Почему функция map не работает с некоторыми массивами в JavaScript и что с этим делать

Первый шаг к рабочему и эффективному коду — разобраться в механизмах, которые лежат в основе языка. Например, некоторые функции в JavaScript не всегда ведут себя так, как того ожидает программист. Рассмотрим один из таких примеров.

Сценарий

Для более наглядной демонстрации предположим, что вам нужно сгенерировать массив чисел от 0 до 99. Как можно это сделать? Например, так:

```
const arr = [];
for (let i = 0; i < 100; i++) {
  arr[i] = i;
}
```

Возможно, цикл for в JavaScript заставляет вас чувствовать себя неуютно. На самом деле, многие программисты годами не используют for-циклы благодаря функциям forEach(), map(), filter(), и reduce(). Декларативное функциональное программирование рулит!

Если вы еще не поддались массовому помешательству на почве функционального программирования, решение выше кажется вам абсолютно нормальным. Технически так и есть, но распробовав магию функций высшего порядка, вы, скорее всего, подумаете: «Должен быть способ получше».

Тогда первая реакция на проблему может быть такой: «Знаю! Я создам пустой массив длиной 100 и присвою каждому элементу значение его индекса с помощью map()!». JavaScript позволяет создать пустой массив длиной n с помощью функции-конструктора:

```
const arr = Array(100);
```

Великолепно, не так ли? У нас есть массив длиной 100, так что нужно просто присвоить значение индекса каждому элементу:

```
const arr = Array(100).map((_, i) => i);
console.log(arr[0] === undefined); // true
```

Что за дела? Первым элементом массива должен был быть 0, но на самом деле это undefined.

Объяснение

Нужно обозначить один важный технический момент, чтобы объяснить, как так вышло. По своей сути массивы в JavaScript — объекты с числами в роли индексов.

Например:

```
['a', 'b', 'c']
```

эквивалентно объекту

```
{
  0: 'a',
  1: 'b',
  2: 'c',
  length: 3
}
```

Пытаясь обратиться к элементу с индексом 0, вы на самом деле обращаетесь к параметру объекта с ключом 0. Это важно, потому что учитывая то, что массив — это объект, и то, как выполняются функции высшего порядка, мы получаем вполне понятную проблему. Создавая массив с помощью функции-конструктора, вы получаете объект-массив, у которого параметр length равен числу, которое вы задали, но больше в объекте ничего нет. В нем нет индексов.

```
{
  // нет индексов!
  length: 100
}
```

Вы получаете undefined, пытаясь обратиться к элементу массива с индексом 0, но это не потому, что в элементе с индексом 0 хранится undefined, а потому, что JavaScript всегда возвращает undefined при попытке обратиться к несуществующему элементу массива. Так устроено, что функции высшего порядка — map(), reduce(), filter() и forEach() — проходятся по индексам от 0 до значения length, но callback-функция вызывается только если в объекте существуют индексы. Это объясняет, почему функция ничего не возвращает, и почему ничего не происходит, когда мы используем функцию map() на массив, в котором нет индексов.

Решение

Теперь легко сделать вывод, что нам нужен массив, чья внутренняя структура содержит индексы для каждого элемента от 0 до length. Лучший способ сделать это — деструктурировать массив в пустой массив.

```
const arr = [...Array(100)].map((_, i) => i);
console.log(arr[0]); // 0
```

Деструктуризация массива в пустой массив позволяет получить массив, содержащий undefined в каждом элементе.

```
{
  0: undefined,
  1: undefined,
  2: undefined,
  ...
  99: undefined,
  length: 100
}
```

Так происходит потому, что оператор деструктуризации проще, чем функция map(). Он просто проходит по массиву (или любому итерируемому объекту) от 0 до length и создает во внешнем массиве новый индекс, содержащий значение, полученное из элемента деструктурированного массива с тем же индексом. JavaScript возвращает undefined при обращении к любому элементу деструктурированного массива (так происходит, потому что в нем отсутствуют элементы и индексы), поэтому мы получаем новый массив, заполненный индексами, и потому доступный для функции map() (а также reduce(),filter() и forEach()).

Заключение

Таким образом, понимая внутреннее представление массивов в JavaScript, можно создавать массивы любой длины, с любым нужным значением элементов.

Материал взят из источника – «Почему функция map не работает с некоторыми массивами в JavaScript и что с этим делать»: https://tproger.ru/translations/why-js-map-doesnt-work/