

# JS, которого нет

 @zede1697  
 @izede  
 @izede

# Цели

- Узнать о разных сторонах JavaScript
- Осознать разницу подходов к восприятию JavaScript
- Задать очень много вопросов себе

Самый неправильно понятый язык программирования в мире стал  
самым популярным в мире языком программирования

*Дуглас Крокфорд*

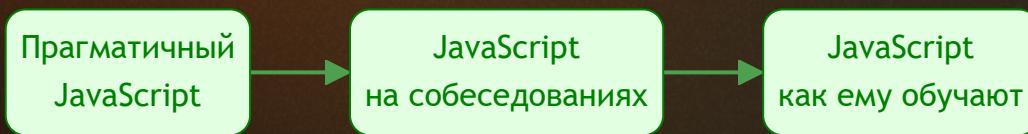
Я подчеркиваю слово Путешествие, потому что знание JS не является пунктом назначения, это направление

*Кайл Симпсон*

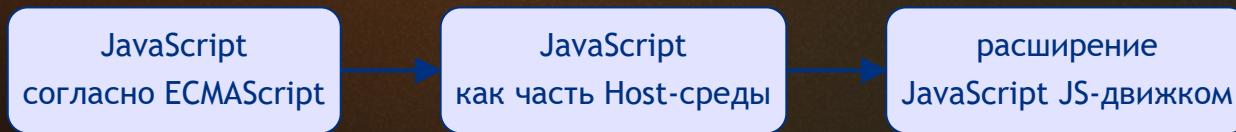
JavaScript **не** существует в единственной форме!



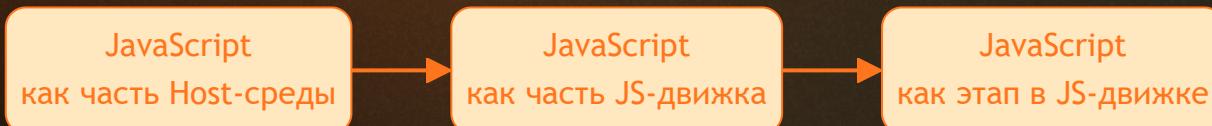
# Мифический JS



# Задокументированный JS



# JS в действии



# Прагматичный JavaScript

- Удобство важнее написания *идеального кода*
- TypeScript
- *Вера в магию движков JS*
- Работает — не трогай!

# Что-то тут не так...

```
// [['a', 1], ['b', 2]] => { a: 2, b: 4 }
pairs.reduce((acc, [key, value]) => {
  acc[key] = value * 2
  return acc
}, {})
```

# JavaScript мира собеседований

- Большой уровень погружения
- Основная зона для *мифов*
- Сами придумываем правила по которым собеседуем

# Типичный вопрос

Что в JS передается по ссылке, а что по значению?

- Объекты передаются по ссылке
- Примитивы по значению

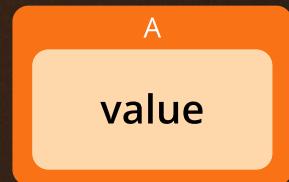
```
let num = 10
let obj = { item: 'initial' }
let obj2 = { item: 'initial' }

function changeStuff(a, b, c) {
  a = a * 10
  b.item = 'changed'
  c = { item: 'changed' }
}

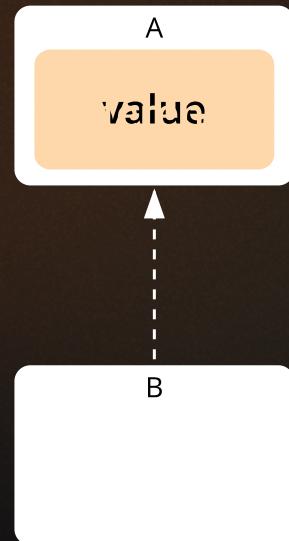
changeStuff(num, obj, obj2)

console.log(num) // 10
console.log(obj) // { item: 'changed' }
console.log(obj2) // { item: 'initial' }
```

# Передача по значению

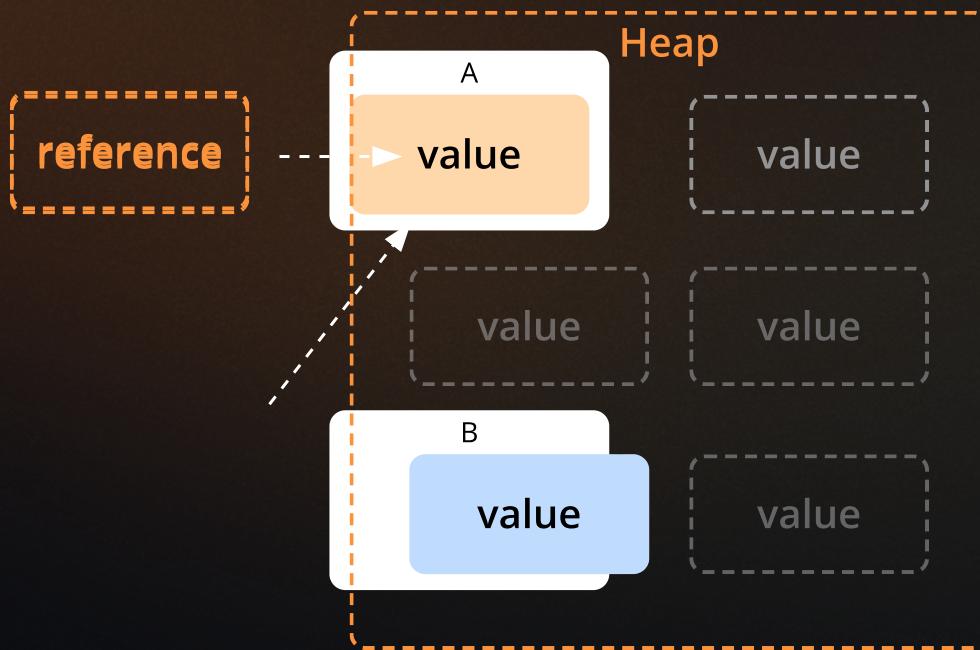


# Передача по ссылке





# Call-by-Sharing



# Что в итоге?

- Pass-by-Sharing
- Зависит от контекста
- Спецификация не оговаривает точный способ
- Движки для JS могут менять поведение для оптимизации
- На разных этапах рантайма движки могут менять поведение

# Как учим JavaScript?

- Изучение по принципу "народных сказаний"
- Описание может противоречить тому — как это работает
- По рекомендациям коллег

~~var~~  
Устарел!

# *Устаревшее* ключевое слово `var`

- В спецификации даже намека на устаревание `var` — нет
- Производительность `var` может быть значительно выше
- Огромный объем кода использует `var`



# TDZ, What Is It Good For?

—

Shu-yu Guo (Google)  
September 2023

# Так что всё-таки с `var`?

- На практике его можно частично считать устаревшим
- Сборщики / Транспайлеры приводят `let` / `const` к `var`
- `var` бывает крайне важен для производительности
- По спецификации — устаревания нет

# А что там по спецификации?

- ECMAScript (ECMA-262)
- JS в *вакууме*
- Созданы свои языки для спецификации
- Единственный источник истины про JS
- Достаточно не противоречить спецификации

# *Заклинания из спецификации*

- Свои типы данных
- Свои абстрактные методы
- Созданы свои языки для спецификации
- Свои уникальные языки для описания алгоритмов
- Есть концепты, которые еще предстоит реализовать Host-среде

# Примеры:

- Environment Records
- Abstract Closure
- VariableEnvironment
- Agent
- Job
- Host Environment

### 20.1.3.6 Object.prototype.toString()

This method performs the following steps when called:

1. If the **this** value is **undefined**, return "[object Undefined]" .
2. If the **this** value is **null**, return "[object Null]" .
3. Let **O** be ! **ToObject**( **this** value).
4. Let **isArray** be ? **IsArray**(**O**).
5. If **isArray** is **true**, let **builtinTag** be "Array" .
6. Else if **O** has a [[ParameterMap]] internal slot, let **builtinTag** be "Arguments" .
7. Else if **O** has a [[Call]] internal method, let **builtinTag** be "Function" .
8. Else if **O** has an [[ErrorData]] internal slot, let **builtinTag** be "Error" .
9. Else if **O** has a [[BooleanData]] internal slot, let **builtinTag** be "Boolean" .
10. Else if **O** has a [[NumberData]] internal slot, let **builtinTag** be "Number" .
11. Else if **O** has a [[StringData]] internal slot, let **builtinTag** be "String" .
12. Else if **O** has a [[DateValue]] internal slot, let **builtinTag** be "Date" .
13. Else if **O** has a [[RegExpMatcher]] internal slot, let **builtinTag** be "RegExp" .
14. Else, let **builtinTag** be "Object" .
15. Let **tag** be ? **Get**(**O**, @@toStringTag ).
16. If **tag** is not a String , set **tag** to **builtinTag**.
17. Return the string-concatenation of "[object ", **tag**, and "]".

*ForInStatement*<sub>[?Yield, ?Await, ?Return]</sub>:

**for** ( [lookahead ≠ **let** [ ] *LeftHandSideExpression*<sub>[?Yield, ?Await]</sub> **in** *Expression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

**for** ( **var** *ForBinding*<sub>[?Yield, ?Await]</sub> **in** *Expression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

**for** ( *ForDeclaration*<sub>[?Yield, ?Await]</sub> **in** *Expression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

**for** ( [lookahead ∈ { **let** , **async** **of** }] *LeftHandSideExpression*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

**for** ( **var** *ForBinding*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

**for** ( *ForDeclaration*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

[+Await] **for await** ( [lookahead ≠ **let** ] *LeftHandSideExpression*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

[+Await] **for await** ( **var** *ForBinding*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

[+Await] **for await** ( *ForDeclaration*<sub>[?Yield, ?Await]</sub> **of** *AssignmentExpression*<sub>[+In, ?Yield, ?Await]</sub> ) *Statement*<sub>[?Yield, ?Await, ?Return]</sub>

# Итого

- Не рассчитана на JS-программистов
- Требует очень много времени и усилий
- Полученные знания сложно применить на практике

# JavaScript в среде обитания

Host-среда - оболочка в рамках которых JavaScript может функционировать

- Браузер (HTML)
- Node.js
- bun
- QML в Qt
- PostgreSQL (plv8)

# Спецификация в HTML

- Дружелюбная к начинающим
- Содержит в себе описание Host-среды для работы браузера

### 8.1.7.2 Queuing tasks

To queue a task on a task source *source*, which performs a series of steps *steps*, optionally given an event loop *event loop* and a document *document*:

1. If *event loop* was not given, set *event loop* to the implied event loop.
2. If *document* was not given, set *document* to the implied document.
3. Let *task* be a new task.
4. Set *task*'s steps to *steps*.
5. Set *task*'s source to *source*.
6. Set *task*'s document to the *document*.
7. Set *task*'s script evaluation environment settings object set to an empty set.
8. Let *queue* be the task queue to which *source* is associated on *event loop*.
9. Append *task* to *queue*.

**Task queues** are **sets**, not **queues**, because the **event loop** processing model grabs the first **runnable task** from the chosen queue, instead of dequeuing the first task.

**Очереди задач** это **наборы**, а не **очереди**, так как модель выполнения **Event Loop**-а забирает первую **возможную для запуска задачу** из выбранной очереди, а не снимает первую задачу

# Предупреждение

Будем рассматривать преимущественно v8

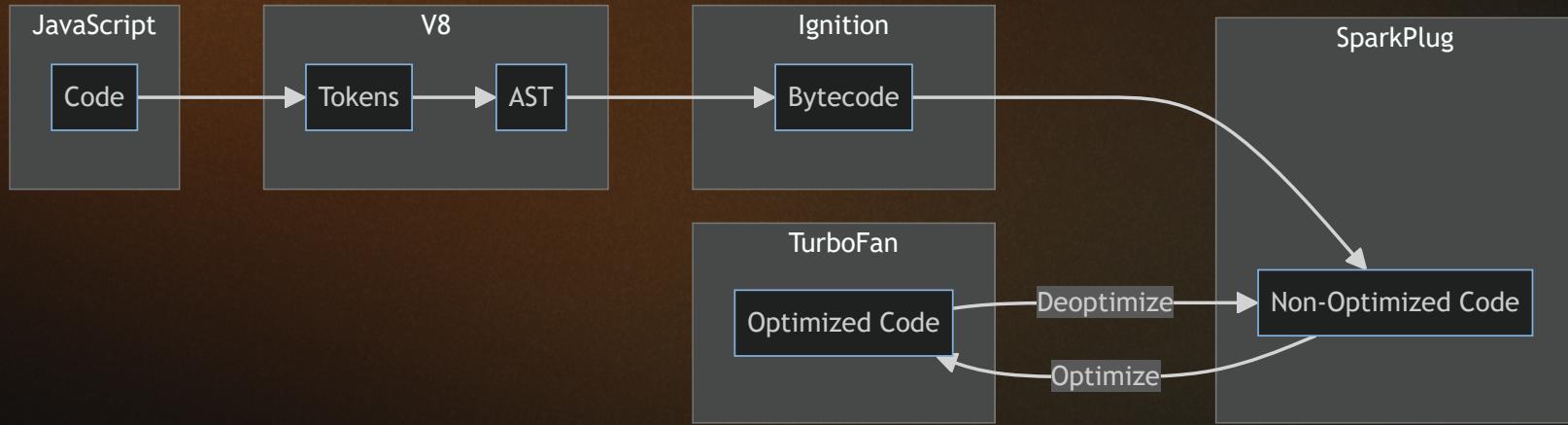
- Самый популярный
- Больше всего открытой информации
- Самый навороченный
- Многое справедливо и для других движков

# JavaScript в v8

- Непредсказуемый
- Значительно оптимизирующий
- Имеет несколько этапов жизни скрипта
- При работе с ним, нужно понимать, что делаешь

# Свои термины

- регистры
- SMI
- Fast Object
- Object Shape
- Hidden Classes
- Transition Map
- Bytecode Cache



```
const myConstantC = 9
function sum(a, b) {
    return a + b + myConstantC;
}

sum(1, 6)
```

Bytecode length: 14  
Parameter count 3  
Register count 1  
Frame size 8

Ldar a1  
Add a0, [0]  
Star0  
LdaImmutableCurrentContextSlot [2]  
ThrowReferenceErrorIfHole [0]  
Add r0, [1]  
Return

Constant pool (size = 1)  
0000023D00119591: [FixedArray] in OldSpace  
- map: 0x023d00000565 <Map(FIXED\_ARRAY\_TYPE)>  
- length: 1  
 0: 0x023d001193a1 <String[11]: #myConstantC>

# Что стоит учитывать о v8?

- Регистровый движок
- Ленивый
- Выжидает момент
- Крайне конфигурируемый
- Очень изменчивая среда
- Оптимизации v8 могут сменить даже сложность алгоритма

# Выводы:

- JS-код нельзя оценивать лишь по одному уровню
- К JS малоприменимы классические подходы оценки
- Развиваться в изучении JS можно бесконечно
- Задавайтесь вопросами при изучении

