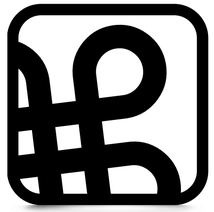


Macoun

10000 mal schneller Swift!
Oder?

Maxim Zaks

Vor langer langer Zeit …
2015

FlatBuffers

An open source project by FPL.

	FlatBuffers (binary)	Protocol Buffers LITE	Rapid JSON	FlatBuffers (JSON)	pugixml	Raw structs
Decode + Traverse + Dealloc (1 million times, seconds)	0.08	302	583	105	196	0.02
Decode / Traverse / Dealloc (breakdown)	0 / 0.08 / 0	220 / 0.15 / 81	294 / 0.9 / 287	70 / 0.08 / 35	41 / 3.9 / 150	0 / 0.02 / 0
Encode (1 million times, seconds)	3.2	185	650	169	273	0.15
Wire format size (normal / zlib, bytes)	344 / 220	228 / 174	1475 / 322	1029 / 298	1137 / 341	312 / 187
Memory needed to store decoded wire (bytes / blocks)	0 / 0	760 / 20	65689 / 4	328 / 1	34194 / 3	0 / 0
Transient memory allocated during decode (KB)	0	1	131	4	34	0

https://google.github.io/flatbuffers/flatbuffers_benchmarks.html

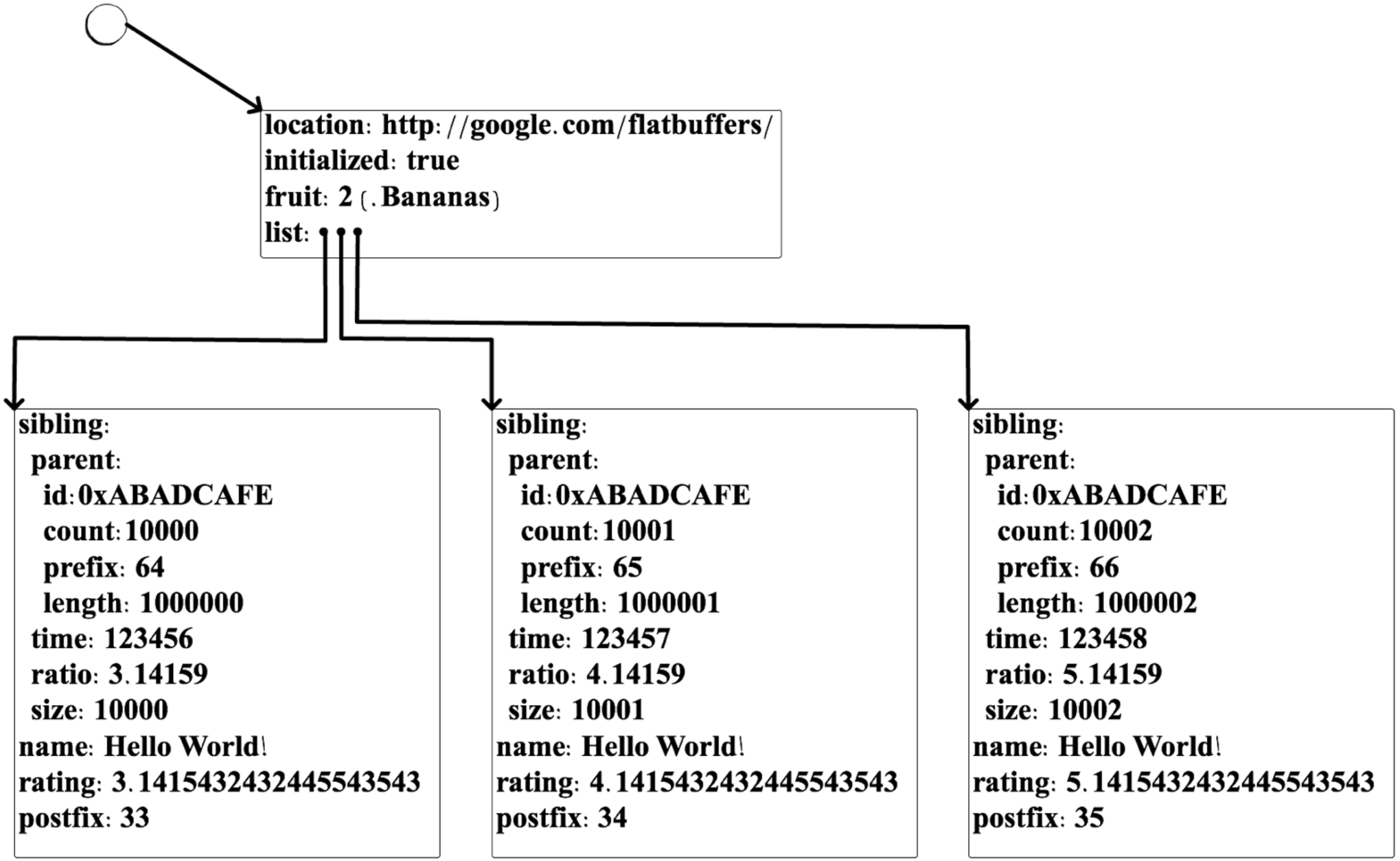
Was ist das für ein Hexenwerk?
Dieses FlatBuffers

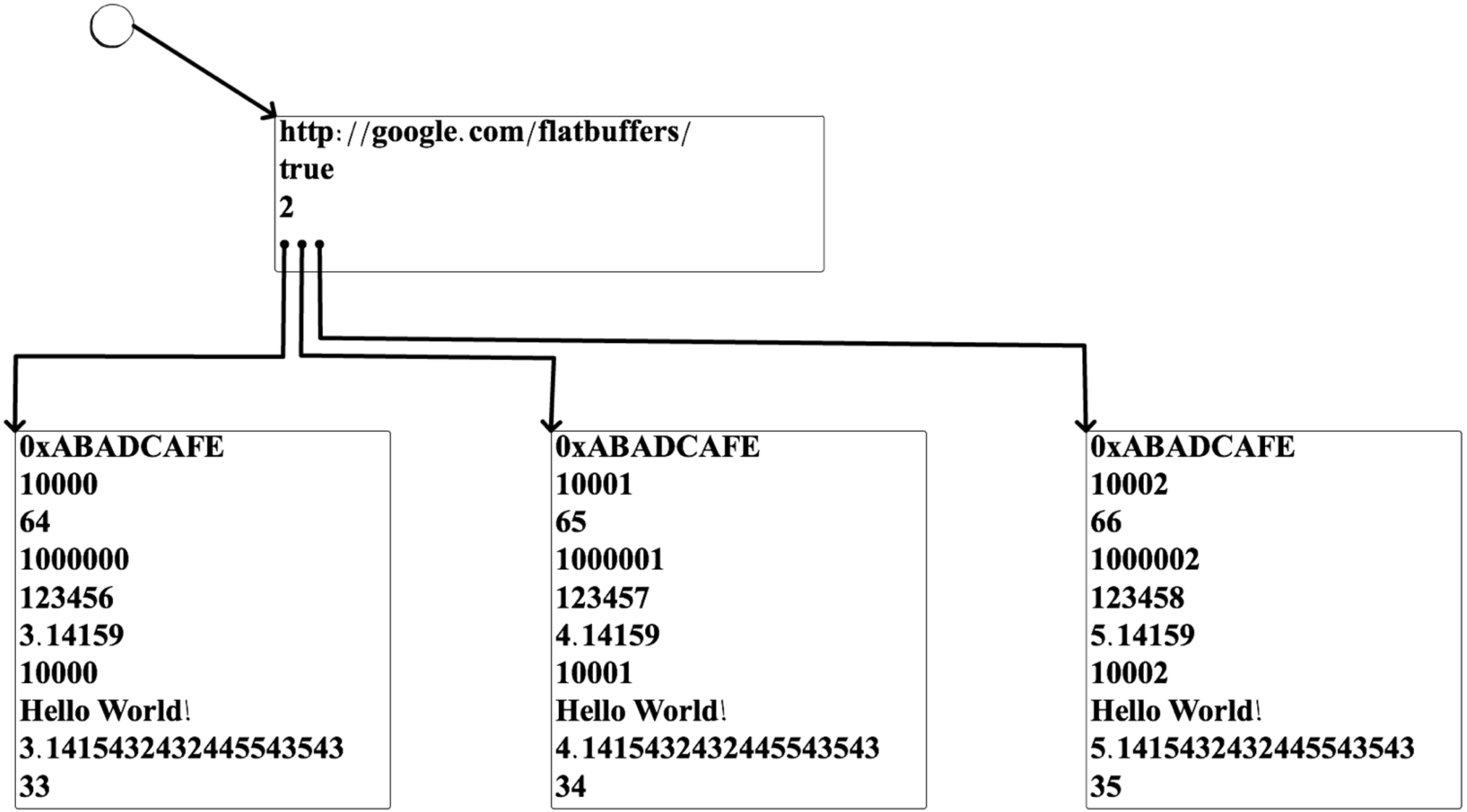
- IDL (Schema)
- 344 bytes
- 1Mio mal lesen

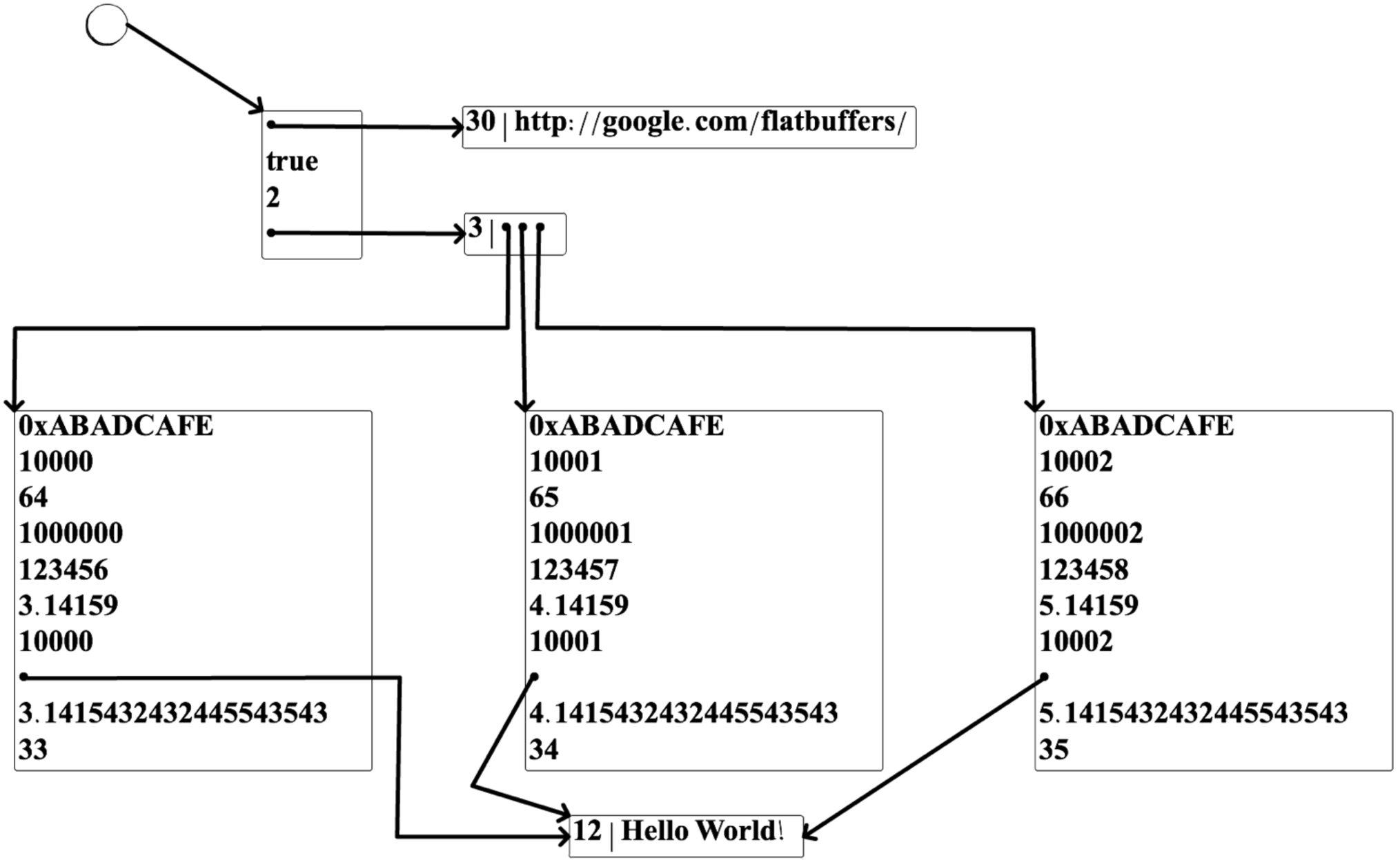
bench.fbs

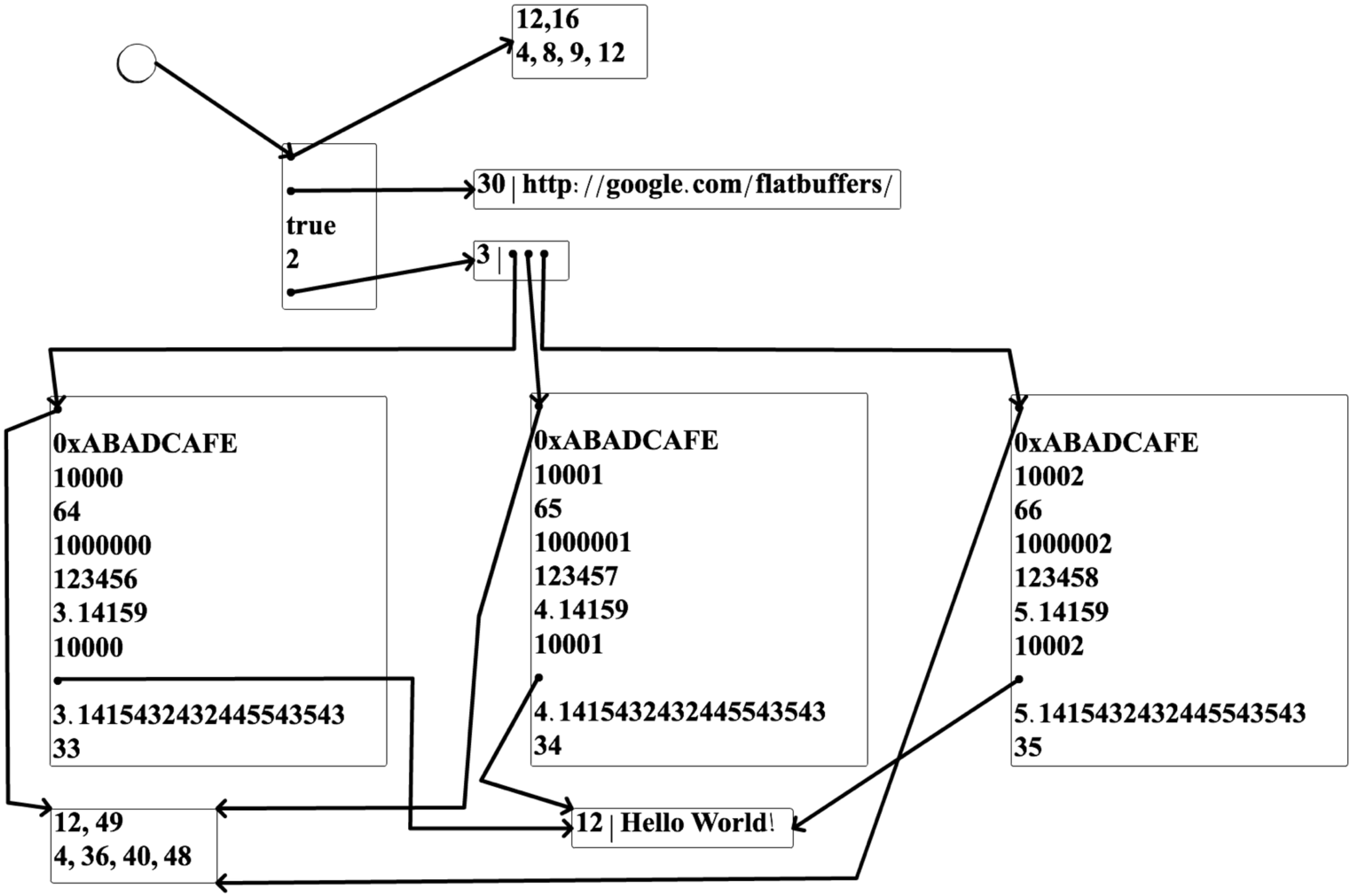
```
namespace benchfb;

enum Enum : short { Apples, Pears, Bananas }

struct Foo {
  id:ulong;
  count:short;
  prefix:byte;
  length:uint;
}

struct Bar {
  parent:Foo;
  time:int;
  ratio:float;
  size:ushort;
}

table FooBar {
  sibling:Bar;
  name:string;
  rating:double;
  postfix:ubyte;
}

table FooBarContainer {
  list:[FooBar];  // 3 copies of the above
  initialized:bool;
  fruit:Enum;
  location:string;
}

root_type FooBarContainer;
```


```
{
  "location":"http://google.com/flatbuffers/",
  "initialized":true,
  "fruit":"Bananas",
  "list":[
    {
      "sibling":{
        "parent":{
          "i_d":0xABADCAFE,
          "count":10000,
          "prefix":64,
          "length":1000000
        },
        "time":123456,
        "ratio":3.14159,
        "size":10000
      },
      "name":"Hello, World!",
      "rating":3.1415432432445543543,
      "postfix":33
    },{
      "sibling":{
        "parent":{
          "i_d":0xABADCAFE,
          "count":10001,
          "prefix":65,
          "length":1000001
        },
        "time":123457,
        "ratio":4.14159,
        "size":10001
      },
      "name":"Hello, World!",
      "rating":4.1415432432445543543,
      "postfix":34
    },{
      "sibling":{
        "parent":{
          "i_d":0xABADCAFE,
          "count":10002,
          "prefix":66,
          "length":1000002
        },
        "time":123458,
        "ratio":5.14159,
        "size":10002
      },
      "name":"Hello, World!",
      "rating":5.1415432432445543543,
      "postfix":35
    }
  ]
}
```


location: http://google.com/flatbuffers/
initialized: true
fruit: 2 (.Bananas)
list:
sibling:
parent:
id:0xABADCAFE
count:10000
prefix: 64
length: 1000000
time: 123456
ratio: 3.14159
size: 10000
name: Hello World!
rating: 3.14154324325455435 43
postfix: 33
sibling:
parent:
id:0xABADCAFE
count:10001
prefix: 65
length: 1000001
time: 123457
ratio: 4.14159
size: 10001
name: Hello World!
rating: 4.14154324325455435 43
postfix: 34
sibling:
parent:
id:0xABADCAFE
count:10002
prefix: 66
length: 1000002
time: 123458
ratio: 5.14159
size: 10002
name: Hello World!
rating: 5.14154324325455435 43
postfix: 35


http://google.com/flatbuffers/
true
2
0xABADCAFE
10000
64
1000000
123456
3.14159
10000
Hello World!
3.1415432432445543543
33
0xABADCAFE
10001
65
1000001
123457
4.14159
10001
Hello World!
4.1415432432445543543
34
0xABADCAFE
10002
66
1000002
123458
5.14159
10002
Hello World!
5.1415432432445543543
35


true
2
30 | http://google.com/flatbuffers/
3 |
0xABADCAFE
10000
64
1000000
123456
3.14159
10000
3.14154324324455435 43
33
0xABADCAFE
10001
65
1000001
123457
4.14159
10001
4.14154324324455435 43
34
0xABADCAFE
10002
66
1000002
123458
5.14159
10002
5.14154324324455435 43
35
12 | Hello World!


12,16
4, 8, 9, 12
true
2
30 | http://google.com/flatbuffers/
3 |
0xABADCAFE
10000
64
1000000
123456
3.14159
10000
3.1415432432445543543
33
0xABADCAFE
10001
65
1000001
123457
4.14159
10001
4.1415432432445543543
34
0xABADCAFE
10002
66
1000002
123458
5.14159
10002
5.1415432432445543543
35
12, 49
4, 36, 40, 48
12 | Hello World!

•,12,16,4,8,9,12,•,•,true,2,•,30,http://google.com/flatbuffers/,
3,•,•,•,•,0xABADCAFE,
10000,64,1000000,123456,3.14159,10000,•,
3.14154324324455435 43,33,•,0xABADCAFE,
10001,65,1000001,123457,4.14159,10001,•,
4.14154324324455435 43,34,10,49,4,36,40,48,•,0xABADCAFE,
10002,66,1000002,123458,5.14159,10002,•,
5.14154324324455435 43,35,12,Hello World!

Ich habe mich verliebt

```
enum Enum : short {
   Apples, Pears, Bananas
}
struct Foo {
  id      : ulong;
  count   : short;
  prefix  : byte;
  length  : uint;
}
struct Bar {
  parent  : Foo;
  time    : int;
  ratio   : float;
  size    : ushort;
}
```

```
public enum Enum : Int16 {
    case Apples, Pears, Bananas
}
public struct Foo : Scalar {
    public let id      : UInt64
    public let count   : Int16
    public let prefix  : Int8
    public let length  : UInt32
}
public struct Bar : Scalar {
    public let parent  : Foo
    public let time    : Int32
    public let ratio   : Float32
    public let size    : UInt16
}
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
table FooBar {
  sibling : Bar;
  name    : string;
  rating  : double;
  postfix : ubyte;
}
```

```
public final class FooBar {
    public var sibling : Bar? = nil
    public var name    : String? = nil
    public var rating  : Float64 = 0
    public var postfix : UInt8 = 0
    public init(){}
    public init(sibling: Bar?,
                name: String?,
                rating: Float64,
                postfix: UInt8)
    {
        self.sibling = sibling
        self.name    = name
        self.rating  = rating
        self.postfix = postfix
    }
}
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
table FooBarContainer {
  list:[FooBar];
  initialized:bool;
  fruit:Enum;
  location:string;
}

root_type FooBarContainer;
```

```
public final class FooBarContainer {
    public var list : ContiguousArray<FooBar?> = []
    public var initialized : Bool = false
    public var fruit : Enum? = Enum.Apples
    public var location : String? = nil
    public init(){}
    public init(list: ContiguousArray<FooBar?>,
         initialized: Bool,
               fruit: Enum?,
            location: String?)
    {
        self.list        = list
        self.initialized = initialized
        self.fruit       = fruit
        self.location    = location
    }
}
```

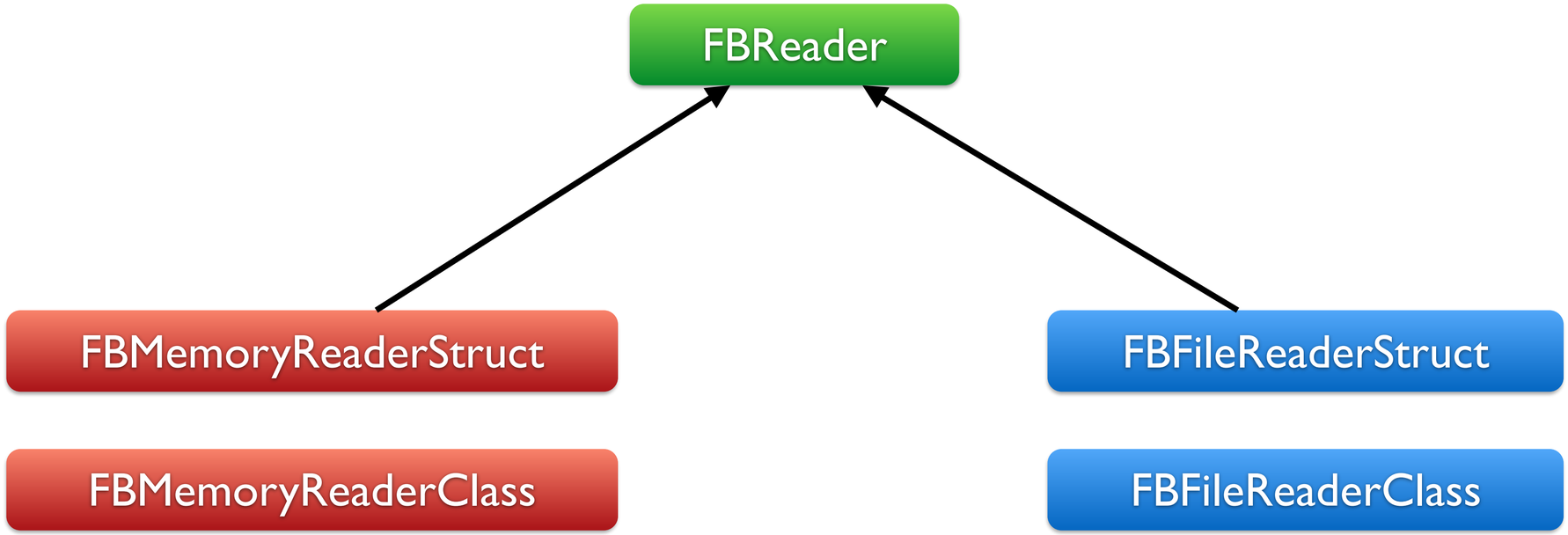
Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
public extension FooBarContainer {
    public func toFlatBufferBuilder (builder : FlatBufferBuilder) throws -> Void {
        let offset = addToByteArray(builder)
        try builder.finish(offset, fileIdentifier: nil)
    }
}

let container = FooBarContainer()
// do something
let builder   = FlatBufferBuilder(config: BinaryBuildConfig())
do {
    container.toFlatBufferBuilder(builder)
} catch { }
let data = builder.data
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
public extension FooBarContainer {
    public static func fromReader(reader : FBReader) -> FooBarContainer? {
        let objectOffset = reader.rootObjectOffset
        return create(reader, objectOffset : objectOffset)
    }
}

let reader = FBMemoryReaderStruct(buffer: buffer, count: count)
let container = FooBarContainer.fromReader(reader)
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden



Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

FlatBuffers

An open source project by FPL.

	FlatBuffers (binary)	Protocol Buffers LITE	Rapid JSON	FlatBuffers (JSON)	pugixml	Raw structs
Decode + Traverse + Dealloc (1 million times, seconds)	0.08	302	583	105	196	0.02
Decode / Traverse / Dealloc (breakdown)	0 / 0.08 / 0	220 / 0.15 / 81	294 / 0.9 / 287	70 / 0.08 / 35	41 / 3.9 / 150	0 / 0.02 / 0
Encode (1 million times, seconds)	3.2	185	650	169	273	0.15
Wire format size (normal / zlib, bytes)	344 / 220	228 / 174	1475 / 322	1029 / 298	1137 / 341	312 / 187
Memory needed to store decoded wire (bytes / blocks)	0 / 0	760 / 20	65689 / 4	328 / 1	34194 / 3	0 / 0
Transient memory allocated during decode (KB)	0	1	131	4	34	0

https://google.github.io/flatbuffers/flatbuffers_benchmarks.html

```
operation: flatcc for C encode (optimized)
elapsed time: 0.573 (s)
iterations: 1000000
size: 336 (bytes)
bandwidth: 586.619 (MB/s)
throughput in ops per sec: 1745889.304
throughput in 1M ops per sec: 1.746
time per op: 572.774 (ns)

operation: flatcc for C decode/traverse (optimized)
elapsed time: 0.027 (s)
iterations: 1000000
size: 336 (bytes)
bandwidth: 12550.426 (MB/s)
throughput in ops per sec: 37352457.792
throughput in 1M ops per sec: 37.352
time per op: 26.772 (ns)
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_eager_class
==================================
1359 ms encode
3890 ms decode
1093 ms use
4983 ms decode+use
==================================

decode_eager_struct
==================================
1372 ms encode
3259 ms decode
1110 ms use
4369 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

C vs. Swift
0,027s vs. 4,369s


```swift
private func flatuse(foobarcontainer : FooBarContainer, start : Int) -> Int
{
    var sum:Int = Int(start)
    sum = sum &+ Int(foobarcontainer.location!.utf8.count)
    sum = sum &+ Int(foobarcontainer.fruit!.rawValue)
    sum = sum &+ (foobarcontainer.initialized ? 1 : 0)

    for i in 0..<foobarcontainer.list.count {
        let foobar = foobarcontainer.list[i]!
        sum = sum &+ Int(foobar.name!.utf8.count)
        sum = sum &+ Int(foobar.postfix)
        sum = sum &+ Int(foobar.rating)

        let bar = foobar.sibling!

        sum = sum &+ Int(bar.ratio)
        sum = sum &+ Int(bar.size)
        sum = sum &+ Int(bar.time)

        let foo = bar.parent
        sum = sum &+ Int(foo.count)
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

Die Performance Sünden

- Ständige Erstellung von neuen Klassen vs. Object Pooling
- Konvertierung zu String vs. UnsafeBufferPointer<UInt8>
- let i = Int(b) vs. let i = b ? 1 : 0

```
private func functionalDecode(buffer : UnsafePointer<UInt8>) -> Int{

    let fooBarContainerOffset = getFooBarContainerRootOffset(buffer)

    var sum:Int = 0

    sum = sum &+ Int(getLocationFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset).count)

    sum = sum &+ Int(getFrootFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset).rawValue)
    sum = sum &+ (getInitializedFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset) ? 1 : 0)

    for i in 0..< getListCountFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset) {
        let foobarOffset = getFooBarOffsetFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset, listIndex: i)

        sum = sum &+ Int(getNameFrom(buffer, fooBarOffset: foobarOffset).count)
        sum = sum &+ Int(getPostfixFrom(buffer, fooBarOffset: foobarOffset))
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_functions
==================================
1381,2804818153 ms encode
0,0563263893 ms decode
0,4110932350 ms use
0,4674196243 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

C vs. Swift
0,027s vs. 0,00047s







Ich muss ein Blog Post schreiben

Wie sonst finde ich heraus dass ich ein Vollidiot bin?

Date	Views ↑	Reads	Read ratio	Recommends
10 thousand times faster Swift 6 min read · View story · Referrers	30K	12.6K	41%	156

Loop-invariant code motion

```
for (int i = 0; i < n; i++) {
   x = y + z;
   a[i] = 6 * i + x * x;
}
```

```
x = y + z;
t1 = x * x;
for (int i = 0; i < n; i++) {
   a[i] = 6 * i + t1;
}
```

https://en.wikipedia.org/wiki/Loop-invariant_code_motion

```
private func functionalDecode(buffer : UnsafePointer<UInt8>, start : Int) -> Int{

    let fooBarContainerOffset = getFooBarContainerRootOffset(buffer)

    var sum:Int = start

    sum = sum &+ Int(getLocationFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset).count)

    sum = sum &+ Int(getFrootFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset).rawValue)
    sum = sum &+ (getInitializedFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset) ? 1 : 0)

    for i in 0..< getListCountFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset) {
        let foobarOffset = getFooBarOffsetFrom(buffer,
                        fooBarContainerOffset: fooBarContainerOffset, listIndex: i)

        sum = sum &+ Int(getNameFrom(buffer, fooBarOffset: foobarOffset).count)
        sum = sum &+ Int(getPostfixFrom(buffer, fooBarOffset: foobarOffset))
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

C vs. Swift
0,026s vs. 0,042s

The End?

Nicht so schnell

Understanding Swift Performance
Session 416
WWDC 2016

Kann man mit mehr 🤔
bessere API hinkriegen?

```
extension FooBar {
    public struct Direct1 : Hashable {
        private let reader : FBReader
        private let myOffset : Offset
        init(reader: FBReader, myOffset: Offset){
            self.reader = reader
            self.myOffset = myOffset
        }
        public var sibling : Bar? {
            get { return reader.get(myOffset, propertyIndex: 0)}
        }
        public var name : UnsafeBufferPointer<UInt8>? { get { return
reader.getStringBuffer(reader.getOffset(myOffset, propertyIndex:1)) } }
        public var rating : Float64 {
            get { return reader.get(myOffset, propertyIndex: 2, defaultValue: 0) }
        }
        public var postfix : UInt8 {
            get { return reader.get(myOffset, propertyIndex: 3, defaultValue: 0) }
        }
        public var hashValue: Int { return Int(myOffset) }
    }
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_direct1_class
==================================
1365 ms encode
0 ms decode
2869 ms use
2869 ms decode+use
==================================

decode_direct1_struct
==================================
1380 ms encode
0 ms decode
1571 ms use
1571 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
extension FooBar {
    public struct Direct2 : Hashable {
        private let reader : FBMemoryReaderStruct
        private let myOffset : Offset
        init(reader: FBMemoryReaderStruct, myOffset: Offset){
            self.reader = reader
            self.myOffset = myOffset
        }
        public var sibling : Bar? {
            get { return reader.get(myOffset, propertyIndex: 0)}
        }
        public var name : UnsafeBufferPointer<UInt8>? { get { return
reader.getStringBuffer(reader.getOffset(myOffset, propertyIndex:1)) } }
        public var rating : Float64 {
            get { return reader.get(myOffset, propertyIndex: 2, defaultValue: 0) }
        }
        public var postfix : UInt8 {
            get { return reader.get(myOffset, propertyIndex: 3, defaultValue: 0) }
        }
        public var hashValue: Int { return Int(myOffset) }
    }
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_direct2
==================================
1394 ms encode
0 ms decode
87 ms use
87 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

Protocol vs. Struct
1,571s vs. 0,087s

Protocol Witness

Was währe mit Klasse 🤔

\+ `Reference sharing`?

```
extension FooBar {
    public struct Direct3 : Hashable {
        private let reader : Unmanaged<FBMemoryReaderClass>
        private let myOffset : Offset
        init(reader: FBMemoryReaderClass, myOffset: Offset){
            self.reader = Unmanaged.passUnretained(reader)
            self.myOffset = myOffset
        }
        public var sibling : Bar? {
            get { return reader.takeUnretainedValue().get(myOffset, propertyIndex:
0)}
        }
        public var name : UnsafeBufferPointer<UInt8>? { get { return
reader.takeUnretainedValue().getStringBuffer(reader.takeUnretainedValue().getOffset
(myOffset, propertyIndex:1)) } }
        public var rating : Float64 {
            get { return reader.takeUnretainedValue().get(myOffset, propertyIndex:
2, defaultValue: 0) }
        }
        public var postfix : UInt8 {
            get { return reader.takeUnretainedValue().get(myOffset, propertyIndex:
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_direct3
==================================
1370 ms encode
0 ms decode
296 ms use
296 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

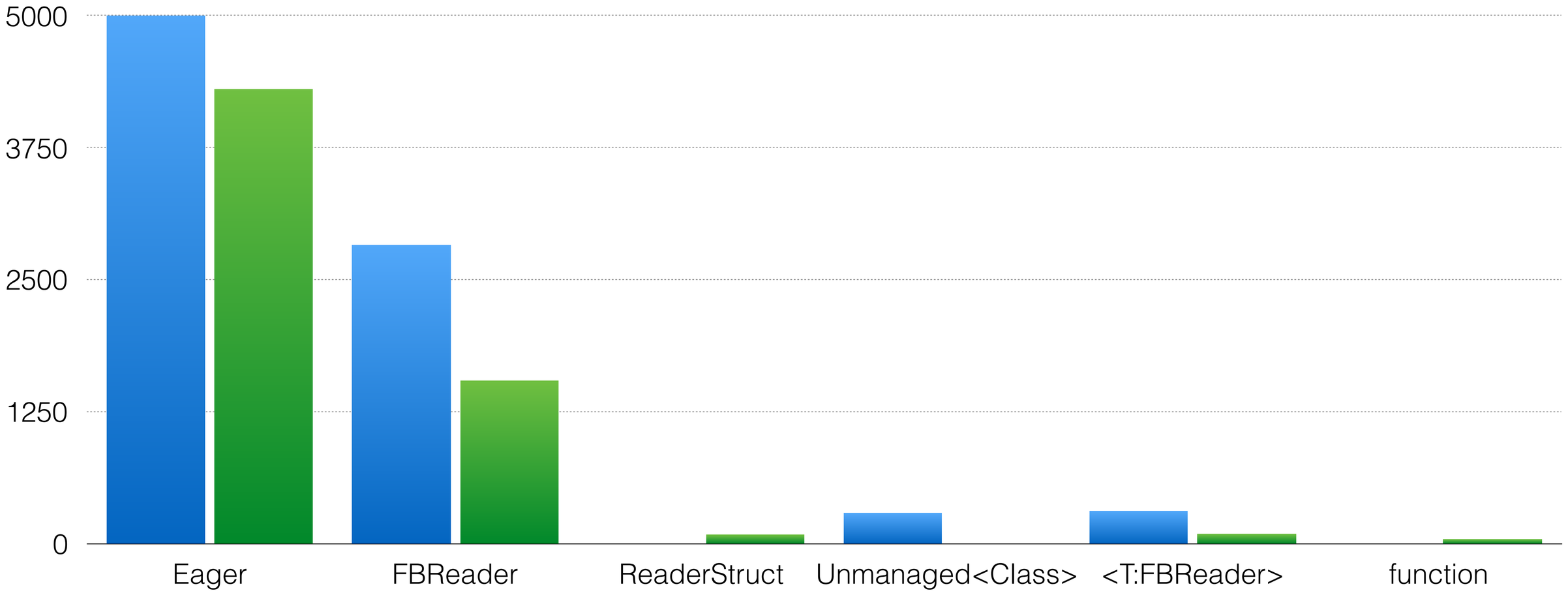
Haben sie nicht was von Generics erzählt 🤔

```
public struct FooBarDirect<T: FBReader> : Hashable {
    private let reader : T
    private let myOffset : Offset
    init(reader: T, myOffset: Offset){
        self.reader = reader
        self.myOffset = myOffset
    }
    public var sibling : Bar? {
        get { return reader.get(myOffset, propertyIndex: 0)}
    }
    public var name : UnsafeBufferPointer<UInt8>? { get { return
reader.getStringBuffer(reader.getOffset(myOffset, propertyIndex:1)) } }
    public var rating : Float64 {
        get { return reader.get(myOffset, propertyIndex: 2, defaultValue: 0) }
    }
    public var postfix : UInt8 {
        get { return reader.get(myOffset, propertyIndex: 3, defaultValue: 0) }
    }
    public var hashValue: Int { return Int(myOffset) }
}
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden

```
decode_direct4_class
==================================
1360 ms encode
0 ms decode
312 ms use
313 ms decode+use
==================================

decode_direct4_struct
==================================
1379 ms encode
0 ms decode
96 ms use
96 ms decode+use
==================================
```

Diese Linie ist markiert den sichtbare Bereich.
Darunter keinen Text mehr. Der kann verdeckt werden


5000
3750
2500
1250
0
Eager
FBReader
ReaderStruct
Unmanaged<Class>
<T:FBReader>
function

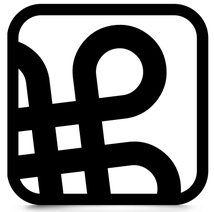
Was heißt das alles für meine Projekte?

Fragen?

Referenzen

- FlatBuffers (https://google.github.io/flatbuffers/)
- FlatBuffersSwift (https://github.com/mzaks/FlatBuffersSwift)
- Der Performance Test (https://github.com/mzaks/FBTest)
- Understanding Swift Performance (https://developer.apple.com/videos/play/wwdc2016/416/)

Vielen Dank



Macoun