# Oefeningen Xml Technologie Javacript 1

**Vooraf** : wanneer je het oefenproject *SilverlightJavascript* opent en de website opstart, zal je zien dat de grid niet het volledige scherm vult. Dat zou normaal moeten, omdat je in de xaml geen width en height opgeeft.

Oplossing : zet zowel in het object element in default.html als in de css file de width en de height op 100%.

<object

id="DefaultPage"

width="100%"

height="100%"

type="application/x-silverlight">

<param name="source" value="Xaml/Default.xaml"/>

</object>

body

{

width: 100%;

height: 100%;

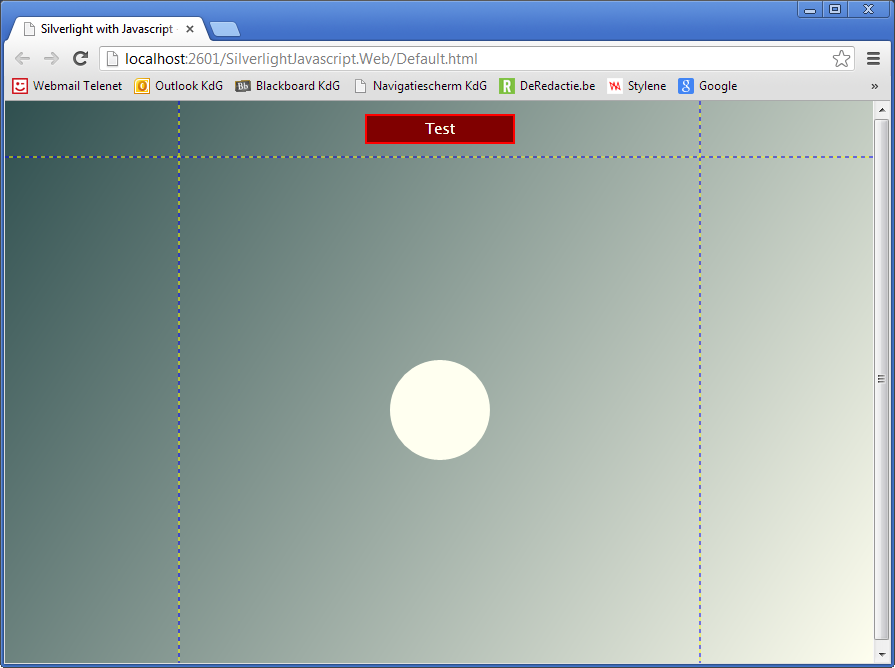
padding: 0;

margin: 0;

}

Save dit project en gebruik het als sjabloon voor de volgende oefeningen, en voor de praktijkopdracht.

1. Vertrek van het oefen project, en maak er het volgende van (intypen in Visual Studio):



Een grid met 3 kolommen en 2 rijen. 1 Button (= Border + TextBlock) in de middelste kolom, bovenste rij, en 1 ellipse van 100 op 100 pixels in de middelste kolom, tweede rij. Geef de Border ook een attribuut *Cursor* met als waarde “Hand”. Als je nu over de Border beweegt verandert de cursor.

* Als je op de button klikt, vergroot de cirkel (eenmaal) in beide richtingen met 100 pixels.
* Als je op de button klikt, verandert de kleur van de cirkel in Green.
* Als je op de button klikt, verspringt de cirkel naar de eerste kolom. Om met de column van een Grid te werken in Javascript gebruik je [‘Grid.Column’]. Anders geraakt de Javascript parser in de war. Bijvoorbeeld:

ellipse['Grid.Column'] = 2;

* Als je de eerste keer op de button klikt, verspringt de cirkel naar de eerste kolom. Als je nu nog eens op de button klikt, verspringt hij naar de derde kolom. Tip: gebruik een globale variabele om te onthouden de hoeveelste klik het is.
* Zorg er nu voor dat je kan blijven klikken en dat de cirkel van kolom naar kolom blijft springen bij elke klik.

1. Werk verder op de vorige oefening. Je zet nu 3 cirkels in de kolommen. De eerste en de derde zet je in de xaml code op invisible (surf eens rond het xaml attribuut *Visibility*). Programmeer dan de volgende use cases:

* Als je op de button klikt, vergroot de middelste cirkel (eenmaal) met 5 pixels.
* Als je blijft klikken, blijft de cirkel vergroten, oneindig.
* Als je blijft klikken, blijft de cirkel vergroten tot wanneer de width groter wordt dan 400 pixels.
* Als je op de button klikt, verschijnen de 2 andere cirkels.
* Als je blijft klikken, verschijnen en verdwijnen de 3 cirkels, in een oneindige loop.

1. Zelfde grid, 1 rode cirkel van 100 op 100 pixels in de middelste cel. Als je op de cirkel klikt, verandert hij in een blauwe rectangle van dezelfde dimensies. Gebruik hier ook het attribuut *Visibility*. Zet de rectangle al klaar maar onzichtbaar.
2. Zelfde grid, 1 rode cirkel. Als je op de cirkel klikt, wordt hij elke 100 milliseconden 1 pixel kleiner. Gebruik hiervoor setInterval. Meer informatie op

<http://www.elated.com/articles/javascript-timers-with-settimeout-and-setinterval/>.

1. Werk verder op de vorige oefening. Als de cirkel nog 1 pixel groot is, laat je hem terug naar 100 op 100 pixels groeien, ook met setInterval.
2. Programmeer het volgende spelletje. Maak een grid van 3 op 3 cellen en zet er cirkels in. Er zijn 2 rode, 3 ivory en 3 blauwe cirkels. Zet de cirkels in random cellen en laat 1 cel open, want er zijn maar 8 cirkels. Ofwel zet je ze zelf hardcoded op de random plaatsen, ofwel surf je eens naar Math.Random in Javascript om de plaatsing te laten gebeuren.

Dat is de startpositie.

Als je nu op een cirkel klikt, springt hij naar de lege cel en wordt zijn cel de lege cel. De speler kan dit blijven doen tot wanneer de bovenste rij 3 ivory, de tweede rij 3 blauwe en de derde rij 2 rode cirkels en een lege cel bevat. Als dat het geval is, veranderen alle cirkels in het zwart.

Game Over.