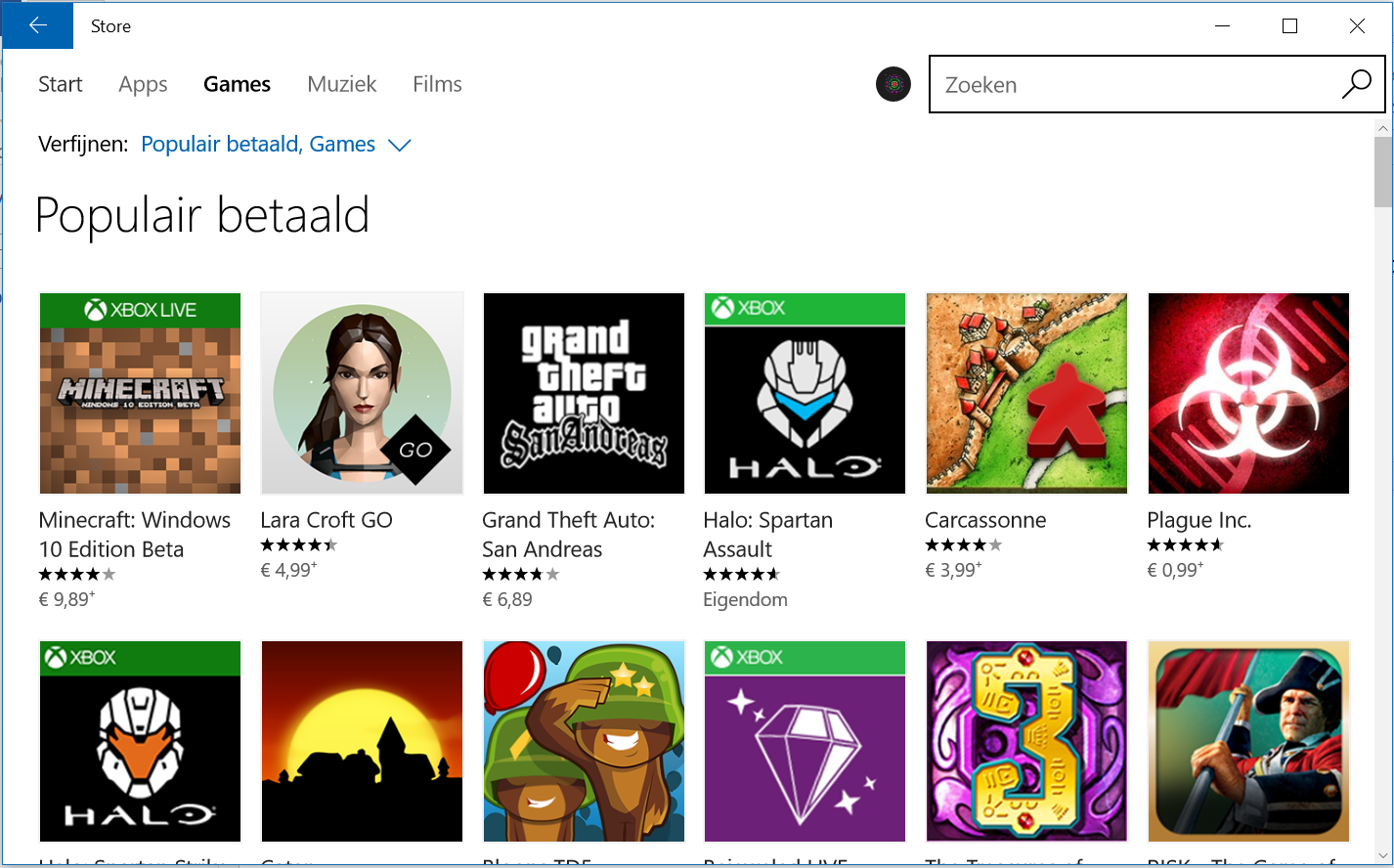
De foto app vertoont een gelijkaardig gedrag.

### Kalender



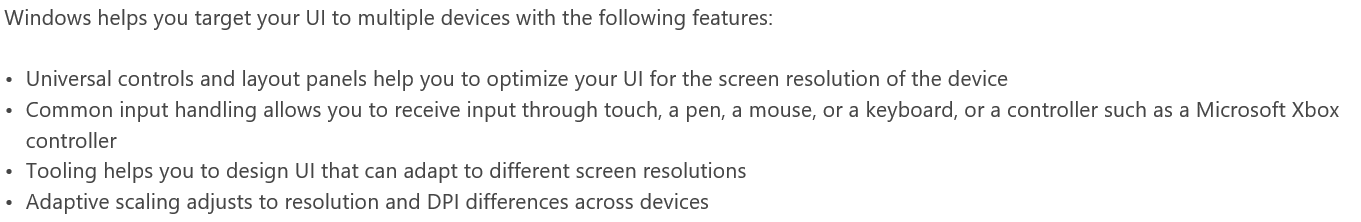
Hierboven worden dezelfde gegevens verschillend gepresenteerd in functie van de beschikbare breedte.

### Store application



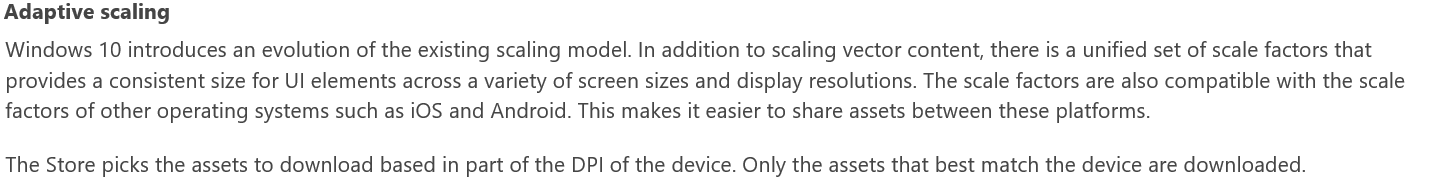
Indien het scherm breed genoeg is wordt een extra navigatie getoond: een adaptive UI.

## Cross device family support



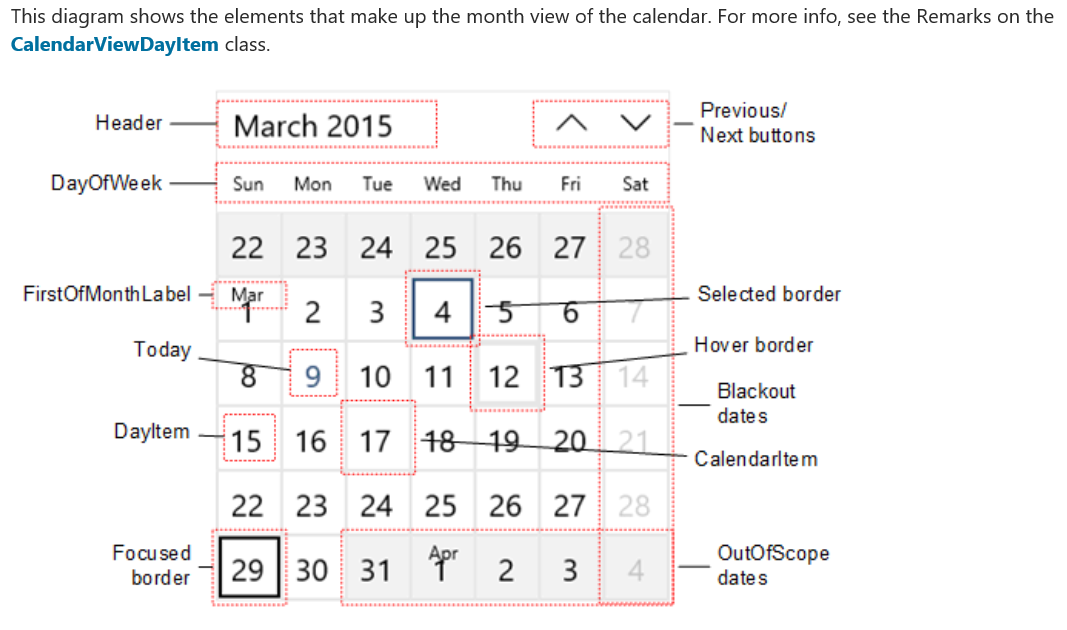
## Adaptive UI support

### Adaptive Scaling



### Adaptive UI: nieuwe controls

#### Calendar control



#### [RelativePanel](https://msdn.microsoft.com/en-us/library/windows/apps/windows.ui.xaml.controls.relativepanel.aspx)

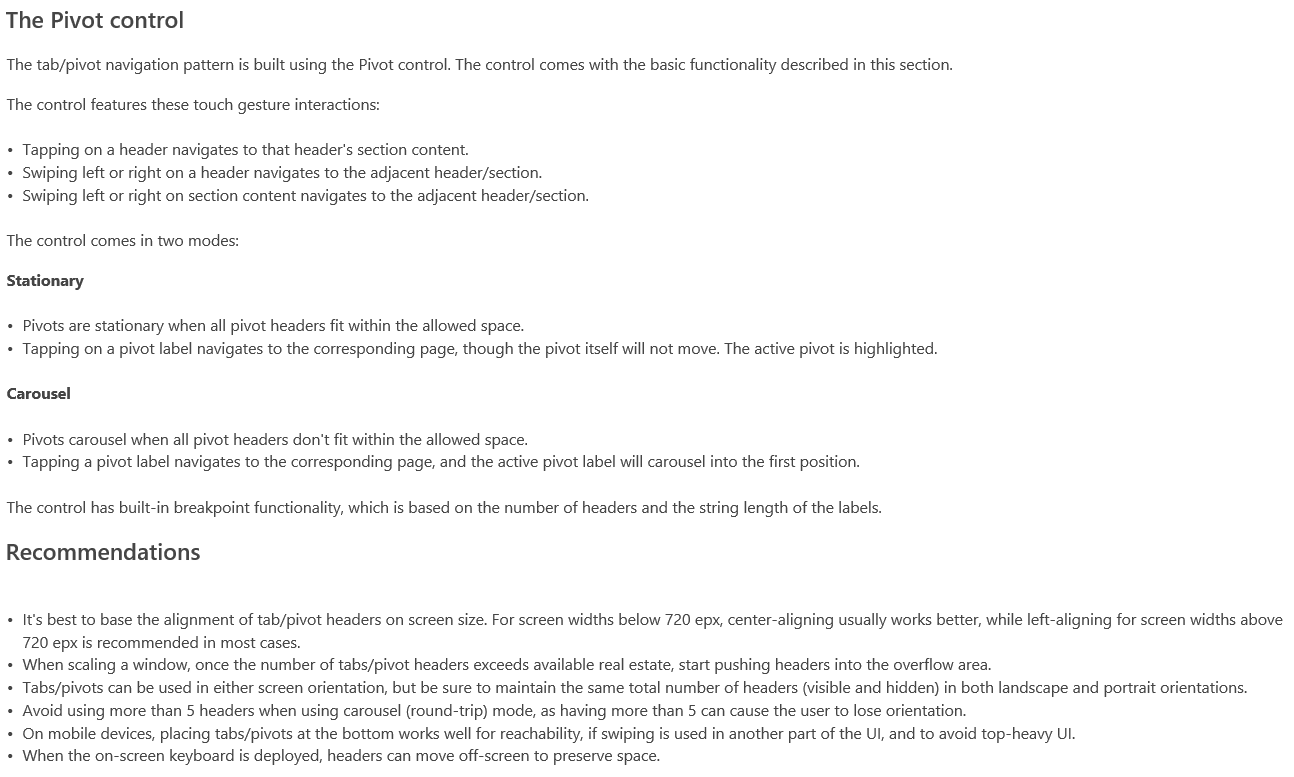
Positioneert controls ten opzichte van elkaar. Dit verschilt van een Grid die zijn child- controls in cellen plaatst. Voor de geïnteresseerden: [UWP-017 - XAML Layout with RelativePanel (Channel 9)](https://channel9.msdn.com/Series/Windows-10-development-for-absolute-beginners/UWP-017-XAML-Layout-with-RelativePanel).

Wat voorbeeld XAML code uit de tekst maakt dit wellicht duidelijker:



Bemerk dat de syntax gelijkaardig is aan de Grid.Column syntax. Nu wordt gebruik gemaakt van bijvoorbeeld RelativePanel.RightOf dependency properties (DependencyProperties worden toegelicht in de cursus .NET programmer solutions). U zal deze positioneringen wijzigen in verschillende states (zie verder)

### Pivot control



(bovenstaande screenshot is onderdeel van [Guidelines for tabs and pivots](https://msdn.microsoft.com/en-us/library/windows/apps/dn997788.aspx#examples).

### Adaptive UI: Visual State Triggers

#### VisualStateManager

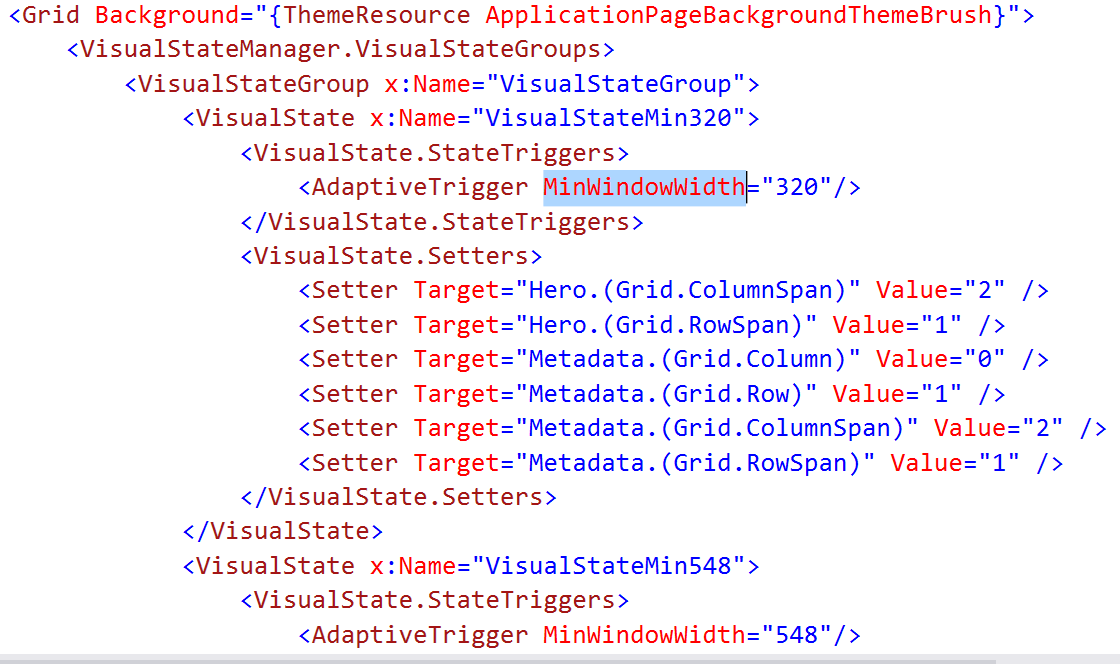
De VisualStateManager definieert ‘toestanden’ (states) en hun overgangen (transitions). In deze les zullen we vooral de states bespreken, en minder aandacht besteden aan de transitions.

De VisualStateManager class is reeds langere tijd aanwezig in het .NET framework en werd tot nu toe vooral gebruikt om states en hun geanimeerde overgangen te beschrijven. De navigatie tussen verschillende states gebeurt in de codebehind (zie het hoofdstuk omtrent lookless controls (templated controls, custom controls) in de cursus .NET programmer solutions ☺).

UWP introduceert statetriggers die toelaten om declaratief (via XAML) State- voorwaarden te definiëren: indien devoorwaarde voldaan is wordt deze state van toepassing. De states in een StateGroup moeten disjunct zijn (er kan slechts één state van een stategroup actief zijn).

#### AdaptiveTrigger

De UWP omgeving komt met één voorgedefinieerde StateTrigger: [AdaptiveTrigger](https://msdn.microsoft.com/query/dev14.query?appId=Dev14IDEF1&l=EN-US&k=k(Windows.UI.Xaml.AdaptiveTrigger);k(VS.XamlEditor);k(TargetFrameworkMoniker-.NETCore,Version%3Dv5.0)&rd=true). Die kan als volgt gebruikt worden:



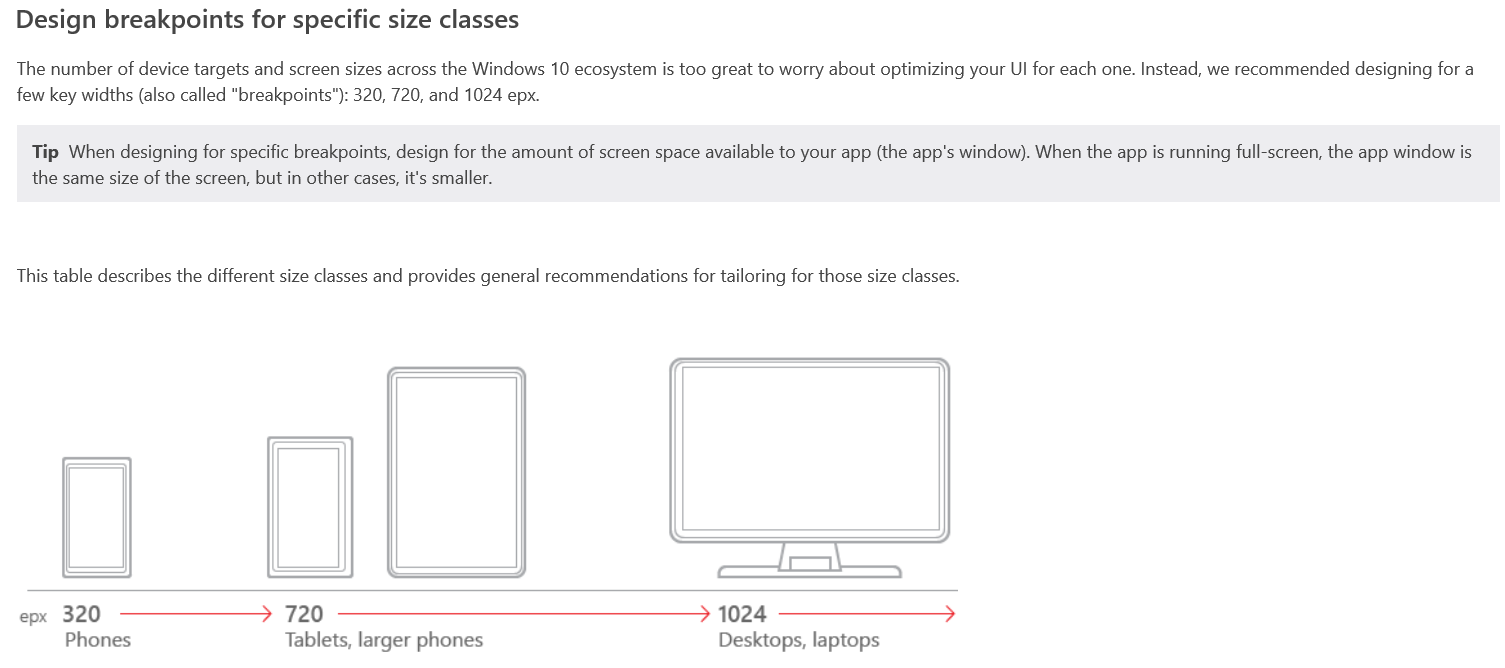
De bovenste state wordt actief indien het getoonde Window minstens 320 pixels breed is. Indien dit Window minstens 548 pixels breed is wordt de tweede visual state actief. U merkt dat een actieve state Setters kan definiëren die (in dit voorbeeld) een control verplaatsen binnen een Grid. Ook de RelativePanel eigenschappen kunnen ingesteld worden, net zoals ongeveer gelijk welke andere eigenschap van een control.

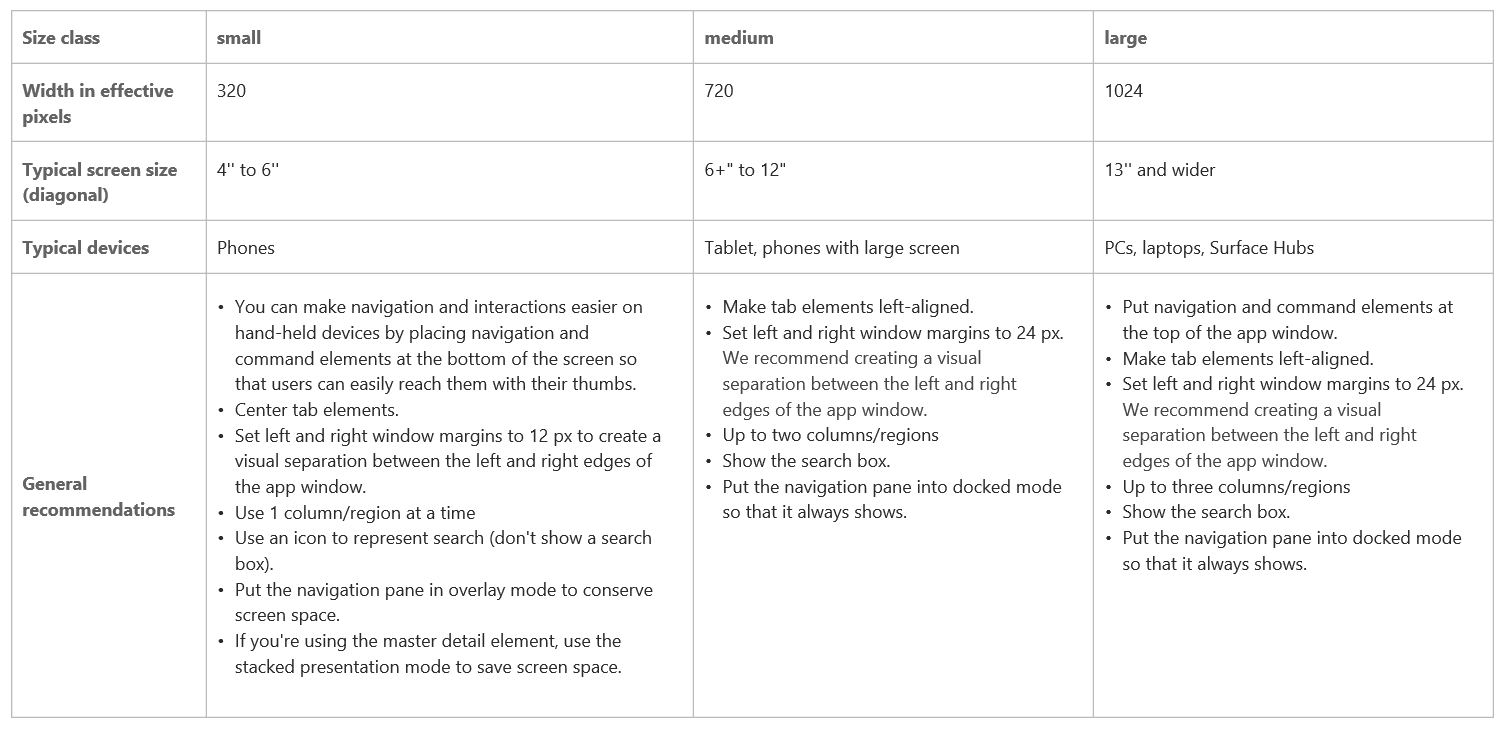
Bemerk dat in bovenstaand voorbeeld een breedte tussen 0 en 320 niet tot een statewijziging leidt ten opzichte van de originele definitie.

Een eerste voorbeeld oefening omtrent adaptive UI’s zal ik overnemen van de Windows 10 developers day. U vindt het materiaal hiervoor op leho.

#### Design breakpoints for specific size classes

Onderstaande screenshots zijn genomen uit [Responsive design 101 for Universal Windows Platform (UWP) apps](https://msdn.microsoft.com/en-us/library/windows/apps/dn958435.aspx)

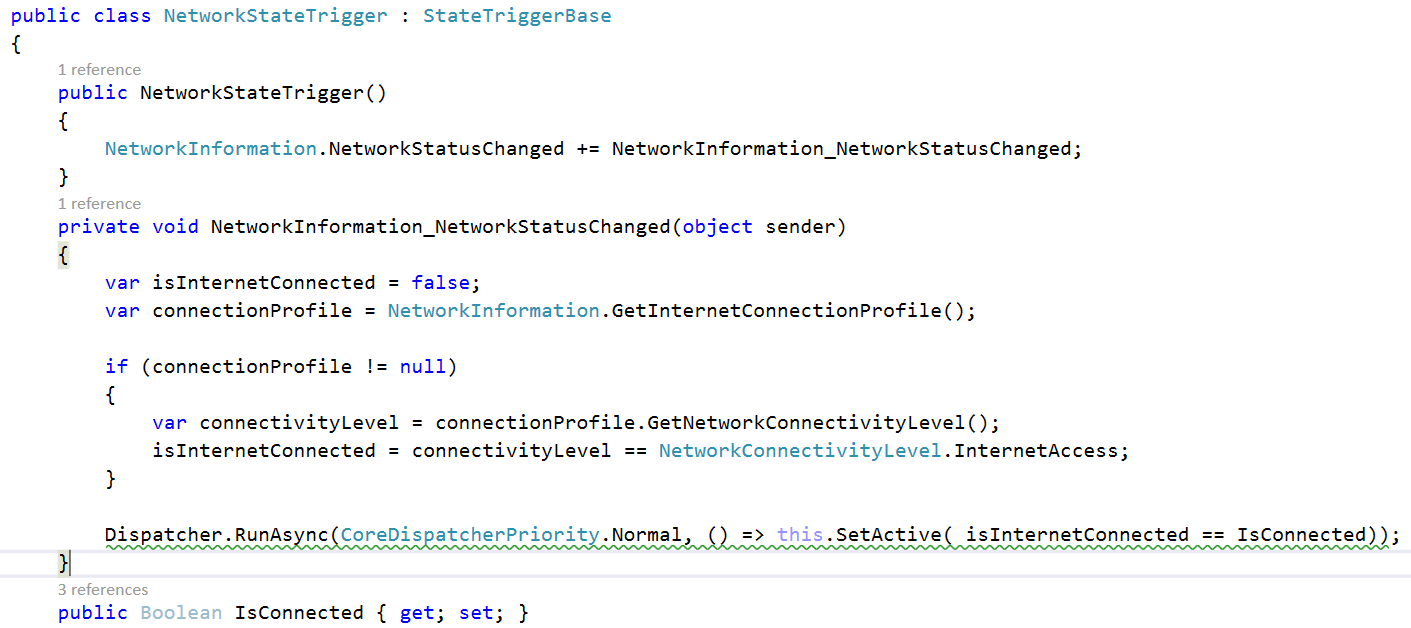




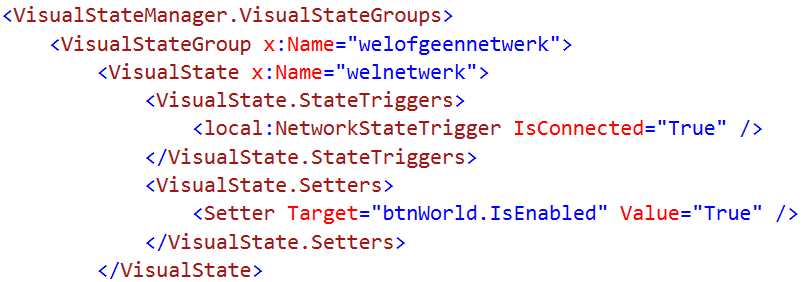
#### NetworkStateTrigger

Het is helemaal niet moeilijk om zelf een StateTrigger te ontwikkelen. Ervend van [StateTriggerBase](https://msdn.microsoft.com/en-us/library/windows.ui.xaml.statetriggerbase.aspx) voorzien we een klasse van de gepaste logica om op het juiste moment de SetActive/1 routine uit te voeren:

1. We hebben een property (IsConnected in onderstaande voorbeeld) waarmee devoorwaarden worden gezet vooraleer de state actief wordt gezet. Er kunnen uiteraard meerdere properties zijn die worden ingesteld
2. In de constructor van de StateTrigger wordt een event opgevangen dat invloed heeft op de State- situatie (NetworkInformation.NetworkStatusChanged in het voorbeeld, SizeChanged (van het Window) indien we de MinWidth zouden willen hermaken))
3. Wanneer de situatie zo blijkt te zijn dat aan de voorwaarden van de state is voldaan wordt SetActive(True) uitgevoerd, anders SetActieve(False)



De groen onderlijnde oproep is nodig omwille van multithreading redenen: het NetworkStatusChanged event wordt uitgevoerd op een extra thread, die niet gebruikt mag worden om de UI aan te passen. Omdat een state changes allerhande setters uitvoert wijzigt dit de UI, wat op de UI- thread moet gebeuren. We keren terug naar de UI-thread door middel van de Dispatcher oproep (zie .NET programmer solutions voor meer details).



### XAML views

Indien de verschillende DeviceFamilies een sterk verschillende layout hebben kan het interessanter zijn om voor elke DeviceFamily een aparte layout te maken, eerder dan te proberen met state- setters allerhande zaken te verplaatsen, morphen, deleten en creeëren.

De topic [Tailored Layouts](https://msdn.microsoft.com/en-us/library/windows/apps/mt228350.aspx) beschrijft het proces om per DeviceFamily een xaml-view toe te voegen aan een solution. De tekst [THREE WAYS TO SET SPECIFIC DEVICEFAMILY XAML VIEWS IN UWP](http://igrali.com/2015/08/02/three-ways-to-set-specific-devicefamily-xaml-views-in-uwp/) beschrijft drie manieren om hetzelfde te bekomen. Persoonlijk vind ik deze laatste tekst duidelijker dan de voorgaande tekst.

Wat u voor deze cursus dient te onthouden: indien de logica van een Page over DeviceFamilies heen dezelfde blijft, maar de visualisatie wijzigt sterk, dan is het mogelijk om per DeviceFamily een xaml- view te definiëren die de layout beschrijft. Deze xaml- view gebruikt de code behind van de oorspronkelijke Page. Indien er geen xaml view voor de uitvoerende DeviceFamily werd gedefinieerd wordt de default definitie gebruikt.

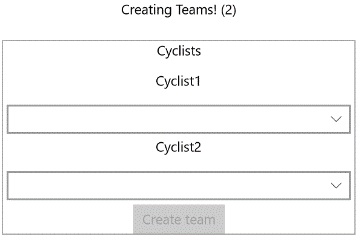
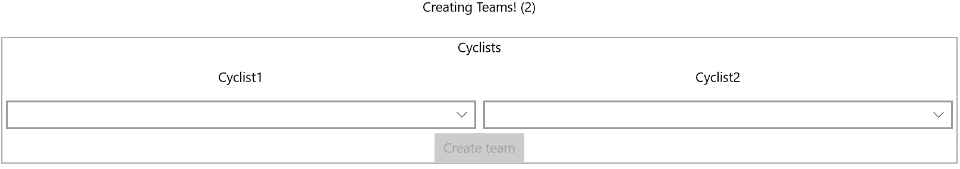
### Adaptive UI: Grid

Belangrijk bij het maken van UI’s die schalen in functie van de schermgrootte is de Grid klasse. De proportionele verhoudingen (\* breedtes en hoogtes) maken het mogelijk om één UI’s te maken die goed functioneert op toestellen wiens schermgrootte dicht bij elkaar liggen. Daarom kunnen we bijvoorbeeld één UI maken die werkt op zowel 4,5 inch en 4,8 inch toestellen. Indien we grote schermverschillen wensen te ondervangen maken we gebruik van bovenstaande technieken

## Werkwijze: Bereid u voor op adaptive UI’s

Indien u samenhangende code elementen hebt dan kan u deze best in een DataTemplate (voor het ViewModel) stoppen, ook indien deze elementen niet een model item vormen. Deze werkwijze zal er voor zorgen dat u deze gemeenschappelijke stukken gemakkelijk elders zal kunnen recupereren. U zal dan gebruik maken van ContentControl placeholders waarop een ContentTemplate wordt ingesteld.

### Creating teams: intro

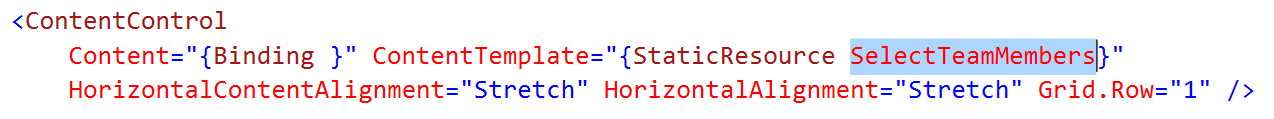


Bovenstaande visualisaties laten toe om een duo (Team) voor een zesdaagse competitie samen te stellen. Indien het scherm breed genoeg is worden de renners naast elkaar geplaatst, anders komen ze onder elkaar te staan. Bemerk dat de linke visualisatie hoger is dan de rechter visualisatie.

Omdat deze controls tezamen horen werden ze in een datatemplate SelectTeamMembers geplaatst.

### Creating teams: ContentControl

In de xaml-code van de page werd volgende code opgenomen:



* Een ContentControl visualiseert zijn Content met behulp van de ingestelde ContentTemplate. De Content- Binding in bovenstaande XAML-code stelt de volledige DataContext van de ContentControl in als Content. Dit wordt de datacontext van de XAML code toegepast via de ContentTemplate.

### Creating Teams: DataTemplate basis



Bovenstaande template code is behoorlijk normaal, enkel het toevoegen van de UserControl tag (ter voorbereiding van de VisualStateManager) is reeds extra toegevoegd.

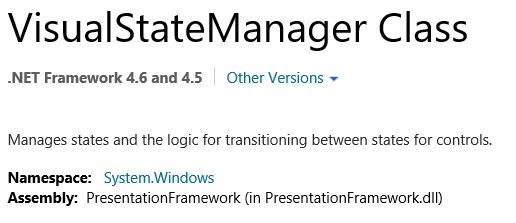
### Creating Teams: DataTemplate, VisualStateManager.VisualStateGroups



Bemerk dat de setters kunnen gebruikt worden om controls te verplaatsen binnen een Grid. Op gelijkaardige wijze kunnen ze natuurlijk binnen een RelativePanel verplaatst worden.

Wat u wou dat u niet vergeten was!

Visualstategroups tag moet onder frameworkelement: usercontrol in datatemplate toevoegen, en dan onder eerste Tag

Indien u in een DataTemplate een VisualStateManager tag wenst te gebruiken moet u zich aan volgende afspraken houden:

* Plaats een UserControl tag in de DataTemplate
* Plaats een UIElement (Grid?) in de UserControl;
* Plaats de VisualStateManager onmiddellijk onder dit UIElement.

# Leesmateriaal (niet verplicht, geen examen material)

* [Responsive design 101 for Universal Windows Platform (UWP) apps](https://msdn.microsoft.com/en-us/library/windows/apps/dn958435.aspx)
* [Introduction to Universal Windows Platform (UWP) apps for designers](https://msdn.microsoft.com/en-us/library/windows/apps/dn958435.aspx)
* [What’s new for developers in Windows 10](https://dev.windows.com/en-US/getstarted/whats-new-windows-10)