MP2 TO MP3

LOGISTICS

- MP2 Share back!
- Intro to MP3
- Break, install Node
- Dev toolchain walkthrough and activity

Reminder: No in-person class next Tuesday

MP2 SHARE BACK!

- Add a link to your sketch gallery to *this google sheet*
- Go look at your peer’s galleries!

MP3

- Practice web dev using a modern toolchain!
- Requirements:
  - Uses at least two external libraries, installed with npm
  - Runs on a local development server
  - Build with Rollup
  - Deploy with gh-pages

PROJECT OPTIONS

- A game
  - Could use P5.play, phaser, pixijs, threejs
- Interactive data viz
  - D3, chart.js
- Something else!
  - *Talk to me first*

BREAK AND INSTALL NODE

Install Node using *the installer* from their website

DEV TOOLCHAIN OVERVIEW

1. Package manager
2. Local development server
3. Build tools, e.g. bundlers and minifiers
4. Deployment platform

Also - testing frameworks (important, but beyond the scope of this class)

EXAMPLE

Follow along by looking at my *example repository*

PACKAGE.JSON

- contains information about your app
- everything someone else needs in order to install, run, develop, and deploy

PACKAGE.JSON EXAMPLE

```
{
  "name": "cubing",
  "scripts": {
    "start": "web-dev-server --node-resolve --open --watch",
    "build": "rollup --config",
    "deploy": "npm run build && gh-pages -d dist"
  },
  "dependencies": {
    "three": "^0.149.0"
  },
  "devDependencies": {
    "@rollup/plugin-node-resolve": "^15.0.1",
    "@web/dev-server": "^0.1.35",
    "gh-pages": "^5.0.0",
    "rollup": "^2.79.1",
    "rollup-plugin-copy": "^3.4.0"
  }
```

PACKAGE.JSON CONTENTS

main things of interest:

- **`scripts`**: useful scripts that you can run, relevant to your project
- **`dependencies`**: libraries your app needs to run
- **`devDependencies`**: dependencies you use *during development*

INSTALLING DEPENDENCIES

- Until now, we've used a CDN, which has limitations
- Two main options: NPM and yarn
- We will use NPM (node package manager)
- NPM is to JavaScript what Pip is to Python

NPM

- packages are installed to a folder called `node_modules`
- installing a dependency: `npm install packagename`
- installing a dev dependency: `npm install --save-dev packagename`
- automatically updates `package.json`

LOCAL DEVELOPMENT

- We’ve been using vscode’s live server extension
- Presents a problem when using installed dependencies!
- Need to import them from `node_modules`
- Need to use a server that can *resolve imports*

DEV SERVER OPTIONS

- web dev server (what we will be using)
- webpack dev server (also great, works well with webpack)
- vite
- others

IN THE EXAMPLE

- Installed Three.js and an example application
- Installed a local development server
- Added a `start` script:

```
{
  "start": "web-dev-server --node-resolve --open --watch"
}
```

BUILDING FOR DEPLOYMENT

Building can involve a number of tasks:

- Putting all the code we need in one place
  - minifying our JS
  - copying over assets
- Don’t want to include code only used for development (e.g. our dev server)

USING ROLLUP

- Installed rollup and two plugins dev dependencies
- Added a script to run rollup using our config

```
{
  "build": "rollup --config"
}
```

Run it with `npm run build`

ROLLUP.CONFIG.JS

```
import { nodeResolve } from "@rollup/plugin-node-resolve";
import copy from "rollup-plugin-copy";

module.exports = {
  input: "index.js",
  output: {
    dir: "dist",
  },
  plugins: [
    copy({
      targets: [{ src: "index.html", dest: "dist" }],
    }),
    nodeResolve(),
  ],
};
```

DEPLOYING

Again, many options!

- Github pages (used by the example)
- Render
- Heroku
- Firebase
- more

IN THE EXAMPLE

- installed the `gh-pages` package
- made a script to push everything in the `dist` folder to the gh-pages branch:

```
{
  "deploy": "npm run build && gh-pages -d dist"
}
```

Run it with `npm run deploy`