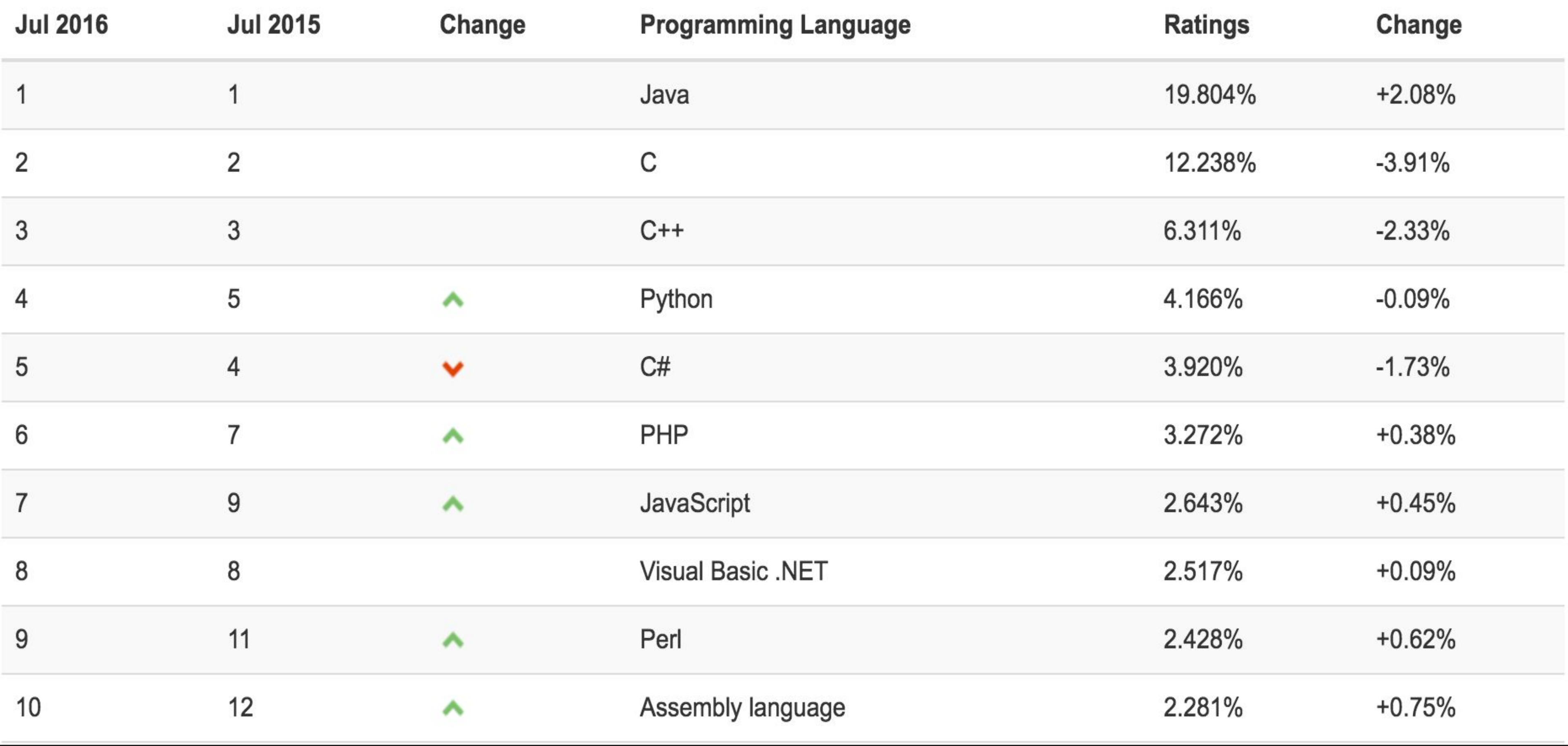
Java 8 - λ & Streams

Why do we care?



Jul 2016	Jul 2015	Change	Programming Language	Ratings	Change
1	1		Java	19.804%	+2.08%
2	2		C	12.238%	-3.91%
3	3		C++	6.311%	-2.33%
4	5	▲	Python	4.166%	-0.09%
5	4	▼	C#	3.920%	-1.73%
6	7	▲	PHP	3.272%	+0.38%
7	9	▲	JavaScript	2.643%	+0.45%
8	8		Visual Basic .NET	2.517%	+0.09%
9	11	▲	Perl	2.428%	+0.62%
10	12	▲	Assembly language	2.281%	+0.75%

Source: http://www.tiobe.com/tiobe_index

What is a functional language?

- Data exists as inputs and outputs to functions
    - No state
- Functions do not create side effects
- Structures are typically immutable

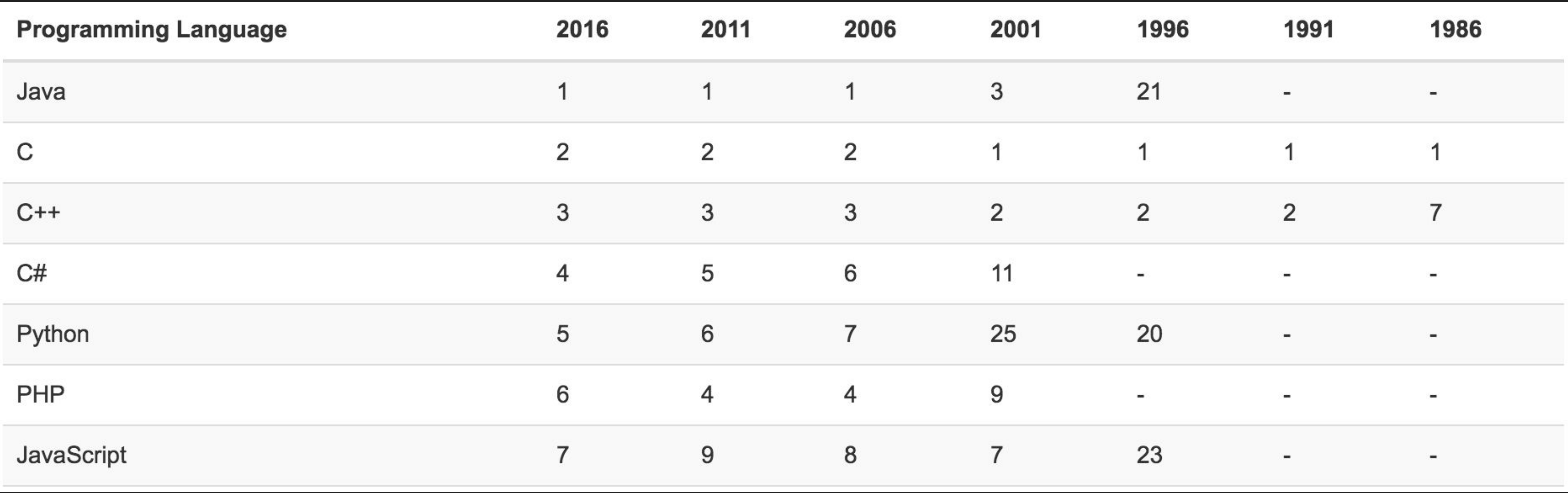
Why were they created?

- Predictability
- Scalability
- Immutability

Why does nobody use pure functional languages?

```
define          (void))
 (define          (void))
 (define              (void))
 (define                         (void))
 (define       (void))
 (define       (void))
 (begin
   (set!       0)
   (call/ec
    (lambda (break)
     ((lambda ($seq16             )
        (begin
          (begin
            (if (set?            )
              (for-set                         )
              (if (tuple?            )
                (for-tuple                         )
                (if (py-list?            )
                  (for-py-list                         )
                  (if (dict?            ) (for-dict                         ) (void)))))
            ((lambda () (begin (py-print       )))))))
      (py-list* 1 2 3)
      (lambda (i16)
        (call/ec
         (lambda (continue)
           (begin
             (set!            )
             ((lambda ()
                (begin (py-print       ) (set!       (+            ))))))))))))
```

Why were OO languages created?



Programming Language	2016	2011	2006	2001	1996	1991	1986
Java	1	1	1	3	21	-	-
C	2	2	2	1	1	1	1
C++	3	3	3	2	2	2	7
C#	4	5	6	11	-	-	-
Python	5	6	7	25	20	-	-
PHP	6	4	4	9	-	-	-
JavaScript	7	9	8	7	23	-	-

Source: http://www.tiobe.com/tiobe_index

Performance & Readability

- Mutable collections
- Garbage collection was slow
  - Create less objects (singletons, pass by reference)
- Objects and state are easy to understand

Downsides of OO

- Scalability is a challenge
  - How to share state between machines?
- Performance gains usually negligible
  - How many people routinely iterate over billions of elements?
- Mutability creates unexpected side effects

Can we marry the two paradigms?

Benefits of merging

- Easy to read code
- Allows benefits of both to be leveraged
- Better scalability for OO languages
- Allows compiler to be smarter

Examples of great mergers

python™

JavaScript

How did Java implement it?

Three Huge Changes

- Lambdas
- Method references
- Stream API

What is a Lambda (λ)?

- Lambdas are small blocks of code declared inline
- `x => x * x`
- `(x, y) => x + y`

Method References

- Allow existing methods to be referenced directly
- `String::valueOf`
- `this::myPrivateFunc`
- `instanceVar::myFunc`

Stream API

$$\begin{tikzcd}
\text{Stream} \arrow[dr] & & & \\
& \text{Filter} \arrow[dr] & & \\
& & \text{Map} \arrow[dr] & \\
& & & \text{Reduce}
\end{tikzcd}$$

Demo time

Problems with implementation

- Parallel streams are broken
- A lot of missing operations due to support for parallel

Some frameworks to fill the gap

- jOOλ (https://github.com/jOOQ/jOOL)
  - Aims to add back the missing sequential operations
- Rx for java
  - A lot of similar features as the stream API