Зависимости в приложении

Щепотьев Евгений

HTML Academy

evgenii.schepotiev@gmail.com

@zeckson

“При написании этих модулей ни одной структуры данных не пострадало”

Но это не точно

Модуль *config.js* — загружает настройки

```
const env = typeof process === `undefined` ? window : process.env;

const config = {
  appName: `Demo APP`,
  isDev: env.NODE_ENV !== `production`
};

module.exports = config;
```

Модуль *logger.js* — настраивает вывод логов

```
const config = require(`./config`);

const logger = {
  debug(message) {
    if (config.isDev) {
      console.log(message);
    }
  }
};

module.exports = logger;
```

Модуль *main.js* — входная точка

```
const config = require(`./config`);
const logger = require(`./logger`);

logger.debug(`The application "${config.appName}" is up and running!`);
```

```
node main.js

The application "Demo APP" is up and running!

Process finished with exit code 0
```

Добавим в *config.js* логов

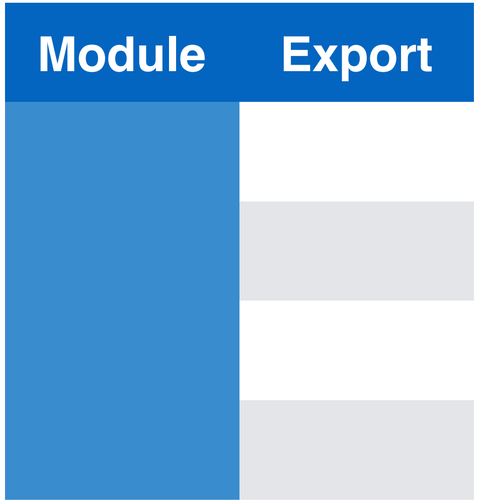
```
const logger = require(`./logger`);

const env = typeof process === `undefined` ? window : process.env;

const config = {
  appName: `Demo APP`,
  isDev: env.NODE_ENV !== `production`
};

logger.debug(config);

module.exports = config;
```

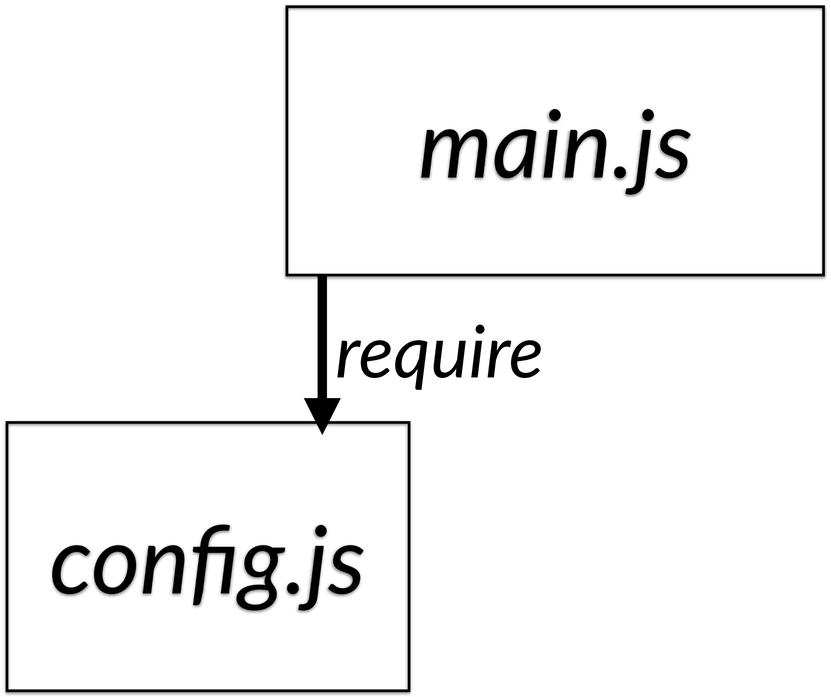
Модуль *main.js* — входная точка

```
const config = require(`./config`);
const logger = require(`./logger`);

logger.debug(`The application "${config.appName}" is up and running!`);
```

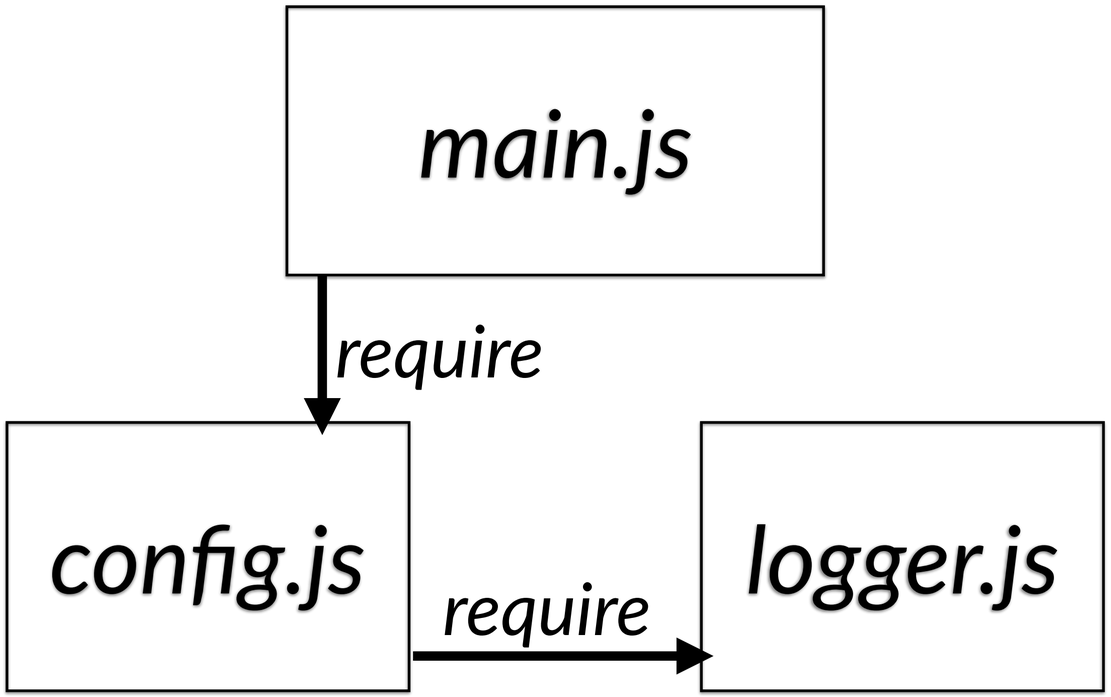
```
node main.js

Process finished with exit code 0
```

main.js

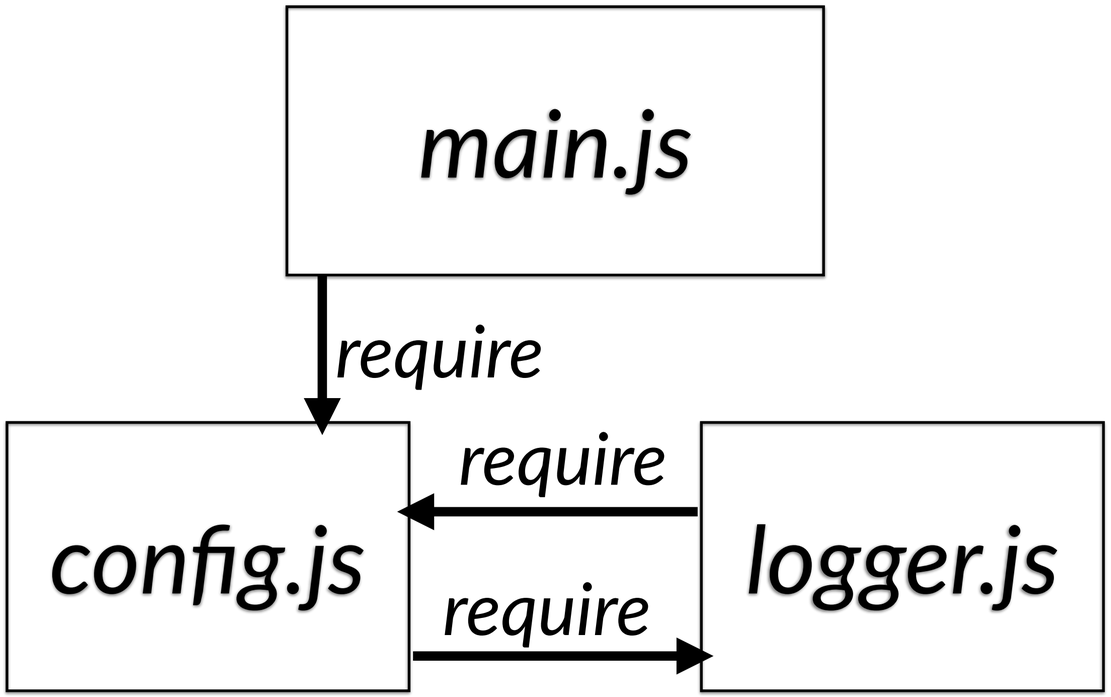




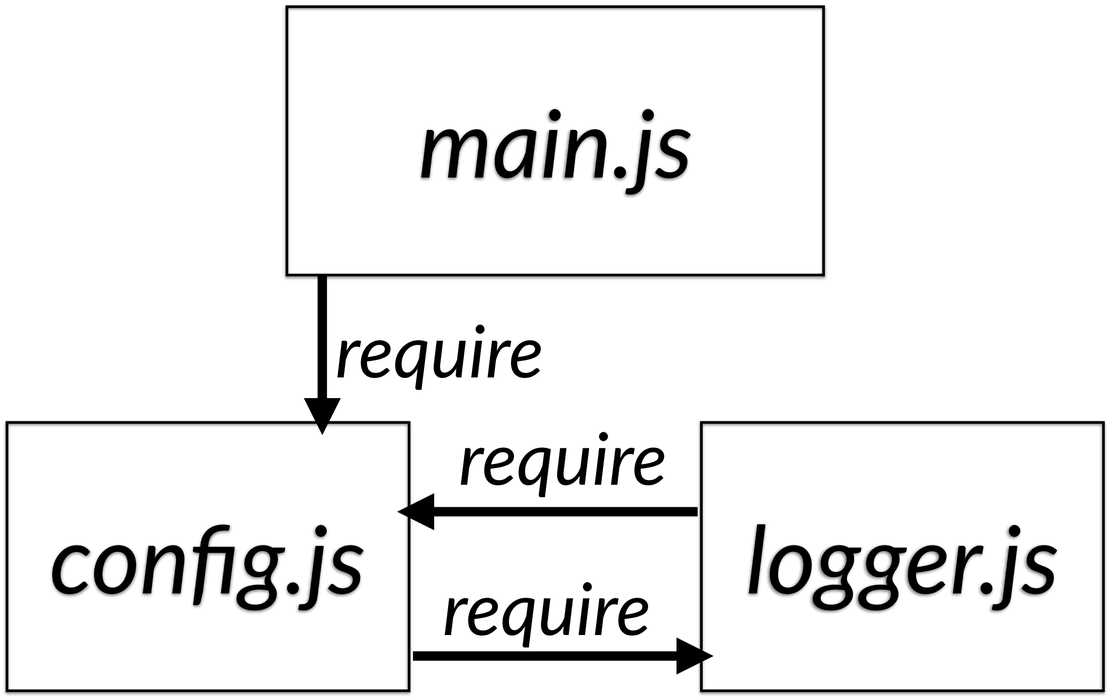
Module	Export
main.js	{}



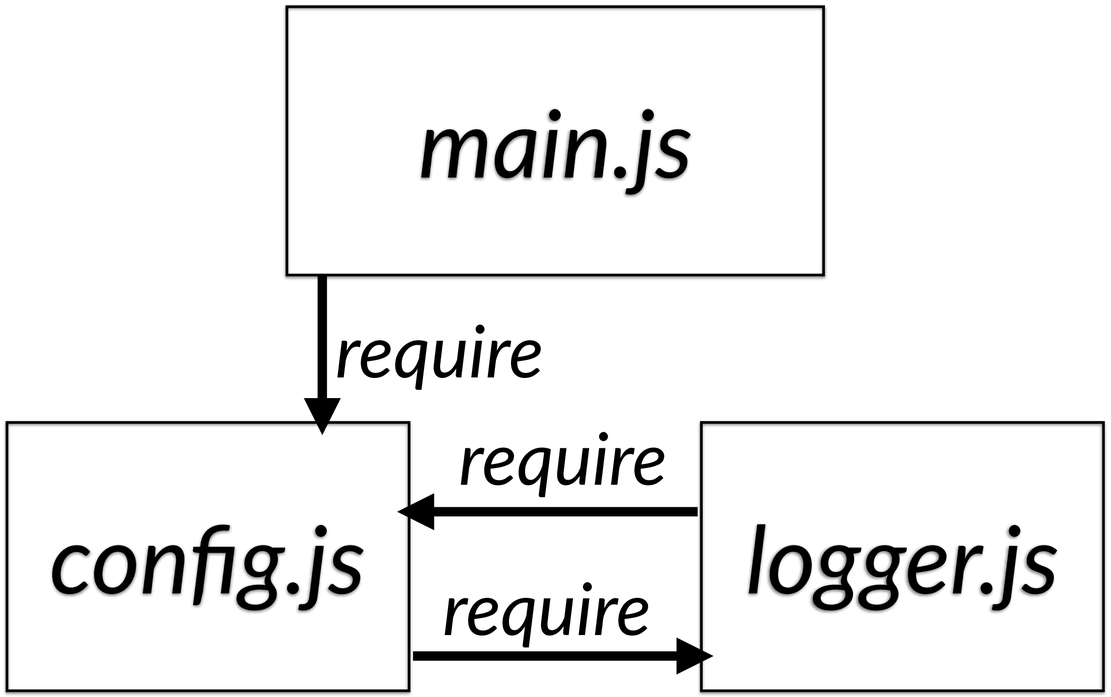
Module	Export
main.js	{}
config.js	{}



Module	Export
main.js	{}
config.js	{}
logger.js	loaded



Module	Export
main.js	{}
config.js	loaded
logger.js	loaded



Module	Export
main.js	loaded
config.js	loaded
logger.js	loaded

Модуль *logger.js* — отладка

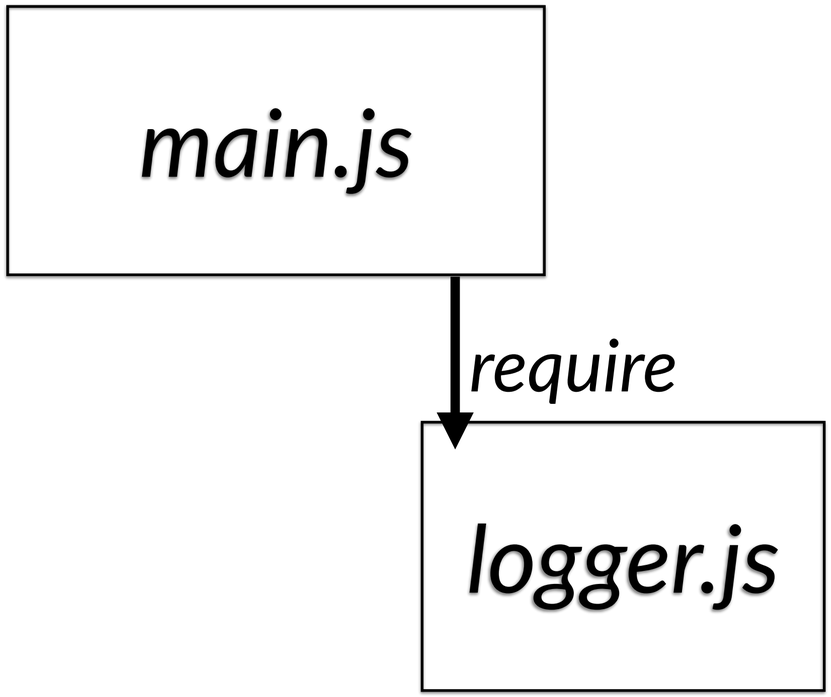
```
const config = require(`./config`);     config: Object {}

const logger = {
  debug(message) {
    if (config.isDev) {     config.isDev: undefined
      console.log(message);
    }
  }
};

module.exports = logger;
```

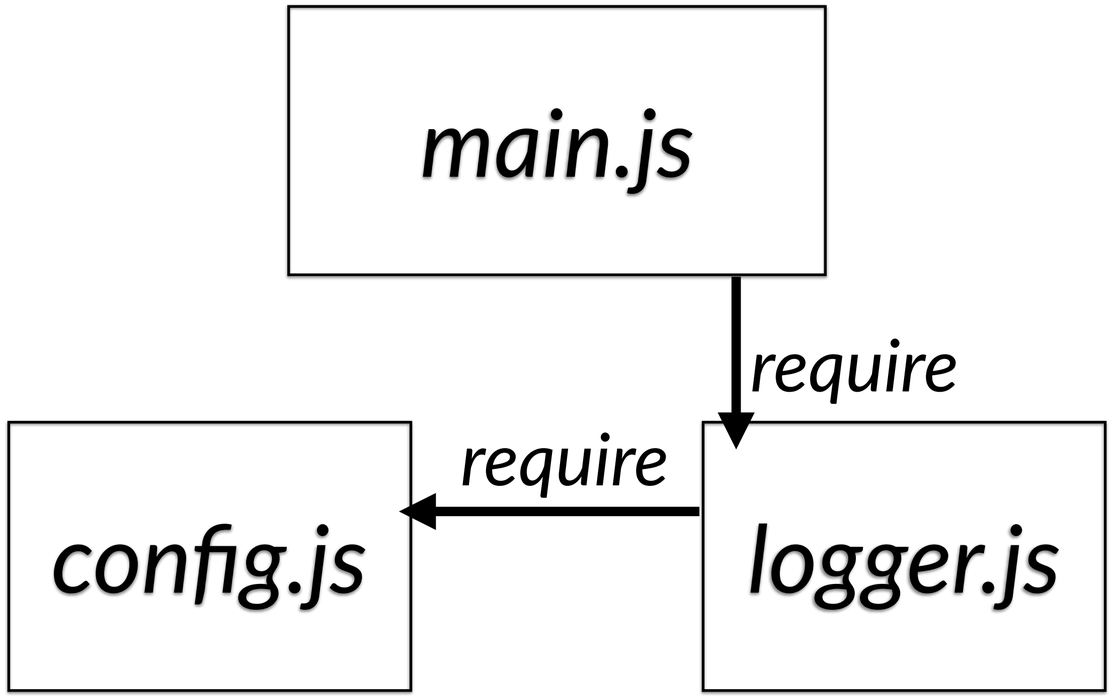
Модуль *main.js* — поменяем модули местами

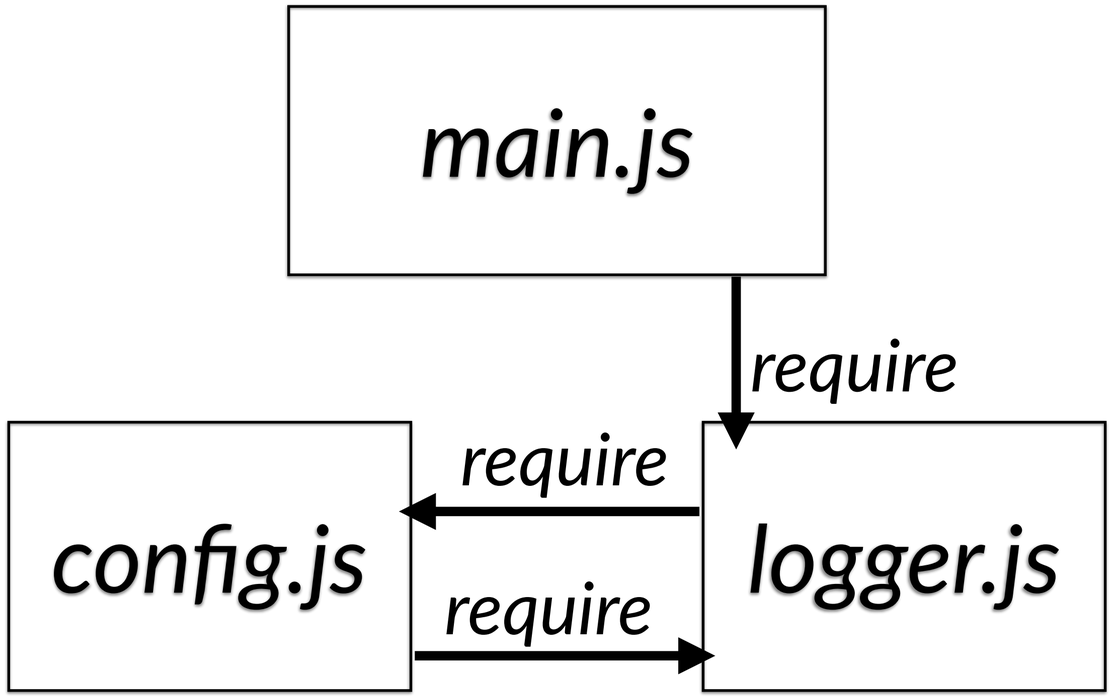
```
const logger = require(`./logger`);
const config = require(`./config`);

logger.debug(`The application "${config.appName}" is up and running!`);
```

```
/config.js:10
logger.debug(config);
       ^

TypeError: logger.debug is not a function
    at Object.<anonymous> (/config.js:10:8)
    at Module._compile (module.js:649:30)
...
```

main.js


main.js
require
logger.js


main.js
require
require
config.js
logger.js


main.js
require
require
config.js
logger.js
require

Итого

Итого

– Мы получили цикл в зависимостях

Итого

– Мы получили цикл в зависимостях

– Поведение программы недетерменировано

CommonJS модули

```
const config = require(`./config`);
const logger = require(`./logger`);

logger.debug(`The application "${config.appName}" is up and running!`);
```

Node.js

Не выводит сообщение
Нет ошибок исполнения

Webpack

Не выводит сообщение
Нет ошибок исполнения

ECMAScript 2015 модули

```
import config from './config.js';
import logger from './logger.js';

logger.debug(`The application "${config.appName}" is up and running!`);
```

Node.js

```
ReferenceError:
config is not defined
```

Browser (Chrome 64)

```
ReferenceError:
config is not defined
```

Webpack

```
TypeError:
Cannot read property
'isDev' of undefined
```

ECMAScript 2015 (Rollup)

```
(function () {
'use strict';

const logger = {
  debug(message) {
    if (config.isDev) {
      console.log(message);
    }
  }
};

const env = typeof process === `undefined` ? window : process.env;

const config = {
  appName: `Demo APP`,
  isDev: env.NODE_ENV !== `production`
};

logger.debug(config);

logger.debug(`The application "${config.appName}" is up and running!`);
```

Проблемы циклических зависимостей

Проблемы циклических зависимостей

– Поведение зависит от порядка подключения модулей

Проблемы циклических зависимостей

– Поведение зависит от порядка подключения модулей

– Поведение зависит от загрузчика модулей

Проблемы циклических зависимостей

– Поведение зависит от порядка подключения модулей

– Поведение зависит от загрузчика модулей

– Поведение зависит от вида модулей

Проблемы циклических зависимостей

– Поведение зависит от порядка подключения модулей

– Поведение зависит от загрузчика модулей

– Поведение зависит от вида модулей

– Непредсказуемость поведения приложения

Как найти цикл

Как найти цикл

– В продакшене

Как найти цикл

- В продакшене
  - Поймать ошибку на клиенте

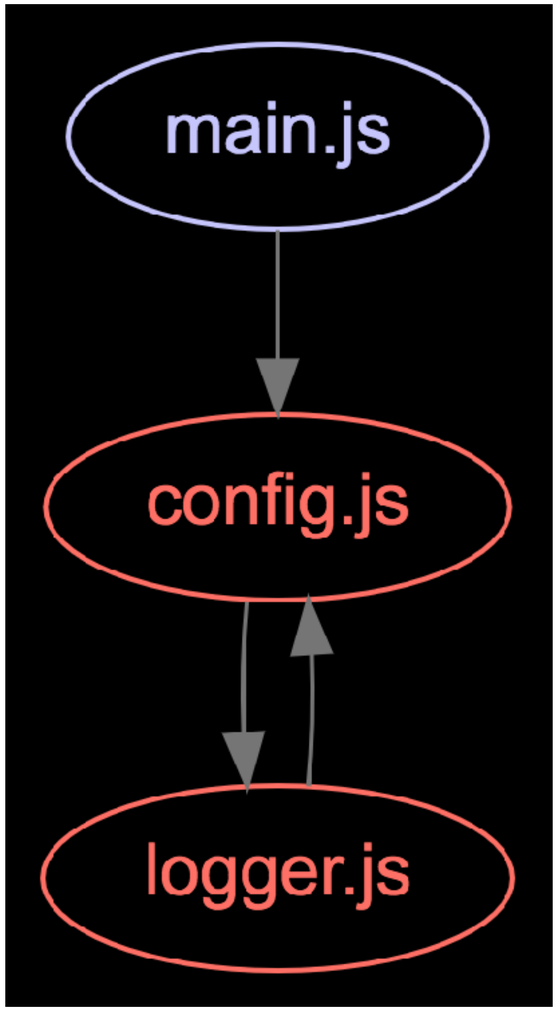
Как найти цикл

- В продакшене
  - Поймать ошибку на клиенте
  - Дождаться отзывов пользователей

Как найти цикл

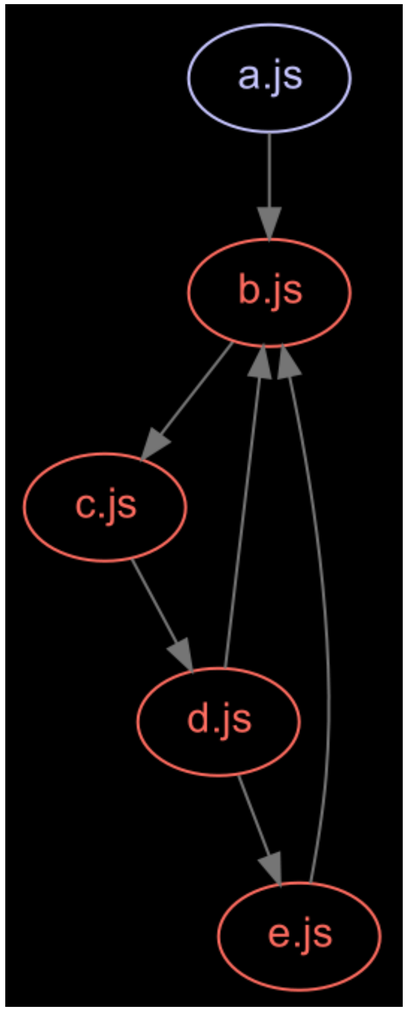
- В продакшене
  - Поймать ошибку на клиенте
  - Дождаться отзывов пользователей
- Использовать утилиту, которая анализирует зависимости в приложении

madge



github.com/pahen/madge

madge



github.com/pahen/madge

Как избавиться от цикла

Как избавиться от цикла

– Обобщение: вынести общий код в родительский модуль

Как избавиться от цикла

– Обобщение: вынести общий код в родительский модуль

– Инверсия: отделить зависимость от использования

Вынести общий код в родительский модуль

```
// Файл config.js
const env = typeof process === `undefined` ? window : process.env;

const config = {
  appName: `Demo APP`,
  isDev: env.NODE_ENV !== `production`
};

module.exports = config;

// Файл main.js
const config = require(`./config`);
const logger = require(`./logger`);

logger.debug(config);
logger.debug(`The application "${config.appName}" is up and running!`);
```

Модуль *config.js*

```
const env = typeof process === `undefined` ? window : process.env;

const config = {
  appName: `Demo APP`,
  isDev: env.NODE_ENV !== `production`,
  print(logger) {
    logger.log(config);
  }
};

module.exports = config;
```

Модуль *logger.js*

```
const logger = {
  log(message) {
    if (this.config.isDev) {
      console.log(message);
    }
  },
  init(config) {
    this.config = config;
  }
};

module.exports = logger;
```

Отделить зависимость от использования

```
const logger = require(`./logger`);
const config = require(`./config`);

logger.init(config);
config.print(logger);

logger.debug(`The application "${config.appName}" is up and running!`);
```

Dependency Inversion

Логгеру всё равно какой конфиг и откуда ему его передадут

Конфигу всё равно в какой логгер писать

Dependency inversion

github.com/gedbac/di4js

Dependency inversion

– Плюсы:

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями
  - Изолирует использование от содержимого

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями
  - Изолирует использование от содержимого
  - Облегчает написание тестов и увеличивает поддерживаемость кода

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями
  - Изолирует использование от содержимого
  - Облегчает написание тестов и увеличивает поддерживаемость кода
- Минусы:

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями
  - Изолирует использование от содержимого
  - Облегчает написание тестов и увеличивает поддерживаемость кода
- Минусы:
  - Появляется дополнительная фаза настройки

github.com/gedbac/di4js

Dependency inversion

- Плюсы:
  - Уменьшает связанность между модулями
  - Изолирует использование от содержимого
  - Облегчает написание тестов и увеличивает поддерживаемость кода
- Минусы:
  - Появляется дополнительная фаза настройки
  - Логика связывания может быть неочевидна

github.com/gedbac/di4js

Заключение

Заключение

– Держите модули как можно менее связанными — Low Coupled

Заключение

– Держите модули как можно менее связанными — Low Coupled

– Для сильно связанных модулей лучше отделять интерфейс от использования

Заключение

– Держите модули как можно менее связанными — Low Coupled

– Для сильно связанных модулей лучше отделять интерфейс от использования

– Если от цикла не избавиться убедитесь, что он ведёт себя ожидаемо

Заключение

- Держите модули как можно менее связанными — Low Coupled
- Для сильно связанных модулей лучше отделять интерфейс от использования
- Если от цикла не избавиться убедитесь, что он ведёт себя ожидаемо
- Если зависимости сильно кастомоизированы в приложении, то возможно стоит использовать Dependency Inversion

Спасибо

Щепотьев Евгений

evgenii.schepotiev@gmail.com

@zeckson