Гродзинская А. ДКИ-303

Форум

Стандарт EcmaScript 5:

<http://www.ecma-international.org/ecma-262/5.1/>

Изменения в EcmaScript 6:

<http://wiki.ecmascript.org/doku.php?id=harmony:specification_drafts>

1. **Цикл for-of**

ES5: «let arr = [ "blue", "green" ];

arr.notAnIndex = 123;

Array.prototype.protoProp = 456;

**for**(var x **in** arr) { console.log(x); // Напечатает blue, green, notAnIndex, protoProp

} »

ES6: « **for**(var x **of** arr) {

console.log(x); // Напечатает только blue, green

} »

Комментарий: в EcmaScript 5 отсутствует цикл for-of и используется цикл for-in , который итерирует по всем полям объекта (включая наследованных), т.е. в результате получаем не совсем ожидаемые вещи, как в нашем примере: blue, green, notAnIndex, protoProp. Решает данную проблему появившийся в EcmaScript 6 цикл for-of, в результате которого получаем то, что нужно: blue, green.

1. **Arrow-функции**

ES5: « let squares = [ 1, 2, 3 ].map(function (x) { return x \* x });»

ES6: «let squares = [ 1, 2, 3 ].map(x => x \* x);»

Комментарий: в EcmaScript 6 появились так называемые «стрелочные функции», которые отличаются от традиционных рядом параметров:

· Лексическое связывание. Значения специальных переменных *this, super иarguments* определяются не тем, как стрелочные функции были вызваны, а тем, как они были созданы.  
· Неизменяемые *this, super и arguments*. Значения этих переменных внутри стрелочных функций остаются неизменными на протяжении всего жизненного цикла функции.  
· Стрелочные функции не могут быть использованы как конструктор и кидают ошибку при использовании с оператором new.  
· Недоступность «собственного» значения переменной *arguments*.

Стрелочные функции выгодны в использовании по нескольким причинам:

* Более удобная форма записи
* Ранее могла возникнуть ситуация, при которой можно было потерять нужное значение this, что приводило к ошибкам. Использование же стрелочных функций убирает эту, своего рода, путаницу.
* JS-движки смогут легко оптимизировать выполнение стрелочных функций за счет этих ограничений (в противоположность традиционным функциям, которые могут быть использованы в качестве конструктора и которые свободны для модификации специальных переменных)