ОНЛАЙН-ОБРАЗОВАНИЕ

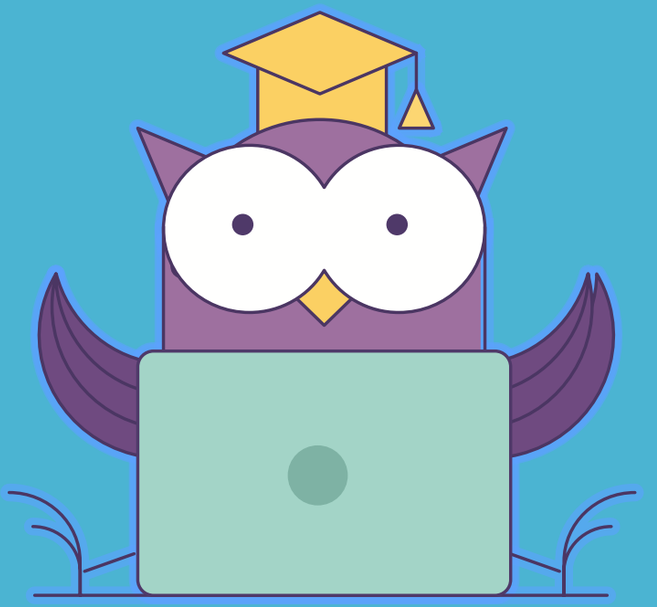
OTUS

Modern JavaScript Frameworks

Стандартная библиотека Node

Александр Коржиков

Как меня слышно и видно?



> Напишите в чат

+ если все хорошо

– если есть проблемы со звуком или с видео

- Web Assembly

Друзья! Мы думаем добавить в курс еще одну тему - `WebAssembly`.
Мы рассмотрим, что представляет из себя `WebAssembly`, и где его можно использовать уже сегодня.
Подробно разберем архитектуру `JavaScript` движка, на примере `v8`, для того, чтобы лучше понять, какое место занимает
И рассмотрим пару примеров.

Интересна ли вам такая тема?

- https://ru.wikipedia.org/wiki/WebAssembly
- https://webassembly.org/
- https://webassembly.studio/

Темы предыдущего занятия

- Unit-тестирование
- TDD
- Mocks
- BDD
- DDT

Цели занятия

- Разобраться с шаблоном проектирования **Module** и вариантами его имплементации в **JavaScript**
- Импорт и экспорт зависимостей в **Node** с
  - **CommonJS**,
  - **ES Modules**

- Modules
  - Pattern
  - Classic
  - AMD
  - CommonJS
  - ES Modules
  - Native ES Modules - Gil Tayar

- Node Modules API
- Learning JavaScript Design Patterns Addy Osman
- ES modules: A cartoon deep-dive - Lin Clark
- Mocks Aren't Stubs - Martin Fowler


BACK
TO THE FUTURE

Commands

- General
  - **init** - new package
  - **install** - dependencies
  - **start** - run application
  - **test** - run tests
  - **run** - any script
- Other
  - **ci** - install exact package versions
  - **npx** - run binary

- **dependencies**
  - **devDependencies** разработчика
  - **peerDependencies** плагины
  - bundleDependencies дистрибутив (сокращает время установки зависимостей)
  - **optionalDependencies** необязательные
- **--global**
- **node_modules**

```
"dependencies": {
  "commander": "^2.7.1",
  "lodash.get": "^4.0.0",
  "lodash.isequal": "^4.0.0",
  "validator": "^9.0.0"
},
"devDependencies": {
  "coveralls": "^3.0.0",
  "grunt": "^1.0.1",
  "grunt-browserify": "^5.2.0",
  "grunt-cli": "^1.2.0",
  "grunt-contrib-copy": "^1.0.0",
  "grunt-jscs": "^3.0.1",
  "grunt-lineending": "^1.0.0",
  "jasmine-node": "^1.14.5",
  "jasmine-reporters": "^2.2.1",
  "remapify": "^2.1.0"
}
```

- Locks
  - package-lock.json
  - npm-shrinkwrap.json
- Альтернативы
  - yarn
  - bower
  - turbo
- Proposals
  - tink

```
# A{B,C}, B{C}, C{D}
A
+-- B
+-- C
+-- D

# A{B,C}, B{C,D@1}, C{D@2}
A
+-- B
+-- C
    +-- D@2
+-- D@1
```

NPM Q&A

Архитектурный шаблон проектирования, помогающий организовать отдельные части кода

- Classic
- AMD
- CommonJS
- ES Modules

```
// Global module
var myModule = (function (jQ, _) {
  function privateMethod1() {
    jQ(".container").html("test")
  }
  return {
    publicMethod: function () {
      privateMethod1()
    }
  }
})(jQuery, _)

myModule.publicMethod()
```

- Immediately-Invoked Function Expression (IIFE)
- Object Notation
- Namespace with **init()** method

```
define('myModule', // name
  ['jQuery'], // dependencies
  function myModuleFactory($) { // factory
    function writeTest() {
      $('.container').html('test')
    }
    return {
      init: () => {
        writeTest()
      }
    }
  }
)
// or
define(function (require, exports, module) {
  // ...
})
```

- **require()**
- **define()**

```
// require
const circle =
  require('./circle.js')
// exports
exports.square =
  (r) => r ** 2
```

Node

- **require()** - импорт
- **exports, module.exports** - экспорт API из модуля

```
exports.version = 123
module.exports = { build, version }
```

- **require()** - импорт глобальных и локальных зависимостей
- **module, module.exports**

```
(function (exports, require, module, __filename, __dirname) {
  const circle = require('./circle.js')
  exports.square = (r) => r ** 2
})
```

```
// Node - lib/internal/modules/cjs/loader.js
Module.wrap = function(script) {
  return Module.wrapper[0] + script + Module.wrapper[1];
};

Module.wrapper = [
  '(function (exports, require, module, __filename, __dirname) { ',
  '\n});'
];
```

- **require()** кэширует модуль
- **require.cache** хранит загрузки
- **require.resolve()** возвращает путь до зависимости

Вопрос

Что будет выведено в консоль?

```
const empty = require('./empty.js')
empty.test = 123
console.log(require('./empty.js').test)
console.log('finished')
```

- Demo **require('./b.js')**
- Karma example

Вопрос

Что здесь происходит?

```
const cachedGlob = require('glob')
delete
  require.cache[require.resolve('glob')]

const originalGlob = require('glob')
cachedGlob.Glob = originalGlob.Glob
```

https://nodejs.org/api/modules.html#modules_all_together

- **require('core')** - встроенные модули
- **require('/')** или './', '../' - откуда
- **file, .js, .json, .node** - расширения файлов
- **X/index.js** - index
- **X/package.json** - "модуль"
- **node_modules** - загрузка по иерархии

- **b.js > require('./a')**
- **a.js > require('./b')**

Код первого модуля внутри второго будет *"незавершенным"*
Что будет выведено в консоль?

```
node a
```

```
// a.js
const b = require('./b')
console.log(b.b)
exports.a = 'a'
console.log('finished')
```

```
// b.js
const a = require('./a')
console.log(a.a)
exports.b = 'b'
```

Поддерживается в расширении ***.mjs**

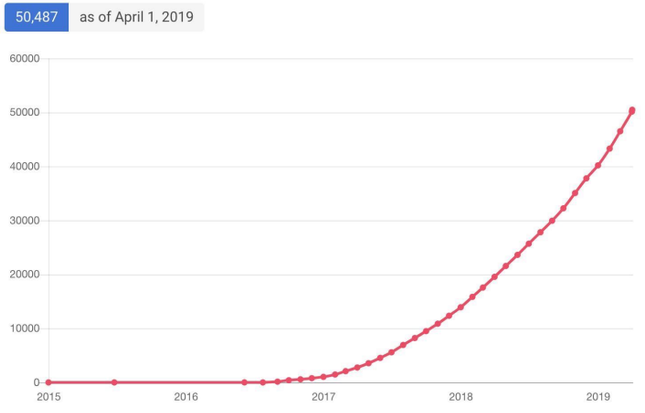
```
import * as core
  from '@uirouter/core'
// complete module import

export default 'ui.router'
// default export

export const name = 'myName'
// named exports

import('./a').then(({ a }) => {
 console.log(a)
})
// dynamic import
```

Number of ES modules on npm:



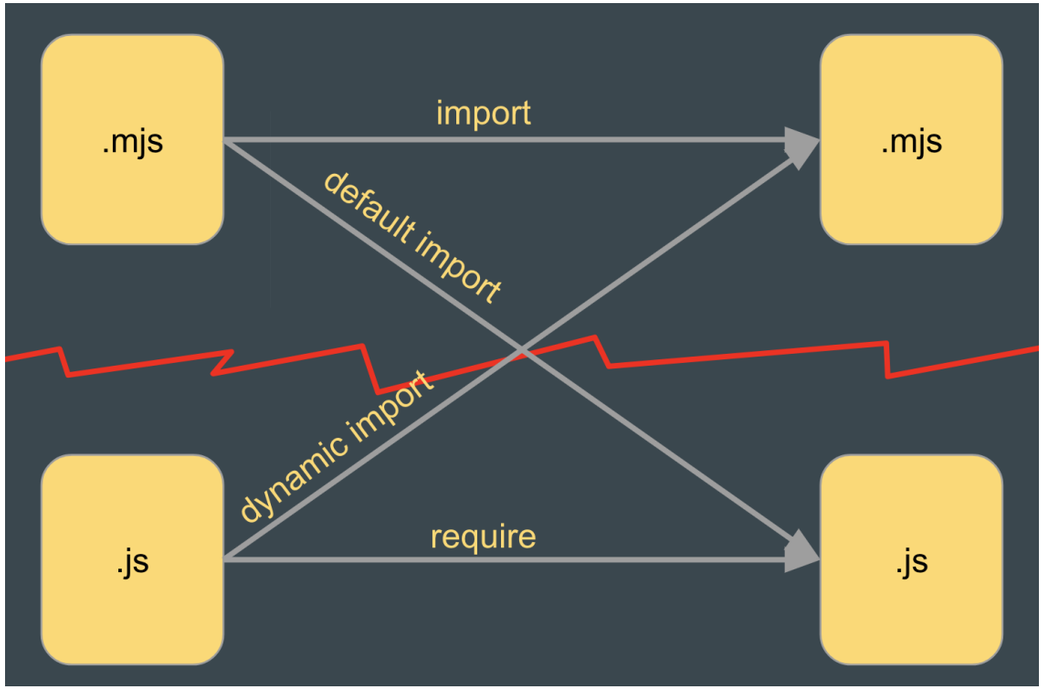
```
<script type="module" ...>
```

- Declarative
- Static declarations at the top level
- Strict mode
- Asynchronous
- Scoped
- **--experimental-modules**
- Imports are read-only views on exports

- Native ES Modules - something almost, but not quite entirely unlike CommonJS
- Gil Tayar, November 2018, @giltayar
- https://github.com/giltayar/node-esm-tea
- Node ES Modules - something almost, but not quite entirely unlike CommonJS - Gil Tayar
- https://medium.com/@giltayar/native-es-modules-in-nodejs-status-and-future-directions-part-i-ee5ea3001f71

Использовать Node@10 для стандартного импорта

- Импортировать **default export** из CJS в MJS
- Импортировать значение с помощью **dynamic import** из MJS в CJS

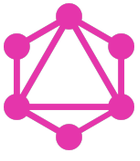


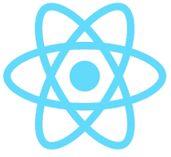
Modules Q&A

- Разобрали различие **CommonJS** и **ES Modules**
- Поняли как работают **require** и **exports** для экспорта и импорта зависимостей






express






TS



NOT THIS TIME!

Спасибо за внимание!

Пожалуйста, пройдите опрос

в личном кабинете

- Все ли темы были понятны? (да - нет)
- Легкий материал или нет? (1 просто - 10 сложно)