

FRONTEND DEVELOPMENT

WINTERSEMESTER 2020

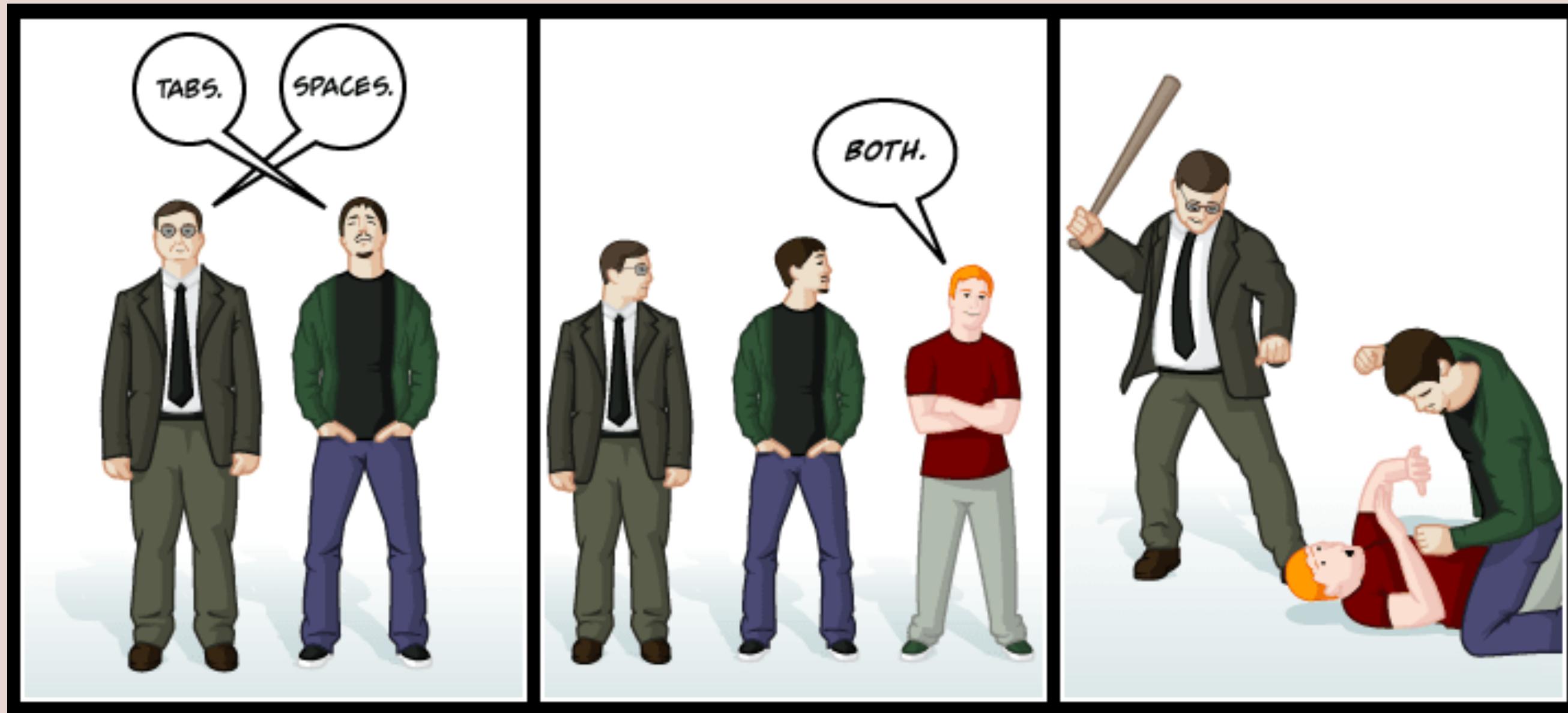
TOOLING

LINTING¹

“Linting is the automated checking of your source code for programmatic and stylistic errors”

¹ <https://www.perforce.com/blog/qac/what-lint-code-and-why-linting-important>

LINTING⁵



⁵ source https://dev.to/_shadz/tabs-vs-space-4915

LINTING

- » Prevent programming errors earlier
- » tells problematic sections of the code earlier
- » Enforce code consistency in projects

LINTING IN JS

- » spaces vs. tabs (see:
<https://www.youtube.com/watch?v=Sso0G6ZeyUI>)
- » naming (camelCase vs.
snake_case)
- » prevent == vs === errors
- » using single vs double
quotes
- » code formatting
- » ...

LINTING TOOLS

- » various tools exist to validate code consistency
- » EditorConfig
- » eslint (defacto standard)
- » prettier (more code formatter)

EDITORCONFIG



EDITORCONFIG

“EditorConfig helps maintain consistent coding styles for multiple developers working on the same project across various editors and IDEs.”

```
# Unix-style newlines with a newline ending every file
[*.{js,py}]
indent_style = space
indent_size = 4
end_of_line = lf
insert_final_newline = true
```

ESLINT



ESLINT

- » standard tool for linting
- » easy to extend
- » autofix option
- » can be configured ²

² and is configured in every project differently

ESLINT PRESETS

- » presets bundle rule-sets together
- » well known presets are:
 - » eslint-config-airbnb
 - » eslint-config-standard
 - » eslint-config-google
 - » ...

ESLINT

INSTALL ESLINT³

```
npx eslint --init # wizard opens (answer questions)  
npx eslint : # lints all the files  
npx eslint : --fix # lints all the files and fixes most errors
```

³ see <https://eslint.org/docs/user-guide/getting-started> for up to date information

ESLINT

ADD ESLINT TO PACKAGE.JSON

```
{  
  // other contents of package.json  
  "scripts": {  
    "start": "http-server .",  
    "lint": "npx eslint ."  
  },  
}  
  
// npm run lint  
// npm run lint -- --fix  
//           ^^  
// are needed so npm passes --fix to eslint
```

ESLINT RULE CONFIGURATION

```
// .eslintrc.js

module.exports = {
  rules: {
    "no-await-in-loop": "off", // Possible values are "off", "warn", "error"
  }
}
```

ESLINT DISABLE RULES⁴

» sometimes disabling rules is ok

```
// eslint-disable-next-line max-len
const anExtremelyLongLine = "....."
```

```
const anExtremelyLongLine = "....." // eslint-disable-line max-len
```

⁴ see <https://eslint.org/docs/user-guide/configuring#disabling-rules-with-inline-comments> for all options

VERIFYING LINTING RULES

- » verify linting continuously
 - » on a CI server (eg. github actions)
 - » on a git hook (eg. pre-commit, pre-push hook)

GIT HOOKS

- » are executed when something happens
- » available hooks are

pre-commit // <- we'll be using this

pre-push

post-commit

post-push

//...

PRE-COMMIT HOOK WITH HUSKY⁶

- » Husky adds git hooks during installation
- » in package.json add the following:

```
// run: `npm install husky --save-dev`  
  
// add hooks to package.json  
{  
  "husky": {  
    "hooks": {  
      "pre-commit": "npm run lint",  
      "...": "..."  
    }  
  }  
}
```

⁶ see <https://www.npmjs.com/package/husky>

ESLINT AUTOFIX STAGED FILES BEFORE COMMIT¹⁵

- » lint-staged only lints files which have changed
- » adds autofix possibility to githook
- » installation npx mrm lint-staged

¹⁵ docs <https://github.com/okonet/lint-staged>



TASK TIME

TASK (20 MINUTES)

- » Go into your homework group
- » Add eslint and pre-commit hook to your homework
- » Fix all errors in your code
- » I'll try to join each room and help when I'm needed

JS BUNDLING

JS BUNDLING

“A bundler compiles small pieces of code into something larger and more complex, such as a library or application ⁷”

⁷ source <https://rollupjs.org/guide/en/>

WHY MODULES¹

- » Maintainability
- » Namespacing
- » Reusability

¹ <https://www.perforce.com/blog/qac/what-lint-code-and-why-linting-important>

JS BUNDLING

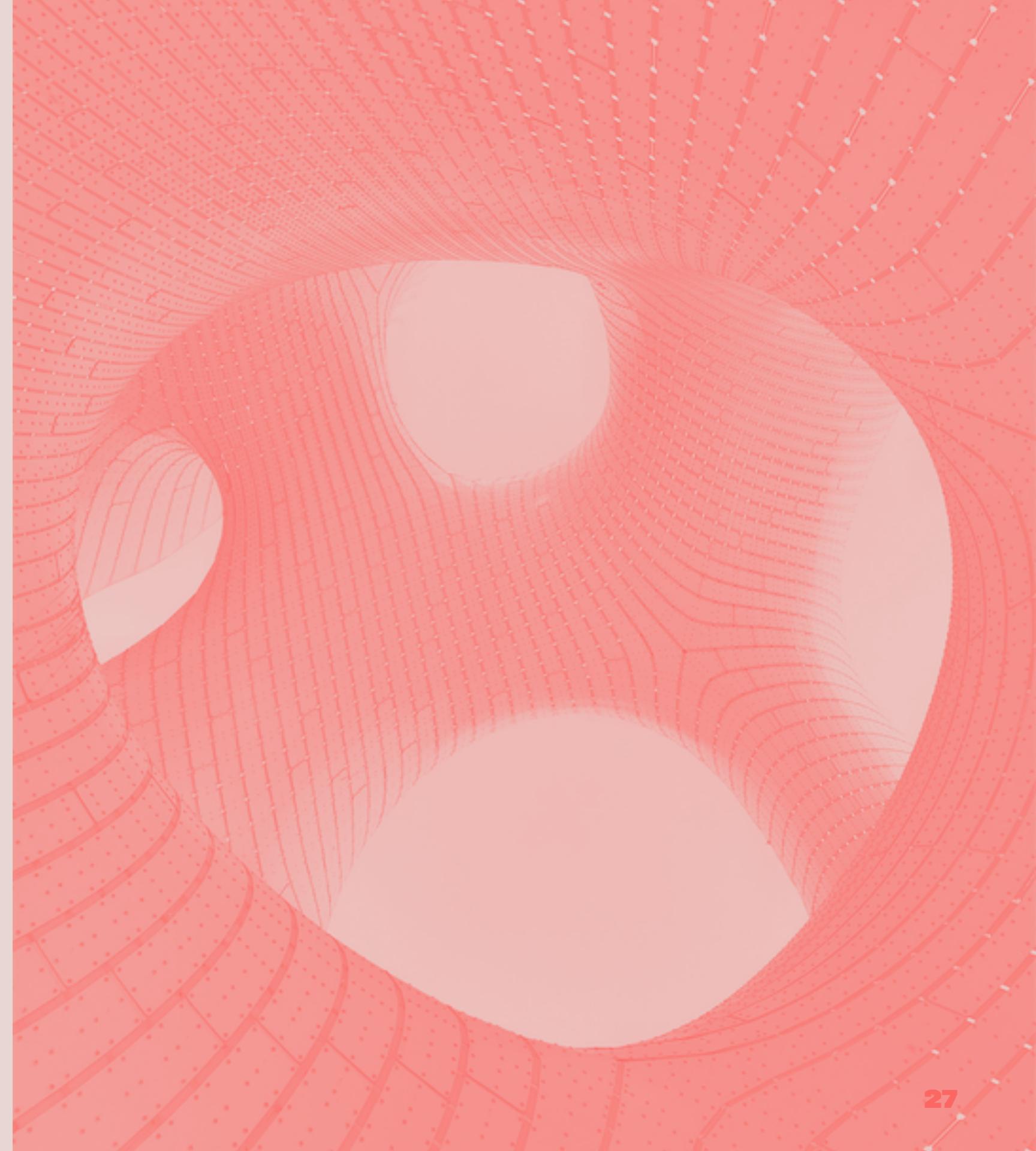
WHY BUNDLING

- » JS apps are a bundle of modules ⁸
- » older browsers might not understand JS modules
- » older node version might not understand JS modules

⁸ see module slides

JS BUNDLING TOOLS

- » Tools which resolve modules locally and bundle them together into a bigger bundle
- » Bundling tools are:
 - » Rollup
 - » Parcel
 - » Webpack



JS BUNDLING TOOLS WITH ROLLUP

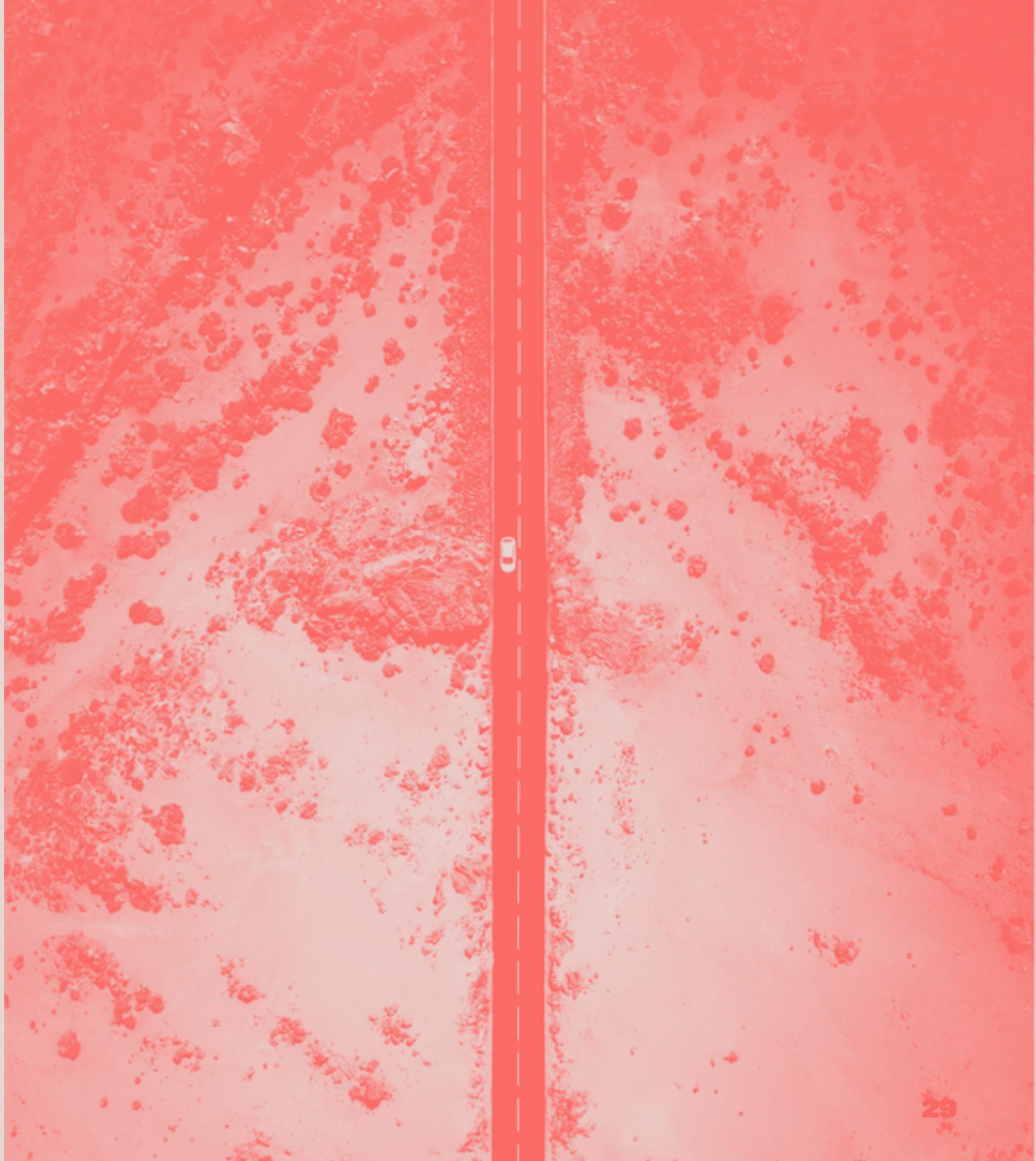
```
// rollup.config.js
export default {
  input: "index.js",
  output: [
    { file: "./build/index.bundle.js", format: "iife" },
  ],
};

// # add build directory to gitignore
// echo ./build > .gitignore

// # build the app
// npx rollup -c
```

TREE-SHAKING

- » when adding a module everything gets added to bundle
- » large libraries could increase the bundle unnecessarily
- » bundlers which support tree shaking remove unused parts



TREE-SHAKING EXAMPLE

```
// a.js
export const aFunction = () => { console.log('a') }
export const bFunction = () => { console.log('a') }

// b.js
import {aFunction} from './a.js'

// Result: b.bundle.js
const aFunction = () => { console.log('test'); };
aFunction();
```

JS BUNDLING TOOLS WITH ROLLUP (NO CONFIG REQUIRED)

```
npx rollup ./index.js -d build/
```

```
# bFunction won't be part of the bundle
```

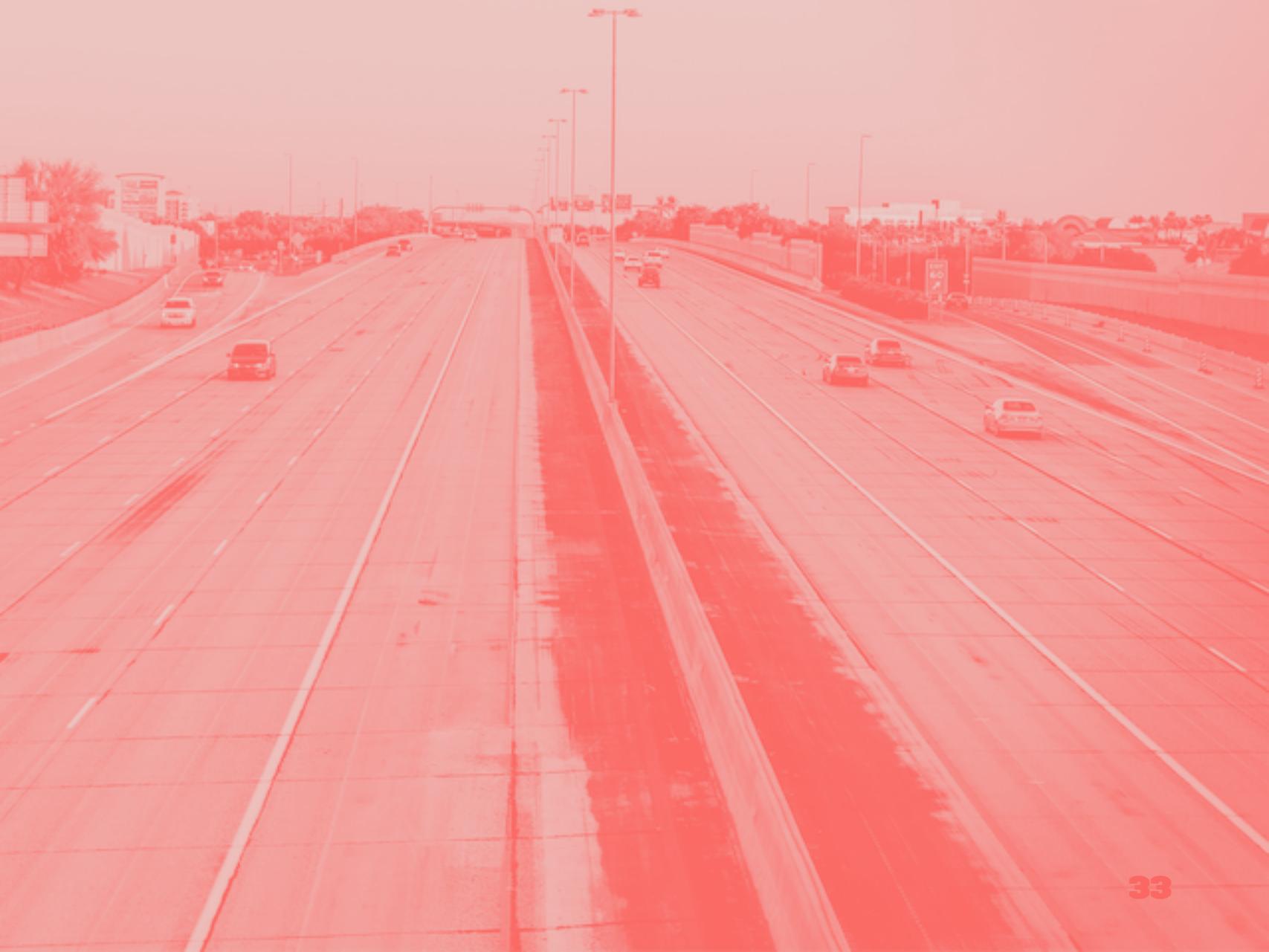
MINIFICATION⁹

“Minification refers to the process of removing unnecessary or redundant data without affecting how the resource is processed by the browser.”

⁹ source <https://developers.google.com/speed/docs/insights/MinifyResources#:~:text=Minification%20refers%20to%20the%20process,specific%20optimizations%20to%20learn%20more.>

MINIFICATION

- » decrease bundle/download size by removing
 - » comments
 - » formatting
 - » unused code
 - » shorten variable/function names
 - » ...
- » get better google page



MINIFICATION

- » minification can be done to:
- » js
- » html
- » css
- » ...



MINIFICATION

MINIFICATION WITH TERSER PLUGIN

```
» npm i rollup-plugin-terser
```

```
// rollup.config.js
import { terser } from "rollup-plugin-terser";

export default {
  input: "index.js",
  plugins: [terser()]
  //          ^^^^^^^^^^^^
  // add the terser plugin which will minify the sources
  output: [
    {
      file: "./build/index.bundle.js",
      format: "iife",
    },
  ],
};

};
```

SOURCEMAPS

- » debugging minified code is no fun
- » Sourcemap way to revert minimization of code
- » are defined as a special comment at the end of the file
- » external
 - » `//# sourceMappingURL=http://example.com/path/to/your/sourcemap.map`

CODE SPLITTING

- » split your JS bundle into multiple smaller bundles
- » eg. create a vendor bundle (with libraries)
- » this file can be cached by the browser
- » recurring users don't need to download libs twice

USING ESNEXT FEATURES IN LEGACY BROWSER

TRANSPILING

- » source to source translator
- » takes code and converts it to code of a different language
- » eg. convert ESNext for old browsers

TRANSPILING WITH BABEL & ROLLUP

» babel is a transpiler¹⁰

» it converts new syntax for the use in old
browsers

```
// eg. es6 arrow functions
const myFunction = () => {}
```

```
// will become
const myFunction = function myFunction () {}
```

¹⁰ see babel repl

TRANSPILING WITH BABEL & ROLLUP¹¹

```
# install babel transpiler
npm install --save-dev @babel/core @babel/preset-env

# install babel plugin for rollup
npm install --save-dev @rollup/plugin-babel
```

¹¹ installation instructions for other bundlers <https://babeljs.io/en/setup>

TRANSPILING CONFIGURE BABEL & ROLLUP

```
// rollup.config.js

import { terser } from "rollup-plugin-terser";
import babel from '@rollup/plugin-babel';

export default {
  input: "index.js",
  plugins: [
    babel({ presets: [['@babel/env', {}]] }),
//1) ^^^^^^
//2)           ^^^^^^^^^^

//1) add the babel plugin
//2) define the @babel/env

    // ... other plugins
  ],
};


```

TRANSPILING

@BABEL/PRESET-ENV

- » a preset is a collection of transformations
 - » eg. arrow function to function expression
- » preset-env is smart enough to only add transforms which are required
- » can be configured via .browserslistrc

// .browserslistrc

> 5% in AT

POLYFILLS



POLYFILLS¹³

“New language features may include not only syntax constructs and operators, but also built-in functions.”

¹³ source <https://javascript.info/polyfills>

POLYFILLS

- » Math.trunc removes the decimal part of a number
- » Math.trunc(1.23) === 1
- » older browsers might not implement Math.trunc
- » Polyfills patch the browser with new APIs

```
Math.trunc = function trunc(it) {  
  return (it > 0 ? floor : ceil)(it);  
}
```

POLYFILLS BUNDLE WITH OUR APPLICATION

» Install dependencies

```
# dependencies needed for bundling polyfills
npm i @rollup/plugin-node-resolve @rollup/plugin-commonjs --save-dev
```

```
# install polyfills
npm i core-js@3 --save
```

POLYFILLS CONFIGURE ROLLUP¹⁴

```
// rollup.config.js

import { terser } from "rollup-plugin-terser";
import babel from '@rollup/plugin-babel';
import resolve from '@rollup/plugin-node-resolve';
import commonjs from '@rollup/plugin-commonjs';

export default {
  input: "index.js",
  plugins: [
    babel({
      presets: [['@babel/env', {
        babelHelpers: 'bundled', // add require statements for polyfills
        exclude: 'node_modules/**',
        presets: [
          ['@babel/env', { "useBuiltIns": "usage", corejs: { version: 3 } }]
        ]
      }]]
    }),
    commonjs(), // signalize rollup that it should bundle commonjs modules
    resolve(), // inline libraries from node_modules
  ],
  // ...
};
```

¹⁴ complete configuration <https://gist.github.com/webpapaya/45c5aae75bbe4e8eb72bc19c33e080bf>



ASK TIME

TASK

- » Go into your homework group
 - » on a dedicated branch
- » Add rollup and build your app into one file
 - » add minification
 - » add babel
 - » play around with different .browserlistrc configs and see the difference in bundle size

HOMEWORK

- » Finish the quiz
- » when not already done during todays lecture
- » finish linting your application
- » I'll test via npm run lint
- » any error will result in -2 points

FEEDBACK

- » Questions: tmayrhofer.lba@fh-salzburg.ac.at
- » Feedback Link