Scriptie ingediend tot het behalen van de graad van

PROFESSIONELE BACHELOR IN DE ELEKTRONICA-ICT

[Visualize your data]

A data driven web application

Dries Crauwels

Departement Wetenschappen en Techniek

Opleiding Elektronica-ICT

Academiejaar 2013-2014

Interne promotor: [Erik Vanherck]
Externe promotor: [Tom Peeters]

Versie: 10 juni 2014

ARTESIS PLANTIJN
HOGESCHOOL ANTWERPEN

Dankwoord

Graag zou ik eerst en vooral mijn medestudent Sam De Jongh willen bedanken. Samen hebben wij 4 maanden gewerkt bij Inventive Designers. Dankzij hem heb ik nooit de moed verloren en was het aangenaam werken. Onze interne promoter Erik Vanherck heeft ons beide goed begeleid en interessante opdrachten gegeven. Ten slotte bedank ik ook nog graag Tom Peeters en Tim Dams voor hun feedback en snelle response op vragen en opmerkingen over onze bachelorproef.

Antwerpen, 10 juni 2014
Dries Crauwels

Abstract

Big data is vandaag in de mode. De vraag ernaar stijgt en meer en meer bedrijven stellen hun data publiekelijk ter beschikking. Big data draait om grote hoeveelheden, de snelheid waarmee de data binnenkomt en opgevraagd kan worden, en de diversiteit van de data. Denk maar aan landkaarten, data afkomstig van CERN, internetverkeer enzovoorts. De technologie hiervoor bestaat al enige tijd maar de mindset is nu helemaal anders. Alle data op een hoop gooien en iedereen kan er gebruik van maken, dat is nieuw. Echter niemand heeft iets aan een berg bits en bytes. We moeten iets met deze data doen.

We gaan populaire datasets op het web(JSON,CSV,XML) visualiseren in een web applicatie. De gebruiker zal zijn eigen data kunnen uploaden en de website zal deze voor hem visualiseren. Daarna zal je de visualisatie kunnen exporteren om op te nemen in jouw publicatie. Dit kan zeer nuttig zijn voor onder andere tijdschriften en kranten die graag info beter naar de lezer willen overbrengen.

Visualiseren van data kan heel ver gaan. Van infographics over de ganse wereld tot de volledige opbouw van een genoom. Deze applicatie zal dan ook nooit volledig klaar zijn en men zal er blijven aan moeten werken om nieuwe soorten data op verschillende manieren te visualiseren. Deze website is een eerste stap in de goede richting. Ik streef er dan ook naar om onderhoudbare en universele code te schrijven zodat andere mensen er makkelijk verder mee kunnen en functionaliteit kunnen toevoegen.

De volledige code zal bestaan uit jQuery, Javascript, HTML 5 en CSS 3 en de D3 javascript library. Deze D3 library is de hart van de applicatie. Ze is geschreven specifiek om makkelijker SVG elementen in een HTML pagina te genereren. Deze SVG elementen vormen dan een grafische voorstelling van de data. De code zal opgebouwd zijn uit één library die alle code bevat voor het tekenen van de data in HTML.De rest van de code zal de look and feel van de applicatie zijn en de gebruikers interface.

Totzover heeft de applicatie 4 werkende wizards voor 4 verschillende charts en één grotere wizard voor een infographic waarin de 4 kleinere wizards in zijn verwerkt.

Inhoudsopgave

Dankwoord **i**

Abstract **ii**

1 Inleiding **1**
1.1 Programmeertalen . . . 1
1.1.1 HTML 5 . . . 1
1.1.2 CSS 3 . . . 2
1.1.3 JavaScript . . . 2
1.1.4 jQuery . . . 3
1.1.5 D3 . . . 3

2 Uitvoering **4**
2.1 Opzetten van een lokale webserver . . . 4
2.2 Algemene mappenstructuur . . . 6
2.3 Algemene code-architectuur en opmaak . . . 7
2.4 Web IDE . . . 9
2.5 Git en Versiecontrole . . . 10
2.6 SVG en D3 . . . 11
2.7 Debugging . . . 13
2.8 Hulpbronnen . . . 15
2.9 Homepage . . . 15
2.10 Basic Chart Wizards . . . 16
2.10.1 Area Chart . . . 16
2.10.2 Bar Chart . . . 40
2.10.3 Donut Chart . . . 41
2.10.4 Line Chart . . . 42
2.11 Infographic Wizard . . . 43

3 Resultaten **47**

3.1 Toekomst 47

4 Besluit **48**

Lijst van figuren

2.1 Screenshot Eclipse server settings 5
2.2 Screenshot Tomcat Homepage 6
2.3 Screenshot SourceTree 11
2.4 Screenshot Chrome Debugging Interface 14
2.5 Screenshot Chrome Debugging Interface: Sources 14
2.6 Screenshot Area Chart Wizard 17
2.7 Screenshot Bar Chart Wizard 41
2.8 Screenshot Donut Chart Wizard 42
2.9 Screenshot Line Chart Wizard 43
2.10 Screenshot Infographic Wizard 44

Hoofdstuk 1

Inleiding

De opdracht bestaat er in om een interfaces te coderen en te designen op een website waarnaar de gebruiker zijn eigen data kan uploaden om er een visuele voorstelling van te krijgen.

Hiervoor moet je al goed weten in welke soort formaten de meeste data voorkomen en moet de applicatie de gebruiker zelf opleggen om een bepaalde syntax te gebruiken in zijn data. Men kan de code aanpassen om meerdere syntaxen uit te lezen maar dit vraagt meer werk dan dat de gebruiker wat eigenschappen aanpast in zijn bestand.

Op het web wordt vooral JSON CSV en XML data gebruikt als datasets. De gebruiker moet één van deze bestandssoorten kunnen uploaden naar de server en de applicatie leest deze dan uit en tekent SVG elementen. Deze SVG elementen vormen dan een chart die men kan exporteren als een SVG, PNG bestand of HTML code.

1.1 Programmeertalen

Aangezien we met de D3 library een web applicatie moeten ontwikkelen en dit een javascript library is, is het vanzelfsprekend voor een programmeur om de volgende programmeertalen te gebruiken als ondersteuning voor deze opdracht.

1.1.1 HTML 5

> "HTML or HyperText Markup Language is the standard markup language used to create web pages.
>
> HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>). HTML tags most commonly come in pairs like <h1>and

</h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags).

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page. HTML describes the structure of a website semantically along with cues for presentation, making it a markup language rather than a programming language."

Source: HTML - `https://en.wikipedia.org`

1.1.2 CSS 3

"Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language. While most often used to style web pages and interfaces written in HTML and XHTML, the language can be applied to any kind of XML document, including plain XML, SVG and XUL. CSS is a cornerstone specification of the web and almost all web pages use CSS style sheets to describe their presentation.

CSS is designed primarily to enable the separation of document content from document presentation, including elements such as the layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple pages to share formatting, and reduce complexity and repetition in the structural content (such as by allowing for tableless web design)."

Source: Cascading Style Sheets - `https://en.wikipedia.org`

1.1.3 JavaScript

"JavaScript is a prototype-based scripting language with dynamic typing and has first-class functions. Its syntax was influenced by C. JavaScript copies many names and naming conventions from Java, but the two languages are otherwise unrelated and have very different semantics. The key design principles within JavaScript are taken from the Self and Scheme programming languages. It is a multi-paradigm language, supporting object oriented, imperative, and functional programming styles."

Source: JavaScript - `https://en.wikipedia.org`

1.1.4 jQuery

> "jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript."

Source: What is jQuery? - `http://jquery.com`

1.1.5 D3

> "D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation."

Source: D3 - `http://d3js.org`

Hoofdstuk 2

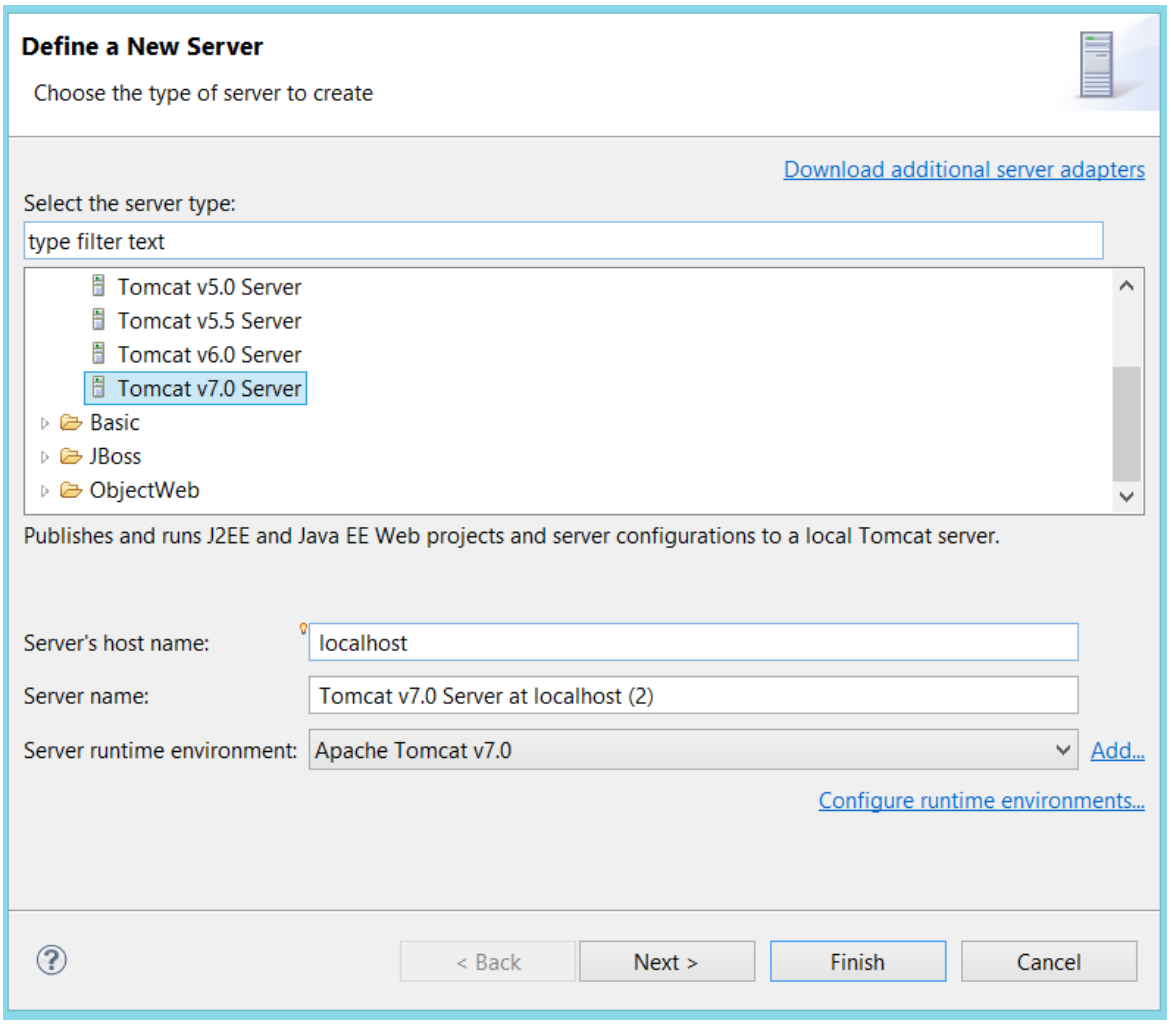
Uitvoering

2.1 Opzetten van een lokale webserver

Op aanraden van één van de collega's zetten we een Tomcat Apache server op via Eclipse om onze website op te draaien. Via de volgende links kan je deze software downloaden:

- Apache Tomcat v7.0: `http://tomcat.apache.org/download-70.cgi`
- Eclipse IDE for Java EE Developer: `https://www.eclipse.org/downloads/`

Start Eclipse op en maak een nieuwe workspace aan. Klik met je rechtermuisknop onderaan in de Servers tab. Klik dan op New >Server en selecteer de volgende instellingen:

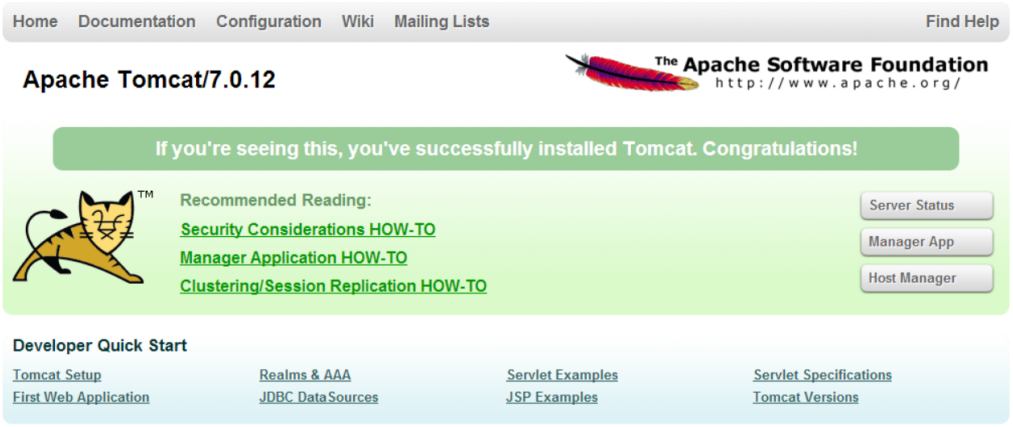


Figuur 2.1: Screenshot Eclipse server settings

Klik Next en klik Finish. De server zal wanneer hij gestart is de files uploaden naar localhost:8080. Deze files zitten in de Tomcat folder die je eerder hebt geinstalleerd met het volgende pad:

- C:\Users\Crauwels Dries\Documents\Tomcat\webapps\ROOT

Hierin komt dan ook onze folder die onze website zal bevatten. Om de server te starten selecteren we ze in Eclipse in de servers tab en drukken we op start. Als alles werkt kunnen we naar http://localhost:8080/ surfen en zien we de homepage van Tomcat:



Figuur 2.2: Screenshot Tomcat Homepage

De link naar onze eigen files wordt dan: http://localhost:8080/MyFolder/index.html
Nu kunnen we van start gaan met de structuur van de applicatie en het schrijven van onze html paginas.

2.2 Algemene mappenstructuur

De code zal opgesplitst zijn in 3 typen bestanden: .HTML bestanden, .CSS bestanden en .JS bestanden. De HTML bestanden zullen de basis HTML bevatten voor onze website. Het is het skelet van de structuur en bevat de inhoud van de pagina. Het stijlen van de pagina gebeurt in CSS en zoveel mogelijk code zal in aparte .JS bestanden staan. Zoals onze eigen library met alle D3 code en externe plugins.

Wanneer we refereren naar één van deze bestanden vanuit een HTML pagina dan moeten we het pad opgeven dat vertrekt vanuit de plaats waar de HTML pagina zelf bevindt. Met andere woorden als index.html in onze MyFolder zit en daarin zit een JS map met library.js, dan moet de index pagina het /JS/library.js pad gebruiken om hieraan te kunnen. Het is dan ook voor de hand liggend dat we een mappenstructuur handhaven die er als volgt uitziet:

Alle HTML paginas zitten in de website folder. CSS in de CSS folder en de Javascript bestanden in JS. Voorts is er nog een map voor figuren, data en fonts.

- D3-Chart-Wizard
  - home.html
  - barchart.html
  - linechart.html
  - donutchart.html
  - areachart.html

- infographic.html
- js
  - jquery-ui.js
  - chartwizard.js
  - ...
- css
  - jquery-ui.css
  - charts.css
  - menu.css
  - infographic.css
  - ...
- data
- fonts
- images

2.3 Algemene code-architectuur en opmaak

Goede code schrijven is belangrijk. Zoals Mr. Dams altijd zegt: schrijf geen dubbele code, wees consistent, begrijp wat je schrijft en hoe minder code nodig om het te doen laten werken hoe beter. Elke soort code zal een andere opmaak en structuur hebben naargelang de mogelijkheden van de taal.

HTML

HTML is geen programmeertaal maar een opmaaktaal. Dit wil echter niet zeggen dat we geen structuur moeten handhaven en niet aan een aantal afspraken moeten houden. In de HTML paginas is het belangrijkste dat de ID attributen uniek zijn en dat ze een logische naam hebben. Een logische naam moet kort zijn maar uniek en verwijzen naar welk element het inzit. Stel dat het een ID is van een div waarin meerdere divs zitten dan kunnen we dit een wrapper of container noemen. Bijvoorbeeld is de div waarin de SVG elementen zitten. Verder zullen de class names in alle tags gebruikt worden voor de opmaak van het element. Class attributen mogen meerdere keren voorkomen zodanig dat elk element met dezelfde class name dezelfde stijling krijgt en kan een javascript functie alle elementen aanroepen met deze class name.

De opmaak van de code kan je makkelijk doen door je code op een online source-formatter te kopieren. Zo krijgen alle lijnen de juiste inspringing en krijg je veel beter overzicht van je code indien je slim genoeg bent om hier geen tijd in te steken of het gewoon niet goed kan. Dreamweaver kan dit ook maar minder goed.

`http://jsbeautifier.org/`

CSS

De CSS bestanden hebben geen specifieke opmaak aangezien CSS niets speciaal is wat code betreft.

JavaScript

Het meeste van de JavaScript code zal in onze eigen library staan. Deze library is een namespace waarin we classes en functions kunnen steken die we kunnen aanroepen vanuit de HTML pagina als de library geinclude wordt. variabele charts is de namespace en elke functie die volgt na een "."zit in deze namespace. Als we dit .js bestand includen in onze HTML pagina kunnen we deze functies aanroepen met "charts.class1.subclass1.functie()"bijvoorbeeld.

De opmaak ziet er zo uit:

```
1  // JavaScript Library Document
2  // Library.js
3
4  var charts = charts || {};
5
6  (function () {
7          //General variables
8          var margin = {
9                  top: 0,
10                 right: 0,
11                 bottom: 76,
12                 left: 76
13         };
14
15
16         var svgWidth = 1080 - margin.left - margin.right,
17                 svgHeight = 400 - margin.top - margin.bottom;
18
19         var HDsvgWidth = 1920,
20                 HDsvgHeight = 1080;
21
22         charts.infographic = function (type) {
23                  var graphicType;
24
25                  graphicType = type;
26                  switch(graphicType){
27                      case "DNA-graphic":
28
29                      charts.infographic.DNAgraphic = function (divclass) {
```

Elke HTML pagina zal zijn eigen code hebben maar er kunnen meerdere paginas zijn die dezelfde CSS gebruiken en libraries. Zo is er één CSS die de twee navigatie menus bovenaan stijlt, een CSS voor de infographic wizard en één voor alle chart wizards.

Alle JavaScript code die we in de HTML pagina gebruiken verdelen we onder in 4 categorieën: initialisations, jQuery UI, D3 en JavaScript. Al deze code zetten we onderaan in de <body>om de pagina sneller te doen laden.

In initialisations zetten we alles wat in de document.ready functie moet.

```
////////////////////////////////////////////////////////////////
//////////////////////INITIALISATIONS///////////////////////////
////////////////////////////////////////////////////////////////
```

jQuery heeft voor het aanroepen van al zijn UI elementen methodes nodig. Deze komen onder de jQuery UI comment.

```
////////////////////////////////////////////////////////////////
//////////////////////  JQUERY UI   ////////////////////////////
////////////////////////////////////////////////////////////////
```

Onder D3 komt alle nodige D3 code die te klein is om in de library te steken.

```
////////////////////////////////////////////////////////////////
//////////////////////       D3     ////////////////////////////
////////////////////////////////////////////////////////////////
```

Al de JavaScript die niet in de vorige categorieen thuishoort zetten we hier.

```
////////////////////////////////////////////////////////////////
//////////////////////    JAVASCRIPT   ////////////////////////
////////////////////////////////////////////////////////////////
```

Indien er globale variabelen moeten zijn komen die bovenaan onder elke categorie te staan.

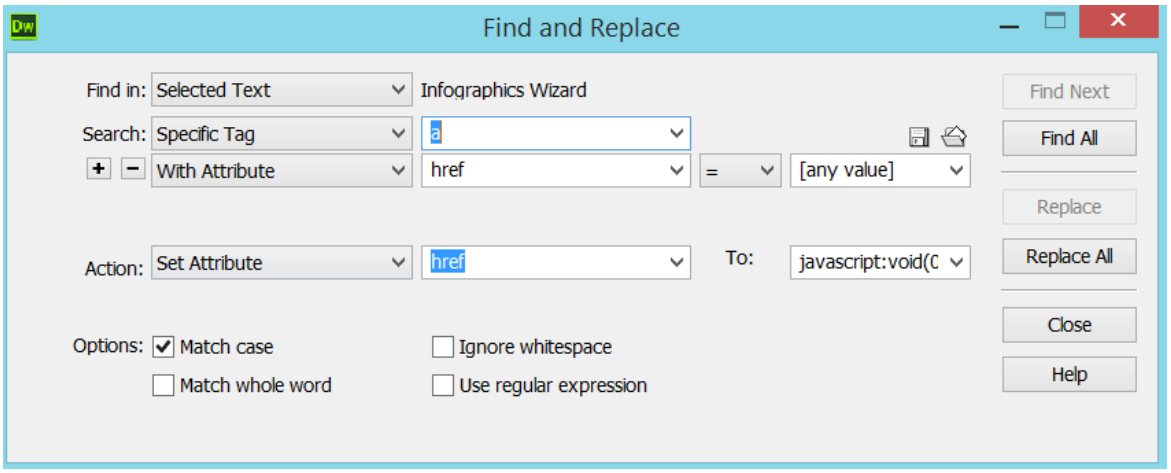
2.4 Web IDE

Als je iets gaat ontwikkelen voor het web dat zowel CSS, HTML als JavaScript bevat, bestaat er geen standaard ontwikkelomgeving. Deze applicatie is geschreven met behulp van Dreamweaver CS6[1] van Adobe dat nog een degelijke ondersteuning aanbiedt. De nieuwste Visual Studio[2] blijkt ook goed mee te vallen maar na wat onderzoek te hebben gedaan achteraf is gebleken dat WebStorm[3] van JetBrains de beste IDE is voor web development. JetBrains is trouwens dezelfde producent van ReSharper voor Visual Studio. Helaas is geen enkel van deze software gratis.

- [1]: `http://www.adobe.com/be_nl/products/dreamweaver/features.html`

- [2]: http://www.visualstudio.com/explore/application-development-vs#Scenario2_1
- [3]: http://www.jetbrains.com/webstorm

Wat wel zeer handig is in Dreamweaver is de Find and Replace feature. Deze is simpel maar krachtig. De andere web IDE's zullen dit zeker ook wel hebben. Hiermee kan je verschillende tags selecteren en bijvoorbeeld bepaalde attributen met een bepaalde waarde veranderen. Zo hoef je ze niet één voor één alles aan te passen.



2.5 Git en Versiecontrole

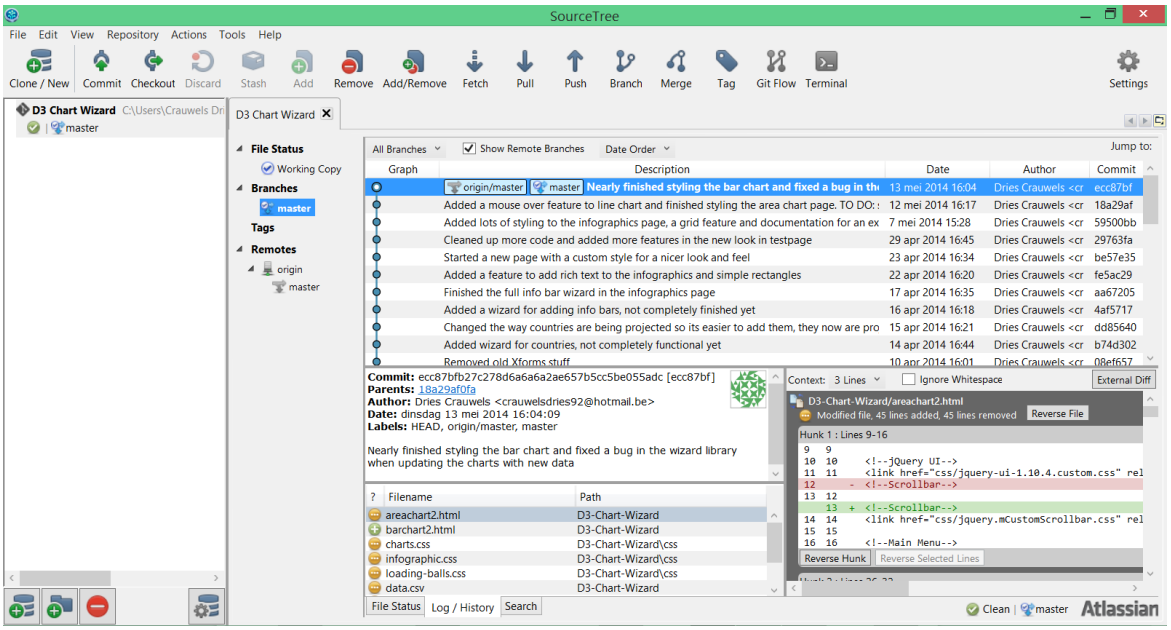
Bij Inventive Designers maken ze gebruik van Stash van Atlassian. Een manier om beter gebruik te maken van Git met servers, taakverdeling en meerdere gebruikers.

Git repository management your way On-premises source code management for Git that's secure, fast, and enterprise grade. Create and manage repositories, set up fine-grained permissions, and collaborate on code - all with the flexibility of your servers.

Source: Stash - https://www.atlassian.com

Git GUI

Om makkelijker overweg te kunnen met Git zonder commandline maken we gebruik van SourceTree. Dit is een grafishe interface waarmee we eenvoudig de repository kunnen binnenhalen van Stash en er een lokale map van maken. Hierin steken we dan de map met onze volledige website in. We voeren dan een commit en push uit als we deze willen uploaden naar Stash na een grondige verandering. Bij elke commit voegen we commentaar bij met wat er precies veranderd en toegevoegd is.



Figuur 2.3: Screenshot SourceTree

2.6 SVG en D3

Scalable Vector Graphics(SVG) zijn HTML elementen om twee-dimensionale figuren weer te geven op het web. Deze elementen worden geschreven in een XML format net zoals ander gewone HTML elementen. Omdat deze elementen vectorieel zijn, is er geen kwaliteitsverlies bij het inzoomen of vergroten en verkleinen. Er is mogelijkheid tot animatie met SVG elementen en je kan ze stijlen met CSS.

Voorbeeld van een afgeronde rechthoek met SVG:

```
1 <svg width="400" height="180">
2   <rect x="50" y="20" rx="20" ry="20" width="150" height="150"
3   style="fill:#CCC;stroke:black;stroke-width:5;" />
4 </svg>
```

Fiddle: `http://jsfiddle.net/4FSgJ/`

D3(Data Driven Documents) bindt arbitraire data aan een Document Object Model(DOM) en transformeert deze naar het document op een Data-Driven manier. Met andere woorden je kan eender welk HTML element invoegen waarvan de eigenschappen afkomstig zijn uit een dataset zoals een array van objecten. D3 is een JavaScript library en is volledig mee geinteregeerd met de HTML 5 CSS 3 en SVG standaarden.

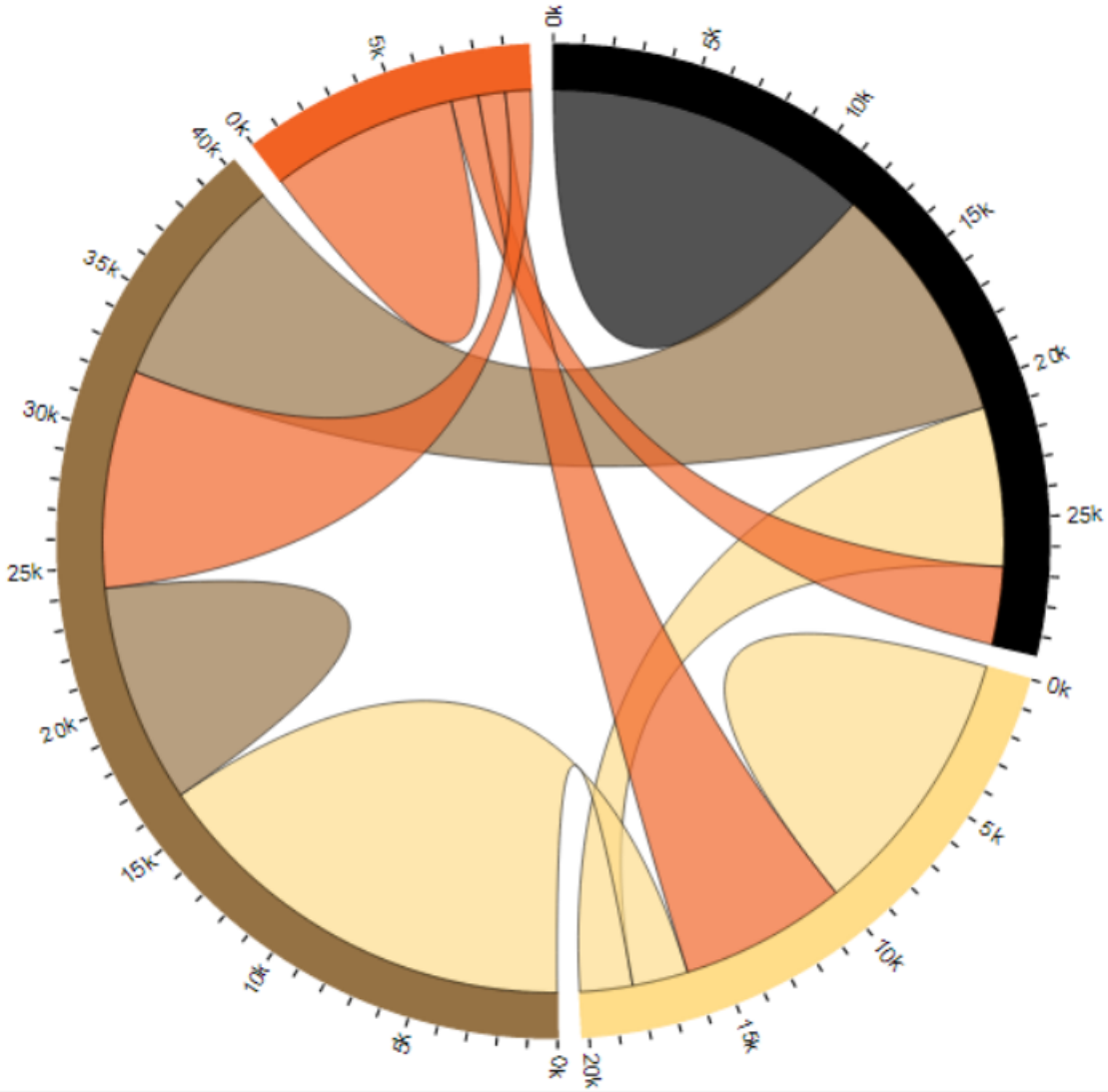
Voorbeeld van een afgeronde rechthoek met D3:

```
1  <div class="container"></div>
2  var svg = d3.select(".container")
3              .append("svg")
4              .attr("width",400)
5              .attr("height",180);
6
7  svg.append("rect")
8          .attr("x",50)
9          .attr("y",20)
10         .attr("rx",20)
11         .attr("ry",20)
12         .attr("width",150)
13         .attr("height",150)
14         .style({"fill":"#CCC",
15                 "stroke":"black",
16                 "stroke-width":5});
```

Fiddle: `http://jsfiddle.net/aZT7c/`

Zoals je kan zien krijgen we twee identieke figuren. Echter de code om deze te genereren met D3 is veel te veel voor wat je krijgt. D3 wordt pas interessant wanneer we dit vierkant gaan visualiseren aan de hand van data en de gebruiker de mogelijkheid geven om dit allemaal zelf te doen. Het eerste vierkant is dan ook "hard-coded"zoals dat heet en is niet dymanisch. Voor een wizard moet alles dynamisch opgebouwd zijn en mag niets vast staan. Elk element en attribuut moet aangepast, toevoegd, of verwijderd kunnen worden.

Een complexer voorbeeld van wat D3 kan:



Voorbeelden met animaties

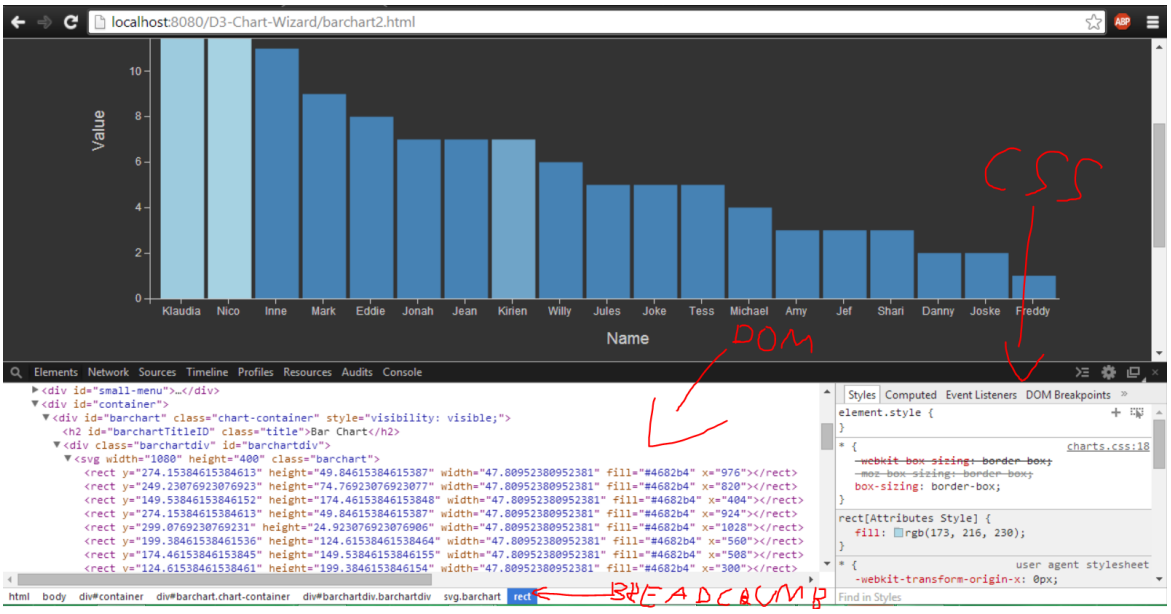
`http://bl.ocks.org/mbostock/10685278`

`http://bl.ocks.org/mbostock/10571478`

2.7 Debugging

Er is niets zo simpel als debuggen wanneer je code schrijft voor het web. Je moet niet wachten om je code te compilen, je slaat je HTML pagina gewoon op en refresht de pagina. Het is de browser zelf die de code zal compilen. Als er dan errors zijn kan je die in de debugger van je browser zien. Deze krijg je te zien als je ergens op de pagina met je rechtermuisknop klikt

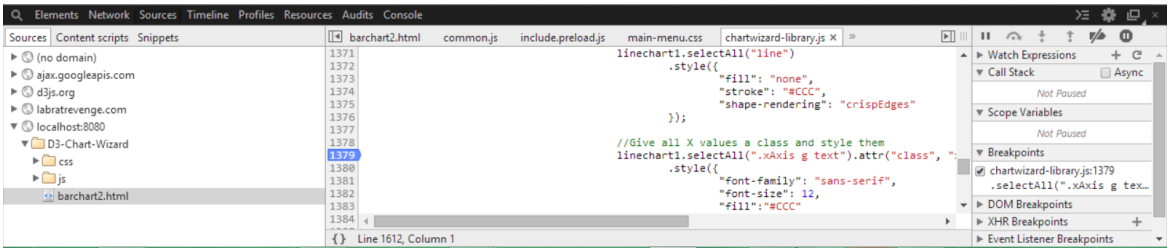
en "Element inspecteren"selecteert. De meeste browsers hebben een soortgelijke debugging interface. In deze opdracht is altijd Google Chrome gebruikt die er als volgt uitziet:



Figuur 2.4: Screenshot Chrome Debugging Interface

Zoals is aangeduid op de screenshot hebben we links onderaan in onze browser de DOM inspector die alle elementen en hun attributen weergeeft. Als we met onze muis over een element gaan wordt deze geselecteerd in de browser en weten we precies welk element het is. Daaronder zien we ook in welke andere tags het element genest zit. Dit heet een breadcrumb-trail. Ten slotte zien we ook de CSS rechts van het geselecteerde element. We zien welke CSS eigenschappen dit element allemaal krijgt en van waar ze komen.

Indien we met externe JavaScript bestanden werken kunnen we deze zien in de "Sources"tab in de debugger. Hierin kunnen we ook break-points zetten waar de browser zal pauzeren bij het refreshen van de pagina als het break-point eenmalig is of wanneer we op deze plaats in de code zitten bij het uitvoeren van een handeling. Net zoals eender welke andere debugger.



Figuur 2.5: Screenshot Chrome Debugging Interface: Sources

We kunnen zelfs live onze code testen door de HTML van de pagina aan te passen in de debugger. Je moet deze niet refreshen, dat is de kracht van HTML. Geen event handlers of iets dergelijks. Je verandert gewoon een attribuut of voegt iets toe en deze verandering zal direct

te zien zijn op de pagina.

2.8 Hulpbronnen

W3 Schools `http://www.w3schools.com/default.asp`

Stackoverflow `http://stackoverflow.com`

D3 Wiki, voorbeelden en API `https://github.com/mbostock/d3/wiki`

Ebook Interactive Data Visualization for the Web `http://chimera.labs.oreilly.com/books/1230000000345/pr01.html`

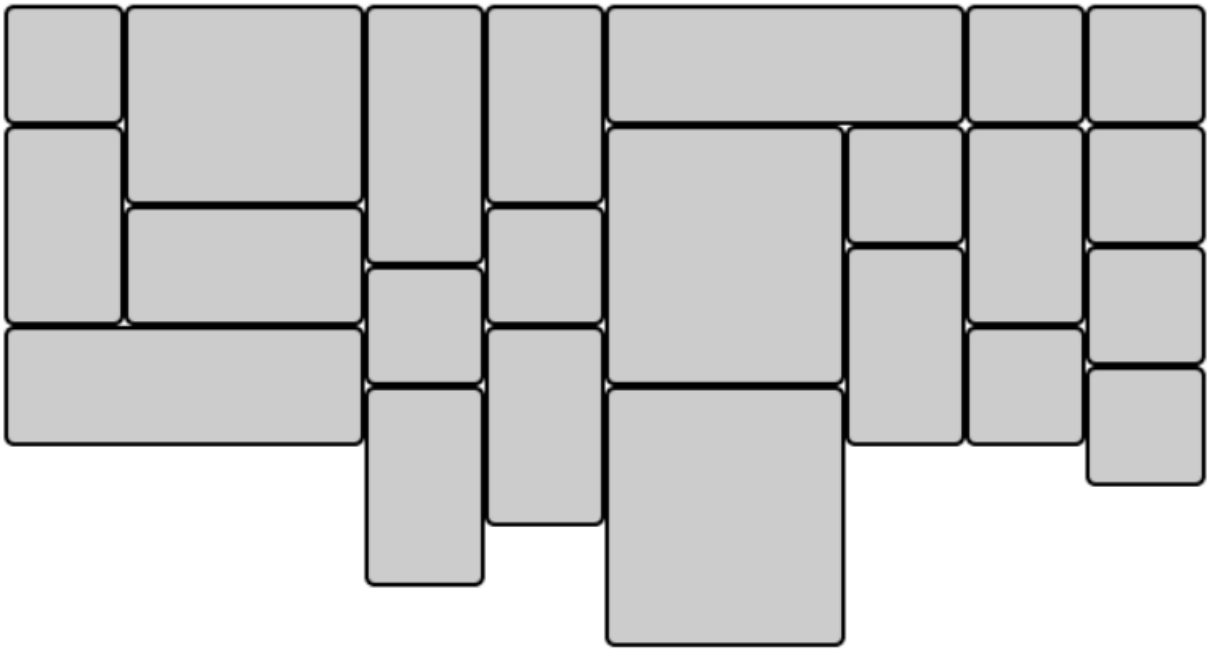
2.9 Homepage

Op de homepage van de website zal je meteen alle wizards te zien krijgen. Zowel de vier basic chart wizards als die voor de infographics. Om de pagina op te vullen en om later uit te breiden zijn er ook links toegevoegd voor andere basic charts. Maar deze links staan uit omdat die wizards nooit zijn klaar geraakt.

Bovenaan staat het navigatie-menu dat volledig is opgemaakt uit eigen CSS. Dit menu wordt in de andere paginas ook weergegeven. Het goede aan eigen CSS en minder libraries te gebruiken is dat er dan minder HTTP "gets"moeten worden uitgevoerd en dit doet de pagina veel sneller laden. De CSS voor het navigatie-menu staat in een aparte CSS file genaamd: "main-menu.css".

De blokken waarin elke link naar de wizard staan zijn dynamisch opgebouwd met behulp van de Masonry.js library. Deze library zorgt ervoor dat elementen worden opgestapeld zoals een bakstenen muur. Het plaatst ze op een zo goed mogelijke manier zodanig dat er zo min mogelijk plaats wordt overgelaten. Masonry kan nog veel meer dan dit maar die functionaliteiten hebben we niet nodig.

Voorbeeld van hoe Masonry elementen kan stapelen:



2.10 Basic Chart Wizards

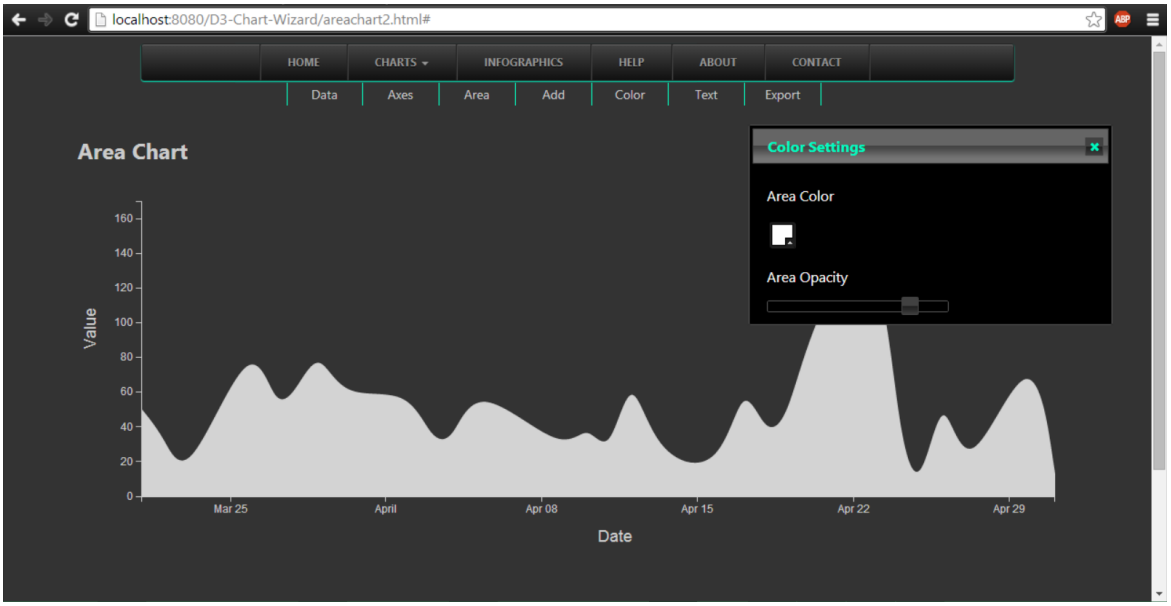
Elke basic chart wizard is opgemaakt met dezelfde structuur. Zowel de interface als de code. Bovenaan staat het algemene navigatie menu. Net daaronder staan de knoppen met alle opties om de chart te bewerken. Elke knop opent een extra dialoogvenster waarin de nodige functies staan. Je kan deze slepen over het scherm zodat je de verandering op de chart direct kan zien. Zoals eerder vermeld zal de belangrijkste code in onze eigen javascript library staan. Dit is de D3 code om de chart te genereren.

De code voor elke chart wizard is zeer analoog. Dit is zowel toevallig als met opzet. Toevallig omdat de charts sterk op elkaar lijken wiskundig gezien(voor elke X waarde steeds maximum één Y waarde) en omdat goede code hergebruikt kan worden. Daarom zullen we ook maar één chart wizard volledig uitleggen de andere drie zullen in het kort worden toegelicht. Voor de volledige code verwijs ik u naar de bestanden zelf.

2.10.1 Area Chart

Een area chart geeft data weer in de vorm van een opgevuld gebied met X en Y coordinaten. Wiskundig is hier niets speciaal aan. Maar we kunnen dit gebied wel manipuleren. D3 heeft ingebouwde functies om dit te doen. Er zijn verschillende interpolatie mogelijkheden. Om bijvoorbeeld de grafiek af te vlakken kunnen we de Basis-Interpolatie toepassen. De wiskunde

hierachter en de kennis om dit in JavaScript te doen is van een hoog niveau. Maar dat is één van de sterke punten van D3. Wij hoeven dit niet zelf te doen. Let wel op dat deze functie niet de data afkomstig van de .CSV file bewerkt maar alleen de lijn die het gebied tekent. Verder kan je nog de kleur en doorzichtigheid aanpassen van de chart. Een moving average lijn toevoegen, je eigen data inlezen en de chart exporteren naar HTML, SVG of een PNG figuur.



Figuur 2.6: Screenshot Area Chart Wizard

Code areachart.html

Omdat veel code weergeven moeilijk gaat in een document verwijs ik u naar de broncode indien ze niet leesbaar is.

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5          <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
6          <title>Area Chart</title>
7
8          <!--CSS for every chart wizard-->
9          <link href="css/charts.css" rel="stylesheet" />
10
11         <!--jQuery UI-->
12         <link href="css/jquery-ui-1.10.4.custom.css" rel="stylesheet" />
13
14         <!--Scrollbar-->
15         <link href="css/jquery.mCustomScrollbar.css" rel="stylesheet" type="text/css
16
17         <!--Main Menu-->
18         <link href="css/main-menu.css" rel="stylesheet" type="text/css" />
19
```

```
20          <!--Loading animation-->
21          <link href="css/loading-balls.css" rel="stylesheet" type="text/css" />
22
23          <!--Color picker-->
24          <link rel="stylesheet" media="screen" type="text/css" href="css/colorpicker.
25
26          <!--Icons-->
27          <link href="css/glyphicons.css" rel="stylesheet" />
28  </head>
```

In de head van de HTML pagina zetten we alle referenties naar de CSS paginas die we gebruiken voor deze pagina. De titel van de pagina is de titel die wordt weergegeven in je browser bovenaan in het tabblad.

```
1
2  <body>
3           <nav id="menu-wrap">
4                   <div id="menu-trigger">Menu</div>
5                   <ul id="menu">
6                           <li id="disabled"> </li>
7                           <li><a href="home2.html">Home</a>
8                           </li>
9                           <li> <a href="#">Charts <b class="caret"></b></a>
10                                  <ul>
11                                          <li><a href="areachart2.html">Area Chart</a>
12                                          </li>
13                                          <li><a href="barchart2.html">Bar Chart</a>
14                                          </li>
15                                          <li><a href="donutchart2.html">Donut Chart</
16                                          </li>
17                                          <li><a href="#">Bivariate Area Chart</a>
18                                          </li>
19                                          <li><a href="#">Grouped Bar Chart</a>
20                                          </li>
21                                          <li><a href="linechart2.html">Line Chart</a>
22                                          </li>
23                                          <li><a href="#">Multi-Line Chart</a>
24                                          </li>
25                                          <li><a href="piechart2.html">Pie Chart</a>
26                                          </li>
27                                          <li><a href="#">Scatterplot</a>
28                                          </li>
29                                  </ul>
30                          </li>
31                          <li><a href="infographic.html">Infographics</a>
32                          </li>
33                          <li><a href="jezuscampraptorravepage.html">Help</a>
34                          </li>
35                          <li><a href="#about">About</a>
36                          </li>
37                          <li><a href="#contact">Contact</a>
```

```
38                          </li>
39          </nav>
40          <div id="small-menu">
41                  <ul>
42                          <li><a href="#" onClick="showDataDialog()">Data</a>
43                          </li>
44                          <li><a href="#" onClick="showAxesDialog()">Axes</a>
45                          </li>
46                          <li><a href="#" onClick="showAreaDialog()">Area</a>
47                          </li>
48                          <li><a href="#" onClick="showMovingAverageDialog()">Add</a>
49                          </li>
50                          <li><a href="#" onClick="showColorsDialog()">Color</a>
51                          </li>
52                          <li><a href="#" onClick="showTextDialog()">Text</a>
53                          </li>
54                          <li><a href="#" onClick="showExportDialog()">Export</a>
55                          </li>
56                  </ul>
57          </div>
```

Het eerste deel van de body bevat de semantische opmaak van de twee menus. Het algemene navigatie menu en alle knoppen voor de wizard. In het algemene menu wordt nog een extra item toegevoegd in het begin en op het einde dat geen link bevat maar alleen dient als opmaak. De knoppen voor de wizard hebben allemaal een onClick event dat gekoppeld is aan een jQuery functie die de dialoog vensters zal openen.

```
1           <div id="container">
2                   <div id="areachart-container" class="chart-container">
3                           <h2 id="areachartTitleID" class="title">Area Chart</h2>
4                           <div class="areachartdiv" id="areachartdiv"></div>
5                           <!--A canvas to copy the svg to when we're exporting it-->
6                           <canvas width="960" height="500" style="display:none"></canv
7                   </div>
8           </div>
```

In deze container zit het div element waar de chartwizard-library een SVG element en de rest van de chart in zal tekenen. Er is ook een head element voor de titel van de chart die aangepast kan worden. En een canvas dat niet weergegeven wordt dat gebruikt wordt voor de SVG te exporteren naar een png figuur.

```
1           <!--Dialogs-->
2           <div id="data-dialog" title="Data Settings" class="dialog">
3                   <div class="btn-group">
4                           <div class="fileinput fileinput-new" data-provides="fileinpu
5                                   <div class="input-group">
6                                           <input id="input-file" type="file" name="...
7                                           <a id="csv-upload" href="#" class="btnz" dat
8                                   </div>
9                                   <button type="button" class="btnz btn-default" onCli
10                          </div>
```

```
11                  </div>
12          </div>
13          <div id="axes-dialog" title="Axes Settings" class="dialog">
14                  <div class="row">
15                          <div class="col-lg-6">
16                                  <p>X Axis Name</p>
17                                  <input id="xaxis-name" type="text" class="form-input
18                          </div>
19                          <div class="col-lg-6">
20                                  <p>Y Axis Name</p>
21                                  <input id="yaxis-name" type="text" class="form-input
22                          </div>
23                  </div>
24          </div>
25          <div id="area-dialog" title="Area Settings" class="dialog">
26                  <h3>Interpolation</h3>
27                  Sets the interpolation mode to the specified function. The following
28                  <div class="btn-group">
29                          <label class="btnz btn-primary">
30                                  <input type="radio" name="options" id="linear-option
31                          <label class="btnz btn-primary">
32                                  <input type="radio" name="options" id="step-option-b
33                          <label class="btnz btn-primary">
34                                  <input type="radio" name="options" id="basis-option-
35                          <label class="btnz btn-primary">
36                                  <input type="radio" name="options" id="bundle-option
37                          <label class="btnz btn-primary">
38                                  <input type="radio" name="options" id="cardinal-opti
39                          <label class="btnz btn-primary">
40                                  <input type="radio" name="options" id="monotone-opti
41                  </div>
42                  <div id="area-options-description">
43                          <h3>Description</h3>
44                          No interpolation mode selected.</div>
45          </div>
46          <div id="moving-average-dialog" title="Add a Moving Average Line" class="dia
47                  <p>Show Moving Average Line</p>
48                  <input type="checkbox" onClick="charts.chart.Areachart.Generate.Movi
49          </div>
50          <div id="colors-dialog" title="Color Settings" class="dialog">
51                  <p>Area Color</p>
52                  <span style="height:64px; width:64px;">
53    <div id="area-color-picker" class="color-picker">
54      <div style="background-color: steelblue;"> </div>
55    </div>
56    </span>
57                  <p>Area Opacity</p>
58                  <div id="opacity-slider" style="width:200px;"></div>
59          </div>
60          <div id="text-dialog" title="Text Settings" class="dialog">
61                  <p>Chart Title</p>
```

```
62                  <input id="chart-title" type="text" value="Area Chart" class="form-i
63                  <p>Text & Axis Color</p>
64                  <span style="height:64px; width:64px;">
65    <div id="text-color-picker" class="color-picker">
66      <div style="background-color: #07FFC1;"> </div>
67    </div>
68    </span>
69          </div>
70          <div id="export-dialog" title="Export" class="dialog">
71                  <button type="button" class="btnz" onClick="GenerateSVG()">Export SV
72                  <button type="button" class="btnz" onClick="GenerateHTML()">Export H
73                  <button type="button" class="btnz" onClick="GenerateJPEG()">Download
74          </div>
75          </div>
76          <div class="footer-wrapper">
77                  <div id="footer">Developed by Dries Crauwels</div>
78          </div>
```

Alle dialoogvensters voor de chart te bewerken zijn jQuery UI elementen. En worden dus ook gestijld door je eigen jQuery CSS. Deze dialoog vensters komen altijd vooraan te staan en je kan ze slepen waar je wilt.

```
1           <!-- jQuery -->
2           <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.0/jquery.min.
3
4           <!--D3 Library-->
5           <script src="http://d3js.org/d3.v3.min.js"></script>
6
7           <!-- Some library for tooltips-->
8           <script src="http://labratrevenge.com/d3-tip/javascripts/d3.tip.v0.6.3.js"><
9
10          <!---Custom chart wizard library-->
11          <script src="js/chartwizard-library.js"></script>
12
13          <!--Spinner-->
14          <script src="js/spin.js"></script>
15
16          <!--JQuery UI -->
17          <script src="js/jquery-ui-1.10.4.js"></script>
18
19          <!--Color picker-->
20          <script type="text/javascript" src="js/colorpicker.js"></script>
```

Een HTML pagina wordt van boven naar beneden ingelezen. De volgorde waarin we de externe javascript libraries inladen is dus van belang. Als we onze eigen library eerst in laden voor die van D3 zelf zal deze niet werken omdat ze de D3 library nodig heeft.

```
1
2           <script>
3
4
```

```
5                   ////////////////////////////////////////////////////////////////
6                   ////////////////////////////////////////////////INITIALISATIONS//
7                   ////////////////////////////////////////////////////////////////
8
9
10                  $(document).ready(function () {
11
12
13                          //Get the CSV data, then generate the chart as callback
14                          charts.readCSV("areachartdata2.csv", function (outputdata) {
15
16                                  data = outputdata;
17
18
19
20                                  charts.chart("AreaChart");
21                                  charts.chart.Areachart("#areachartdiv");
22                                  charts.chart.Areachart.Generate(data);
23
24                          });
25                  });
```

De document.ready functie is een jQuery functie die wordt uitgevoerd wanneer het HTML document volledig is binnengehaald. Dit wil zeggen als de gebruiker dus niet meer moet wachten op het laden. Hierin kunnen we dan andere functies zetten die moeten worden uitgevoerd als dit is gebeurd. We zetten hierin de functie van de eigen chartwizard-library om de chart te tekenen met zijn initiele data.

In javascript kunnen we eender welke functie uitvoeren na dat een andere functie klaar is. Een callback heet dit. Dit hebben we hier nodig omdat als we niet eerst wachten tot de CSV file is ingelezen en de data in een array is gestopt, dan tekenen we de chart al zonder data en krijgen we niets te zien. De CSV inlezen is een asynchrone functie van D3 en het tekenen van de chart moet synchroon gebeuren.

charts.chart("Areachart") zegt ons dat we een areachart willen tekenen. Dus dat we in deze case van de switch-case zitten. Daarna zeggen we in welke div we de SVG willen tekenen. De output van de readCSV() functie is een array van objecten en deze geven we mee aan de laatste functie die de chart effectief zal tekenen.

```
1                   ////////////////////////////////////////////////////////////////
2                   ////////////////////////////////////////////////JQUERY UI///////
3                   ////////////////////////////////////////////////////////////////
4
5
6                   //Give the buttons the jQuery style
7
8                   $(".btnz").button();
9
10                  //Color picker widget
11                  $('#text-color-picker').ColorPicker({
```

```
12                          //Default value
13                          color: '#000',
14                          onShow: function (colpkr) {
15                                  $(colpkr).fadeIn(500);
16                                  return false;
17                          },
18                          onHide: function (colpkr) {
19                                  $(colpkr).fadeOut(500);
20                                  return false;
21                          },
22                          onChange: function (hsb, hex, rgb) {
23                                  //Change color of country
24                                  changeChartTextColor((hex));
25                                  //Change background of color picker
26                                  $('#text-color-picker div').css('backgroundColor', '
27                                  //Set a global variable that we need when we turn of
28                                  chartTextColor = hex;
29                          }
30                  });
31                  $('#area-color-picker').ColorPicker({
32                          //Default value
33                          color: '#000',
34                          onShow: function (colpkr) {
35                                  $(colpkr).fadeIn(500);
36                                  return false;
37                          },
38                          onHide: function (colpkr) {
39                                  $(colpkr).fadeOut(500);
40                                  return false;
41                          },
42                          onChange: function (hsb, hex, rgb) {
43                                  //Change color of country
44                                  charts.chart.Areachart.Generate.UpdateAreaColor(hex)
45                                  //Change background of color picker
46                                  $('#area-color-picker div').css('backgroundColor', '
47
48                          }
49                  });
50
51
52                  $('#data-dialog').dialog({
53                          autoOpen: false,
54                          title: 'Import Your Data',
55                          modal: false,
56                          resizable: false,
57                          height: 220,
58                          width: 450
59                  });
60
61                  $('#axes-dialog').dialog({
62                          autoOpen: false,
```

```
63                      title: 'Axes Settings',
64                      modal: false,
65                      resizable: false,
66                      height: 240,
67                      width: 550
68              });
69
70              $('#area-dialog').dialog({
71                      autoOpen: false,
72                      title: 'Area Settings',
73                      modal: false,
74                      resizable: false,
75                      height: 300,
76                      width: 740
77              });
78
79              $('#moving-average-dialog').dialog({
80                      autoOpen: false,
81                      title: 'Add a Moving Average Line',
82                      modal: false,
83                      resizable: false,
84                      height: 240,
85                      width: 450
86              });
87
88              $('#colors-dialog').dialog({
89                      autoOpen: false,
90                      title: 'Color Settings',
91                      modal: false,
92                      resizable: false,
93                      height: 220,
94                      width: 400
95              });
96
97              $('#text-dialog').dialog({
98                      autoOpen: false,
99                      title: 'Text Settings',
100                     modal: false,
101                     resizable: false,
102                     height: 220,
103                     width: 450
104             });
105
106             $('#export-dialog').dialog({
107                     autoOpen: false,
108                     title: 'Export Your Chart',
109                     modal: false,
110                     resizable: false,
111                     height: 220,
112                     width: 450
113             });
```

```
114
115                 $(function () {
116                         $("#opacity-slider").slider({
117                                 value: 100,
118                                 min: 0,
119                                 max: 100,
120                                 step: 1,
121                                 slide: function (event, ui) {
122                                         charts.chart.Areachart.Generate.UpdateOpacit
123
124                                 }
125                         })
126                         $("#opacity-slider").val($("#opacity-slider").slider("value"
127                 });
128
129                  //Dialogs
130                 function showDataDialog() {
131                         $("#data-dialog").dialog("open");
132                 }
133
134                 function showAxesDialog() {
135                         $("#axes-dialog").dialog("open");
136                 }
137
138                 function showAreaDialog() {
139                         $("#area-dialog").dialog("open");
140                 }
141
142                 function showMovingAverageDialog() {
143                         $("#moving-average-dialog").dialog("open");
144                 }
145
146                 function showColorsDialog() {
147                         $("#colors-dialog").dialog("open");
148
149                 }
150
151                 function showTextDialog() {
152                         $("#text-dialog").dialog("open");
153                 }
154
155                 function showExportDialog() {
156                         $("#export-dialog").dialog("open");
157                 }
158
159                 function checkOptionsState() {
160                         if ($("#linear-option-btn").prop("checked") == true) {
161                                 charts.chart.Areachart.Generate.UpdateArea("linear")
162                                 document.getElementById("area-options-description").
163                         }
164                         if ($("#step-option-btn").prop("checked") == true) {
```

```
165                                     charts.chart.Areachart.Generate.UpdateArea("step");
166                                     document.getElementById("area-options-description").
167                             }
168                             if ($("#basis-option-btn").prop("checked") == true) {
169                                     charts.chart.Areachart.Generate.UpdateArea("basis");
170                                     document.getElementById("area-options-description").
171                             }
172                             if ($("#bundle-option-btn").prop("checked") == true) {
173                                     charts.chart.Areachart.Generate.UpdateArea("bundle")
174                                     document.getElementById("area-options-description").
175                             }
176                             if ($("#cardinal-option-btn").prop("checked") == true) {
177                                     charts.chart.Areachart.Generate.UpdateArea("cardinal
178                                     document.getElementById("area-options-description").
    target='_blank' href='http://en.wikipedia.org/wiki/Cubic_Hermite_spline#Cardinal_spl
179
180                             }
181                             if ($("#monotone-option-btn").prop("checked") == true) {
182                                     charts.chart.Areachart.Generate.UpdateArea("monotone
183                                     document.getElementById("area-options-description").
184                             }
185                             if ($("#normal-option-btn").prop("checked") == true) {
186                                     charts.chart.Areachart.Generate.UpdateArea("none");
187                                     document.getElementById("area-options-description").
188                             }
189
190                     }
```

Elk jQuery interface element moet geinitialiseerd worden. Dit gebeurt allemaal in bovenstaande code. Verder is er ook nog de instellingen voor een color-picker-widget van een externe library.

Indien nodig worden ook nog de functies uitgevoerd bij het aanpassen van de waardes van deze elementen. Zo wordt de kleur veranderd van de chart als we die in de color-picker veranderen. De aangeduidde interpolatie functie wordt uitgevoerd als de radio-button gecheckt staat en we kunnen ook de opacity van de chart aanpassen.

```
1                       ////////////////////////////////////////////////////////////////
2                       ////////////////////////////////////////////////JAVASCRIPT//////
3                       ////////////////////////////////////////////////////////////////
```

Onder de JavaScript comment komt alle overige code die nodig is voor de chart te bewerken. Er kan jQuery of D3 code gebruikt worden maar er wordt niets gestijld of geïnitialiseerd.

```
1                       var chartTextColor = "";
2
3                       function changeChartTextColor(inputcolor) {
4                               chartTextColor = "#" + inputcolor;
5
6                               var xAxisText = d3.selectAll(".areachart .xAxis text");
7                               var xAxisLines = d3.selectAll(".areachart .xAxis line");
8                               var xAxis = d3.selectAll(".areachart .xAxis path");
```

```
 9
10                      xAxisText.style("fill", chartTextColor);
11                      xAxisLines.style("stroke", chartTextColor);
12                      xAxis.style("stroke", chartTextColor);
13
14                      var yAxisText = d3.selectAll(".areachart .yAxis text");
15                      var yAxisLines = d3.selectAll(".areachart .yAxis line");
16                      var yAxis = d3.selectAll(".areachart .yAxis path");
17
18                      yAxisText.style("fill", chartTextColor);
19                      yAxisLines.style("stroke", chartTextColor);
20                      yAxis.style("stroke", chartTextColor);
21
22                      $("#areachartTitleID").css("color", chartTextColor);
23
24              }
```

Bovenstaande functie doet de kleur van de tekst veranderen op de assen, de assen zelf en de titel van de chart. We selecteren de text, line en path elementen en veranderen dan hun attribuutwaarden. Belangrijk om te weten is dat we voor de tekst het attribuut "fillïn de style moeten aanpassen en niet "color". Voor de lijnen is het "stroke". Ten slotte voor de titel dat buiten het SVG element staat, kunnen we jQuery code gebruiken om deze te selecteren en zijn "color"style aan te passen.

```
 1              function GenerateSVG() {
 2                      var html = d3.select(".areachart")
 3                              .attr("title", "areachartdiv")
 4                              .attr("version", 1.1)
 5                              .attr("xmlns", "http://www.w3.org/2000/svg")
 6                              .node().parentNode.innerHTML;
 7
 8                      var newwindow = window.open('', 'name', 'height=400,width=10
 9
10
11
12
13                      var newwindowroot = d3.select(newwindow.document.body).appen
14                              .attr("id", "SVG")
15                              .append("img")
16                              .attr("src", "" + btoa(htm
17              }
18
19              function GenerateHTML() {
20                      var html = d3.select("#areachartdiv")
21                              .attr("title", "areachartdiv")
22                              .attr("version", 1.1)
23                              .attr("xmlns", "http://www.w3.org/2000/svg")
24                              .node().parentNode.innerHTML;
25
26                      var newwindow = window.open('', 'name', 'height=400,width=10
27
```

```
28                          newwindow.document.write(html);
29                          newwindow.document.write("<textarea>");
30                          newwindow.document.write(html);
31                          newwindow.document.write("</textarea>");
32
33                          return html;
34
35                  }
36
37                  function GenerateJPEG() {
38
39
40
41
42                          var html = d3.select(".areachart")
43                                  .attr("title", "areachartdiv")
44                                  .attr("version", 1.1)
45                                  .attr("xmlns", "http://www.w3.org/2000/svg")
46                                  .node().parentNode.innerHTML;
47
48                          var imgsrc = '' + btoa(html);
49                          var img = '<img src="' + imgsrc + '">';
50                          //d3.select("#svgdataurl").html(img);
51
52                          var canvas = document.querySelector("canvas");
53                          var context = canvas.getContext("2d");
54
55                          //Clear the canvas incase we drew something on the canvas al
56                          // Store the current transformation matrix
57                          context.save();
58
59                          // Use the identity matrix while clearing the canvas
60                          context.setTransform(1, 0, 0, 1, 0, 0);
61                          context.clearRect(0, 0, canvas.width, canvas.height);
62
63                          // Restore the transform
64                          context.restore();
65
66                          var image = new Image;
67                          image.src = imgsrc;
68                          image.onload = function () {
69
70                                  context.drawImage(image, 0, 0);
71
72                                  var canvasdata = canvas.toDataURL("image/jpg");
73
74                                  var pngimg = '<img src="' + canvasdata + '">';
75                                  //d3.select("#pngdataurl").html(pngimg);
76
77                                  var a = document.createElement("a");
78                                  a.download = "sample.jpg";
```

```
79                                  a.href = canvasdata;
80                                  a.click();
81                          };
82                  }
```

Het exporteren zal altijd een nieuw venster openen dat laat zien wat je krijgt. Niet alles kan geëxporteerd worden. De mouse-over events worden niet mee doorgegeven als we de HTML exporteren omdat deze in de chartwizard-library zitten. Net zoals de CSS die in een extern bestand kan zitten. Je moet dus de CSS van de charts inline in de code zetten.

De functies om de chart te exporteren naar een SVG, HTML of PNG bestand zijn niet zo eenvoudig. Om te exporteren naar een SVG bestand komt het erop neer om het volledige SVG element te selecteren en dit om te zetten naar base 64 data om het in de "src"van een <img>element te stoppen. De gebruiker kan dan op de rechtermuisknop klikken om de SVG op te slaan.

De HTML exporteren is niet al te ingewikkeld. We selecteren de volledige innerHTML van de waar de SVG in zit en laten deze zien in een nieuw venster en in een textarea zodat we deze makkelijk kunnen kopieren.

Het exporteren naar een PNG is het meest ingewikkelde. De innerHTML van de waar de chart in zit wordt geselecteerd en deze wordt gekopieerd naar een HTML canvas element. Indien we dit meerdere keren willen doen moet het canvas uiteraard altijd leeggemaakt worden voor we iets neiuws tekenen. Een post op Stackoverflow[1] legt uit hoe dit op verschillende manieren kan. Daarna kunnen we een functie aanroepen die eigen is aan het canvas element om alles wat hierin getekend is te rasterizen naar een PNG bestand.

- [1]: http://stackoverflow.com/questions/2142535/how-to-clear-the-canvas-for-redrawin

```
1
2                   function ReadCSV() {
3                           var stringpath = document.getElementById("input-file").value
4                           return stringpath.slice(12);
5                   }
```

Het inlezen van de CSV geeft de naam van het bestand dat we hebben gekozen in het Data dialoogvenster terug. Als we de waarde van het element met het ID input-file opvragen is dit een string met het pad van het bestand. Hiervan laten we dan de eerste 12 karakters weg om alleen de bestandsnaam te krijgen.

Merk op dat deze bestanden in de map zitten naast de andere .HTML bestanden en we gewoon andere selecteren. Er is geen upload functie die bestanden op de server zet en deze inleest.

```
1                   function UpdateChartTitle(inputtext) {
2                           document.getElementById("areachartTitleID").innerHTML = inpu
3                   }
```

Functie om de titel van de chart te veranderen naar de text input.

```
1                   function updateChart() {
2                           clearChart();
```

De chart updaten is simpelweg alle elementen in het SVG element verwijderen. Niet het SVG element zelf. En dan de chart opnieuw genereren met de nieuwe data.

```
1
2                           charts.getCSV(ReadCSV(), function (outputdata) {
3
4                                   data = outputdata;
5
6                                   charts.chart.Areachart.Generate(data);
7
8                           });
9
10                  }
11
12                  function clearChart() {
13
14                          d3.selectAll(".area").remove();
15                          d3.selectAll(".average").remove();
16                          d3.selectAll(".xAxis").remove();
17                          d3.selectAll(".yAxis").remove();
18                  }
19          </script>
20  </body>
21
22  </html>
```

Code chartwizard-library.js voor de area chart

```
1
2           case "AreaChart":
3
4           charts.chart.Areachart = function (divclass) {
5
6                   var areachart1 = d3.select(divclass)
7                           .append("svg")
8                           .attr("width", svgWidth + margin.left + margin.right)
9                           .attr("height", svgHeight + margin.top + margin.bottom)
10                          .attr("class", "areachart")
11                          .append("g")
12                          .attr("transform", "translate(" + (70) + "," + (20) + ")");
```

De eerste functie in de chartwizard-library van elke stuk code voor een chart te genereren bestaat uit een SVG element te tekenen in een . Je geeft de classname mee als parameter en je krijgt een SVG element terug met daarin een <g>element. Een <g>element is een tag om andere elementen te groeperen. Dit maakt het makkelijker om groepen en subgroepen te maken om je chart elementen te ordenen.

```
1
```

```
2           charts.chart.Areachart.Generate = function (inputdata) {
3                   var data = inputdata;
4
5                   var parseDate = d3.time.format("%d-%b-%y").parse;
6
7                   var x = d3.time.scale()
8                           .range([0, svgWidth]);
9
10                  var y = d3.scale.linear()
11                          .range([svgHeight, 0]);
```

De inputdata die wordt meegegeven aan de Generate() functie is een array van objecten. Deze array komt van het inlezen van het CSV bestand. De parseDate variabele is een functie(javascript variabelen kunnen ook functies zijn) van D3 om de datum te parsen. De x en y variabelen zijn arrays van minimum en maximum waarden die worden berekent met de scale functies binnen D3. Omdat de chart een maximum vaste breedte en hoogte heeft moeten we de data altijd herschalen naar deze waarden. Het kan zijn dat er te veel data is en alles te dicht opeen staat en dat de chart gewoon te klein is. Als dit het geval is moet de gebruiker dit zelf inzien en minder data meegeven in de CSV. Een goede uitleg over hoe het schalen werkt binnen D3 vind je hieronder.

http://chimera.labs.oreilly.com/books/1230000000345/ch07.html

```
1                   var xAxis = d3.svg.axis()
2                           .scale(x)
3                           .orient("bottom");
4
5                   var yAxis = d3.svg.axis()
6                           .scale(y)
7                           .orient("left");
```

De xAxis en yAxis variabelen stellen de assen voor. Hiervoor worden de x en y waarden gebruikt die hiervoor zijn berekend om de minimum en maximum in te stellen.

```
1                   var area = d3.svg.area()
2                           .x(function (d) {
3                                                       //Add +1 to move it one pixe
4                                   return x(d.date)+1;
5                           })
6                           .y0(svgHeight)
7                           .y1(function (d) {
8                                   return y(d.close);
9                           });
10
11
12
13                  data.forEach(function (d) {
14                          d.date = parseDate(d.date);
15                          d.close = +d.close;
16                  });
```

```
17
18
19                  x.domain(d3.extent(data, function (d) {
20                          return d.date;
21                  }));
22                  y.domain([0, d3.max(data, function (d) {
23                          return d.close;
24                  })]);
```

Het berekenen van de waarden nodig om het path element te tekenen gebeurt hierboven. d.close en d.date verwijzen naar de data. close en date zijn de namen van de variabelen in de CSV. Dus als we die veranderen zal de code niet werken. Het is dus belangrijk dat de gebruiker weet dat hij zijn twee waarden de namen date en close moet geven in het CSV bestand. De chart is nog niet getekend.

```
1
2                   //remove previous chart if there was one
3                   areachart1.selectAll("path").remove();
4
5                   areachart1.append("path")
6                           .datum(data)
7                           .attr("class", "area")
8                           .attr("d", area)
9                           .style({
10                                  "fill": "steelblue"
11                          });
```

De data is nu in de juiste vorm om het gebied te tekenen. We geven ze een class name en tekenen het path element met een "fill"style voor de kleur.

```
1
2                   areachart1.select(".xAxis").remove();
3                   areachart1.select(".yAxis").remove();
4
5
6                   areachart1.append("g")
7                           .attr("class", "xAxis")
8                           .attr("transform", "translate(0," + svgHeight + ")")
9                           .call(xAxis)
10                          .append("text")
11                          .attr("class", "xAxisText")
12                          .attr("y", 50)
13                          //Hoever van de as verwijdert
14                          //ten opzichte van de y as
15                  .attr("x", svgWidth / 2)
16                          .text("Date")
17                          .style({
18                                  "font-family": "sans-serif",
19                                  "font-size": 18,
20
21                          });;
```

```
22
23                  areachart1.append("g")
24                          .attr("class", "yAxis")
25                          .call(yAxis)
26                          .append("text")
27                          .attr("class", "yAxisText")
28                          .attr("transform", "rotate(-90)")
29                          .attr("y", -50)
30                          .attr("x", -svgHeight / 2)
31                          .text("Value")
32                          .style({
33                                  "font-family": "sans-serif",
34                                  "font-size": 18,
35
36                          });
37
38                  //inline CSS styling
39
40                  areachart1.selectAll(".domain")
41                          .style({
42                                  "fill": "none",
43                                  "stroke": "#CCC",
44                                  "shape-rendering": "crispEdges"
45                          });
46
47                  areachart1.selectAll("line")
48                          .style({
49                                  "fill": "none",
50                                  "stroke": "#CCC",
51                                  "shape-rendering": "crispEdges"
52                          });
53
54                  //Give all X values a class and style them
55                  areachart1.selectAll(".xAxis g text").attr("class", "xAxisValues")
56                          .style({
57                                  "font-family": "sans-serif",
58                                  "font-size": 12,
59
60                          });
61
62                  //Give all Y values a class and style them
63                  areachart1.selectAll(".yAxis g text").attr("class", "yAxisValues")
64                          .style({
65                                  "font-family": "sans-serif",
66                                  "font-size": 12,
67
68                          });
69
70                                                  //Color the axis names too
71                                              areachart1.select(".xAxisText").styl
72                                              areachart1.select(".yAxisText").styl
```

Het toevoegen van de assen is analoog met het gebied. Let op dat de assen wel in het <g>element zitten in de SVG en elke as nog is in een aparte <g>. Je kan nooit teveel <g>elementen tekenen. Net zoals er nooit te veel elementen kunnen zijn. Zoals ervoor al is vermeld, wordt de styling inline gebeurt zodanig dat die wordt behouden bij het exporteren. De SVG moet één geheel vormen en mag niets extern nodig hebben.

```
1
2
3                   //Calculate moving average
4                   // Setup the moving average calculation.
5                   // Currently is a hacky way of doing it by manually storing and usin
6                   // Looking for another way to address previous values so we can make
7                   var prevPrevVal = 0;
8                   var prevVal = 0;
9                   var curVal = 0
10                  var movingAverageLine = d3.svg.line()
11                          .x(function (d, i) {
12                                  return x(d.date);
13                          })
14                          .y(function (d, i) {
15                                  if (i == 0) {
16                                          prevPrevVal = y(d.close);
17                                          prevVal = y(d.close);
18                                          curVal = y(d.close);
19                                  } else if (i == 1) {
20                                          prevPrevVal = prevVal;
21                                          prevVal = curVal;
22                                          curVal = (prevVal + y(d.close)) / 2.0;
23                                  } else {
24                                          prevPrevVal = prevVal;
25                                          prevVal = curVal;
26                                          curVal = (prevPrevVal + prevVal + y(d.close)
27                                  }
28                                  return curVal;
29
30                          })
31                          .interpolate("basis");
32
33                  //Draw moving average but don't show it yet
34                  areachart1.append("path")
35                          .attr("class", "average")
36                          .attr("d", movingAverageLine(data))
37                          .style({
38                                  "stroke": "darkviolet",
39                                  "stroke-width": 1,
40                                  "fill": "none",
41                                  "opacity": 0
42                          });
```

Als extra functionaliteit wordt hierboven een moving average lijn berekend en getekend. De lijn is echter wel nog doorzichtig. Dit wordt pas veranderd indien de gebruiker de checkbox aanzet

voor het weergeven van de moving average. Herinner dat SVG elementen geen "z-index" hebben en dus worden gestapeld in volgorde van het tekenen. Het laatst toegevoegde element wordt dus bovenop de rest geplaatst. Het is dus belangrijk dat de moving average pas wordt getekend na het tekenen van het gebied om ze er bovenop te plaatsen.

```
1
2               charts.chart.Areachart.Generate.UpdateArea = function (inputsmoothingtyp
3
4                       if (inputsmoothingtype == "linear") {
5
6                               area = d3.svg.area()
7                                       .interpolate("linear")
8
9                               .x(function (d) {
10                                      return x(d.date)+1;
11                              })
12                                      .y0(svgHeight)
13                                      .y1(function (d) {
14                                              return y(d.close);
15                                      });
16
17                              areachart1.selectAll(".area").remove();
18
19                              areachart1.append("path")
20                                      .datum(data)
21                                      .attr("class", "area")
22
23                              .attr("d", area)
24                                      .style({
25                                              "fill": "steelblue"
26                                      });
27                      }
28
29                      if (inputsmoothingtype == "step") {
30
31                              area = d3.svg.area()
32                                      .interpolate("step")
33
34                              .x(function (d) {
35                                      return x(d.date)+1;
36                              })
37                                      .y0(svgHeight)
38                                      .y1(function (d) {
39                                              return y(d.close);
40                                      });
41
42                              areachart1.selectAll(".area").remove();
43
44                              areachart1.append("path")
45                                      .datum(data)
46                                      .attr("class", "area")
```

```
47                                      .attr("d", area)
48                                      .style({
49                                              "fill": "steelblue"
50                                      });
51                      }
52
53                      if (inputsmoothingtype == "basis") {
54
55                              area = d3.svg.area()
56                                      .interpolate("basis")
57                                      .x(function (d) {
58                                              return x(d.date)+1;
59                                      })
60                                      .y0(svgHeight)
61                                      .y1(function (d) {
62                                              return y(d.close);
63                                      });
64
65                              areachart1.selectAll(".area").remove();
66
67                              areachart1.append("path")
68                                      .datum(data)
69                                      .attr("class", "area")
70                                      .attr("d", area)
71                                      .style({
72                                              "fill": "steelblue"
73                                      });
74                      }
75
76                      if (inputsmoothingtype == "bundle") {
77
78                              area = d3.svg.area()
79                                      .interpolate("bundle")
80                                      .x(function (d) {
81                                              return x(d.date)+1;
82                                      })
83                                      .y0(svgHeight)
84                                      .y1(function (d) {
85                                              return y(d.close);
86                                      });
87
88                              areachart1.selectAll(".area").remove();
89
90                              areachart1.append("path")
91                                      .datum(data)
92                                      .attr("class", "area")
93                                      .attr("d", area)
94                                      .style({
95                                              "fill": "steelblue"
96                                      });
97                      }
```

```
 98
 99                     if (inputsmoothingtype == "cardinal") {
100
101                             area = d3.svg.area()
102                                     .interpolate("cardinal")
103                                     .x(function (d) {
104                                             return x(d.date)+1;
105                                     })
106                                     .y0(svgHeight)
107                                     .y1(function (d) {
108                                             return y(d.close);
109                                     });
110
111                             areachart1.selectAll(".area").remove();
112
113                             areachart1.append("path")
114                                     .datum(data)
115                                     .attr("class", "area")
116                                     .attr("d", area)
117                                     .style({
118                                             "fill": "steelblue"
119                                     });
120                     }
121
122                     if (inputsmoothingtype == "monotone") {
123
124                             area = d3.svg.area()
125                                     .interpolate("monotone")
126                                     .x(function (d) {
127                                             return x(d.date)+1;
128                                     })
129                                     .y0(svgHeight)
130                                     .y1(function (d) {
131                                             return y(d.close);
132                                     });
133
134                             areachart1.selectAll(".area").remove();
135
136                             areachart1.append("path")
137                                     .datum(data)
138                                     .attr("class", "area")
139                                     .attr("d", area)
140                                     .style({
141                                             "fill": "steelblue"
142                                     });
143                     }
144                     if (inputsmoothingtype == "none") {
145
146
147
148                             area = d3.svg.area()
```

```
149                                     .interpolate("linear")
150                                     .x(function (d) {
151                                             return x(d.date)+1;
152                                     })
153                                     .y0(svgHeight)
154                                     .y1(function (d) {
155                                             return y(d.close);
156                                     });
157
158                             areachart1.selectAll(".area").remove();
159
160                             areachart1.append("path")
161                                     .datum(data)
162                                     .attr("class", "area")
163                                     .attr("d", area)
164                                     .style({
165                                             "fill": "steelblue"
166                                     });
167
168                     }
169             }
```

Dankzij D3 kunnen we het gebied manipuleren met ingebouwde wiskundige functies. We passen de area variabele aan naargelang welke interpolatie is gekozen door de gebruiker en tekenen de chart opnieuw. Denk eraan dat alleen de area variabele wordt aangepast en niet de data-array afkomstig van de CSV.

```
1               charts.chart.Areachart.Generate.MovingAverage = function (inputstate) {
2
3
4                       if (inputstate == true) {
5
6
7                               areachart1.select(".average").remove();
8
9                               areachart1.append("path")
10                                      .attr("class", "average")
11                                      .attr("d", movingAverageLine(data))
12                                      .style({
13                                              "stroke": "darkviolet",
14                                              "stroke-width": 1,
15                                              "fill": "none",
16                                              "opacity": 1
17                                      });
18
19
20                              // Draw the moving average version of the data, as a lin
21
22                      } else
23                              areachart1.select(".average")
24                                      .style({
```

```
25                                                  "opacity": 0
26                                          });
27                          //areachart1.select(".average").remove();
28                  }
```

Om de moving average te plaatsen wordt alleen de opacity aangepast omdat de lijn al ervoor was getekend maar onzichtbaar was. De inputstate verwijst naar het feit of de checkbox aangeduid is of niet.

```
 1                  charts.chart.Areachart.Generate.UpdateYaxisName = function (inputtex
 2                          yAxisText = inputtext;
 3
 4                          areachart1.select(".yAxisText")
 5                                  .text(inputtext);
 6                  }
 7
 8                  charts.chart.Areachart.Generate.UpdateXaxisName = function (inputtex
 9                          xAxisText = inputtext;
10                          areachart1.select(".xAxisText")
11                                  .text(inputtext);
12                  }
13
14                  charts.chart.Areachart.Generate.UpdateAreaColor = function (inputcol
15
16                          areachart1.select(".area")
17                                  .style({
18                                          "fill": inputcolor
19                                  });
20
21                  }
22
23                  charts.chart.Areachart.Generate.UpdateOpacity = function (inputvalue
24                          areachart1.select(".area")
25                                  .style({
26                                          "opacity": (inputvalue / 100)
27                                  });
28                  }
29
30          }
31
32  }
33
34  break;
```

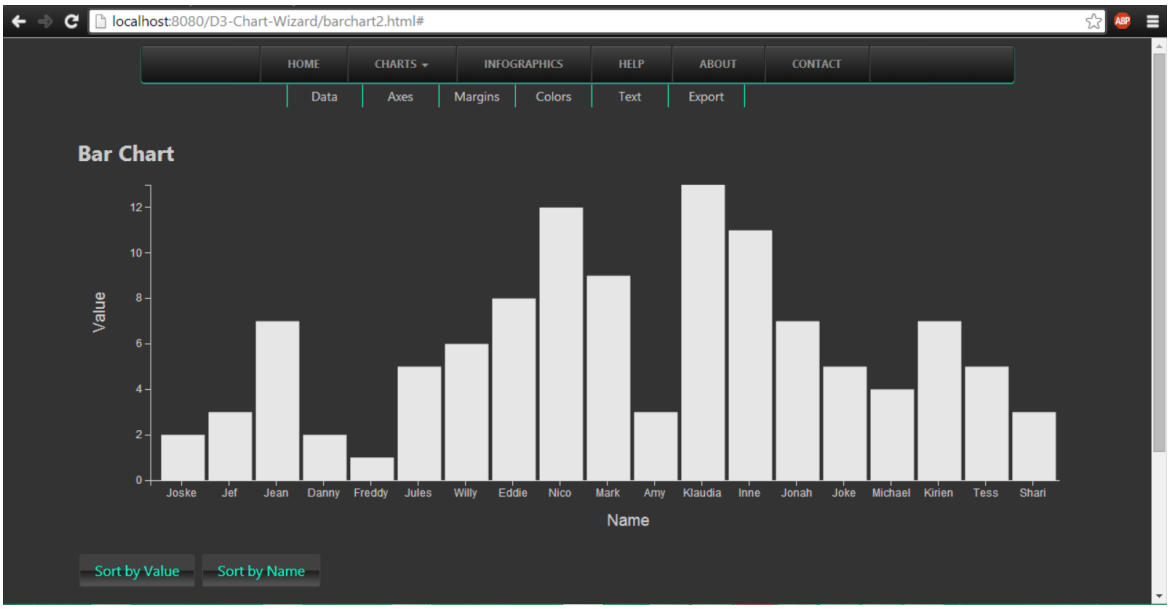
Ten slotte worden op het einde de update functies gezet om de chart aan te passen naar de wensen van de gebruiker. Steeds wordt het specifieke element geselecteerd met zijn class of ID en dan worden de attributen aangepast. Soms is dit niet mogelijk en moeten we de chart helemaal opnieuw tekenen. Het hangt ervan af wat je wilt veranderen.

CSV bestand

```
 1  date,close
 2  1-May-12,13
 3  30-Apr-12,98
 4  27-Apr-12,0
 5  26-Apr-12,70
 6  25-Apr-12,0
 7  24-Apr-12,28
 8  23-Apr-12,170
 9  20-Apr-12,98
10  19-Apr-12,44
11  18-Apr-12,34
12  17-Apr-12,70
13  16-Apr-12,12
14  13-Apr-12,23
15  12-Apr-12,77
16  11-Apr-12,20
17  10-Apr-12,44
18  9-Apr-12,23
19  5-Apr-12,68
20  4-Apr-12,31
21  3-Apr-12,32
22  2-Apr-12,63
23  30-Mar-12,55
24  29-Mar-12,86
25  28-Mar-12,62
26  27-Mar-12,48
27  26-Mar-12,98
28  23-Mar-12,05
29  22-Mar-12,34
30  21-Mar-12,50
```

2.10.2 Bar Chart

De bar chart wizard heeft net dezelfde functionaliteiten als de area chart wizard met nog een paar extra. Je kan de x waarden alfabetisch ordenen of volgens de y waarden van hoog naar laag. Er is ook een mouse-over functie om de huidige waarde van elke balk te laten weergeven.



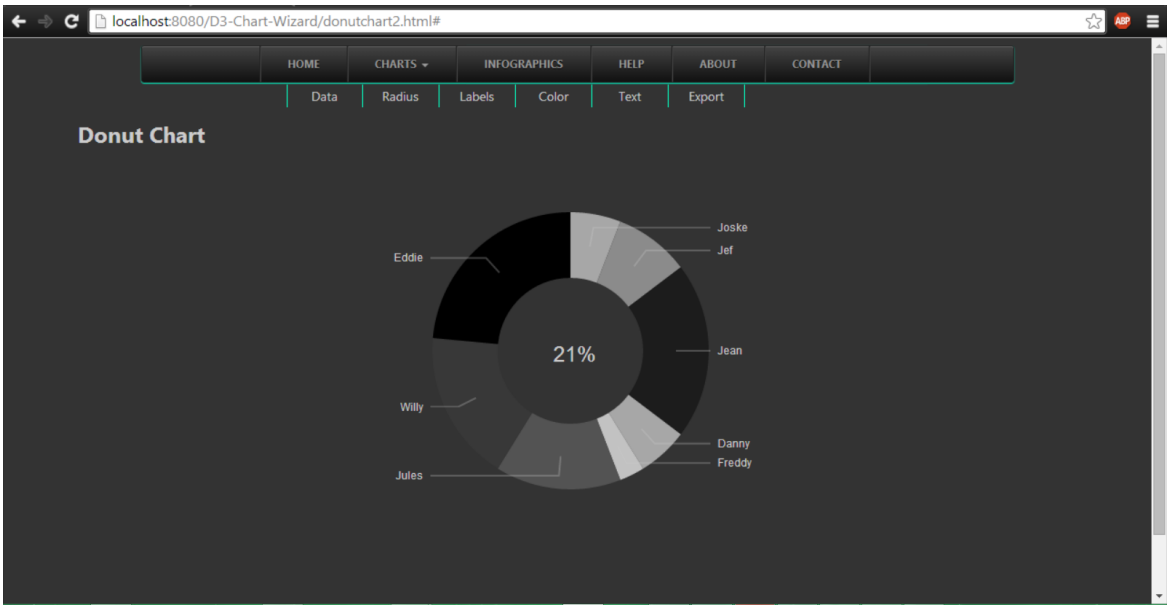
Figuur 2.7: Screenshot Bar Chart Wizard

CSV bestand

```
1  key,value
2  Joske,2
3  Jef,3
4  Jean,7
5  Danny,2
6  Freddy,1
7  Jules,5
8  Willy,6
9  Eddie,8
10 Nico,12
11 Mark,9
12 Amy,3
13 Klaudia,13
14 Inne,11
15 Jonah,7
16 Joke,5
17 Michael,4
18 Kirien ,7
19 Tess,5
20 Shari,3
```

2.10.3 Donut Chart

Een donut chart geeft de data volledig anders weer. De data wordt verdeeld over een volledige cirkel met een gat in het midden. Je kan de straal aanpassen om de cirkel groter of kleiner te maken, de kleur aanpassen en labels aan- of uitzetten.

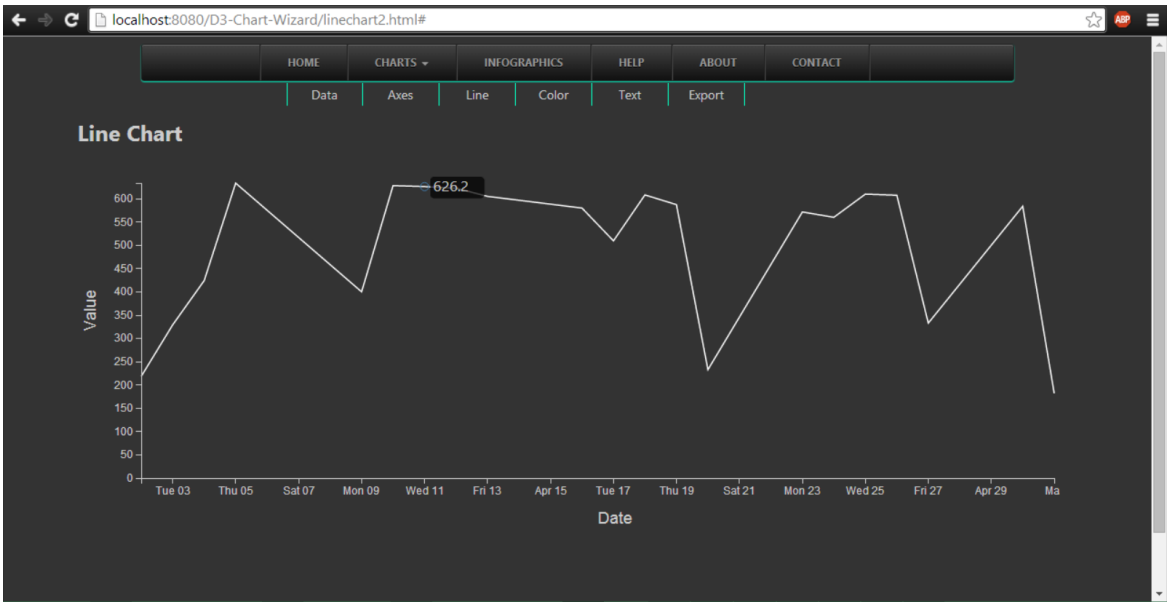


Figuur 2.8: Screenshot Donut Chart Wizard

In dit voorbeeld wordt hetzelfde CSV bestand gebruikt als die van de bar chart.

2.10.4 Line Chart

De line chart wizard is identiek aan de area chart. Alleen is de lijn niet opgevuld. Om ervoor te zorgen dat deze wizard toch iets anders is, is er een mouse-over functie toegevoegd om de huidige waarde weer te geven. Dit is vooral handig als er veel waardes zijn. Jammer werkt deze mouse-over functie niet meer wanneer we de lijn veranderen met de interpolatie functies zoals in de area chart. Zoals eerder vermeld komt dit doordat de D3 functies alleen het path element veranderen en niet de data. De mouse-over functie volgt de data en geeft deze weer, niet het pad van de lijn.

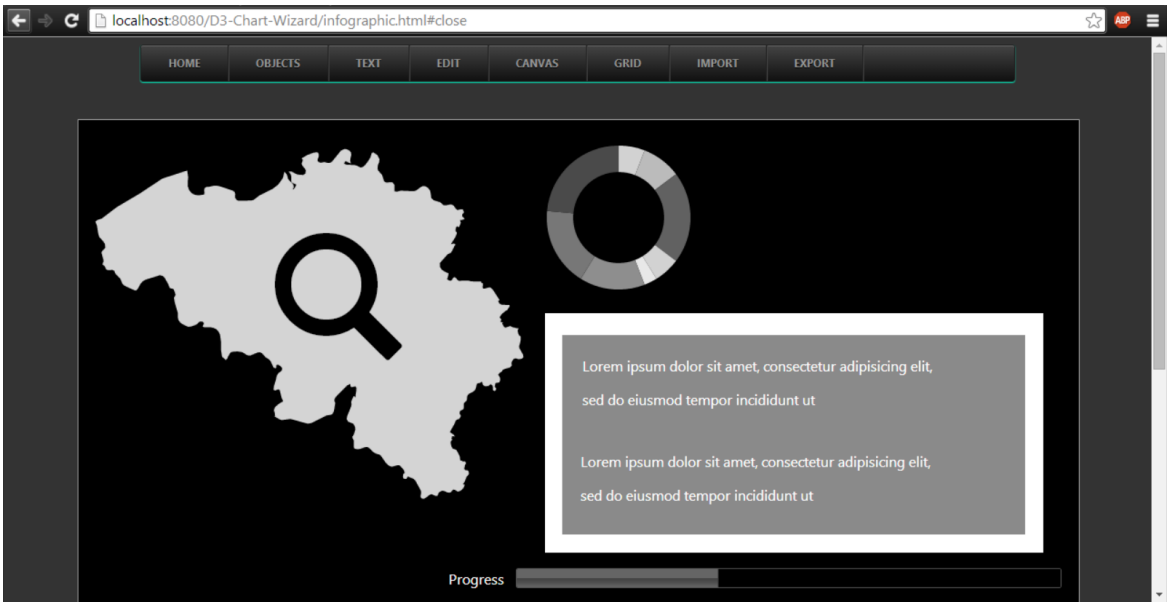


Figuur 2.9: Screenshot Line Chart Wizard

In dit voorbeeld wordt hetzelfde CSV bestand gebruikt als die van de area chart.

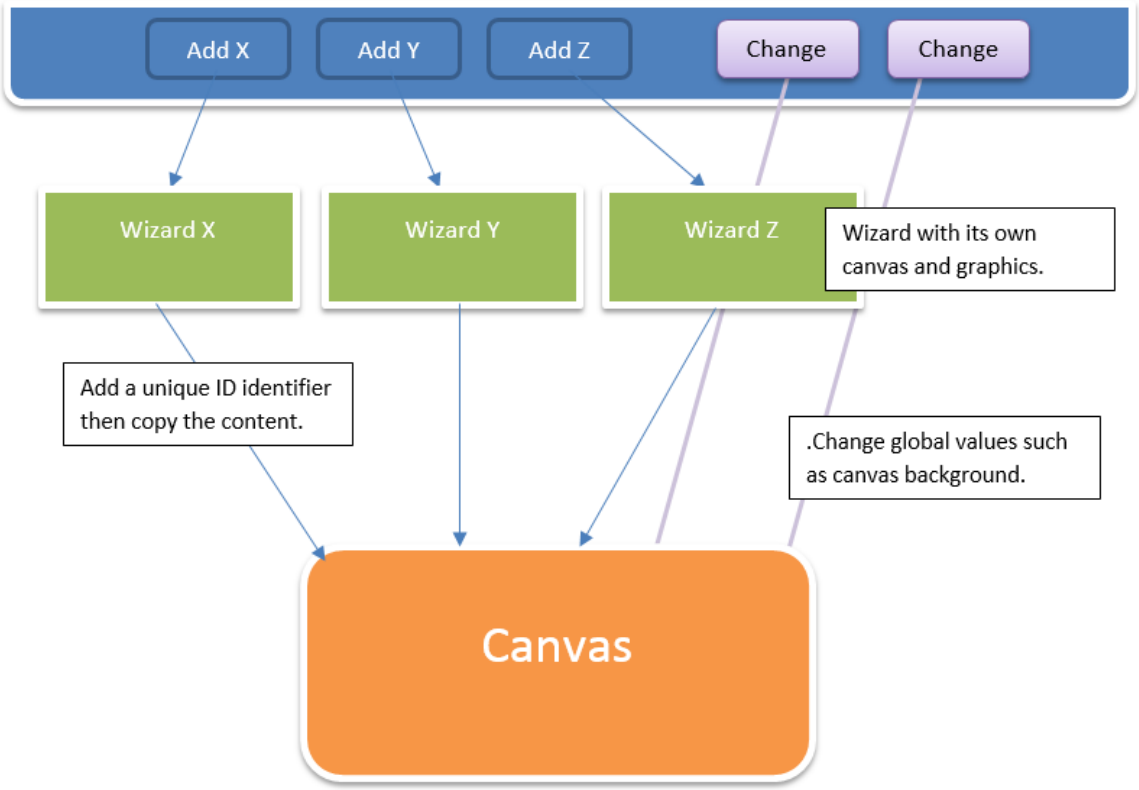
2.11 Infographic Wizard

Information Graphics of kortweg, infographics, zijn bedoeld om informatie te weergeven door middel van tekst en beeld. Dit kan simpelweg een paar pictrogrammen zijn of een hele uitleg rond de economische toestand van een land. Deze wizard is dan ook bedoeld om zoiets te kunnen nabootsen met de D3 library. Dit vraagt enorm veel code omdat de wizard de fantasie en ideeen van de gebruiker niet mag beperken. Met andere woorden alles moet mogelijk zijn. Deze applicatie is dan ook zeker niet volledig afgeraakt.



Figuur 2.10: Screenshot Infographic Wizard

Bovenaan staat het menu met alle functionaliteiten. De algemene workflow, en ook de structuur van de code, is als volgt: je kiest wat voor object je wilt toevoegen en bewerkt deze in een mini wizard. Dan wordt deze gekopieerd en toegevoegd in het canvas met een unieke ID en kan je deze eender waar slepen. De unieke ID zorgt ervoor dat deze niet opnieuw geselecteerd wordt in de wizard als we nog eenzelfde object willen toevoegen.



Er is nog geen functionaliteit voorzien om deze objecten terug te selecteren en te bewerken. Je moet ze allemaal verwijderen en opnieuw beginnen als het niet naar wens is. Je kan ook een grid aanzetten dat altijd boven de objecten zal komen te staan om ze te ordenen. Je kan de resolutie van het canvas aanpassen en uiteraard ook tekst toevoegen.

De volgende objecten kunnen toegevoegd worden:

1. Charts
   (a) Donut Chart
2. Countries
3. Icons
4. Shapes
5. Progress Bars

De wizard voor de donut chart is min of meer dezelfde als die van ervoor. Alleen hier is ze iets compacter en is er geen export mogelijkheid omdat we ze in de infographic steken en die dan exporteren. De andere charts zijn nog niet toegevoegd.

Het uiteindelijke resultaat van de infographic is een mix van en SVG elementen. Dit is geen ideale code omdat dit het heel lastig maakt om het te exporteren. We kunnen de SVG elementen exporteren maar niet met de elementen erbij. Er is echter een

element genaamd "foreign object"binnen D3 dat gewone HTML elementen aanschouwt als SVG elementen.. Maar er was niet genoeg tijd om dit te testen en volledig uit te werken.

`https://gist.github.com/mbostock/1424037`

Totzover kun je kiezen uit 4 landen om toe te voegen. Dit komt omdat het geen gemakkelijke opdracht is om de data in de juiste vorm te krijgen en ze te dan te visualiseren. Indien we een land naar keuze willen toevoegen in onze code moeten we een aantal stappen ondernemen. Eerst downloaden we de wereld.

`http://www.naturalearthdata.com/downloads/10m-cultural-vectors/`

We kunnen dit shape bestand openen met QGIS.

`http://www.qgis.org/en/site/`

Open QGIS en sleep het shape bestand in de editor. Zorg ervoor dat de optie van bewerken aanstaat. Nu kan je eender welk land selecteren. Klik met de rechtermuisknop op het land en klik op save selection as geoJSON data. Nu hebben we de data, alleen staat deze nog niet in de vorm die D3 vraagt. Men moet deze nog comprimeren naar topoJSON data. Dit kan je doen op de volgende website:

`http://shancarter.github.io/distillery/`

Het is belangrijk te weten onder welk bestandsnaam je dit opslaat omdat er een variabele is die dezelfde naam heeft in de code. En deze zal gebruikt worden door D3 om de data in te lezen. Misschien is er snellere, betere manier om dit te doen maar dit was de eerste oplossing.

Meer uitleg vind je op `http://bost.ocks.org/mike/map/`

Hoofdstuk 3

Resultaten

Het resultaat van deze boeiende opdracht is een stijlvolle krachtige website die 4 wizards heeft voor basic charts en één grotere wizard voor infographics te tekenen. Je kan alleen CSV bestanden uploaden maar wel naar 3 verschillende formaten exporteren. Een handige tool voor websites die vaak informatieve tekst tonen en hierbij een relevante figuur willen.

3.1 Toekomst

Indien er ooit verder zou worden gewerkt aan deze code dan lijst ik hier nog enkel van mijn ideeën om de applicatie uit te breiden.

- Zorg voor meer functionaliteiten voor elke basic chart wizard.
- Voeg een optie toe voor de eenvoudige charts om zelf je data te kunnen invoegen zonder een bestand te moeten gebruiken.
- Zorg ervoor dat de website een FTP server heeft waar de data naar kan worden geupload.
- Zorg ervoor dat alle eigenschappen die je kan aanpassen aan de chart een globale variabele zijn en deze kan meegeven aan de functie die de chart tekent zodat je deze niet terug moet aanpassen wanneer je de chart opnieuw tekent met andere data.
- Voeg functionaliteit toe om meerdere datasets te kunnen inlezen.
- Voeg alle landen toen in de infographics wizard.
- Zorg ervoor dat je alle elementen afzonderlijk kan selecteren, verwijderen en bewerken in de infographics wizard.
- Maak wizards voor complexere charts

Hoofdstuk 4

Besluit

D3 is uw zwitsers zakmes om charts en SVG elementen op een Data-Driven manier weer te geven op het web. Het duurt wel enige tijd voor je alles begrijpt en er mee weg bent. Gelukkig zijn er heel veel bronnen op het internet om je op weg te helpen.

Een volledige wizard met D3 maken is geen gemakkelijke opdracht. Deze applicatie is maar een beginsel. Ik zie zeker wel een nut in de basic chart wizards. Zeker wanneer er nog meer functionaliteit wordt toegevoegd om de charts te bewerken. De infographics wizard heeft nog niet genoeg features om echt gebruikt te kunnen worden en ik vrees dat de meeste web designers en grafici toch photoshop zullen gebruiken. Daarom moet deze infographic wizard meer gelinkt zijn met echte data om toch uniek te zijn. Het kunnen invoegen van landen via data is wel iets interessant.

Uiteindelijk ben ik tevreden over deze opdracht en heb ik enorm veel bijgeleerd. Ik kan nu met trots zeggen dat ik een degelijke kennis heb van web development en in het bijzonder over het visualiseren van data.