
Яндекс

Яндекс

NPM

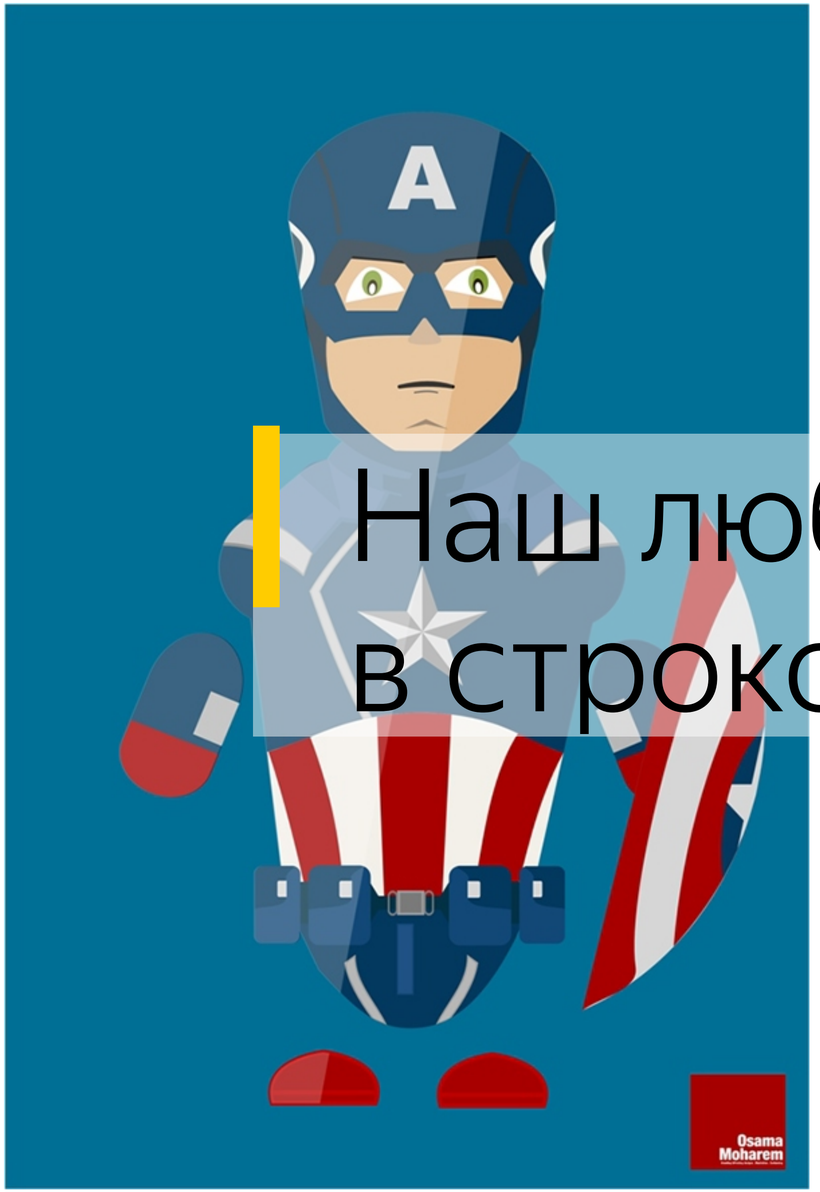
В предыдущих сериях

Знаем, что в node.js require позволяет заиспользовать код из другого файла, если этот другой файл предоставляет этот код в module.exports

Если путь явно не указан, то поиск модулей идет в подпапке ./node_modules, а также в «глобальной папке», в которую установлен npm

Во фронтэнде мы уже сталкивались с npm и знаем, что команда npm i приносит модули в папку node_modules

Давайте подробно разберемся в том, как это на практике работает, что является необходимым для приложения, что - для библиотеки, и как это все вместе живет



Наш любимый шаблонизатор в строковом виде

Публикация пакета

- Регистрируемся на https://www.npmjs.com
- Подтверждаем почту
- npm init
- npm publish
- ?????
- PROFIT!!

Использование пакета

Сейчас «пакет» пока подозрительно простой, состоит из одной функции, и на фронтенде достаточно просто тега <script>, с соответствующим src

Для node.js окружения (если мы хотим одним пакетом шаблонизировать и в браузере и на сервере) так работать не будет, там у нас require и module.exports.

Можно ли подготовить пакет так, чтобы он работал в обоих окружениях? Да, мы тоже уже это знаем

Добавляем module.exports и перевыпускаем пакет

Использование пакетов в разных окружениях

Вы не всегда будете иметь возможность «контролировать» пакет, а создатель пакета не всегда будет поддерживать возможность работы в разных окружениях.

Тем не менее, существуют разные решения (например, browserify и подобные) которые тем или иным образом решают проблему подключения node.js пакетов в браузерное окружение.

Пакеты, которые используют DOM никак не заиспользовать в node.js окружении

Запуск в командной строке

Command line interface

Мы умеем запускать js файлы командой node.

Но часто Javascript-приложения (такие как gulp, webpack, и так далее - тысячи их) запускаются просто, командой типа webpack прямо в консоли.

При этом оставаясь проектами, написанными на js.

Но как?

Создаем файл cli.js

```
#!/usr/bin/env node

const [,, ...args] = process.argv;

console.log(`Hello world, ${args}`);
```

Теперь cli.js можно запускать без node

Добавляем в package.json

```
"bin": {
    "my-awesome-command": "./cli.js"
}
```

Перевыпускаем пакет

Command line interface

Теперь после установки вашего пакета в папке проекта будет доступна команда npx my-awesome-command

Если же установить пакет глобально, то приставка npx не понадобится, но в этом случае версия пакета будет едина для всех проектов (это не всегда желательно, если, например, в разных проектах у вас разная версия шаблонизатора/сборщика/другое)

Зависимости vs dev-зависимости

Яндекс

Express

Express

Вы уже знаете модуль http, который позволяет сделать базовый веб-сервер и отдавать что-либо в ответ на HTTP запросы.

Использовать его для полноценного приложения средней руки бывает неудобно, маловато возможностей.

Познакомимся с express - фреймворком, который позволит мыслить также, как с http, но может больше

Hello world

npm i express --save

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => res.send('Hello World!'));

app.listen(port,
  () => console.log(`Example app listening on port ${port}!`)
);
```

???????

PROFIT!!!!

Роутинг

Роутинг

Роутинг осуществляется максимально просто - для любого типа запроса вида

app.post
app.put
app.delete

И подобные. Таким образом, вы можете обрабатывать любой тип запроса единообразно, даже не задумываясь о том, какие там детали HTTP запроса

Ответы

Ответ на запрос

У экспресса полная корзинка простых методов для разнообразных ответов на запрос

```
res.send();
```

```
res.sendStatus();
```

```
res.json();
```

```
res.redirect();
```

```
res.download();
```

и так далее

Передача данных и более сложный роутинг

Роутинг посложнее

Передача данных в «продвинутых» (RESTful-like) API часто осуществляется в url, без использования GET параметров, вот так

http://домен/users/1 - где 1 - это, например, id пользователя.

В express легко поддержать такой запрос

```
app.get('/users/:userId', function (req, res) {
    res.send(req.params)
});
```

GET-параметры запроса прорастают аналогично в req.query

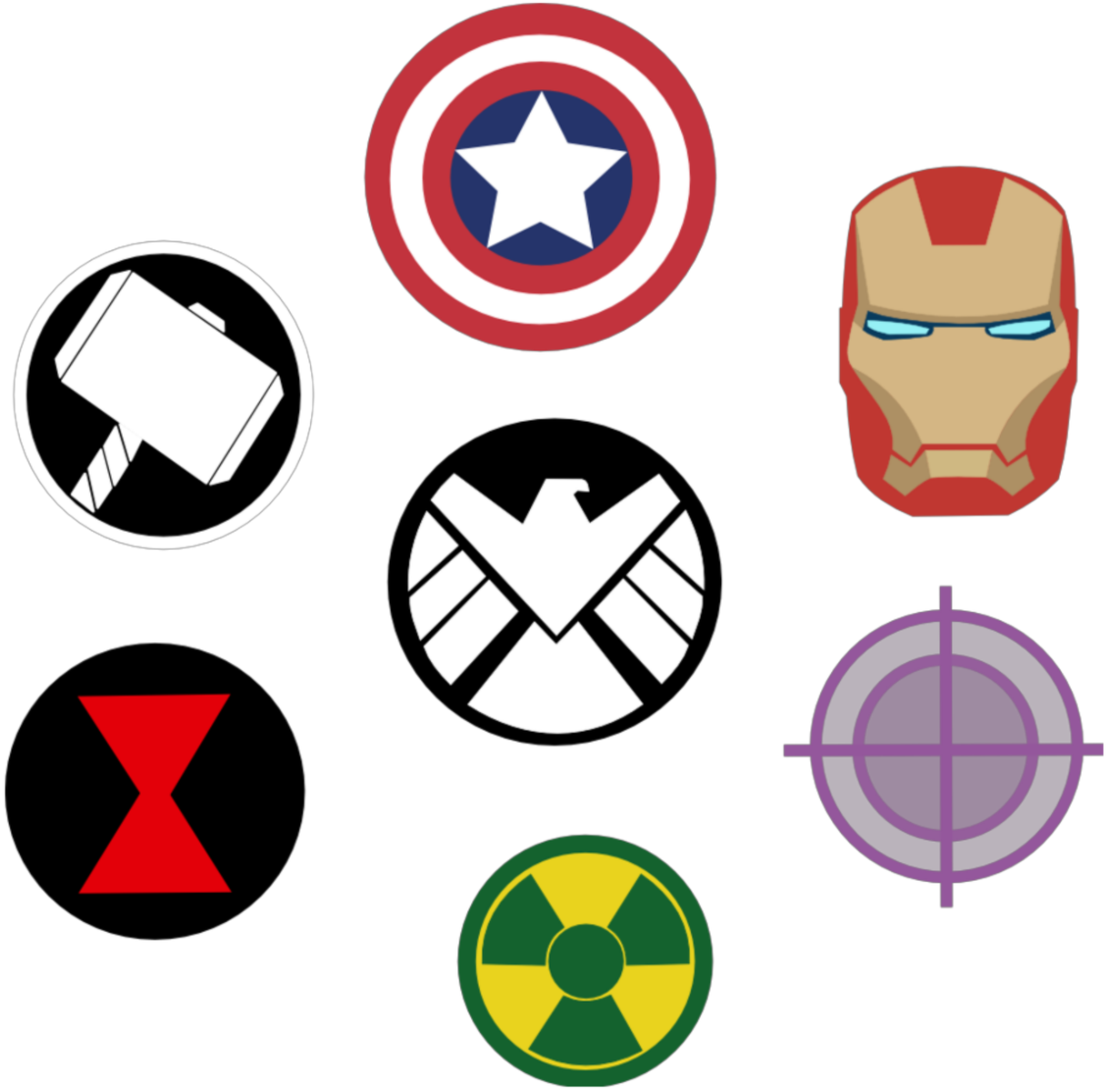
POST данные

Если вам надо передать данные в POST-запросе (или запросе другого типа, предполагающего передачу данных), то ищите их в req.body

В express легко поддержать такой запрос

```
app.post('/user/register', function (req, res) {
    res.send(req.body)
});
```

Миддлвары



app.use

Существует большое количество дополнительных модулей про http взаимодействие, безопасность, куки, которые подключаются как «миддлвары». Чтобы это не выглядело магией, рассмотрим простейший кастомный миддлвар.

```
var myLogger = function (req, res, next) {
    console.log('LOGGED');
    next();
}

app.use(myLogger);

app.get('/', function (req, res) {
    res.send('Hello World!')
});
```

Статика

Яндекс

Спасибо

Шлейко Александр

Разработчик интерфейсов

 dusty@yandex-team.ru

 @dustyo_O