Onderzoek Percolator

**Versie 1.1**

**Gemaakt door:**

David Kerkkamp (521142)

Thomas Peters (537702)

Joeri Smits (524292)

Serhat Tunç (491186)

Leroy Witteveen (523896)

**Course:**

DWA Project

**Begeleiders:**

Theo Theunissen

Lars Tijsma

**Datum:**

14 november 2014

Versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Datum** | **Wijziging** | **Naam** |
| 1.0 | 13 november 2014 | Eerste opzet | Allen |
| 1.1 | 19 november 2014 | Eerste keer feedback verwerkt | Allen |
| 1.2 | 21 november 2014 | Tweede keer feedback verwerkt | Allen |

Inhoud

Versiebeheer

Inhoud

1. Probleemstelling

1.1 De huidige situatie

1.2 De gewenste situatie

1.3 Het verschil tussen de huidige en gewenste situatie

2. Doelstelling

3. Vraagstellingen

5. Grafische Libraries

5.1 RaphaëlJS

5.2 BonsaiJS

5.3 FabricJS

5.4 PaperJS

5.5 oCanvas

5.6 Conclusie

6. Teksteditors

6.1 Overzicht

6.2 Ace

6.3 CodeMirror

6.4 SyntaxHighlighter

6.5 Edit Area

6.6 LDT (Lightweight Decorator for Textareas)

6.7 Conclusie

7. Realtime samenwerken

7.1 Zelf implementeren bovenop Socket.IO

7.2 Zelf implementeren bovenop WebRTC

7.3 TogetherJS gebruiken

7.4 Conclusie

8. Bijlagen

9. Bibliografie

1. Probleemstelling

1.1 De huidige situatie

Op dit moment heeft de projectgroep nog geen duidelijk inzicht in welke libraries er beschikbaar zijn voor onze functionaliteiten binnen de Percolator. De projectgroep weet wel dat er libraries beschikbaar zijn voor de volgende functionaliteiten:

* Geheugenmodellen tekenen
* Een teksteditor voor het weergeven van code in de browser
* Real-time (hagerAly, 2013) communicatie tussen gebruikers.

1.2 De gewenste situatie

De projectgroep wil weten welke libraries het meest geschikt zijn voor de bovenstaande functionaliteiten. Daarnaast wil de groep ook graag weten of er wijzigingen kunnen worden aangebracht aan de bestaande libraries.

Sommige projectleden hebben al ervaring met bepaalde libraries. Deze ervaring komt goed van pas tijdens dit onderzoek. Waar de projectgroep voor moet uitkijken is dat er geen subjectieve voorkeur komt voor een bepaalde library.

1.3 Het verschil tussen de huidige en gewenste situatie

Op het huidige moment is het probleem dat de projectgroep nog geen kennis heeft van de benodigde libraries voor de Percolator. In dit onderzoek gaat de projectgroep onderzoek doen naar welke libraries op basis van de gestelde eisen het meest geschikt zijn om te gebruiken voor het project. Deze eisen zijn te vinden onder de vraagstellingen.

De gewenste situatie is dan ook dat de projectgroep kennis heeft over welke libraries geschikt zijn voor de bovenstaande functionaliteiten.

Het verschil tussen deze twee situaties is dus de nu nog schaarse kennis van de projectgroep over de verschillende libraries. De projectgroep wil deze kennis op doen door onderzoek te doen.

2. Doelstelling

In dit onderzoeksrapport worden drie libraries onderzocht die aan de gestelde eisen voldoen om geheugenmodellen te tekenen, real-time samen te werken, en code weer te geven.

Voor een geschikte library zoekt de projectgroep kennis voor een aantal functionaliteiten die bij onze vraagstelling worden genoemd. Om deze kennis te krijgen moet de projectgroep onderzoek doen naar de verschillende functionaliteiten per library. Informatie wordt verkregen door middel van:

* Lezen van de officiële documentatie (indien aanwezig)
* Andere gerelateerde bronnen over de library
* Zelf experimenteren
* Voorbeelden bekijken waarin de library is gebruikt.

*Wanneer is een library een goede library?*

De projectgroep beoordeeld de libraries op basis van de hoeveelheid functionaliteiten en de kwaliteit hiervan die de library biedt. De kwaliteit van een library wordt beoordeeld op basis van hoe de implementatie is en hoe functionaliteiten beschreven zijn (documentatie). Deze functionaliteiten staan beschreven onder de vraagstelling.

De library die het meest voldoet aan de eisen komt voor het project het beste uit de test en wordt in gebruik genomen.

3. Vraagstellingen

**Vraagstelling 1**

Welke grafische library kan de projectgroep op basis van onderstaande eisen het best gebruiken om geheugenmodellen te kunnen tekenen?

De eisen aan een grafische tekenlibrary zijn:

1. Figuren tekenen (vierkanten, rechthoeken, etc)
2. Tekst plaatsen in de figuren
3. Positie en grootte van figuren kunnen aanpassen
4. Figuren verbinden door middel van lijnen en pijlen
5. Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON)
6. Een goede documentatie waarin alle functionaliteiten worden beschreven

**Vraagstelling 2**

Welke teksteditor kan de projectgroep op basis van onderstaande eisen het best gebruiken om de code bij de geheugenmodellen goed weer te geven?

De eisen aan een teksteditor zijn:

1. Ondersteuning van meerdere programmeertalen (in ieder geval Java en JavaScript)
2. Syntax highlighting en inspringing
3. Regel nummering
4. Geen merkbare performance impact
5. Implementatie moet zo min mogelijk tijd kosten
6. Goede documentatie

**Vraagstelling 3**

Welke technologie kan de projectgroep op basis van onderstaande eisen het beste gebruiken om de realtime interactie tussen verschillende gebruikers te faciliteren?

De eisen aan een implementatie van real-time interactie zijn:

1. Zo min mogelijk merkbare latentie
2. Implementatie moet zo min mogelijk tijd kosten
3. Zo veel mogelijk browser compatibiliteit

4. Data verzamelen

De volgende lijst met grafische libraries is tot stand gekomen door in Google te zoeken. De populairste zoekresultaten zijn bekeken en uiteindelijk is daar een lijst uit ontstaan. De gebruikte zoektermen zijn:

* JavaScript graphic libraries
* JavaScript drawing libraries
* RaphaëlJS - <http://raphaeljs.com/>
* BonsaiJS - <http://bonsaijs.org/>
* FabricJS - <http://fabricjs.com/>
* PaperJS - <http://paperjs.org/>
* OCanvas - <http://ocanvas.org/>

De volgende lijst met teksteditors is tot stand gekomen door te zoeken met behulp van Google. De populairste resultaten zijn bekeken en daaruit is een lijst ontstaan. De gebruikte zoektermen zijn:

* JavaScript syntax highlighter
* JavaScript code editor
* Ace - <http://ace.c9.io>
* CodeMirror - <http://codemirror.net>
* SyntaxHighlighter - <https://code.google.com/p/syntaxhighlighter/>
* Edit area - <http://www.cdolivet.com/editarea/>
* LDT - <https://github.com/kueblc/LDT/>

De projectgroep kent maar één kant en klare implementatie van realtime samenwerken en dat is TogetherJS. Er zijn twee protocollen waarop een eigen implementatie van realtime samenwerken gemaakt kan worden en dat zijn Web Sockets en WebRTC. Omdat de projectgroep met de course "Develop a Web Application" ervaring op heeft gedaan met Socket.IO wordt deze abstractie van Web Sockets in de lijst opgenomen in plaats van Web Sockets.

Dit resulteert in de volgende technologiëen voor realtime samenwerken:

* Zelf implementeren bovenop socket.io
* Zelf implementeren bovenop WebRTC
* TogetherJS gebruiken - <https://togetherjs.com>

5. Grafische Libraries

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Library** | **Vierkanten tekenen** | **Tekst in vierkant** | **Versleep-**  **baar** | **Lijnen kunnen trekken** | **Exporteren  naar bv.  JSON** | **Documentatie** | **Rang** |
| RaphaëlJS | ja | ja | ja | ja | nee | goed | **4** |
| BonsaiJS | ja | nee | ja | nee | nee | matig | **3** |
| FabricJS | ja | ja | ja | ja | ja | zeer goed | **5** |
| PaperJS | ja | nee | nee | deels | ja | matig | **2** |
| OCanvas | ja | nee | nee | deels | nee | goed | **1** |

5.1 RaphaëlJS

Zie bijlage 1.1 voor meer informatie en code voorbeelden.

**Figuren tekenen en tekst plaatsen**

RaphaëlJS library heeft eenvoudige functies met attributen waarmee je verschillende figuren mee kunt tekenen. Ook is het mogelijk om tekst te schrijven op een bepaald x- en y-positie.

**Figuren aanpassen en verbinden**

De aangemaakte figuren zijn te verslepen met de muis. De figuren kunnen hierna ook verbonden worden met een lijn die handmatig bepaald moet worden met een x- en y-positie.

**Data exporteren**

RaphaëlJS ondersteunt alleen het importeren van JSON bestanden, maar niet het exporteren ervan.

**Documentatie**

Alle functies inclusief de attributen zijn goed gedocumenteerd en uitgewerkt.

Voor bijna elke functie is een voorbeeld code geschreven om je op weg te helpen.

**Conclusie**

RaphaëlJS voldoet niet aan al de eisen. De tekening moet geëxporteerd kunnen worden naar een JSON bestand. Daarnaast is het verbinden van figuren met een lijn ook twijfelachtig, omdat de functie niet specifiek bedoeld is om figuren te verbinden.

5.2 BonsaiJS

Zie bijlage 1.2 voor meer informatie en code voorbeelden.

**Figuren tekenen en tekst plaatsen**

BonsaiJS library heeft eenvoudige functies met attributen waarmee je verschillende figuren mee kunt tekenen. Het is niet mogelijk met BonsaiJS teksten te schrijven of te plaatsen in figuren.

**Figuren aanpassen en verbinden**

BonsaiJS ondersteunt ook het verslepen van aangemaakte figuren met de muis, maar kunnen niet verbonden worden met een lijn.

**Data exporteren**

BonsaiJS ondersteunt het importeren of exporteren van een canvas naar JSON niet.

**Documentatie**

De documentatie is matig vergeleken met andere libraries. Er is geen goede inhoudsopgave gemaakt voor een snelle navigatie door de documentatie. De functies en attributen zijn wel goed uitgewerkt en voor de meeste functies is er ook demo geschreven.

**Conclusie**

BonsaiJS voldoet niet aan de eisen. Er kunnen geen teksten geplaatst worden en het verbinden van figuren is ook niet mogelijk. De tekening moet geëxporteerd kunnen worden naar een JSON bestand en dat kan ook niet.

5.3 FabricJS

Zie bijlage 1.3 voor meer informatie en code voorbeelden.

**Figuren tekenen en tekst plaatsen**

Met FabricJS is het mogelijk verschillende figuren, zoals rechthoeken en cirkels, te tekenen. Een figuur wordt aangemaakt als object en daarin kunnen alle relevante eigenschappen (positie, grootte, kleur en hoek) worden meegegeven.

Om tekst te plaatsen op het canvas heeft FabricJS een functie om een los stuk tekst aan te maken. Ook kan er een groep van objecten worden aangemaakt die behandelt wordt als één object. Hierdoor kan tekst in bijvoorbeeld een vierkant worden geplaatst.

**Figuren aanpassen en verbinden**

FabricJS heeft een ingebouwde functie om figuren door middel van slepen groter te maken of de positie of hoek aan te passen.

Om figuren te verbinden kunnen er lijnen worden aangemaakt en die kunnen worden vastgezet aan een figuur.

**Data exporteren**

De getekende figuren kunnen later opnieuw gebruikt worden door het hele canvas te exporteren naar JSON en later weer in te laden. Op deze manier kan de data makkelijk worden opgeslagen in een database.

**Documentatie**

De documentatie van FabricJS is uitgebreid en alle beschreven functies hebben voorbeeldcode.

**Conclusie**

FabricJS voldoet aan alle vereiste functionaliteiten en is makkelijk te gebruiken zonder extra benodigde functionaliteiten voor dit project handmatig te hoeven programmeren.

Er kan makkelijk mee gewerkt worden door de goede documentatie en voorbeelden en data kan makkelijk in een database worden opgeslagen. Hierdoor is deze technologie zeer geschikt om gebruikt te worden voor dit project.

5.4 PaperJS

Zie bijlage 1.4 voor meer informatie en code voorbeelden.

**Figuren tekenen en tekst plaatsen**

In PaperJS kunnen verschillende soorten figuren worden getekend. Door middel van een functie kan een figuur worden aangemaakt met een positie. Andere eigenschappen kunnen later met een andere functie worden aangepast of worden toegevoegd.

Er is een functie om tekst op het canvas te plaatsen, maar dit wordt weergegeven als afbeelding en kan niet in andere figuren worden geplaatst.

**Figuren aanpassen en verbinden**

Er zijn functies om de eigenschappen van een figuur, waaronder positie en grootte, aan te passen. Er is echter geen functionaliteit om bijvoorbeeld het figuur te verplaatsen door te slepen.

Het is mogelijk lijnen te tekenen in PaperJS, maar de manier waarop is niet waar het projectteam naar op zoek is in dit project.

**Data exporteren**

Data kan worden geëxporteerd naar JSON om het op te slaan in de database en later weer in te laden.

**Documentatie**

De documentatie is matig; niet alle functies lijken te zijn beschreven en niet overal is voorbeeldcode aanwezig.

**Conclusie**

PaperJS biedt een aantal van de functionaliteiten die nodig zijn voor dit project en voldoet daarmee aan een paar van de eisen. Wel moeten sommige functionaliteiten zelf geprogrammeerd worden bovenop de functionaliteiten die PaperJS biedt, zoals het verbinden van figuren met lijnen. Een andere functionaliteit, tekst plaatsen in figuren, is niet of niet goed mogelijk. Ook is de documentatie beperkt. Hierdoor is deze functionaliteit niet voldoende geschikt voor dit project.

5.5 oCanvas

Zie bijlage 1.5 voor meer informatie en code voorbeelden.

**Figuren tekenen en tekst plaatsen**

In oCanvas kunnen verschillende figuren worden getekend. Een figuur wordt als een object aangemaakt en kunnen de positie, grootte en kleur worden meegegeven.

Tekst plaatsen op het canvas is mogelijk door middel van een functie. Aan deze tekst kan verschillende eigenschappen worden meegegeven, maar er is geen functionaliteit die een tekst in een figuur zet.

**Figuren aanpassen en verbinden**

Er zijn functies aanwezig voor het veranderen van de positie of grootte, maar er is geen ingebouwde functionaliteit die dit door middel van slepen mogelijk maakt.

Ook is er een functie aanwezig voor het tekenen van lijnen, er is geen functie om deze vast te zetten aan een figuur.

**Data exporteren**

Data exporteren naar een bestandstype dat makkelijk in een database kan worden opgeslagen is niet mogelijk.

**Documentatie**

De functies zijn goed gedocumenteerd met voorbeeldcode.

**Conclusie**

oCanvas voldoet aan de eis om figuren te kunnen te tekenen. Verder wordt er niet aan de eis voldaan om tekst in een figuur te kunnen plaatsen en de data makkelijk te kunnen exporteren om op te slaan in een database. Ook bij deze technologie zullen er een aantal functionaliteiten zelf moeten worden geprogrammeerd. Hierdoor is ook deze technologie niet voldoende geschikt voor dit project.

5.6 Conclusie

Uit het onderzoek is gebleken dat FabricJS de beste keus is voor de projectgroep. Alle benodigde functionaliteiten zijn aanwezig en door de duidelijke documentatie is deze library het meest voor de hand liggend om te implementeren.

Alle andere onderzochte libraries missen één of meer functionaliteiten, waardoor ze niet voldoen aan onze eisen en verwachtingen van de libraries.

6. Teksteditors

In de Percolator moet er de mogelijkheid zijn om code op de webpagina te schrijven. Omdat de Percolator alleen de geheugenmodellen moet kunnen weergeven hoeft de code natuurlijk niet uitgevoerd te worden, maar het moet wel weer kunnen geven welke regel code bij het huidige geheugenmodel hoort. Hier zal dus een soort highlighting beschikbaar moeten zijn.

Een stuk code zonder opmaak en standaard letterkleur is niet te onderscheiden van een normaal stukje tekst. De teksteditor zal dus functies, statements en variabelen zichtbaar moeten onderscheiden van elkaar. Dit kan gebeuren door ze een aparte kleur of opmaak te geven.

**Requirements:**

* Structuur
  + De code wordt leesbaarder gemaakt door gebruik te maken van inspringende tekst.
* Onderscheidt gereserveerde woorden
  + Dit moet worden gedaan door gereserveerde woorden een aparte kleur te geven en/of tekstopmaak toe te passen.
* Programmeertalen
  + De Percolator zou voor meerdere programmeertalen gebruikt kunnen worden. Deze talen moeten natuurlijk wel ondersteund worden door de teksteditor.

Na een zoektocht op Google zijn de volgende teksteditors gevonden, zoals eerder al aangegeven:

* Ace - <http://ace.c9.io>
* CodeMirror - <http://codemirror.net>
* SyntaxHighlighter - <http://alexgorbatchev.com/SyntaxHighlighter/>
* Edit area - <http://www.cdolivet.com/editarea/>
* LDT - <https://github.com/kueblc/LDT/>

6.1 Overzicht

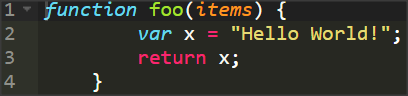
In de onderstaande tabel wordt de uitslag van de betreffende teksteditors weergegeven.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Teksteditor** | **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie (slecht, matig, goed)** | **Implementatie** |
| **Ace** | Ja, goed | Ja | Ja | 110 | goed | Zeer goed |
| **CodeMirror** | Ja | matig | Zelf instellen | 60 | goed | Onvoldoende |
| **SyntaxHighlighter** | Ja | Nee | Ja | 23 | Matig | Goed |
| **Edit Area** | Ja | Ja | Ja | 19 | matig | goed |
| **LDT** | Gedeeltelijk | Nee | Nee | Onbekend | Redelijk | Goed |

6.2 Ace

Samenvatting:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie**  **(slecht, matig, goed)** | **Implementatie** |
| Ja, goed | Ja | Ja | 110 | goed | Zeer goed |



**Structuur:**

Zoals op bovenstaande foto is te zien, zorgt Ace voor een gestructureerde weergave van de code. Er is duidelijk inspringing toegepast.

**Onderscheidt gereserveerde woorden:**

Zoals op bovenstaande foto is te zien, hebben de woorden verschillende kleuren. Op deze manier is het gemakkelijk om functies, variabelen en gereserveerde woorden van elkaar te onderscheiden.

**Aantal programmeertalen:**

Ace ondersteunt maar liefst 110 verschillende programmeertalen (waaronder Java en JavaScript). Echter, in de broncode moet worden aangegeven welke programmeertaal op de webpagina ondersteunt wordt. Er zal dus een functie moeten worden geschreven waarmee je de programmeertaal aan kunt passen.

**Documentatie:**

Alle mogelijke functies zijn gedocumenteerd. Hier staan echter geen voorbeelden bij. Enkel een korte omschrijving (1-2 zinnen) bij elke methode moet voor verduidelijking zorgen.

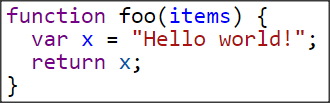
**Implementatie:**

De implementatie van Ace is erg gemakkelijk. Op de website van Ace staat de juiste informatie om Ace te implementeren. Dit proces vergt slechts enkele minuten.

6.3 CodeMirror

Samenvatting:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie**  **(slecht, matig, goed)** | **Implementatie** |
| Ja | matig | Zelf instellen | 60 | goed | Onvoldoende |



**Structuur:**

In de bovenstaande afbeelding is te zien dat CodeMirror structuur toe past. Er wordt rekening gehouden met inspringing.

**Onderscheidt gereserveerde woorden:**

Op de bovenstaande afbeelding is te zien dat CodeMirror inderdaad gereserveerde woorden onderscheidt door ze een andere kleur te geven. Helaas zijn de kleuren erg eentonig.

**Aantal programmeertalen:**

Op de website van CodeMirror wordt aangegeven dat CodeMirror meer dan zestig programmeertalen ondersteund. Dit is, net als bij Ace, niet dynamisch te selecteren. Hier voor zal dus ook een functie gebouwd moeten worden.

**Documentatie:**

In eerste instantie is de documentatie van CodeMirror lastig te vinden. Op de website staat niet duidelijk aangegeven waar het te vinden is. Wat er in de documentatie beschreven staat, is echter erg duidelijk. De beschreven functies en mogelijkheden worden duidelijk uitgelegd.

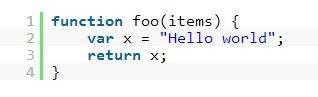
**Implementatie:**

De implementatie van CodeMirror is geen gemakkelijke klus. Omdat de documentatie lastig is, is het niet duidelijk hoe CodeMirror geïmplementeerd moet worden. Hier zal dus de nodige tijd in gestoken moeten worden.

**6.4 SyntaxHighlighter**

Samenvatting:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie**  **(slecht, matig, goed)** | **Implementatie** |
| Ja | Nee | Ja | 23 | Matig | Goed |



**Structuur:**

In de bovenstaande afbeelding zie je duidelijk dat SyntaxHighlighter structuur toe past. Er wordt ook rekening gehouden met inspringing.

**Onderscheidt gereserveerde woorden:**

SyntaxHighlighter maakt geen onderscheid in gereserveerde woorden. Deze gereserveerde woorden worden als variabele herkent.

**Aantal programmeertalen:**

SyntaxHighlighter heeft zelf 23 talen geïmplementeerd in hun code editor. Dit zijn voornamelijk de populairste talen op dit moment. Mocht de taal niet worden ondersteund door SyntaxHighlighter dan geven ze als optie om deze op te zoeken in een door gebruikers gemaakte lijst van talen. Mocht hij daar nog steeds niet in staan dan kun je deze taal toevoegen aan de “wishlist”.

**Documentatie:**

De documentatie van SyntaxHighlighter is matig. Er is een goede pagina met instructies hoe deze te implementeren valt. Er is alleen minder documentatie over hoe de SyntaxHighlighter nu precies werkt. Er is één pagina die dat heel mager uitlegt.

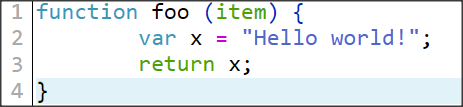
**Implementatie:**

De implementatie van SyntaxHighlighter is makkelijk. SyntaxHighlighter is snel te gebruiken door de al gemaakte test file die wordt meegeleverd.

6.5 Edit Area

Samenvatting:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie**  **(slecht, matig, goed)** | **Implementatie** |
| Ja | Ja | Ja | 19 | matig | goed |



**Structuur:**

Zoals te zien is in de afbeelding, past ook Edit Area structuur toe aan de code. Hier wordt ook gebruik gemaakt van inspringing.

**Onderscheidt gereserveerde woorden:**

Edit Area onderscheidt gereserveerde woorden door gebruik te maken van verschillende tekstkleuren.

**Aantal programmeertalen:**

Edit Area heeft negentien verschillende programmeertalen geïmplementeerd. Deze zijn niet dynamisch. Er zal voor Edit Area een functie moeten worden gemaakt die dit mogelijk maakt..

**Documentatie:**

Op de website van Edit Area is de documentatie te vinden. De documentatie is wat lastiger te vinden door de onduidelijke indeling van de pagina (geen opmaak). Verder is wat er wordt beschreven in de documentatie duidelijk geformuleerd. Er is echter geen structuur of een inhoudsopgave aanwezig. Ook is de documentatie verspreidt over meerdere pagina’s die niet aan elkaar gelinkt zijn.

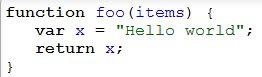
**Implementatie:**

De implementatie van Edit Area is gemakkelijk. Door een paar regels code aan de HTML toe te voegen is Edit Area klaar voor gebruik. De betreffende code is gemakkelijk op de website van Edit Area te vinden.

6.6 LDT (Lightweight Decorator for Textareas)

Samenvatting:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Past structuur toe** | **Onderscheidt gereserveerde woorden** | **Regelnummering** | **Aantal programmeertalen** | **Documentatie**  **(slecht, matig, goed)** | **Implementatie** |
| Gedeeltelijk | Nee | Nee | Onbekend | Redelijk | Goed |



**Structuur:**

LDT past gedeeltelijk de structuur aan van de code. LDT past wel onderscheid door middel van verschillende kleuren toe, maar springt niet in. Hierdoor moeten er steeds spaties worden gezet om de code overzichtelijk te houden.

**Onderscheidt gereserveerde woorden:**

LDT onderscheidt geen gereserveerde woorden met een andere kleur. Gereserveerde woorden worden als variabelen gemarkeerd.

**Aantal programmeertalen:**

Er is niet beschreven hoeveel programmeertalen LDT ondersteunt. Door middel van testen komt boven water dat alleen variabelen en keywords in de syntax van programmeertalen worden gekleurd.

**Documentatie:**

Er is alleen een README.md beschikbaar op een GitHub (Finley, 2012) pagina. In deze README staat informatie hoe je de editor kan installeren en staat er een korte API beschreven met verschillende functies.

**Implementatie:**

Implementatie van LDT is eenvoudig. Door de code in een <textarea> HTML tag te zetten wordt de editor in werking gezet.

6.7 Conclusie

Na het testen van de vijf genoemde tekst editors zijn we tot de conclusie gekomen dat Ace het meest aan de eisen voldoet. Ace zorgt voor een goed gestructureerde code waarbij rekening wordt gehouden met het onderscheiden van gereserveerde woorden. Ook geeft Ace standaard een duidelijke regelnummering weer. Van alle tekst editors ondersteunt Ace het grootste aantal programmeertalen. Hierdoor zal het eenvoudig zijn om met de Percolator meerdere programmeertalen te kunnen ondersteunen. De documentatie van Ace is overzichtelijk en gemakkelijk te vinden. De implementatie van Ace is de eenvoudigste methode van de vijf genoemde editors ervaren.

7. Realtime samenwerken

De technologieën worden beoordeeld op de volgende drie criteria:

* Merkbare latentie: wanneer de merkbare latentie zo laag mogelijk is zal de gebruikerservaring beter zijn.
* Makkelijk te implementeren: als de technologie makkelijk te implementeren is, kost het voor de projectgroep minder tijd om het te implementeren.
* Browser compatibiliteit: als een bepaalde technologie met meer browsers werkt, kan deze technologie meer gebruikers van dienst zijn.

|  |  |  |  |
| --- | --- | --- | --- |
| **Technologie** | **Latentie (rang, lager is beter)** | **Implementatie (rang, lager is beter)** | **Browser compatibiliteit** |
| **Socket.IO** | #2 | #2 | Meer dan 84.4% |
| **WebRTC** | #3 | #3 | 55.44% |
| **TogetherJS** | #1 | #1 | 84.4% |

7.1 Zelf implementeren bovenop Socket.IO

1. Merkbare latentie:

Latentie is afhankelijk van de bandbreedte en vertraging van de verbinding tussen client en server. Het is mogelijk zelf kleine verbeteringen toe te passen, zoals meerdere events bundelen en versturen in één keer.

1. Makkelijk te implementeren:

Het is lastig om klikken, muis slepen en typen tussen verschillende gebruikers te delen en te bouwen vanaf scratch. Ook zul je zelf disconnects moeten afvangen.

1. Browser compatibiliteit:

Socket.IO is gebaseerd op Web Sockets. Web Sockets wordt ondersteund door 84.4% (Alexis Deveria, 2014) van alle browsers. Wanneer Web Sockets niet worden ondersteund, valt Socket.IO terug op Flash sockets, JSONP polling of AJAX long polling. Dit maakt dat elke browser die het zou kunnen ondersteunen wordt ondersteund.

7.2 Zelf implementeren bovenop WebRTC

1. Merkbare latentie:

Omdat WebRTC een peer to peer protocol is, ben je afhankelijk van de internetsnelheid van de gebruikers om een snelle ervaring te kunnen leveren. De kans is erg groot dat een gebruiker een minder snelle internetverbinding heeft dan een server in een datacenter. Wanneer een client een verandering aanbrengt zal hij deze moeten sturen naar alle andere clients, wat zorgt voor hogere eisen voor de clients dan in een client-server model.

1. Makkelijk te implementeren:

WebRTC is gemaakt voor peer to peer (client naar client) communicatie, welke lastiger te implementeren is dan een client-server model. Daarnaast moet alles dat bij Socket.IO moet worden geïmplementeerd hier ook worden gedaan.

1. Browser compatibiliteit:

WebRTC wordt ondersteund door 55.44% (Alexis Deveria, 2014) van alle browsers. Bij desktop gebruikers vallen Internet Explorer en Safari buiten de boot.

7.3 TogetherJS gebruiken

1. Merkbare latentie:

Net iets minder latentie dan Socket.IO omdat het native Web Sockets gebruikt. Je kunt instellen dat alleen clicks op het canvas verwerkt worden. Dit heeft minder netwerkverkeer als resultaat, wat de latentie ten goede komt.

1. Makkelijk te implementeren:

TogetherJS heeft het realtime samenwerken al ingebouwd. (Voice)chat, klikken en typen wordt gedeeld met andere gebruikers. TogetherJS moet wel een beetje aangepast worden wil je events als bijvoorbeeld je muis slepen gebruiken.

1. Browser compatibiliteit:

Web Sockets worden ondersteund door 84.4% (Alexis Deveria, 2014) van alle browsers. WebRTC wordt ondersteund door 55.44% (Alexis Deveria, 2014) van alle browsers. Wanneer WebRTC niet ondersteund wordt werkt alleen chatten niet, dit is dus geen groot probleem. Bij desktop gebruikers vallen bij het Web Sockets gedeelte IE9 en lager buiten de boot.

7.4 Conclusie

Socket.IO werkt met de meeste browsers (Web Sockets werkt met meer browsers dan WebRTC, Socket.IO gebruikt Web Sockets maar valt terug op andere technologieën als Web Sockets niet beschikbaar is), TogetherJS is veruit het makkelijkst te implementeren en TogetherJS heeft de minste merkbare latentie.

TogetherJS komt als beste uit het onderzoek want:

* Minder merkbare latentie dan WebRTC en Socket.IO.
* Socket.IO werkt wel met meer browsers, maar TogetherJS werkt met 84.4%, waaronder alle moderne desktop browsers.

Het feit dat TogetherJS makkelijker te implementeren is dan WebRTC en Socket.IO is voor de projectgroep het belangrijkste.

8. Bijlagen

1. Onderzochte grafische libraries
   1. RaphaëlJS

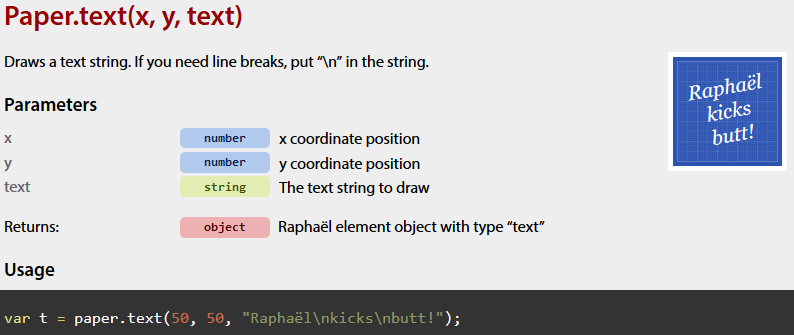
* Vierkanten tekenen: ja
  1. Deze library heeft eenvoudige functies met attributen waarmee je verschillende vormen kunt tekenen.
  2. Voorbeeld om een cirkel te tekenen:

|  |
| --- |
| // Creates canvas 320 × 200 at 10, 50 **var** paper = Raphael(10, 50, 320, 200);  // Creates circle at x = 50, y = 40, with radius 10 **var** circle = paper.circle(50, 40, 10); // Sets the fill attribute of the circle to red (#f00) circle.attr(*"fill"*, *"#f00"*);  // Sets the stroke attribute of the circle to white circle.attr(*"stroke"*, *"#fff"*); |

* Tekst in vierkant plaatsen: ja

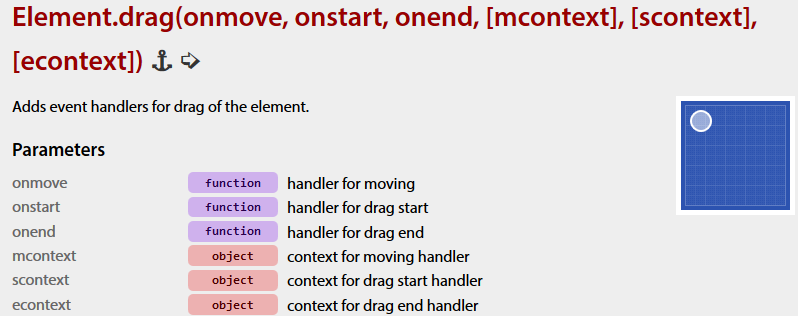
1. Met deze functie kun je tekst schrijven op een bepaald x- en y-positie.
2. Voorbeeld om tekst te plaatsen:

|  |
| --- |
| **var** t = paper.text(50, 50, *"Raphaël\nkicks\nbutt!"*); |



* Versleepbaar: ja

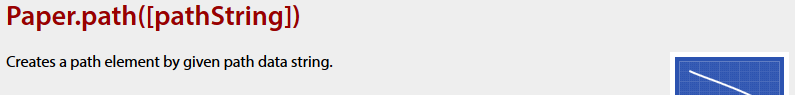
1. Met deze functie kun je aangemaakte vormen verslepen met de muis.



* Lijnen kunnen trekken: twijfel

1. Met deze functie kun je lijnen tekenen op een bepaald x- en y-positie. Je zou dit kunnen gebruiken om handmatig vormen te verbinden.
2. Voorbeeld om een lijn te tekenen:

|  |
| --- |
| **var** c = paper.path(*"M10 10L90 90"*); // draw a diagonal line: // move to 10,10, line to 90,90 |



* Exporteren naar JSON: nee
* Documentatie: zeer goed

1. Alle functies inclusief de attributen zijn goed gedocumenteerd en uitgewerkt.
2. Voor bijna elke functie is een voorbeeld code geschreven om je op weg te helpen.
   1. BonsaiJS

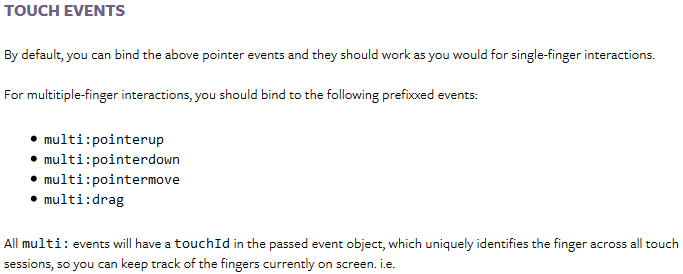
* Vierkanten tekenen: ja

1. Deze library heeft verschillende functies met attributen waarmee je allerlei vormen mee kunt tekenen.
2. Voorbeeld om een vierkant te tekenen:

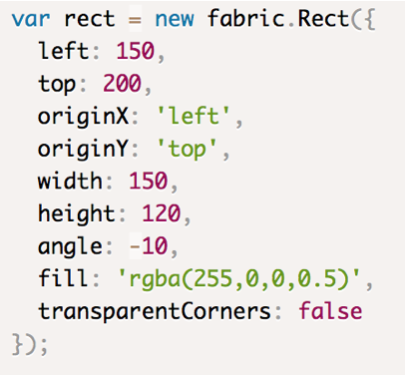
|  |
| --- |
| **new** Rect(x, y, width, height, [cornerRadius]) |

* Tekst in vierkant plaatsen: nee
* Versleepbaar: ja

1. Met deze functie kun je aangemaakte vormen verslepen met de muis.



* Lijnen kunnen trekken: nee
* Exporteren naar JSON: nee
* Documentatie: matig
  1. FabricJS
* Figuren tekenen ja
  1. Deze library heeft functies om verschillende figuren als objecten aan te maken
  2. Het onderstaande stuk code is een simpel voorbeeld van een manier om een figuur, in dit geval een vierkant, aan te maken in FabricJS. Hierin kunnen eigenschappen zoals de positie, grootte, hoek en kleur worden meegegeven, zie afbeelding f1 (FabricJS, Controls customization demo).



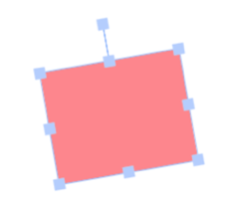
Afbeelding f1

* Tekst plaatsen in de figuren ja

1. FabricJS beschikt over een aantal functies om tekst aan het canvas toe te voegen en deze vorm te geven doormiddel van eigenschappen
2. In FabricJS is het ook mogelijk een groep van FabricJS objecten aan te maken als één entiteit. Op die manier is het mogelijk om een figuur met tekst erin te behandelen als één object (FabricJS, Introduction to Fabric.js).

* Positie en grootte van figuren kunnen aanpassen ja

1. Getekende figuren kunnen worden versleept en geroteerd en de grootte is aan te passen
2. Door met de muis in het figuur te klikken en vervolgens te slepen, kan het figuur worden verplaatst. Door in een hoek te klikken en te slepen, wordt de grootte aangepast. Als laatste kan de hoek worden aangepast zoals te zien is in afbeelding f2.



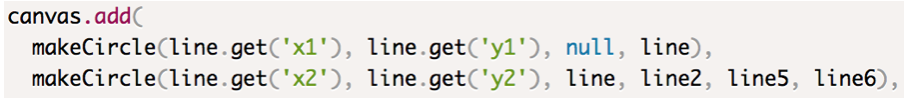
Afbeelding f3

* Figuren verbinden door middel van lijnen en pijlen ja

1. Het is mogelijk om in FabricJS lijnen te tekenen en deze vast te maken aan andere figuren.
2. Door eerst een lijn aan te maken en vervolgens figuren te tekenen op de x en y coördinaten van de lijn kunnen figuren worden verbonden zoals in afbeeldingen f3 en f4 (FabricJS, Stickman demo) te zien is.



Afbeelding f3



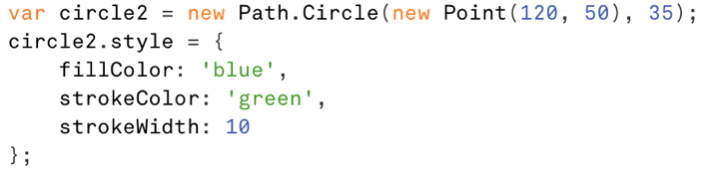
Afbeelding f4

* Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) ja

1. De data van alle getekende elementen is te exporteren naar JSON.
2. Met de methode canvas.toJSON() worden alle elementen op het canvas in een JSON object gestopt. Met de methode canvas.loadFromJSON() is deze data vervolgens weer in te laden (FabricJS, Introduction to Fabric.js).

* Goede documentatie uitstekend
  1. FabricJS heeft een uitgebreide documentatie met voorbeeldcode
  2. Er zijn veel functies om eigenschappen van getekende figuren op te vragen en bij te werken.
  3. PaperJS
* Figuren tekenen ja

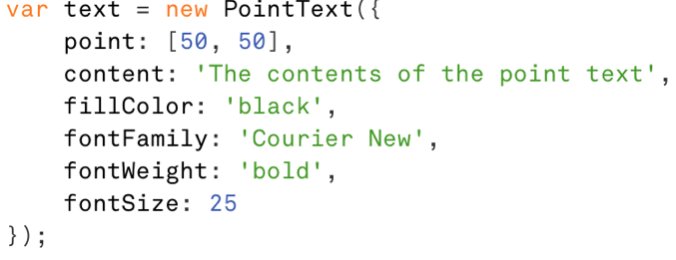
1. Het is mogelijk om verschillende soorten figuren, zoals cirkels te tekenen
2. Onderstaand stukje code (afbeelding p1) geeft een voorbeeld van het tekenen van een cirkel



Afbeelding p1

* Tekst plaatsen in de figuren nee

1. Het plaatsen van tekst op het canvas is mogelijk (volgens PaperJS documentatie), maar de tekst wordt als afbeelding op het scherm weergegeven
2. In afbeelding p2 wordt een PointText aangemaakt



Afbeelding p2

* Positie en grootte van figuren kunnen aanpassen gedeeltelijk

1. Het is mogelijk figuren te verplaatsen, maar dat moet gebeuren via functies. Er is geen ingebouwde functionaliteit voor het verslepen van figuren (PaperJS, Transforming Items).

* Figuren verbinden door middel van lijnen en pijlen gedeeltelijk

1. Het is in PaperJS mogelijk om lijnen/pijlen te tekenen, maar dit moet op een ingewikkelde manier en er is geen ingebouwde functionaliteit voor het vastmaken van pijlen aan andere figuren. Het tekenen van deze lijnen is ook eigenlijk voor wiskundige doeleinden (PaperJS, Vector Geometry).
2. In afbeelding p3 wordt een voorbeeld getoond



Afbeelding p3

* Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) ja

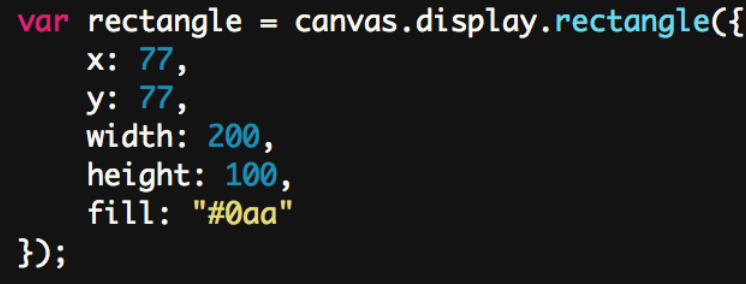
1. Via de functies exportJSON() en importJSON() is het mogelijk de data van alle objecten op het canvas te exporteren of te importeren (PaperJS, Importing / Exporting JSON and SVG).

* Goede documentatie matig

1. Er is documentatie aanwezig, er worden echter maar weinig functies in genoemd
2. Niet bij alle beschreven functies is voorbeeldcode aanwezig
   1. oCanvas

* Figuren tekenen ja

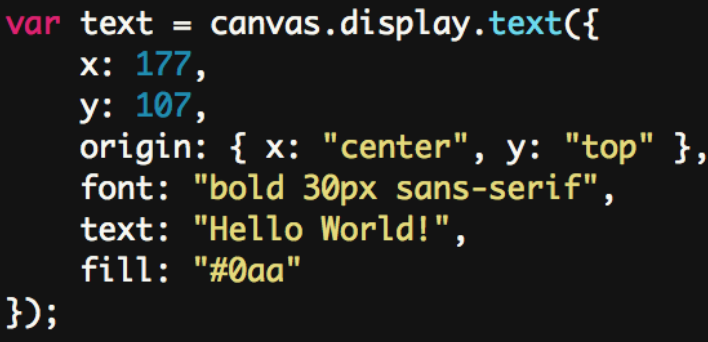
1. In oCanvas is het mogelijk figuren zoals vierkanten te tekenen (volgens oCanvas documentatie)
2. Afbeelding o1 toont een voorbeeld waarin een rechthoek wordt aangemaakt



Afbeelding o1

* Tekst plaatsen in de figuren nee

1. Er zijn functies aanwezig om tekst op het canvas te plaatsen, maar niet om deze in of op een ander object te plaatsen (volgens oCanvas documentatie).
2. In afbeelding o2 wordt een nieuw text object aangemaakt en er worden een aantal eigenschappen meegegeven



Afbeelding o2

* Positie en grootte van figuren kunnen aanpassen gedeeltelijk

1. Het is mogelijk figuren te verplaatsen, maar dat moet gebeuren via functies. Er is geen ingebouwde functionaliteit voor het verslepen van figuren.

* Figuren verbinden door middel van lijnen en pijlen gedeeltelijk

1. Het is via een functie mogelijk om lijnen te tekenen, er is geen ingebouwde functie voor het vastzetten van een lijn aan een figuur.
2. Onderstaande afbeelding (afbeelding o3), toont een voorbeeld waarin een lijn wordt getekend



Afbeelding o3

* Alle data die nodig is om het gene dat getekend is later opnieuw te kunnen laten tekenen moet kunnen worden geëxporteerd naar een formaat dat makkelijk in een database kan worden opgeslagen. (Bijvoorbeeld JSON) Nee

1. Er is geen functionaliteit gevonden voor het exporteren naar een bestandstype dat makkelijk in een database kan worden opgeslagen

* Goede documentatie Ja

De functies die er zijn, zijn gedocumenteerd met voorbeeldcode

9. Bibliografie

* hagerAly. (2013, november 3). *Real Time Web Development - Brief*. Opgeroepen op 11 november, 2014, van Code project: <http://www.codeproject.com/Tips/677819/Real-Time-Web-Development-Brief>
* Alexis Deveria. (2014, oktober). *Can I use websockets?*. Opgeroepen op 14 november, 2014, van caniuse.com: [http://caniuse.com/#search=websockets](http://caniuse.com/%23search=websockets)
* Alexis Deveria. (2014, oktober). *Can I use webrtc?*. Opgeroepen op 14 november, 2014, van caniuse.com: [http://caniuse.com/#search=webrtc](http://caniuse.com/%23search=webrtc)
* FabricJS. (geen datum). *Fabric.js demos · Stickman*. Geraadpleegd op 14 november 2014, van [*http://fabricjs.com/stickman/*](http://fabricjs.com/stickman/)
* FabricJS. (geen datum). *Fabric.js demos · Controls customization*. Geraadpleegd op 14 november 2014, van [*http://fabricjs.com/controls-customization/*](http://www.fabricjs.com/controls-customization/)
* FabricJS. (geen datum). *Introduction to Fabric.js. Part 3.* Geraadpleegd op 14 november 2014, van<http://fabricjs.com/fabric-intro-part-3/>
* FabricJS. (geen datum). *FabricJS Docs.* Geraadpleegd op 14 november 2014, van<http://fabricjs.com/docs/>
* PaperJS. (geen datum). *Transforming Items.* Geraadpleegd op 14 november 2014, van [*http://paperjs.org/tutorials/project-items/transforming-items/*](http://paperjs.org/tutorials/project-items/transforming-items/)
* PaperJS. (geen datum). *PointText.* Geraadpleegd op 14 november 2014, van [*http://paperjs.org/reference/pointtext/*](http://paperjs.org/reference/pointtext/)
* PaperJS. (geen datum). *Vector Geometry.* Geraadpleegd op 14 november 2014, van [*http://paperjs.org/tutorials/geometry/vector-geometry/*](http://paperjs.org/tutorials/geometry/vector-geometry/)
* *oCanvas. (geen datum). Text. Geraadpleegd op 14 november 2014, van* [*http://ocanvas.org/docs/Display-Objects/Text*](http://ocanvas.org/docs/Display-Objects/Text)
* *oCanvas. (geen datum). Rectangle.* Geraadpleegd op 14 november 2014, van<http://ocanvas.org/docs/Display-Objects/Rectangle>
* *oCanvas. (geen datum). Line.* Geraadpleegd op 14 november 2014, van<http://ocanvas.org/docs/Display-Objects/Line>
* *oCanvas. (geen datum). Documentation.* Geraadpleegd op 14 november 2014, van<http://ocanvas.org/docs>
* Finley, K. (2012, July 14). *What Exactly Is GitHub Anyway?* Opgeroepen op November 18, 2014, van TechCrunch: <http://techcrunch.com/2012/07/14/what-exactly-is-github-anyway/>
* Ace (geen datum). Geraadpleegd op 14 november 2014. <http://ace.c9.io>
* CodeMirror(geen datum). Geraadpleegd op 17 november 2014. <http://codemirror.net>
* SyntaxHighlighter(geen datum). Geraadpleegd op 18 november 2014. <http://alexgorbatchev.com/SyntaxHighlighter/>
* Edit Area(geen datum). Geraadpleegd op 18 november 2014. <http://www.cdolivet.com/editarea/>
* LDT(2012, ebruari 29). Geraadpleegd op 18 november 2014. <https://github.com/kueblc/LDT/>