Practicum Client Technology (91732)

Vakcode : ICT.SE.CLNT.V17 (ICT.SE.CLNT.V16) (t1)

Datum : woensdag 27 februari 2019

Tijd : 14.30 - 16.30 uur

|  |  |  |
| --- | --- | --- |
| Klas: | Lokaal: | Aantal: |
| ICTSE2a,ICTSE2b,ICTSE2c,ICTSE2d,ICTSE2e | D1.84, D1.90, D2.08, D2.88 | 170 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Opgesteld door : Bolt, Ernst

Docenten : BTE08

Gecontroleerd door : ??

Rekenmachine : geen

Literatuur : Internet

**Niet toegestaan is het gebruik van telefoons, social media, forums, dropbox en**

**alles wat je in contact met andere personen kan brengen. Als dit toch gebeurt,**

**is er sprake van fraude en wordt dit aangegeven bij de examencommissie.**

Overige hulpmiddelen : geen

Tentamen maken op : laptop

Opgaven inleveren : ja

CONTROLEER VOORAF DE VOLGENDE GEGEVENS:

Dit tentamen bevat:

4 opgaven

#3# genummerde pagina's

Waarschuw de surveillant als één van deze aantallen niet klopt!

|  |  |  |  |
| --- | --- | --- | --- |
| Studentnummer | Naam | Klas | Cijfer |
|  |  |  |  |
| Tijd van inleveren: |  | | |

Aanwijzingen:

In totaal zijn voor de vragen 46 punten te halen. Het eindcijfer wordt verkregen door de behaalde punten te delen door 4,6. Het laagst te behalen cijfer is een 1.

**Inleverinstructie:**

1) Verwijder de map node\_modules. Indien niet verwijderd, dan wordt je practicum niet nagekeken.

2) Lever het practicum in als zip-file op de ELO bij het inleverpunt van het practicum.

De zipfile heeft als naam je klas (a,b,c, of d), naam en studentnummer:

<klas naam studentnummer>.zip

Bijvoorbeeld:

ictse2a ErnstBolt s1060639.zip

Instructie over package.json en dependencies:

Je mag **geen wijzigingen** aanbrengen in de **package.json** en de **package.json.lock** bestanden die zich bevinden de verstrekte bestanden. Het is **niet toegestaan** om **extra dependencies** te installeren.

**Startinstructie**

Op de ELO staat een map **Week 4 Practicum**. Download uit deze map het .zip-bestand en pak dit bestand uit in een lege map. Installeer de packages met $ npm install

# Opgave 1 Gulp (9 punten)

1. In de root van het project staat een gulpfile.js. Voeg drie gulp-taken: html,js en css en laat de taken uitvoeren in een build taak.

**Tip:** Je mag een watch-taak opnemen en Browsersync gebruiken.

1. Verplaats met de html-taak de html-bestanden in de src map naar de distributiemap.
2. De js-taak voert twee taken uit: 1) concateneer de js-bestanden in de vendor-map en de src-map tot één bestand. 2) sla het geconcateneerde bestand op in de distributie (dist) map. De bestandsnaam is app.js.
3. Ook de css-taak voert twee taken uit: 1) concateneer de css-bestanden in de vendor-map en de src-map tot één bestand 2) sla het geconcateneerde bestand op in de distributie (dist) map. De bestandsnaam is style.css.

# Opgave 2 HTML & CSS (11 punten)

In dit practicum maak je een eenvoudige chat-library en een SPA. Er is sourcecoude beschikbaar:

chat-library javascript vendor/chatLib.js

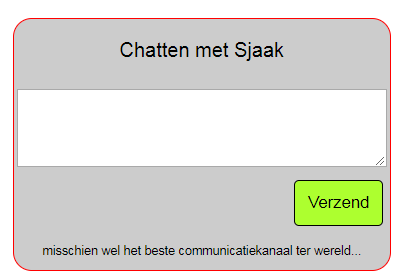
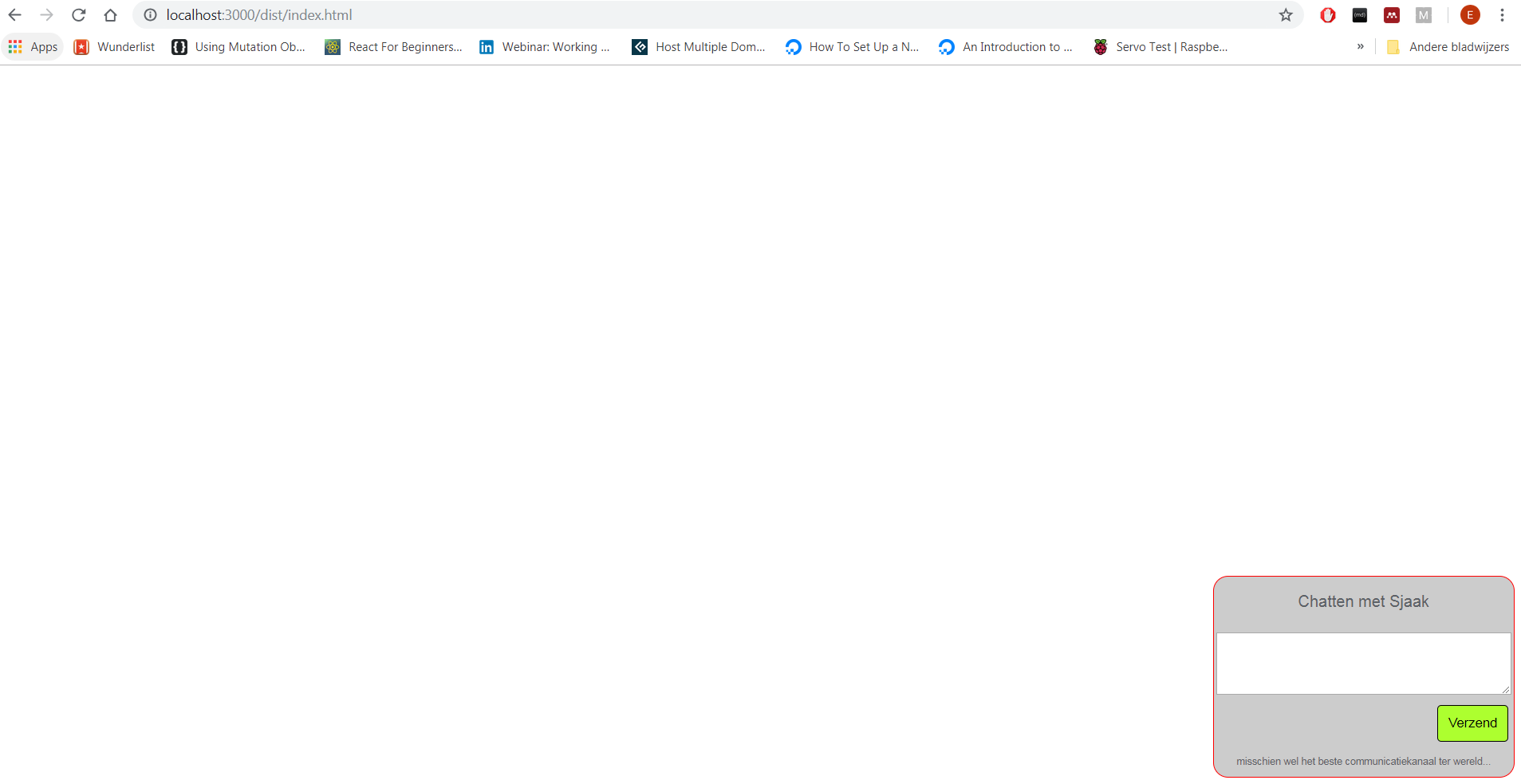
SPA javascript src/app.js

SPA css src/chat.css

homepage html src/index.html

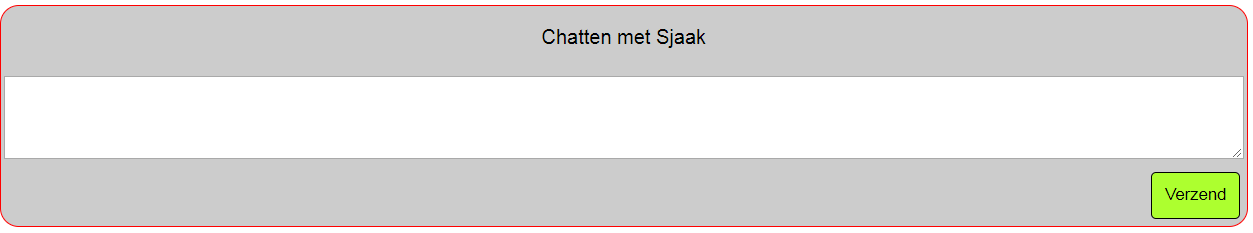
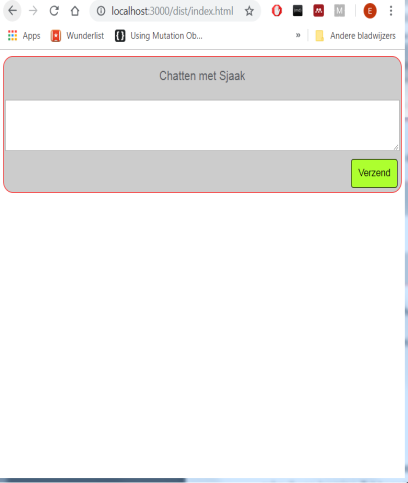
testpagina html/js test/test.html

1. Geef de bestaande html-tags in src/index.html van de chatbox passende semantische tags. **Je mag de id’s niet wijzigen!**
2. Style de chatbox zodat die overkomt met onderstaande afbeelding:

1. Voeg responive design toe aan de chatbox volgens onderstaande afbeelding.

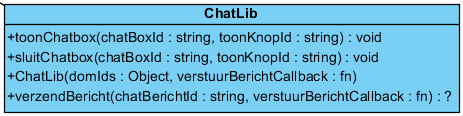
Bij een smal scherm is de footer niet zichtbaar. De header en invoerveld hebben een breedte van 100%.



1. Geef de chatbox een fade-in animatie met css.

# Opgave 3 OO en prototype (14 punten)

1. Klasse ChatLib heeft een eigen javascript file heeft in de map vendor: chatLib.js. Maak in het bestand een klasse ChatLib. Deze klasse is beschikbaar in de global scope. Geef de klasse de constructor en de methoden die staan in onderstaande diagram. De constructor gooit een error als één van de parameters niet meegegeven is, er wordt geen controle gedaan op de juistheid van de meegegeven parameters. Sla de parameters van de constructor op in lokale variabelen. Voeg de methoden toe als prototype methoden.



1. Er is in het project een werkende testpagina test/test.html aanwezig, met werkende testsuites. Gebruik deze testsuites om je testen toe te voegen. De testpagina maakt gebruik van Jasmine. Er staat een testsuite (die beginnen met de functieaanroep describe) klaar. Schrijf een test in de testsuite chat-lib die controleert of de klasse ChatLib beschikbaar (defined) is.
2. De knop ‘Toon chat’ is na het laden van de pagina in de browser zichtbaar. Zodra op de knop wordt geklikt verschijnt de chatbox en de knop ‘Toon chat’ is dan verborgen.

Implementeer deze functionaliteiten in de methode toonChatbox:

* 1. Verberg de toonKnop
  2. De chatBox wordt getoond

1. De chatbox heeft een knop waarmee het dialoog-scherm wordt gesloten, de sluit-knop. Zodra op de knop wordt geklikt wordt de chatbox verborgen en de knop ‘Toon chat’ getoond.

Implementeer de volgende functionaliteiten in de methode sluitChatbox:

* 1. Toon de toonKnop
  2. De chatBox wordt gesloten
  3. Schrijf twee testen: 1) een test die de zichtbaarheid van de sluitKnop controleert, 2) een test die de zichtbaarheid van de chatBox controleert.

1. De klasse ChatLib heeft een methode verzendBericht. Implementeer de volgende functionaliteit in de methode:
   1. De methode haalt het tekstbericht uit het tekstveld van de chatbox
   2. De methode leegt het tekstveld van de checkbox
   3. Het tekstbericht wordt meegegeven aan de callbackfunctie
   4. De methode retourneert terwijl het de functie aanroept: return verzendBericht(<bericht>). De methode weet dus niet wat het retourneert.
2. Koppel in de constructor eventhandlers (ChatLib toonChatbox etcetera) aan de toon-chat knop, de sluit-knop en de verzend-knop. De constructor ontvangt de volgende id’s van de html:

{

toonKnop: 'toon-chat',

chatBox: 'chat-box',

sluitKnop: 'sluit-chat',

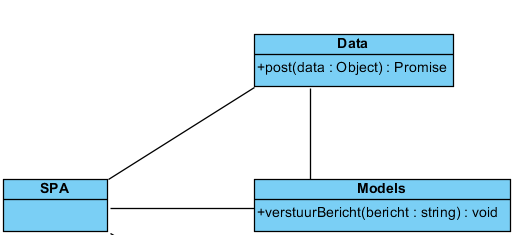
chatBericht: 'chat-bericht',

verzendKnop: 'verzend-bericht',

}

# Opgave 4 Module pattern, Promise en AJAX (12 punten)

1. De SPA-applicatie bestaat uit drie modules. Van de modules is alleen de SPA module in de global scope beschikbaar. Maak een architectuur volgens onderstaande afbeelding in bestand src/app.js. De methoden worden later geïmplementeerd, maar moeten wel al aanwezig zijn.



1. De module Data heeft een methode post. Verstuur in deze methode berichten naar de server door middel van een AJAX-request. Gebruik een willekeurige url in het AJAX-request, bijvoorbeeld: ‘abc’. De methode retourneert een Promise en resolved de response body. Als er sprake is van een error/fout dan gooit de methode een error met de melding: ‘Er is een fout opgetreden in de data module in de methode post’.
2. Geef de module Models een methode verzendBericht, die het bericht verstuurt via de post methode van de module Data.
3. Schrijf een test voor de verzendBericht methode van de module Models module om te controleren of succesvolle requests goed worden afgehandeld. Maak gebruik van een mock van het AJAX request. Geef de mock de volgende responseText: message received
4. Geef de module SPA een publieke init methode waarin een instantie van de ChatLib wordt gemaakt. Geef de ChatLib constructor de volgende waarden mee:

**Parameter domIds:**

{

toonKnop: 'toon-chat',

chatBox: 'chat-box',

sluitKnop: 'sluit-chat',

chatBericht: 'chat-bericht',

verzendKnop: 'verzend-bericht',

}

**Parameter verstuurBerichtCallback:**

de verzendBericht methode van de module Models.

**Einde Tentamen**

Vergeet niet je naam, klas en studentnummer op iedere pagina te vermelden