TypeScript, turned up to 11

First things first: compiler config

- [ ] Who here uses TypeScript?
- [ ] Who has strict mode enabled?
- [ ] https://www.typescriptlang.org/tsconfig/#strict
- [ ] Cool aside, node 24 natively support .ts now!

Example in strict-false.ts

Also wanna check those indexes

- noUncheckedIndexedAccess
- https://www.typescriptlang.org/tsconfig/#noUncheckedIndexedAccess

```ts
const processResponseBad = (responses: Response[]) => {
  const firstResponse = responses[0];
  console.log(firstResponse.status);
};
```

Example in index-access.ts

Features to avoid

Non null assertion

```ts
const logUserEmailInUppercaseBad = (user: { name: string; email?: string }) => {
  console.log(user.email!.toUpperCase());
};
```

non-null-assertion.ts

Don’t cast it, narrow it!

```ts
const badUser = JSON.parse('{"name": "John"}') as {
  name: string;
  age: number;
};
```

https://www.typescriptlang.org/docs/handbook/2/narrowing.html

casting-is-bad.ts

- [ ] For more complex narrowing you can use a library like Zod or Valibot

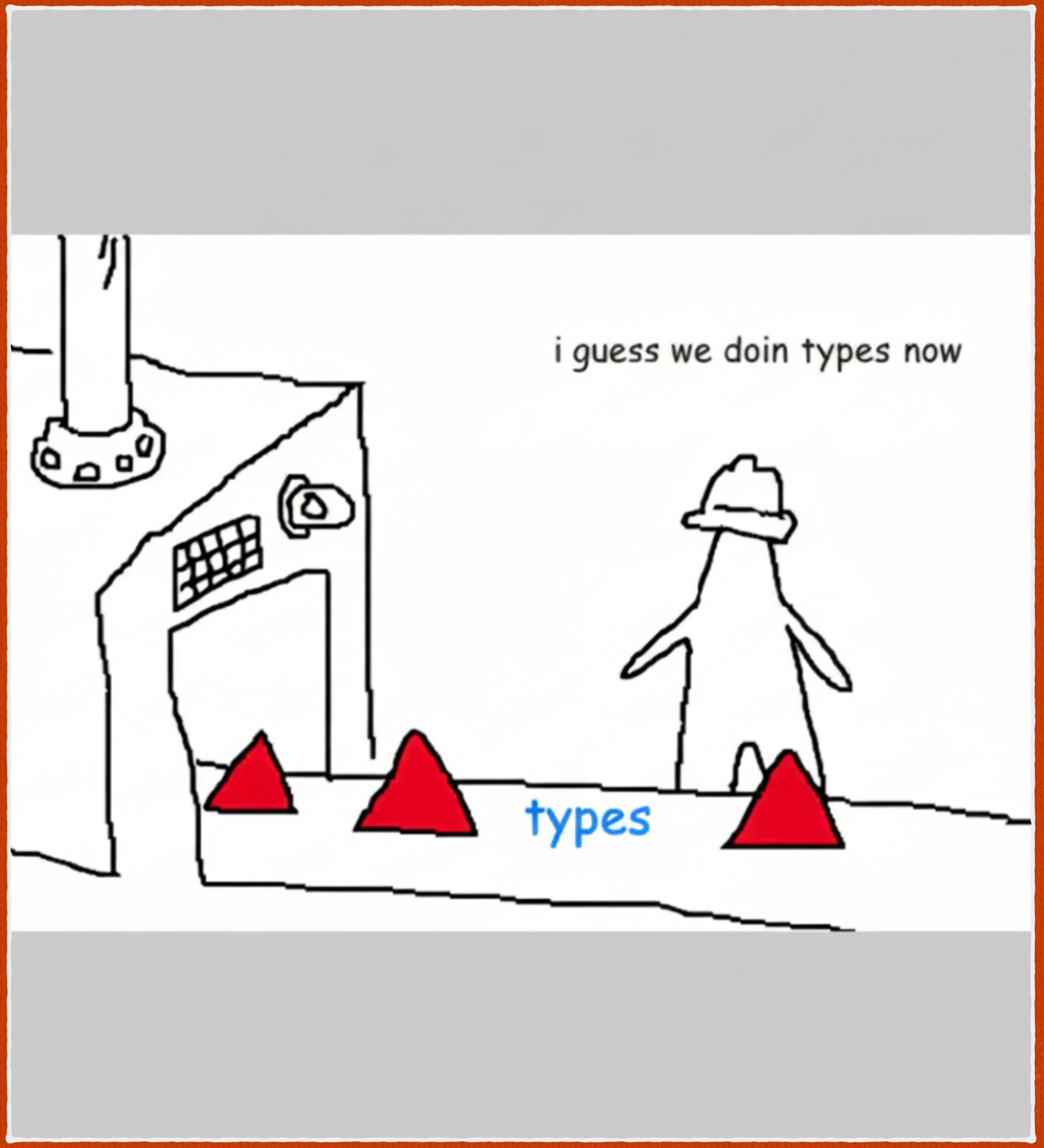
valitbot.ts

Avoid any at any cost!

- ☐ any can be assigned to every type!
- ☐ Use unknown instead!
- ☐ Unknown will force you to safely narrow the type

Teaching ESLint to understand types

Who uses ESLint?



What can type aware ESLint help with?

- [ ] Catch issues with promises
- [ ] Find areas where any is sneaking into code
- [ ] Prevent [object Object] and other string mishaps

unsafe-examples.ts

floating-promises.ts

How do you tighten standards in existing code?

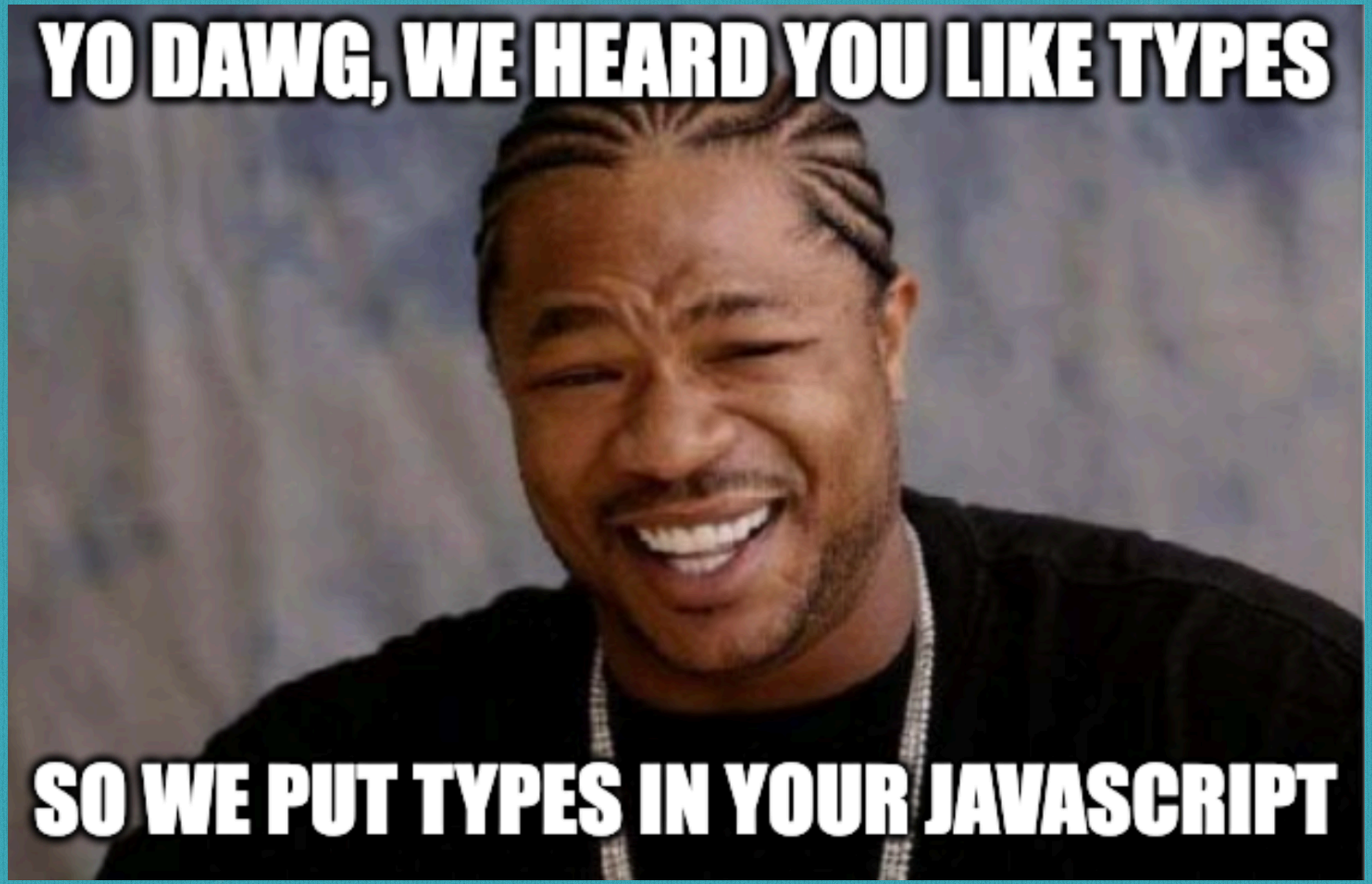
- [ ] Start with leaf modules. They tend to be more self contained.
- [ ] Converting older code can be the one time where casting with as and using the non null assertion is useful!

- [ ] Grant exceptions to compiler errors for old code with targeted disables

```
// eslint-disable-next-line @typescript-eslint/no-unsafe-call
```

- [ ] You can also use sparingly use ts-expect-error

```
// @ts-expect-error this code is unsound, but we're short on time so we're gonna let it slide for now
```


YO DAWG, WE HEARD YOU LIKE TYPES
SO WE PUT TYPES IN YOUR JAVASCRIPT

- [ ] Add "checkJs": true to your tsconfig to check all js files
- [ ] Or incrementally check files with "allowJs": true and // @ts-check comment at the top of the js file
- [ ] You can type almost anything in JS via comments now: https://www.typescriptlang.org/docs/handbook/jsdoc-supported-types.html

You can find the code on https://github.com/fleck/typescript-turned-up-to-11

Questions?