



NodeJS

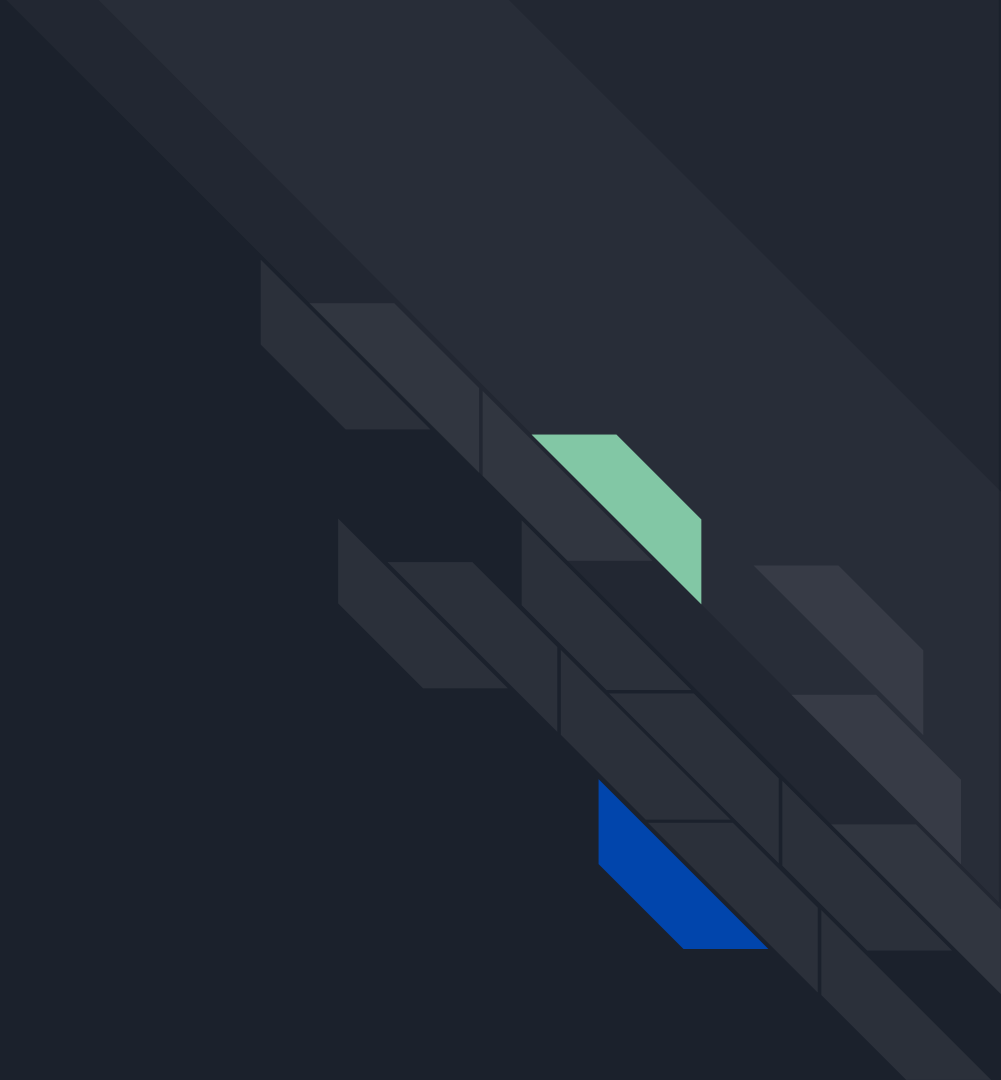
Stephanie Mah



First things

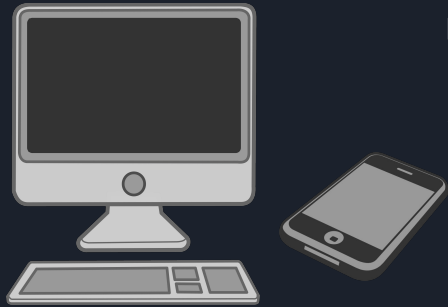
Run `npm install -g express-generator`

Big Picture



Client vs Server

Client



Server



Database





Client vs Server (310 throwback edition)

Model View Controller (MVC) Pattern





Client vs Server

Client

- Your code runs on devices you don't control
- Eg. JS running in a web browser

Server

- Your code runs on devices you do control
- Eg. JS running on a NodeJS engine on a computer that the app developer configures and maintains



Implications of Client vs Server

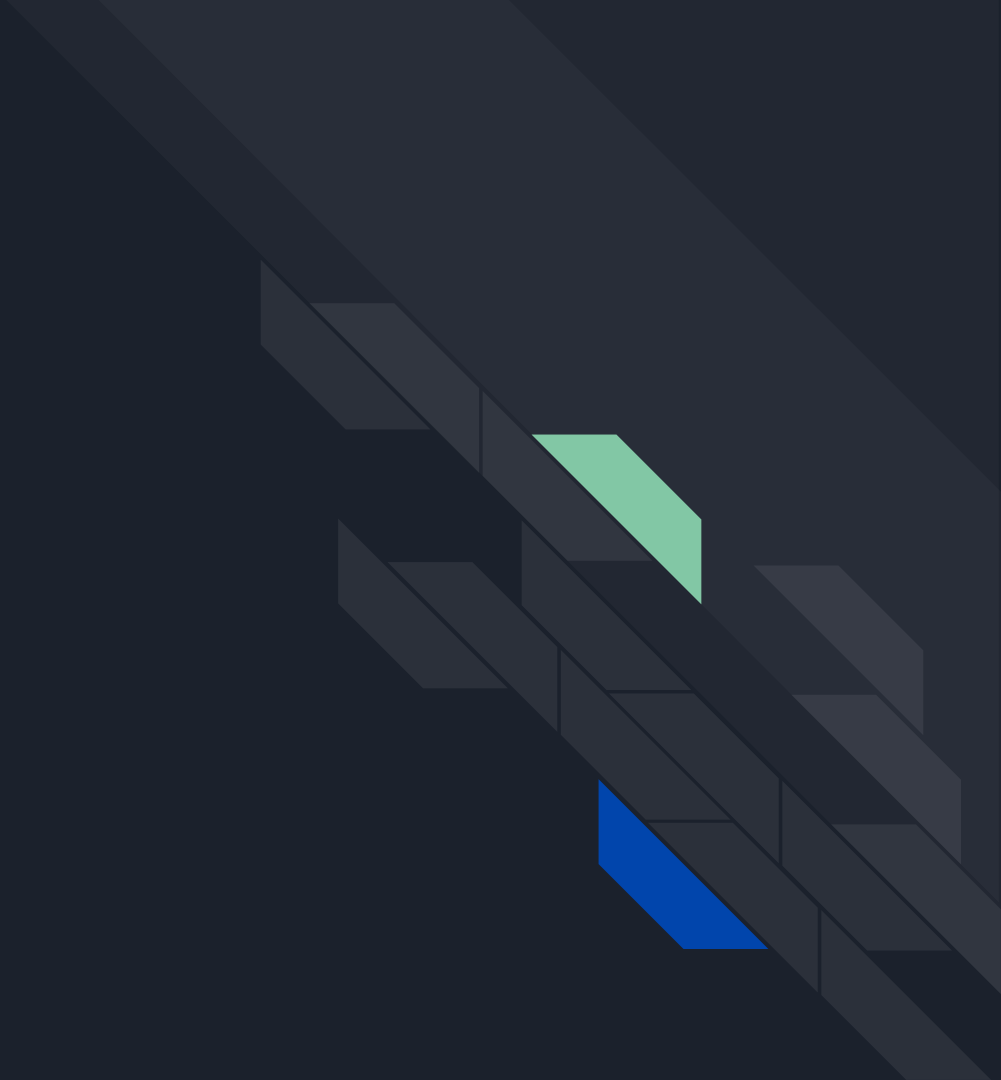
Client

- Your code is easy for users/competitors to read
- Depending on the platform, your code is easy for users to modify
- Data storage can be manipulated
- Users can keep running old versions of your code
- Performance unknown

Server

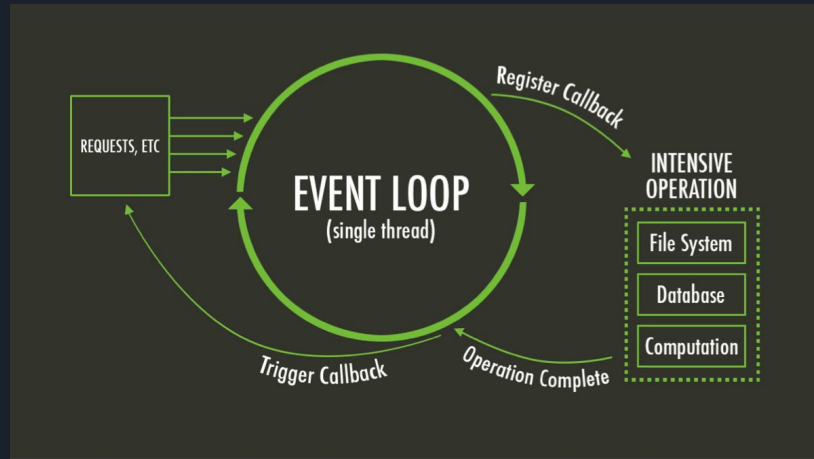
- Your code is private (if you want it to be)
- Your users can't change what your code is doing
- Users can't modify data unless you want them to (as long as you follow good security practices)
- You control when your code is updated
- You control CPU, memory, bandwidth etc.

NodeJS



NodeJS

- Javascript on your server
- The “node” part refers to the javascript runtime that is built off of Google Chrome’s v8 javascript engine
- Runs in a asynchronous event looped environment - Operations that wait on slow resources yield the single thread to other events and wait for a callback from the slow operation





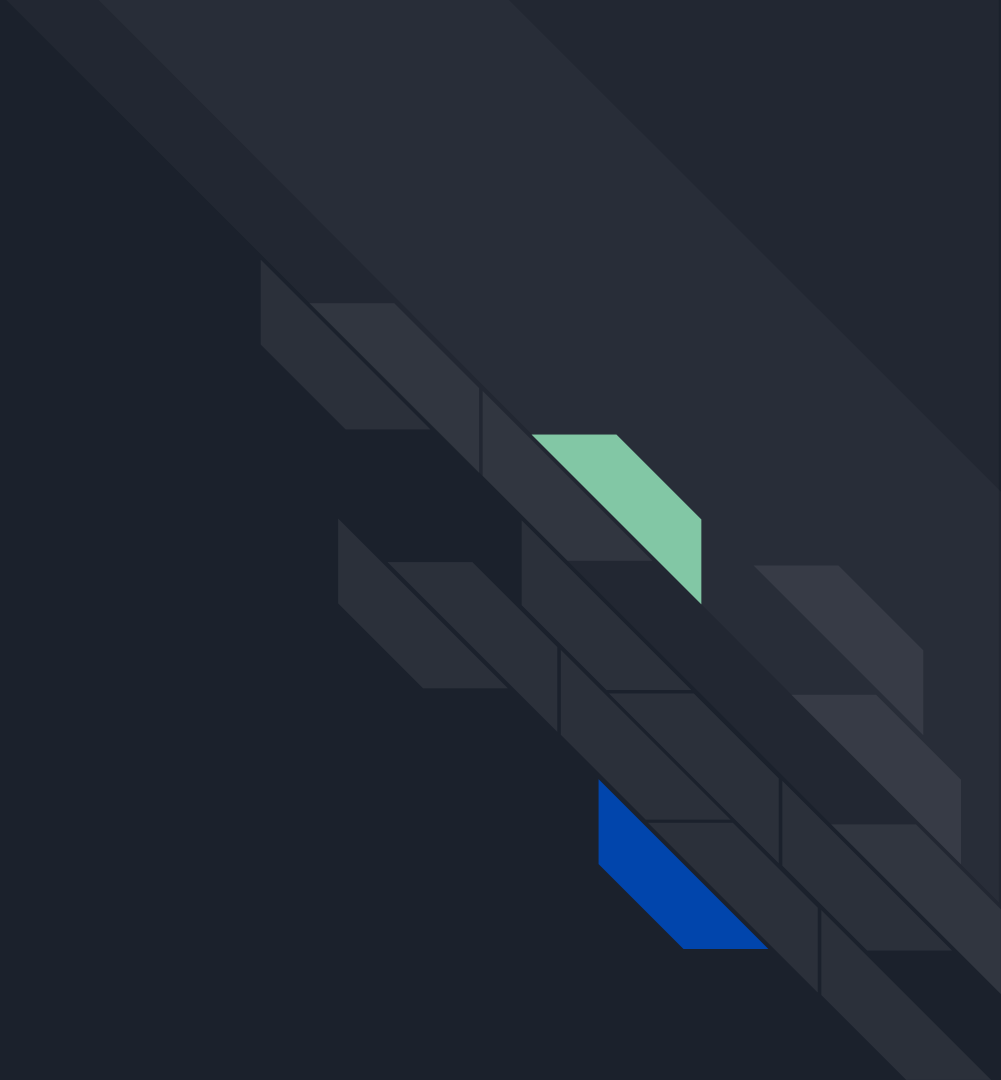
Simple NodeJS Server in 30 seconds

```
const http = require('http');
const hostname = 'localhost';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://\${hostname}:\${port}/`);
});
```

Express





Express

- “Fast, unopinionated, minimalist web framework for Node.js”
- Express takes repetitive low level code (eg. parsing the request or formatting the response) and gives you a higher level API to use
- Faster to code in and has more features than vanilla NodeJS.



Express App Setup

```
cd <react-app-directory>  
npm install express-generator -g  
Express server --no-view --git  
cd server  
npm install  
npm start
```

Now check that it's running :)

Once you can see that it's running, stop the server (Ctrl-C). We'll come back to it in a bit.

APIs



API

Application Programming Interface

Provides a connection between computers or computer applications.



REST

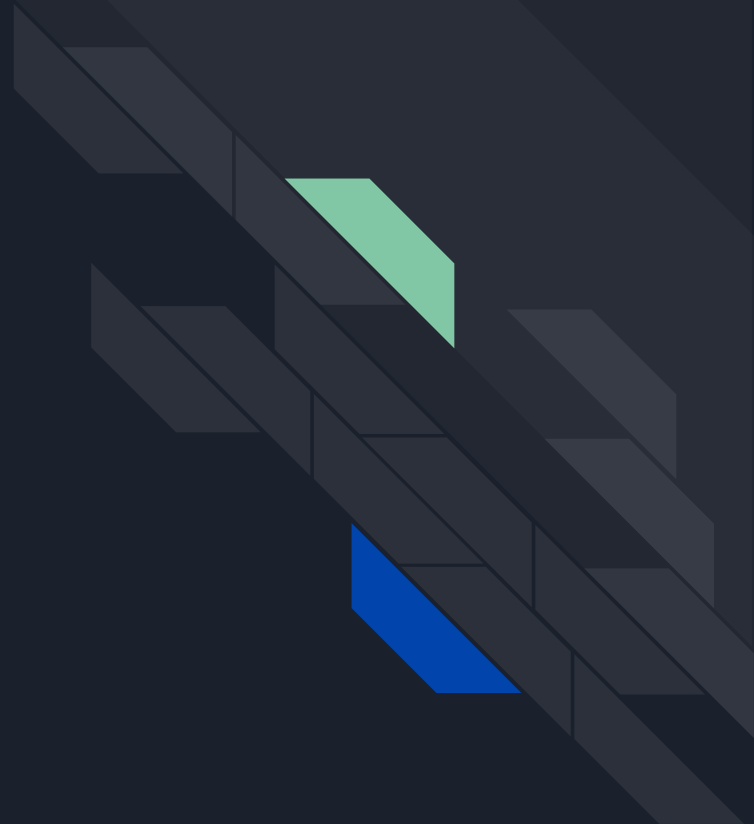
Representational State Transfer

Architectural style that defines a set of constraints for the access and manipulation of textual data on the internet.



Idempotence

Property of an operation such that repeating the operation produces the same result.





HTTP Methods

Method	CRUD	Idempotent	Purpose
GET	Read	Yes	Read 1+ resources
POST	Create	No	Create 1+ new resource
PUT	Create/Update	Yes	Create/Replace resource with given id
PATCH	Update	No	Modify a resource with given id
DELETE	Delete	Yes	Delete resource with given id



POST vs PUT vs PATCH

POST

- Will always create a new resource
- NOT idempotent - POSTing multiple times with the same request body will result in multiple resources being created

PUT

- Will **replace whole** resource or **create** a new one if none exist with the given ID
- Idempotent - PUTting multiple times with the same request body will always result in the same resource at the given ID

PATCH

- Will **replace part** of a resource with the given ID
- Idempotent - PATCHing multiple times



Anatomy of a Network Request

Request URL - the address of the resource you're requesting

Request Method - GET/POST/PUT/PATCH/DELETE

Request Headers - unrelated to the message content - typically used to provide context to the request (ie. `Authorization`) but can also allow for conditional requests (ie. `If-Modified-Since`)

Request Body - the message content for PUT/POST/PATCH requests; commonly JSON object or array



Routes

- A route is the request URL of the API you intend to hit
- Grouped by common resources (an entity in your system, usually as it corresponds to a collection in your DB)
- May include the id of a resource, a MUST for certain operations
- Eg. /users or /users/:userId



Example operations for /users

GET - return a list of users

POST - add a new user

PUT - typically not supported - remember: PUT will create/replace resource with given id

PATCH - typically not supported - remember: PATCH will modify a resource with given id

DELETE - typically not supported - remember: DELETE will delete resource with given id

Sage advice from someone who TA'd this course - **Do not use POST for what should be a DELETE call. Configure your routes to accept an ID, don't just pass it in a POST body because you can. PLEASE.**



Example operations for /users/:userId

GET - return a user record with id = userId

POST - typically not supported - remember: POST will create a new resource; use PUT instead

PUT - add or replace the user record with id = userId

PATCH - update part of the user record with id = userId

DELETE - remove the user record with id = userId

Let's configure a route





Configuring Express

Change the port so it doesn't conflict with your react port:

1. Go to your new express app > bin > www. (Does the code look a bit similar?)
2. On line 15, change the '3000' to '3001'

If you run your express server again, you should be able to see it working properly on localhost:3001 now.



Configuring the /user route

As you can see, express-generator has already started a users route for you at routes > users.js

Here you can configure your /user routes. Here are a couple examples.

```
router.get('/:userId', function (req, res, next) {  
  const foundUser = users.find(user => user.id === req.params.userId);  
  return res  
    .setHeader('Content-Type', 'application/json')  
    .send(users);  
});
```

```
router.post('/', function (req, res, next) {  
  const user = { id: uuid(), name: req.body.name };  
  users.push(user);  
  return res  
    .setHeader('Content-Type', 'application/json')  
    .status(201)  
    .send(user);  
});
```



Let's call our API

cURL is a CLI for http requests

```
curl --request GET --header 'content-type:applicaton/json' --url 'localhost:3001/users'
```


```
curl --request POST --header 'content-type:application/json' --url 'localhost:3001/users' \  
--data '{"name": "Steve"}'
```



React App integration

- Create the functions that will call your API
 - You can use the http request library `fetch` that's built in, or something fancier like `axios`
- Where should you use the function?
 - In your component
 - In Redux ~thunks~

Let's do it!



Oh no! CORS!

CORS (Cross Origin Resource Sharing) - a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served

In your express app folder, run `npm install --save cors`

Then import it in your app.js file

```
var cors = require('cors');
```

And use it

```
app.use(cors());
```



Things to consider when designing APIs

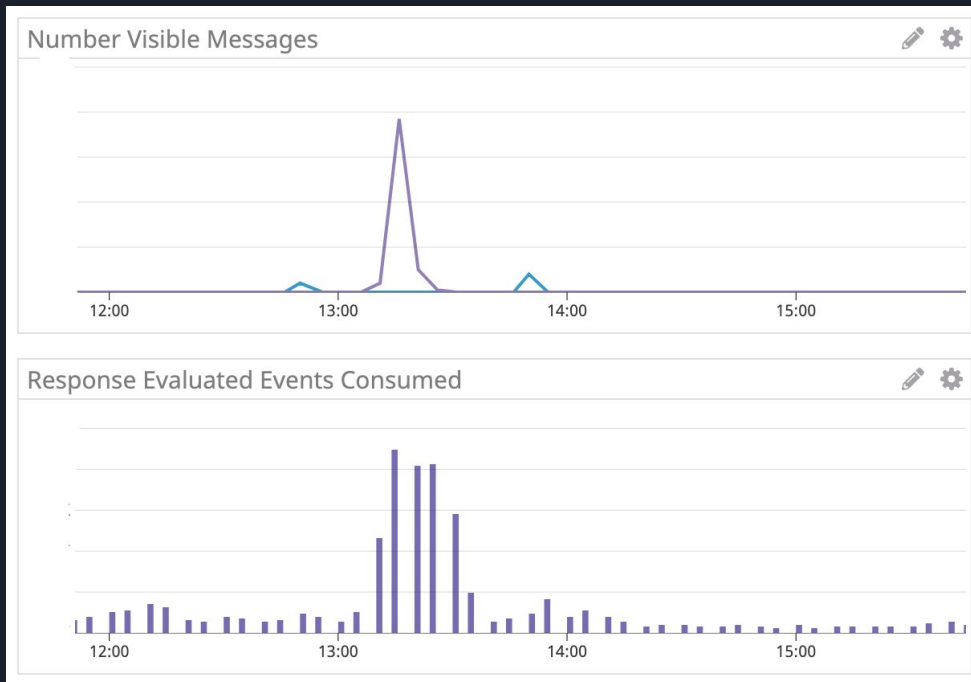
- Pagination
 - Limit/offset?
 - Cursor-based?
- Meta-data
 - Response headers? (ex. Total counts, pagination links, etc.)
- Query parameters
 - Filtering? Sorting? Search? Field selection?
- Error handling
 - What error code should you return?

Advanced backend development topics



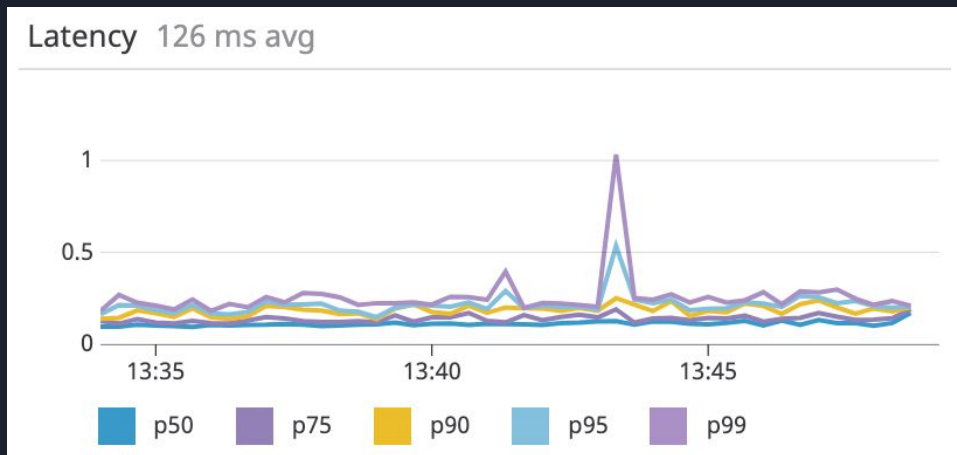
Scalability & Performance

- How can you continue to quickly handle requests when you go from 100 requests to 100k?
- How do you handle unpredictable and spiky loads?
- What is the average response time for 5 requests/second on a single endpoint? What about 100? 100,000? What is the failure rate?



Observability

- When you process millions of requests or events every day how do you keep track of what's going on in your system?
- If you have a bug that occurs only once every 100k requests how can you figure it out?



BREAK TIME



About me





Background

- Started Electrical Engineering in 2009 at UBC
 - Hated it. The math was too hard. :(
 - Left after 2.5 years. No, I really wasn't close to graduating. :/
- Dropped out to do a Bachelor of Fashion Design at Kwantlen
 - The month I spent preparing for my portfolio review was one of the most stressful months of my life.
 - Going to fashion school was a lot of all-nighters, panic attack and crying. Decided that kind of life wasn't for me and quit 2 weeks before starting second year.
 - I can draft my own clothes though :D



Background

- Finally got a Bachelor of Commerce at Sauder
 - Optioned in Finance initially, but added on Business Technology Management too because I could.
 - Worked at a start-up private equity firm FT/PT for 2 years or so
- Decided that the finance life wasn't for me and enrolled in the B.CS program at UBC
 - Graduated in 2020
 - Did my co-op at PAI Health where an 8 month co-op term turned into 15 months turned into a FT contract position.



Background

- Worked at PayByPhone for a year (2021)
 - First time working in a bigger, more corporate setting
 - Didn't like it very much, but stayed for the learning experience
- Started at Rivian in January 2022
 - Now working in a much bigger corporate setting
 - Doing a lot of cool new greenfield stuff
 - Don't get discounts on cars :(

Advice





Things to do in a technical interview

- DO: Talk through your thought process
 - Even if you have no idea what to do, explain what parts are tripping you up and why.
- DO: Ask clarifying questions
 - The questions they give you likely won't specify every detail and edge case. Think about it and ask.
- DON'T: Act like a know-it-all
 - If your interviewer tries to help you, listen and let them. Being able to take direction and criticism is important. Needing help is not a dealbreaker.
 - If your interviewer asks if you think there are ways your code could be more efficient, don't say no. They're usually asking for a reason and there's almost always a way to make it better - refactoring, edge-cases you missed, data-storage options...



Things to ask in an interview

- Ratio of developers to QAs
 - Useful for estimating the workload of a QA if there is one, how much of the testing effort developers are responsible for, etc.
- Makeup of the developer team (senior developers ratio, etc.)
 - Having a solid base of people to ask advice from is crucial to how much you can learn; if your team is made up of other co-ops or interns, you may not learn how to write good code.
- Code review/pair programming practices
 - If no one ever looks at your code, you'll never get feedback on how to improve or learn a quicker or better way of doing things.
- Testing strategy? Automated testing? TDD?
 - As much as many of us hate writing tests, they are critical for quality and robustness of the code base. Similar to above, it's good to know what you're walking into as a developer or a QA



Things I rarely do

- Write code outside of work
- Work more than 8 hours a day - but sometimes I'm on a roll and I'll keep going for a few more
- Network
- Do Leetcode/what-have-you
- Do Hackathons



General Co-op Advice

- You'll always learn something, so don't feel bad for prioritizing comfort (a job looking for skills you already possess) over stretching yourself (a job with an entirely new stack) - there will be opportunities to stretch yourself at that comfy job too!
- Pick the hard project - but don't be afraid to ask for A LOT of help with it if you need to
- Ask your coworkers about how to improve your code - it's easy to get away with subpar code, but it won't do you any favours
- Start-ups have their benefits and so do large companies, but they also have their drawbacks too
 - You may do and touch more things at a startup, but may lack code quality and mentors
 - You can learn a lot from a well-established code base and process-wise, but your projects may be less exciting or flexible
- You're a co-op. Nobody is expecting you to be batting 100, 24 hours a day out of the gate.

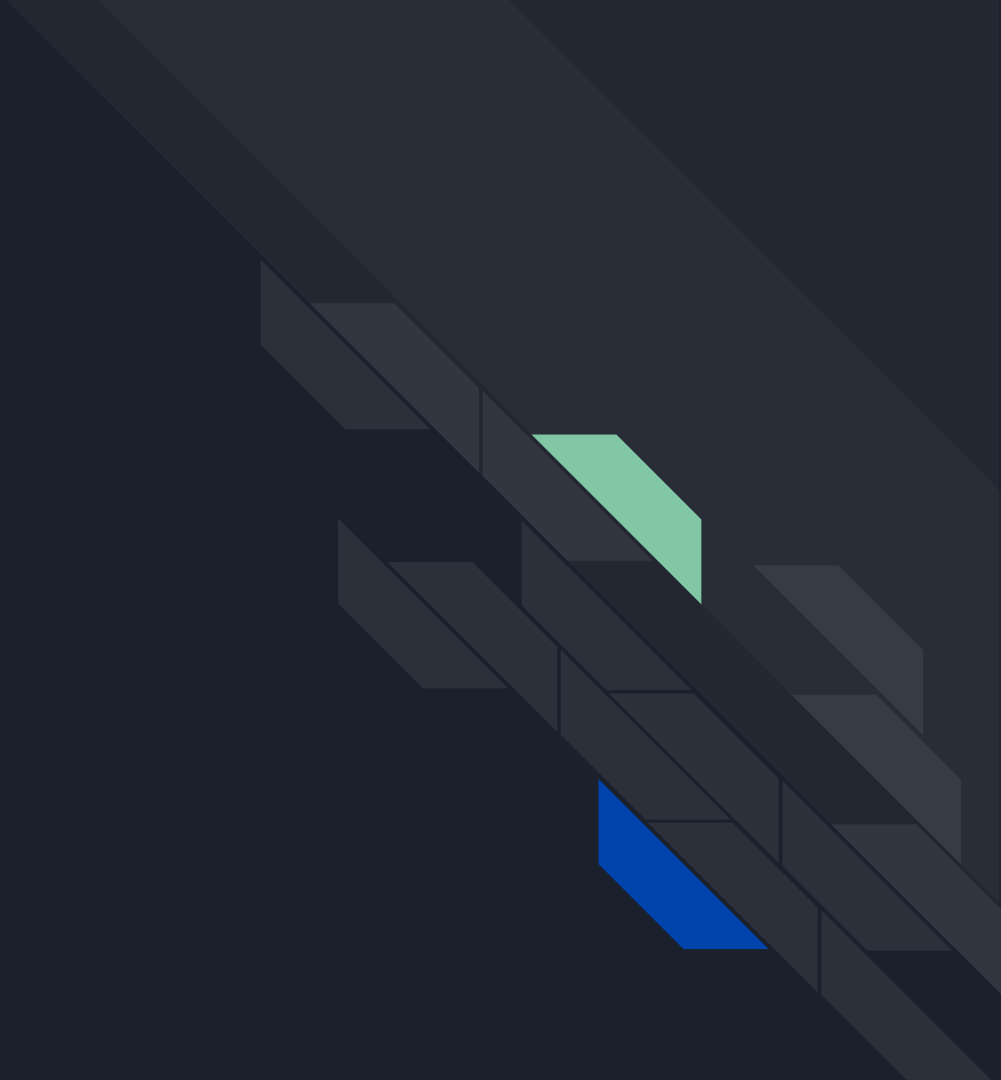


How to write a kickass coverletter in 5 parts

1. Why I think your company is THE BEST THING SINCE SLICED BREAD and why I would die to work there. [Look at the company's website - what do they think is the most amazing thing about what they're doing? Talk to this point.]
2. Slightly personalized but generic reason for why I would be a great fit at your company. Don't believe me? Here are some concrete examples:
3. One verbatim requirement for this position: a concrete example of how I did that
4. Another requirement for this position : a different concrete example of how I did that
5. I believe that my unique experience with XXX gives me YYY skills beyond that of other students. As a company that values, ZZZ, I hope you'll consider me an asset to your team. Blah blah, look forward to hearing from you, blah blah.

Once you've done this once or twice and have enough examples for part 3 and 4, you'll really only need to write part 1. BOOM.

Real Talk





Stephanie's heartfelt advice

- If you have a meandering past like mine, own it. An ex-boss of mine told me that I talked about my past like it was series of mistakes. Don't be like me. My past was not a series of mistakes, but rather a testament to how I refused to settle for a job/degree that I wasn't passionate about.
- Get a post-grad job offer that's perfect, except for salary? Ask for more. Just do it. Thank them for the offer, but tell them you were looking for something closer to the \$\$ range and ask if they can do anything for you. Be realistic though, if you actually want the job, I personally wouldn't go higher than 20-25% higher than their initial offer. The worst that can happen is they say no and counter with their original offer; they likely won't take the offer away completely because they did want you to begin with and if they do, their loss, not yours. You can always ask for a higher title too or in lieu of higher pay, but again, be realistic.
- Be humble and don't hide your mistakes. It's not easy to admit your mistakes, but you'll gain more respect from your peers if you do. On the flipside, imposter syndrome is real, y'all - trust yourself! Run your implementation plan by a coworker; they'll either be impressed, or help you improve it. **I believe in you!**

That's enough about me



Next assignment and
project progress





Administrative stuff

Design reviews starting shortly

- 10 minutes - present your progress
- 5 minutes - questions
- NOTE: We will be timing you but don't stress, it's just for logistics :)

Scrum Reports due at 10pm tomorrow, Sunday June 12th!

- These are *INDIVIDUAL*, do not write one for the whole group.