CCE-UI Take-home (Frontend)

Background

This take home exercise consists of a single question. We encourage you to time box this exercise between 4-6 hours but don't require that you take only this amount of time to complete it.

The goal of this exercise is for us to determine how well you know JavaScript, HTML, CSS, and other aspects of UI engineering. We are also looking at how you make tradeoffs.

During the interview

You should be able to demo your application and explain your choices. What would you have done differently in a real application? How would you handle performance if we needed to hypothetically render thousands of calendar events in this grid? How would you go about measuring the performance of this application using the Chrome DevTools?

If you are unable to demo the application or complete the assignment, you should be ready to explain what might have gone wrong and what the problems were.

Submitting your work

1. When you feel the assignment is complete, send us a link to your code on GitHub or Glitch, or you may zip the code and send via email.
2. There's no need to deploy as a web application. We can run your project locally to look at it.

Please feel free to reach out to our team at studio-ui-media-eng@netflix.com if you have any questions!

Build a Netflix Originals calendar

Netflix needs a calendar application that displays our slate of upcoming original titles for the year. The task is to create a Calendar UI that displays Netflix Original launches.

Requirements

We will assess your work based on the criteria below, *but we are also implicitly looking for code quality and anti-patterns*. Pretend that you are submitting this for code review to your team.

- ▢ Use JavaScript, TypeScript or Flow to create a Single Page Application in the framework[1] of your choice[2].
- ▢ Include a README.md with instructions for how to run the app. Ideally, setup consists of something like npm install && npm run start.
- ▢ In the README.md or through code commenting, please include any tradeoffs you made (e.g. ran out of time so you didn't do X) or decisions you'd like to explain.
- ▢ The calendar should render events in the "Monthly" view (see example below)
- ▢ The app should be URL driven. So visiting http://localhost:<port>/2018/4 should take a user to April of 2018. Visiting an invalid URL should take the user to the current month.
- ▢ Provide a simple *Previous* and *Next* month button to allow navigation between months.
- ▢ Figure out a client-side algorithm for title launch placement within the calendar cells. Use the events provided here as a sample. *Note that while the provided events are finite, in practice the number of titles launched in a month are in the thousands.*
- ▢ Fetch the provided events JSON from a simple node.js server.
- ▢ Figure out a client-side algorithm for how you will render the weeks and days.
- ▢ Use Flexbox or CSS Grid to render and style the calendar. Do not use the Table element. While this is not a design position, we do look for a sense of design, so feel free to make this look good, but if you find yourself up against the 4 hour mark, don't sweat it.
- ▢ *Do not use*: any UI component library (e.g. material-ui, react-datepicker).
- ▢ *Do use:* functional helper libraries (if you want), such as Lodash, Moment, or date-utils.

[1] (P)React, Angular, Vue, Ember, etc.
[2] We use React for most of our apps, but use the one you are most comfortable / confident with. Whichever you choose, it should be idiomatic code for that framework. Ultimately, we are evaluating this submission based on your knowledge of JavaScript/TypeScript and the DOM, not the framework specifically

Your finished calendar layout should look similar to this wireframe:

← August 2017 →

Su	**Mo**	**Tu**	**We**	**Th**	**Fr**	**Sa**
		1	2	3	4	5
6	7	8	9	10 Orange is.. House of...	11	12
13	14	15	16	17	18 Stranger ...	19
20	21	22	23	24	25	26
27	28	29	30	31		

Starter Templates

Setting up a modern frontend stack can take a decent amount of time. While you're welcome to roll your own setup, we encourage you to use some existing templates. For instance:

create-react-app
create-react-app-typescript
create-react-app-typescript-node (provides a node server to simplify specification #8)