

Творческая работа

Обзор алгоритмов

Все алгоритмы будут сравниваться по времени “real” стандартного вывода времени исполнения программы командой time в UNIX-системах.

Все алгоритмы будут запускаться на одной и той же машине, в одинаковых условиях: параллельная нагрузка на ЭВМ изменяться не будет.

Время исполнения всех алгоритмы будут считаться с учетом времени, потраченного на стандартный вывод в консоль результатов работы алгоритма.

Для каждого алгоритма будет сделано 3 замера времени выполнения.

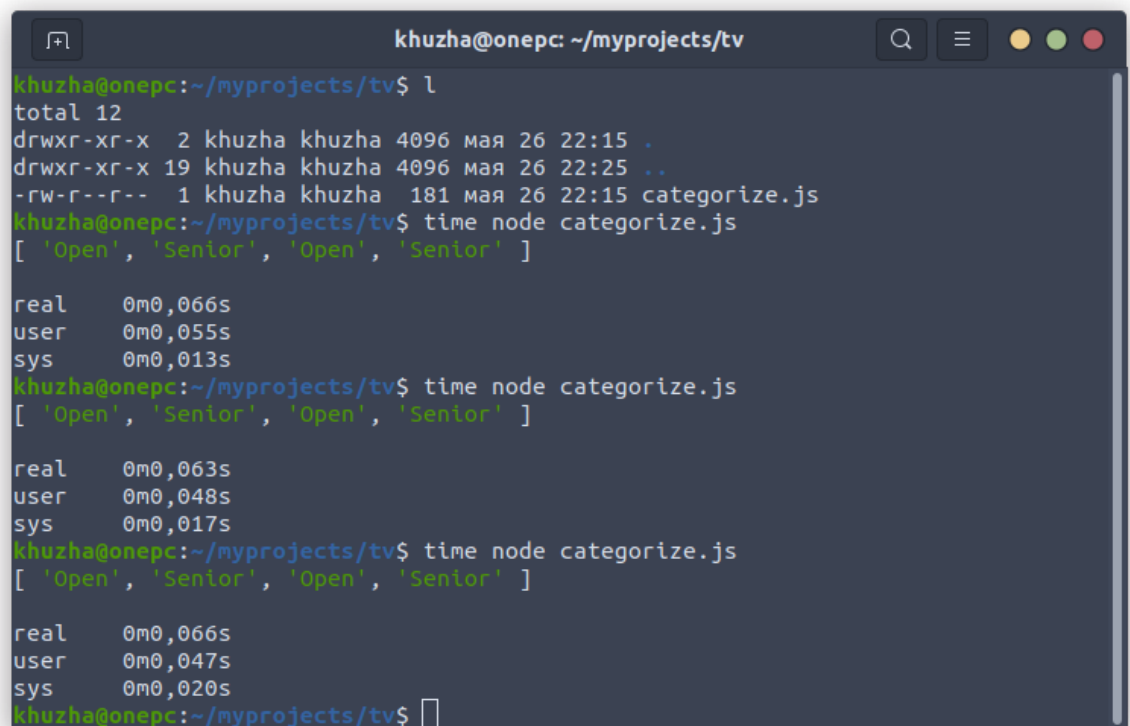
1. Написать функцию, которая на вход принимает массив массивов (двумерный массив) числовых значений в виде аргумента, создает новый массив (одномерный), куда запишет в конечный массив слово “Senior” или “Open”, сравнивая элементы массивов второго уровня из аргумента. Если первый элемент больше либо равно 55 и второе строго больше 7, запишется “Senior”, иначе запишется “Open”.

Решение №1:

Используем метод `.map()` класса `Array.prototype`

```
1. function openOrSenior(data) {  
2.   return data.map(person => (person[0] >= 55 && person[1] > 7) ?  
   'Senior' : 'Open')  
3. }  
4.  
5. console.log(openOrSenior([[45, 12],[55,21],[19, -2],[104,  
   20]]))
```

Код функции занимает 3 строки и выполняется в среднем за 65 микросекунд:



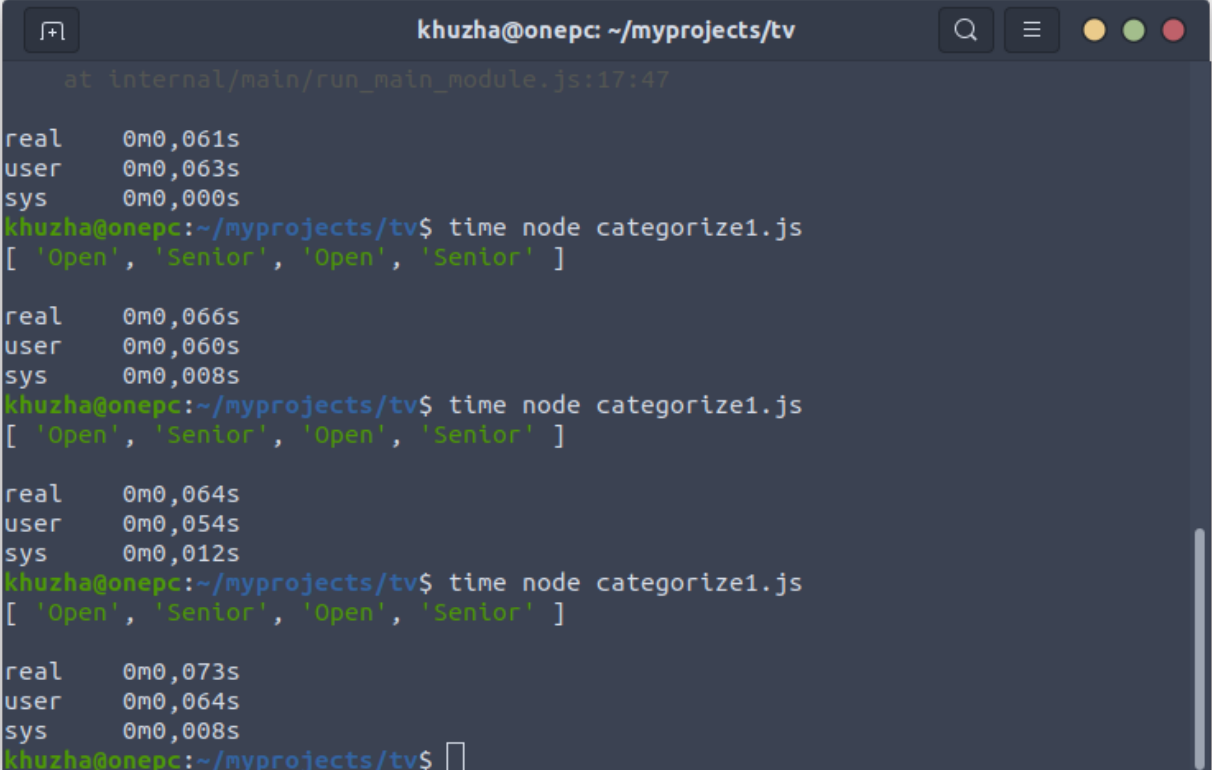
```
khuzha@onepc: ~/myprojects/tv  
khuzha@onepc:~/myprojects/tv$ l  
total 12  
drwxr-xr-x  2 khuzha khuzha 4096 мая 26 22:15 .  
drwxr-xr-x 19 khuzha khuzha 4096 мая 26 22:25 ..  
-rw-r--r--  1 khuzha khuzha  181 мая 26 22:15 categorize.js  
khuzha@onepc:~/myprojects/tv$ time node categorize.js  
[ 'Open', 'Senior', 'Open', 'Senior' ]  
  
real    0m0,066s  
user    0m0,055s  
sys     0m0,013s  
khuzha@onepc:~/myprojects/tv$ time node categorize.js  
[ 'Open', 'Senior', 'Open', 'Senior' ]  
  
real    0m0,063s  
user    0m0,048s  
sys     0m0,017s  
khuzha@onepc:~/myprojects/tv$ time node categorize.js  
[ 'Open', 'Senior', 'Open', 'Senior' ]  
  
real    0m0,066s  
user    0m0,047s  
sys     0m0,020s  
khuzha@onepc:~/myprojects/tv$
```

Решение №2:

Используем цикл `for`, конструкцию `key ... of ...` и явно создаем конечный массив `res`, в него результаты записываем с использованием метода `.push()` класса `Array.prototype`

```
1. function openOrSenior(data) {
2.   const res = []
3.
4.   for (const person of data) {
5.     if (person[0] >= 55 && person[1] > 7) {
6.       res.push('Senior')
7.     } else {
8.       res.push('Open')
9.     }
10.  }
11.  return res
12. }
13.
14. console.log(openOrSenior([[45, 12],[55,21],[19, -2],[104,
    20]]))
```

Код функции занимает 12 строк и выполняется в среднем за 68 миллисекунд:



```
khuzha@onepc: ~/myprojects/tv
at internal/main/run_main_module.js:17:47

real    0m0,061s
user    0m0,063s
sys     0m0,000s
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

real    0m0,066s
user    0m0,060s
sys     0m0,008s
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

real    0m0,064s
user    0m0,054s
sys     0m0,012s
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

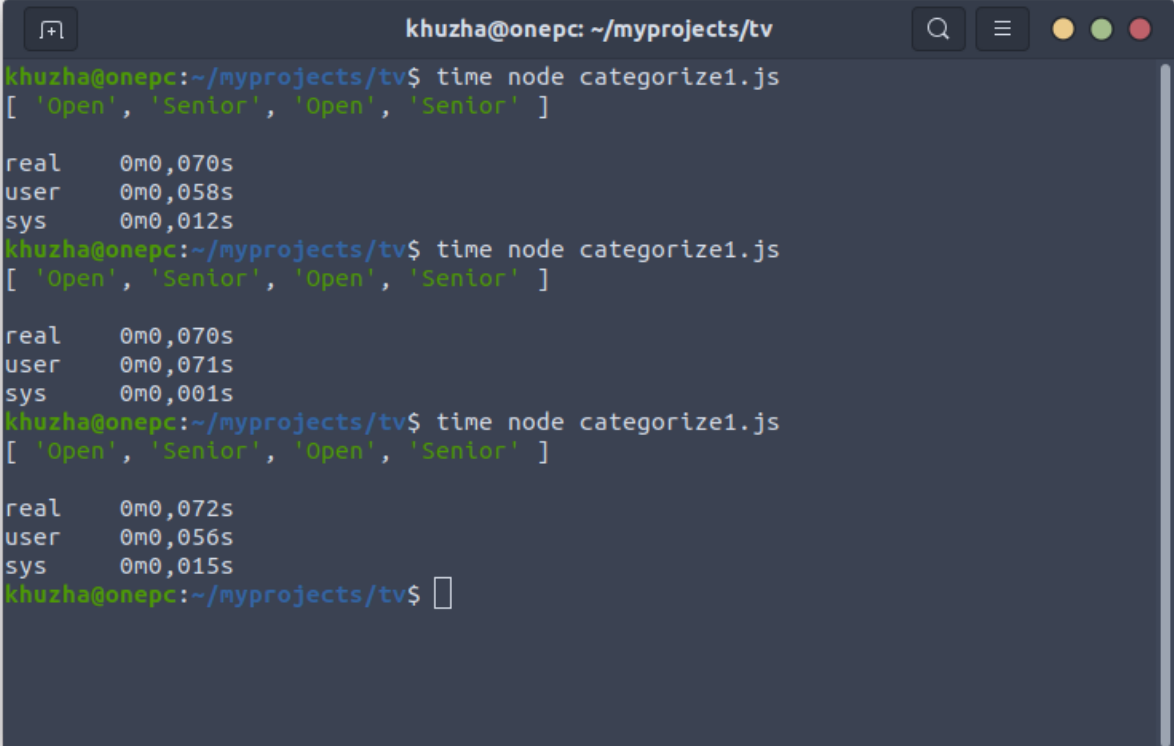
real    0m0,073s
user    0m0,064s
sys     0m0,008s
khuzha@onepc:~/myprojects/tv$
```

Решение №3:

Аналогичное второму решению, но без конструкции `const ... of ...`

```
1. function openOrSenior(data) {
2.   const res = []
3.
4.   for (let i = 0; i < data.length; i++) {
5.     if (data[i][0] >= 55 && data[i][1] > 7) {
6.       res.push('Senior')
7.     } else {
8.       res.push('Open')
9.     }
10.  }
11.  return res
12. }
13.
14. console.log(openOrSenior([[45, 12],[55,21],[19, -2],[104, 20]]))
```

Код функции занимает 12 строк и выполняется в среднем за 68 миллисекунд:

A terminal window titled 'khuzha@onepc: ~/myprojects/tv' showing the execution of a JavaScript file 'categorize1.js' using Node.js. The command 'time node categorize1.js' is run three times. Each time, it outputs an array ['Open', 'Senior', 'Open', 'Senior'] and timing statistics. The first run shows 0m0,070s real, 0m0,058s user, and 0m0,012s sys. The second run shows 0m0,070s real, 0m0,071s user, and 0m0,001s sys. The third run shows 0m0,072s real, 0m0,056s user, and 0m0,015s sys. The terminal window has a dark theme and standard window controls at the top.

```
khuzha@onepc: ~/myprojects/tv
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

real    0m0,070s
user    0m0,058s
sys     0m0,012s
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

real    0m0,070s
user    0m0,071s
sys     0m0,001s
khuzha@onepc:~/myprojects/tv$ time node categorize1.js
[ 'Open', 'Senior', 'Open', 'Senior' ]

real    0m0,072s
user    0m0,056s
sys     0m0,015s
khuzha@onepc:~/myprojects/tv$
```

2. Написать метод, расширяющий прототип класса String (String.prototype), который вызывается к строке, и делает каждую первую букву каждого слова этой строки заглавными. Словами считаются наборы символов, разделенные пробелами, табами и другими символами, входящими в квантификатор регулярных выражений \s.

В тестах будет использоваться следующий текст в переменной text:

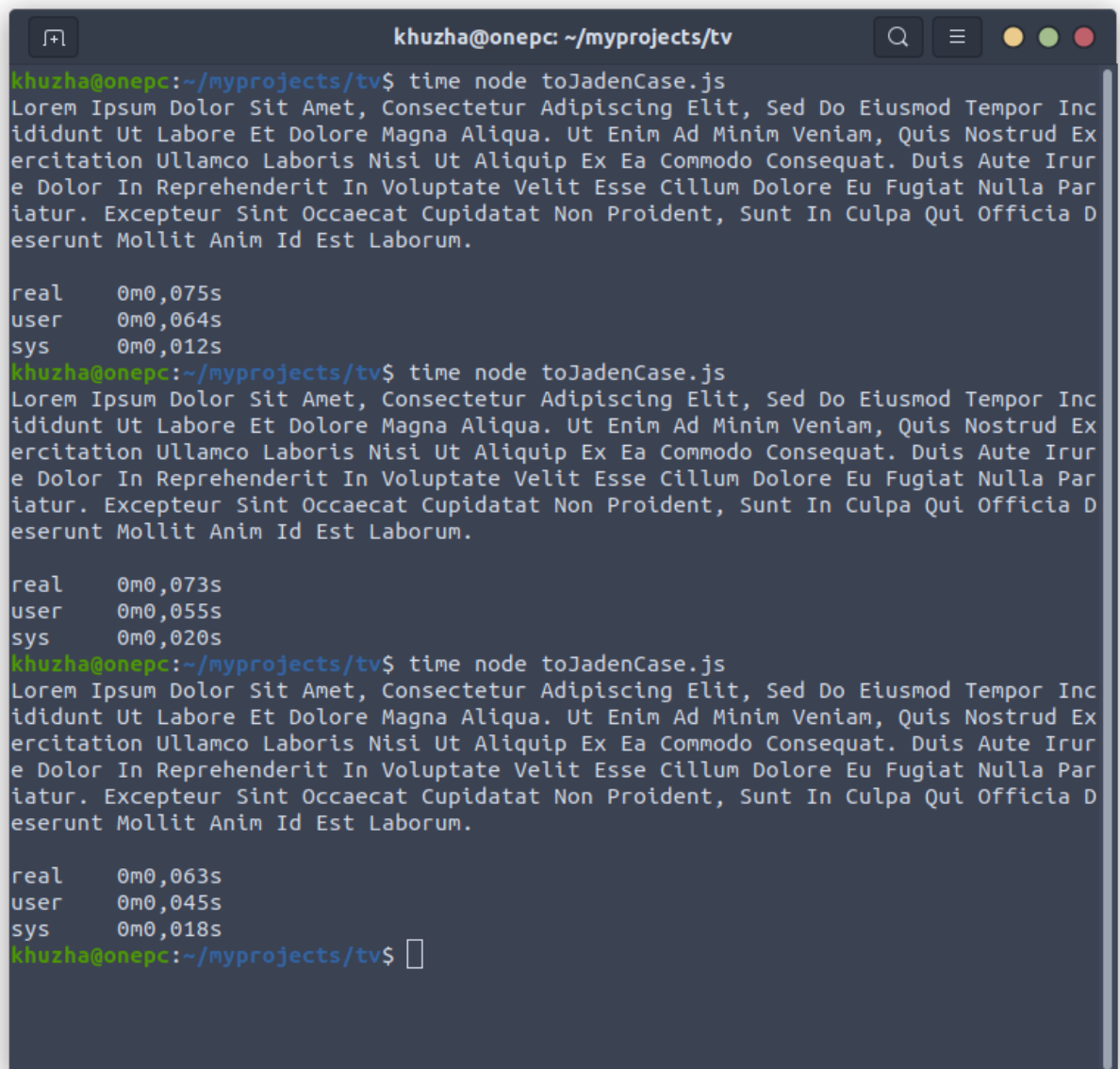
'Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.'

Решение №1

Используем методы `.split()`, `.toUpperCase()`, `.substr()` класса `String` и метод `.map()` класса `Array`

```
1. String.prototype.toJadenCase = function () {  
2.   return this.split(/\s/).map(word => word[0].toUpperCase() +  
   word.substr(1)).join(' ')  
3. }  
4.  
5. text = text.toJadenCase()  
6. console.log(text)
```

Код функции содержит 3 строки и выполняется в среднем за 70 микросекунд:



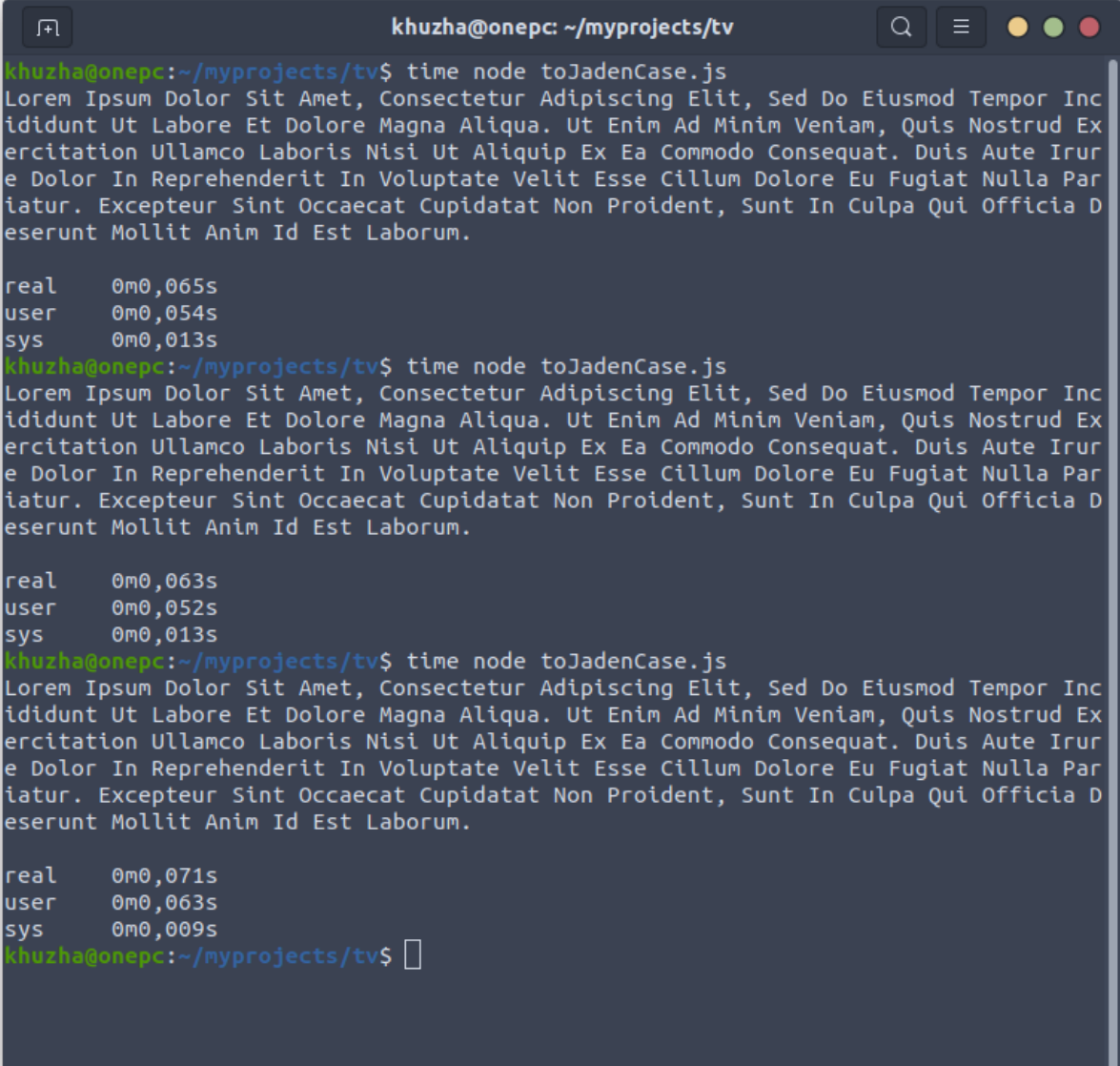
```
khuzha@onepc: ~/myprojects/tv  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,075s  
user    0m0,064s  
sys     0m0,012s  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,073s  
user    0m0,055s  
sys     0m0,020s  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,063s  
user    0m0,045s  
sys     0m0,018s  
khuzha@onepc:~/myprojects/tv$
```

Решение №2:

Используем цикл `for`, методы `.split()`, `.toUpperCase()`, `.substr()` класса `String` и метод `.join()` класса `Array`:

```
1. String.prototype.toJadenCase = function () {  
2.   const words = this.split(/\s/)   
3.   for (let i = 0; i < words; i++) {  
4.     words[i] = words[i][0].toUpperCase() + words[i].substr(1)  
5.   }  
6.   return words.join(' ')  
7. }  
8.  
9. text = text.toJadenCase()  
10. console.log(text)
```

Функция содержит 7 строк кода и выполняется в среднем за 66 миллисекунд:



```
khuzha@onepc: ~/myprojects/tv  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,065s  
user    0m0,054s  
sys     0m0,013s  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,063s  
user    0m0,052s  
sys     0m0,013s  
khuzha@onepc:~/myprojects/tv$ time node toJadenCase.js  
Lorem Ipsum Dolor Sit Amet, Consectetur Adipiscing Elit, Sed Do Eiusmod Tempor Inc  
ididunt Ut Labore Et Dolore Magna Aliqua. Ut Enim Ad Minim Veniam, Quis Nostrud Ex  
ercitation Ullamco Laboris Nisi Ut Aliquip Ex Ea Commodo Consequat. Duis Aute Irur  
e Dolor In Reprehenderit In Voluptate Velit Esse Cillum Dolore Eu Fugiat Nulla Par  
iatur. Excepteur Sint Occaecat Cupidatat Non Proident, Sunt In Culpa Qui Officia D  
eserunt Mollit Anim Id Est Laborum.  
  
real    0m0,071s  
user    0m0,063s  
sys     0m0,009s  
khuzha@onepc:~/myprojects/tv$
```

Вывод

Код нельзя оценивать по его длине, ведь, как и сказал один программист, “Оценивать программу по количеству строк кода - тоже самое, что оценивать самолет по его массе”. Код должен быть читаемым, красивым, подходить под стандарты и быть согласован с конвенциями и, по возможности, занимать меньше места. Но один из главных его показателей, время исполнения кода, не зависит от вышеперечисленных факторов. По крайней мере, не настолько, чтобы быть заметным человеку.