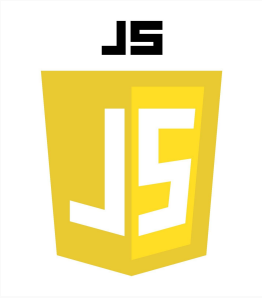
Введение
в Javascript

Общая информация о JavaScript и EcmaScript

EcmaScript - стандарт, разработанный компанией Ecma International.

ES6, ES2015 - шестая редакция стандарта EcmaScript. Самая популярная и актуальная версия.



JavaScript - язык программирования, основанный на основе спецификации EcmaScript

Полезные ресурсы по JavaScript

- https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide - сайт MDN
- https://learn.javascript.ru/ - онлайн-учебник по JavaScript
- https://ru.stackoverflow.com/ - форум с вопросами и ответами

Типы данных и переменные

Типы данных

Существует 8 встроенных типов данных. Все из них, кроме object, являются примитивными
Для определения типа данных используется **typeof**

- null
- undefined
- boolean (true, false)
- number (15, 129, 14.34)
- string (“Javascript”, ‘HTML’)
- symbol
- object
- bigInt

Специальные числовые значения

NaN (Not a Number) - “не число”, как правило возникает вследствие ошибочных операций с числами

```
1 * undefined;     // => NaN

let n = Number('abc');      // => NaN
```

Infinity - значение, обозначающее бесконечность. Может получиться в результате деления числа на ноль или если результат вычислений выходит за допустимый диапазон чисел JavaScript

Преобразование типов данных

- Boolean(), Number(), String()
- parseInt(), parseFloat() - преобразует строки в числа
- toString() - преобразует объекты в строки
- Унарный плюс

```
+”123” => 123

+”aa123” => NaN
```

Переменные

var - объявление локальной или глобальной переменной

let - объявление локальной переменной (область видимости ограничивается блоком, где переменная объявлена)

const - объявление константы. Значение константы должно быть задано при объявлении и не может быть изменено

Имена переменных

- Имя переменной должно содержать только буквы, цифры или символы $ и _.
- Первый символ не должен быть цифрой.
- Имя переменной не должно совпадать с зарезервированным ключевым словом (class, var, for, is, function и т.д.)

Массивы

Массив - структура данных для хранения упорядоченных коллекций элементов

Объявление нового массива:

- `let arr = new Array();`
- `let arr = [];`
- `let numArr = [1, 2, 3, 4, 5];`

Обращение к элементам массива:

```
numArr[0] // => 1
```

Узнать кол-во элементов в массиве:

```
numArr.length // => 5
```

Методы работы с массивами

array.push(el) - добавляет элемент в конец массива

array.unshift(el) - добавляет элемент в начало массива

array.pop() - удаляет элемент из конца массива

array.shift() - удаляет элемент из начала массива

Методы работы с массивами

- **array.splice(pos, itemsNum)** - удаляет определенное количество элементов (itemsNum), начиная с индекса pos
- **array.indexOf(el)** - возвращает индекс элемента. Если элемент отсутствует в массиве, возвращает -1
- **array.find(el)** - определяет наличие элемента в массиве. Возвращает true/false
- **array.join(separator)** - объединяет элементы массива в строку, разделяя их заданным разделителем (separator)
- **array.reverse()** - переворачивает массив. Первый элемент становится последним, последний - первым

Перебор элементов массива

С помощью циклов

```
for (let i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}
for (let element of array) {
  console.log(element);
}
```

Специальные методы перебора

```
arr.forEach(function (value, index) { … } );
arr.map(function (value, index) { … } );
```

Циклы

Циклы while, do while

Проверка условия в начале

```
let i = 0;
while (i < 11) {
  console.log( i );
  i++;
}
```

Проверка условия в конце

```
let i = 0;
do {
  console.log( i );
  i++;
} while (i < 11);
```

Цикл for

```
for (let i = 0; i < 11; i++) {
  console.log(i);
}
```

- **Начало** (`let i = 0`) Выполняется один раз при входе в цикл
- **Условие** (`i < 11`) Проверяется перед каждой итерацией цикла. Если оно будет равно false, цикл остановится.
- **Шаг** (`i++`) Выполняется на каждой итерации после прохождения тела цикла.
- **Тело** (`console.log(i)`) Выполняется снова и снова, пока условие вычисляется в true.

Прерывание цикла

break - завершает выполнение цикла в любой момент

continue - завершает текущую итерацию цикла и переходит к следующей

Структуры Map и Set

Map

Map - структура данных, предназначенная для реализации простого ассоциативного массива. В JavaScript реализована специальным объектом Map.

Ассоциативный массив - массив, у которого в качестве ключей используются любые типы данных.

Ключи у Map могут быть любым типом данных, но каждый ключ должен быть уникален.

Методы и свойства Map

- **let map = new Map()** – создание пустого экземпляра Map
- **map.set(key, value)** – записывает по ключу key значение value.
- **map.get(key)** – возвращает значение по ключу или undefined, если значение с ключом key отсутствует.
- **map.has(key)** – определяет, есть ли в коллекции элемент с ключом key.
- **map.delete(key)** – удаляет элемент по ключу key.
- **map.size** – возвращает размер коллекции

Перебор коллекции Map

Для перебора Map существует несколько методов

- **map.keys()** – возвращает итерируемый объект по ключам,
- **map.values()** – возвращает итерируемый объект по значениям,
- **map.entries()** – возвращает итерируемый объект по парам вида [ключ, значение]
- **map.forEach(function(value, key, map) { ... })** - аналогичный метод, как у массивов

Пример:

```
for (let key of map.keys()) { … }

for (let value of map.values()) { … }

for (let item of map) { // то же самое, что и map.entries()
  ...
}
```

Set

Set - структура данных, которая реализует коллекцию - список уникальных элементов в порядке их добавления. В JavaScript реализована специальным объектом Set.

Ключи у Set отсутствуют, или, другими словами ключами являются сами значения.

Методы и свойства Set

- **let set = new Set()** – новый экземпляр Set.
- **set.add(value)** – добавляет значение (если оно уже есть, то ничего не происходит), возвращает тот же объект set.
- **set.delete(value)** – удаляет значение.
- **set.has(value)** – определяет, присутствует ли значение в множестве.
- **set.clear()** – удаляет все имеющиеся значения.
- **set.size** – возвращает количество элементов в множестве.

Set и Array

Set во многом похож на массив, но есть важные отличия:

- Set.has работает быстрее чем Array.indexOf
- Можно удалять элементы по значению (а не по индексу как массивах)
- Поддерживается уникальность значений.

Можно создать Set из массива и наоборот. При создании Set'а из массива все повторяющиеся элементы в массиве будут утеряны.

```
let arrayFromSet = Array.from(mySet);

let setFromArray = new Set([1,2,3,4]);
```