UI Interview

Let's go over the UI Engineering Interview.

We think this is one of the most fun interviews, because it's the interview that's most relevant to your day to day work as an UI engineer.

When you go into the UI interview, you'll work together with an engineer over a live session — usually over Coderpad or Codepen. You'll encounter two kinds of problems:

Either you'll work together to build a UI — perhaps a modal, autocomplete, drag drop, etc. Or, you'll build a javascript library — perhaps a bunch of lodash functions, or a pub-sub system that takes advantage of some core javascript principles

Through the questions, the interviewer is looking to parse two signals:

- Javascript Experience. They want to know, can you actually build out UIs and be productive at your job? Are you familiar with the practices in the industry, for building client-side applications?
- Engineering Fundamentals. They want to know, do you know more than just javascript? Are you familiar with data structures, can you model data well, can you communicate your ideas, and would you be able to build an end-to-end system?

To communicate this signal the best, we suggest you get comfortable with the following:

Setting up online editors.

You'll constantly be working in Codepen. You'll most likely want to use the libraries you use at work — usually React — and you should be able to set up an environment that renders "hello world", in a few seconds.

Building a bunch of UIs from scratch in online editors.

You have been building UIs at work all the time, but it could have been a while since you have built a modal from scratch, or autocomplete, or worked with drag and drop. In the homework section, we've included about 4 or 5 different UIs that we suggest you build. These are meant to remind you of some of the JS concepts that you may not have gotten to do frequently at work. These questions are also **very frequently** asked in real interviews. Although some examples, using JS's selection api, or drag and drop, are uncommon, building a few projects here will ramp you up very quickly.

Debugging in online editors.

You may already have mastered chrome's developer tools from work, but it always helps to brush up. It'll also be a bit different in online editors. You should be able to freely use debugger and console.logs to quickly narrow down on bugs. We've included an article that goes over all the tools you can use in Google Chrome. This can be helpful in and out of the interview. As you practice building a bunch of UIs, make an effort to use the debugging tools to narrow down your bugs more and more quickly.

Latest and greatest in JS.

The example UIs may not specifically cover what you know and don't know. We want you to reflect — what are the concepts that you've wanted to learn, that you haven't learned yet? Perhaps you've rarely used Canvas, or haven't tried Redux yet, or want to play with service workers. Catching up on these will help send **very positive** signal during the interview — showing you truly love the industry and are a self-learner. We suggest taking a look at some of Wes Bos's courses — he loves the UI world and goes deep on a bunch of the latest concepts.

Data structures and algorithms

Even though you are in the UI engineering loop, and the chance you get asked heavy algorithm questions is lower, we strongly suggest preparing for algorithms anyway. Go through the **Algorithm Interview** section, and prepare for this interview too. If you prepare well for those, most of the coding-related questions in the UI loops will feel easy.

The live interview

When you go through the interview, we suggest you structure your answers like this:

- Write the JS logic, and do not worry about CSS.
- Do not worry about breaking things up into too many different functions or components. This doesn't necessarily send signal, and if the interviewer has a follow-up that changes the question, you'll end up having trouble refactoring.
- Once a js implementation is done, **verify that it all works.** As you verify, communicate what you are checking, and how you are checking it to the interviewer. This will help you catch a bunch of edge cases, sending positive signal to the interviewer.
- If you end up having time, you can start abstracting, or just mention how you would abstract

To get good at doing all this, we suggest you schedule 2-3 practice interviews with your UI engineering friends, or use Pramp. If you follow this, we're confident you will succeed at this interview and grow as a UI engineer!

Homework

1. Set up Codepen with your environment. Go to codepen.io, and create a new pen. Try to set up the environment that you normally use (For example — React + Babel). You can do this by clicking on Settings, and adding the right library. Your goal is to render "hello world"
2. Get comfortable building UIs. Here are sample problems to work through
   - Create Tic-Tac-Toe
   - Create a Trello Board
     - Allow creating boards
     - Allow creating cards
     - Allow editing cards
     - Allow moving cards
     - **Bonus: Allow drag and drop**
   - Draw a sudoku board. This is a 9x9 grid, with each 3x3 square colored differently. Part two, given a 2D array with sudoku solution, render the board with the values and highlight any invalid rows, columns, or squares.
   - Create a Credit Card Input. As you enter the numbers, it should add spaces after every 4th digit
   - Create an event emitter. Think about what is the right data structures to use here? All of your calls — subscribe, unsubscribe should be O(1). See demo code below for expected behavior
   - Create a popover component
   - If you're comfortable with these, ask yourself — what do you want to learn? Create a quick project for that
3. Get good at debugging
4. Catch up on the latest and greatest of JS
   - Wes Bos's courses are great — search up on youtube anything you're curious about — search up talks
   - Check out some interesting articles and start to go deep from there
5. Schedule 2-3 mock interviews with friends to do over Codepen

Event Emitter Demo Code

```
const emitter = new EventEmitter();
const subOne = emitter.subscribe('foo', fnOne);
const subTwo = emitter.subscribe('foo', fnTwo);
const subThree = emitter.subscribe('bar', fnTwo);
```

```
emitter.emit('foo', 1, 2, 3);
// fnOne(1, 2, 3), fnTwo(1, 2, 3) is called

subTwo.unsubscribe()
emitter.emit('foo', 1, 2, 3); // fnTwo is removed
// fnOne(1, 2, 3)
```