



Microsoft Partner
Silver Learning



JavaScript. Расширенные ВОЗМОЖНОСТИ

Новые возможности ES6



ITVVDN
IT VIDEO DEVELOPERS NETWORK

JavaScript. Расширенные возможности

Автор курса



Патёха Сергей



MISTER SERGII PATOKHA

Has successfully completed the requirements to be recognized as a Microsoft Certified Professional.

Date of achievement: 06/07/2018
Certification number: GB48-0117

Satya Nadella
Chief Executive Officer



MCID: 16014173

JavaScript. Расширенные возможности

После урока обязательно



Повторите этот урок в видео формате на [ITVDN.com](http://itvdn.com)



Проверьте как Вы усвоили данный материал на [TestProvider.com](http://testprovider.com)

Новые возможности ES6

JavaScript. Расширенные возможности

Map

Map - (отображение, словарь) представляет объектно-подобную структуру данных, где каждый элемент имеет ключ и значение. В отличие от объектов, в которых ключами могут быть только строки, в Map ключом может быть произвольное значение. Аргументом Map должен быть итерируемый (перебираемый) объект.

```
let myMap = new Map();
```

Методы работы:

Map.set(item) – добавление элементов. Возвращает объект Map.

Map.get(item) – получение элементов по ключу.

Map.has(item) – проверка наличия элементов.

Map.delete(item) – удаление элементов по ключу.

Map.clear() – очистка Map от всех элементов.

JavaScript. Расширенные возможности

Set

Set - (множество) представляют массиво-подобную структуру данных, которая может хранить только уникальные значения. Каждое значение Set может встречаться лишь один раз.

```
let mySet = new Set(1,2,3,4,2,3,5);  
console.log(mySet); // [1,2,3,4,5]
```

Методы работы:

Set.set(item) – добавление элементов. Возвращает объект Set.

Set.get(item) – получение элементов по ключу.

Set.has(item) – проверка наличия элементов.

Set.delete(item) – удаление элементов по ключу.

Set.clear() – очистка Map от всех элементов.

JavaScript. Расширенные возможности

Деструктуризация

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

```
let [var1, var2, var3] = ['Hello', 'World', 'ITVDN'];
```

Spread (оператор разворота) – позволяет разворачивать элементы массива в качестве аргументов функции или в аргументы другого массива.

Rest оператор – позволяет определить остальные параметры в функции. Должен быть записан ВСЕГДА последним аргументом.

JavaScript. Расширенные возможности

Symbols

Symbol - примитивный тип данных. Служит для создания уникальных идентификаторов и, также, создания приватных защищенных полей у объекта. Символ в объекте – не итерируется.

```
let symbol = Symbol('Ukraine');
```

Имя символа указывается только для описания и удобства определения.
Метод for – позволяет символ создать в глобальном реестре.

```
let symbolName = Symbol.for('Ivan');
```

Метод keyFor - чтение символа из глобального реестра. Работает только для символов, созданных в глобальном реестре.

```
console.log(Symbol.keyFor(symbol)); // undefined  
console.log(Symbol.keyFor(symbolName)); // Symbol(Ivan)
```


JavaScript. Расширенные возможности

Iterators

Итератор - шаблон проектирования, где источник элементов скрыт от пользователя.

Итераторы применяются для организации последовательного доступа к элементам коллекции - массивам, объектам Set и Map, строкам, коллекциям DOM дерева.

```
Collection[Symbol.iterator]()
```

Итерируемый объект – объект, содержание которого можно перебрать по очереди.

```
for(let item of Collection) {  
  console.log(item);  
}
```

У итерируемого объекта есть специальный метод: `[Symbol.iterator]()`, который возвращает объект: `Iterator`. `Iterator` содержит метод `next()` – возвращающий `IteratorResult` {}.

```
IteratorResult {  
  done: boolean,  
  Value: any }
```

JavaScript. Расширенные возможности

Generators

Генераторы представляют особый тип функции, которые используются для генерации значений. Генераторы могут приостанавливать своё выполнение, возвращать промежуточный результат и далее возобновлять его позже, в произвольный момент времени.

```
function* gen() {  
  Console.log('Start');  
  yield 'Secret code';  
  Console.log('End');}
```

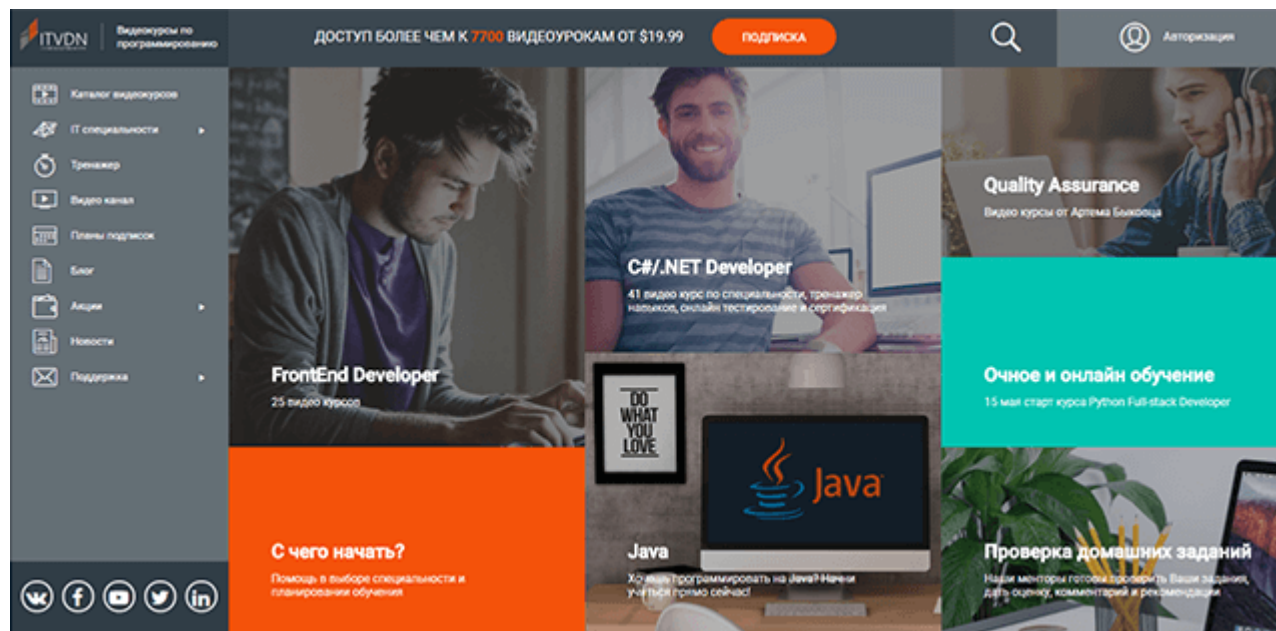
При вызове функции - создается объект итератор, у которого существует метод `next()`. `Yield` - производит и отдает информацию, а также отдает контроль над функцией, позволяя ее приостановить. `Yield` ставит выполнение функции на паузу - пока не произойдет вызов функции `next()`.

`Iterator.throw()` – генерация ошибки, окончание итерации.

`Iterator.return()` – выход из итерации генератора. (`done = true`).

Смотрите наши уроки в видео формате

ITVDN.com



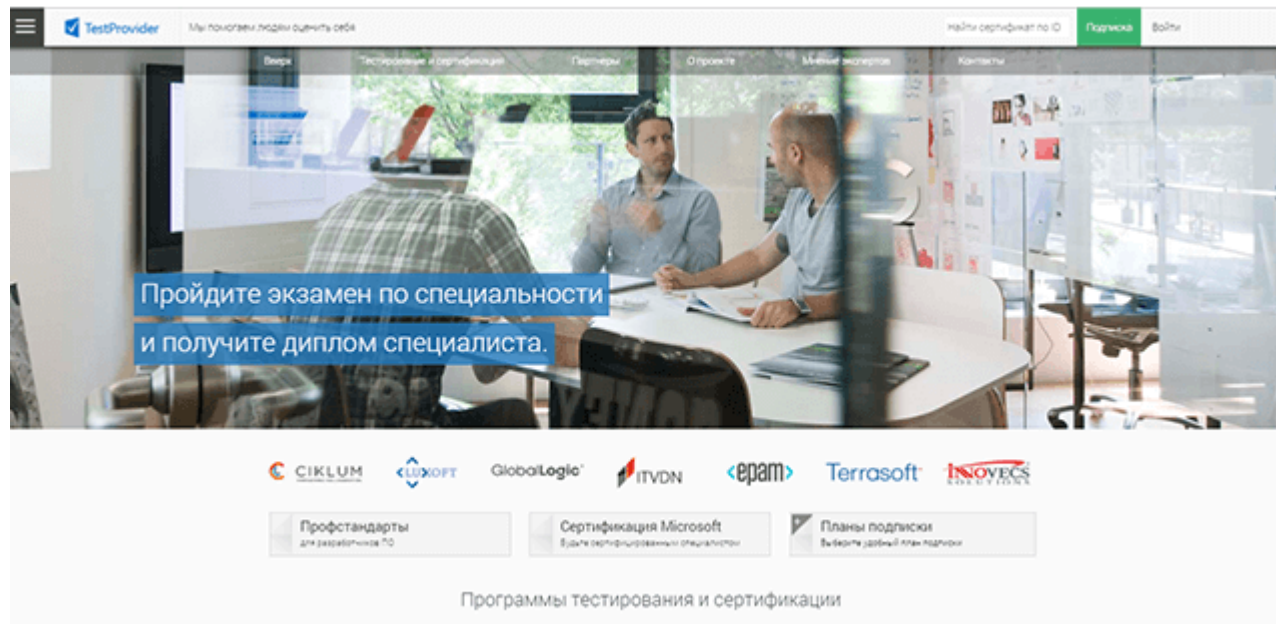
Посмотрите этот урок в видео формате на образовательном портале ITVDN.com для закрепления пройденного материала.

Курсы записаны сертифицированными тренерами, которые работают в учебном центре CyberBionic Systematics и другими высококвалифицированными разработчиками.



Проверка знаний

TestProvider.com



TestProvider – это online сервис проверки знаний по информационным технологиям. С его помощью Вы можете оценить Ваш уровень и выявить слабые места. Он будет полезен как в процессе изучения технологии, так и для общей оценки знаний IT специалиста.

После каждого урока проходите тестирование для проверки знаний на [TestProvider.com](https://testprovider.com)

Успешное прохождение финального тестирования позволит Вам получить соответствующий Сертификат.



Q&A

Информационный видеосервис для разработчиков программного обеспечения

