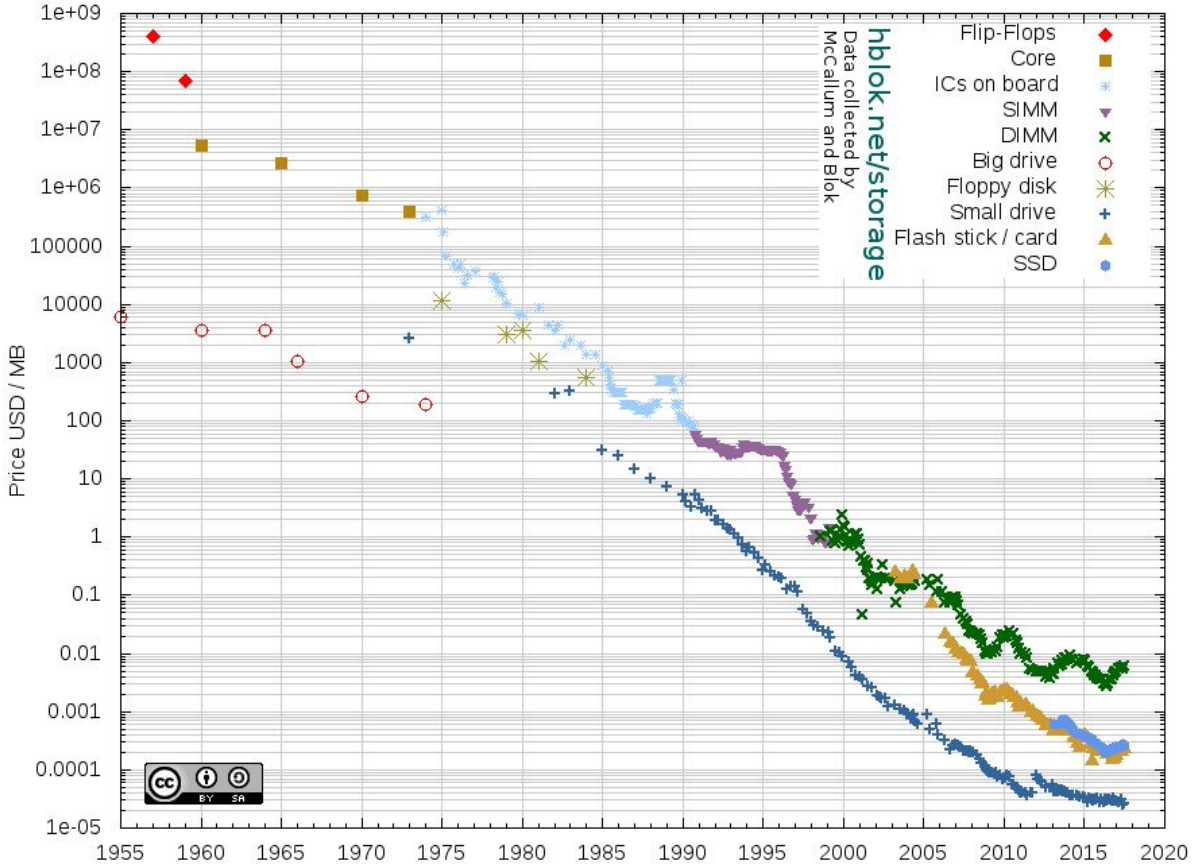
Functional Programming

Historical Cost of Computer Memory and Storage



A programming paradigm

Pure functions

Immutability

High order functions

First class functions

Inpure function :(

```
1   // Inpure
  1 function daysThisMonth () {
  2   const date = new Date()
  3   const year = date.getFullYear()
  4   const month = date.getMonth()
  5   const start = new Date(year, month, 1)
  6   const end = new Date(year, month + 1, 1)
  7   return Math.round((end - start) / (1000 * 60 * 60 * 24))
  8 }
```

Pure function :)

```
10 // Pure
11 function daysInMonth(year, month) {
12   const start = new Date(year, month, 1)
13   const end = new Date(year, month + 1, 1)
14   return Math.round((end - start) / (1000 * 60 * 60 * 24))
15 }
```

Side effect :(

```
10 let counter = 0
 9 function increment() {
 8   counter = counter + 1
 7   return counter
 6 }
 5
```

Side effect free :)

```
counter = 0
function incrementPure(counter) {
  return counter + 1
}
console.log(incrementPure(counter)) // 1
console.log(incrementPure(counter)) // 1
console.log(incrementPure(counter)) // 1
```

First class & High order functions

```
1   // First class
  1 const add = (x, y) => x + y
  2 const mult = (x, y) => x * y
  3
  4 //  High order
  5 const calculate = (fn, x, y) => fn(x, y)
  6
  7 calculate(add, 1, 2) // 3
  8 calculate(mult, 1, 2) // 2
~
```

Map

```
const studants = [
  { name: 'Anna', grade:6 },
  { name: 'John', grade:4 },
  { name: 'Maria', grade:9 }
]

const byName = object => object.name
const studentsByName = students.map(byName)
```

Filter

```
1   const studants = [
  1   { name: 'Anna', grade:6 },
  2   { name: 'John', grade:4 },
  3   { name: 'Maria', grade:9 }
  4 ]
  5
  6 const isApproved = student => student.grade >= 9
  7 const approvedStudents = students.filter(isApproved)
~
```

Pros vs Cons

- Previsibility
- Performance
- Easier to test / debug

- Memory usage

Clojure

Elixir

Erlang

Scala

Haskell

F#