Типове на данни в JavaScript

Виктор Андриевски
Фросина Мулачка
Джем Салимов
Борис Димитров

ЦЕЛ

- Примитивни типове и вътрешно представяне
- Не-примитивни типове и вътрешно представяне
- Итератори
- Генератори

JavaScript

припомняне

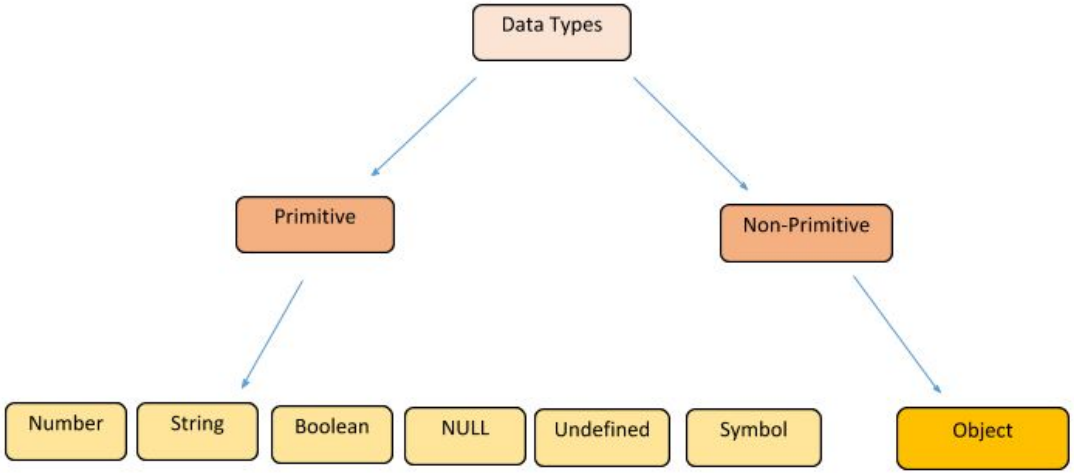
- динамично типизиран
- слабо типизиран

- **динамично типизиран**

```
let foo = 42; // foo е число
foo = "bar"; // foo е стринг
foo = true; // foo е boolean
```

- **слабо типизиран**

```
const foo = 42; // foo е число
const result = foo + "1"; // foo: число -> стринг
console.log(result); // 421
```


Data Types
Primitive
Non-Primitive
Number
String
Boolean
NULL
Undefined
Symbol
Object

Number в JavaScript

- числови данни, **цели и дробни**
- вътрешно представяне: числа с плаваща запетая с двойна прецизност от 64 бита
- диапазон за положителни числа: 2^-1074 до 2^1024
- диапазон за отрицателни числа: -2^-1074 до -2^1024
- Infinity, -Infinity
- NaN
- цели числа в интервала от -(2^53 − 1) до 2^53 − 1 *(MIN_SAFE_INTEGER, MAX_SAFE_INTEGER)*

```
const result1 = Math.sqrt(-1);
console.log(result1); // NaN

const result2 = 0 / 0;
console.log(result2); // NaN

// Infinity
const result3 = 1 / 0;
console.log(result3); // Infinity

// -Infinity
const result4 = -1 / 0;
console.log(result4); // -Infinity
```

BigInt в JavaScript

- цели числа с произволна големина
- по-голяма точност от типа number
- операции извън границите на Number.MAX_SAFE_INTEGER
- низ от байтове

String в JavaScript

- текстови данни - последователност от 16-битови стойности
- Unicode

  пр. **`let str = "Hello";`**,
  H (U+0068), **e** (U+0065), **l** (U+006C) x2, **o** (U+006F)

- представяне с единични, двойни или обратни кавички
- Wrapper object предоставя различни функционалности JavaScript String Methods
- създаване на обект с new String()
- immutable (неизменяеми)

```
const str1 = 'Hello'; // Низ с единични кавички
const str2 = "World"; // Низ с двойни кавички

// С конструктора new String()
const str3 = new String('Hello'); // Създава обект от тип String

// С глобалната функция String()
const str4 = String(123); // Преобразуване на числото в низ (примитивен низ)

// Низови литерали с обратни кавички
const str5 = `JavaScript`; // Низ с обратни кавички

// Интерполация на променлива в низ
const name = 'Alice';
const greeting = `Hello, ${name}!`; // Интерполация на променлива в низ
```

Symbol в JavaScript

- уникална и неизменяема примитивна стойност
- цел - да създадат уникални ключове на свойства на обекти

```
// two symbols with the same description

const value1 = Symbol('hello');
const value2 = Symbol('hello');

console.log(value1 === value2); // false
```

Boolean в JavaScript

- оценка на логически условия
- две стойности - **true** и **false**
- един бит в паметта - 1 за true и 0 за false

Undefined в JavaScript

- липса на стойност
- всички променливи, които не са инициализирани, имат стойност `undefined`
- функция връща undefined, ако не е върната стойност
- достъп до несъществуващо свойство на обект - undefined
- специфична, специална резервирана стойност (маркер)

```
let x;
console.log(x); // ще изведе: undefined

function testFunc(param) {
    console.log(param);
}

testFunc(); // ще изведе: undefined
```

Null в JavaScript

```
let x;
console.log(x); // ще изведе: undefined

function testFunc(param) {
    console.log(param);
}

testFunc(); // ще изведе: undefined
```

```
let y = null;
console.log(y); // ще изведе: null
```

```
console.log(undefined == null); // true
console.log(undefined === null); // false
```

Обекти в JavaScript

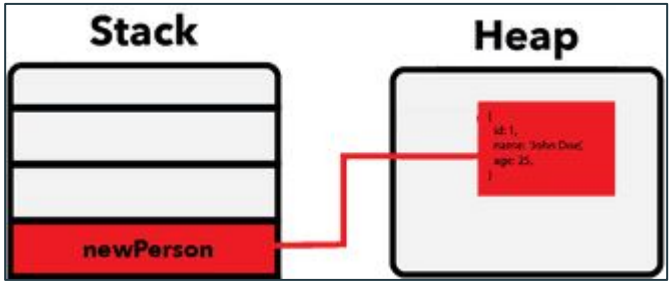
- колекция от свойства
- свойства във формата ключ-стойност
- указатели към обектите се съхраняват в стека, а атрибутите им в хийпа
- плитко (**shallow**) и дълбоко (**deep**) копие

```
const person = {
  id: 1,
  name: "John Doe",
  age: 25
}

const personCopy = person
console.log(person == personCopy) // true

personCopy.id = 5
console.log(person.id) // 5

 // deep copy
const newPerson = JSON.parse(JSON.stringify(person));
```



Масиви в JavaScript

- обекти от тип Array
- динамични
- позволяват смесени типове данни
- не са асоциативни
- индексацията на масивите започва от нула
- вградени операции за копиране на масиви
- length - броя на елементите в масива

Итератори

- next()
- връща { value: <number, string, etc.>, done : false}
- връща {value: <number, string, etc>, done: true}
- Някои типове биват build-in iterables, които по подразбиране са итеративни
- Не-итеративни: обектите

```
function createIterator(items) {

    var i = 0;

    return {
        next: function() {

            var done = (i >= items.length);
            var value = !done ? items[i++] : undefined;

            return {
                done: done,
                value: value
            };

        }
    };
}

var iterator = createIterator([1, 2, 3]);
```

JS

```
function makeRangeIterator(start = 0, end = Infinity, step = 1) {
  let nextIndex = start;
  let iterationCount = 0;

  const rangeIterator = {
    next() {
      let result;
      if (nextIndex < end) {
        result = { value: nextIndex, done: false };
        nextIndex += step;
        iterationCount++;
        return result;
      }
      return { value: iterationCount, done: true };
    },
  };
  return rangeIterator;
}
```

JS

```
const iter = makeRangeIterator(1, 10, 2);

let result = iter.next();
while (!result.done) {
  console.log(result.value); // 1 3 5 7 9
  result = iter.next();
}

console.log("Iterated over sequence of size:", result.value); // [5 numbers returned,
that took interval in between: 0 to 10]
```

- Array итератор
- String итератор

```
const array1 = ['a', 'b', 'c'];
const iterator = array1.values();

for (const value of iterator) {
  console.log(value);
}
```

```
const uint8 = new Uint8Array([10, 20, 30, 40, 50]);
const eArr = uint8.entries();

eArr.next();
eArr.next();

console.log(eArr.next().value);
console.log(eArr.next().value);
console.log(eArr.next().value);
console.log(eArr.next().value);
```

```
const str = "The quick red fox jumped over the lazy dog's back.";

const iterator = str[Symbol.iterator]();
let theChar = iterator.next();

while (!theChar.done && theChar.value !== ' ') {
  console.log(theChar.value);
  theChar = iterator.next();
}
```

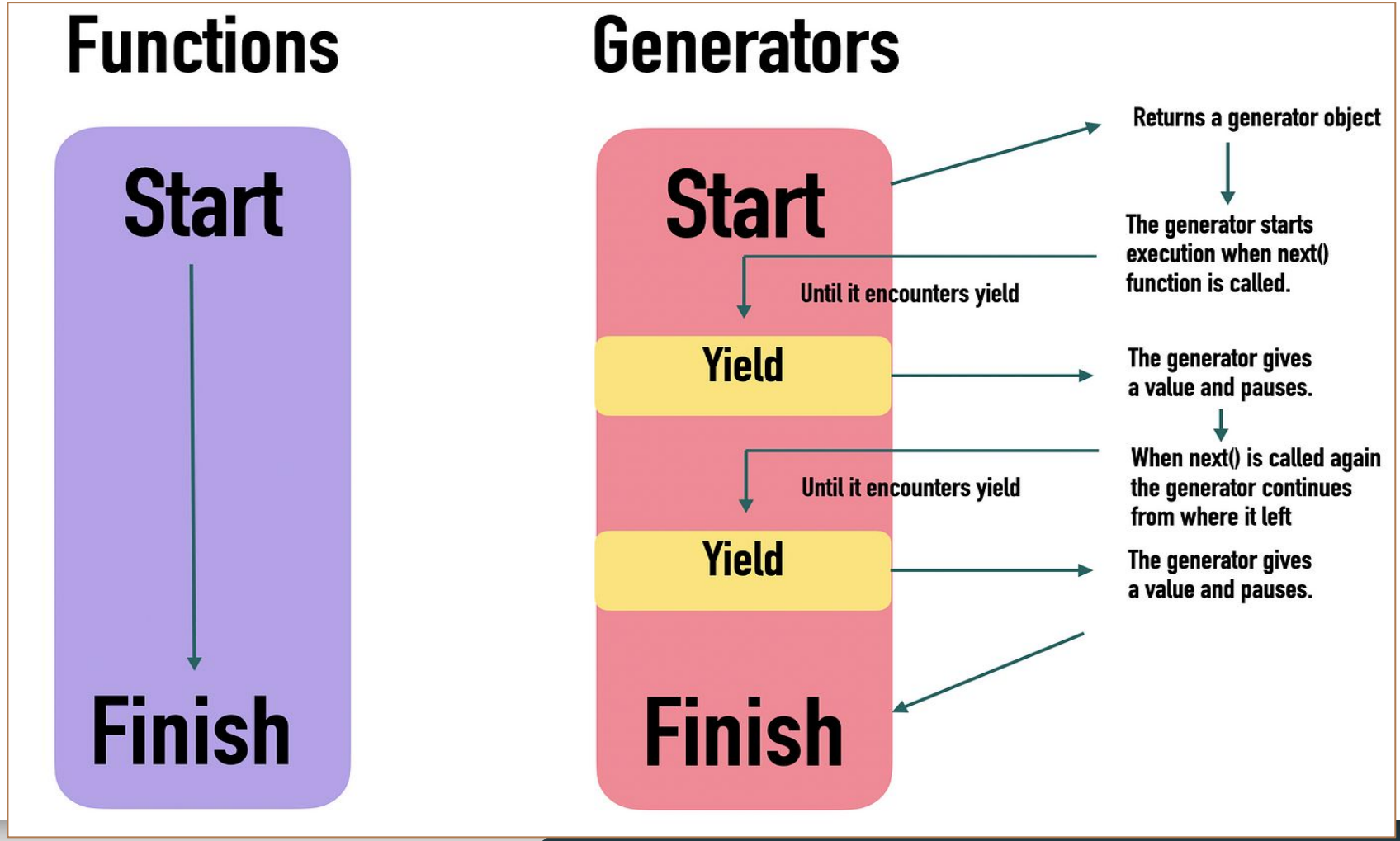
Генератори

- Специален тип **итератори**
- Създават се от генераторни функции (дефинирани с ***function****)
- Всяко извикване, изпълняват генераторната ф-я до достигане на ***yield***

```
function* makeRangeIterator(start = 0, end = Infinity, step = 1) {
  let iterationCount = 0;
  for (let i = start; i < end; i += step) {
    iterationCount++;
    yield i;
  }
  return iterationCount;
}
```

```
const iter = makeRangeIterator(1, 10, 2);

let result = iter.next();
while (!result.done) {
  console.log(result.value); // 1 3 5 7 9
  result = iter.next();
}
```


Functions
Start
Finish
Generators
Start
Yield
Yield
Finish
Returns a generator object
The generator starts execution when next() function is called.
Until it encounters yield
The generator gives a value and pauses.
When next() is called again the generator continues from where it left
Until it encounters yield
The generator gives a value and pauses.

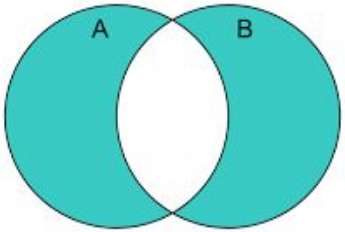
Map/Set в JavaScript

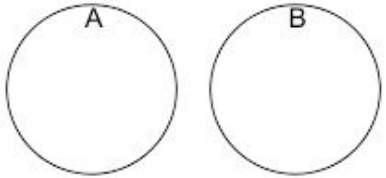
Map

- двойки ключ-стойност
- всеки ключ е уникален
- бързо търсене на стойности по ключ
- Map vs Object

Set

- уникални елементи без дубликати
- изпълнение на операции между множества





WeakMap/WeakSet в JavaScript

- система за почистване на паметта (**garbage collector**)
- В WeakMap и WeakSet обектите, се считат за слаби (“**weak**”)

WeakMap

- ключове - обекти
- стойностите могат да бъдат както обекти, така и примитивни стойности

WeakSet

- съдържа само обекти
- всеки обект може да се съдържа само веднъж

БЛАГОДАРИМ ЗА ВНИМАНИЕТО! :)