how to code good*

Christine Imogu & Helen Yang

*good is subjective

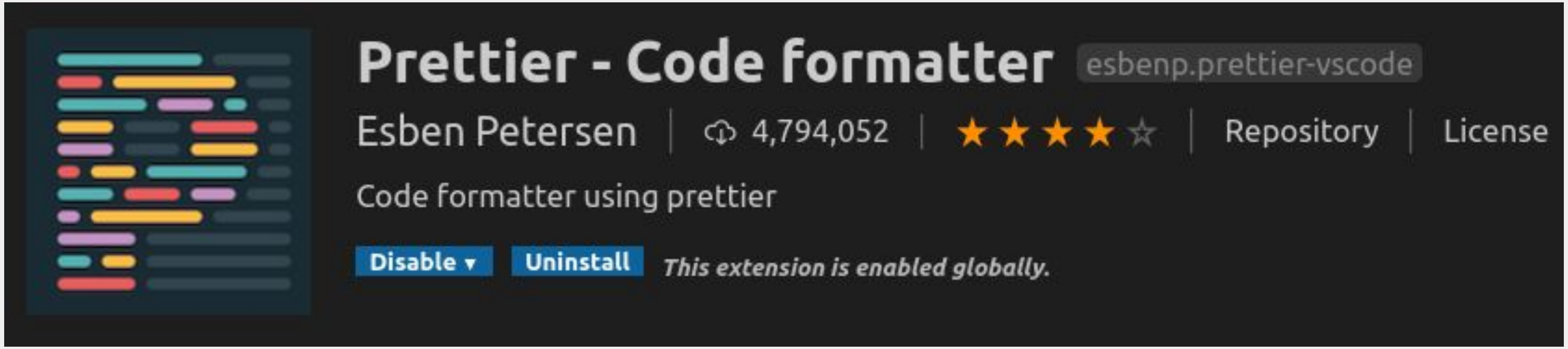
Why code good?

- Having the correct developer practices makes your code more readable, saves time, makes you easier to work with, and leads to less frustration while coding
- For all developers, these mean that coding is more fun, easier to do, and people (and yourself) will understand your code easier
- If you’re a professional developer, all of these mean $$$$$$

Code formatting/style

Use prettier!!

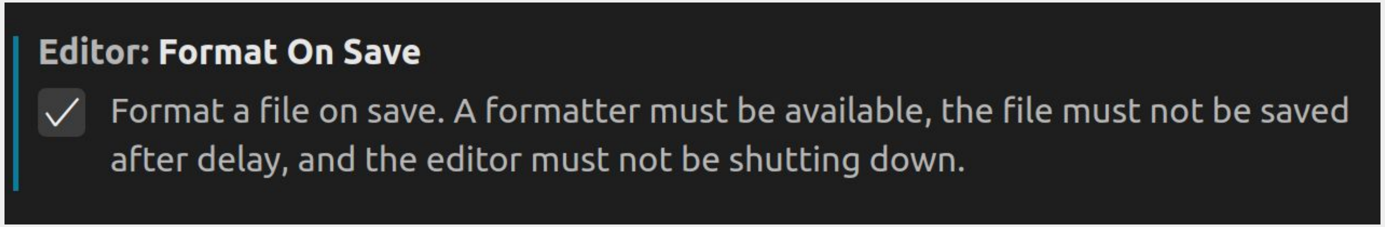
- VSCode extension


Prettier - Code formatter esbenp.prettier-vscode

Esben Petersen | 4,794,052 | ★★★★☆ | Repository | License

Code formatter using prettier

This extension is enabled globally.

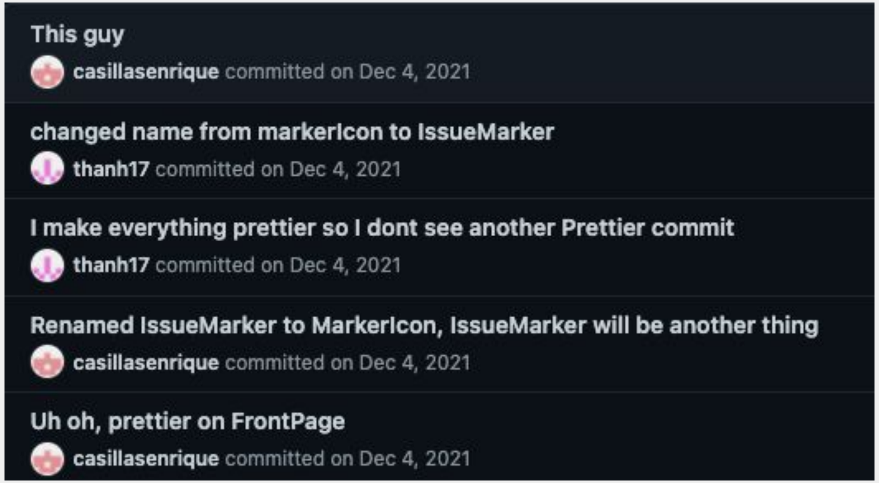

Editor: Format On Save

✓ Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down.

- Makes your code pretty every time you hit “save”

Danger!

- Decide as a team if you want to use prettier
- Either everybody use it, or nobody use it
- **Can run into nasty merge conflicts otherwise**


This guy
casillasenrique committed on Dec 4, 2021

changed name from markerIcon to IssueMarker
thanh17 committed on Dec 4, 2021

I make everything prettier so I dont see another Prettier commit
thanh17 committed on Dec 4, 2021

Renamed IssueMarker to MarkerIcon, IssueMarker will be another thing
casillasenrique committed on Dec 4, 2021

Uh oh, prettier on FrontPage
casillasenrique committed on Dec 4, 2021

Merge Conflict

Inconsistent Prettier package use

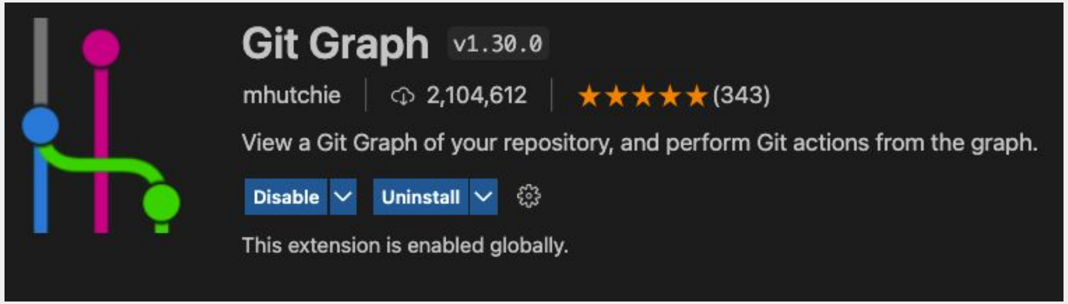
Autoformat with Prettier

Different Tools

*but still make sure you know how to

Git Support

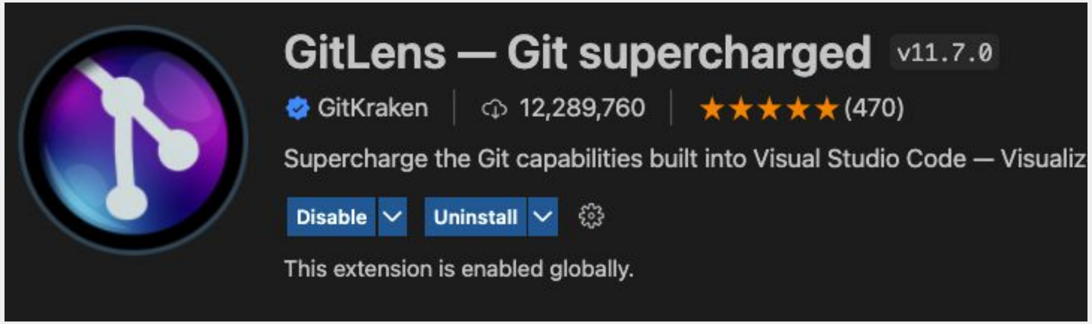
- Don’t like terminal? Don’t use it!*


Git Graph v1.30.0

mhutchie | 2,104,612 | ★★★★★ (343)

View a Git Graph of your repository, and perform Git actions from the graph.

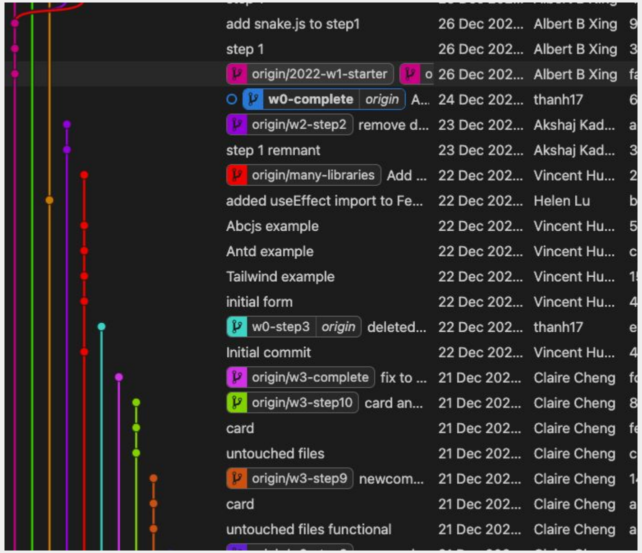
This extension is enabled globally.


GitLens — Git supercharged v11.7.0

GitKraken | 12,289,760 | ★★★★★ (470)

Supercharge the Git capabilities built into Visual Studio Code — Visualiz

This extension is enabled globally.


add snake.js to step1	26 Dec 202...	Albert B Xing	9
step 1	26 Dec 202...	Albert B Xing	3
origin/2022-w1-starter o	26 Dec 202...	Albert B Xing	fa
w0-complete origin A...	24 Dec 202...	thanh17	6
origin/w2-step2 remove d...	23 Dec 202...	Akshaj Kad...	a
step 1 remnant	23 Dec 202...	Akshaj Kad...	3
origin/many-libraries Add ...	22 Dec 202...	Vincent Hu...	2
added useEffect import to Fe...	22 Dec 202...	Helen Lu	b
Abcjs example	22 Dec 202...	Vincent Hu...	5
Antd example	22 Dec 202...	Vincent Hu...	c
Tailwind example	22 Dec 202...	Vincent Hu...	1
initial form	22 Dec 202...	Vincent Hu...	4
w0-step3 origin deleted...	22 Dec 202...	thanh17	e
Initial commit	22 Dec 202...	Vincent Hu...	4
origin/w3-complete fix to ...	21 Dec 202...	Claire Cheng	fo
origin/w3-step10 card an...	21 Dec 202...	Claire Cheng	8
card	21 Dec 202...	Claire Cheng	fe
untouched files	21 Dec 202...	Claire Cheng	c
origin/w3-step9 newcom...	21 Dec 202...	Claire Cheng	1
card	21 Dec 202...	Claire Cheng	a
untouched files functional	21 Dec 202...	Claire Cheng	a

A Different Editor..?

- Webstorm/IntelliJ
  - A lot more resource intensive…RAM goes brrrrrrrrr
  - A lot more features…so can be confusing
- Vim
  - Classic text editor
  - Has a large learning curve, but allows for shortcuts that lets you do some things faster than VSCode

Writing quality code

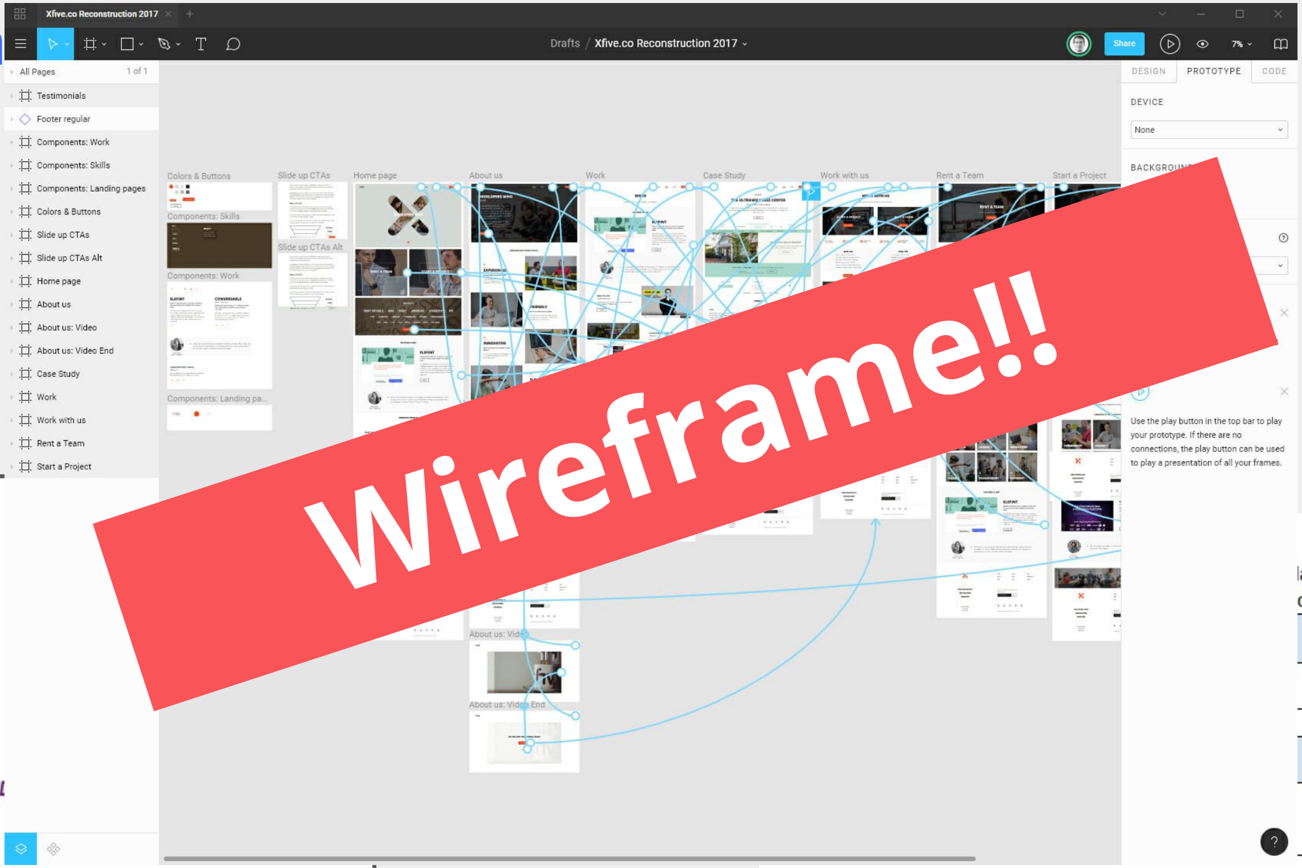
The “best” style is the one that your team agrees on*.

* it may not be objectively the best, and it’s still important to keep your team accountable with tech debt and style

Needing bad code for your website to work is often caused by bad design* choices.

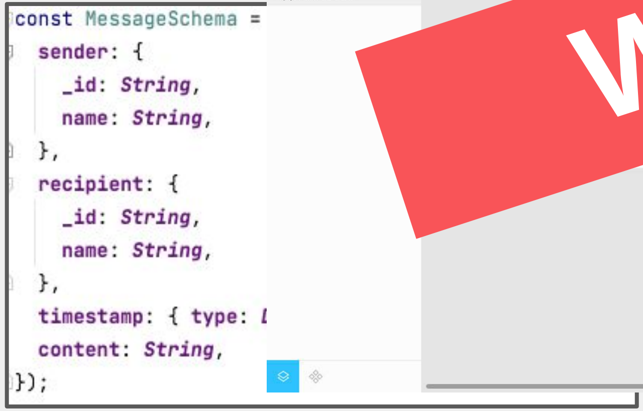
*design here meaning overall system design, not how things look

Design

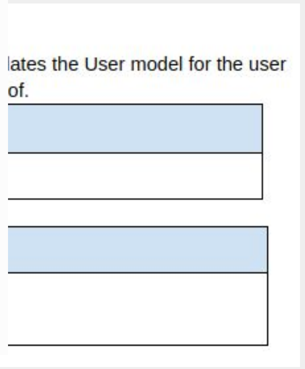


Use the play button in the top bar to play your prototype. If there are no connections, the play button can be used to play a presentation of all your frames.

Wireframe!!



```
const MessageSchema =
  sender: {
    _id: String,
    name: String,
  },
  recipient: {
    _id: String,
    name: String,
  },
  timestamp: { type:
  content: String,
});
```



lates the User model for the user
of.

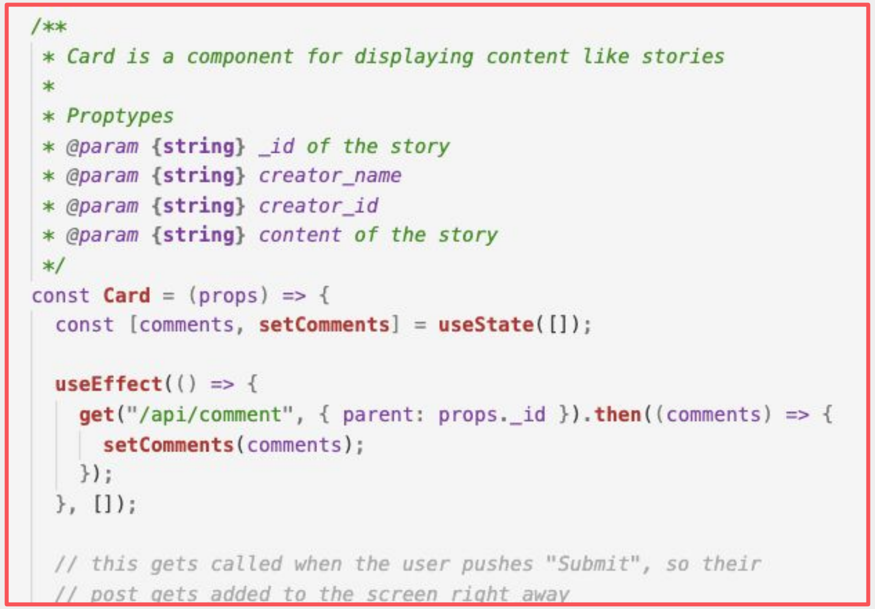
Mongoose schemas

API Specifications

"if you don't have any design, *you don't know what you are making.*"

- some dude from stackoverflow

Ways to Document your Design Choices

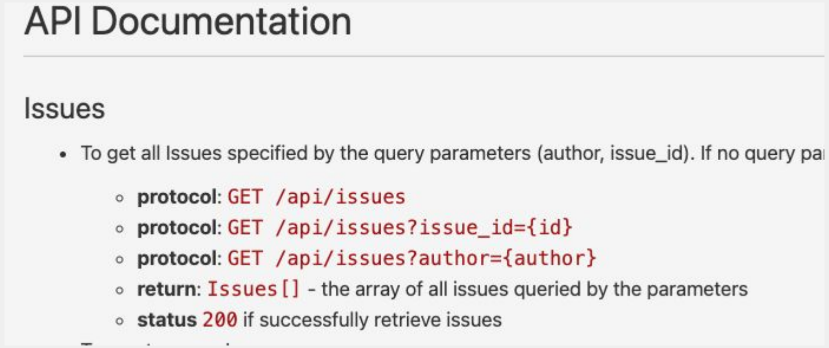


```
/**
 * Card is a component for displaying content like stories
 *
 * Proptypes
 * @param {string} _id of the story
 * @param {string} creator_name
 * @param {string} creator_id
 * @param {string} content of the story
 */
const Card = (props) => {
  const [comments, setComments] = useState([]);

  useEffect(() => {
    get("/api/comment", { parent: props._id }).then((comments) => {
      setComments(comments);
    });
  }, []);

  // this gets called when the user pushes "Submit", so their
  // post gets added to the screen right away
```

JSDocs (Methods/Objects/Handlers)

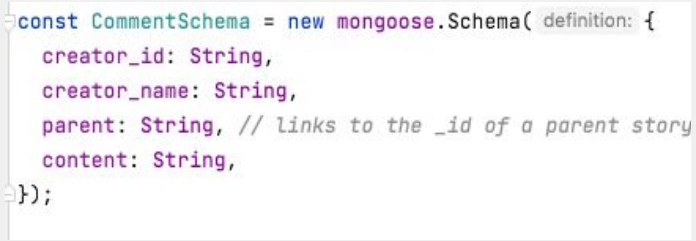


API Documentation

Issues

- To get all Issues specified by the query parameters (author, issue_id). If no query pa
  - **protocol**: `GET /api/issues`
  - **protocol**: `GET /api/issues?issue_id={id}`
  - **protocol**: `GET /api/issues?author={author}`
  - **return**: `Issues[]` - the array of all issues queried by the parameters
  - **status** `200` if successfully retrieve issues

API Routes



```
const CommentSchema = new mongoose.Schema( definition: {
  creator_id: String,
  creator_name: String,
  parent: String, // links to the _id of a parent story
  content: String,
});
```

Database Schemas

API Specification

- For every API route
    - Is it GET or POST (or something else?)
    - What parameters does it expect and what are their data types?
    - What does it change in the database (if any)?
    - How much other work does it need to do (if any)?
    - What does it return and what data types does it return?
    - How does this route interact with other routes? Is it even needed?

Your API routes often depend on your frontend design, not the other way around. (You can also do it the reverse way too)

Scenario: I want to add a new feature, but...

My code is structured in a way that makes this difficult.

Two options:

1. Refactor my code, which might take **3 hours**
2. Come up with a hacky solution, which only takes **30 mins**

What is hacky code?

Code may be hacky if…

- It’s super inefficient
- It’s unintuitive to implement
- Difficult to understand/complicated logic flow
- Might easily break if you need to tweak it in the future
- Messes with a library in a way that wasn’t intended
- and so on…

Avoid the need for hacks at all

- **Design well from the start**
  - Keep in mind your designs will likely change!
  - Draw out a React component hierarchy
  - Discuss the best Mongoose schema with your team
  - Write a specification for your API
  - Feel free to talk to us about this
- In general, plan out and discuss your changes with your team **before, during, and after** implementation.

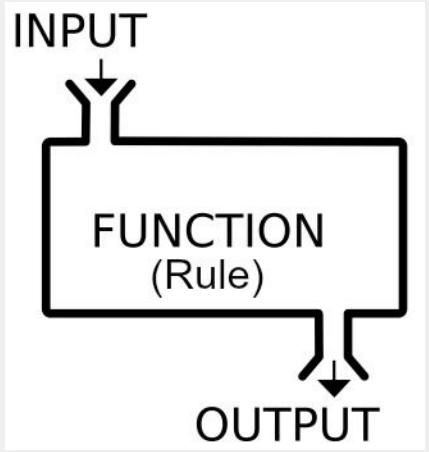
What is clean code?

D.R.Y. Principle

- **D**on’t **R**epeat **Y**ourself!
- If you are repeating code anywhere, you probably want to abstract it into a function
- You’ll have this hammered into your head if you take 6.1020 (6.031)

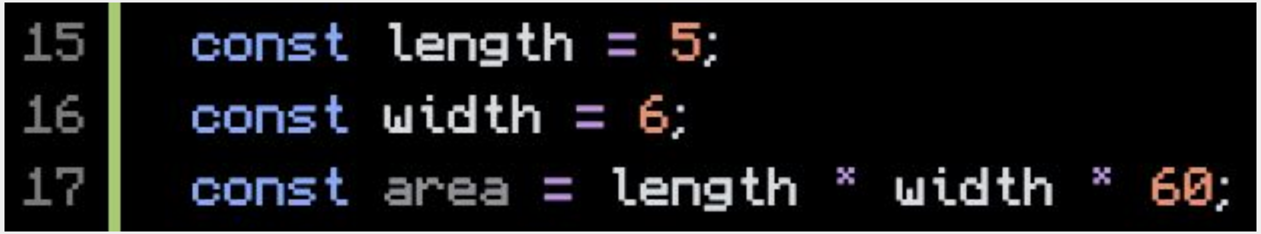
Functions

- Function should only do one thing: they should do it the right way and just do it.
  - A boolean parameter already clearly states that it does more than one thing.
- Clearly define your **specifications**
  - **Preconditions**? What inputs does it take?
  - **Postconditions**? What does your function return?
- Keep it simple



Avoid Magic Numbers

- Magic numbers are numbers that appear out of nowhere
- Example:



```
const length = 5;
const width = 6;
const area = length * width * 60;
```

- In this example, 60 is a magic number.
- To avoid magic numbers, use **meaningful and accurate variable names** for your constants (because they may change in the future!)

Good Variable Names

- Correctly describes what the variable does
- Ex: Say you need a variable to describe how many doors a room has.
  - A not-so-great declaration for this would be: const doors = 2
  - A better declaration for this would be: const numDoorsInRoom = 2
- Good variables names help **readability** and **remove the need to comment your code as much (saves time).**

More Notes on Variables

- Always declare your variables in the **lowest scope** that they're needed.
- **Avoid global variables.**
- Always use **const** unless necessary (this extends to using immutable data structures unless mutable data structures are needed).
- Agree on a naming convention with your team (camelCase, snake_case, etc).

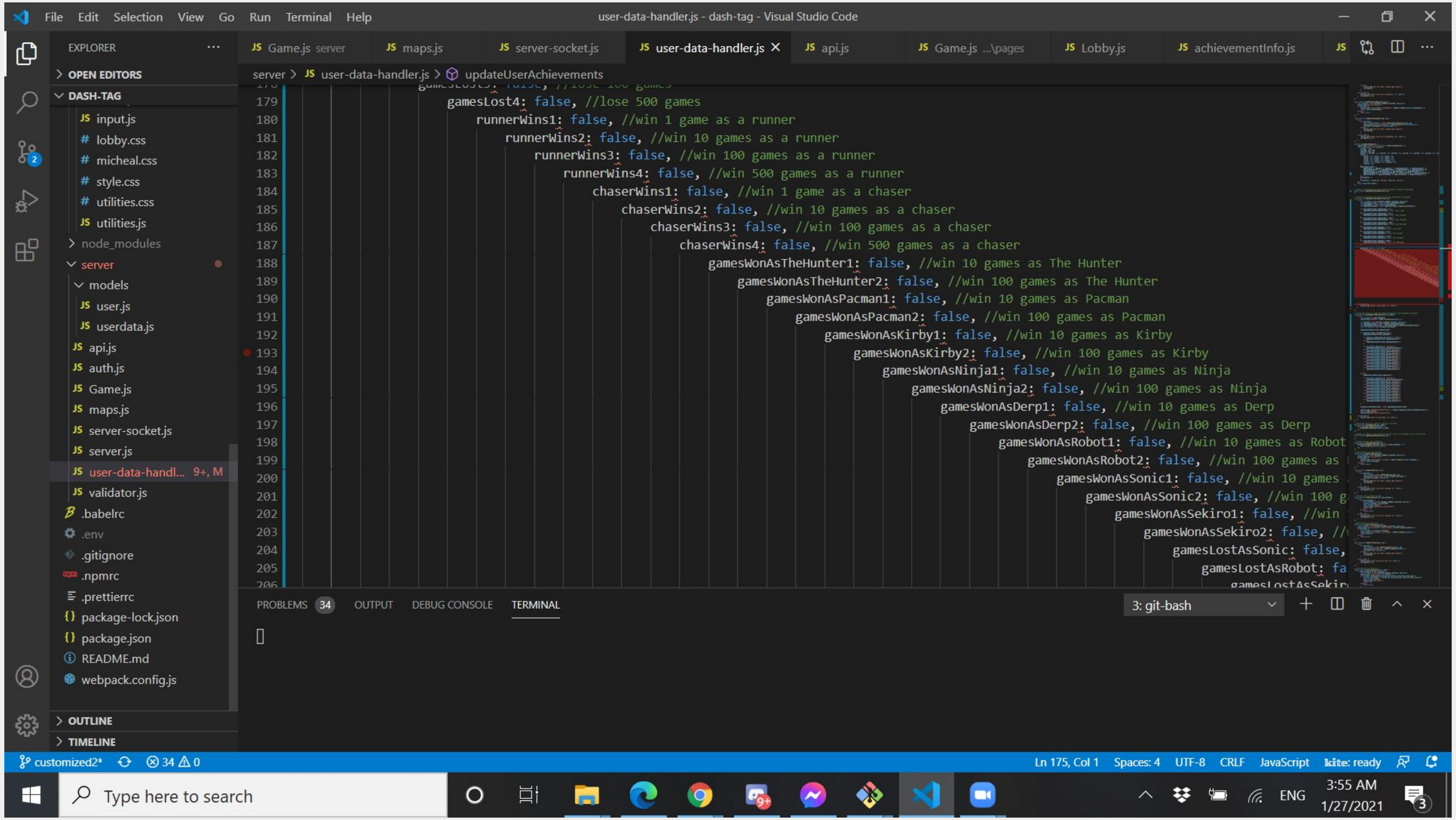
Signs/consequences of janky code

Code Smell

Signs that your code may have some deeper problems…

- Lots of **copy-pasted** code
- Functions that are **way too long**
- Lines of code that are **way too long**
- **Dead code**: Entire functions/files that are unused
- Difficult to understand or add new features
- Lots of bugs, especially random and unexpected ones (from interactions between different components, etc)

If you have code like this:



```
gamesLost4: false, //lose 500 games
    runnerWins1: false, //win 1 game as a runner
        runnerWins2: false, //win 10 games as a runner
            runnerWins3: false, //win 100 games as a runner
                runnerWins4: false, //win 500 games as a runner
                    chaserWins1: false, //win 1 game as a chaser
                        chaserWins2: false, //win 10 games as a chaser
                            chaserWins3: false, //win 100 games as a chaser
                                chaserWins4: false, //win 500 games as a chaser
                                    gamesWonAsTheHunter1: false, //win 10 games as The Hunter
                                        gamesWonAsTheHunter2: false, //win 100 games as The Hunter
                                            gamesWonAsPacman1: false, //win 10 games as Pacman
                                                gamesWonAsPacman2: false, //win 100 games as Pacman
                                                    gamesWonAsKirby1: false, //win 10 games as Kirby
                                                        gamesWonAsKirby2: false, //win 100 games as Kirby
                                                            gamesWonAsNinja1: false, //win 10 games as Ninja
                                                                gamesWonAsNinja2: false, //win 100 games as Ninja
                                                                    gamesWonAsDerp1: false, //win 10 games as Derp
                                                                        gamesWonAsDerp2: false, //win 100 games as Derp
                                                                            gamesWonAsRobot1: false, //win 10 games as Robot
                                                                                gamesWonAsRobot2: false, //win 100 games as
                                                                                    gamesWonAsSonic1: false, //win 10 games
                                                                                        gamesWonAsSonic2: false, //win 100 g
                                                                                            gamesWonAsSekiro1: false, //win
                                                                                                gamesWonAsSekiro2: false, //
                                                                                                    gamesLostAsSonic: false,
                                                                                                        gamesLostAsRobot: fa
```

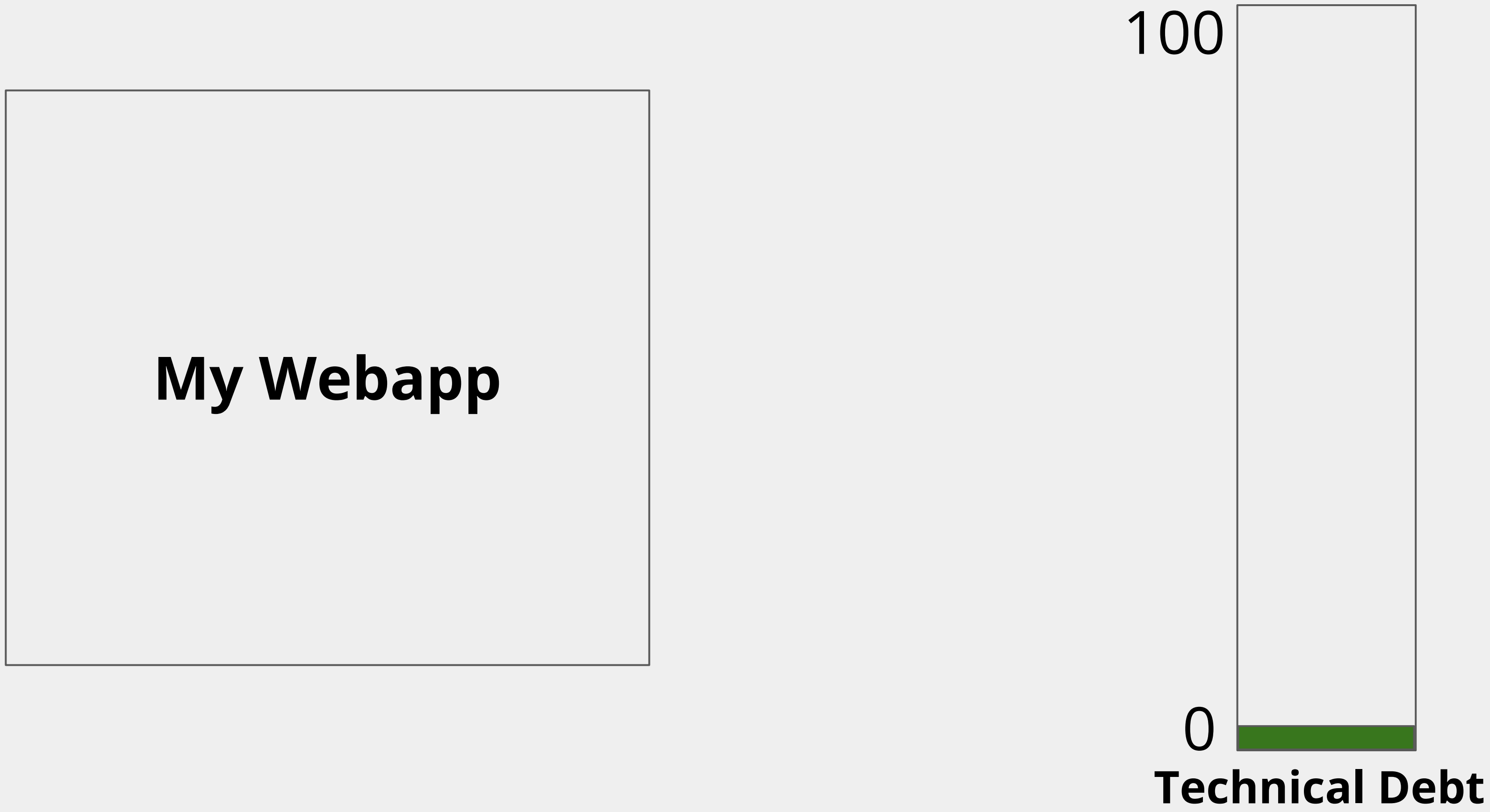
SOMETHING SMELLS
YOUR CODE
imgflip.com

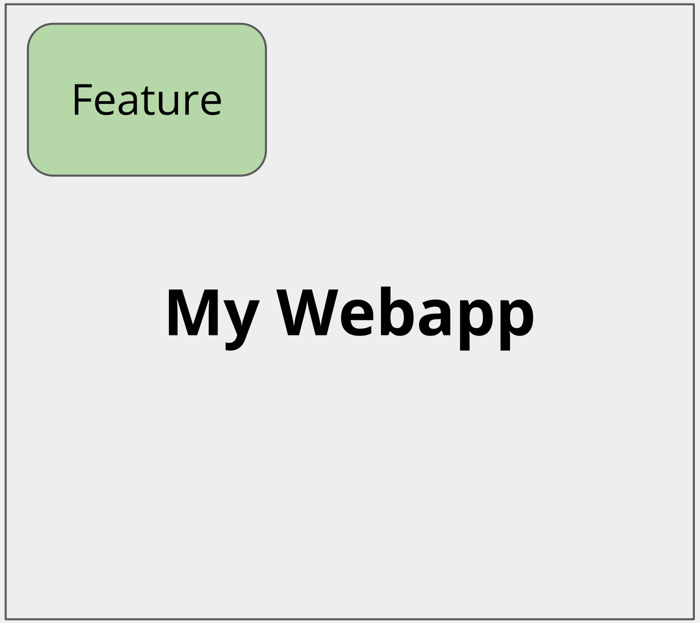
Technical Debt

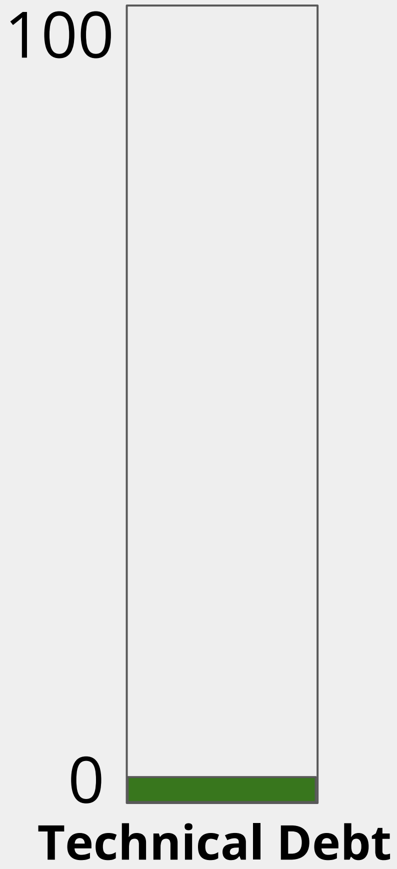
- When you choose a **quick hack** instead of a better solution
- Causes:
  - Lack of time
  - Laziness
  - Developer doesn’t know any better :(
- Creates tech “debt” you may need to pay off later…

My Webapp
100
0
Technical Debt

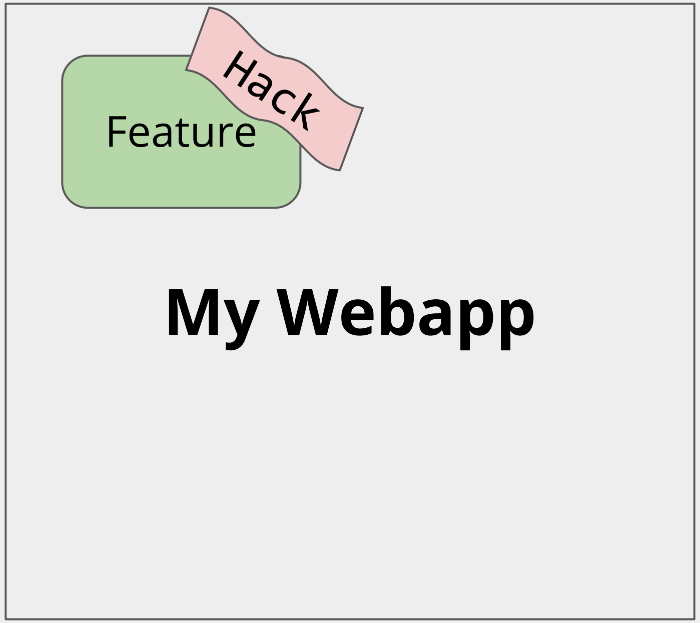
I’m gonna make feature X

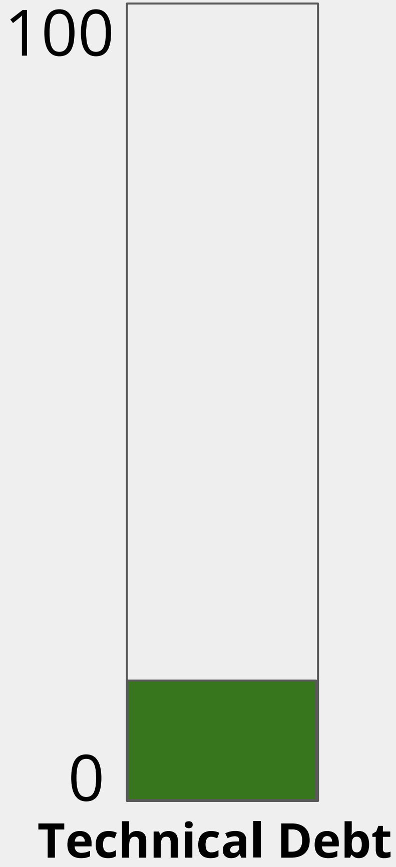



Feature
My Webapp

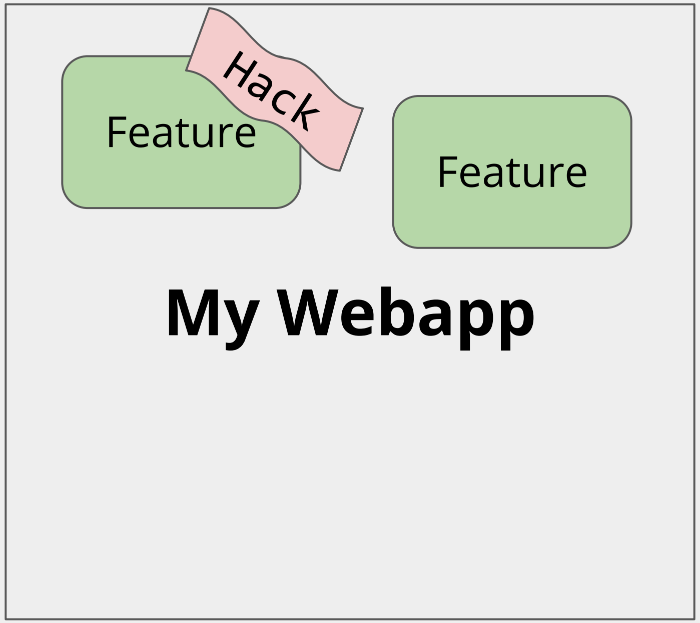

100
0
Technical Debt

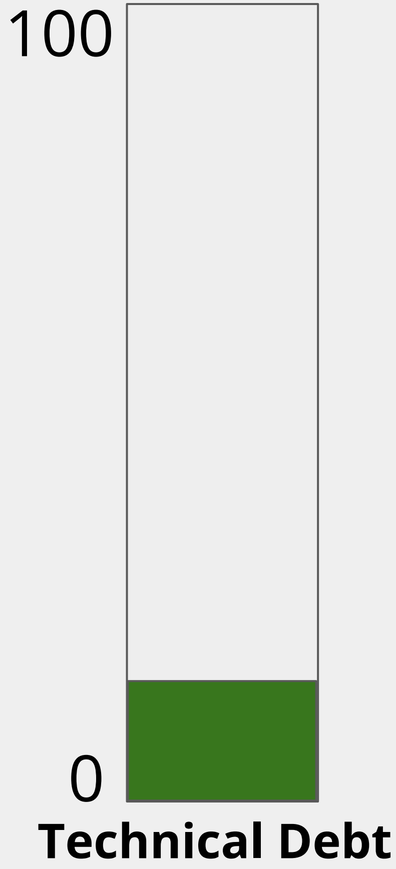
It’s not quite right… let me add a hacky fix



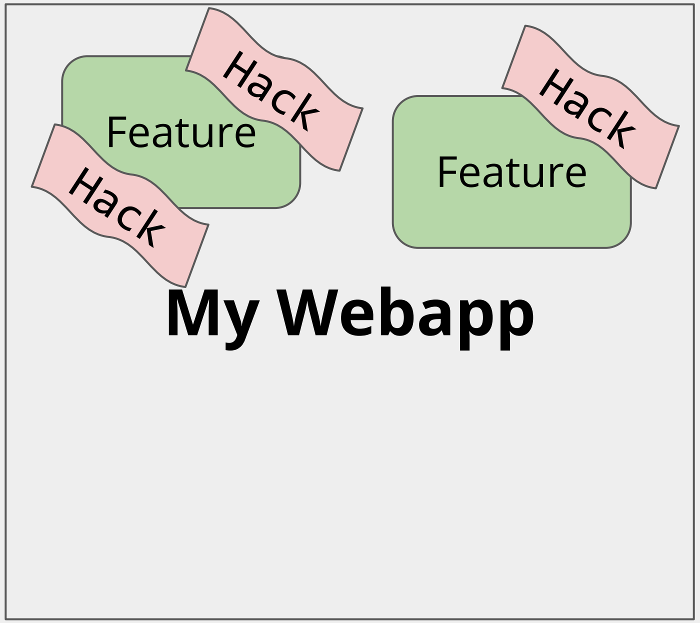


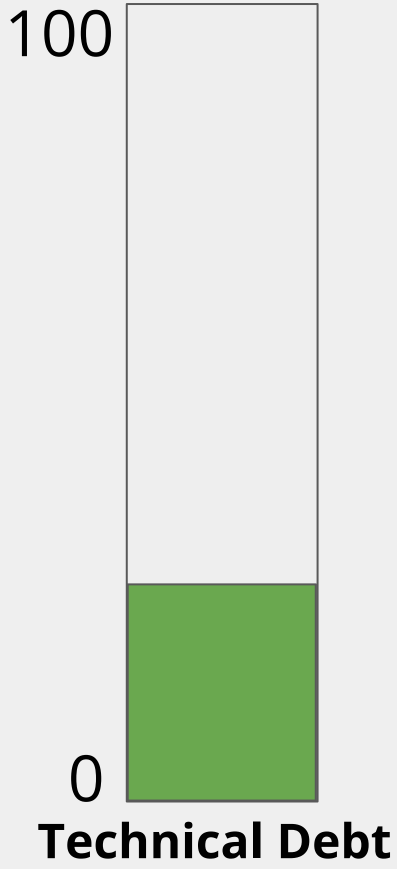
Let’s add some more features

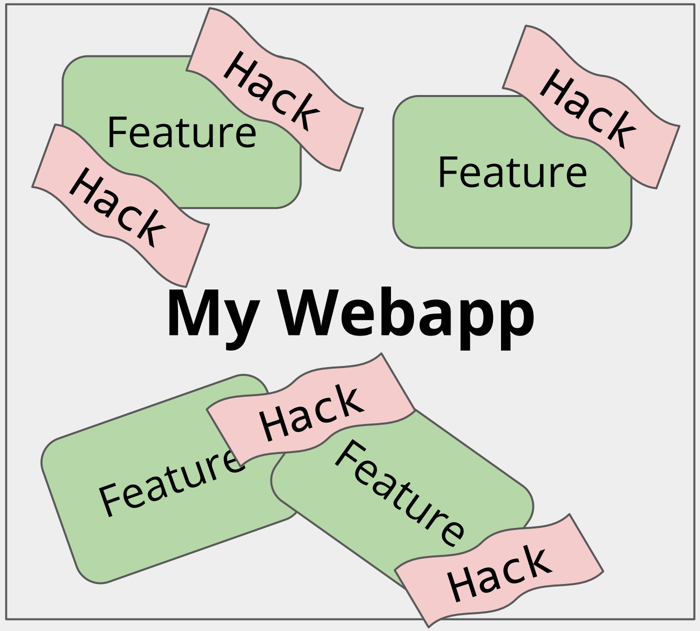


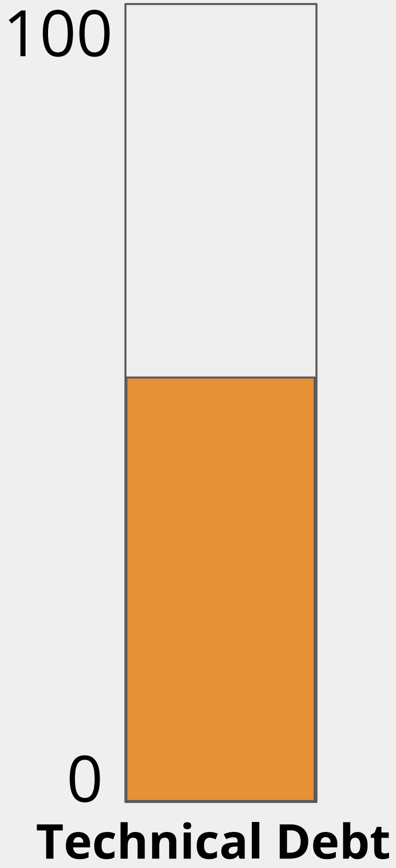


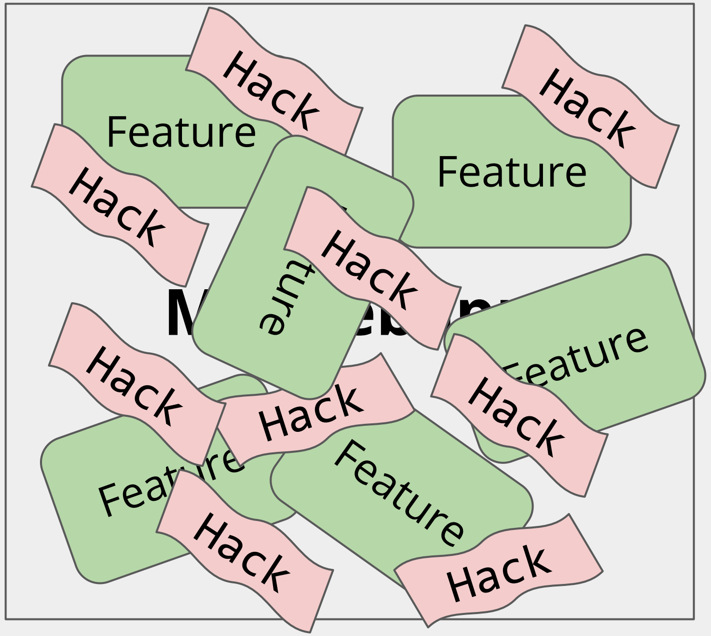
I need some more hacks...

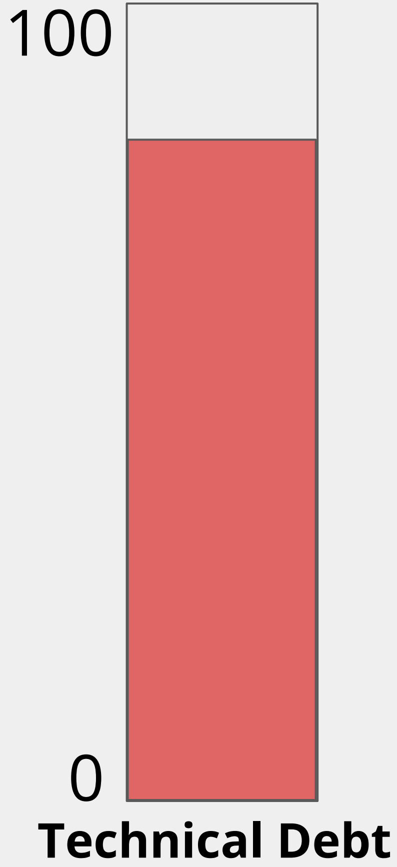




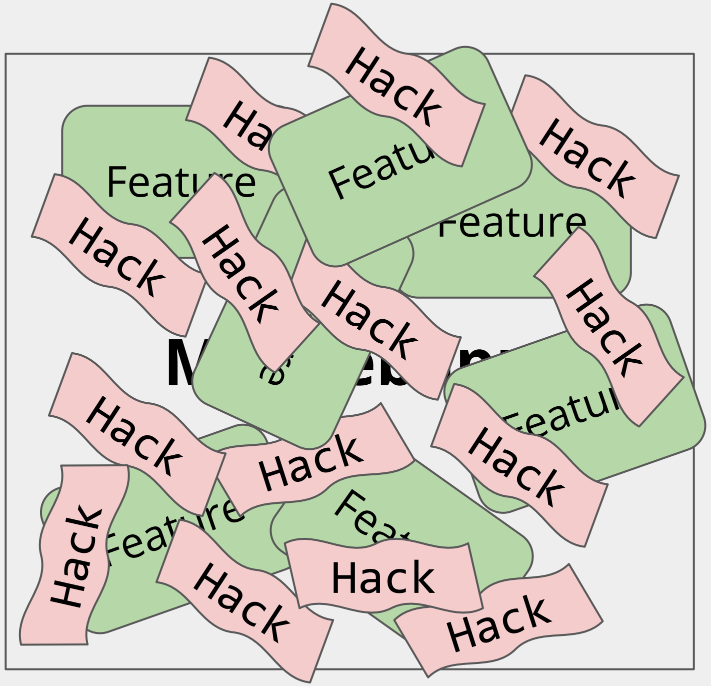

Hack
Feature
Hack
Hack
Feature
My Webapp
Hack
Feature
Feature
Hack

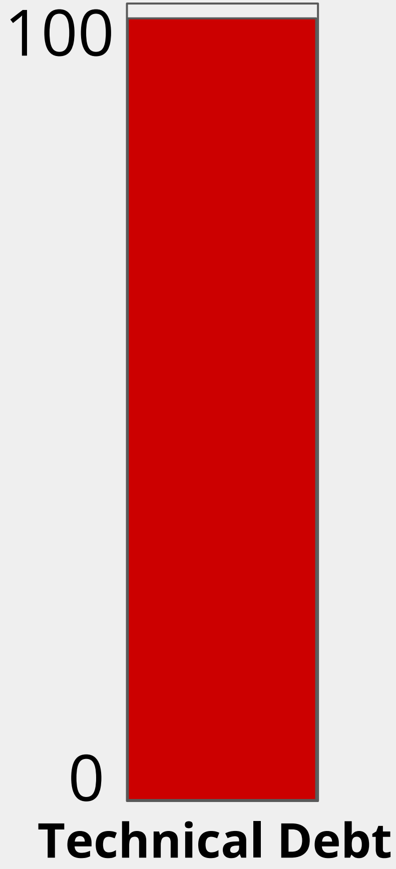

100
0
Technical Debt


Feature
Hack
Feature
Hack
Hack
Hack
Feature
Feature
Hack
Hack
Feature
Feature
Hack
Hack
Hack

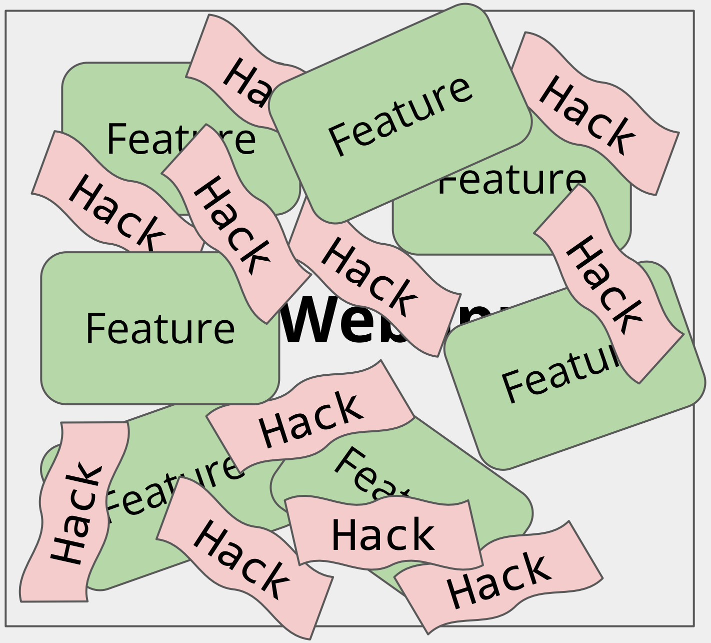

100
0
Technical Debt

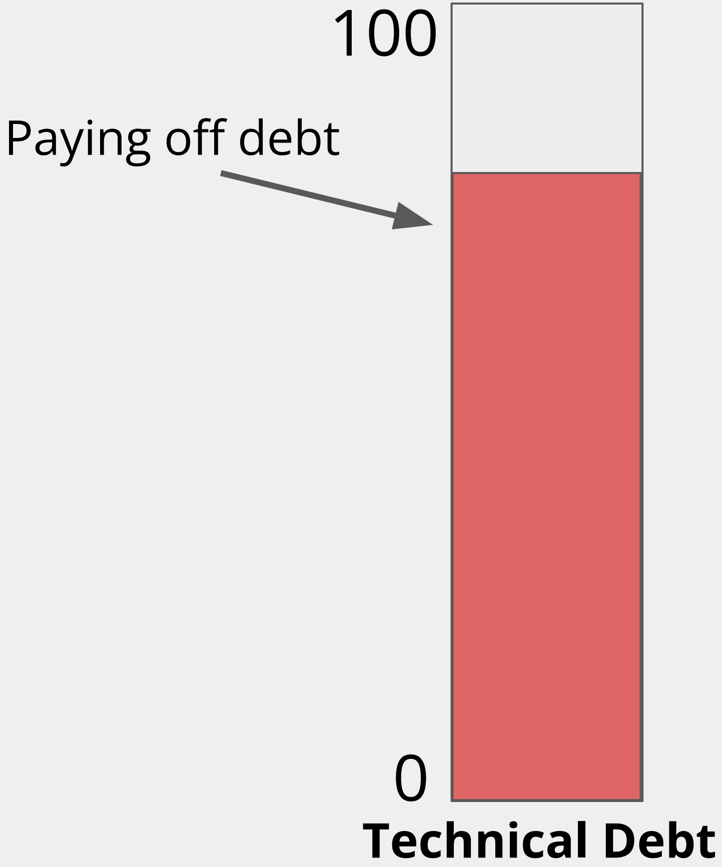
Code is so bad I can't make any more progress!

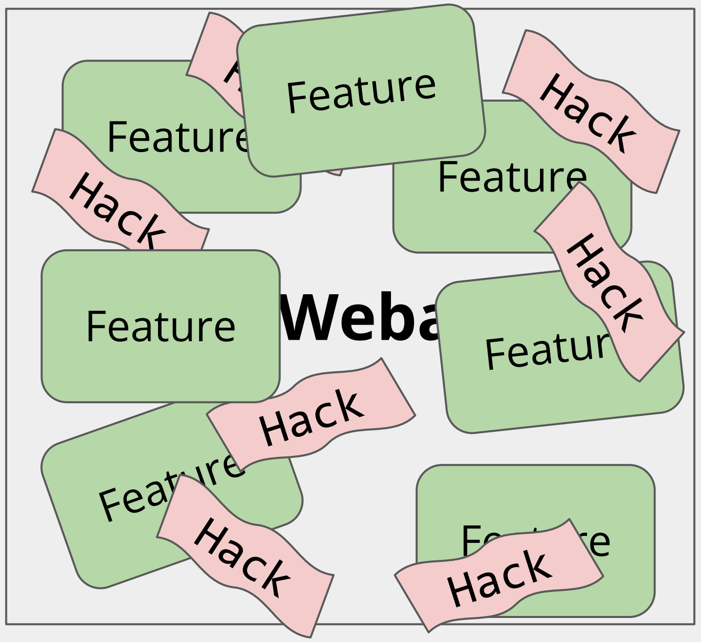


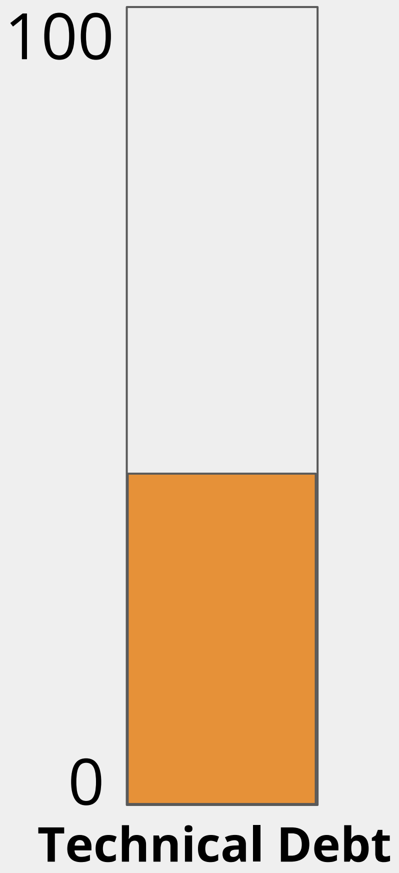


Have no choice but to refactor...






Feature
Feature
Feature
Feature
Feature
Feature
Feature
Hack
Hack
Hack
Hack
Hack
Hack
Hack
Weba

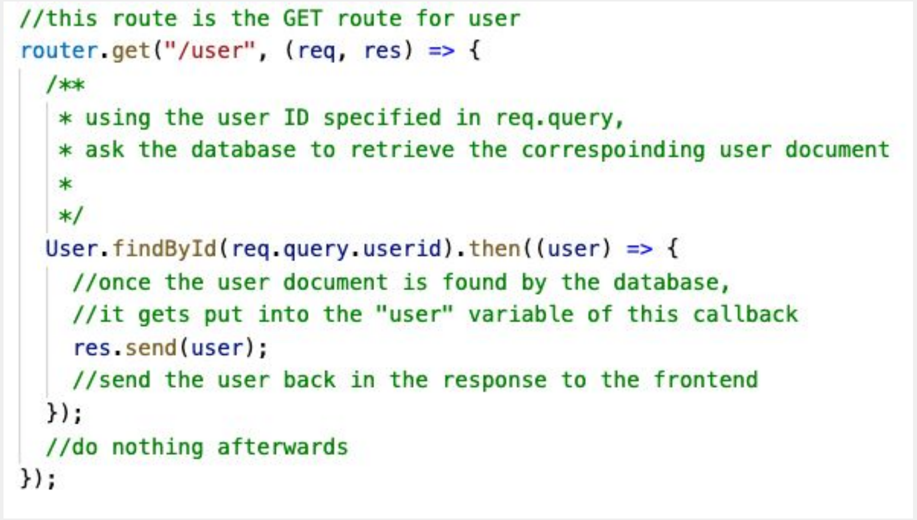

100
0
Technical Debt

Paying off technical debt

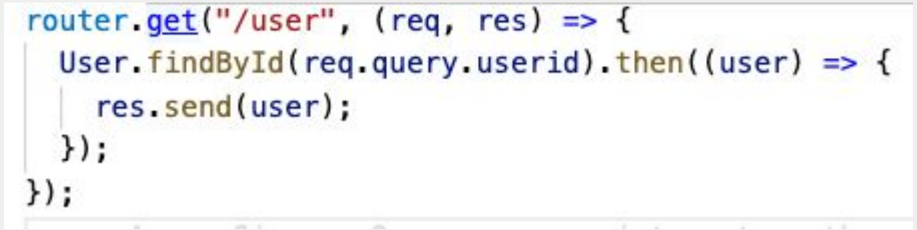
- Accrues **interest** over time
- Would have just been easier to do it the right way from the start…
- Extremely important for projects that need to be **maintained**
- For projects or prototypes that won’t have a long lifespan, it doesn’t matter as much

Comments

Which version is better?

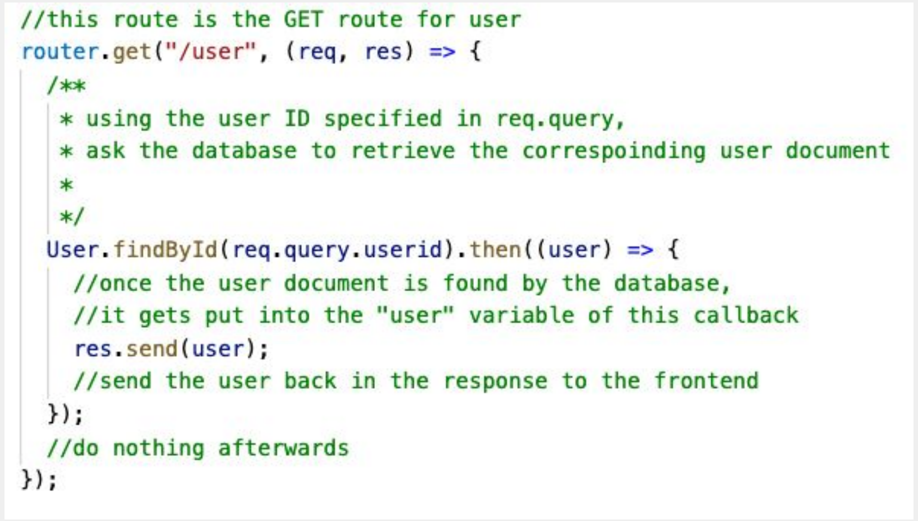


```
//this route is the GET route for user
router.get("/user", (req, res) => {
  /**
   * using the user ID specified in req.query,
   * ask the database to retrieve the correspoinding user document
   *
   */
  User.findById(req.query.userid).then((user) => {
    //once the user document is found by the database,
    //it gets put into the "user" variable of this callback
    res.send(user);
    //send the user back in the response to the frontend
  });
  //do nothing afterwards
});
```

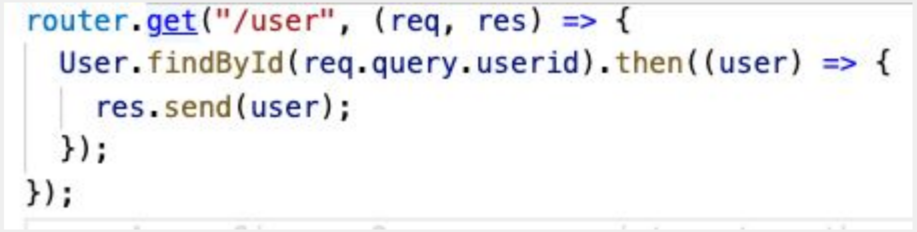


```
router.get("/user", (req, res) => {
  User.findById(req.query.userid).then((user) => {
    res.send(user);
  });
});
```

Which version is better?



```
//this route is the GET route for user
router.get("/user", (req, res) => {
  /**
   * using the user ID specified in req.query,
   * ask the database to retrieve the correspoinding user document
   *
   */
  User.findById(req.query.userid).then((user) => {
    //once the user document is found by the database,
    //it gets put into the "user" variable of this callback
    res.send(user);
    //send the user back in the response to the frontend
  });
  //do nothing afterwards
});
```

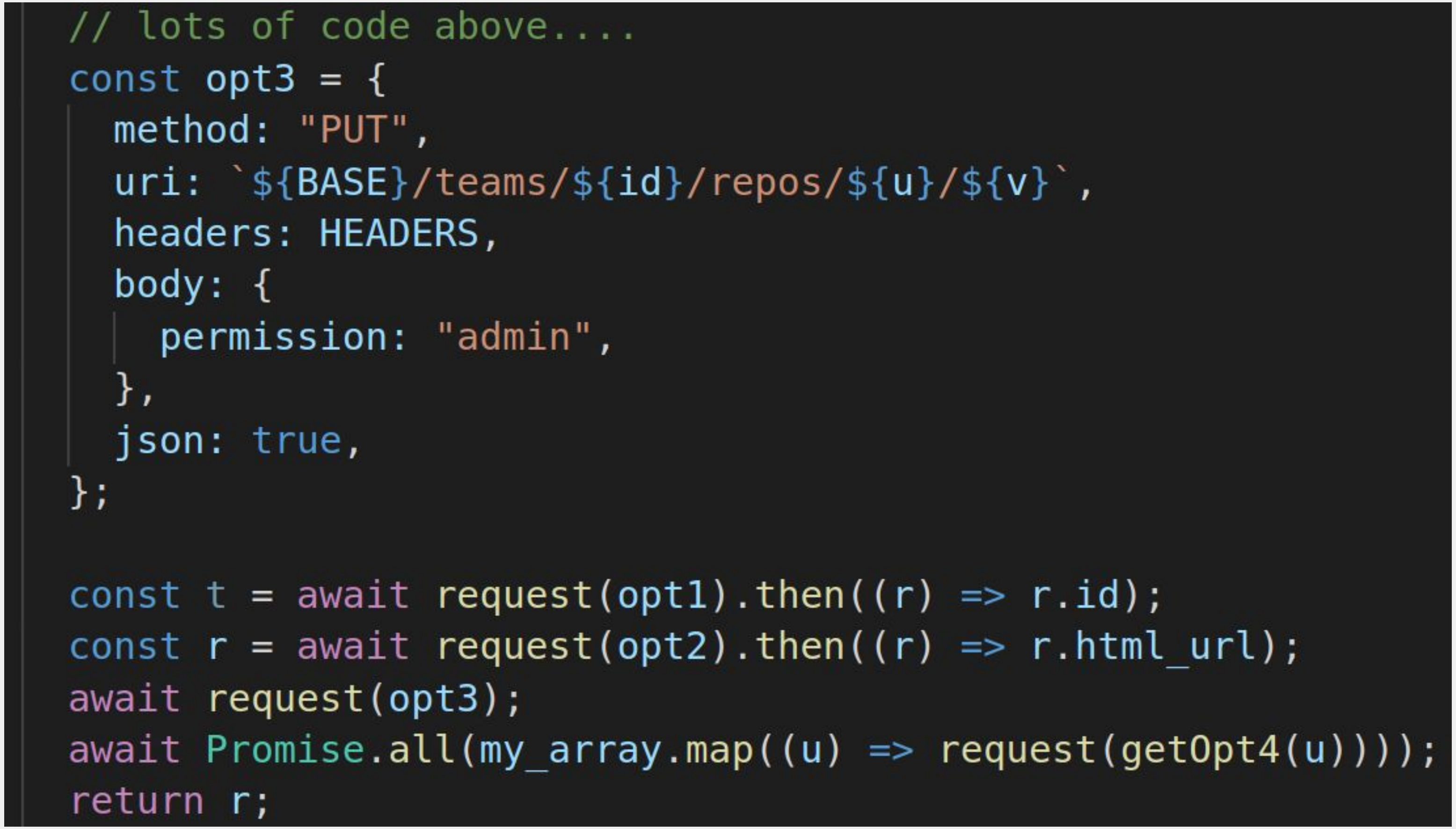


```
router.get("/user", (req, res) => {
  User.findById(req.query.userid).then((user) => {
    res.send(user);
  });
});
```

I like this one

Comments are for explaining why you coded something the way you did, not how you did it.

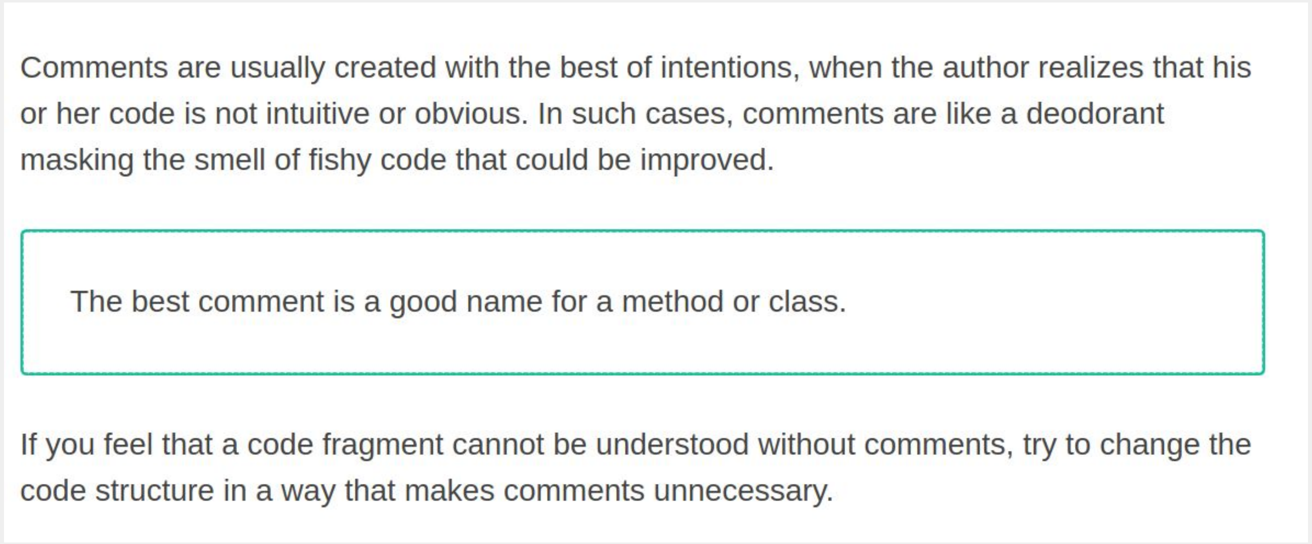
This is hard to understand… should I add comments?



```
// lots of code above....
const opt3 = {
  method: "PUT",
  uri: `${BASE}/teams/${id}/repos/${u}/${v}`,
  headers: HEADERS,
  body: {
    permission: "admin",
  },
  json: true,
};

const t = await request(opt1).then((r) => r.id);
const r = await request(opt2).then((r) => r.html_url);
await request(opt3);
await Promise.all(my_array.map((u) => request(getOpt4(u))));
return r;
```

And sometimes…



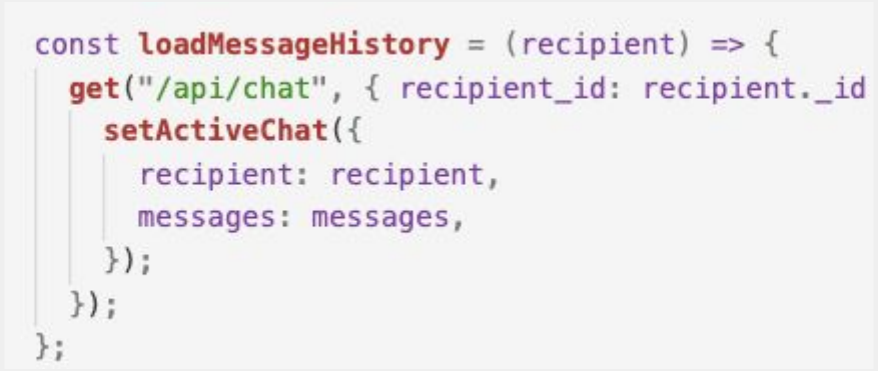
Comments are usually created with the best of intentions, when the author realizes that his or her code is not intuitive or obvious. In such cases, comments are like a deodorant masking the smell of fishy code that could be improved.

The best comment is a good name for a method or class.

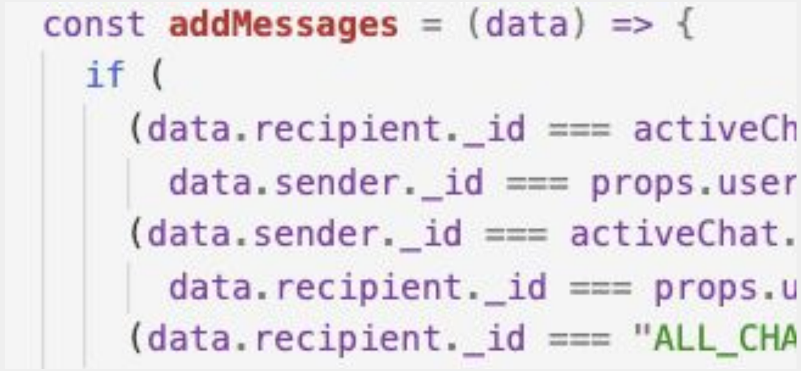
If you feel that a code fragment cannot be understood without comments, try to change the code structure in a way that makes comments unnecessary.

See the Typescript Google style guide, or the 6.1020 [6.031] reading on Code Review for more

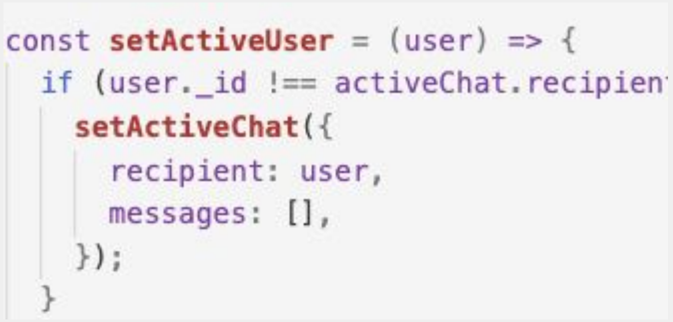
Examples of Clear Function Names



```
const loadMessageHistory = (recipient) => {
  get("/api/chat", { recipient_id: recipient._id
    setActiveChat({
      recipient: recipient,
      messages: messages,
    });
  });
};
```



```
const addMessages = (data) => {
  if (
    (data.recipient._id === activeCh
      data.sender._id === props.user
    (data.sender._id === activeChat.
      data.recipient._id === props.u
    (data.recipient._id === "ALL_CHA
```



```
const setActiveUser = (user) => {
  if (user._id !== activeChat.recipien
    setActiveChat({
      recipient: user,
      messages: [],
    });
  }
```

Questions?

Git Hygiene

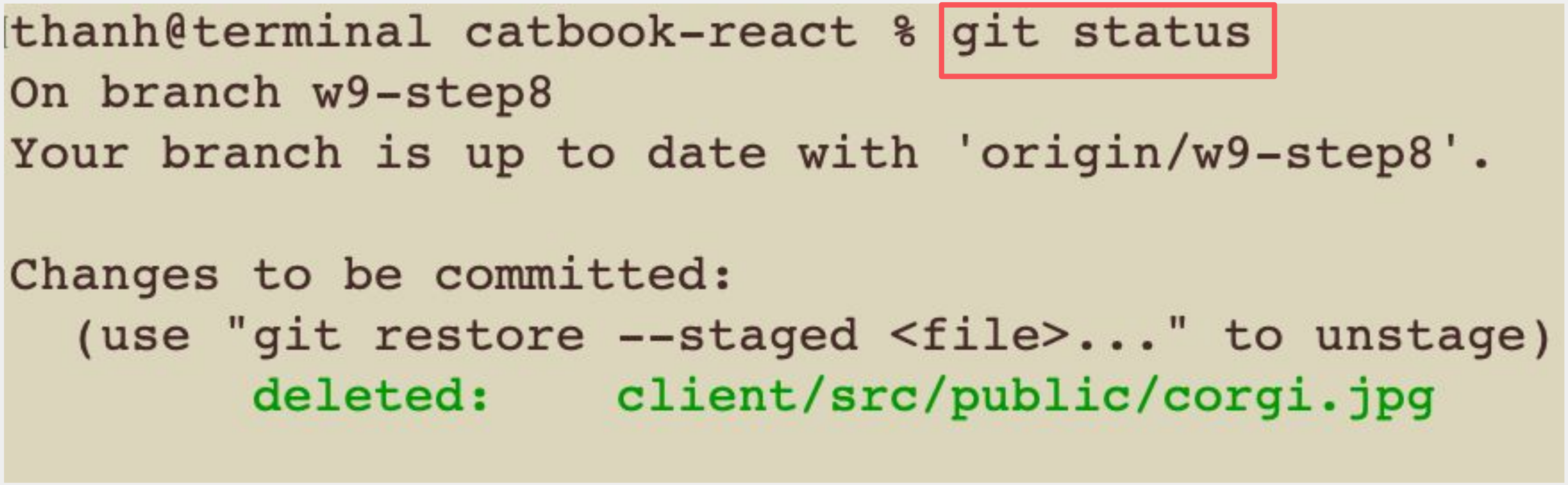
Or… how to remain on good terms with your teammates >_<

Before you start working

- `git pull`
  - Loads your teammates’ changes
  - Do it frequently to avoid merge conflicts

Before you commit

- Before running `git add / git commit`
- Review what changes you've made with `git status`

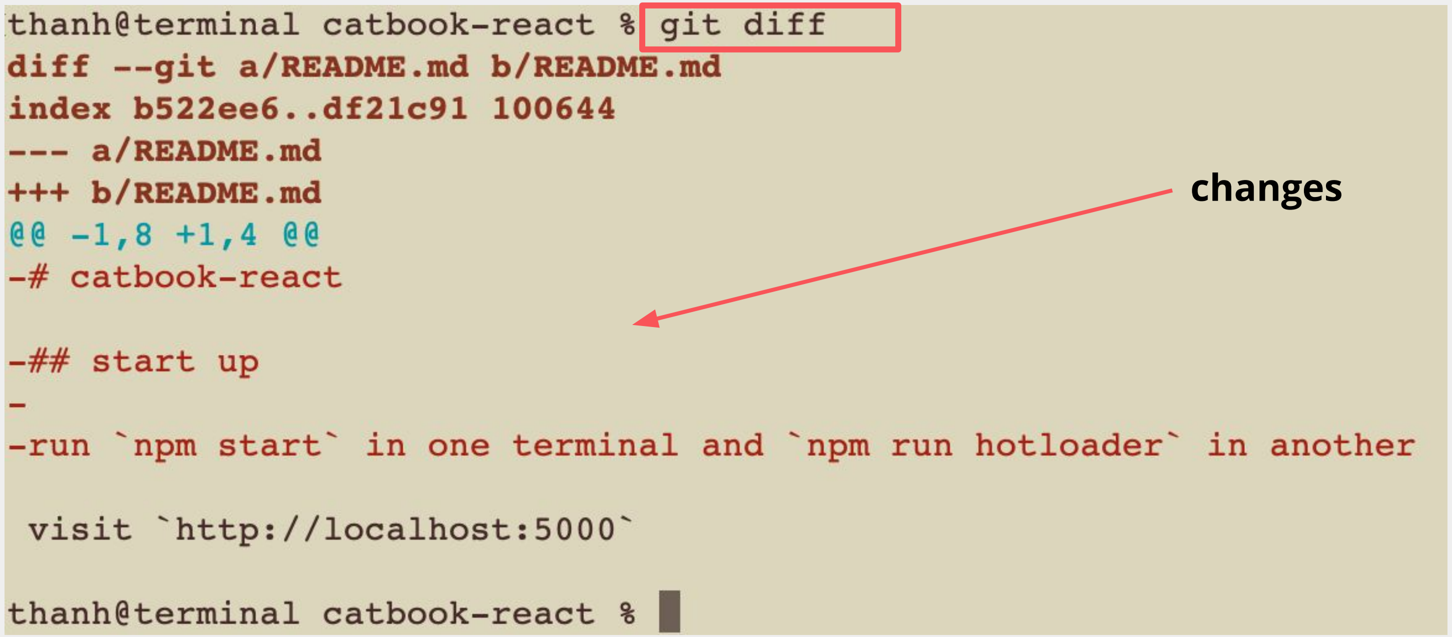


```
thanh@terminal catbook-react % git status
On branch w9-step8
Your branch is up to date with 'origin/w9-step8'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        deleted:    client/src/public/corgi.jpg
```

Before you commit

- Optionally, review your changes more thoroughly with `git diff`



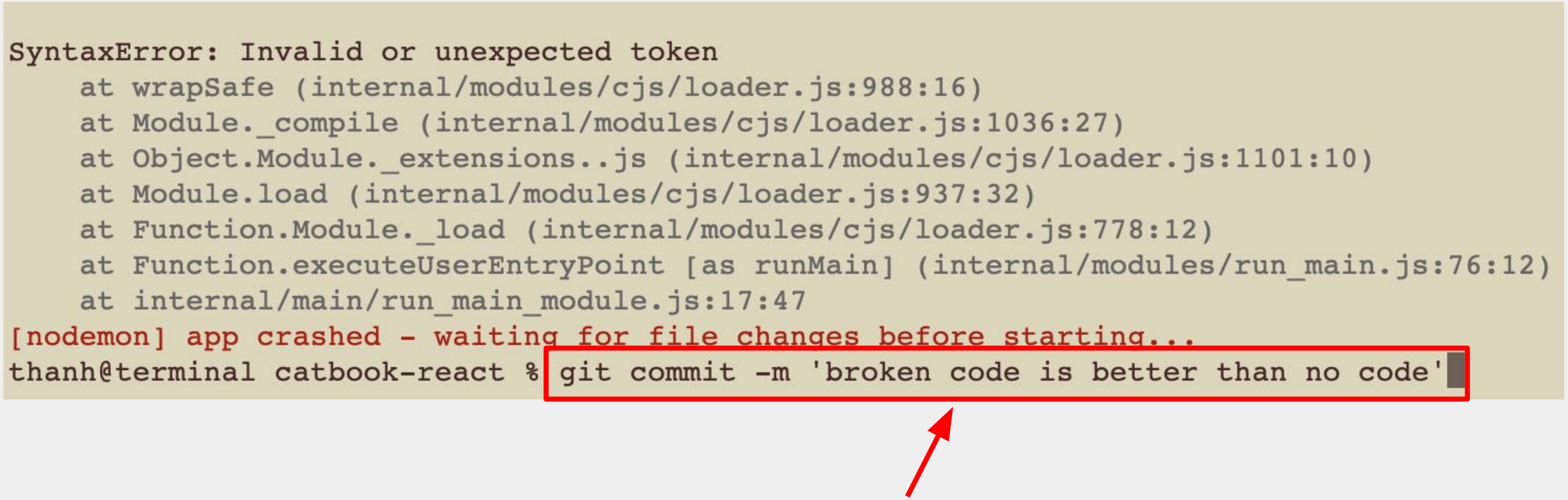
```
thanh@terminal catbook-react % git diff
diff --git a/README.md b/README.md
index b522ee6..df21c91 100644
--- a/README.md
+++ b/README.md
@@ -1,8 +1,4 @@
-# catbook-react

-## start up
-
-run `npm start` in one terminal and `npm run hotloader` in another

 visit `http://localhost:5000`

thanh@terminal catbook-react % 
```

Before you commit

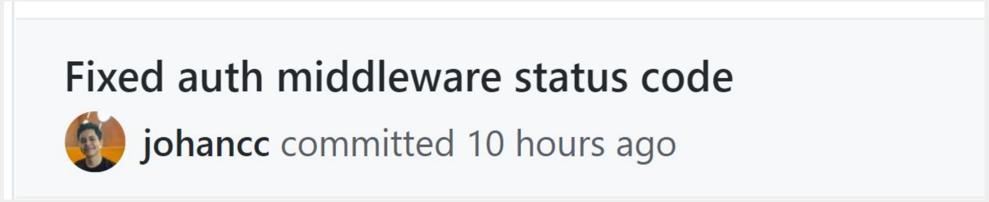


```
SyntaxError: Invalid or unexpected token
    at wrapSafe (internal/modules/cjs/loader.js:988:16)
    at Module._compile (internal/modules/cjs/loader.js:1036:27)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:1101:10)
    at Module.load (internal/modules/cjs/loader.js:937:32)
    at Function.Module._load (internal/modules/cjs/loader.js:778:12)
    at Function.executeUserEntryPoint [as runMain] (internal/modules/run_main.js:76:12)
    at internal/main/run_main_module.js:17:47
[nodemon] app crashed - waiting for file changes before starting...
thanh@terminal catbook-react % git commit -m 'broken code is better than no code'
```

DO NOT!

When to commit

- When you add something and it works, then commit
  - More often is generally better than less often

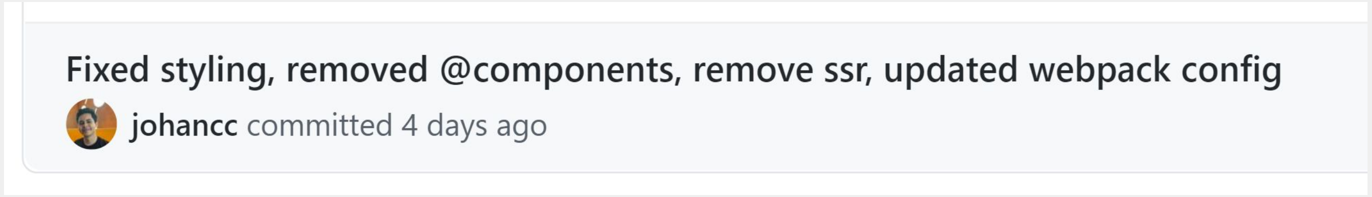


Fixed auth middleware status code

johancc committed 10 hours ago

Commits add a single thing, is focused/clear, and is consistent with how your team/company wants their commit messages

- Avoid mega-commits



Fixed styling, removed @components, remove ssr, updated webpack config

johancc committed 4 days ago

Commit tries to do too much, not focused

DANGEROUS commands!

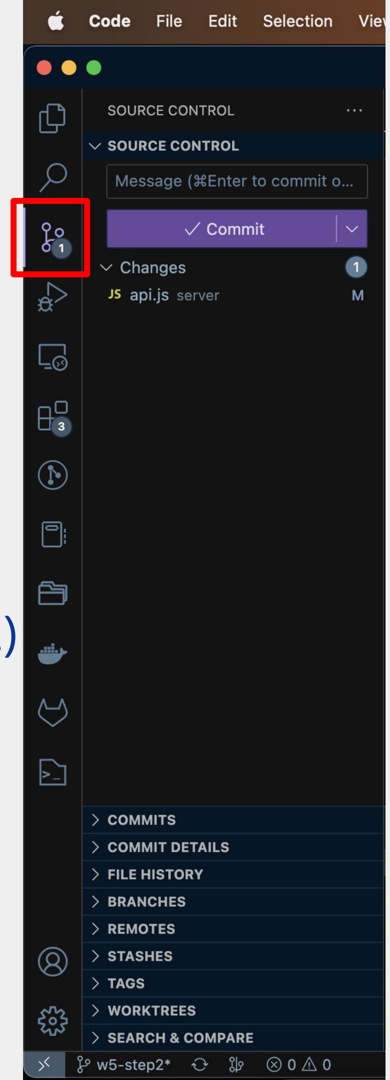
- `git add .`
  - This will add all files in the current directory to git
  - Make sure to setup .gitignore files to avoid committing node_modules or credentials
- `git push --force`
  - **NEVER USE** unless you know what you're doing
- `git reset --hard`
  - Deletes all uncommitted changes
- `git commit --amend`
  - Don't ammend commits you've already pushed
  - Make a new commit instead

Useful git commands

- `git stash`
  - git stash allows you to store your current changes
- `git stash pop`
  - git stash allows you to “pop” the most recent stored changes
- `git branch`
  - git branch allows you to “diverge” and focus on one issue
  - git checkout (used in workshops!) will then allow you to move to different branches

VSCode Git Guide

- Has lots of tabs, but the important one is source control
- Can see the branch on the bottom left and change it
- Source control will show staged and committed changes
- Can click on each file on the changes to see the diff
- Can also do merge requests in VSCode (Will see in part 2)
- If you don't like VSCode or Terminal:
  - Use the GitHub desktop app!



Proper Git Branch Usage

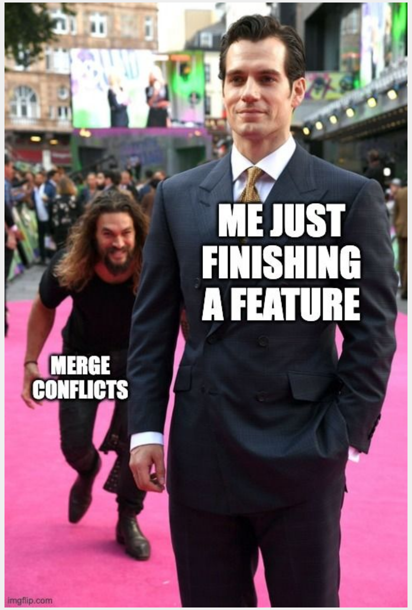
- You should create a Git branch **for every new feature that you make**
- **You should never be working on your main branch**
- Once you're done with a feature and you're sure that it works and is bug-free (hopefully), you can **use a PR (pull request)** and **merge the branch** to get your work on main

What is a PR (Pull Request)?

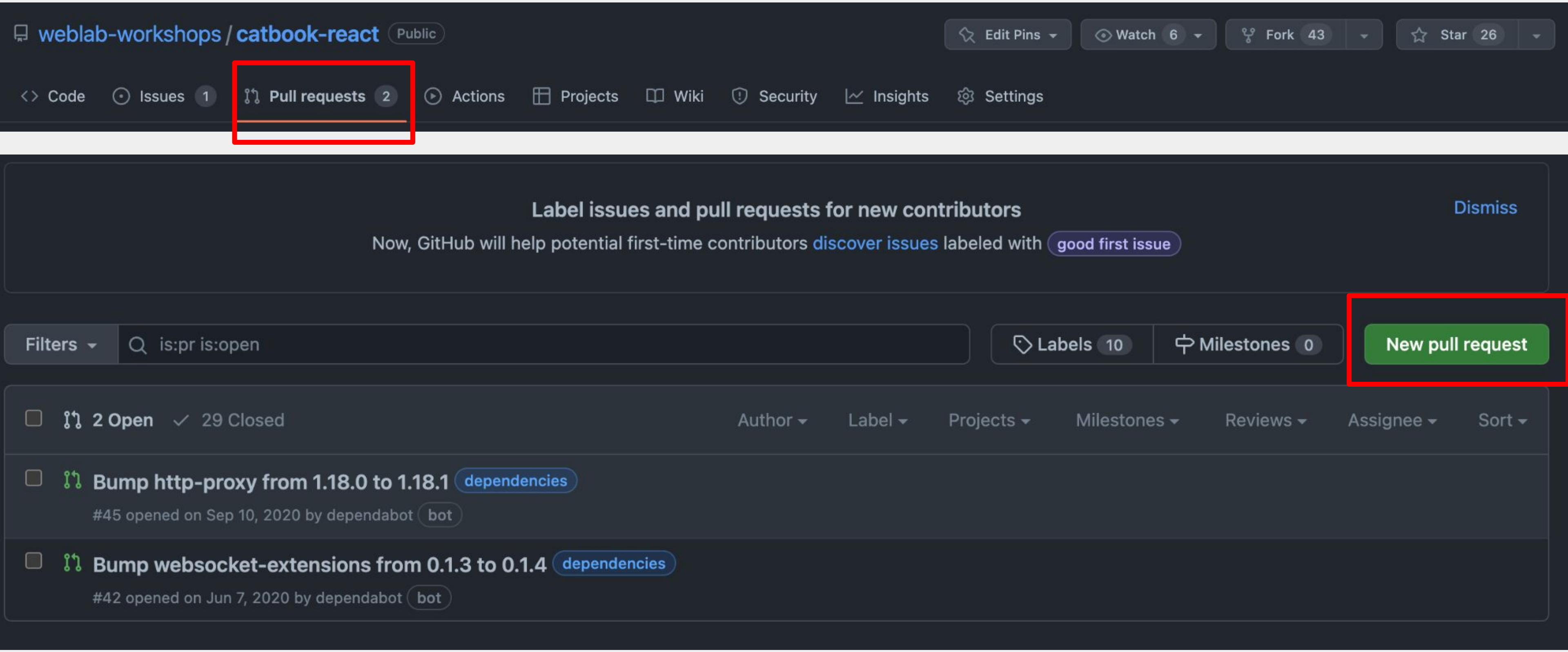
- A pull request is a request that someone makes for the owner of a branch to “pull” the changes from one branch onto another
- Used to **sync changes from and compare one branch to another**
- Differences in these two branches may result in **merge conflicts**
- Can be created in the terminal with **git request-pull**

What is a Merge Conflict?

- A **merge conflict** happens when you there is a conflict between two files of code, usually on different code branches
- The way to resolve merge conflicts is to “merge” both files into the new combined file
- Let’s go through an example!



How to Create and Resolve a PR on GitHub

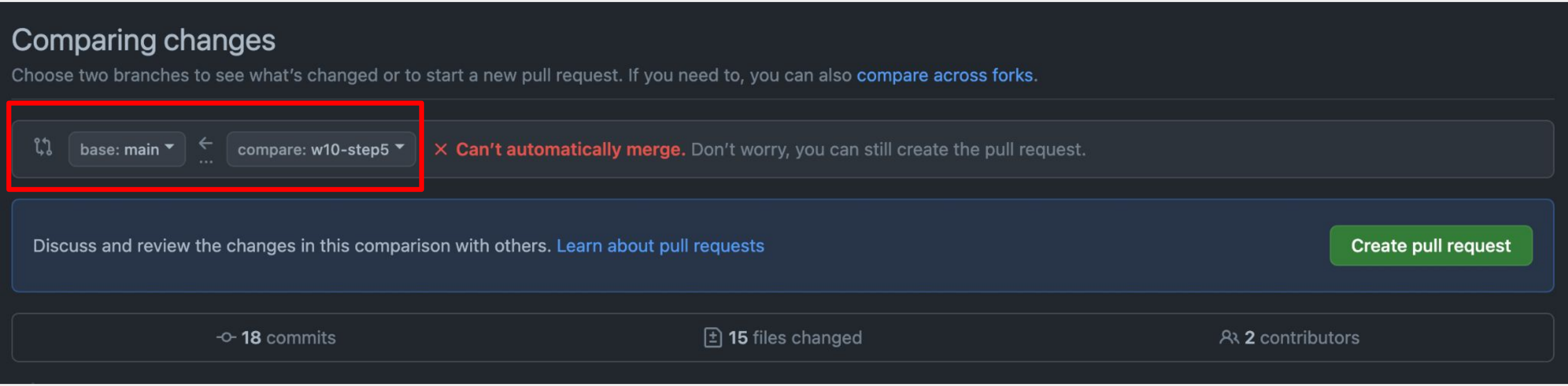


Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with good first issue

Bump http-proxy from 1.18.0 to 1.18.1 dependencies
#45 opened on Sep 10, 2020 by dependabot bot

Bump websocket-extensions from 0.1.3 to 0.1.4 dependencies
#42 opened on Jun 7, 2020 by dependabot bot



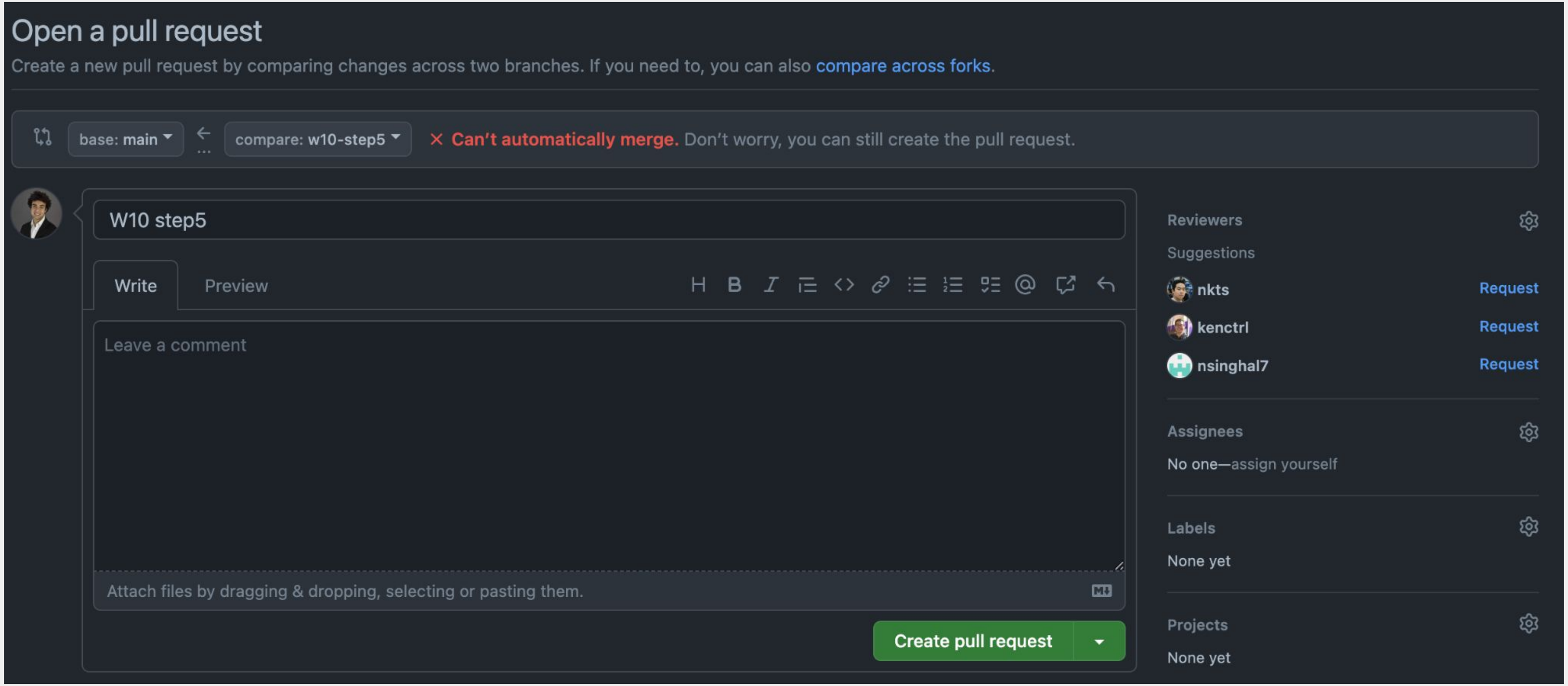
Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

base: main ← compare: w10-step5 ✕ **Can't automatically merge.** Don't worry, you can still create the pull request.

Discuss and review the changes in this comparison with others. Learn about pull requests

18 commits **15** files changed **2** contributors



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base: main ← compare: w10-step5 ✕ Can't automatically merge. Don't worry, you can still create the pull request.

W10 step5

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

Reviewers

Suggestions

nkts Request

kenctrl Request

nsinghal7 Request

Assignees

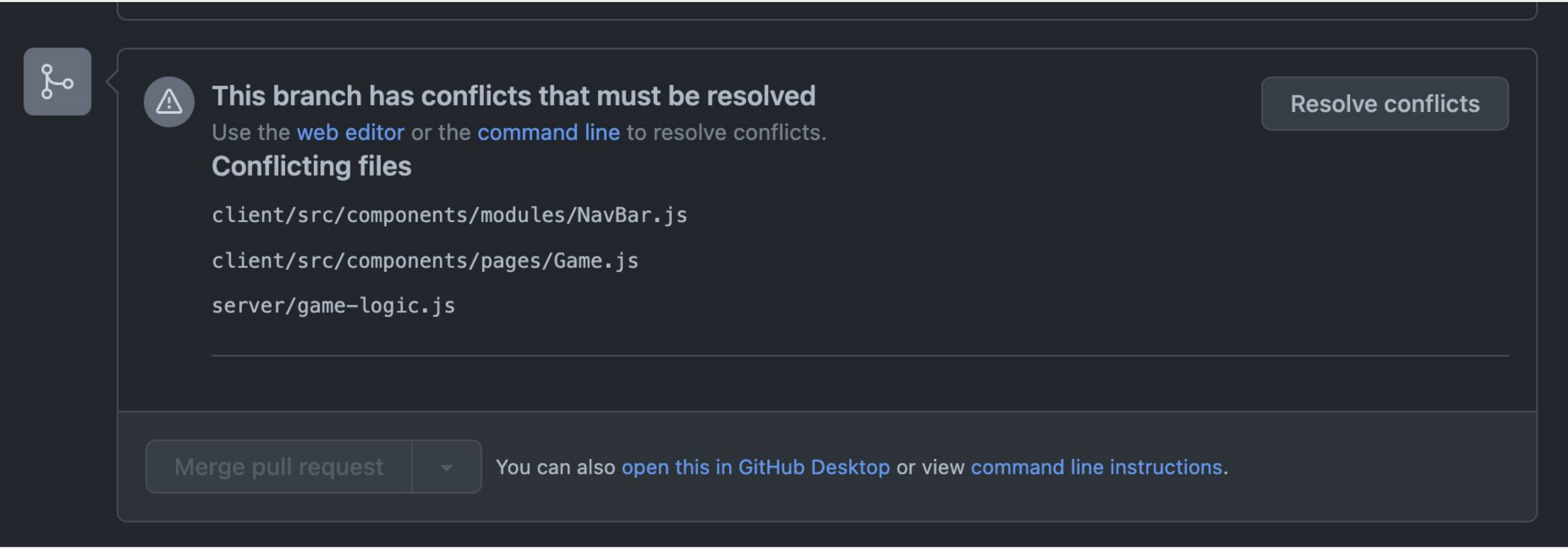
No one—assign yourself

Labels

None yet

Projects

None yet



This branch has conflicts that must be resolved

Use the web editor or the command line to resolve conflicts.

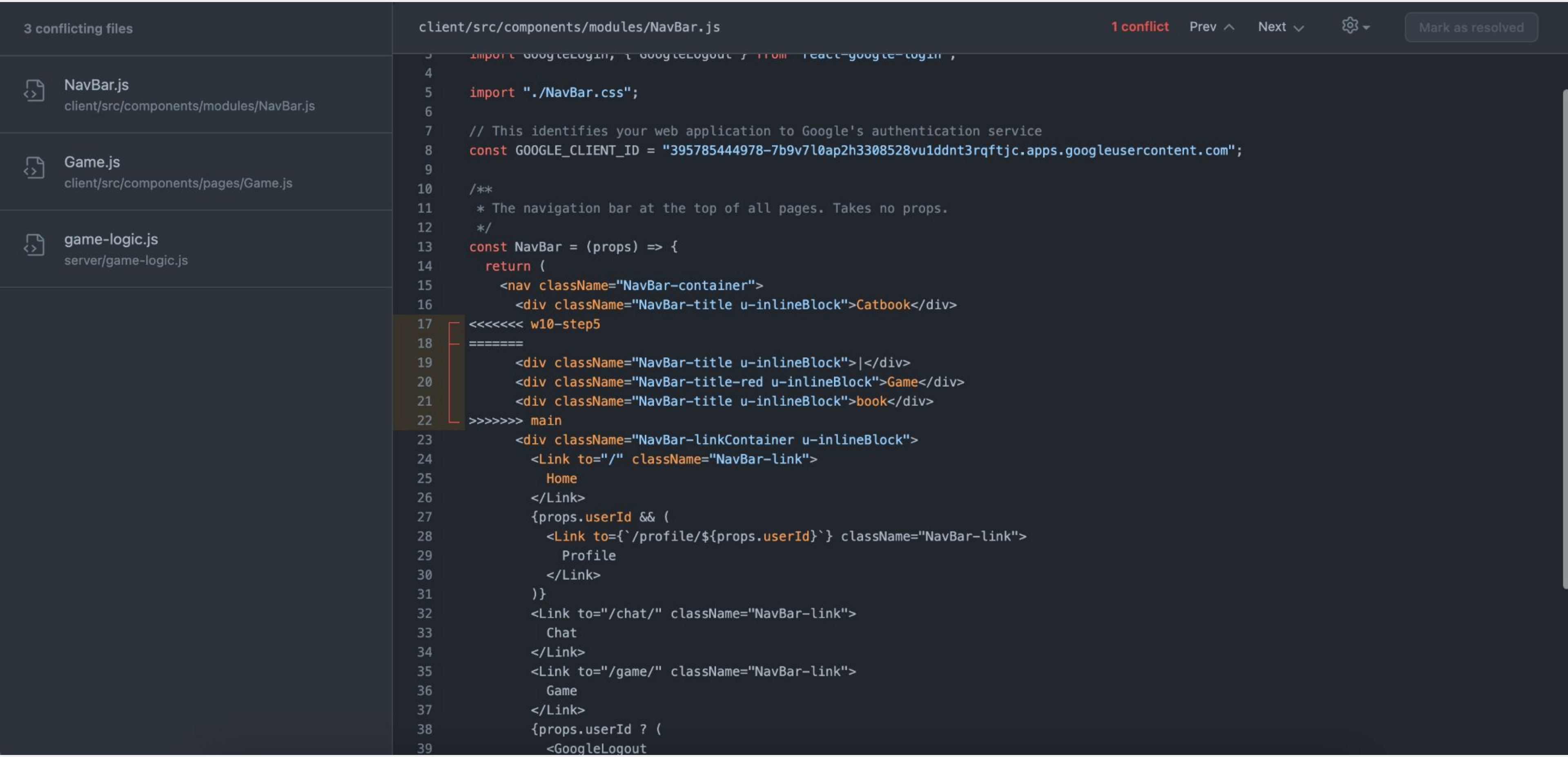
Conflicting files

```
client/src/components/modules/NavBar.js

client/src/components/pages/Game.js

server/game-logic.js
```

You can also open this in GitHub Desktop or view command line instructions.


3 conflicting files

NavBar.js
client/src/components/modules/NavBar.js

Game.js
client/src/components/pages/Game.js

game-logic.js
server/game-logic.js

client/src/components/modules/NavBar.js

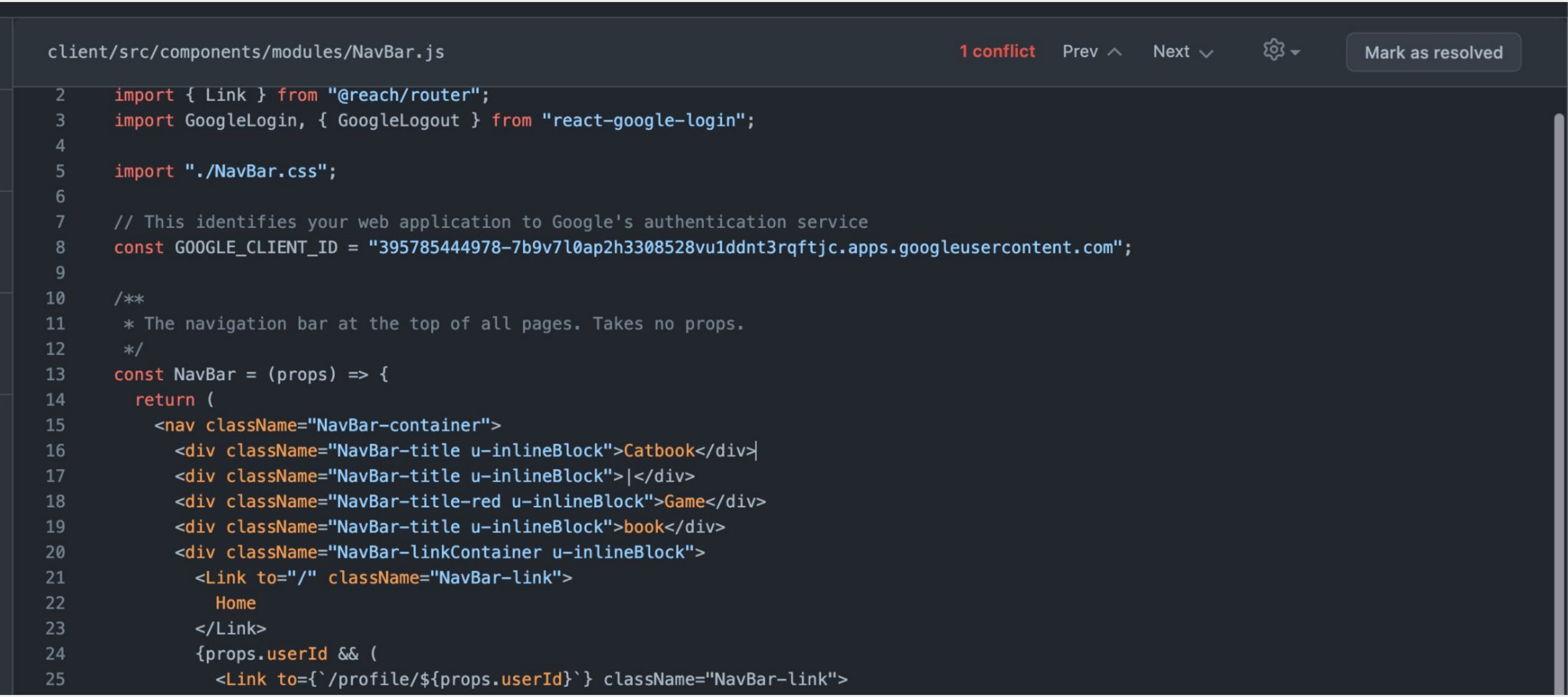
1 conflict Prev Next Mark as resolved

```
import GoogleLogin, { GoogleLogout } from "react-google-login";

import "./NavBar.css";

// This identifies your web application to Google's authentication service
const GOOGLE_CLIENT_ID = "395785444978-7b9v7l0ap2h3308528vu1ddnt3rqftjc.apps.googleusercontent.com";

/**
 * The navigation bar at the top of all pages. Takes no props.
 */
const NavBar = (props) => {
  return (
    <nav className="NavBar-container">
      <div className="NavBar-title u-inlineBlock">Catbook</div>
<<<<<<< w10-step5
=======
      <div className="NavBar-title u-inlineBlock">|</div>
      <div className="NavBar-title-red u-inlineBlock">Game</div>
      <div className="NavBar-title u-inlineBlock">book</div>
>>>>>>> main
      <div className="NavBar-linkContainer u-inlineBlock">
        <Link to="/" className="NavBar-link">
          Home
        </Link>
        {props.userId && (
          <Link to={`/profile/${props.userId}`} className="NavBar-link">
            Profile
          </Link>
        )}
        <Link to="/chat/" className="NavBar-link">
          Chat
        </Link>
        <Link to="/game/" className="NavBar-link">
          Game
        </Link>
        {props.userId ? (
          <GoogleLogout
```



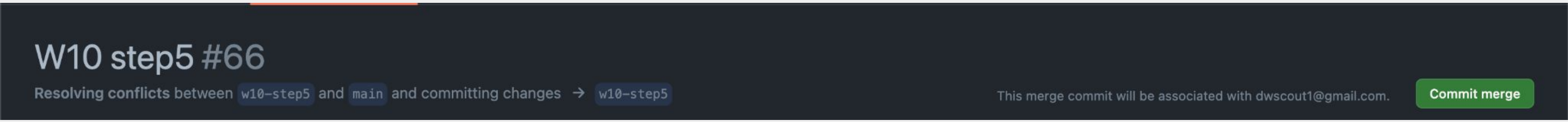
```
import { Link } from "@reach/router";
import GoogleLogin, { GoogleLogout } from "react-google-login";

import "./NavBar.css";

// This identifies your web application to Google's authentication service
const GOOGLE_CLIENT_ID = "395785444978-7b9v7l0ap2h3308528vu1ddnt3rqftjc.apps.googleusercontent.com";

/**
 * The navigation bar at the top of all pages. Takes no props.
 */
const NavBar = (props) => {
  return (
    <nav className="NavBar-container">
      <div className="NavBar-title u-inlineBlock">Catbook</div>
      <div className="NavBar-title u-inlineBlock">|</div>
      <div className="NavBar-title-red u-inlineBlock">Game</div>
      <div className="NavBar-title u-inlineBlock">book</div>
      <div className="NavBar-linkContainer u-inlineBlock">
        <Link to="/" className="NavBar-link">
          Home
        </Link>
        {props.userId && (
          <Link to={`/profile/${props.userId}`} className="NavBar-link">
```

After Merging



W10 step5 #66

Resolving conflicts between `w10-step5` and `main` and committing changes → `w10-step5`

This merge commit will be associated with dwscout1@gmail.com.

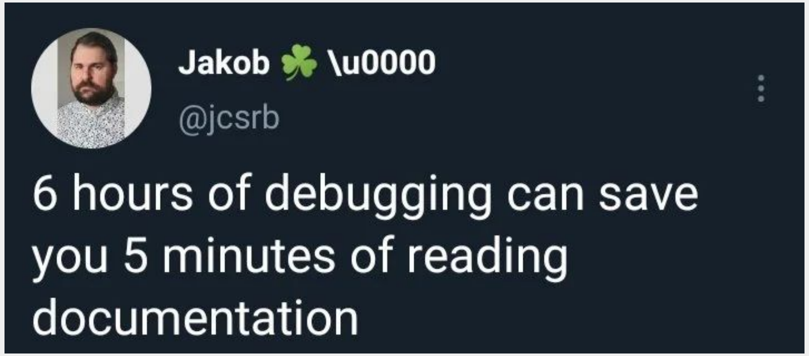
- After any merge conflicts are resolved, you can close the pull request and have the option of deleting the comparison branch
- Congratulations! You've successfully merged your changes onto main!

For more on git, check out abby and tony's lecture here!

Testing

Testing

- **If you don’t test your code, it will eventually break.**
- Most basic form of testing (in web development) is **UI testing**
- Tests for individual components of code (functions, classes, etc) are called **unit tests**
- Tests for how individual components interact with each other are called **integration tests**
- Tests to make sure that behavior stays the same after adding new features are called **regression tests**
- A complete **testing suite** has all of these and more (see 6.031 reading)
- Javascript testing tool (“Jest”): weblab.is/testing
- More on testing:
    - 6.031 [6.102] 😳



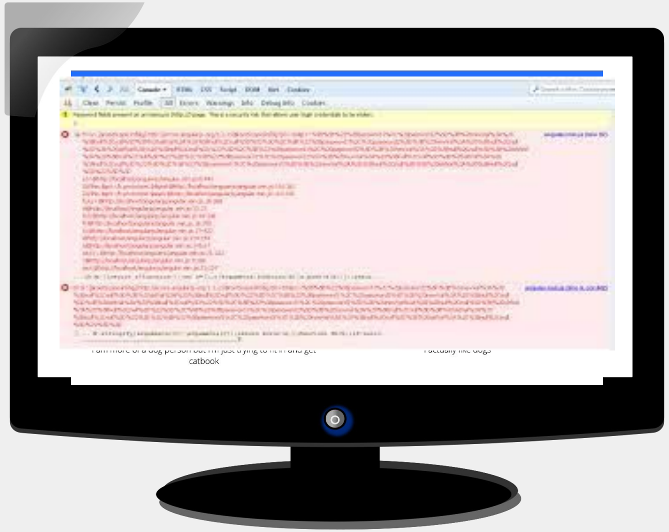
Jakob ☘️ \u0000

@jcsrb

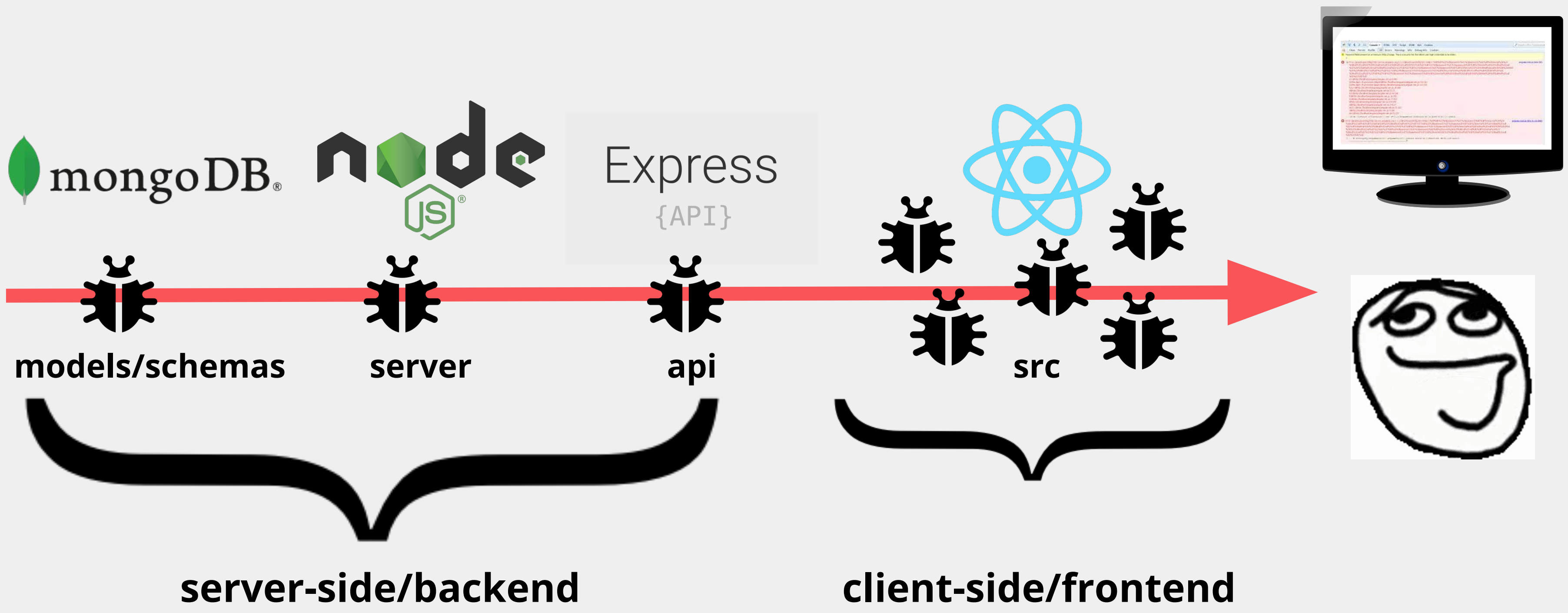
6 hours of debugging can save you 5 minutes of reading documentation

Debugging

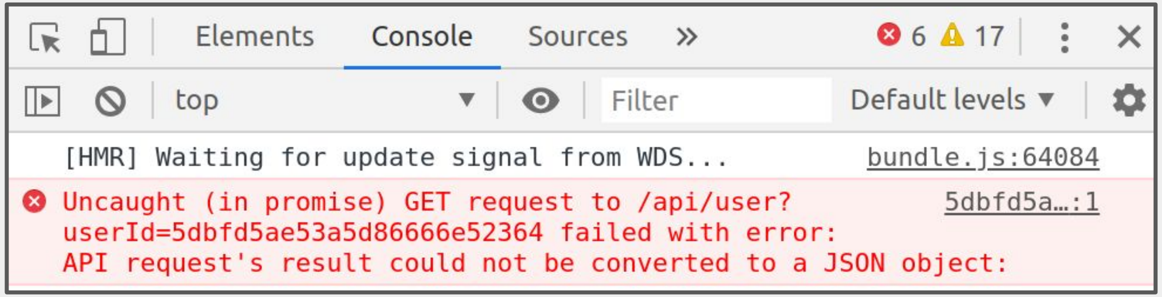
Where is my bug coming from?



Where is my bug coming from?

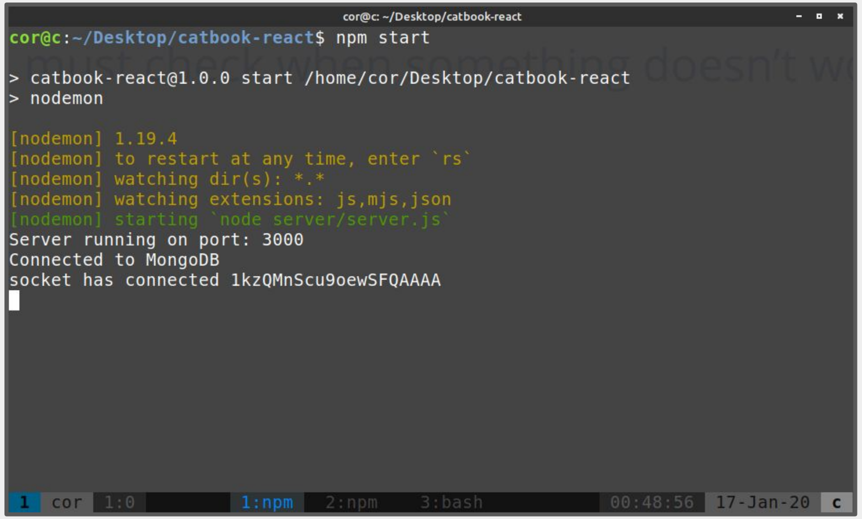


Three things you should check



```
[HMR] Waiting for update signal from WDS...                     bundle.js:64084
Uncaught (in promise) GET request to /api/user?                   5dbfd5a…:1
userId=5dbfd5ae53a5d86666e52364 failed with error:
API request's result could not be converted to a JSON object:
```

Browser console
(F12 or ⌘+⌥+j)



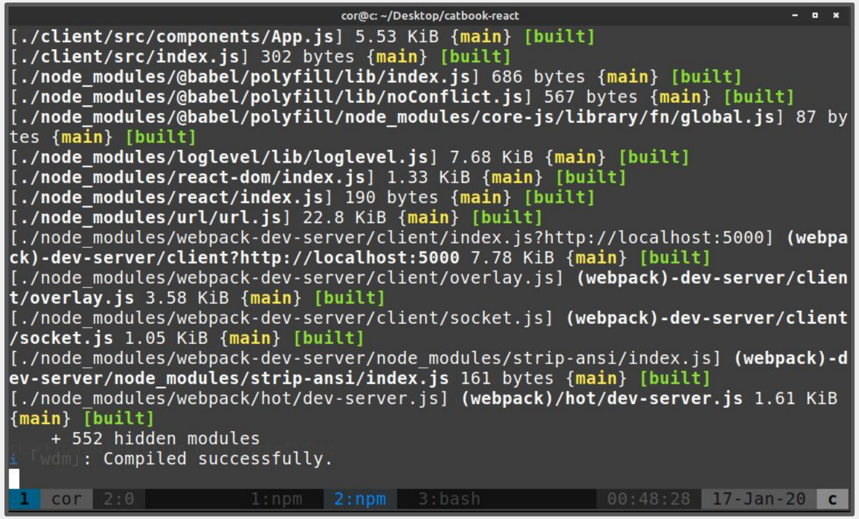
```
cor@c: ~/Desktop/catbook-react
cor@c:~/Desktop/catbook-react$ npm start

> catbook-react@1.0.0 start /home/cor/Desktop/catbook-react
> nodemon

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server/server.js`
Server running on port: 3000
Connected to MongoDB
socket has connected 1kzQMnScu9oewSFQAAAA

 1  cor  1:0        1:npm  2:npm   3:bash       00:48:56  17-Jan-20  c
```

npm start



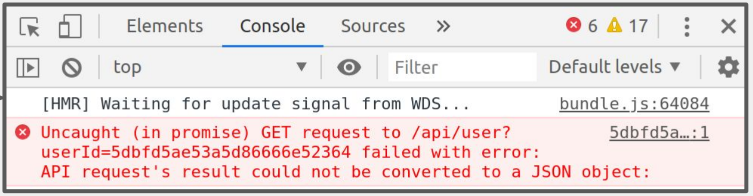
```
cor@c: ~/Desktop/catbook-react
[./client/src/components/App.js] 5.53 KiB {main} [built]
[./client/src/index.js] 302 bytes {main} [built]
[./node_modules/@babel/polyfill/lib/index.js] 686 bytes {main} [built]
[./node_modules/@babel/polyfill/lib/noConflict.js] 567 bytes {main} [built]
[./node_modules/@babel/polyfill/node_modules/core-js/library/fn/global.js] 87 by
tes {main} [built]
[./node_modules/loglevel/lib/loglevel.js] 7.68 KiB {main} [built]
[./node_modules/react-dom/index.js] 1.33 KiB {main} [built]
[./node_modules/react/index.js] 190 bytes {main} [built]
[./node_modules/url/url.js] 22.8 KiB {main} [built]
[./node_modules/webpack-dev-server/client/index.js?http://localhost:5000] (webpa
ck)-dev-server/client?http://localhost:5000 7.78 KiB {main} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/clien
t/overlay.js 3.58 KiB {main} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client
/socket.js 1.05 KiB {main} [built]
[./node_modules/webpack-dev-server/node_modules/strip-ansi/index.js] (webpack)-d
ev-server/node_modules/strip-ansi/index.js 161 bytes {main} [built]
[./node_modules/webpack/hot/dev-server.js] (webpack)/hot/dev-server.js 1.61 KiB
{main} [built]
    + 552 hidden modules
i ｢wdm｣: Compiled successfully.

 1  cor  2:0        1:npm  2:npm   3:bash       00:48:28  17-Jan-20  c
```

npm run hotloader

In React code/frontend

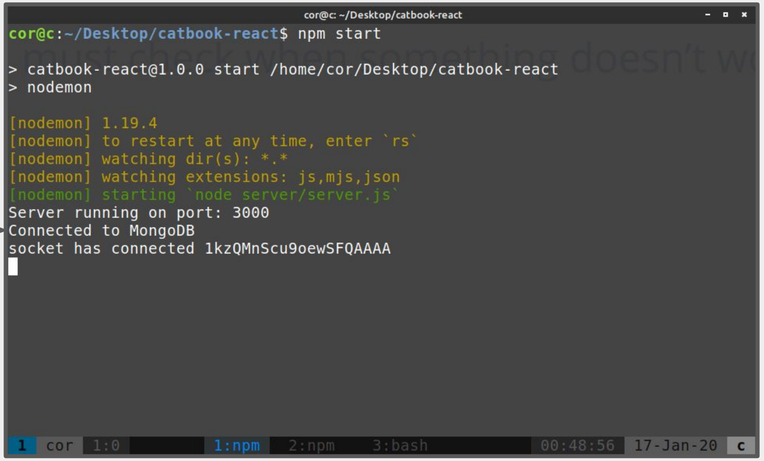
```
console.log("henlo");
```



```
[HMR] Waiting for update signal from WDS...                    bundle.js:64084
Uncaught (in promise) GET request to /api/user?                  5dbfd5a…:1
userId=5dbfd5ae53a5d86666e52364 failed with error:
API request's result could not be converted to a JSON object:
```

In server code/backend

```
console.log("henlo");
```



```
cor@c:~/Desktop/catbook-react$ npm start

> catbook-react@1.0.0 start /home/cor/Desktop/catbook-react
> nodemon

[nodemon] 1.19.4
[nodemon] to restart at any time, enter `rs`
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node server/server.js`
Server running on port: 3000
Connected to MongoDB
socket has connected 1kzQMnScu9oewSFQAAAA

 1  cor  1:0        1:npm   2:npm   3:bash          00:48:56  17-Jan-20  c
```

Finally…Fail Fast!

Never be in this situation

- I finally finished writing 500 lines of code
- I open up http://localhost:5050 to test my code and...
- Nothing works! What do?

Option A:

Option B:

Web.Lab Debugging Checklist

You’ve got a bug and your website doesn’t work as intended! What to do?

1. Check your browser console and see if there are any errors there
2. **READ ANY ERROR MESSAGES**
3. Check your terminal running npm hotloader
4. **READ ANY ERROR MESSAGES**
5. Check your terminal running npm start
6. **READ ANY ERROR MESSAGES**
7. Google any error messages that you may find
8. Ask your team for help
9. Ask staff for hep
10. Cry

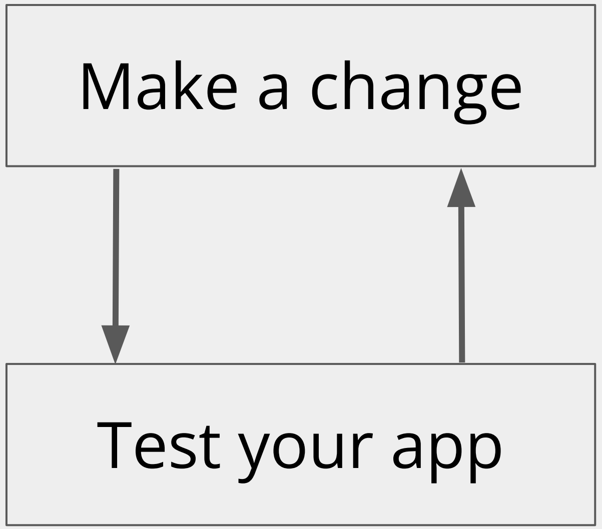
WHEN YOU GO THROUGH THE ENTIRE CHECKLIST AND YOU STILL CANT FIX YOUR BUG
imgflip.com

More lines of code = More to debug

- conservation of bugs

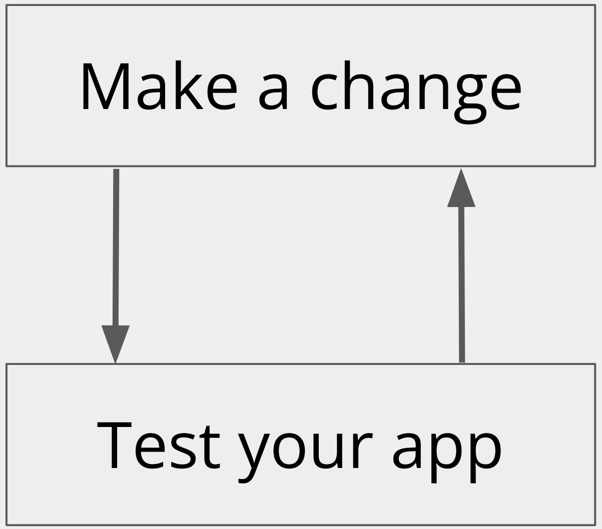
A short development/debug loop

- Make **small, incremental** changes
- Keep functions short and modular
- Test at every opportunity (e.g. `console.log` and make sure it looks right)



Optimizing the debug loop

- Shaving off a few seconds per iteration adds up
- Tools to optimize the debug loop: react hotloader, nodemon



Debugging thoroughly

- Even if your code *seems* to work, there may still be bugs...
- Users may break your app in ways you never imagined
- All team members should thoroughly try to break the site
- Get friend(s) to test your website (expect LOTS of bugs from this)
- **A stable, simple website beats a buggy, complex website**

Overall Summary

1. Git responsibly
2. Limit hacky code
3. Write code that doesn't need lots of comments
4. Develop with a tight debug loop
5. Slow down and fix your bugs
6. **COMMUNICATE WITH YOUR TEAM**

What people think programmers argue about..
fb.me/yuva.krishna.memes
Your sorting algorithm has flaws..
It's faster than quicksort.. I improvised the algorithm..
What we argue about
SAMPLE SALE
That variable should be named dateUpdated..
updatedDate looks better..

Stretch Break :)

Announcements

- You MUST complete all milestones to get credit for the class and/or be eligible for competition
- **OH Tonight 7-9 pm in 32-082**