👻 HORROR STORIES FROM RUNNING 🅰️NGULAR SERVER SIDE RENDERING IN PRODUCTION 🚀 💥

HELLO 👋

- My name is Benjamin Legrand
- 👔 Tech Lead @ onepoint
- 🌍 Nantes, France
- @benjilegnard

INTRODUCTION

DISCLAIMER #1

`@nguniversal/*` === `@angular/ssr`

DISCLAIMER #2

- This is not a diss talk
- SSR with Angular is a lot better now

TABLE OF CONTENTS

WHAT IS SERVER SIDE RENDERING ?

"RENDERING" = CREATING HTML

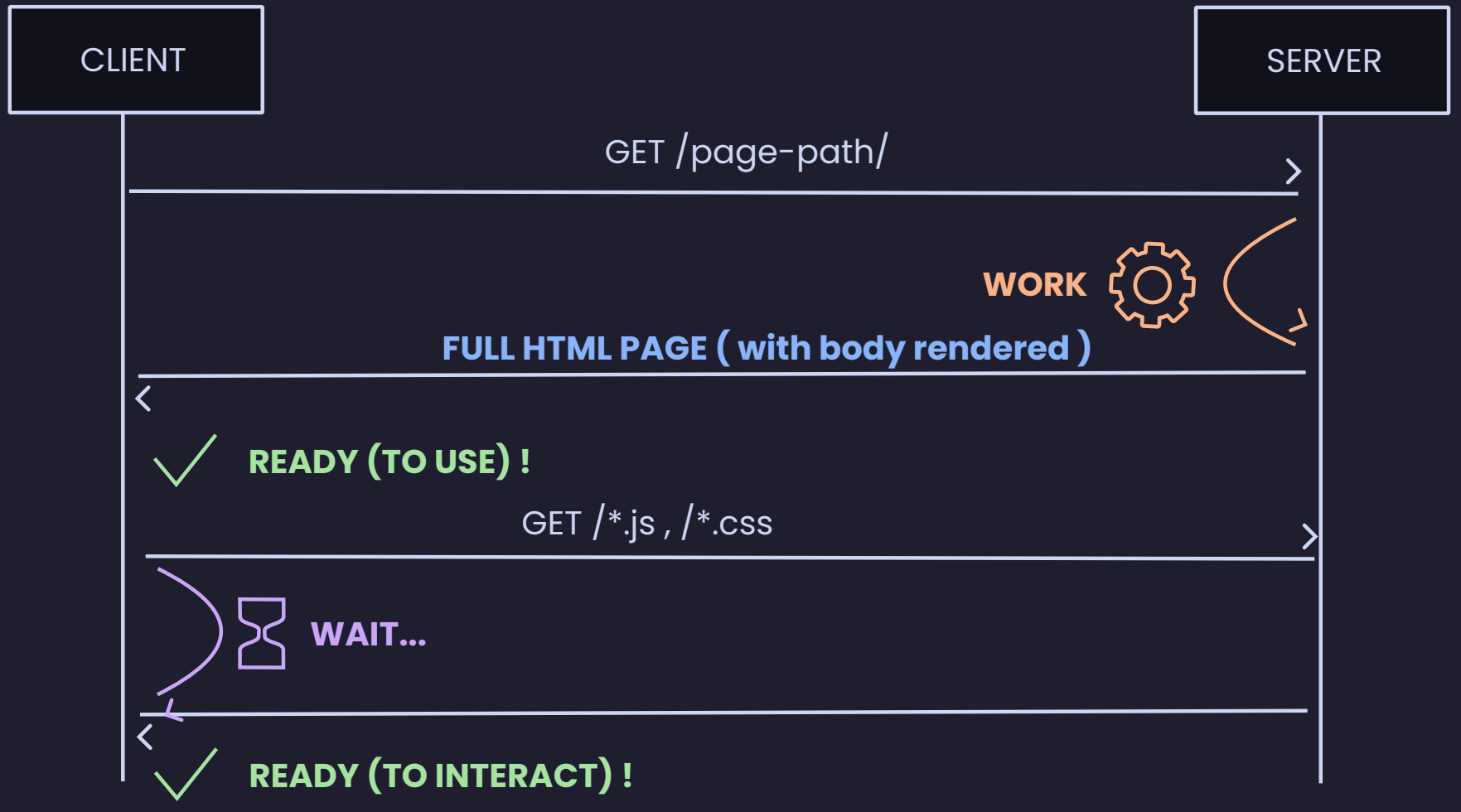
BY DEFAULT, ANGULAR IS A CLIENT SIDE FRAMEWORK

SPA : ASK THE SERVER FOR A PAGE

SPA : YOU GET NOTHING.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Single Page App</title>
</head>
<body>
  <app-root>
    <!-- NOTHING HERE, (or maybe, a loader?) -->
  </app-root>
  <script src="main.js"></script>
</body>
</html>
```

SSR : ASK THE SERVER FOR THE HTML



ADVANTAGES OF SSR

- 🔎 SEO
  (Search Engine Optimization)
- 🚀 Performance
  (faster loading times)
- ♿ Usability / accessibility
  (your site is usable before javascript is loaded)
- 👩‍💻Universal / Isomorphic code
  (same code on server and client)

DRAWBACKS OF SSR

- not for every app/site.
- can be tricky, has some footguns.
- paradigm change, not your typical SPA

TWO EXECUTION CONTEXTS

Browser !== Server

Browser

VS

Server

Browser	Server
window, navigator, geolocation, device, etc...	file system, network, OS APIs, etc...

HOW TO ADD SSR TO YOUR ANGULAR APP

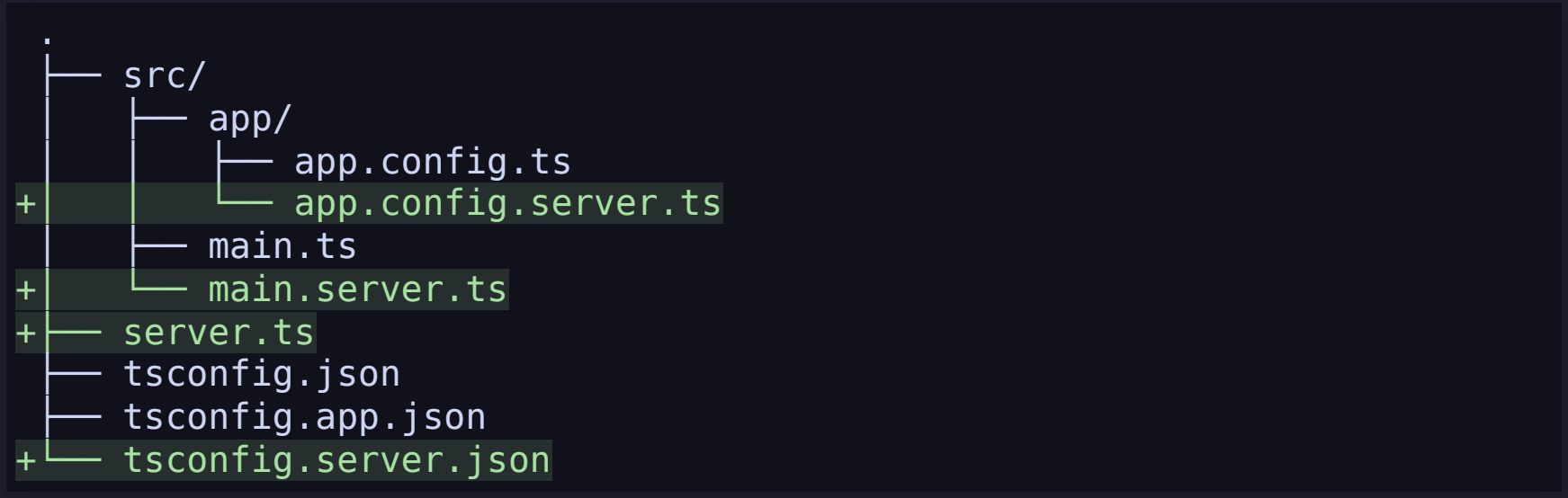
```
ng add @angular/ssr
```

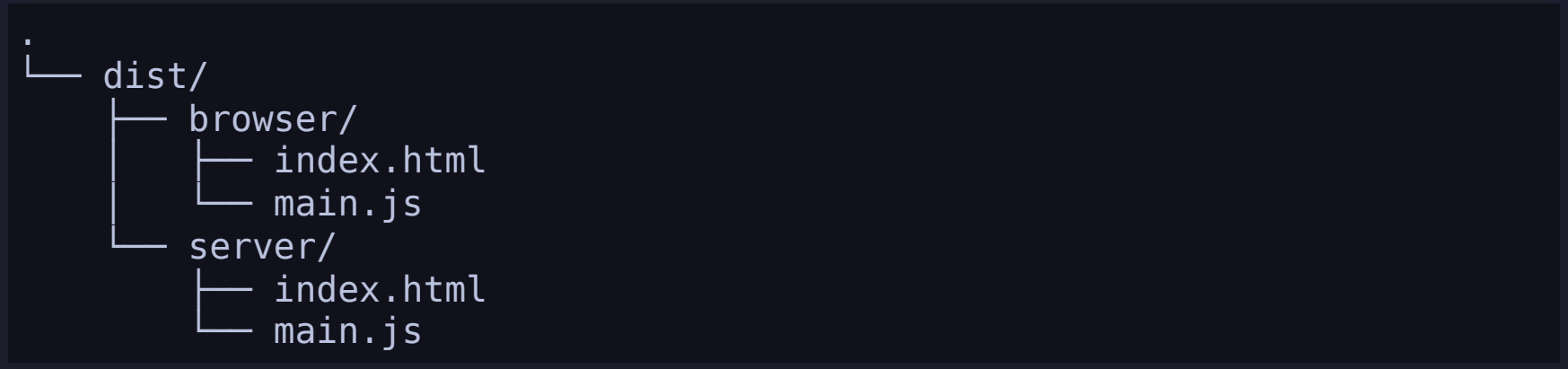
IN YOUR ANGULAR APP :

BEFORE

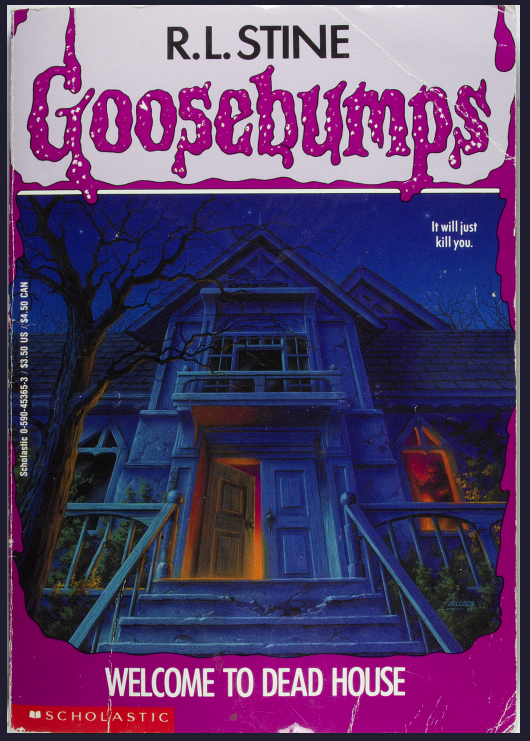
```
.
├── src/
│   ├── app/
│   │   └── app.config.ts
│   └── main.ts
├── tsconfig.json
└── tsconfig.app.json
```

AFTER


.
├── src/
│   ├── app/
│   │   ├── app.config.ts
+│   │   └── app.config.server.ts
│   ├── main.ts
+│   └── main.server.ts
+├── server.ts
├── tsconfig.json
├── tsconfig.app.json
+└── tsconfig.server.json


.
└── dist/
    ├── browser/
    │   ├── index.html
    │   └── main.js
    └── server/
        ├── index.html
        └── main.js

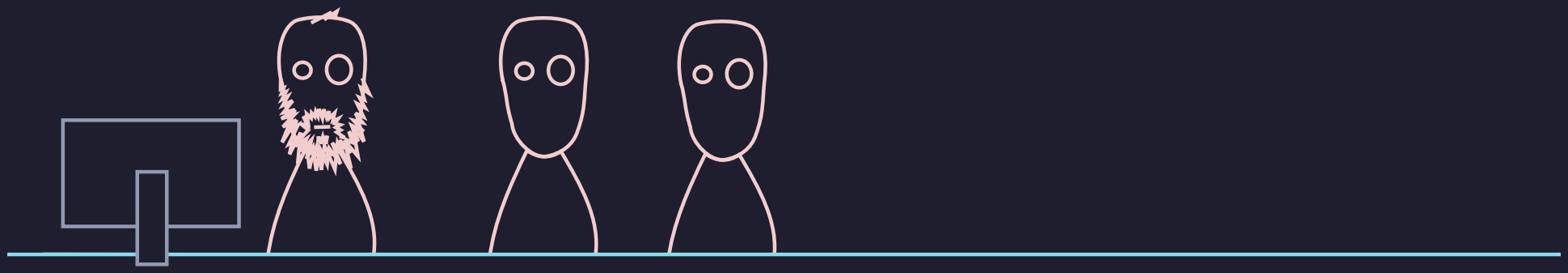
WINDOW IS UNDEFINED

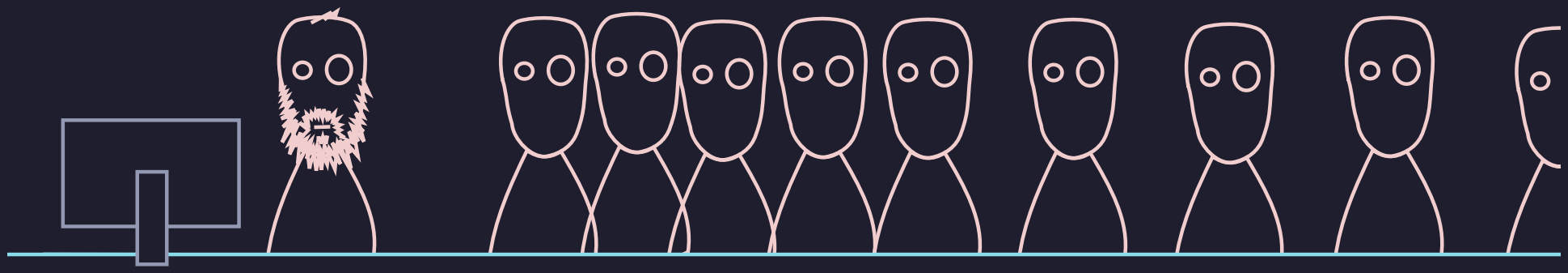


CONTEXT ?

- 👥 Big project, 30+ Developers
- 📚 Lots of daily pull-requests
- ⚡ Very fast code reviews... maybe too fast

WHAT HAPPENED ?





WHAT HAPPENED ?

- 👮 We had safe-guards about not using browser globals. but...
- 📙 "Someone" added a third-party library
- 🪟 that used the window object

WHAT (REALLY) HAPPENED.

- ♻️ Developers were not using the "SSR" devmode, only SPA
- 🐌 "because it is slower"

SOLUTIONS

RECOMMENDED SOLUTION

```
@Injectable({
  providedIn: 'root'
})
export class WindowService {
  private document: Document = inject(DOCUMENT);

  get window(): Window {
    return this.document.defaultView;
  }
}
```

QUICK AND DIRTY FIX

```
globalThis['window'] = {
  // properties you need implemented here...
};
```

A BETTER SOLUTION

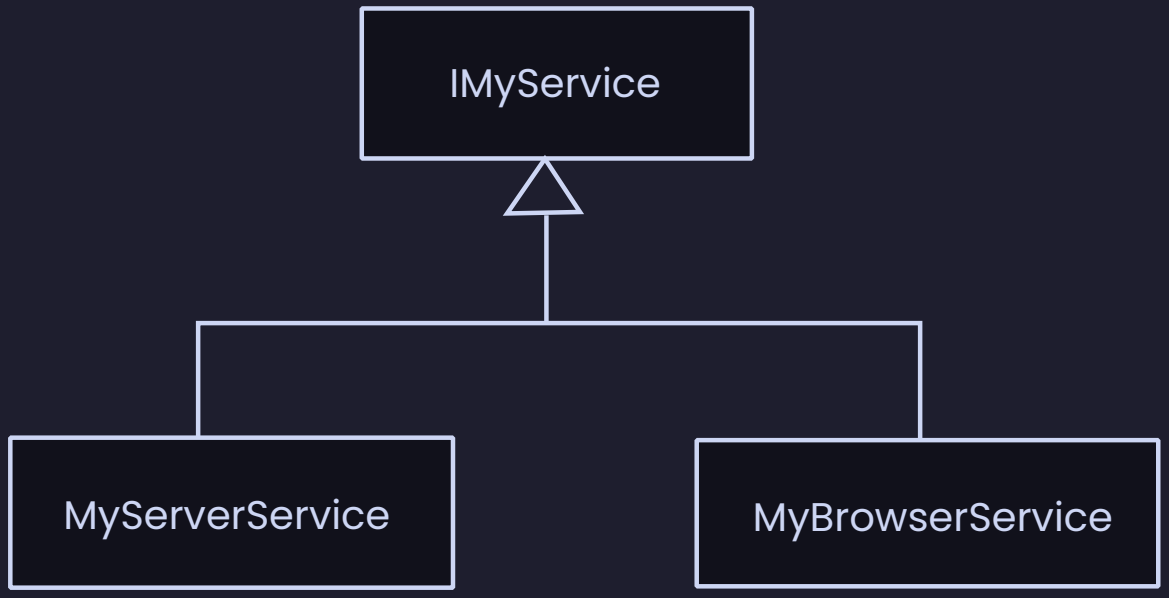
```
1  import { isPlatformBrowser } from '@angular/common';
2  import { PLATFORM_ID } from '@angular/core';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class MyService {
8    private platformId = inject(PLATFORM_ID);
9
10   async loadLibrary() {
11     if (isPlatformBrowser(this.platformId)) {
12       // do something on browser only
13       return await import('external-ib');
14     }
15     // server only case
16     return undefined;
17   }
```

A VERY BAD SOLUTION

Please do not use `fileReplacements` for this

```
{
  "configurations": {
    "server": {
      "fileReplacements": [
        {
          "replace": "src/app/assets.service.ts",
          "with": "src/app/assets.service.server.ts"
        }
      ]
    }
  }
}
```

ANOTHER (GOOD) SOLUTION



ANOTHER (GOOD) SOLUTION

use injection tokens:

```
interface AssetsFetcher {
  fetchAssets<T>(): Promise<T>;
}

export const ASSETS_FETCHER = new InjectionToken<
    AssetsFetcher>('ASSETS_FETCHER');
```

app.config.ts

```ts
import { ASSETS_FETCHER } from './assets-fetcher';
import { AssetsFetcherBrowserService } from './assets-fetcher-browser

export const appConfig: ApplicationConfig = {
    providers: [
        {
            provide: ASSETS_FETCHER,
            useClass: AssetsFetcherBrowserService
        }
    ]
};
```

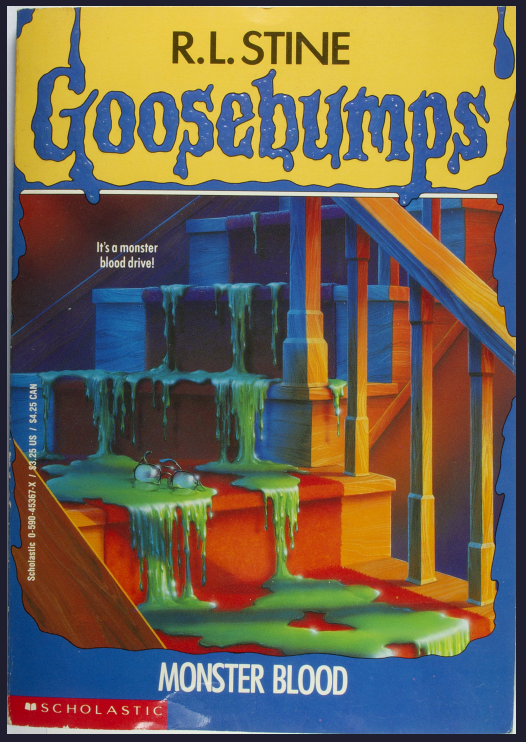
app.server.config.ts

```ts
import {
        ApplicationConfig,
        mergeApplicationConfig,
} from "@angular/core";

import { appConfig } from "./app.config";
import { AssetsFetcherServerService } from "./services/assets/assets-
import { ASSETS } from "./services/assets/assets-fetcher.token";

const serverConfig: ApplicationConfig = {
  providers: [
    provideServerRendering(),
    {
      provide: ASSETS_FETCHER,
      useClass: AssetsFetcherServerService
    },
  ],
```

TAKEAWAYS

- ♻️ Always use the SSR dev mode
- 🧠 Think about the two execution contexts
- ✂️ Clearly separate code paths

OUPS... A MEMORY LEAK



CONTEXT ?

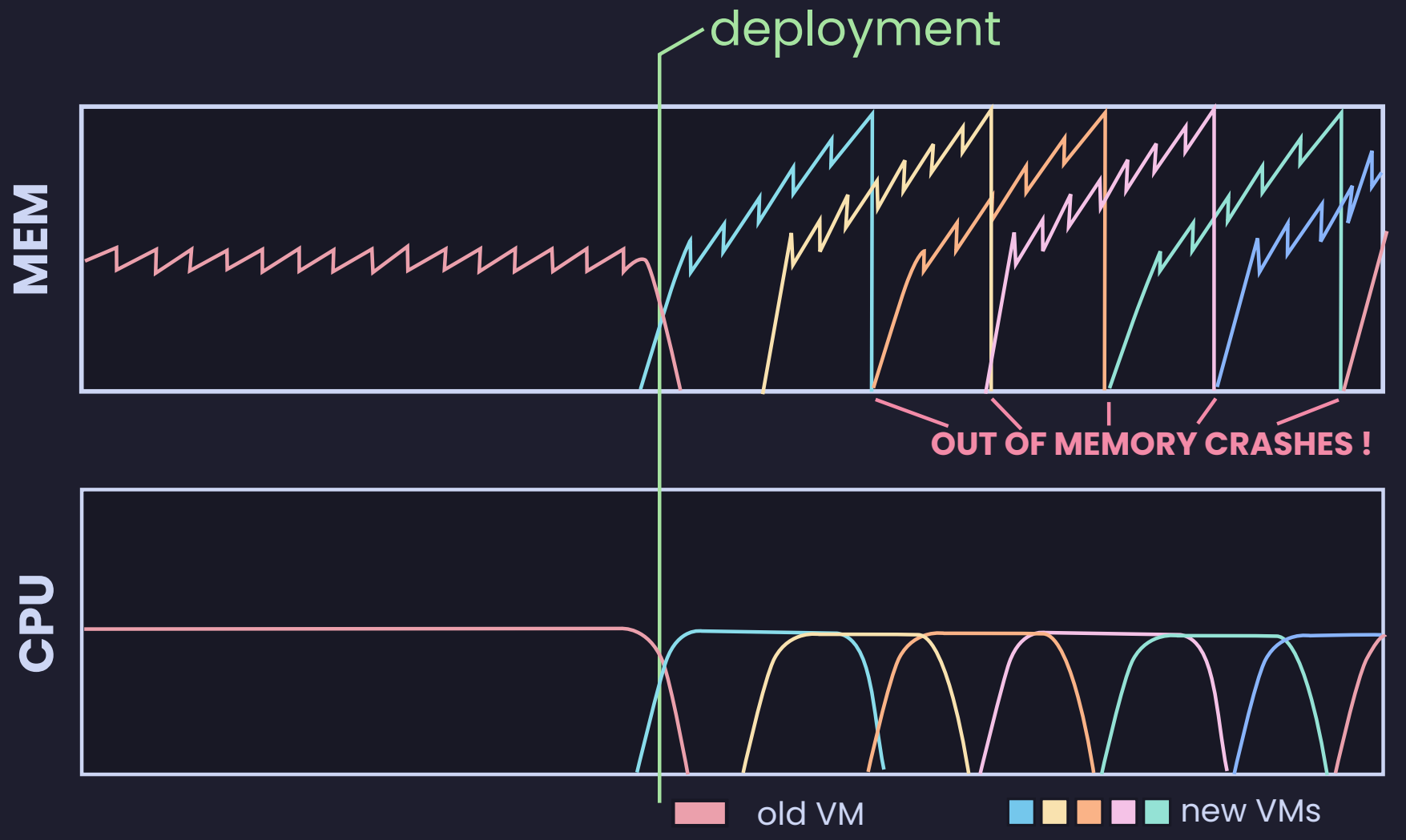
- 👥 Huge project with 50+ developers
- 🚀 One production release per sprint
- 📈 ~= 1 million daily page views

DEPLOYMENT METRICS


deployment
MEM
CPU
old VM
new VM


deployment
MEM
CPU
old VM 1
old VM2
new VM 1
new VM 2

WHAT HAPPENED ?



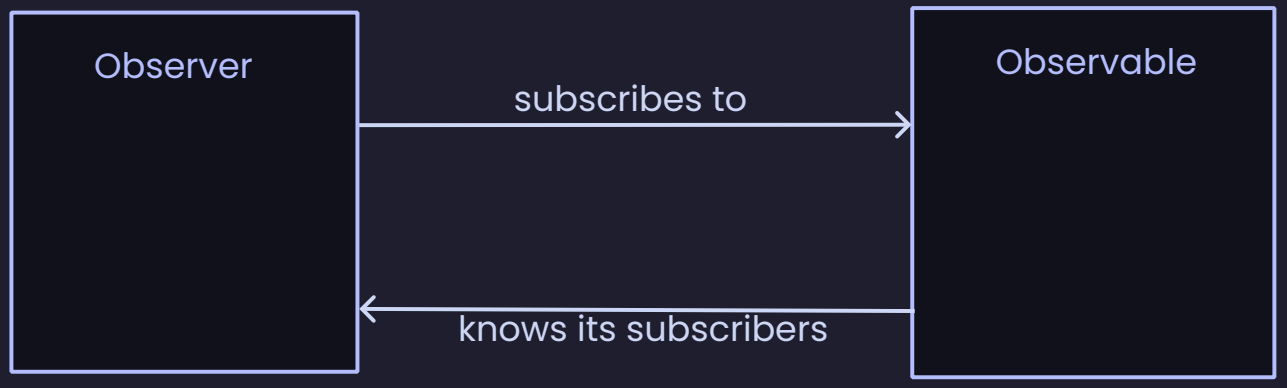
WHAT WAS THE ROOT CAUSE ?

Someone pushed this kind of code:

```
@Injectable({
  providedIn: 'root'
})
export class MyService {
  constructor(otherService: OtherService) {
    this.otherService.observable$.subscribe(() => {
      // do something
    });
  }
}
```

WHAT IS THE ISSUE ?

- 🔭 the Observer pattern leaks



OTHER SOURCE OF MEMORY LEAKS

- addEventListener()
- removeEventListener()

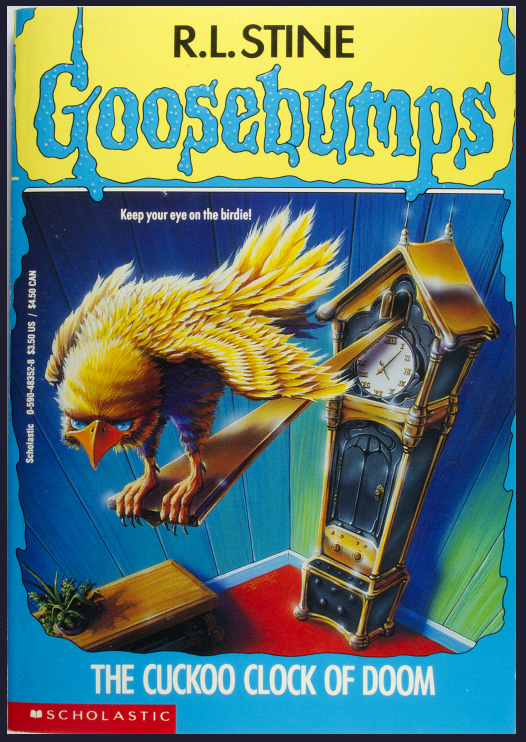
SECOND ISSUE:

- {providedIn:'root'}.
- means the service was instanciated for EVERY page.

SOLUTIONS ?

- 👮 always unsubscribe / 🧹 cleanup behind you
- 🚫 do not use the constructor to initialize observables
- 👍 use "init" methods instead
- ⛔ avoid providedIn:root when you can

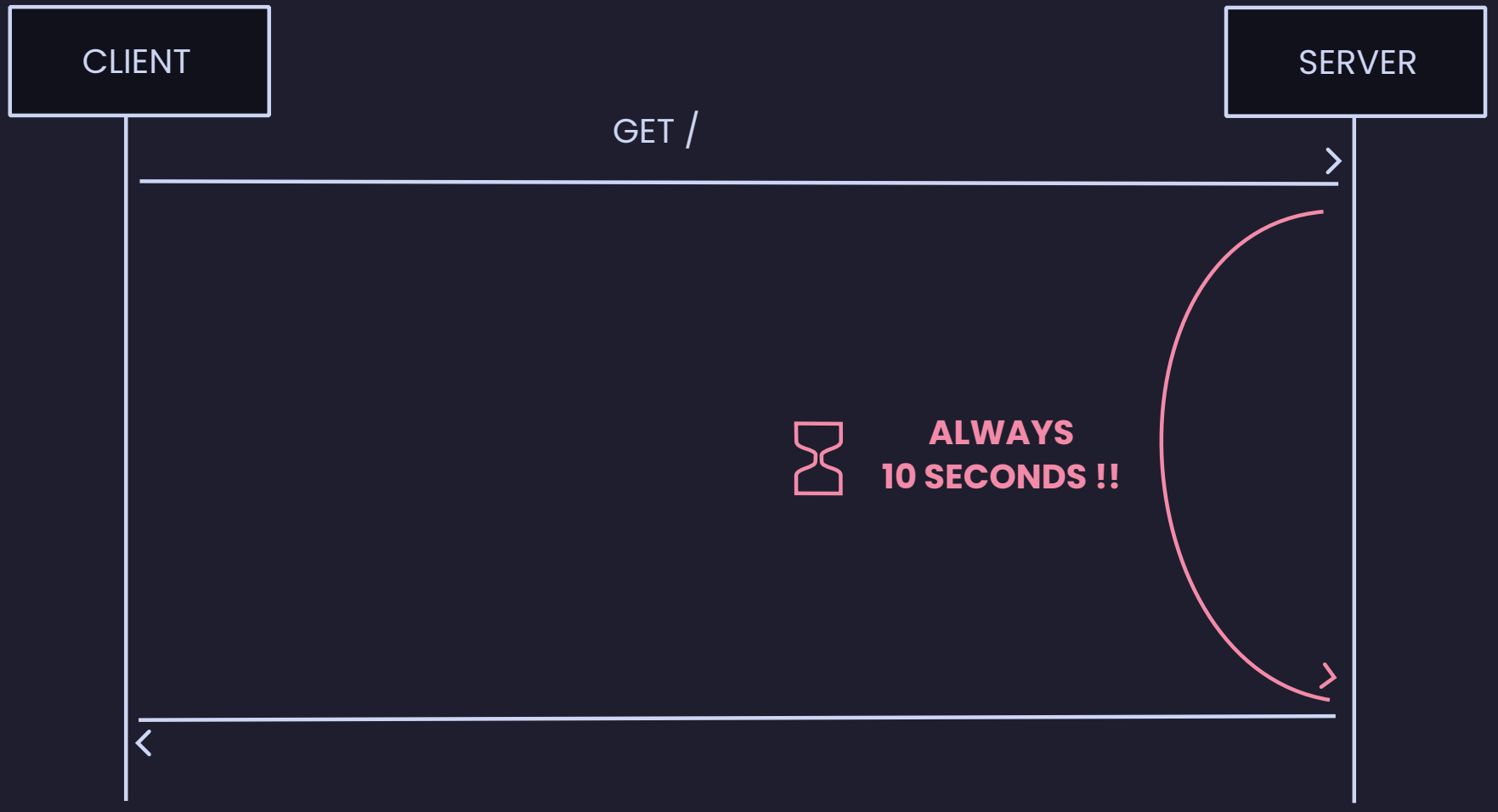
THE SETTIMEOUT TRAP



CONTEXT ?

- 🏗️ Consulting on a "recent" project
- 👶 First task: "please fix the SSR performance"

WHAT WAS HAPPENING ?



WHAT HAPPENED ?

- 🕵️ investigating... 🕵️
- found this code

```
class MyService {

  private maxRetries: number = 4;
  private sleep: (ms: number | undefined) => Promise<unknown> = (
    ms: number | undefined,
  ) => new Promise((r) => setTimeout(r, ms));

  public async sendMessageEvent(message: string): Promise<Status> {

        let sendEventStatus = this.postMessageToIframe(message);

        if (sendEventStatus !== "OK") {
            let retry = 0;
            while (
                sendEventStatus !== Status.OK &&
                retry < this.maxRetries
            ) {
```

WHAT WAS THE ISSUE ?

- zone.js 🫠
- server render will wait for ApplicationRef.isStable()
- will wait for any callback / promise / microtask to finish
- artificial delays

ALSO:

- There is no `window` on the server.
- There is no `iframe` either.

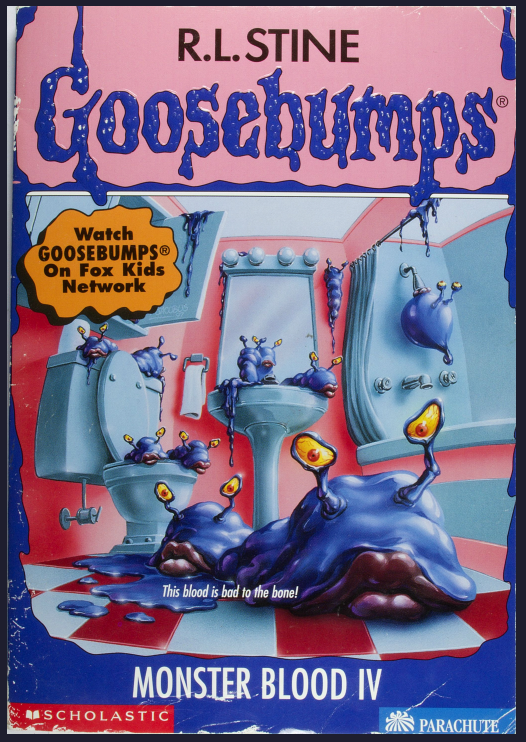
SOLUTIONS

- avoid using `setTimeout` / `setInterval` in SSR
- (same thing for rxjs `delay` / `interval` / timing operators)
- condition them to run only in browser mode

TAKEAWAYS

- ⚠️ be careful with timeouts and intervals.
- 🤖 avoid artificially delaying the server response times.

HELP, MY API IS REQUESTED TWICE.

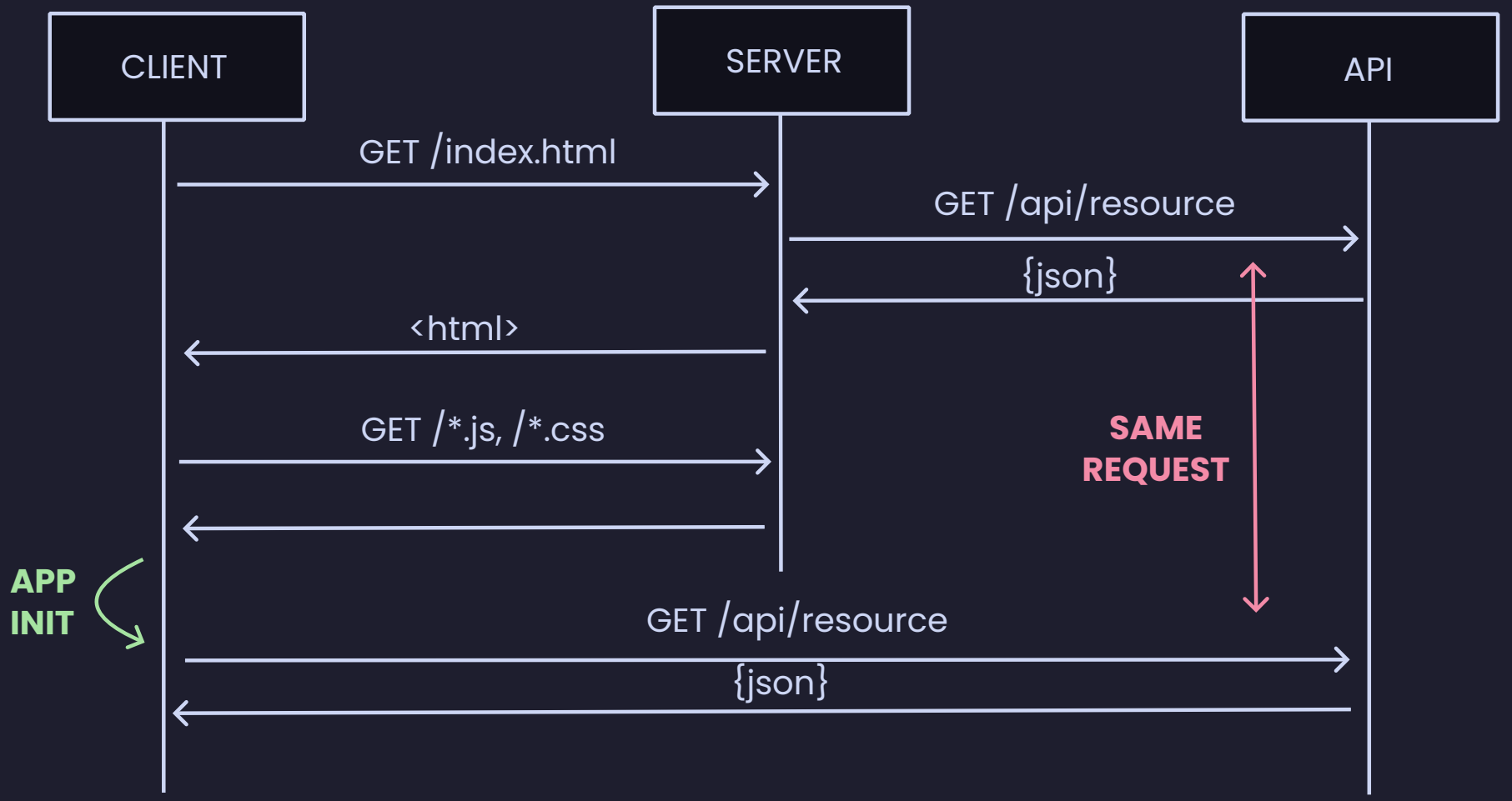


CONTEXT ?

- an app that fetches json data from an api to render.
- pretty standard stuff
- backend service already existed

WHAT IS THE ISSUE ?

- 🚀 new deployment.
- 🚚 put twice the load on the backend.

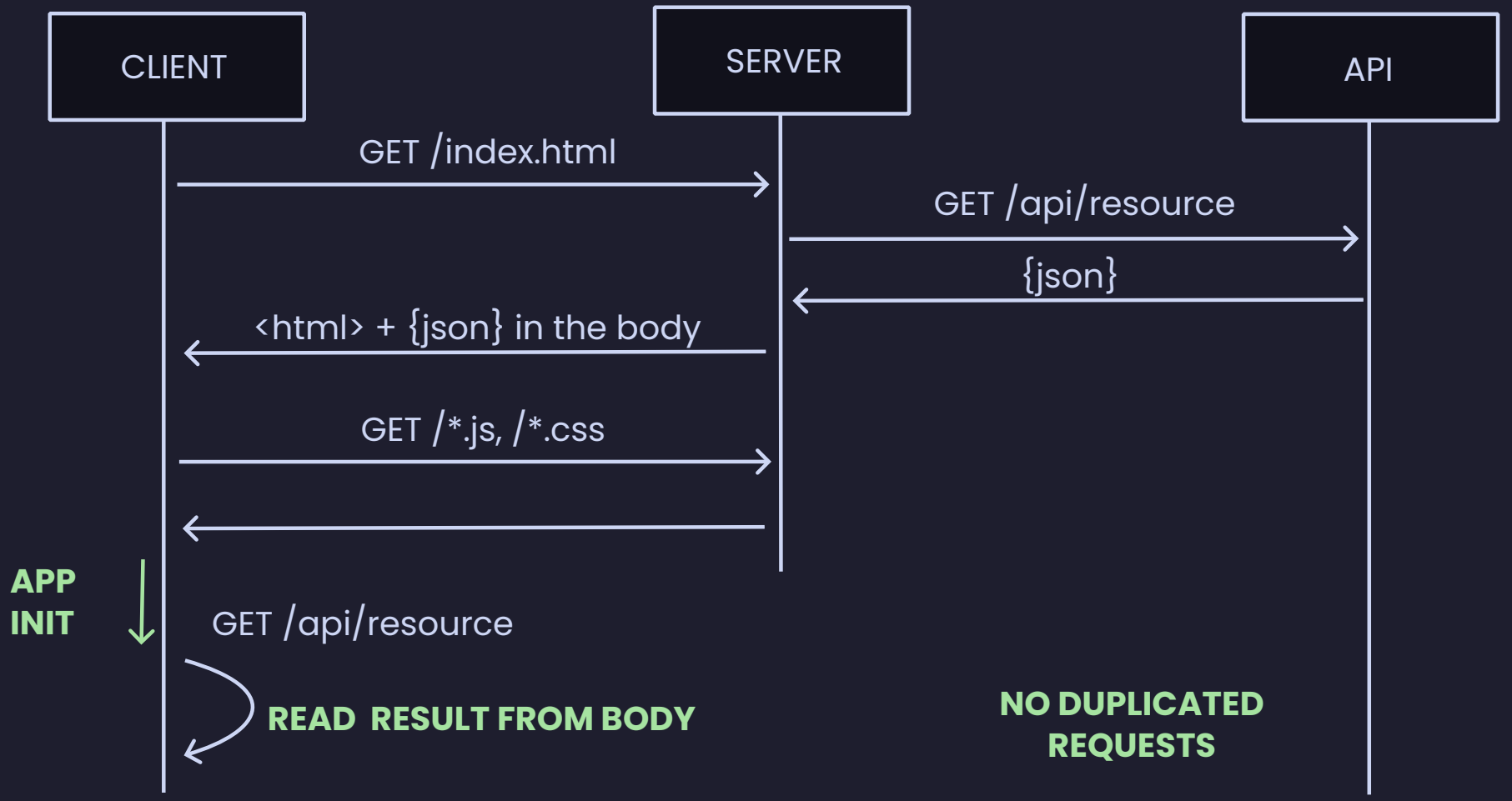

CLIENT
SERVER
API
GET /index.html
GET /api/resource
{json}
<html>
GET /*.js, /*.css
SAME REQUEST
APP INIT
GET /api/resource
{json}

SOLUTION

- TransferState
- HttpInterceptor using it

TRANSFER WHAT ?

```
<!DOCTYPE html>
<html lang="en">
  <body>
    <!-- ... -->
    <script id="ng-state">
        {
                "http://api.domain.com/resource": {
                    "data": {/** ... */}
                }
        }
    </script>
  </body>
</html>
```


CLIENT
SERVER
API
GET /index.html
GET /api/resource
{json}
<html> + {json} in the body
GET /*.js, /*.css
APP INIT
GET /api/resource
READ RESULT FROM BODY
NO DUPLICATED REQUESTS

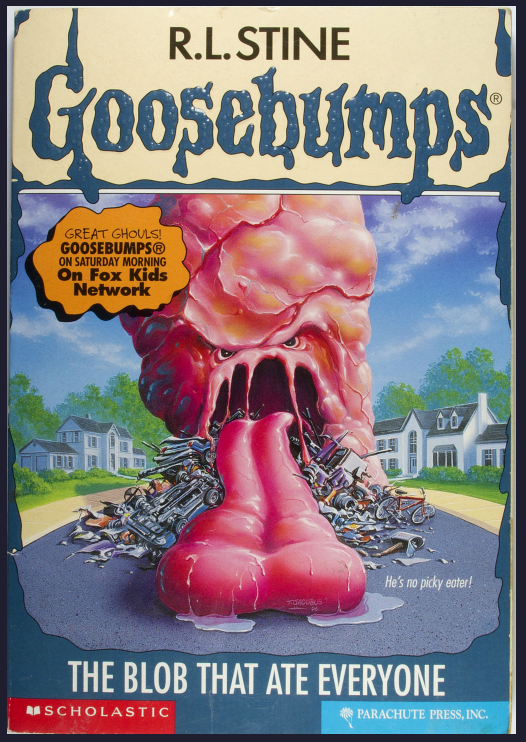
ACTUALLY:

- transfer state is now included by default with `provideServerRendering()`
- you had to explicitly enable it before

TAKEAWAYS

- 👫two execution contexts, two requests
- 🧠think about the cacheability of your data

THE "SCANDAL" OF INLINE CRITICAL CSS

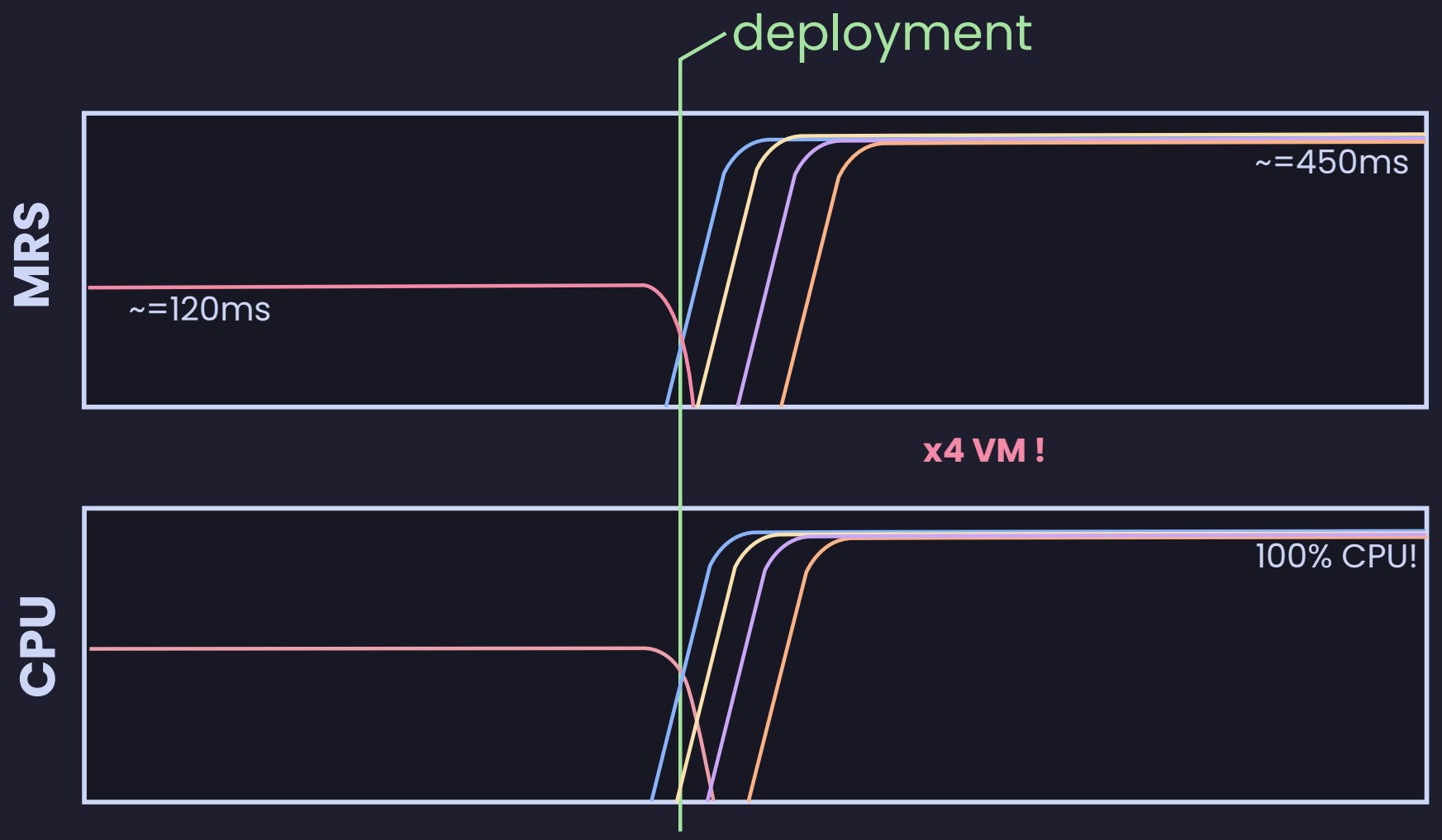


CONTEXT ?

- 👥 huge project, 50+ developers
- 🚀 one production release per sprint
- 🥶 "code freeze" for 3 sprints during christmas 🎅
- = one HUGE release

WHAT HAPPENED ?


deployment
MEM
CPU
old VM
new VM


deployment
MRS
~=120ms
~=450ms
x4 VM !
CPU
100% CPU!

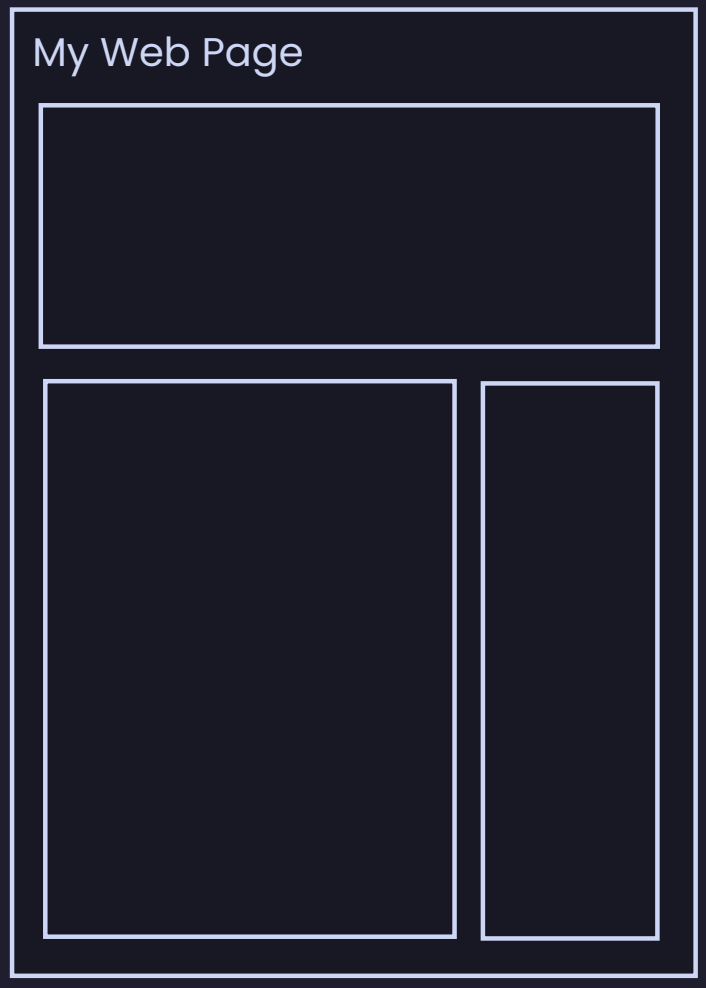
INVESTIGATION

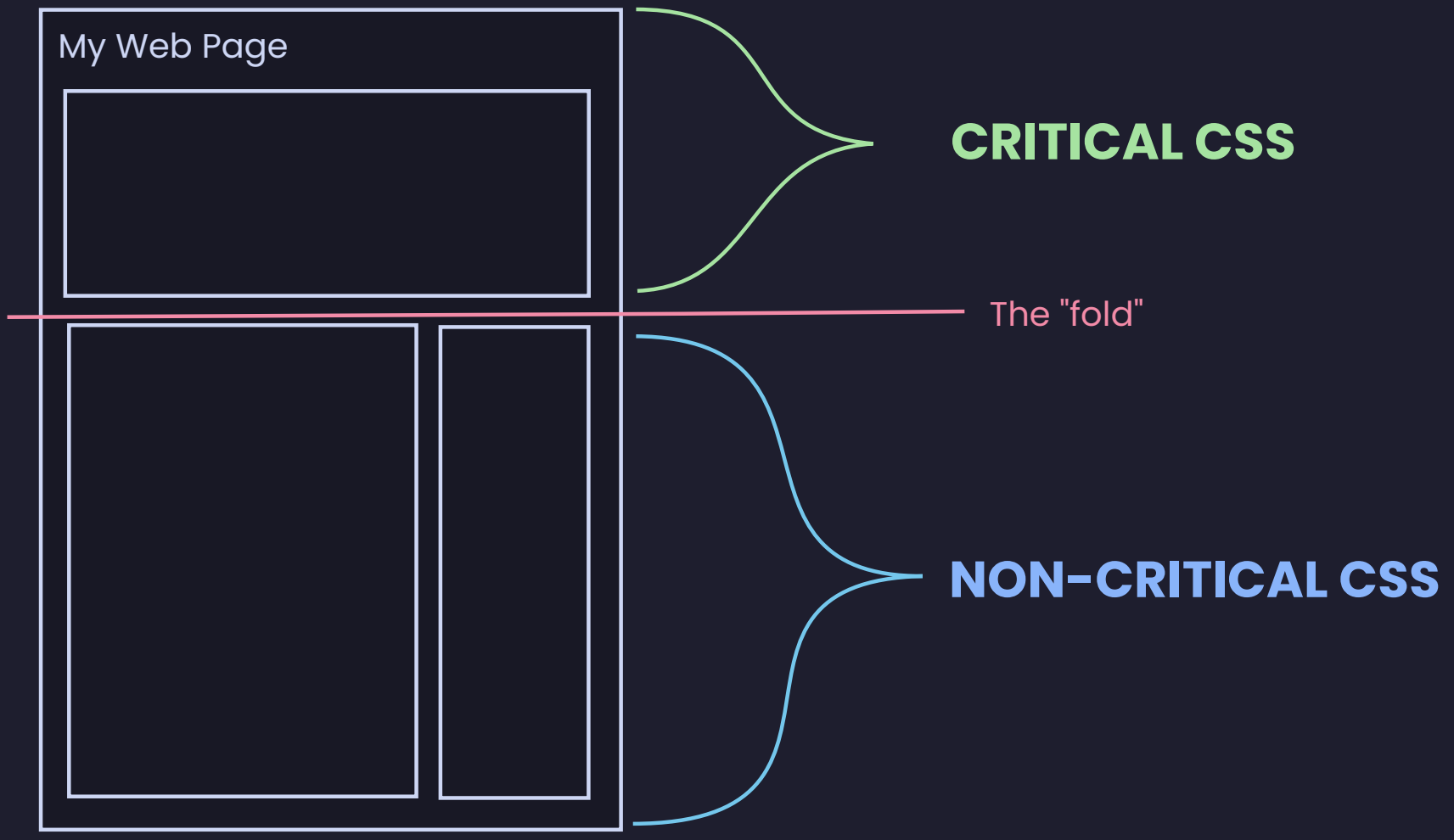
- 🚑 git bisect (~= 700 commits)
- 🤔 what changed ? what are we looking for ?
- ⏱️ While we're looking, hosting the app costs... 💸
- 🎉 thanks to 0x, and the flamegraph

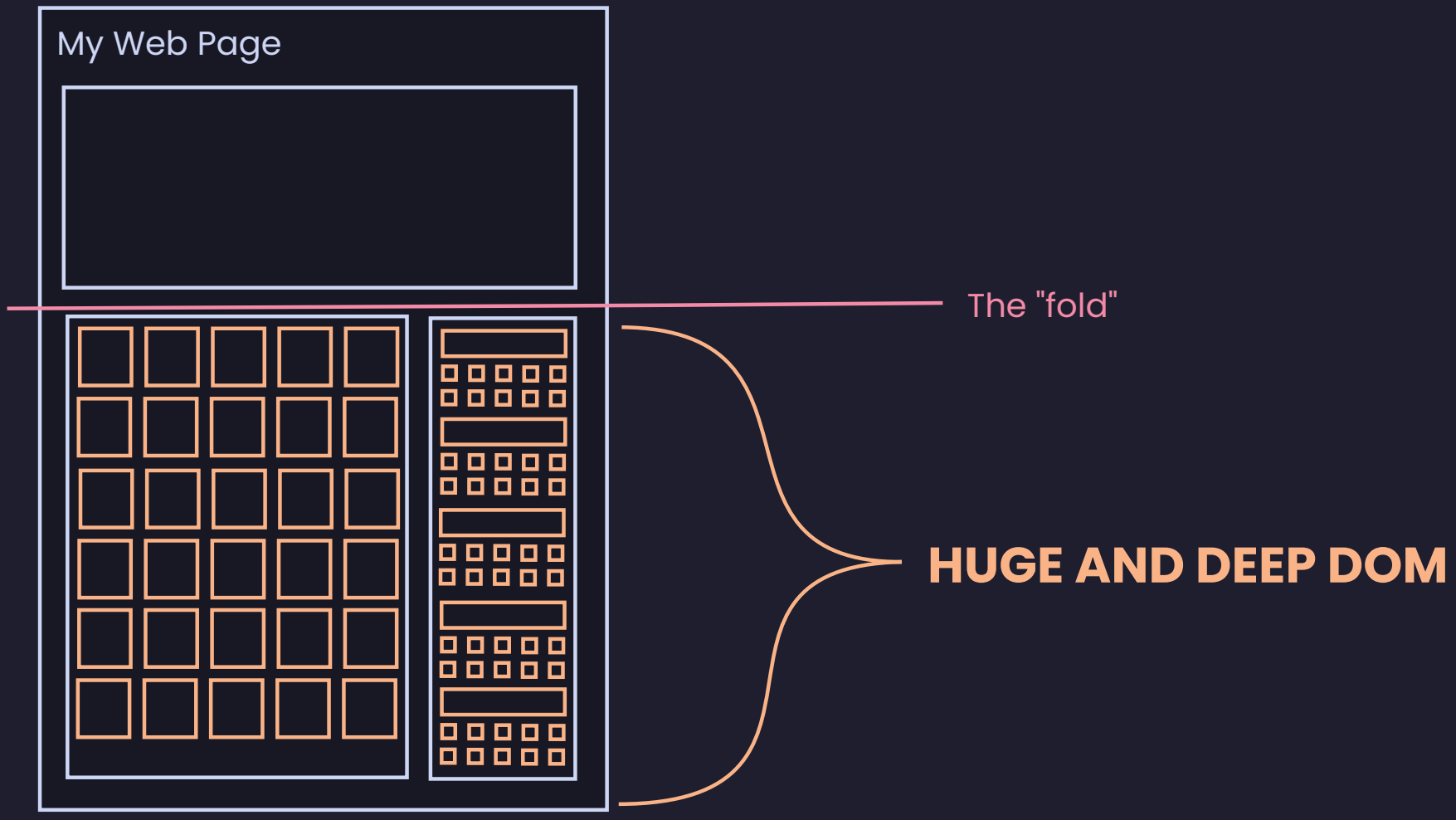
WHAT (REALLY) HAPPENED ?

- Angular 12 upgrade was in the release
- `inlineCriticalCss` feature was enabled by default

WHAT IS CRITTERS ?


My Web Page


My Web Page
CRITICAL CSS
The "fold"
NON-CRITICAL CSS


My Web Page
The "fold"
HUGE AND DEEP DOM

SOLUTIONS

QUICK AND DIRTY FIX

- do not inline critical css

server.ts

```
    commonEngine
      .render({
        bootstrap,
        documentFilePath: indexHtml,
        url: `${protocol}://${headers.host}${originalUrl}`,
        publicPath: distFolder,
+        inlineCriticalCss: false,
        providers: [
          { provide: APP_BASE_HREF, useValue: baseUrl },],
      })
      .then((html) => res.send(html))
      .catch((err) => next(err));
```

angular.json

```
{
    // ...
    "configurations": {
        "production": {
            "optimization": {
                "fonts": true,
                "scripts": true,
                "styles": {
+                   "inlineCritical": false,
                    "minify": true
                }
            }
        },
    },
    // ...
}
```

A BETTER SOLUTION

- use critter comments in HTML

```
<html>
  <body>
    <div class="container">
      <div data-critters-container>
        /* HTML inside this container are used to evaluate critical CSS
      </div>
      /* HTML is ignored when evaluating critical CSS */
    </div>
    <footer></footer>
  </body>
</html>
```

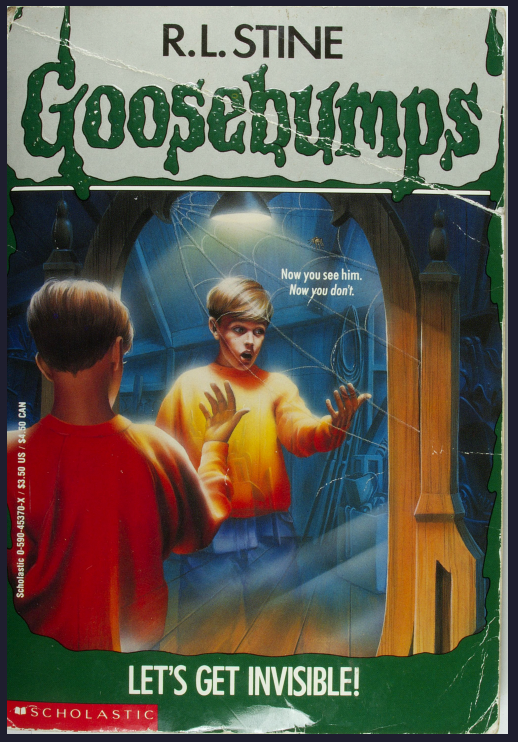
- or in CSS

```
/* critters:exclude start */

.selector1 {
  /* this rule will be excluded from critical CSS */
}

.selector2 {
  /* this rule will be excluded from critical CSS */
}

/* critters:exclude end */
```

TAKEAWAYS

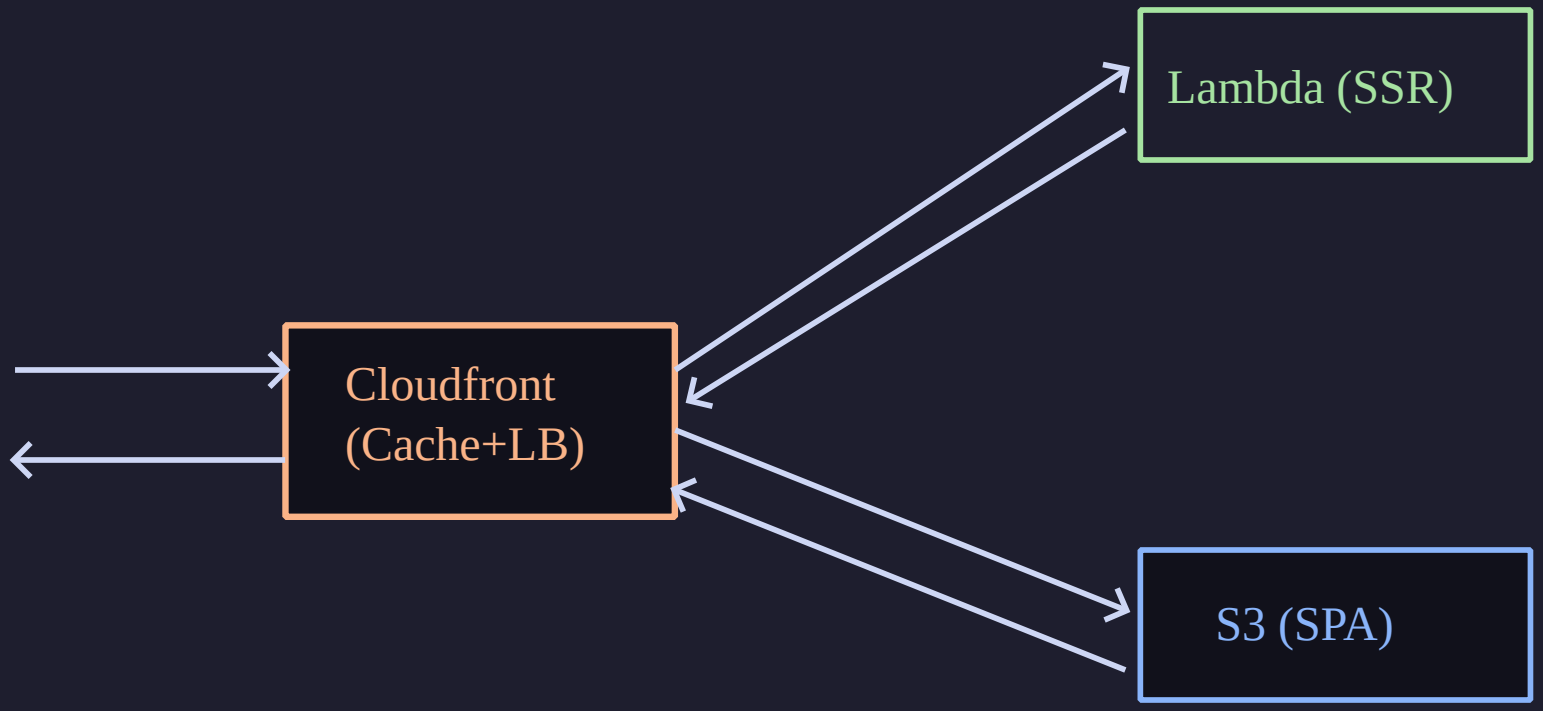
- 🚚 Avoid big releases
-  Release early, release often
- 📦 Load testing is important
- 🤓 Read the Angular changelogs.

THE INVISIBLE INFRASTRUCTURE



CONTEXT ?

- Again, I was asked to "fix the SSR" on a project
- This was their architecture...


Lambda (SSR)
Cloudfront
(Cache+LB)
S3 (SPA)

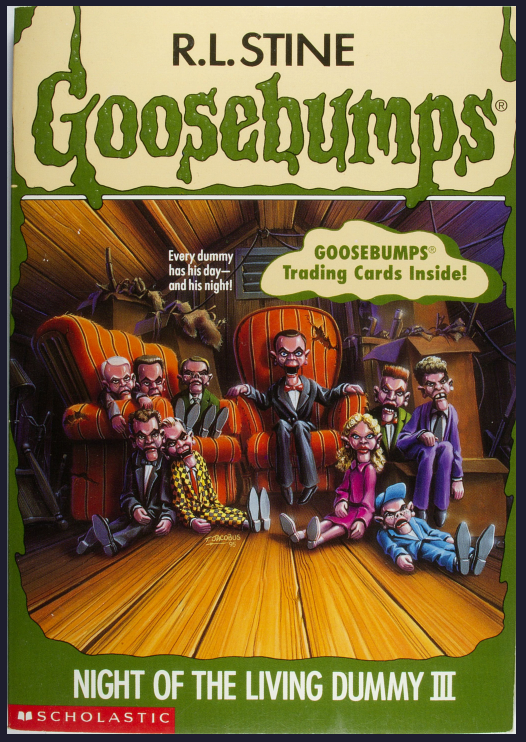
WHAT IS THE ISSUE ?

- server side renderer ALWAYS failed

SOLUTIONS:

- 🕵️ Alerts and observability
- Do not run your code without logs
- SPA fallback is a good idea but :
- DO NOT hide the errors
- A 500 error should NEVER happen

THE LOCALIZATION HELL



CONTEXT ?

- 🗣️ Multi-language app
- 🌎 Multi-country app
- O(n) problem

WHAT WAS THE ISSUE ?

- 🅰️ngular native i18n builds an app per locale
- 🐢 build times (country x lang)
- 🐌 deployment times

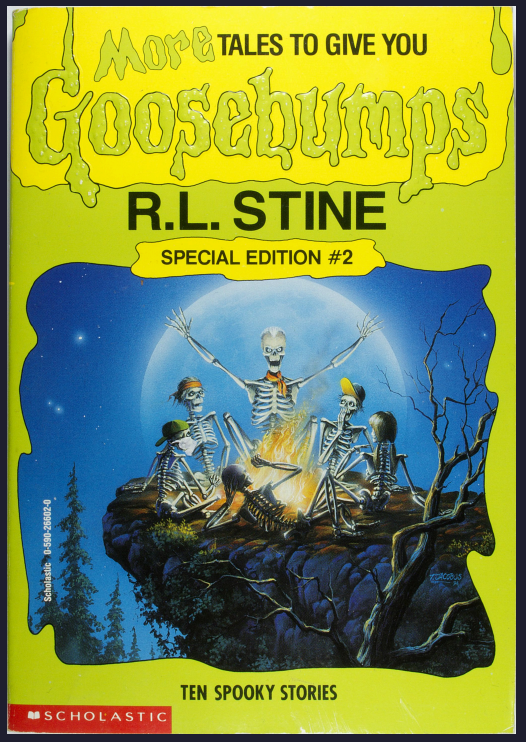
SOLUTIONS

- Build per locale, not per country + locale
- Dynamic lang, do not build an app per multi-language
- Use third party libs
    - @jsverse/transloco
    - ngx-translate

DON'T REINVENT THE WHEEL

- Accept-Language header

OUR CORE WEB VITALS ARE BAD.



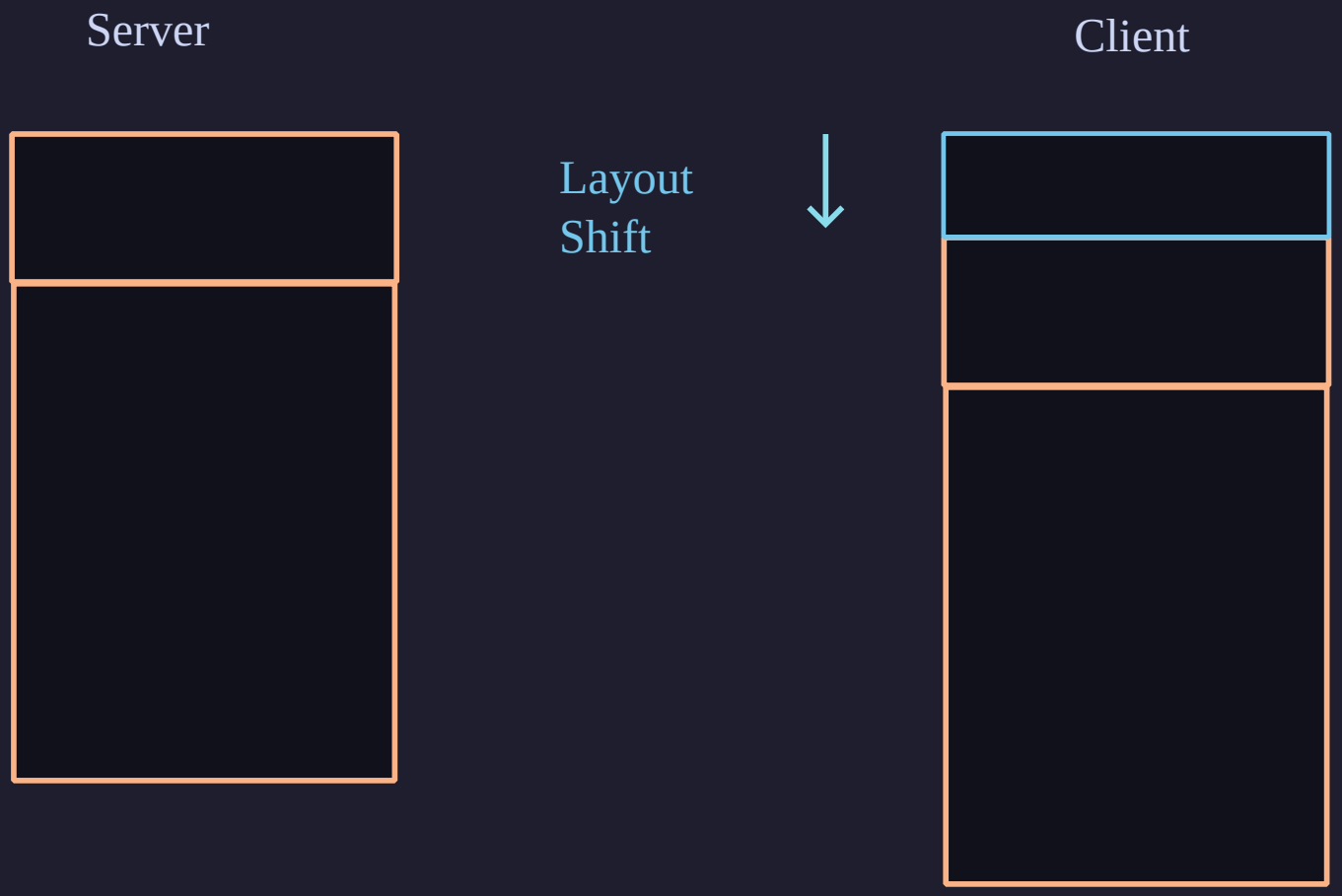
CORE WEB VITALS ?

- LCP (Largest Content Paint)
- INP (Interaction to Next Paint)
- CLS (Cumulative Layout Shift)

CONTEXT

- Let's say we have a "dismissable" banner
- Based on user cookies/localStorage
- But... Your server pages are cached

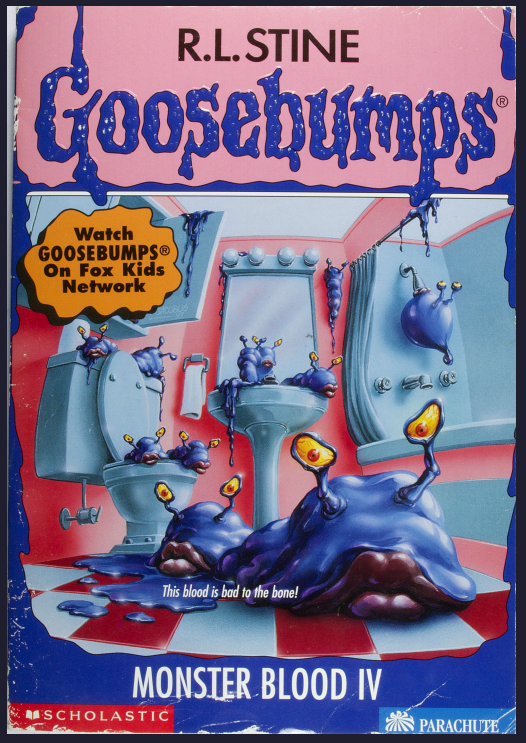
WHAT HAPPENED ? THE PROBLEM :



SOLUTIONS

- ⚠️ avoid using local storage (only) to store persistent page state.
- 🍪 use query params or cookies
- ❗ Take the cookies into consideration in the cache key

HYDRATION ? NOT YET.



CONTEXT

- hydratation ?

THE PROBLEM:

- big loading time
- page was emptied / re-rendered
- huge cpu spike
- INP metric : BAD

SOLUTIONS

- preboot library
- angular 16 hydration to the rescue
- provideClientHydration()
- withEventReplay()

CONCLUSION

- 🤔 Should you still do server-side-rendering?
- Yes

- 🔍 observability is key 🔑
- 🚨 logs, metrics, alerts
- ⚡ performance matters
- ⚠️ do not hide errors
- 🤓 share knowledge

THANK YOU

@benjilegnard