

**Project Webontwikkeling 3**

**Semester 3 – 2013 - 2014**

**Yo-Tina Verbraecken & Koen Cornelis**

Inhoudsopgave:

Opgave 3

2 versies 3

Scripting 3

Stap 1 3

Stap 2 3

Stap 3 4

HTML code 6

jQuery plugin 7

$.weather() 7

$.fn.weather() 7

$.fn.weather.defaults 8

Resultaat 9

Bij opstarten 9

Onmiddellijk wordt het weer opgehaald 9

Na ophalen weersvoorspelling 9

Bronnen 9

# Opgave

Het doel is dat je op een mobile (of ander) device via geolocation uitzoekt waar je zit op de aarde (long/lat coordinaten), met deze coordinaten een adres opzoekt en van dit adres het weerbericht voor vandaag en morgen in een webpagina bijzet. Ik verwacht hier een mooie oplossing (icoontjes voor weertype, temperatuur, neerslag, … in een mooi kadertje, zodat het aantrekkelijk is om jouw oplossing te integreren in een bestaande website). Keuze voor JSON of XML of combinatie.

# 2 versies

We hebben er voor gekozen om de gebruiker 2 keuzes te geven: het ‘gewone’ script dat besproken wordt in het onderdeel scripting hier direct onder, en een jquery plugin. Beide versies zijn beschikbaar minified of in volle versie. De jquery plugin wordt besproken na de html, aangezien die in grote lijnen gelijk loopt met de ‘gewone versie’, zij het dan met meer mogelijkheid om aanpassingen te doen.

# Scripting

## Stap 1

Vraag de latitude en longitude op van de gebruiker die je pagina bezoekt dit kan op volgende manier:

Nu heb je de latitude en longitude van je gebruiker vast en dit moet je doorsturen naar een andere functie om deze waarden te bewaren. In het voorbeeld wordt dit doorgestuurd naar de functie getWeather die 2 variabelen verwacht.

function getLocation(){

$.getJSON("http://freegeoip.net/json/", function(location){

getWeather(location.longitude, location.latitude)

})

}

## Stap 2

De functie getWeather(long, lat) wordt opgeroepen. De functie gaat via een ajax call naar de website worldweatheeronline.com voor een xml file op te halen met het weerbericht voor vandaag en de komende 4 dagen.

function getWeather(long, lat){

$("#output").text("Waiting for results...");

$.ajax({ url:"http://howestproxy.appspot.com/proxer?xml=http://api.worldweatheronline.com/free/v1/weather.ashx?q="+lat+"%2C+"+long+"&format=xml&num\_of\_days=5&key=umk9aaumj5fxra4k56evbqf2",

dataType: "xml",

success: doLocation,

error: showError

});

}

Bij succes ga je naar de functie doLocation. Bij een error ga je naar de functie showError. Het datatype dat wordt opgehaald is een xml file. De link: <http://howestproxy.appspot.com/proxer?xml=> wordt gebruikt om een connectie te kunnen maken met de api services van worldweatheronline. Voor deze website te gebruiken heb je een key nodig, die je gratis kan aanvragen door een account aan te maken. Ook geef je 2 parameters aan de link mee via lat en long. Want hierin zitten de coördinaten van je gebruiker.

## Stap 3

Als je naar de error functie gaat komt er enkel een alert op het scherm dat meld om het opnieuw te proberen.

function showError(){

alert("Couldn't get the weather information. Please try again!");

}

Als je naar de functie doLocation gaat. Zal deze functie de gegevens uit de xml file halen en deze verwerken. De code gaat eerst opnieuw een JSON request sturen om de stad, de provincie en het land van de gebruiker. Deze wordt dan bovenaan geplaatst om aan te tonen van welke stad het weer werd opgehaald. Ook wordt er voor gezorgd dat de huidige temperatuur wordt opgehaald.

function doLocation(xml){  
EuropeanValues = !EuropeanValues;

$.getJSON("http://freegeoip.net/json/", function(location){  
 $("#output").text("The results are in for "+location.city + ", " + location.region\_name +", "+ location.country\_name);  
 });

De variabele EuropeanValues houdt bij of de waarden op het formulier in °C zijn of in °F. Bij true is dit °C en bij false is dit °F. (zie verder).

Daarna wordt een lijst getrokken van de xml file met alle weersverwachtingen in. Daarna wordt per item in de lijst gekeken wat de datum is en word dit toegevoegd aan de html file.

var list = $(xml).find("weather");  
var results = "";  
var current = $(xml).find("current\_condition");  
results = "<p class='current'>Current temperature: ";

if (EuropeanValues) {  
 results += current.find("temp\_C").text() + "&deg;C</p>";  
 }  
 else{  
 results += current.find("temp\_F").text() + "&deg;F</p>";  
 }

Hierna worden de gegevens uit de lijst gehaald die nodig zijn voor het afwerken van de file (date, maxTempC, minTempC, maxTempF, minTempF, windSpeedKph, windSpeedMph en weatherIcon). Op het einde wordt de hele string die in results zit toegevoegd aan de html file waar alle style functies op worden toegepast vanuit de stylesheet basic.css . De code &#176; toont een °

list.each(function (){  
 var date = $(this).find("date").text();  
 results += "<p class='weather'><a class='date'>Date: "+ date +"</a><br>";  
if (EuropeanValues) {  
 var maxTempC = $(this).find("tempMaxC").text();  
 var minTempC = $(this).find("tempMinC").text();  
 results += "Max. temperature: " + maxTempC + "&deg;C<br> Min. temperature: "+ minTempC + "&#176;C <br>";  
 var windSpeedKph = $(this).find("windspeedKmph").text();  
 results += "Windspeed Kph: "+windSpeedKph;  
}  
else{  
 var maxTempF = $(this).find("tempMaxF").text();  
 var minTempF = $(this).find("tempMinF").text();  
 results +="Max. temperature: "+maxTempF+"&deg;F <br>Min. temperature: "+minTempF+"&deg;F<br>";   
 var windSpeedMph = $(this).find("windspeedMiles").text();  
 results += "Windspeed Mph: "+windSpeedMph  
}  
 var weatherIcon = $(this).find("weatherIconUrl").text();  
 results += "<br><img src='" +weatherIcon+"'></p>";  
})

Nadat alle gegevens zijn toegevoegd aan het resultaat moet de achtergrond nog veranderen naar het weer dat vandaag van toepassing is. Dit gebeurd via een switch die in de list gaat kijken naar de “weathercode”. Elke code heeft een betekenis zoals je kunt zien hieronder

switch($(list).find("weathercode")){  
 case(116):  
 $("body").addClass("sunny");  
 break;  
 case(119):  
 $("body").addClass("clouds");  
 break;  
 case(112):  
 $("body").addClass("verycloudy");  
 break;  
 case(226):  
 $("body").addClass("rainy");  
 break;  
 default:  
 $("body").addClass("clouds");  
 break;  
 }  
 $("#googleMap").show();  
 $("#results").html(results);  
}

# HTML code

Enkele basiselementen zijn nodig voor dit te laten werken. Maar er is ook scripting van toepassingen zodat Google Maps wordt aangesproken om een kaartje te tekenen met je huidige locatie.

<body>  
 <input type="button" value="Switch to other standards" id="switch">  
 <div id="output">Idle...</div>  
 <div id="results"></div>  
 <div id="googleMap" style="width:500px;height:380px;"></div>  
</body>

Hieronder zie je de code voor Google Maps te laten werken.

function useGoogleMaps(long, lat) {  
 var mapProp = {  
 center:new google.maps.LatLng(lat,long),  
 zoom:10,  
 mapTypeId:google.maps.MapTypeId.ROADMAP  
 };  
 var map=new google.maps.Map(document.getElementById("googleMap"),mapProp);  
 var marker=new google.maps.Marker({  
 position:new google.maps.LatLng(long, lat),  
 });  
 marker.setMap(map);  
 }

Als er op de knop Switch wordt gedrukt wordt getLocation opnieuw aangesproken.

$("#switch").bind("click", function (){  
 getLocation();  
 });

# jQuery plugin

## $.weather()

Dit is op zich een vrij korte functie, omdat de meeste functionaliteit in de defaultwaarden ‘verstopt’ zit. Dit is gedaan om de gebruiker toe te laten zo veel mogelijk aanpassingen te doen naar eigen smaak indien hij dat wenst. De functie aanvaardt daarvoor een object literal waarmee hij de defaultwaarden kan overschrijven. Dit gebeurt in lijn 2: waar er een lege object literal via de methode $.extend() eerst wordt overschreven door de default opties en daarna door de gebruikersopties. Het is op de variabele die zo overschreven wordt dat alle functies later zullen werken, en wel zodat de defaultwaarden gelijk blijven (tenzij de gebruiker ze uitdrukkelijk overschrijft, iets wat later aan bod zal komen).

Lijn 3 is een extra veiligheid. Indien de gebruiker in zijn object literal aangeeft dat hij het formaat van de datum niet wil laten meeveranderen met de rest van het format, maar vergeet om ook de boolean noTimeChange te overschrijven, dan gebeurt dat hier.

Lijn 5 tot 7 voegen een handler-functie toe voor het click event op de button om de formaten te veranderen … maar enkel indien de gebruiker van formaat wil kunnen veranderen.

Lijn 10 tot slot roept de functie getLocation aan en geeft het options object mee als attribuut. Het is deze functie die het begin is van een cascade die tot de juiste output leidt.

## $.fn.weather()

De $.weather() method heeft als grote nadeel dat ze niet chainable is. Iets wat we doorgaans toch van jQuery functies verwachten, daarom bestaat er dus ook een chainable versie die op een jQuery object kan opgeroepen worden als volgt jQuery-object.weather(). Deze functie heeft net als de bovenstaande een optionele parameter bestaande uit een object literal met opties van de gebruiker.

Opnieuw een zeer korte functie die zorgt voor impliciete iteratie. Voor elk DOM-element uit het jQuery-object wordt het id opgezocht, welks dan verwerkt wordt in een nieuwe object literal waarin de opties van de gebruiker verwerkt worden. Vervolgens wordt de $.weather() functie opgeroepen met het keuzeobject als attribuut. Aan het eind van de rit wordt het originele jQuery-object teruggegeven.

## $.fn.weather.defaults

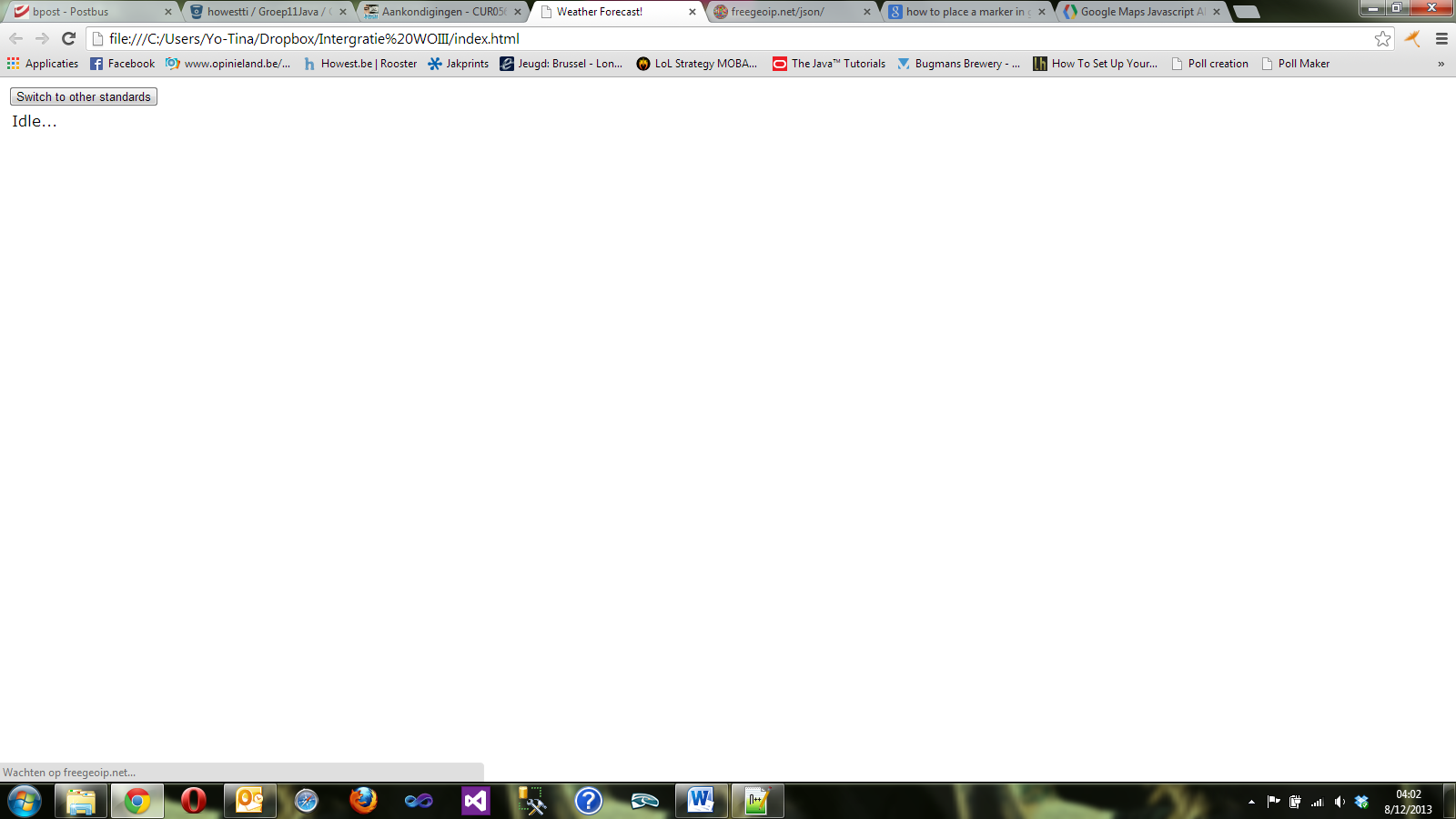
Het defaults object is een object literal waarin alle defaultwaarden opgenomen zijn. Dit betreft dus zowel simple types als functies en andere object literals. Op die manier heeft de gebruiker, indien hij dat wenst de mogelijkheid om de default-waarden permanent te veranderen, hetzij middels een js toewijzing, hetzij via een aanpassing in de broncode van de plugin zelf (na downloaden).

De defaultwaarden staan alfabetisch gerangschikt, deze zijn:

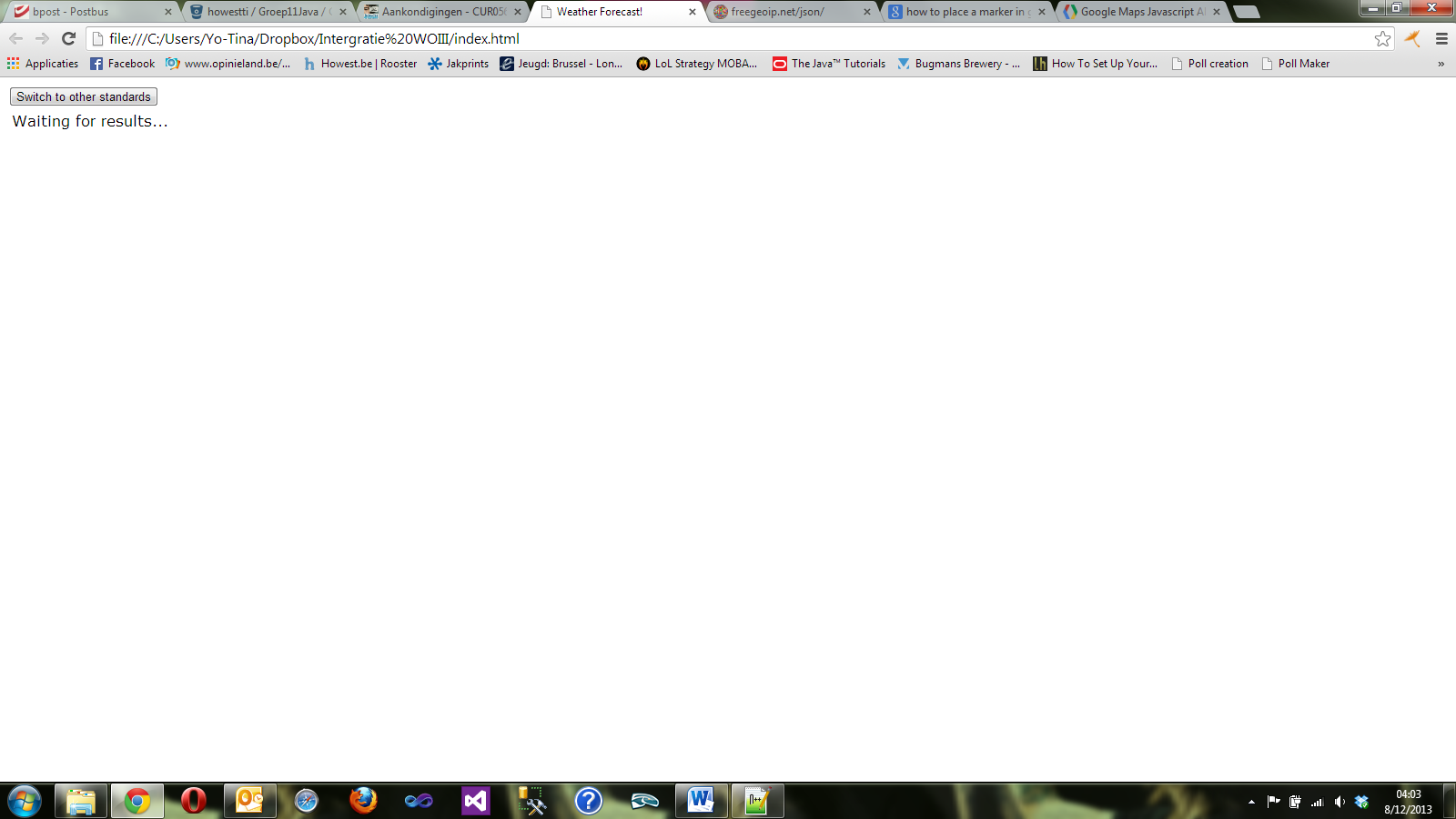
* **button:** default = 'switch'. Het ID van de button waarmee van format veranderd wordt.
* **changeFormat:** default = true. Boolean die aangeeft of je ja of nee van format wil kunnen veranderen.
* **clickHandler:** De functie die gebruikt wordt om een handler te binden aan het click-event voor de button hierboven.
* **europeanValues:** default = true. Boolean die aangeeft of Europees format moet gebruikt worden. True betekent Europees formaat.
* **getLocation:** functie die de json-leteral ophaalt met de locatie van de huidige site-bezoeker.
* **getWeather:** functie die de xml verwerkt en die zorgt voor een correct output van het weerbericht.
* **getXML:** functie die de xml met het weerbericht ophaalt.
* **handleLocation:** functie die de google map toevoegt en die de locatie doorgeeft aan de getXML functie.
* **locationProvider:** de url waar we de geolocatie ophalen. Default = 'http://freegeoip.net/json/'.
* **mapdiv::** default = 'googleMap'. Het ID van de container voor de google map.
* **mapVariables:** een object literal die alle variabelen voor de google map bevat. De default is enkel 'zoom: 10'.
* **noTimeChange:** default = false. Boolean die aangeeft of het tijds-formaat moet wijzigen als de andere warden van Amerikaans naar Europees (en omgekeerd) wijzigen.
* **resultdiv:** default = 'results'. Het ID van de container waar het weerbericht in weergegeven wordt.
* **setMap:** default = true. Boolean die aangeeft of de kaart moet weergegeven worden.
* **setMarker:** default = true. Boolean die aangeeft of de kaart een marker moet bevatten of niet.
* **showError:** functie die opgeroepen wordt als de getXML functie faalt. De default is een pop-up. Vermoedelijk zullen de meeste gebruikers hier een div van maken die wel of niet getoond wordt.
* **timeFormat:** default = true. Boolean die hand in hand gaat met de europeanValues Boolean. Geeft aan of de tijd in het Amerikaans of Europees formaat moet staan. De default waarde betekent Europees format.

# Resultaat

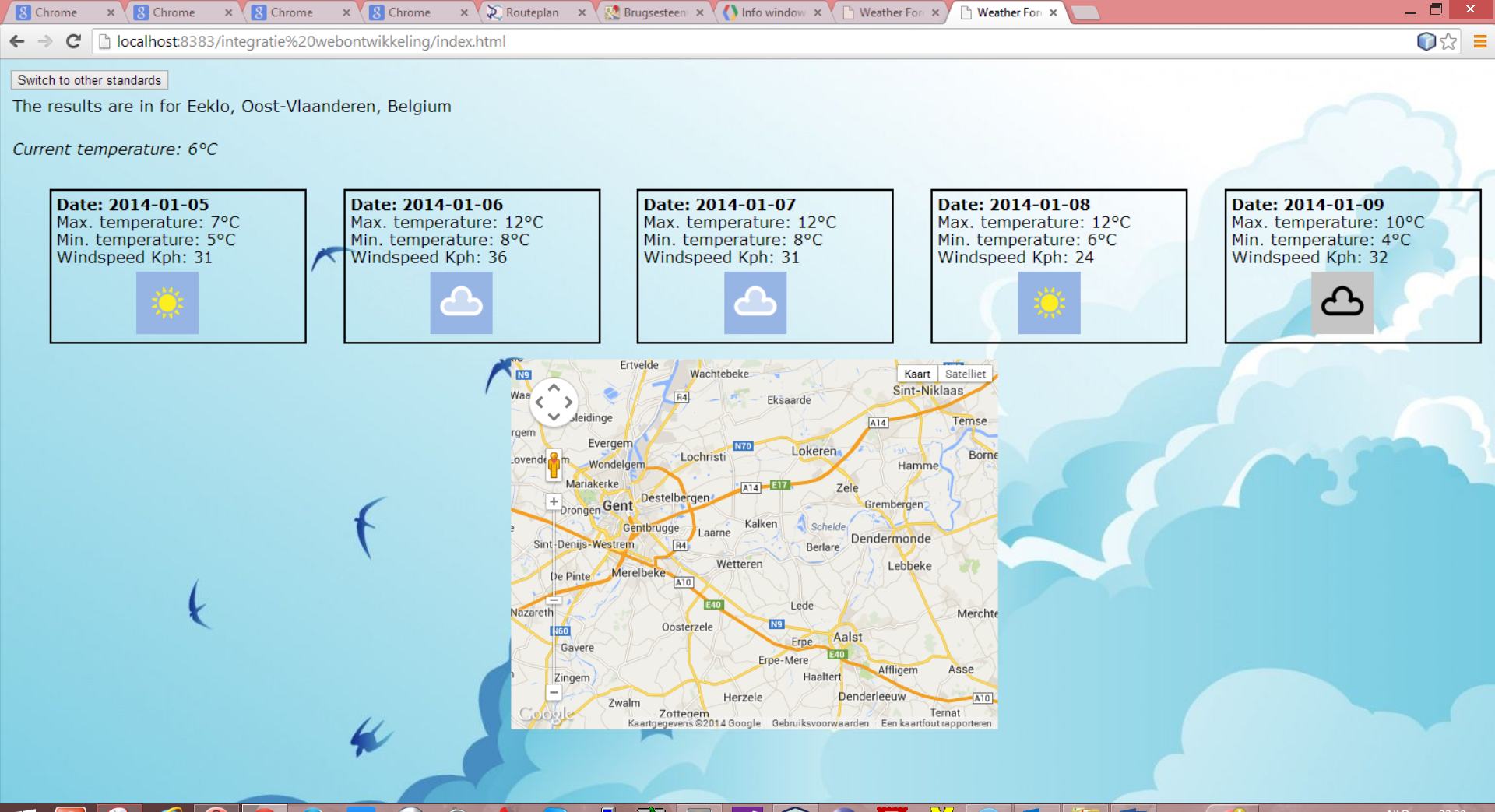
## Bij opstarten



## Onmiddellijk wordt het weer opgehaald



## Na ophalen weersvoorspelling



# Bronnen

GEOLOCATION – asking for a user’s permition for using his coordinates

<http://www.w3schools.com/html/html5_geolocation.asp>

<http://www.w3schools.com/html/tryit.asp?filename=tryhtml5_geolocation>

CONVERTING LATITUDE & LONGITUDE TO ADRESS:

<http://stackoverflow.com/questions/951839/api-to-get-weather-based-on-longitude-and-latitude-coordinates>

REVERSE GEOCODING:

<https://developers.google.com/maps/documentation/javascript/geocoding#ReverseGeocoding>

Developping a jQuery plugin:

Learning jQuery – Fourth Edition (Jonathan Chaffer, Karl Swedberg) => <http://www.packtpub.com/learning-jquery-with-simple-javascript-techniques-fourth-edition/book>