

# PacMan's global rethinking

Simplon - Grenoble | technical director Renaud Dechaux

Study and realisation Syriane B.

# Problems

## Only one file

The code is concentrated in a single file which is imported into an html. It is difficult to maintain because it is not very structured for a developer who arrives on the project

## Browsers compatibility

Although written in js compatible with current browsers, it would be nice to secure our backs for the future by ensuring our source code is compiled for older browsers.

## Project weight

The code is currently not minified on the production, its weight and loading time could be improved.

# Details of the challenges to be won

**1st step**

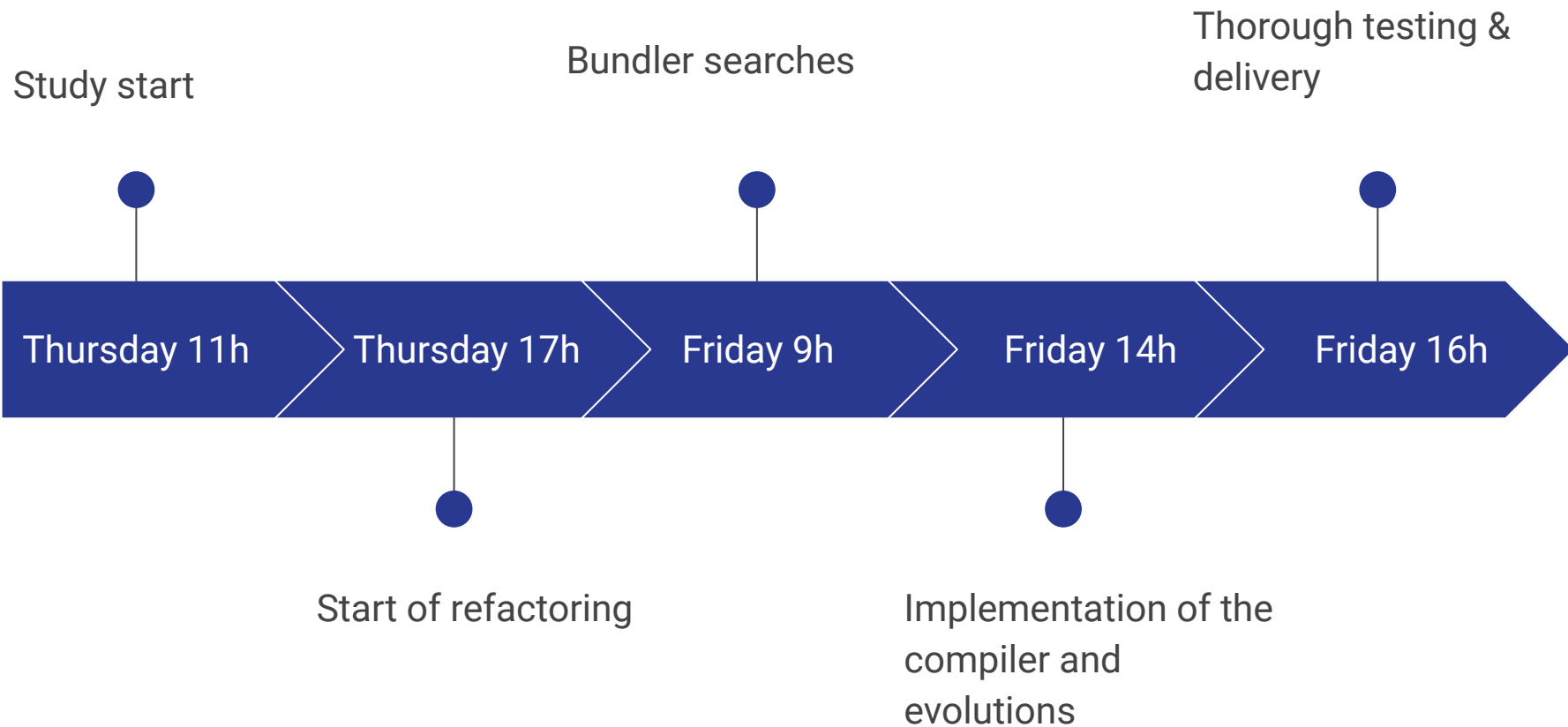
Code study

**2nd step**

Identification of  
possible solutions

**3rd step**

File structuring and  
tests



# Code study

The project code requires refactoring to allow:

- more easily modify game variables
- semantically separate the different elements of the game
- allow in the future to retrieve new variables from an API
- enrich the project with new features:
  - end of Game
  - more different levels
  - colorful css theme customization

---

# Folder organisation


## Useful documentation :

- <https://blog.logrocket.com/the-perfect-architecture-flow-for-your-next-node-js-project/>
- <https://stackoverflow.com/questions/5178334/folder-structure-for-a-node-js-project>
- <https://gist.github.com/lancejpollard/1398757>

## Choices made for our project :

According to the documentation and node recommendation, we can split js code into multiple file. We can order them in directories to help general project code comprehension.

I choose to separate the different classes in a Model directory and an other GameLogic folder. The js code will be splitted in several files. One of them will contain all the gameValues, in the future it could be replaced with a server response.



# Identified bundlers

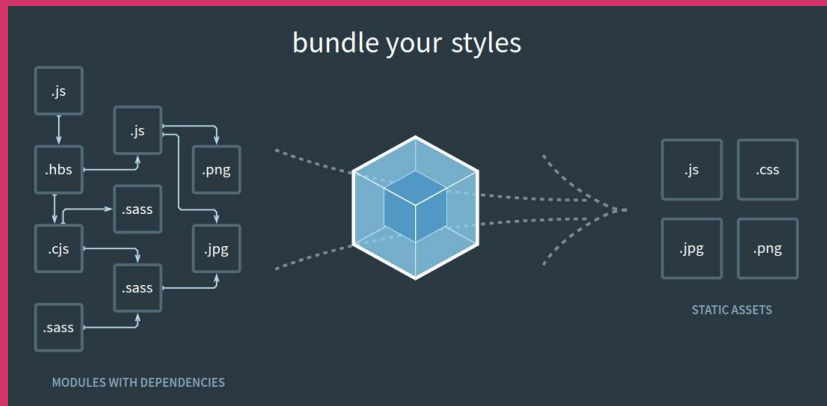
## Parcel

<https://parceljs.org/>



## Webpack

<https://webpack.js.org/>



# Parcel | Pros & cons



- light
- fast to setup
- no configuration required
- babel transpilation included
- automatic file watch in dev mode



- no configurable settings
- Almost 600 issues on github





# Webpack | Pros & cons

+

- Big reactive community
- team is already trained on this technology
- lot of different bundles availables
- same technology uses on our other symfony projects
- prod and dev mode can be setup

-

- need to setup configuration



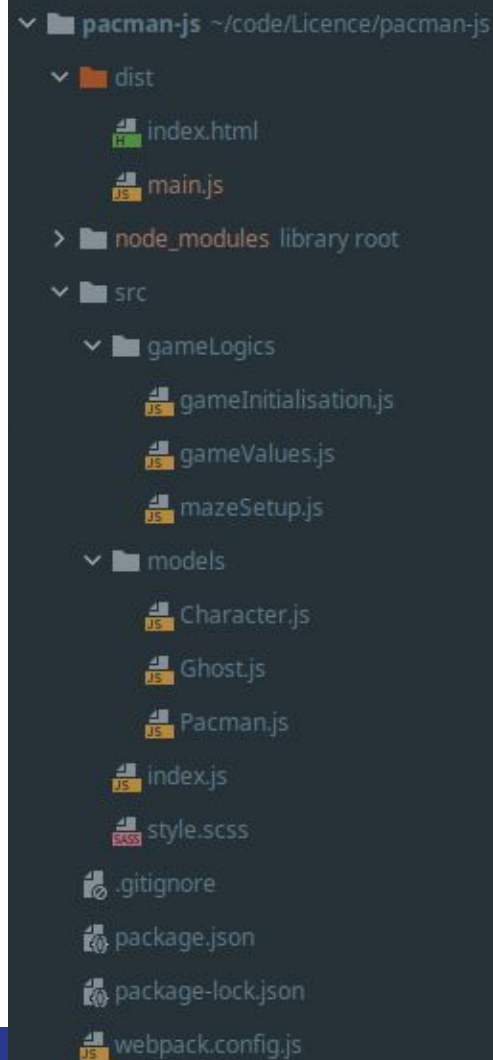


Winner is .....

Webpack !

# New folder architecture

- Dist folder contain the bundled code, the one we need on production server
- node\_modules contains our dependencies
- src folder contain all our code, divided in several file organised in folders.
- webpack.config.js gather all the compilation setup.



# New features

## Sass instead of Css

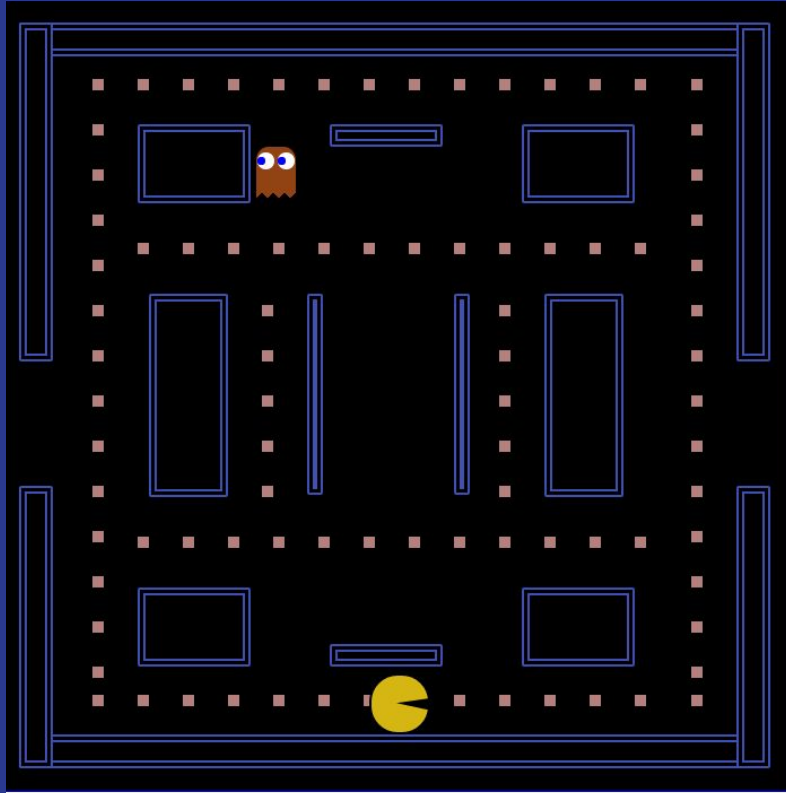
We can now use variables in css code to choose colors or speeds (for pacman mouth for example).

## Ghosts and Pacman stop

At the end of the game, either if its a victory or gameover, Ghosts stop moving and generating, and Pacman mouse top moving.

## Last minute needs

Choose game level and cookie saving the values.



Let's go!