challenge09.md

Procedure

Please follow the instructions below and use the provided Github template for this week's assignment. You will need to upload your code after challenge 10.

You can download the instructions as pdf here.

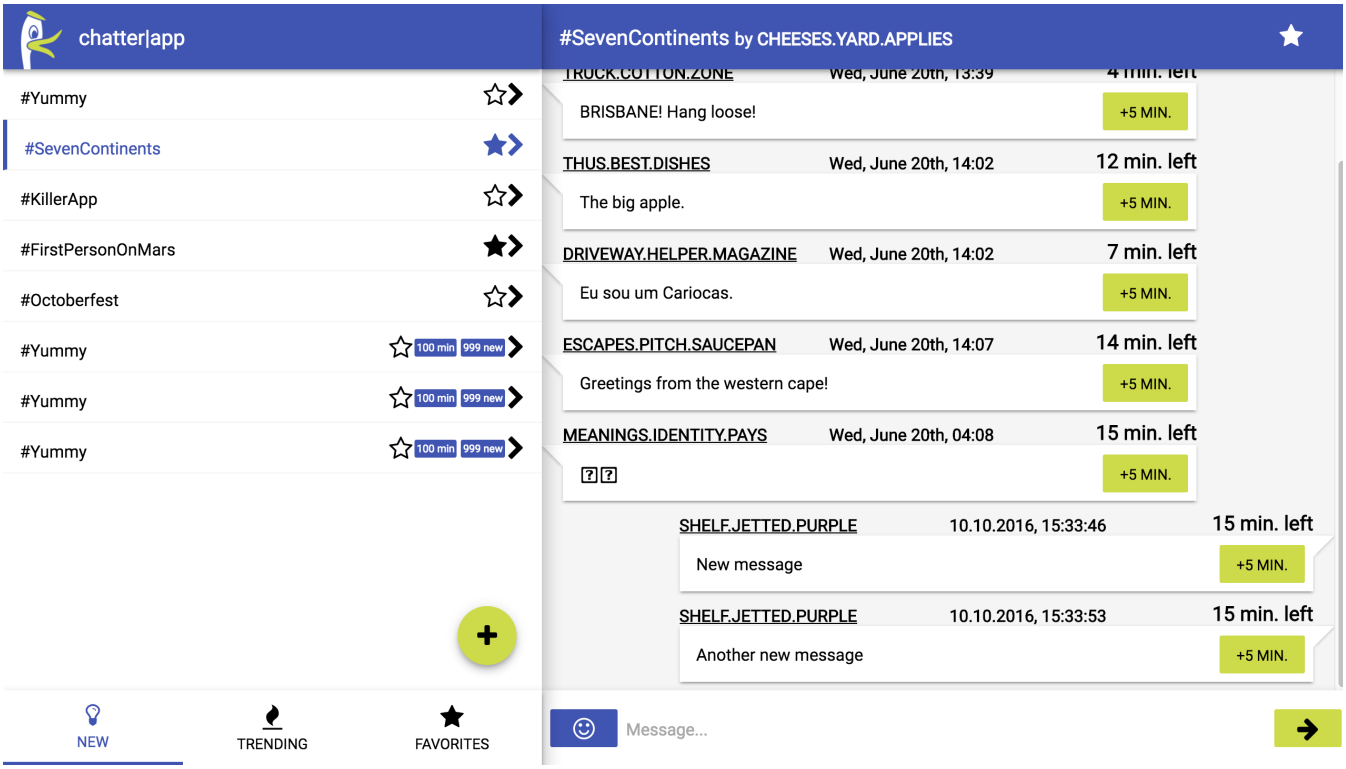
There are different exercise types:

- These exercises are important and you should tackle them.
- (*) These asterisk-marked exercises are a bit more difficult and thus voluntary. They will improve your skills, tackle them only if you want an extra challenge. However, your app will also work without fulfilling the instruction.

Prioritize the challenges. Don't spend too much time on the voluntary challenges!

Challenge Goal

This challenge will make your app shine:



Graded Criteria

`#btns#clss#sml#snd#acc` Both material design button classes have been applied with all properties, each to at least one element.

- 1 Pt. button.primary has been defined and applied to at least one button.
- 1 Pt. button.accent has been defined and applied to at least one button.

`#shdw#elv` Required shadow classes have been defined and applied to one of the required elements.

- 1 Pt. Material design shadow classes have been defined.
- 1 Pt. At least one shadow class has been successfully applied.

`#flt#btn` A floating action button (FAB) has been created complying with material design standards.

- 1 Pt. A round button has been created in CSS.
- 1 Pt. The FAB's dimensions are correct.
- 1 Pt. The FAB has a plus sign and the 'accent' color.
- 1 Pt. The FAB is positioned at the right bottom of the channel area.
- 1 Pt. The FAB has a shadow complying with material design standards.

Your `#syntax` will be graded automatically. In total, 9 points can be obtained.

Instructions

Pretty, pretty buttons `#btns`

- Get an overview of how material buttons look like. Investigate the missing values and define the following properties for **all buttons** in your document. You can convert the unit *dp* used by Material Design for size 1:1 to pixels. Please use the values stated in the table below. If you find values deviating from those given in the Material Design Specs, please ignore them or ask in the discussion board. Find out the missing values! `#all`

Buttons	Properties	Values
All buttons	height	?
	minimum width	?
	text size	?
	corner radius	?
	left and right padding	16px
	margin	8px
	background color	white
	border color	none
	text color	primary text
	text	uppercase

- These styles will conflict with the button styles you've declared in previous sprints. The goal now is to clean up your CSS `#fix`.
  Since you will love the Materials look, remove all other button **style** declarations (e.g. in the tab bar buttons, +5 min. buttons). However, keep and adjust the hitherto custom button **alignment** (position of +5min. buttons, 33% / flex width and without margin in tab bar, ...). Also ensure that you keep the `.selected` declarations in the tab bar. Use the image of the challenge goal for orientation.
- Now we do something which will **safe you loads of time**: generate reusable templates! Define two CSS rules for the button color schemes as shown in the table below. You do not have those classes in your HTML file yet. However, after defining them, you have a style template which you can apply to any HTML element. `#clss`

Buttons	Properties	Values
primary	background color	primary color
	text color	primary color text
	background	

accent	color	accent color
	text color	accent color text

- Make the emoji button a primary button `#sml`
- Make the "send" button an accented button `#snd`
- Turn the +5min. buttons into accent buttons by adding the class to the HTML elements in the messages div as well as in your `createMessageElement()`. `#acc`
- (*) Make the star, in the right app bar, a primary `<button>`. The button should contain the icon. Take care that all starring functionality in your functions `star()` and `switchChannel()` still works afterwards (adapt listeners and either ids or your jQuery selectors) `#str`
- (*) Hovering and pressing the default, primary, or accentuated button, darkens its background color by 15%. Since the buttons hover differently based on the chosen class, you need to combine the pseudo-selectors and the class selectors. Luckily, you can combine selectors. `#hvr`
- (*) Google how to remove the blue focus outline of **buttons** and the text **input** element – and remove them! `#fcs`

Pretty, pretty shadows `#shdw`

- According to material design, one creates depth by elevations. Unfortunately, Google has not specified the exact numbers. Scott Roa has reverse-engineered the CSS classes on stackoverflow. Copy the `shadow--2dp` to `shadow--24dp` classes into your styles and don't forget to give credit (e.g., in a comment). Elevate your components: **app bar**, **tab bar** & **chat bar** (consider both app bars), the left **channels view** (like a nav drawer), and the **eomoji menu** `#elv`. Finally, remove the border of the emoji menu as it isn't needed anymore.

By simply applying the shadow classes, it will not yet look like in Google's examples. The elevations assume that your app-bar is above the other content. But your browser renders the current app differently. It builds the website in the order you've written it, from HTML's top to bottom. Later elements appear above existing ones – and you define your tab-bar first, your channels container second, and so on. So your app-bar is actually below the channels and hence can't drop a shadow on the channels list.

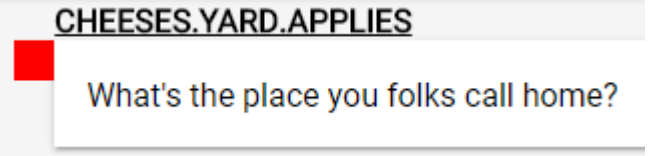
- (*) Figure out how to use the z-index CSS property and fix the shadow errors of the left **channels** view and left **app bar**). `#idx`

Pretty, pretty messages `#msgs`

- Since we'll create the messages programmatic (and there will be a lot), we don't want too many extra classes in our HTML for performance reasons. Ensure that the messages' paragraph (the white-background text) drops an 2dp shadow without adding the shadow classes you've already defined. `#shd`

Let's target the speech bubble. You could add an extra image or an empty `<div>` for the triangular corner and style it. However, we don't want these extra elements either (they will take computing time and storage, and we don't want to modify our `createMessageElement()`). This is where the `:before` pseudo-elements come into the play – CSS can render additional, non-existing 'elements' for you. This is ideally suitable for the speech bubbles' corner, since it's just style and no content.

- (*) In preparation of implementing the desired corner, attach a 20px red squared box to all messages' paragraphs using a `before` pseudo-element (pure CSS). This means, that you don't have to add an extra HTML elements for that box: `#psd`



- (*) Learn, how to make a triangle in CSS (watch the animation). Reshape your box to such a 20px-wide triangle pointing to the left. `#tri`

- (*) In own messages, the arrow should be attached to the right of a message `#own`.
- Notice, that the messages' margins are set to 2.5% and 17.5% respectively? However, your arrows are now 20px wide, so the left margin should be 20% (the indentation) minus 20px (the corner width); resize your browser and you'll notice the issue. Use `calc()` (see documentation) to calculate mixed pixel/percent margins. `#mrg`
- (*) The triangles on the messages look poor without a shadow. Get one. Box-shadow won't work? Well.... You need to create a different shadow, to avoid overlapping (also differently for the left and right triangle). `#ash`

More `#flex`?

- (*) Flex the whole `#chat-bar` and distribute the buttons along the main axis. The first button should be on the start of the container and last one on the end line of the container. Both should also be vertically centered.

A floating plus `#flt`

One of the most characteristic components of Material Designs are Floating Action Buttons, short FAB. You find that one in most apps designed according to MD, like this materialdesignblog

- You have to create a floating action button `#btn`, as well. Check it out in the picture above. It's the plus button in the channel area. Now you just style it. The idea is to let the FAB add new channels, but you do not have to worry about that – for the moment. Find out how to create a circle in CSS, with a plus sign on it. It should be accented. Check out correct sizes for FABs.
    - plus sign
    - color it accented
    - circle
    - correct text size and icon size
    - position: 16px above the tab bar, 16px left to the channels' view's right border
    - correct shadow

`#cleanup`

- Structure and comment your CSS.
- Check code & syntax. Using W3C Validator helps.

Done?

Do not forget to save your code! You will build upon your work in the next implementation challenge.