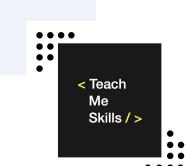
# Modules Destructuring assignment

Модули Деструктуризация

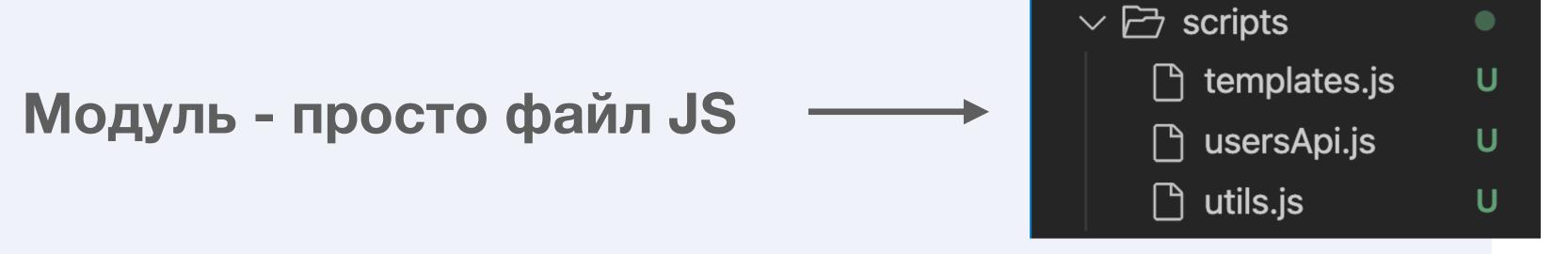


### Урок





### Module





позволяет импортировать функциональность из других модулей

отмечает переменные и функции, которые должны быть доступны вне текущего модуля



Мы можем пометить любое объявление как экспортируемое, разместив export перед ним, будь то функция, массив или переменная.

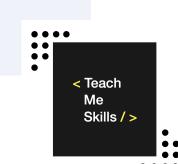
```
index.js
import { template } from './scripts/utils.js';
import { fibonacci } from './scripts/utils.js';
import { count } from './scripts/utils.js';
utils.js
export const template = () =>
export const fibonacci = [0, 1, 1]
export let count = 12
```

Либо использовать экспорт отдельно от объявления.

```
index.js
import { template, fibonacci, count } from './scripts/utils.js';
utils.js
const template = () => ``
const fibonacci = [0, 1, 1]
let count = 12
export {template, fibonacci, count}
```

### Особенности модулей

- Для запуска модулей требуется указать атрибут type='module'
- В модулях по умолчанию включен режим 'use strict'
- Своя область видимости переменных
- Код в модуле выполняется только один раз при импорте
- Объект import.meta содержит информацию о модуле
- В модуле this === undefined
- Модули загружаются отложено, как при применении атрибута defer в script
- Пути модулей обязаны быть относительными или абсолютными
- Доступен экспорт по умолчанию
- Доступен реэкспорт



### Деструктуризация

Деструктуризация (destructuring assignment) – это особый синтаксис присваивания, при котором можно присвоить массив или объект сразу нескольким переменным, разбив его на части.

Деструктуризация просто подразумевает разбивку сложной структуры на простые части. В JavaScript, таковая сложная структура обычно является объектом или массивом.



### Чем более сложную и глубокую структуру имеют наши объекты или массивы - тем сложнее нам получить к ним доступ

```
const student = {
    age: 31,
    group: 'FE-98',
    scores: {
        html: 8,
        css: 6,
        javascript: 8
    }
}

const calcAverageScore = (student) => {
    return (student.scores.html + student.scores.css + student.scores.javascript) / 3
}

calcAverageScore(student) // 7.33333333333
```

### Используя деструктуризацию мы можем выделить фрагменты наших обьектов или массивов и поместить их в переменные

## Деструктуризация объектов



#### Рассмотрим самый простой пример деструктуризации

```
const student = {
    age: 18,
    group: 'FE-98',
    scores: {
       html: 8,
        css: 6,
       javascript: 8
let { age, group, scores } = student
age -> 18
group -> 'FE-98'
scores -> {html: 8, css: 6, javascript: 8}
```

#### Значения по умолчанию

```
const student = {
    group: 'FE-98',
    scores: {
        html: 8,
        css: 6,
        javascript: 8
    }
}
let { age = 18, group, scores } = student

age    -> 18
group   -> 'FE-98'
scores   -> {html: 8, css: 6, javascript: 8}
```

#### Назначение других имен переменных

```
const student = {
    age: 18,
   group: 'FE-98',
    scores: {
       html: 8,
        css: 6,
       javascript: 8
let { age, group, scores: rating } = student
age -> 18
group -> 'FE-98'
rating -> {html: 8, css: 6, javascript: 8}
```

#### Деструктуризация вложенных объектов

```
const student = {
   age: 18,
   group: 'FE-98',
   scores: {
      html: 8,
       css: 6,
      javascript: 8
let { age, group, scores: { html, css, javascript } } = student
group -> 'FE-98'
html -> 8
css -> 6
javascript -> 8
```

### Деструктуризация массивов



#### Деструктуризация массива

```
const fruits = ['apple', 'avocado', 'banana', 'lime']
let [apple, avocado, banana, lime] = fruits

apple   -> 'apple'
avocado -> 'avocado'
banana   -> 'banana'
lime   -> 'lime'
```

#### Значения по умолчанию

```
const fruits = ['apple', 'avocado', 'banana']
let [apple, avocado, banana, lime = 'lime'] = fruits
apple   -> 'apple'
avocado -> 'avocado'
banana   -> 'banana'
lime   -> 'lime'
```

#### Пропуск элементов

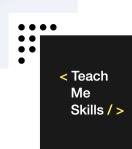
```
const fruits = ['apple', 'avocado', 'banana', 'lime']
let [ , , , lime] = fruits
lime    -> 'lime'
```



#### Обмен переменных

```
let isHappy = true;
let isFull = false;

[isFull, isHappy] = [isHappy, isFull]
isHappy -> false
isFull -> true
```



#### Деструктуризация вложенного массива

```
const fruits = [['apple', 'avocado'], ['banana', 'lime']]
let [[apple, avocado], [banana, lime]] = fruits

apple   -> 'apple'
avocado -> 'avocado'
banana   -> 'banana'
lime   -> 'lime'
```

#### Элемент rest

```
const fruits = ['apple', 'avocado', 'banana', 'lime']
let [apple, ...otherFruits] = fruits

apple          -> 'apple'
otherFruits          ->['avocado', 'banana', 'lime']
```

#### Клонирование массивов

```
const fruits = ['apple', 'avocado', 'banana', 'lime']
let [...fruitsClone] = fruits

fruitsClone -> ['apple', 'avocado', 'banana', 'lime']

fruitsClone === fruits -> false
```

#### Смешанная деструктуризация

```
const student = {
    age: 31,
    payments: [1400, 1300, 1500, 900]
}
let {payments: [firstPayment, secondPayment, thirdPayment, fourthPayment]} = student

firstPayment -> 1400
    secondPayment -> 1300
    thirdPayment -> 1500
    fourthPayment -> 900
```



### Деструктурированные параметры функций



#### Без деструктуризации

```
const student = {
   age: 31,
   name: 'Polina',
   socials: {
        vk: {
            link: 'https://xxx'
        twitter: {
            link: 'https://xxx'
const printUserSocials = (student) => {
    console.log(`Twitter: ${student.socials.twitter.link} VK:
${student.socials.vk.link}`)
printUserSocials(student)
```

#### С деструктуризацией

```
const student = {
   age: 31,
   name: 'Polina',
   socials: {
        vk: {
            link: 'https://xxx'
        twitter: {
            link: 'https://xxx'
const printUserSocials = ({socials: {vk, twitter}}) => {
   console.log(`Twitter: ${twitter.link} VK: ${vk.link}`)
printUserSocials(student)
```