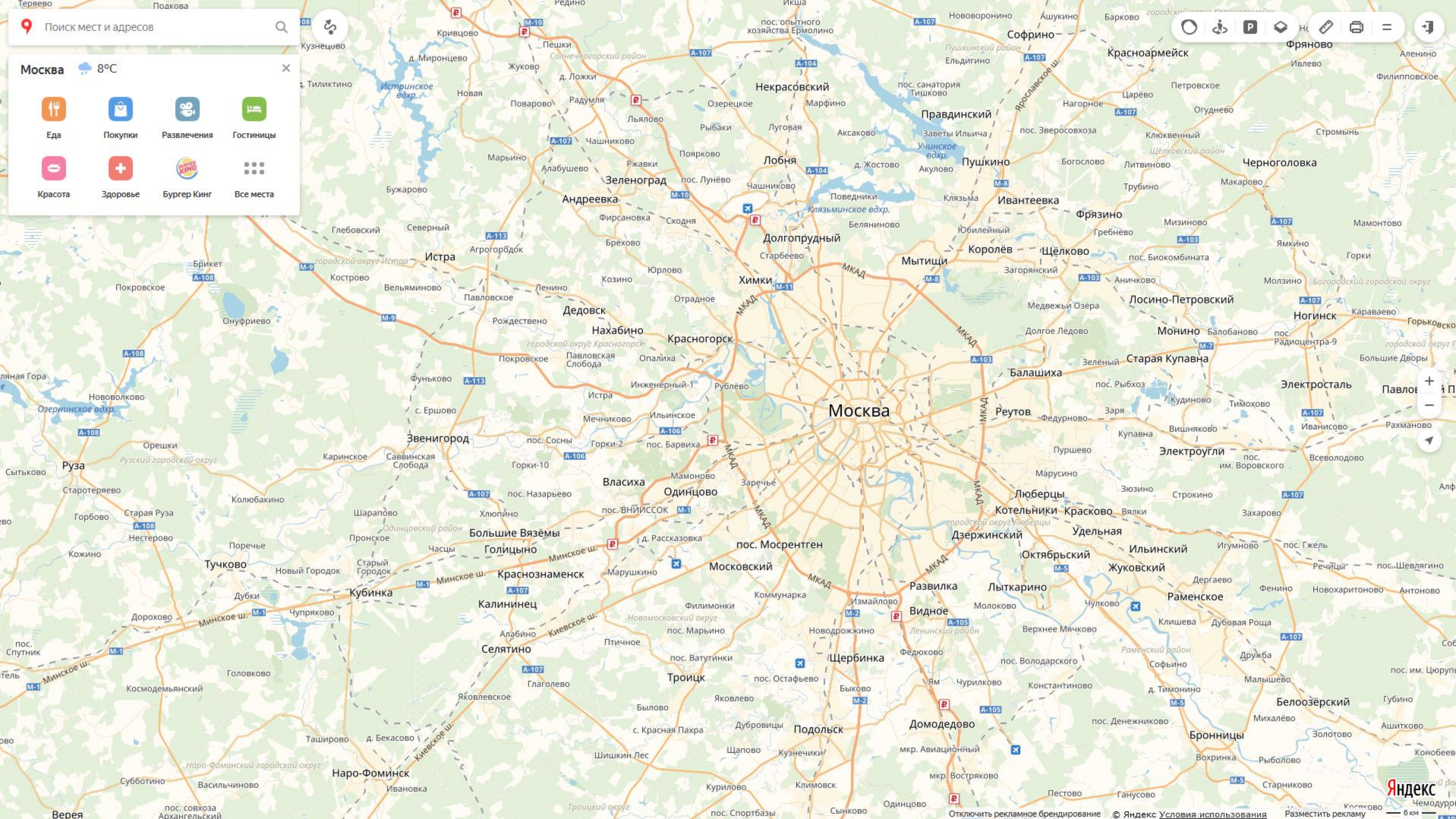

Яндекс

Яндекс Карты

garbage.collect()

Роенко Андрей
Разработчик API Яндекс.Карт

> Если бы в JavaScript действительно работала сборка мусора, большинство npm-модулей удаляли бы сами себя сразу после установки
>
> (наверное, так бы сказал) Роберт Сьюэл


Поиск мест и адресов
Москва
8°C
Еда
Покупки
Развлечения
Гостиницы
Красота
Здоровье
Бургер Кинг
Все места
Москва
Химки
Мытищи
Королёв
Щёлково
Фрязино
Пушкино
Ивантеевка
Черноголовка
Красноармейск
Фряново
Софрино
Правдинский
Некрасовский
Лобня
Долгопрудный
Зеленоград
Андреевка
Долгопрудный
Дедовск
Истра
Нахабино
Красногорск
Лосино-Петровский
Ногинск
Монино
Старая Купавна
Балашиха
Электросталь
Реутов
Электроугли
Люберцы
Котельники
Красково
Дзержинский
Октябрьский
Ильинский
Жуковский
Удельная
Раменское
Развилка
Видное
Домодедово
Подольск
Щербинка
Троицк
Селятино
Наро-Фоминск
Звенигород
Руза
Власиха
Одинцово
Большие Вязёмы
Голицыно
Краснознаменск
Кубинка
Калининец
Тучково
Московский
пос. Мосрентген
Лыткарино
Ильинский
Бронницы
Белоозёрский
Электроугли
Химки
Мытищи
Королёв
Щёлково
Фрязино
Пушкино
Ивантеевка
Черноголовка
Красноармейск
Фряново
Софрино
Правдинский
Некрасовский
Лобня
Долгопрудный
Зеленоград
Андреевка
Дедовск
Истра
Нахабино
Красногорск
Лосино-Петровский
Ногинск
Монино
Старая Купавна
Балашиха
Электросталь
Реутов
Электроугли
Люберцы
Котельники
Красково
Дзержинский
Октябрьский
Ильинский
Жуковский
Удельная
Раменское
Развилка
Видное
Домодедово
Подольск
Щербинка
Троицк
Селятино
Наро-Фоминск
Звенигород
Руза
Власиха
Одинцово
Большие Вязёмы
Голицыно
Краснознаменск
Кубинка
Калининец
Тучково
Московский
пос. Мосрентген
Лыткарино
Ильинский
Бронницы
Белоозёрский
Минское ш.
Киевское ш.
МКАД
Истринское вдхр.
Озернинское вдхр.
Клязьминское вдхр.
Учинское вдхр.
Отключить рекламное брендирование
© Яндекс Условия использования
Разместить рекламу
Яндекс

План доклада

1. Теория
2. Суровая реальность
3. Браузерная реальность
4. Повседневность

Зачем мне все это знать?

- Хорошо иметь представление о том как работает ваш инструмент
- Вы поймете где можно оптимизировать ваши приложения
- Не будете совершать ошибки
- Перестанете делать «оптимизации»

Все нетривиальные абстракции дырявы

Джоэл Спольски

Теория

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y’ };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

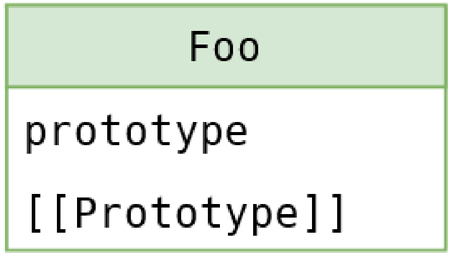
```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

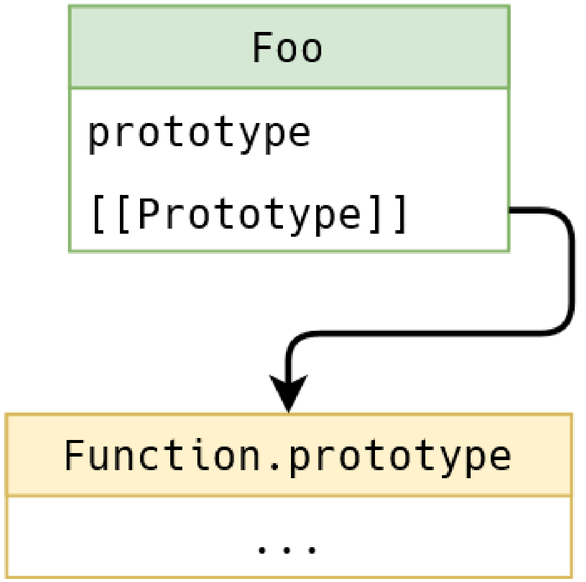
```
window.Foo = class Foo {
  constructor() {
    this.x = { y: 'y' };
  }
  work(name) {
    let z = 'z';
    return function () {
      console.log(name, this.x.y, z);
      this.x = null;
    }.bind(this);
  }
};
```

```
var foo = new Foo();
window.worker = foo.work('Brendan Eich');
window.foo = null;
window.Foo = null;
window.worker();
window.worker = null;
```

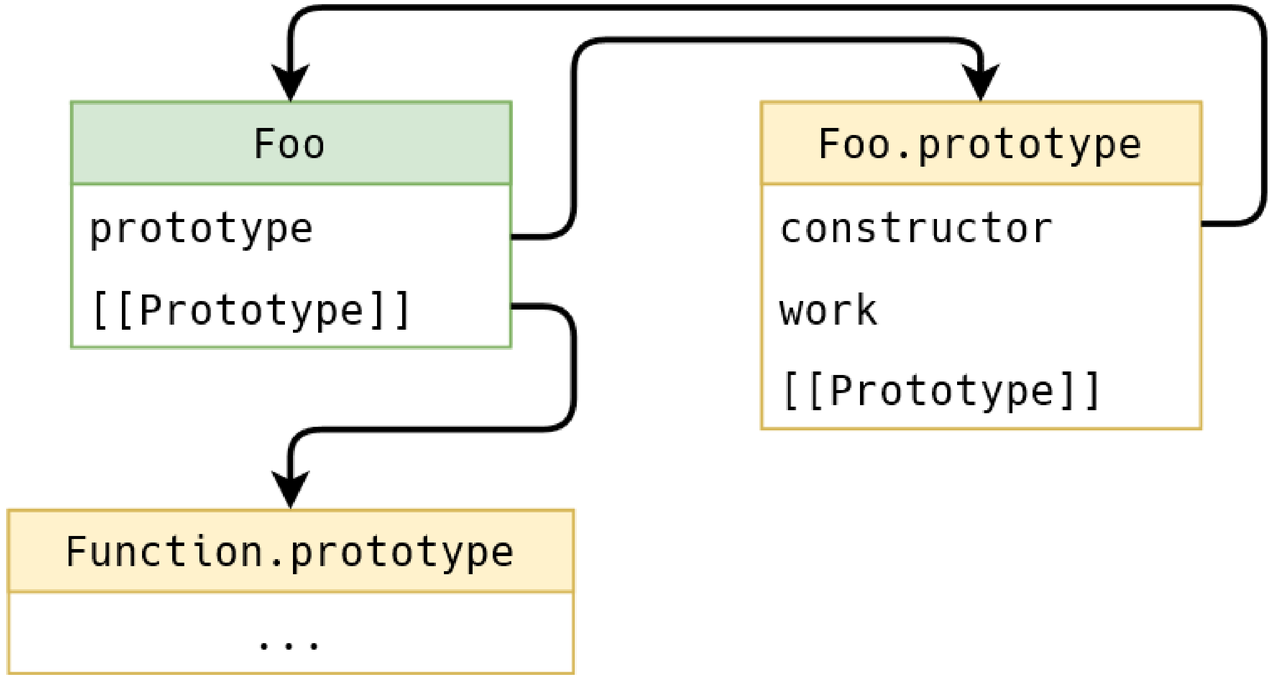
```
class Foo
```



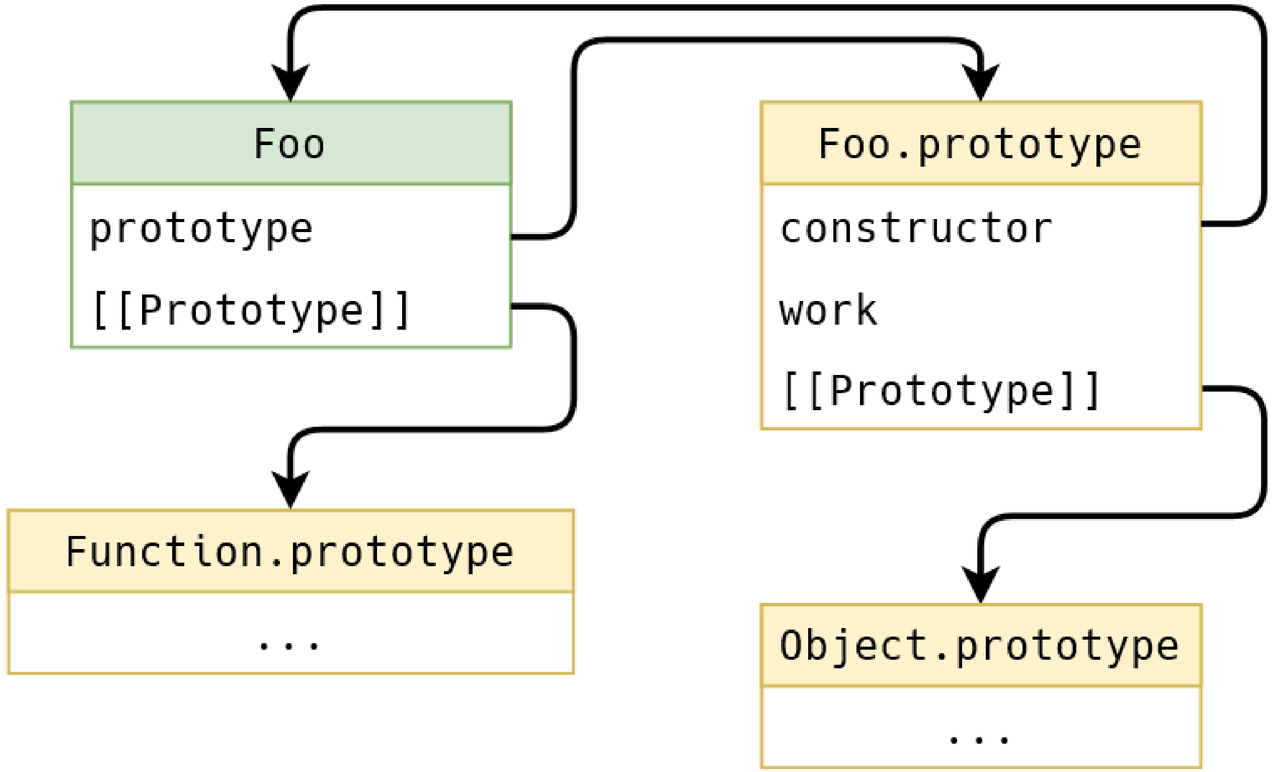
```
class Foo
```



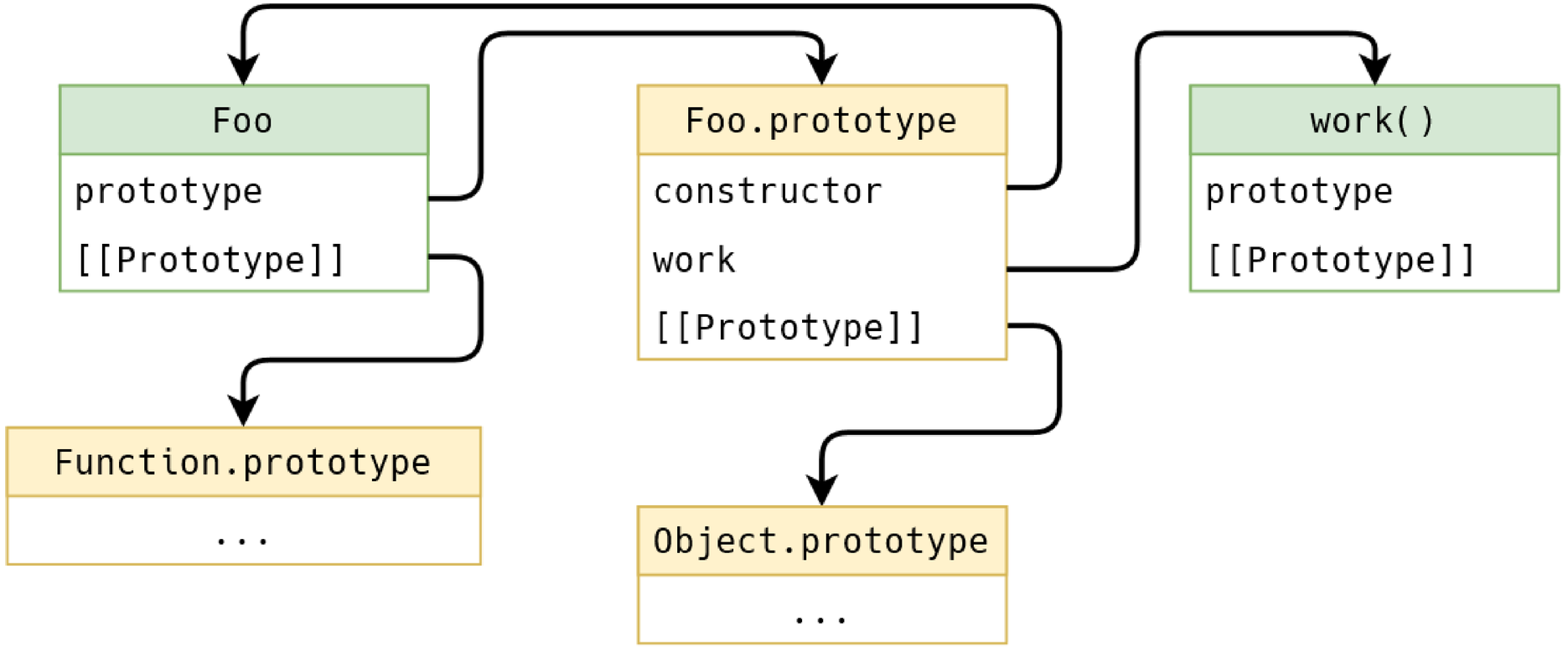
```
class Foo {
  constructor()
}
```



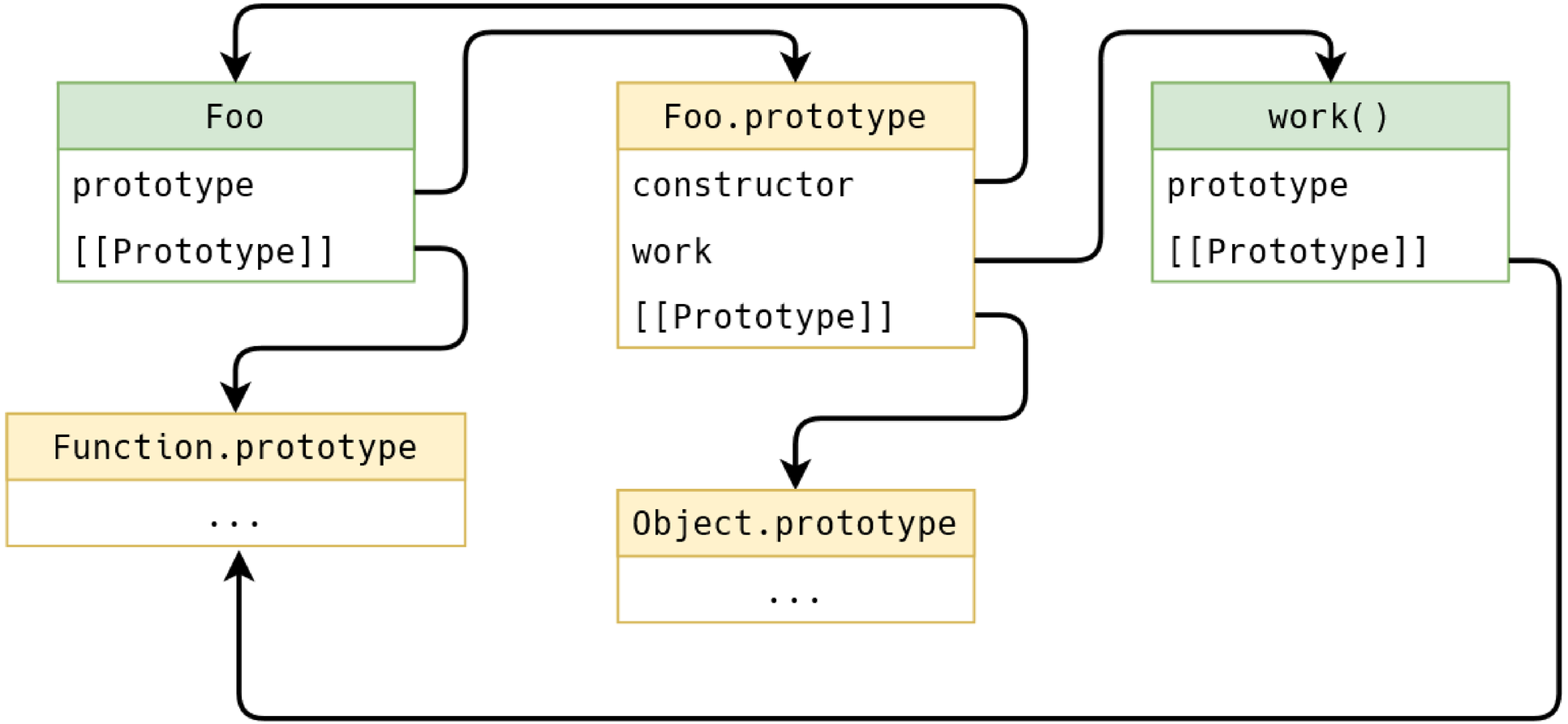
```
class Foo {
  constructor()
}
```

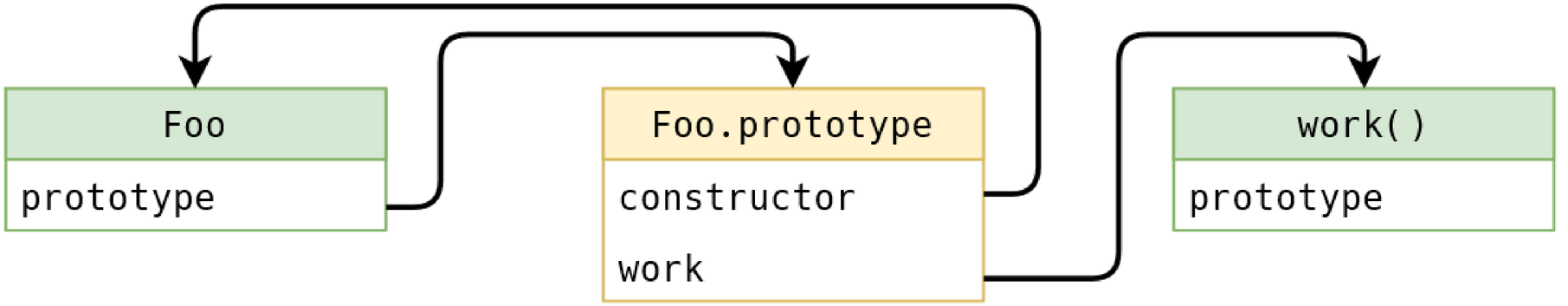


```
class Foo {
  constructor()
  work()
}
```

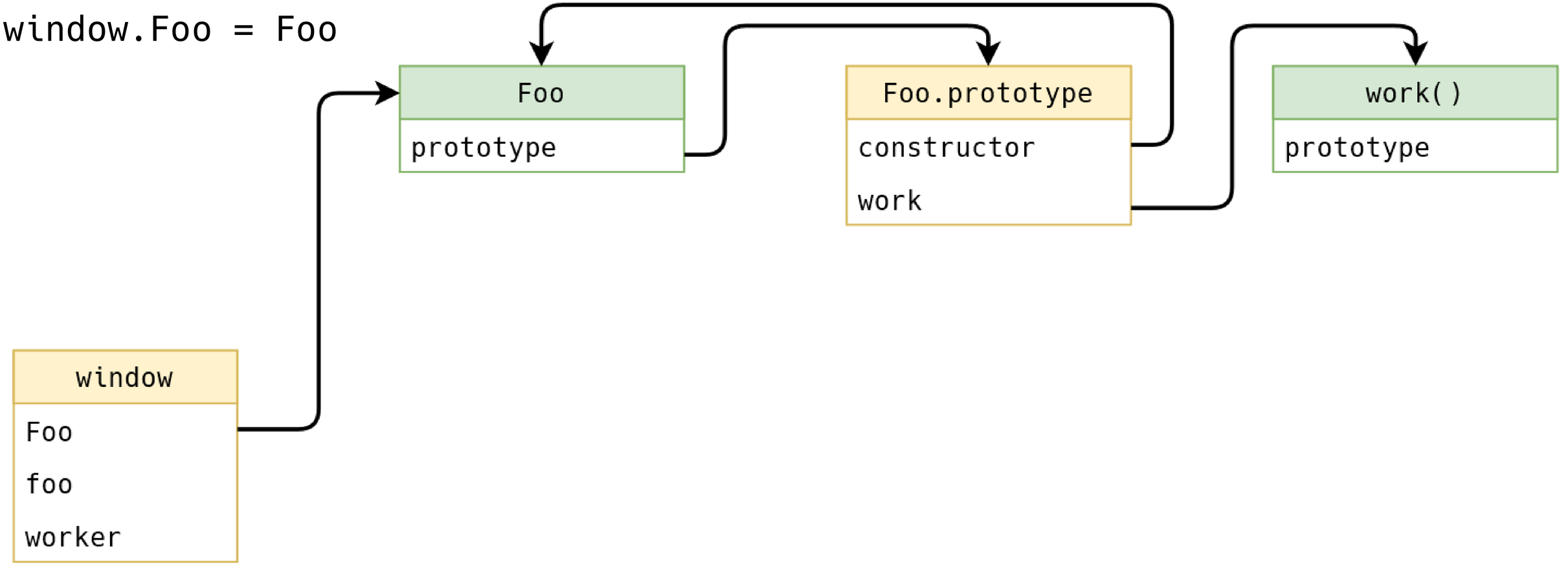


```
class Foo {
  constructor()
  work()
}
```

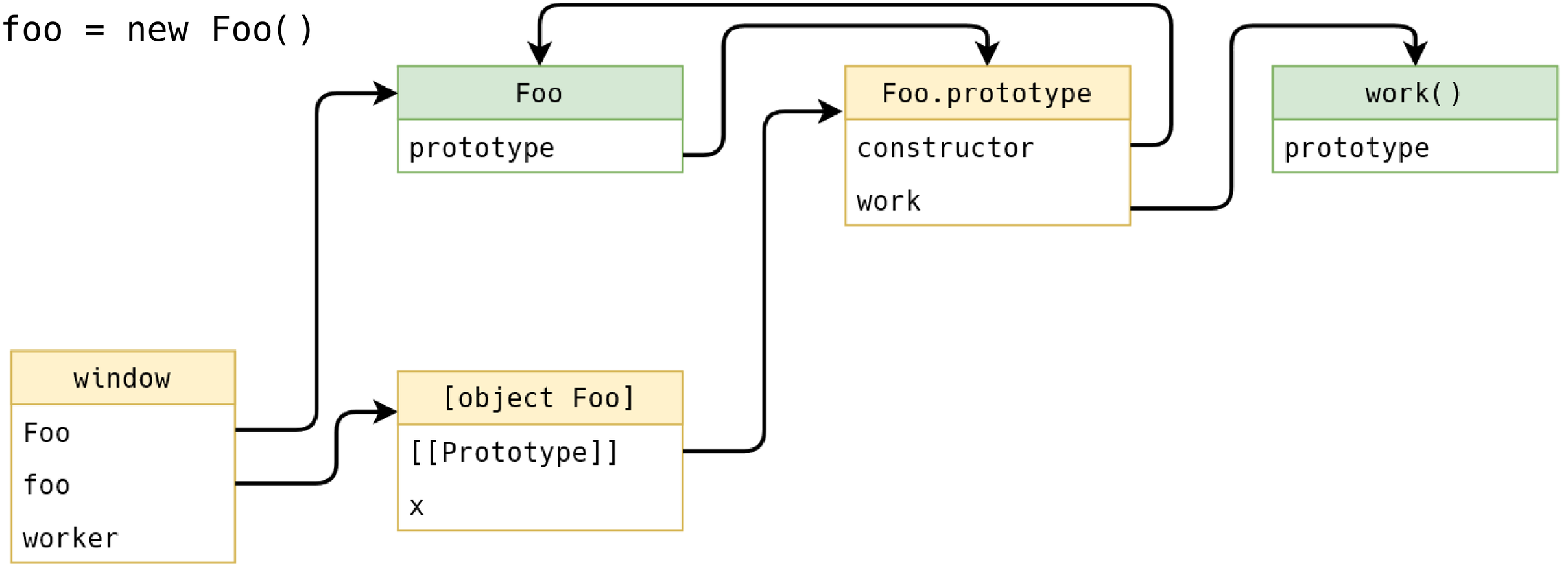


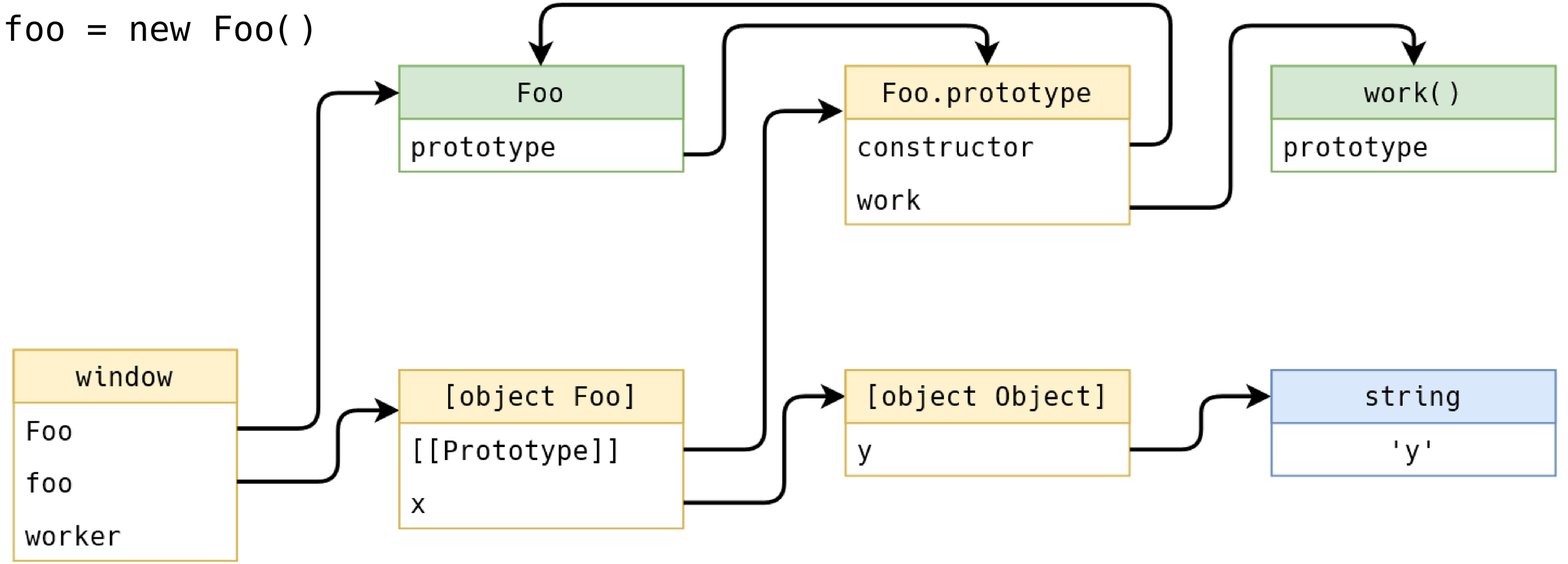

Foo
prototype
Foo.prototype
constructor
work
work()
prototype

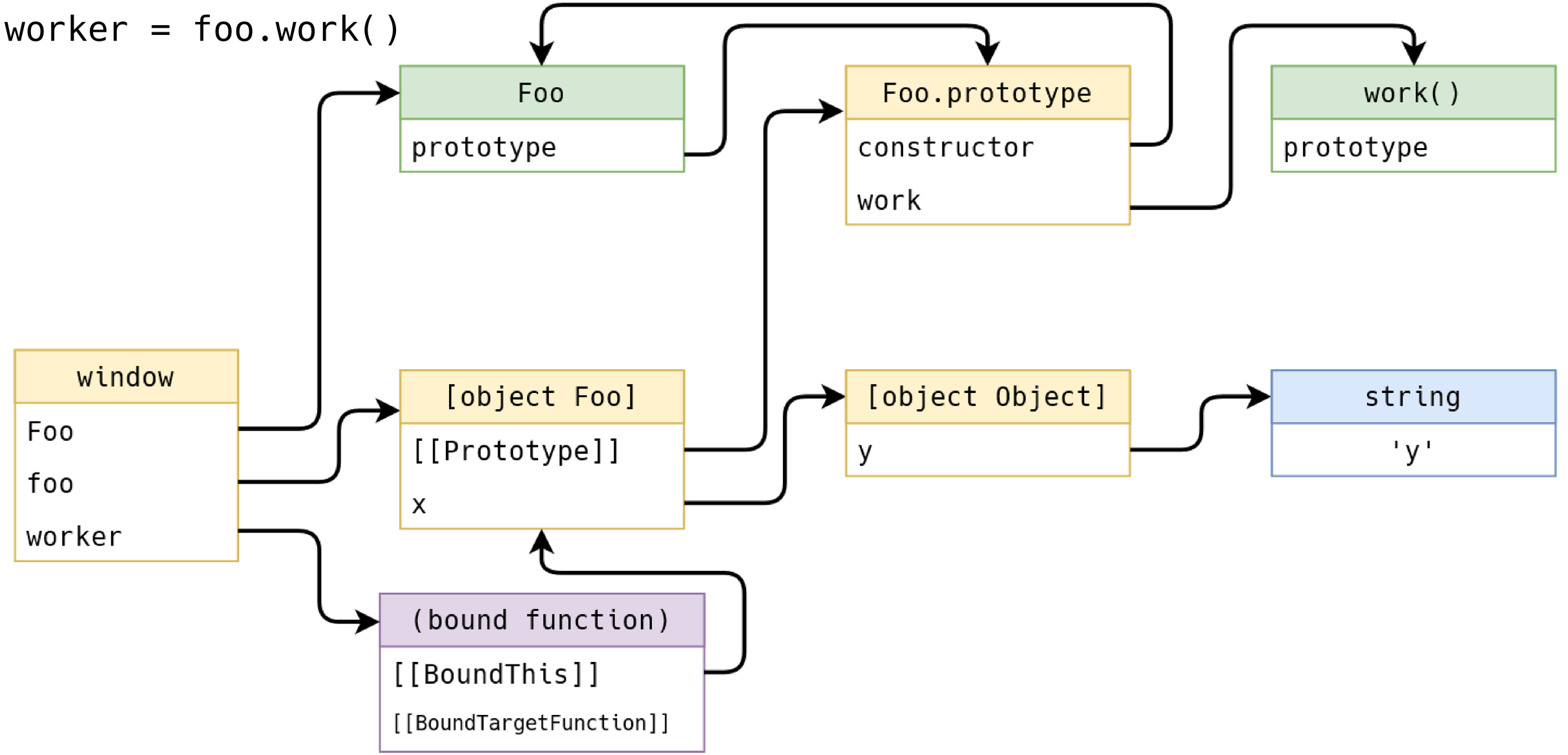
window.Foo = Foo



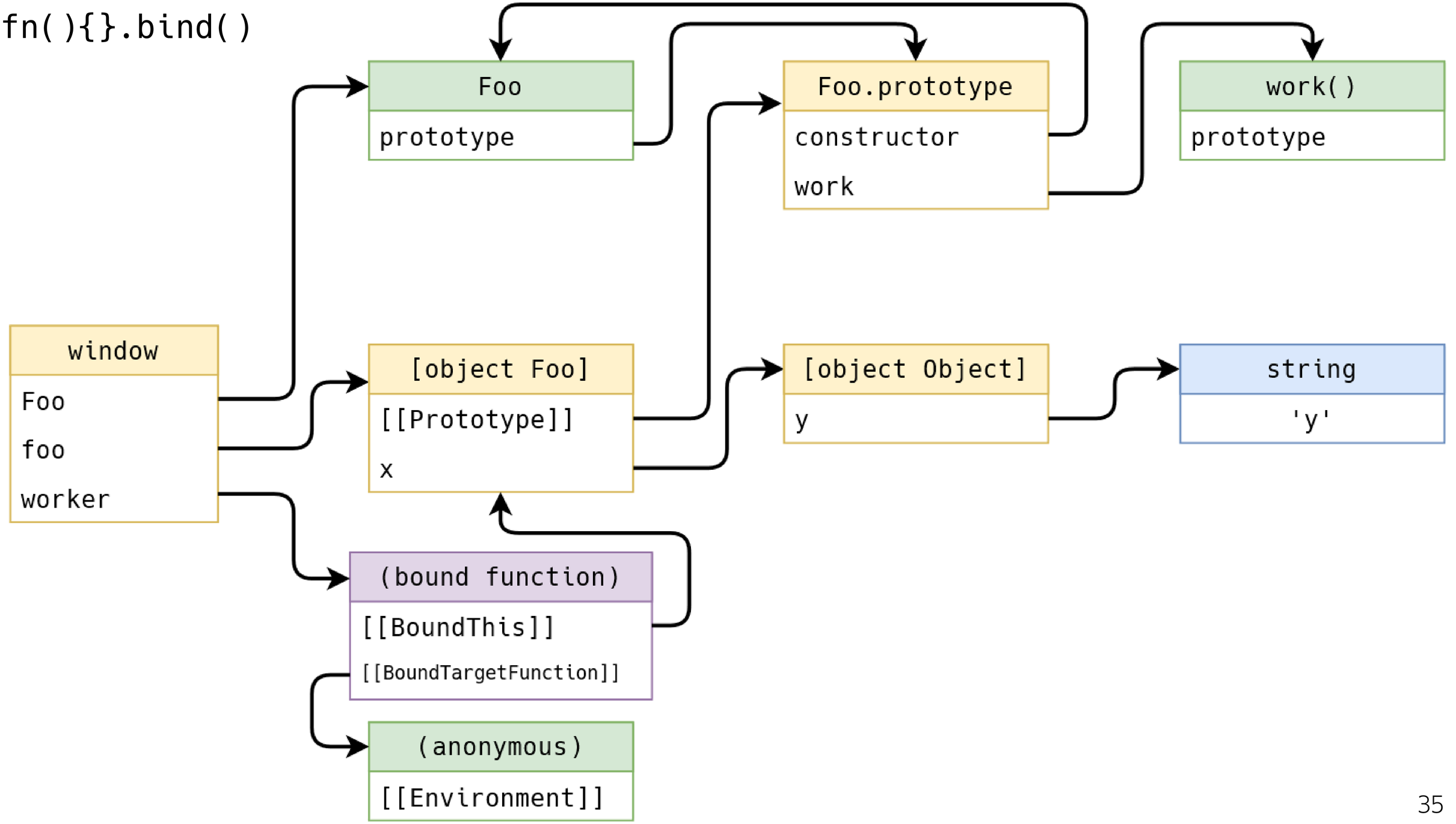
```
foo = new Foo()
```


Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x

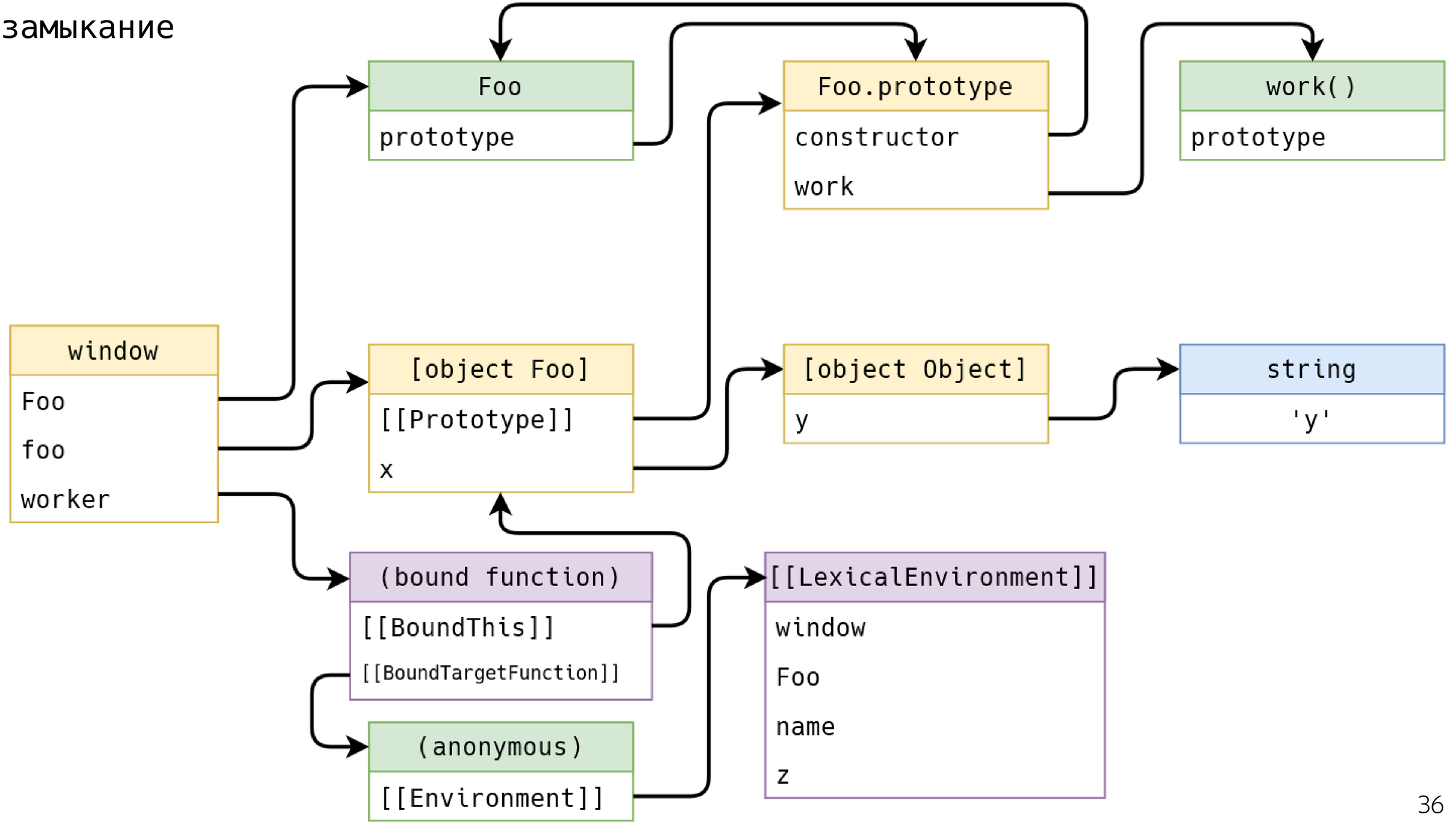

foo = new Foo()
Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'


worker = foo.work()
Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]

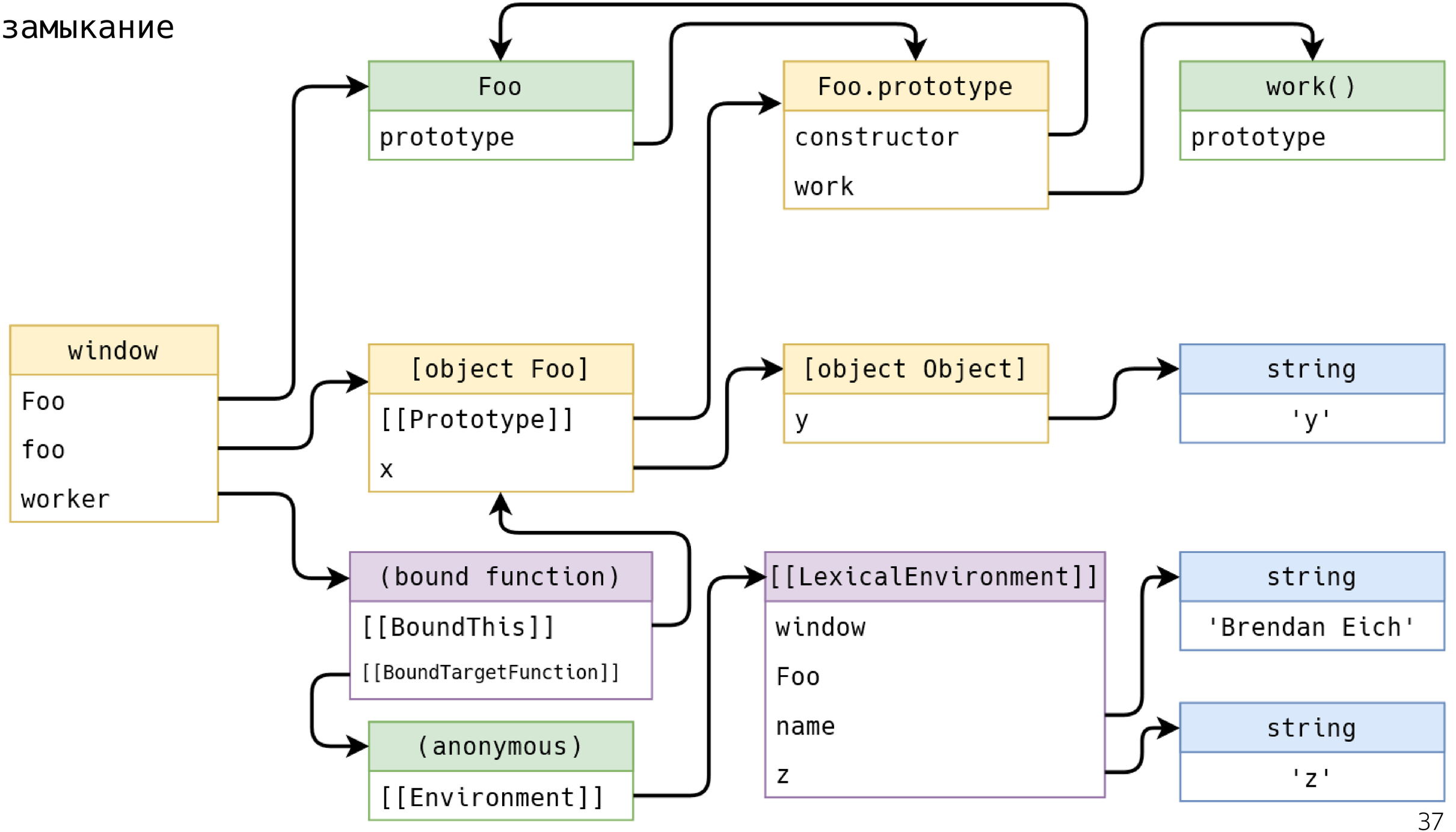
fn(){}.bind()


Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]

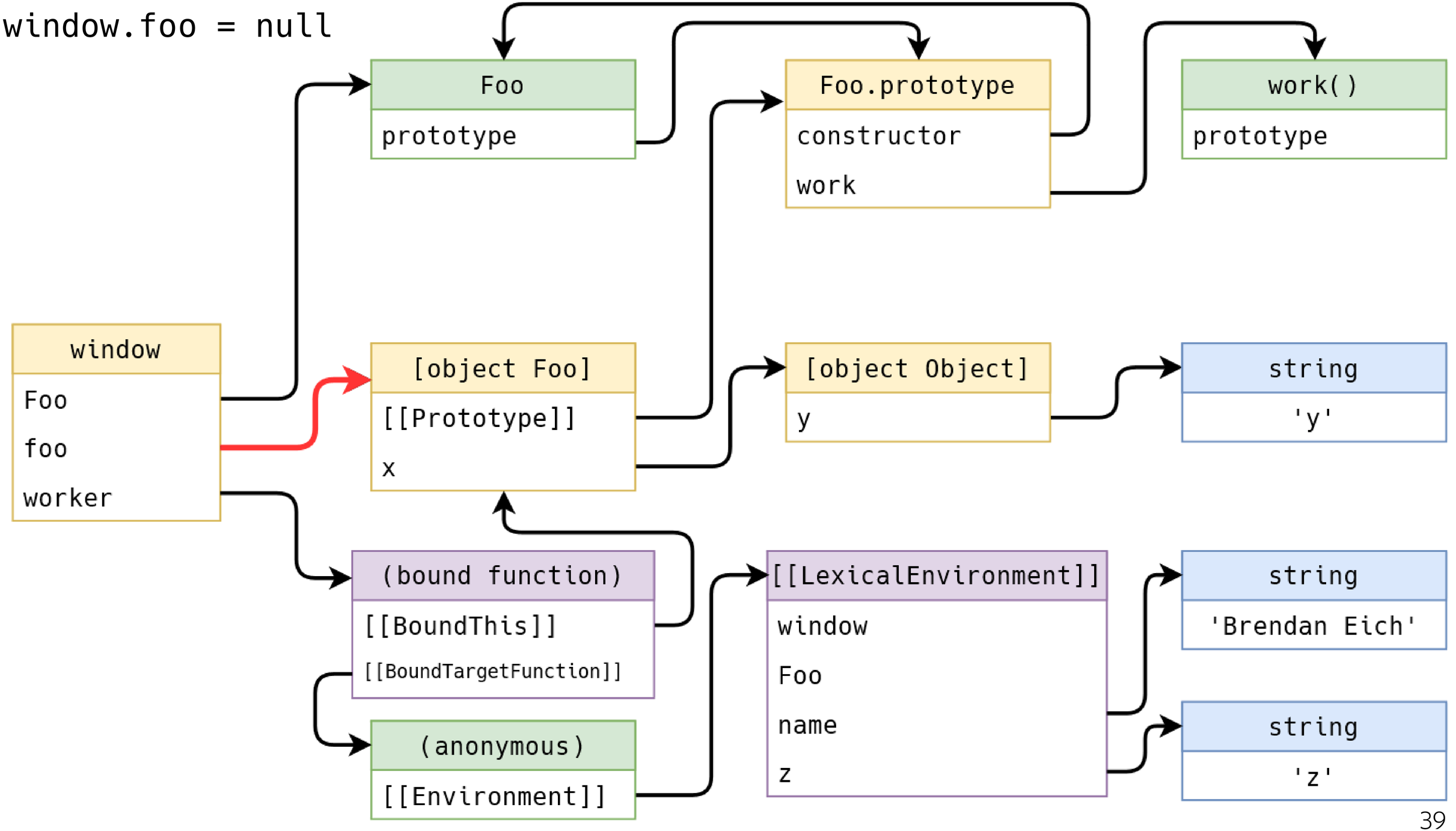
замыкание


Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z

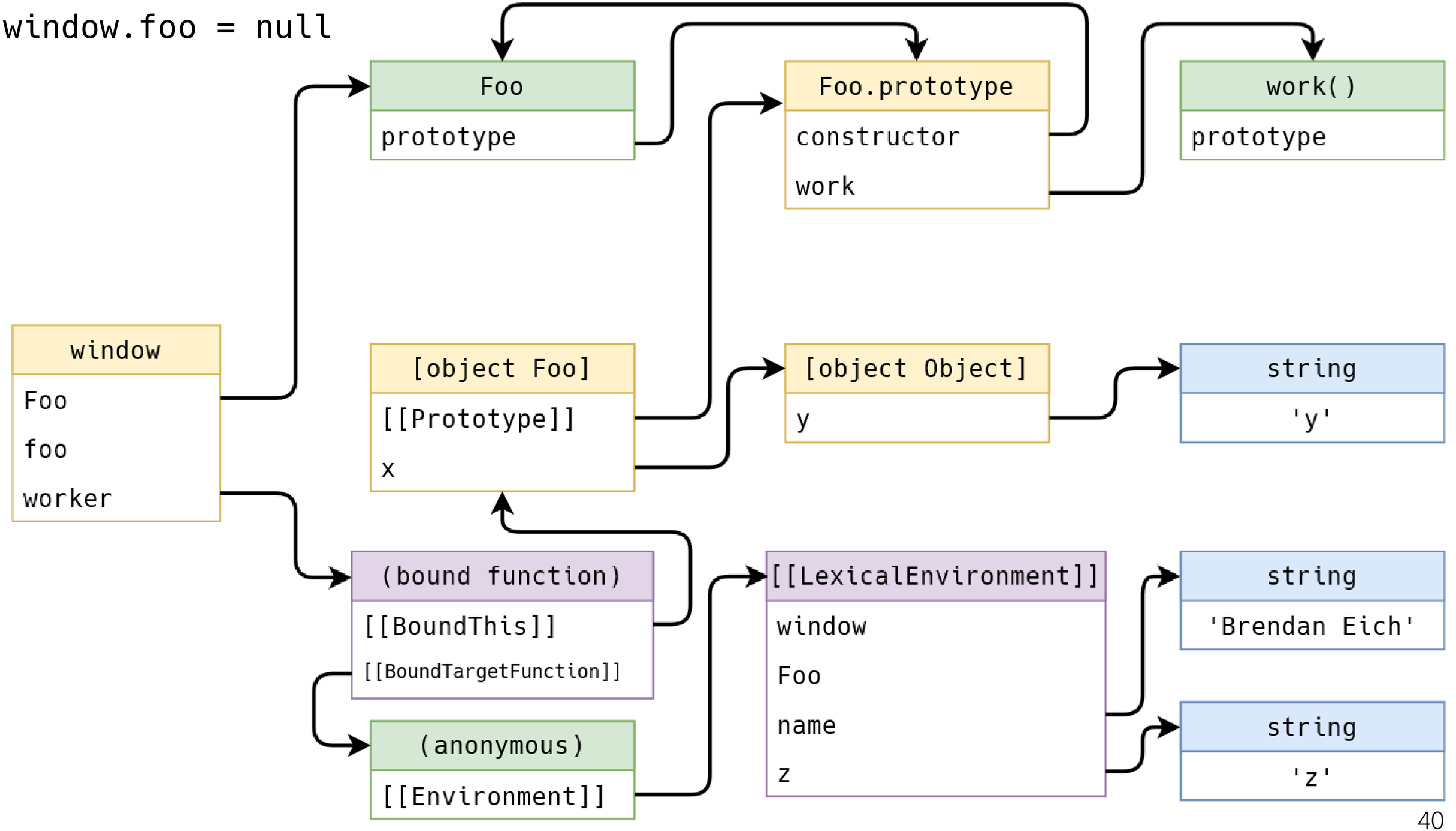
замыкание

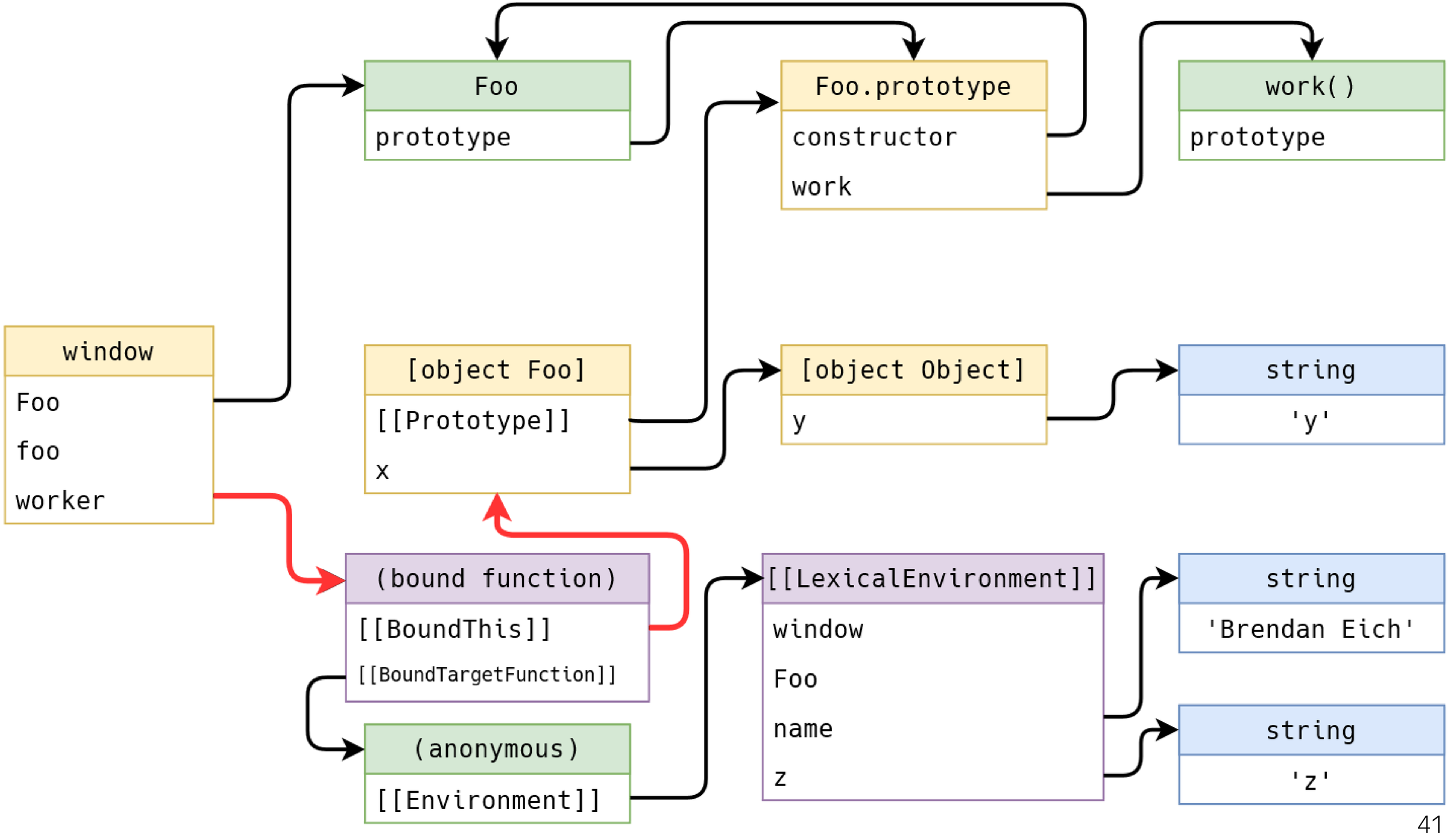

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

window.foo = null



window.foo = null

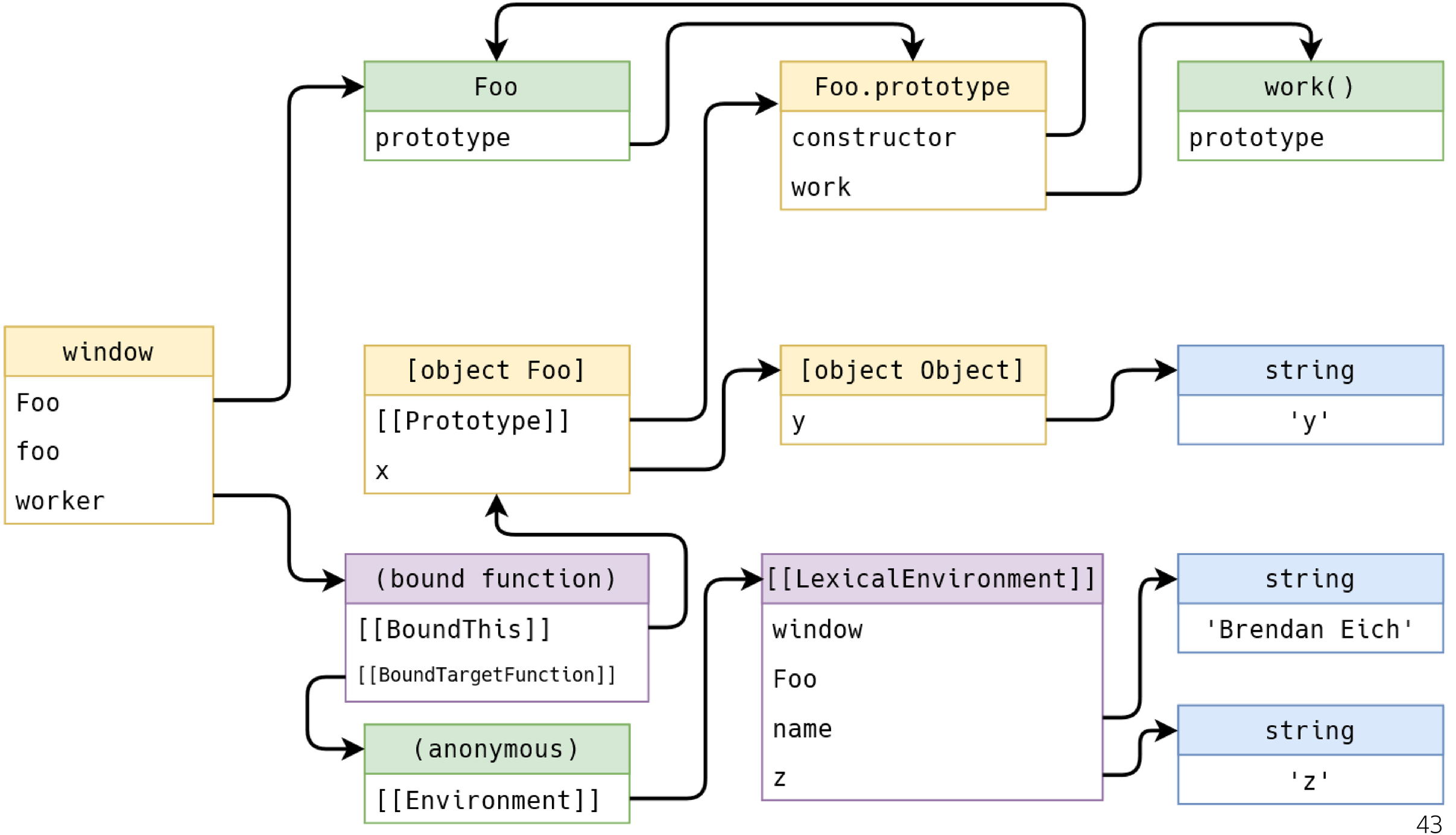



Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

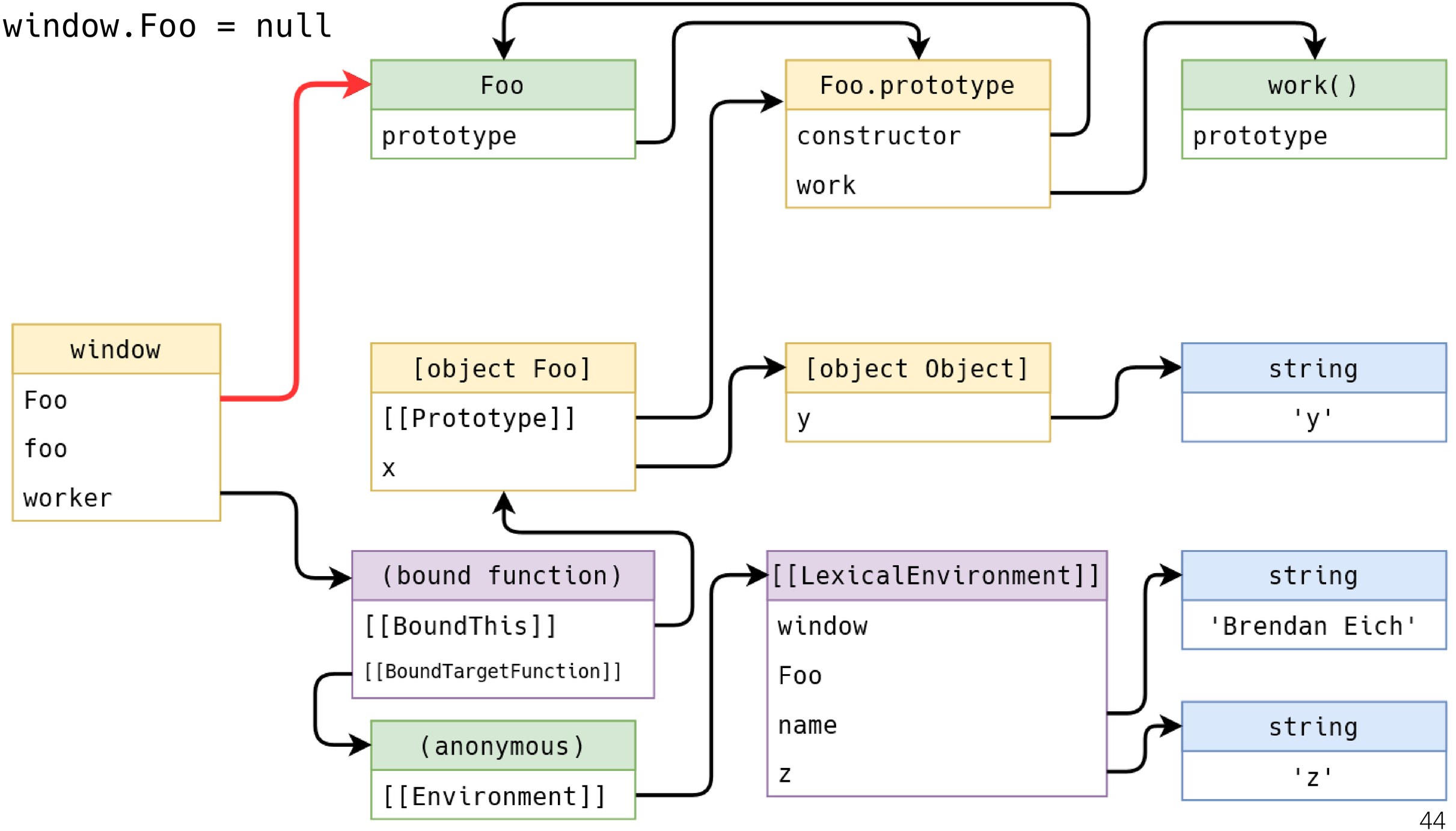
Типичная ошибка: забытая подписка

```
externalElement.addEventListener('click', () => {
  if (this.shouldDoSomethingOnClick) {
    this.doSomething();
  }
})
```

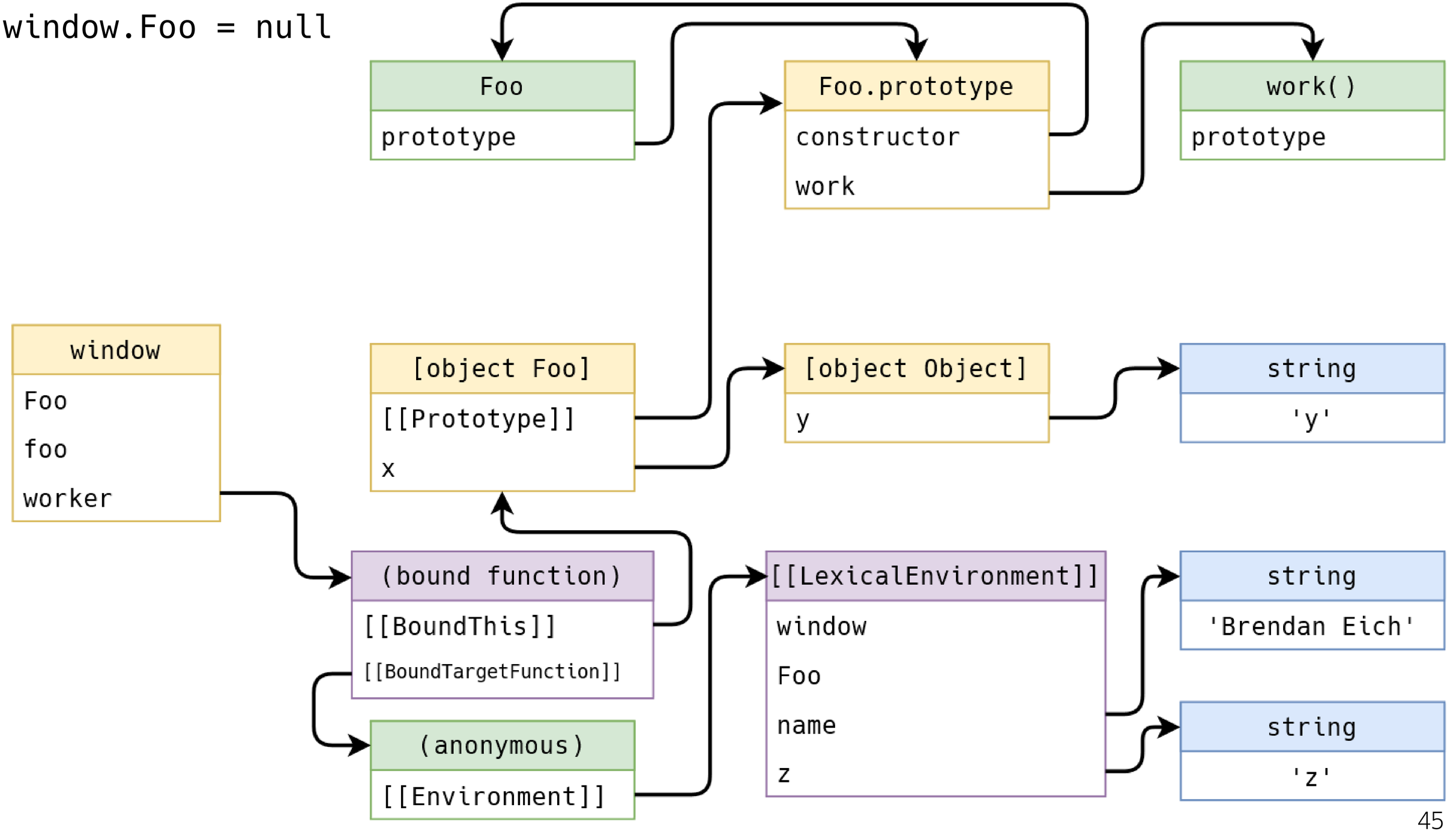
- Отписывайтесь
- Продумайте время жизни подписки и кто ей владеет
- Зануляйте ссылки (whatever = null)
- Используйте WeakMap

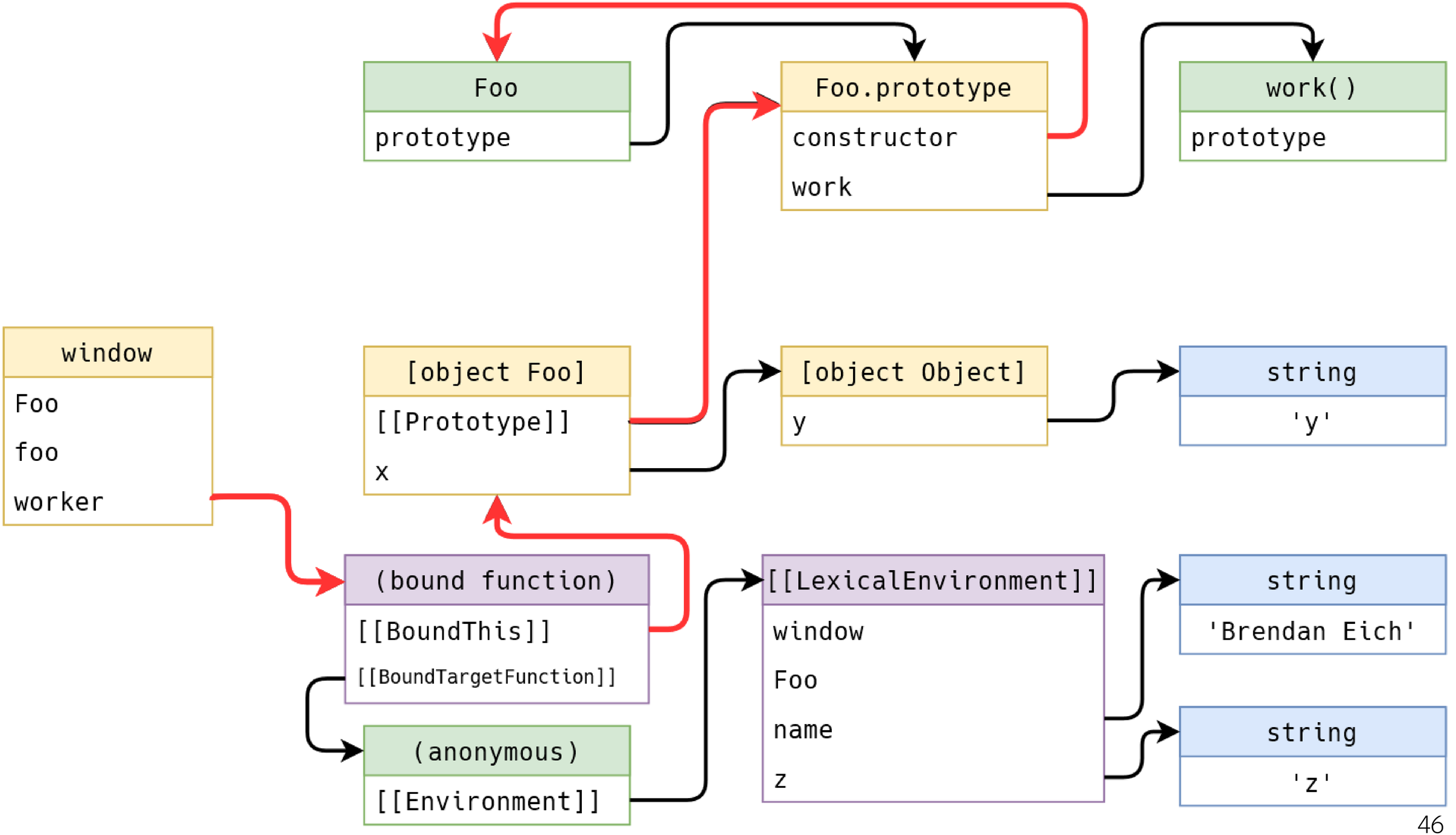

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

window.Foo = null



window.Foo = null




Foo
prototype
Foo.prototype
constructor
work
work( )
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

Типичная ошибка

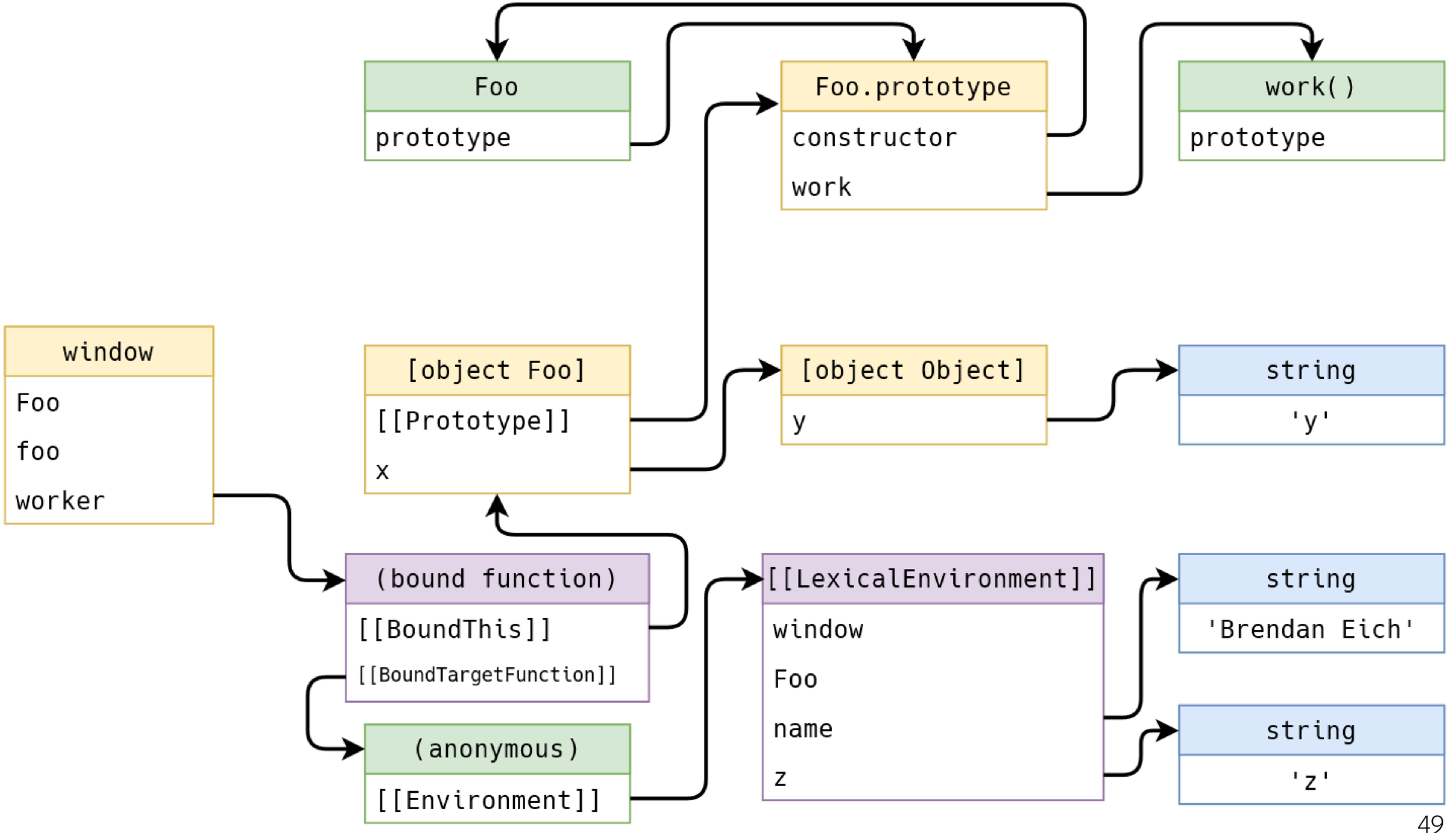
```
destroy() {
  this._x = null;
  this._y = null;
  // еще 10 this._foobar = null
}
```

- Если объект состоит из ссылок на другие объекты, то никакой destroy() вам, скорее всего, не нужен
- Зануляйте когда надо, а когда не надо – не зануляйте

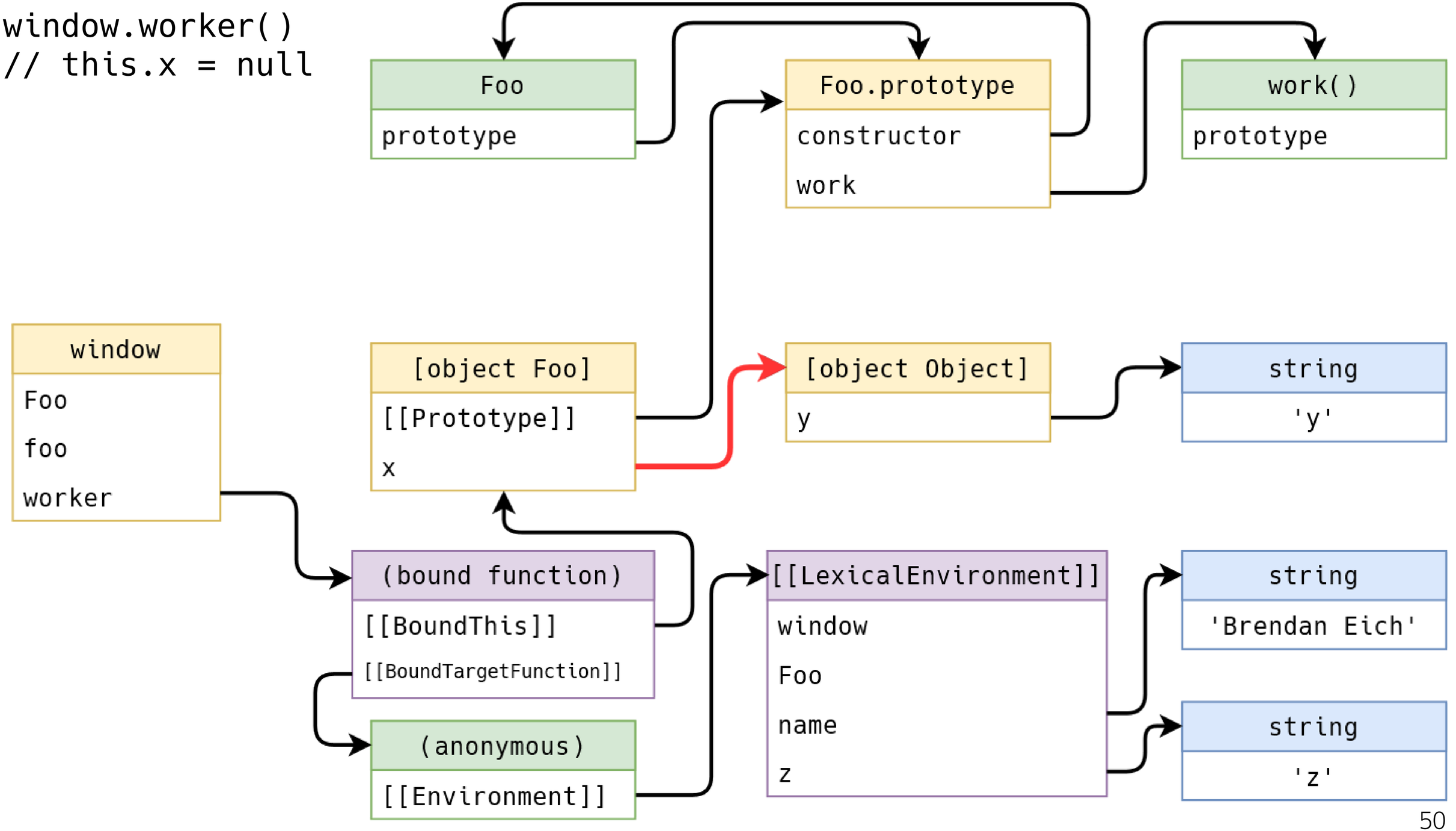
Типичная ~~ошибка~~ бесполезная работа

```
destroy() {
  this._x = null;
  this._y = null;
  // еще 10 this._foobar = null
}
```

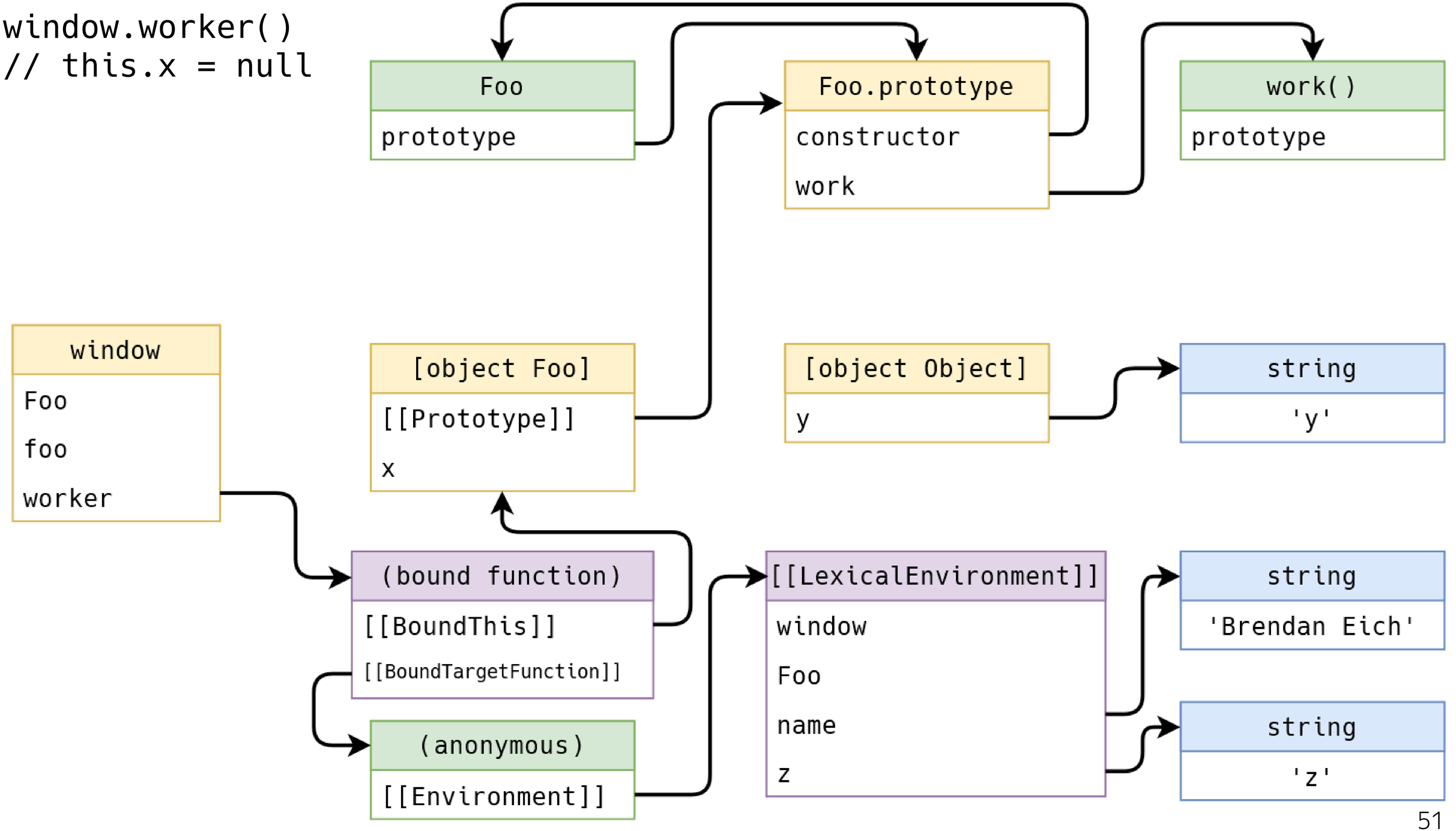
- Если объект состоит из ссылок на другие объекты, то никакой destroy() вам, скорее всего, не нужен
- Зануляйте когда надо, а когда не надо – не зануляйте

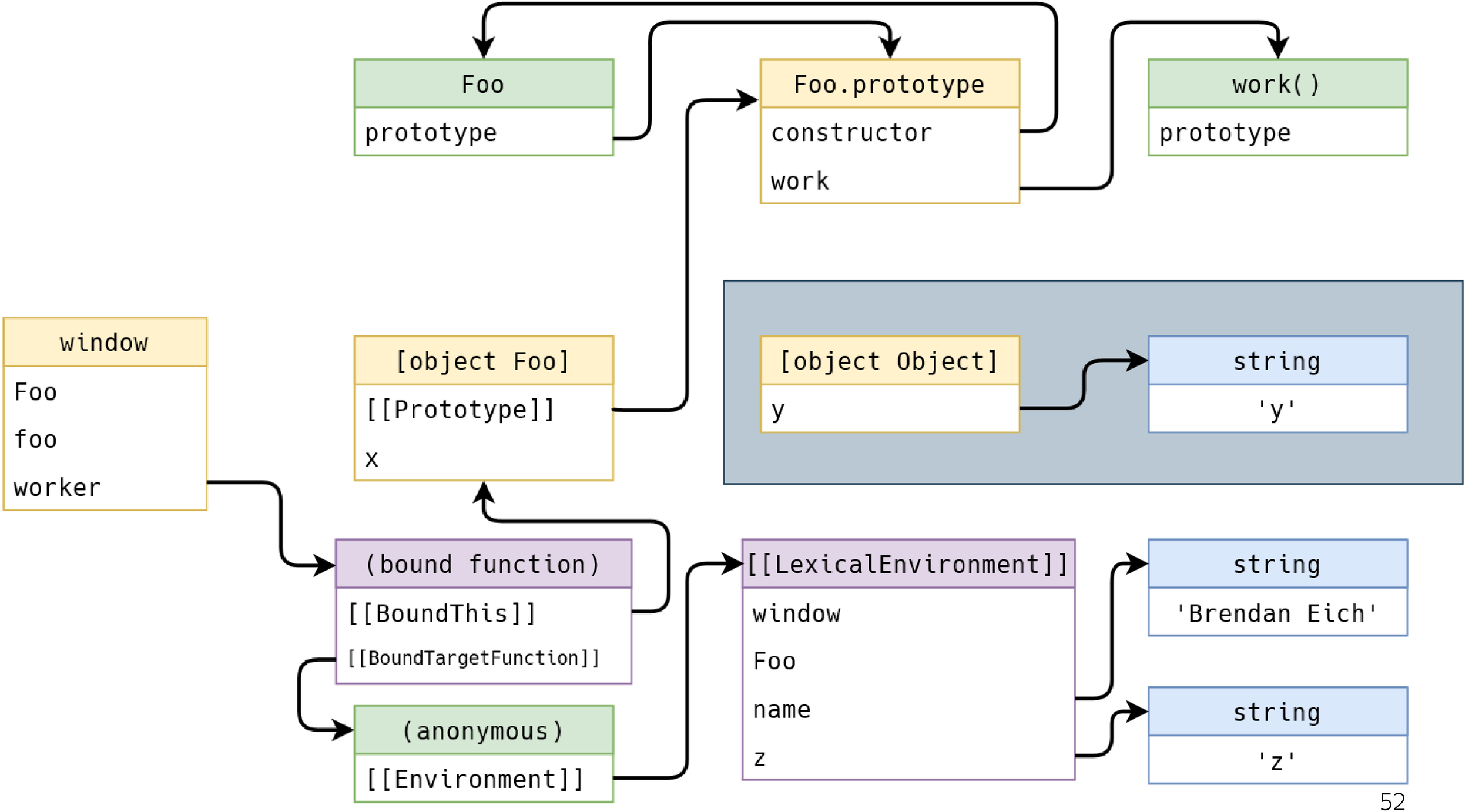

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

```
window.worker()
// this.x = null
```



```
window.worker()
// this.x = null
```

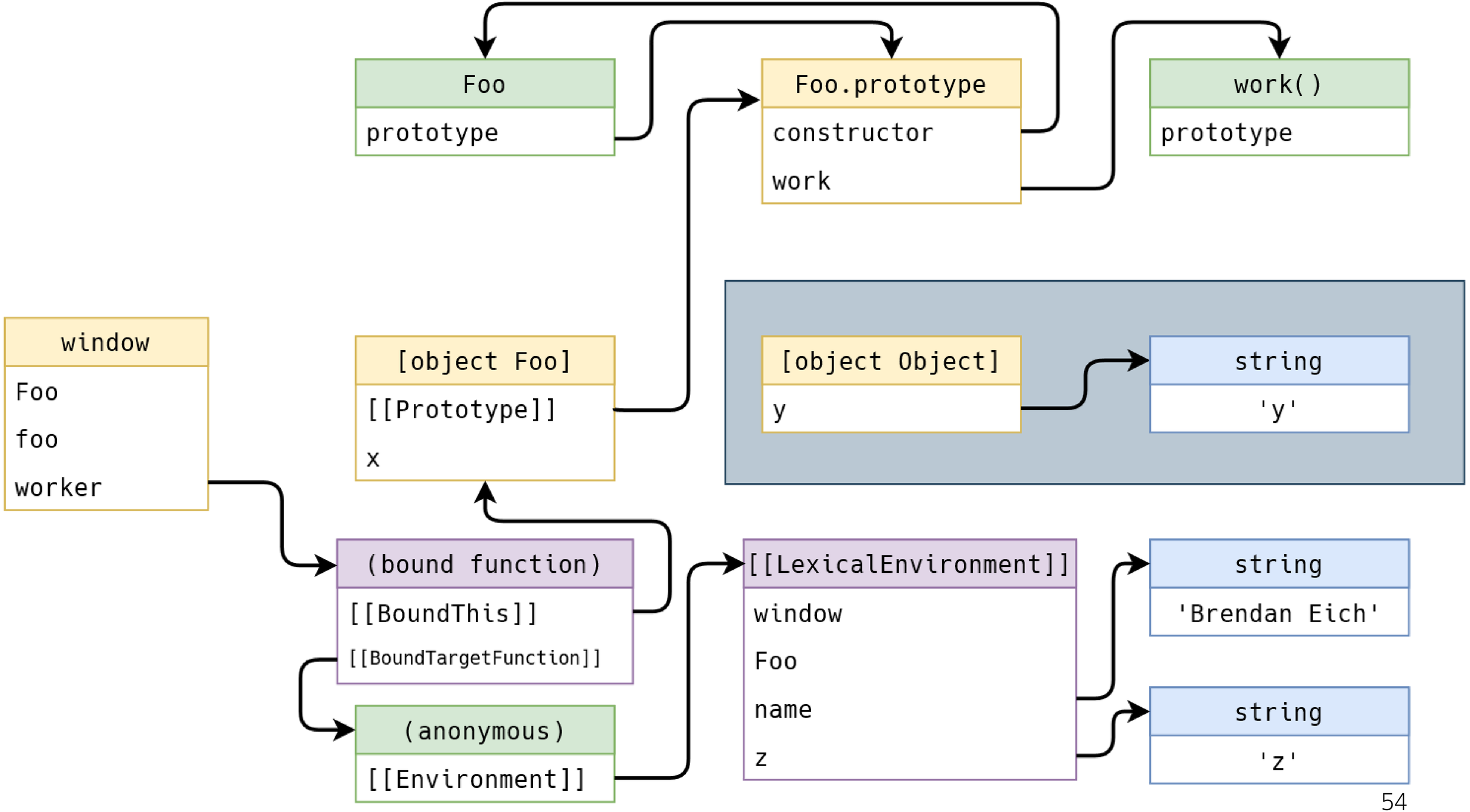


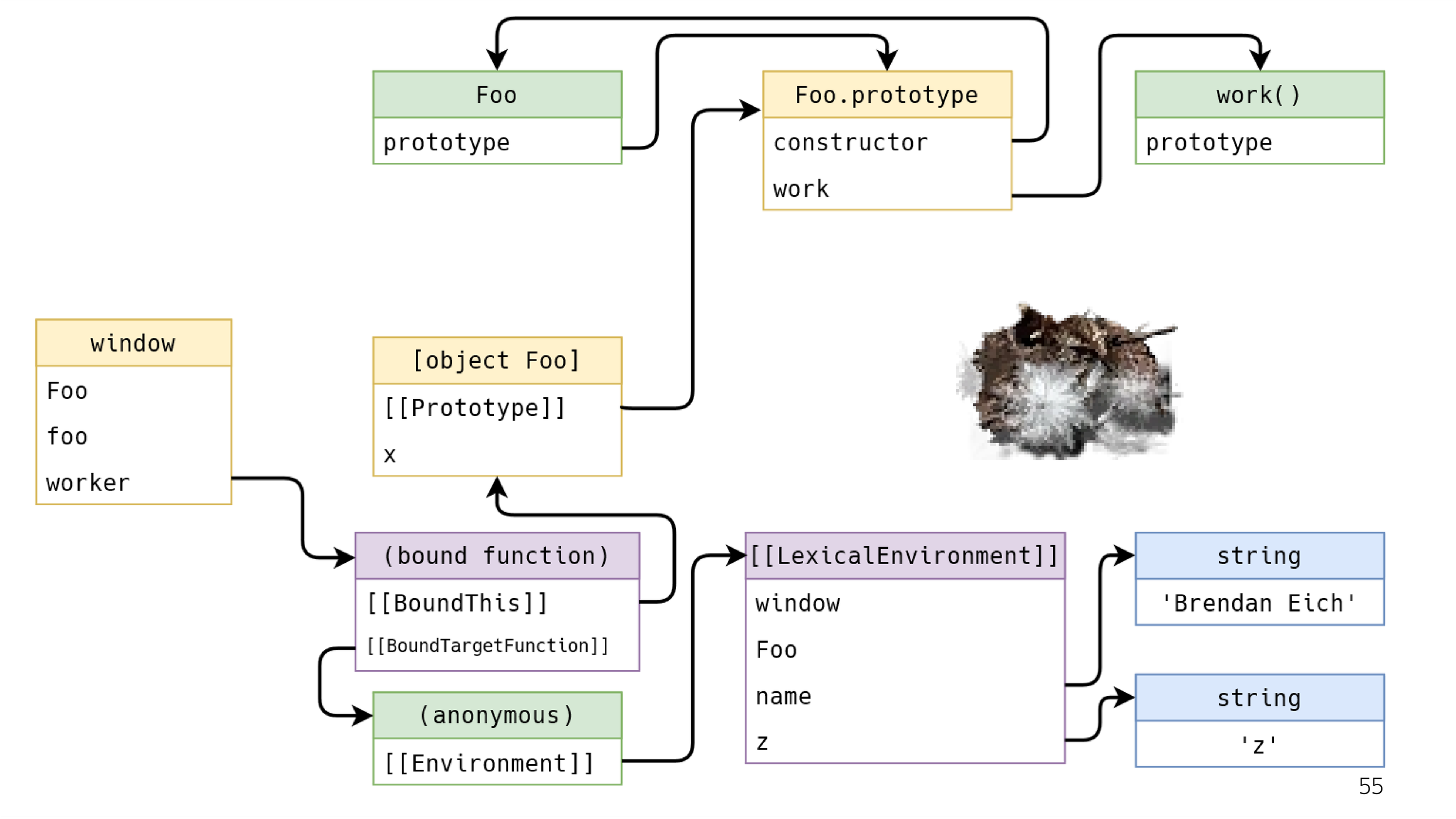

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

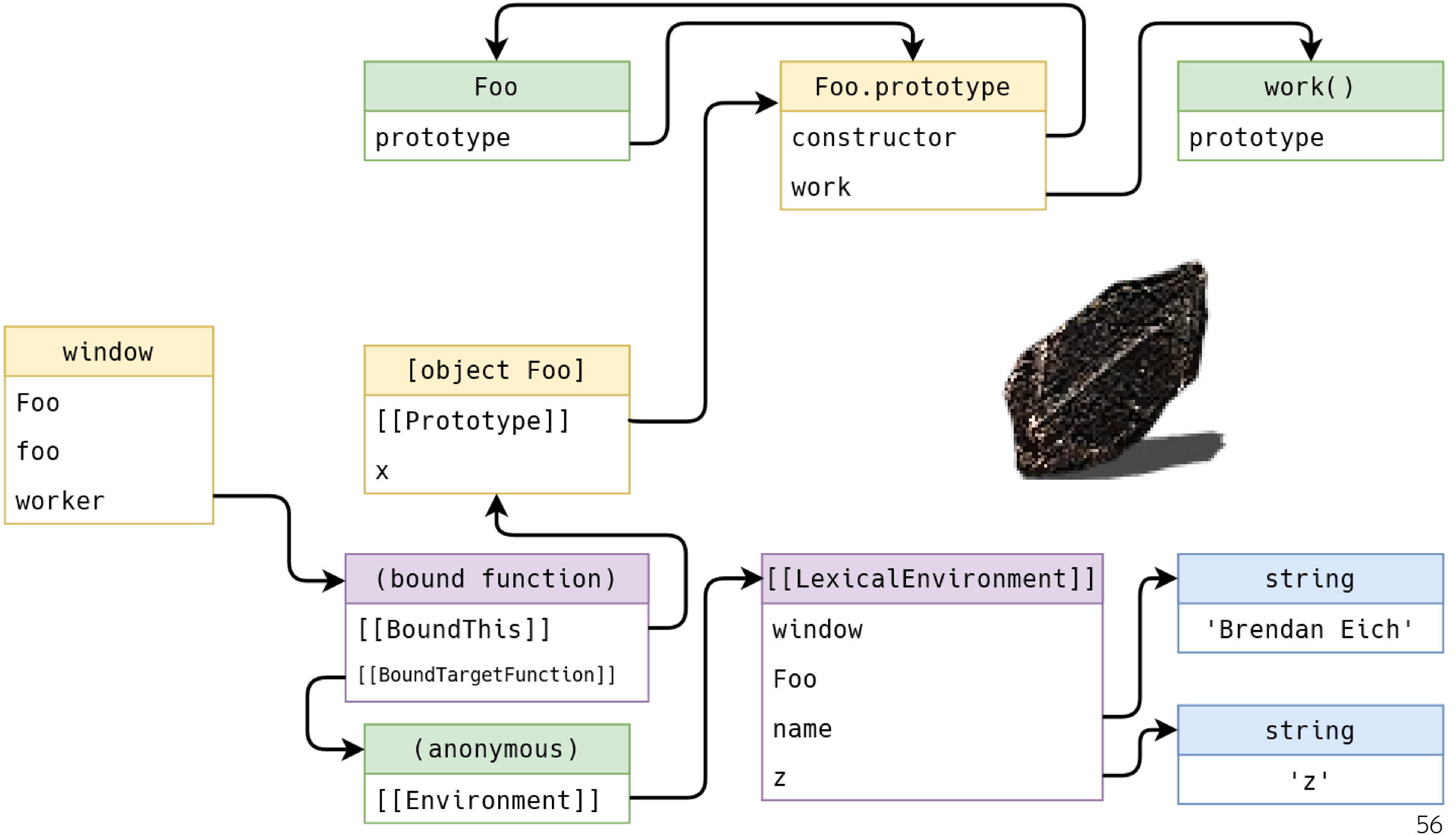
Мусор, который не собирается

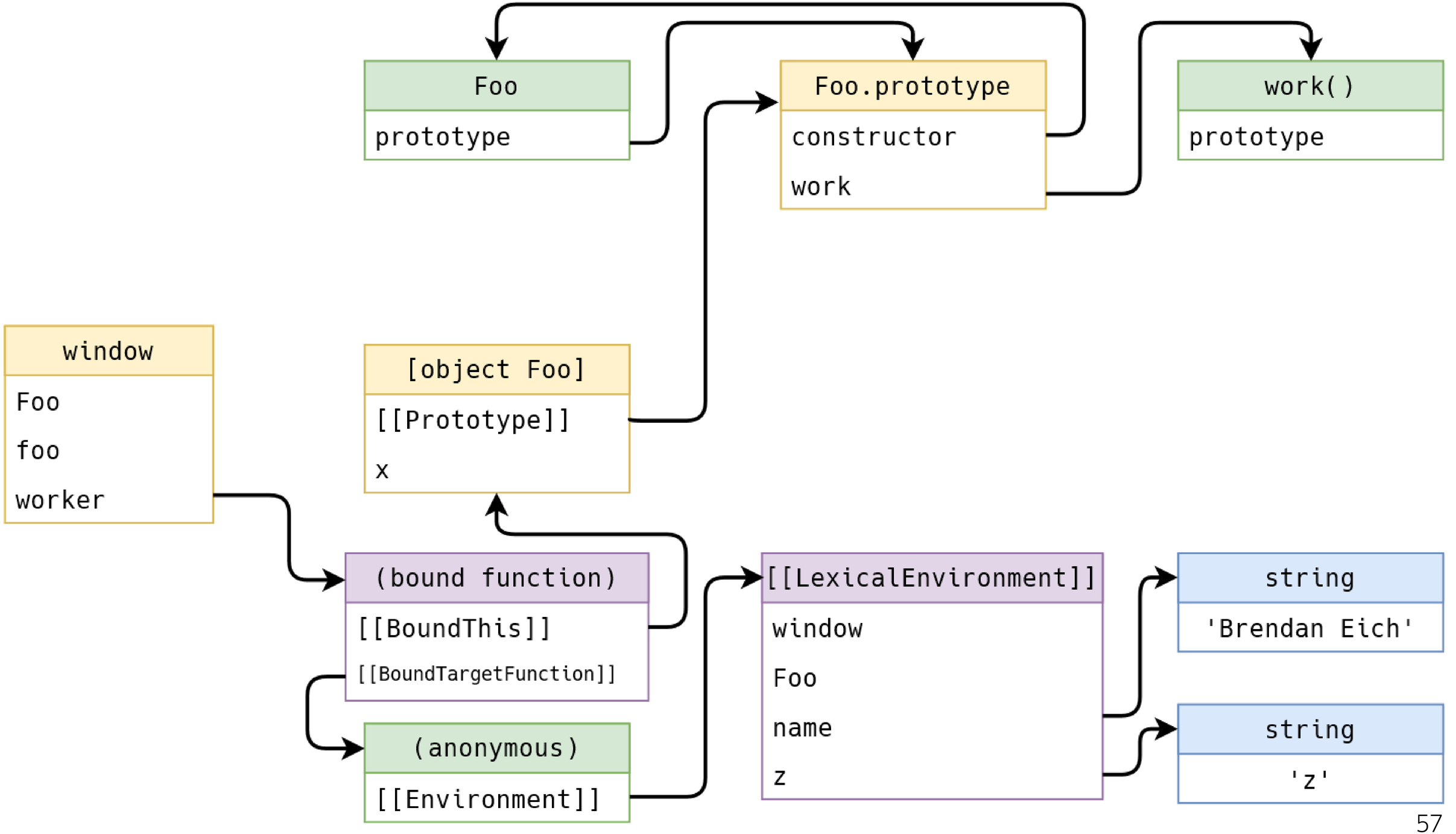
Объекты с парными функциями create/delete:

- createObjectURL(), revokeObjectURL()
- WebGL: create/delete Program/Shader/Buffer/Texture/etc
- ImageBitmap.close()
- indexDb.close()
- ...

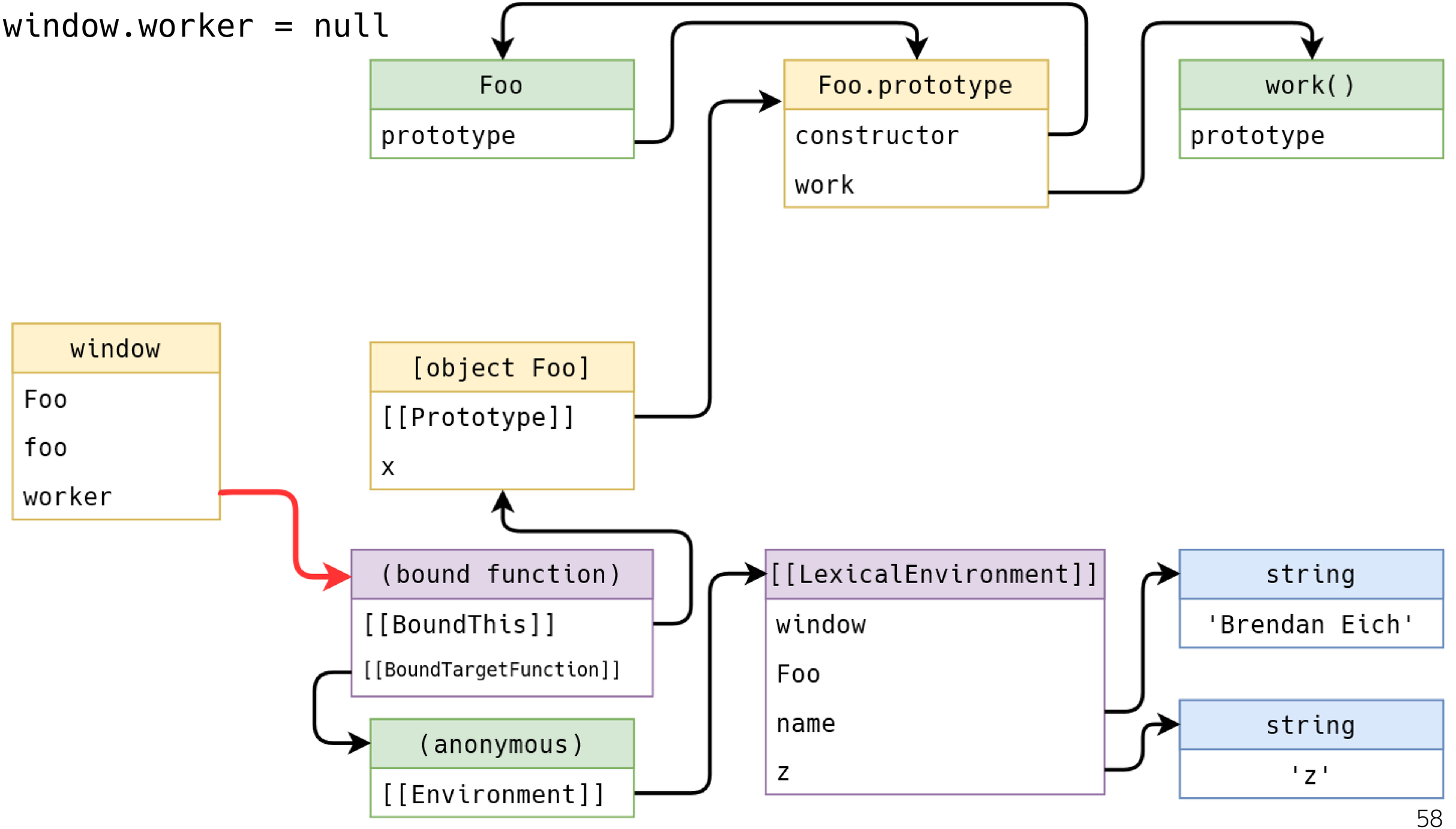

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
[object Object]
y
string
'y'
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'


Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

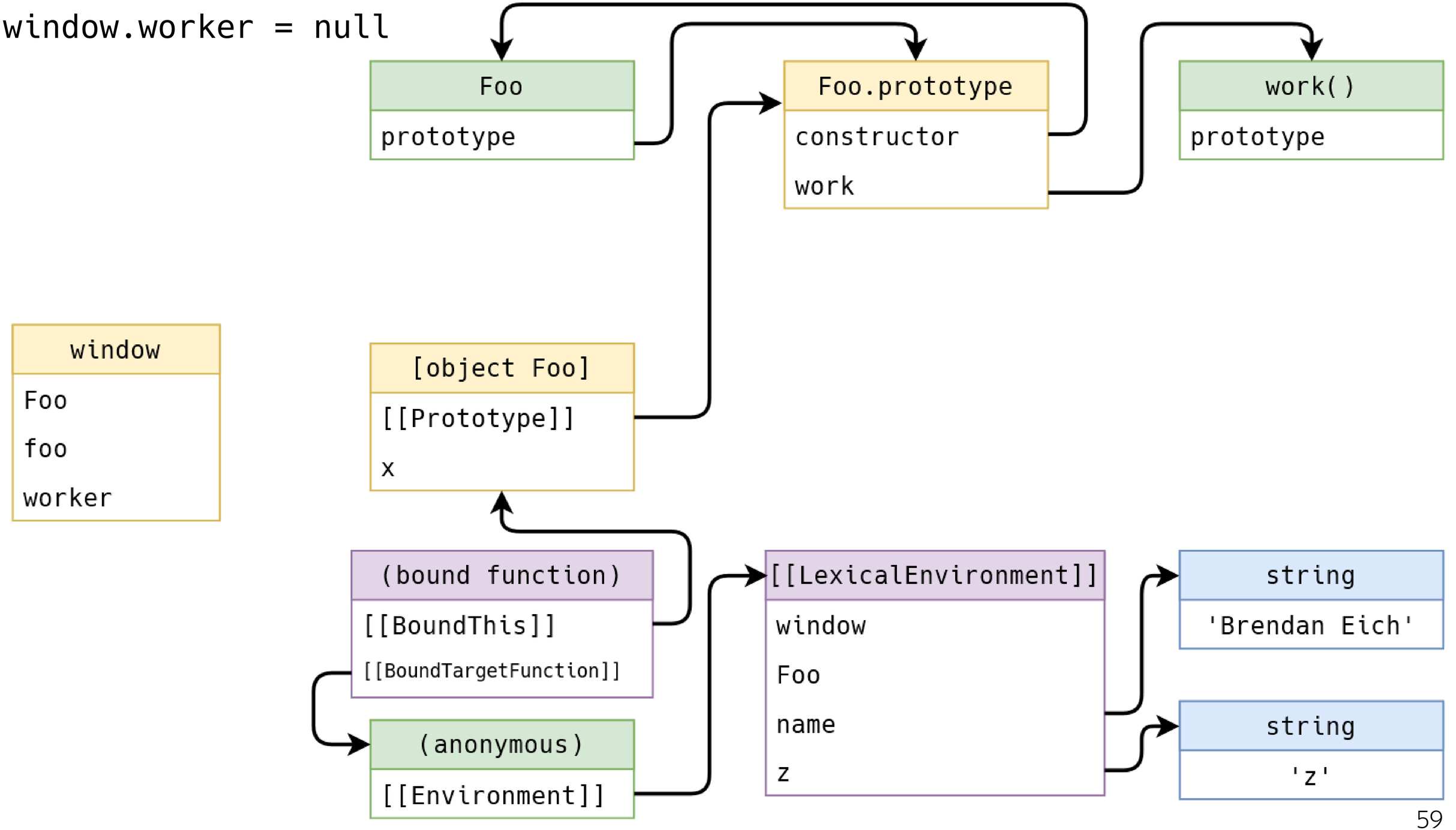

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

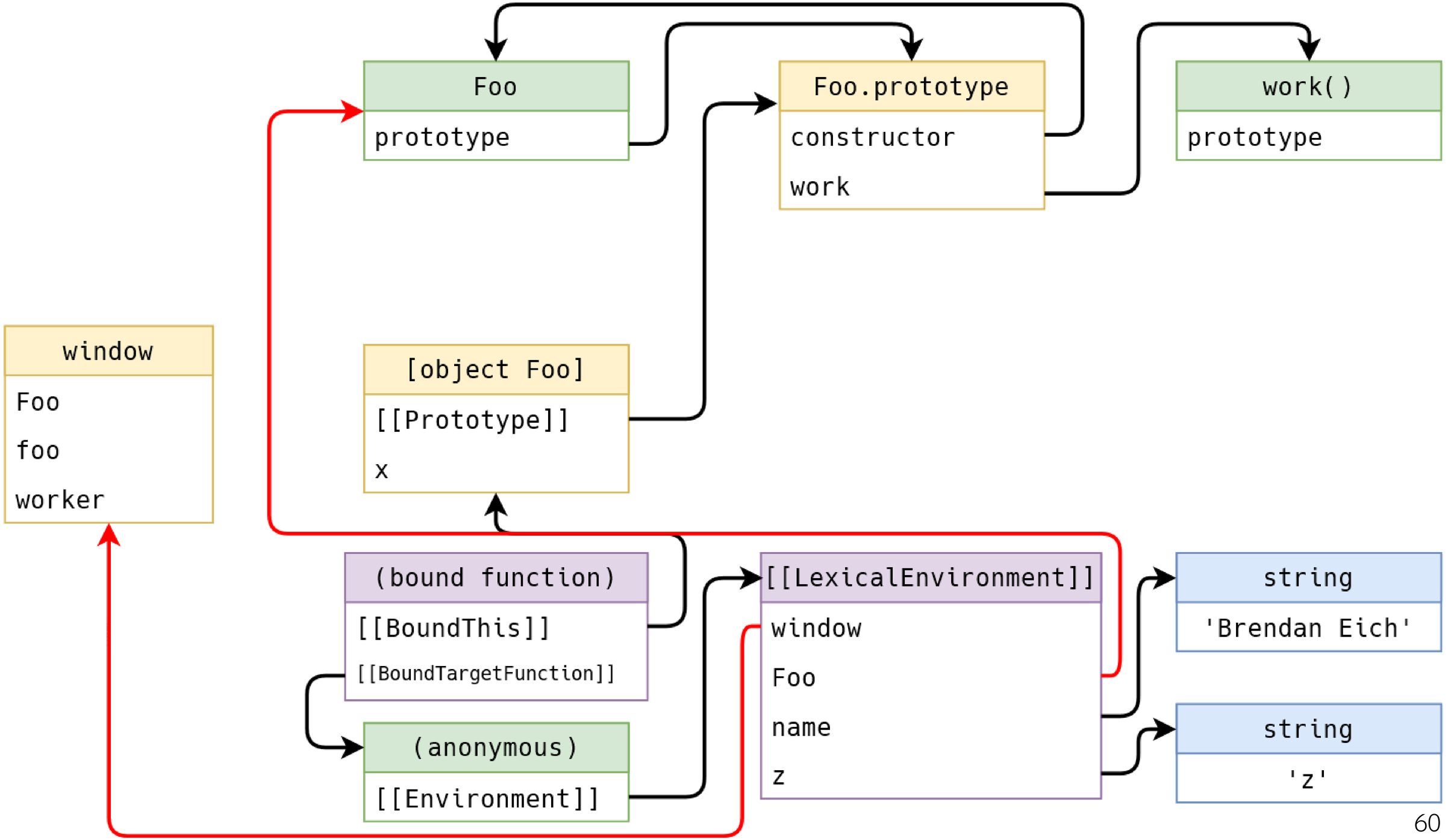

Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

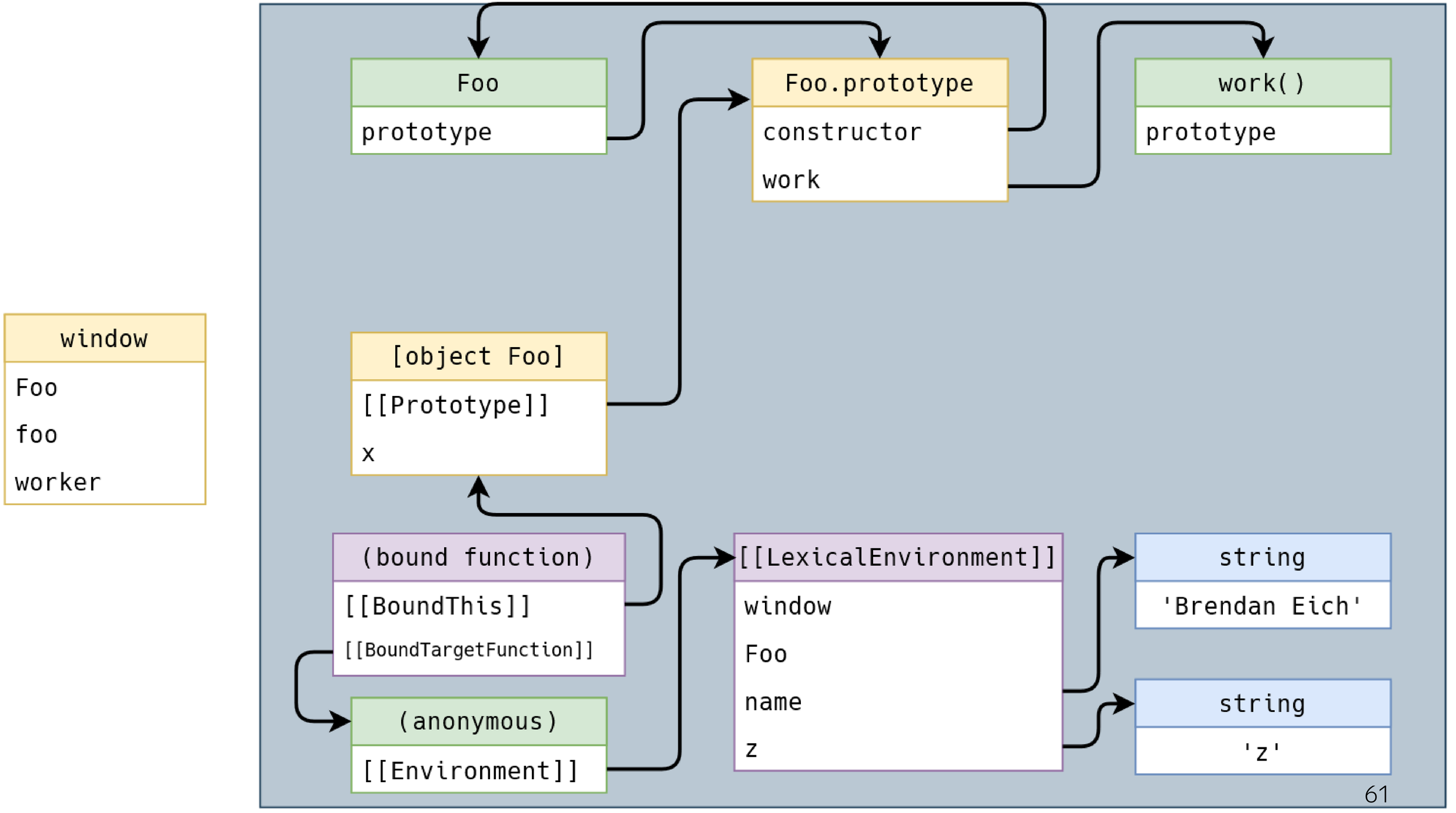
window.worker = null



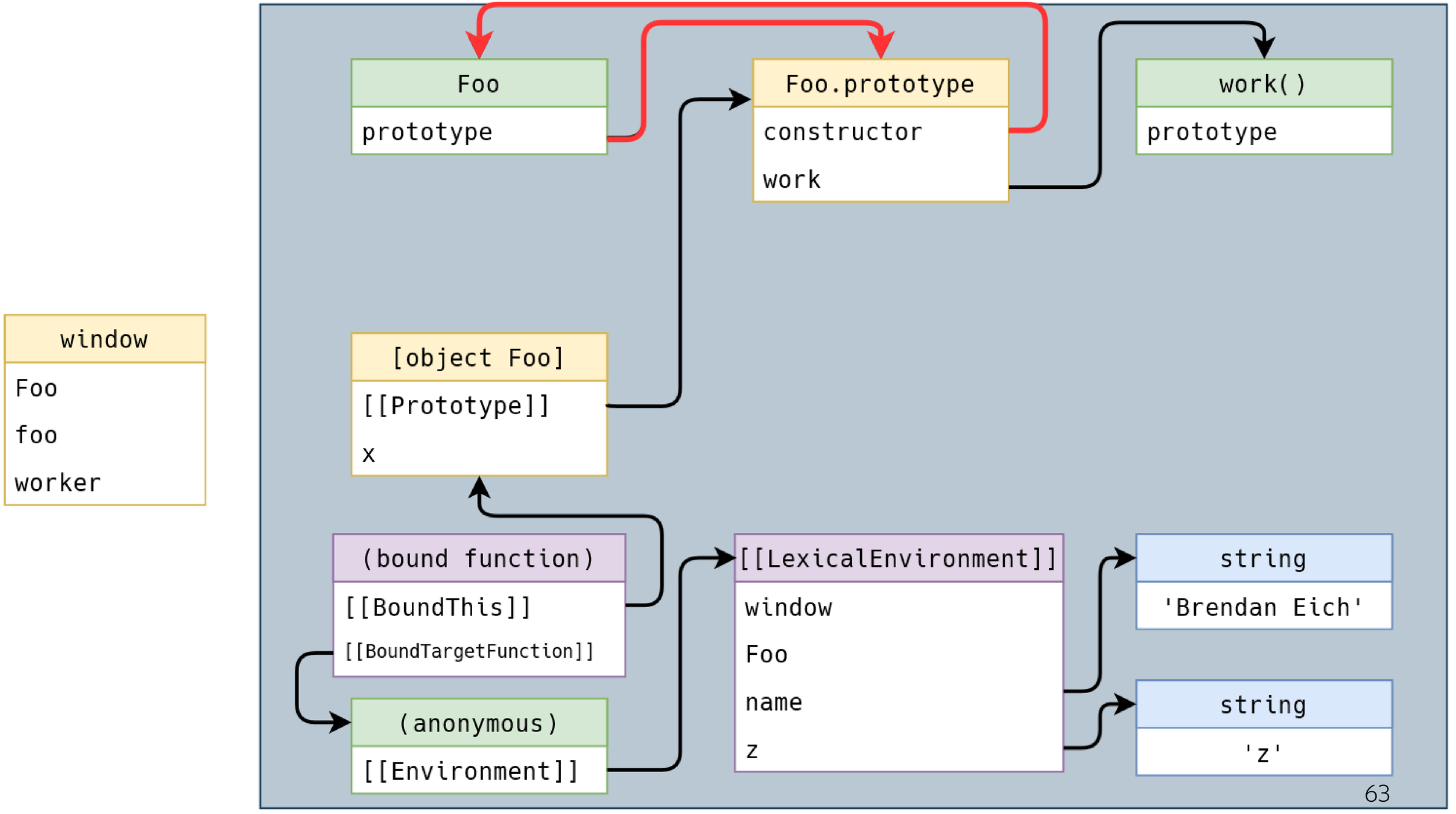
window.worker = null




Foo
prototype
Foo.prototype
constructor
work
work()
prototype
window
Foo
foo
worker
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'


window
Foo
foo
worker
Foo
prototype
Foo.prototype
constructor
work
work()
prototype
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

YOU DIED


window
Foo
foo
worker
Foo
prototype
Foo.prototype
constructor
work
work( )
prototype
[object Foo]
[[Prototype]]
x
(bound function)
[[BoundThis]]
[[BoundTargetFunction]]
(anonymous)
[[Environment]]
[[LexicalEnvironment]]
window
Foo
name
z
string
'Brendan Eich'
string
'z'

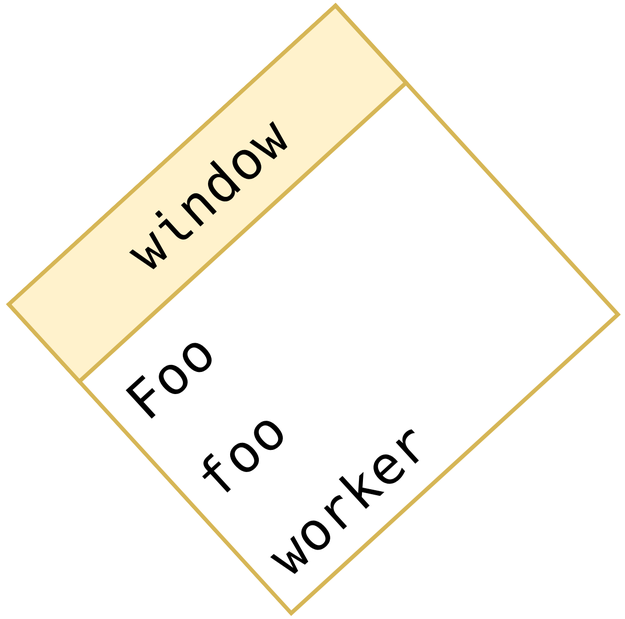

TRASH
REPUBLIC
SERVICES
RepublicServices.com

Что значит мусор?

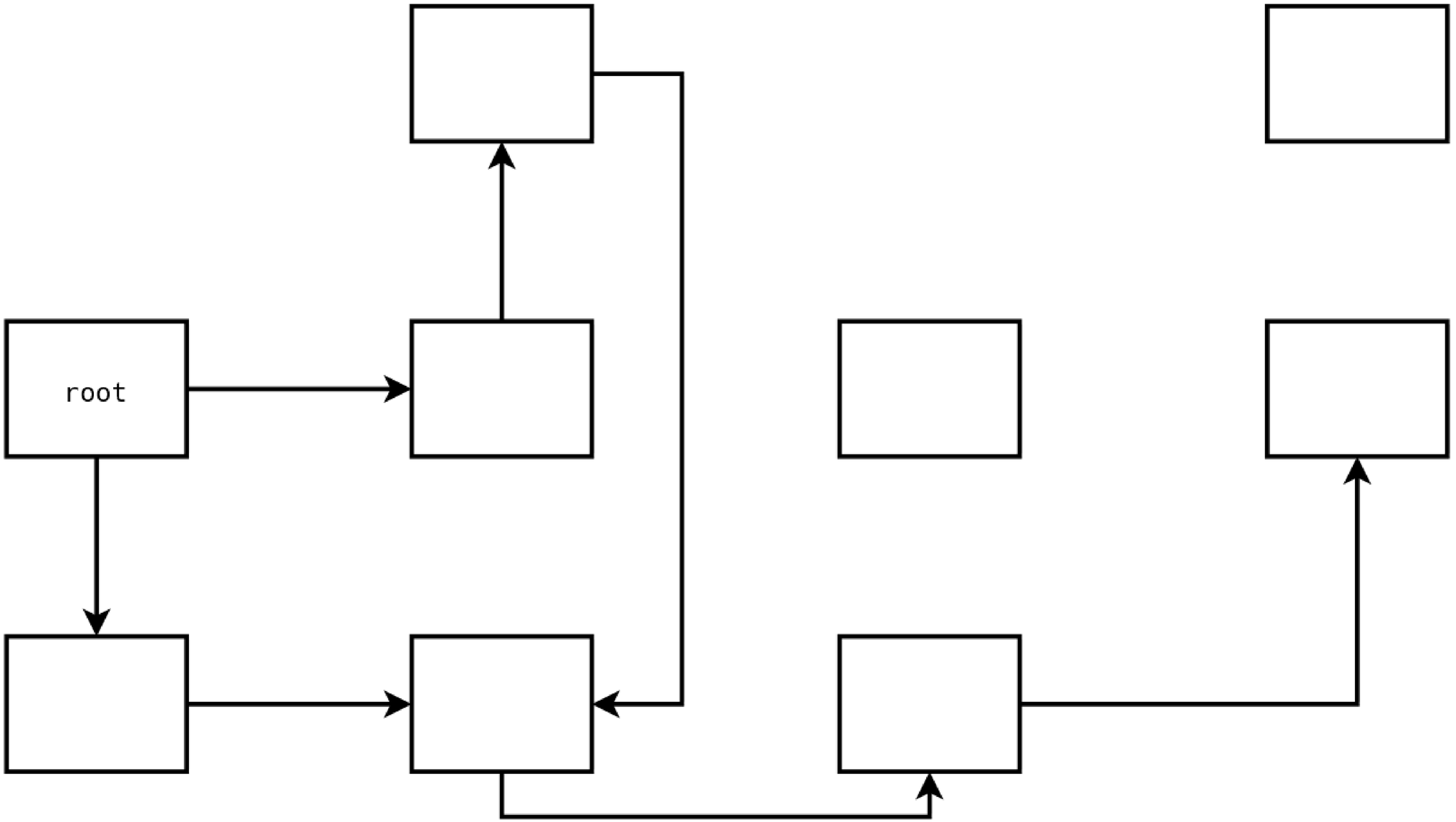
- Мусор – всё, что не является живым объектом

Что значит живой объект?

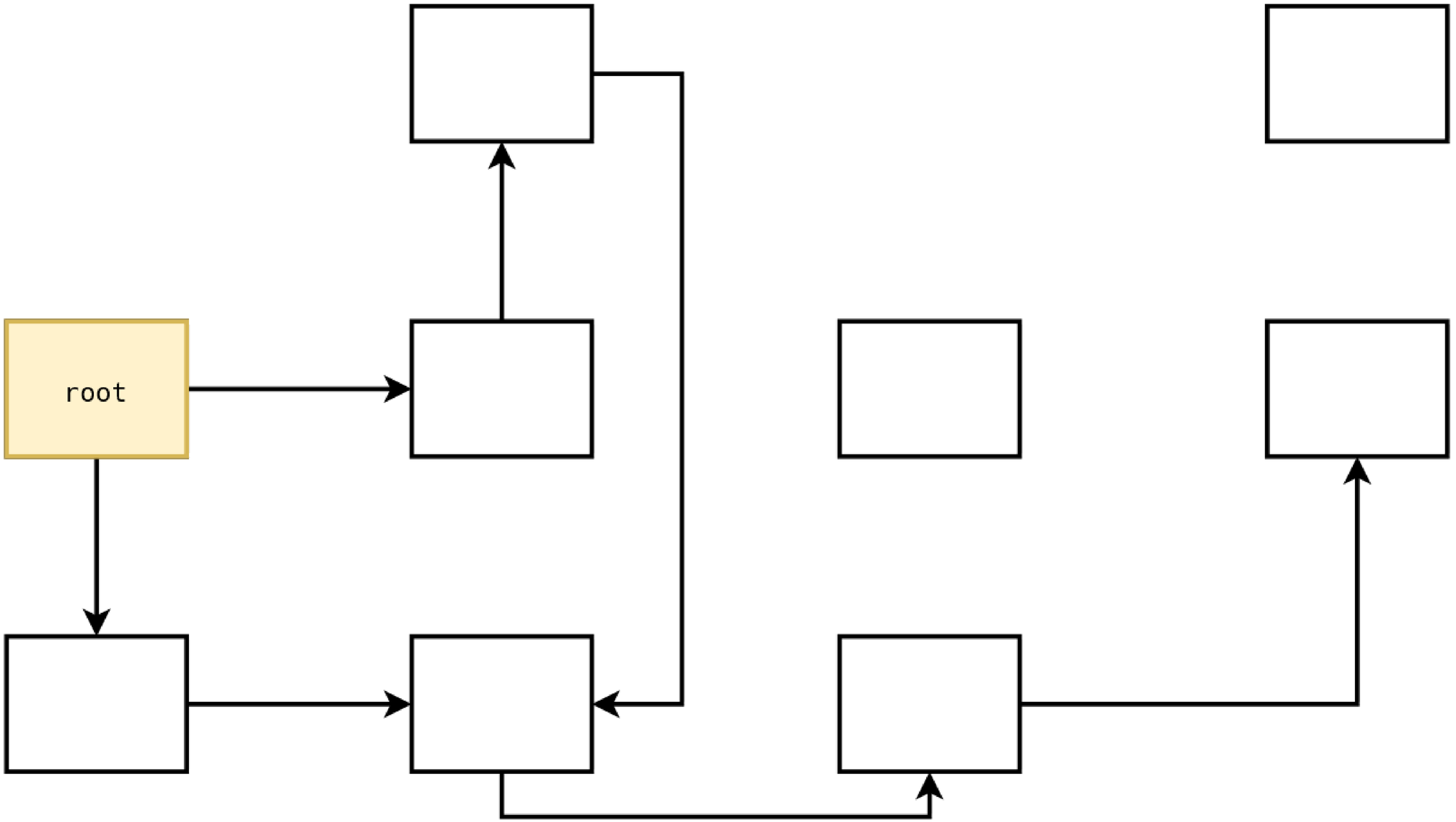
- Живой объект – такой объект, до которого можно дойти по ссылкам от корневого объекта



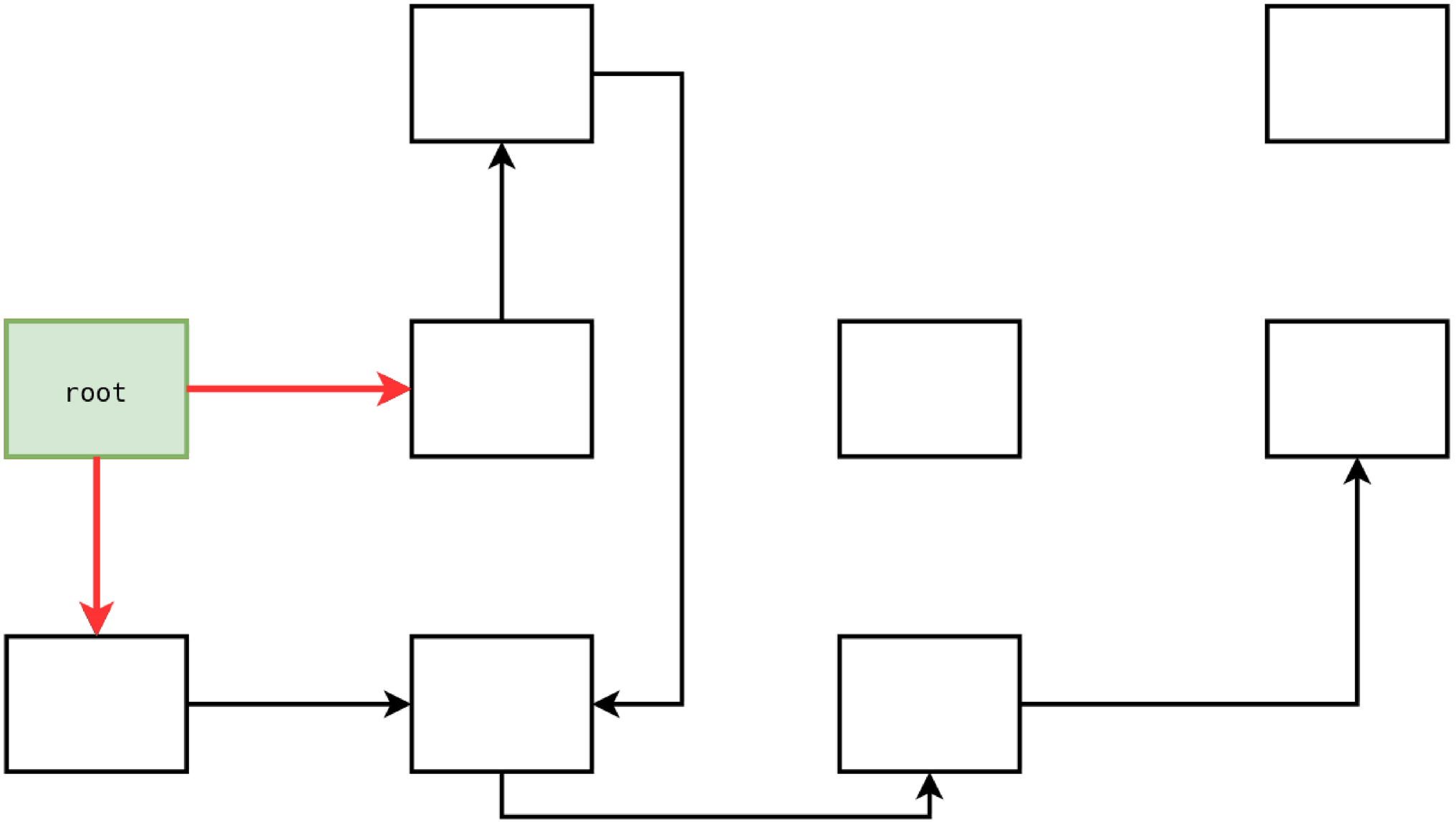
Что значит дойти по ссылкам?



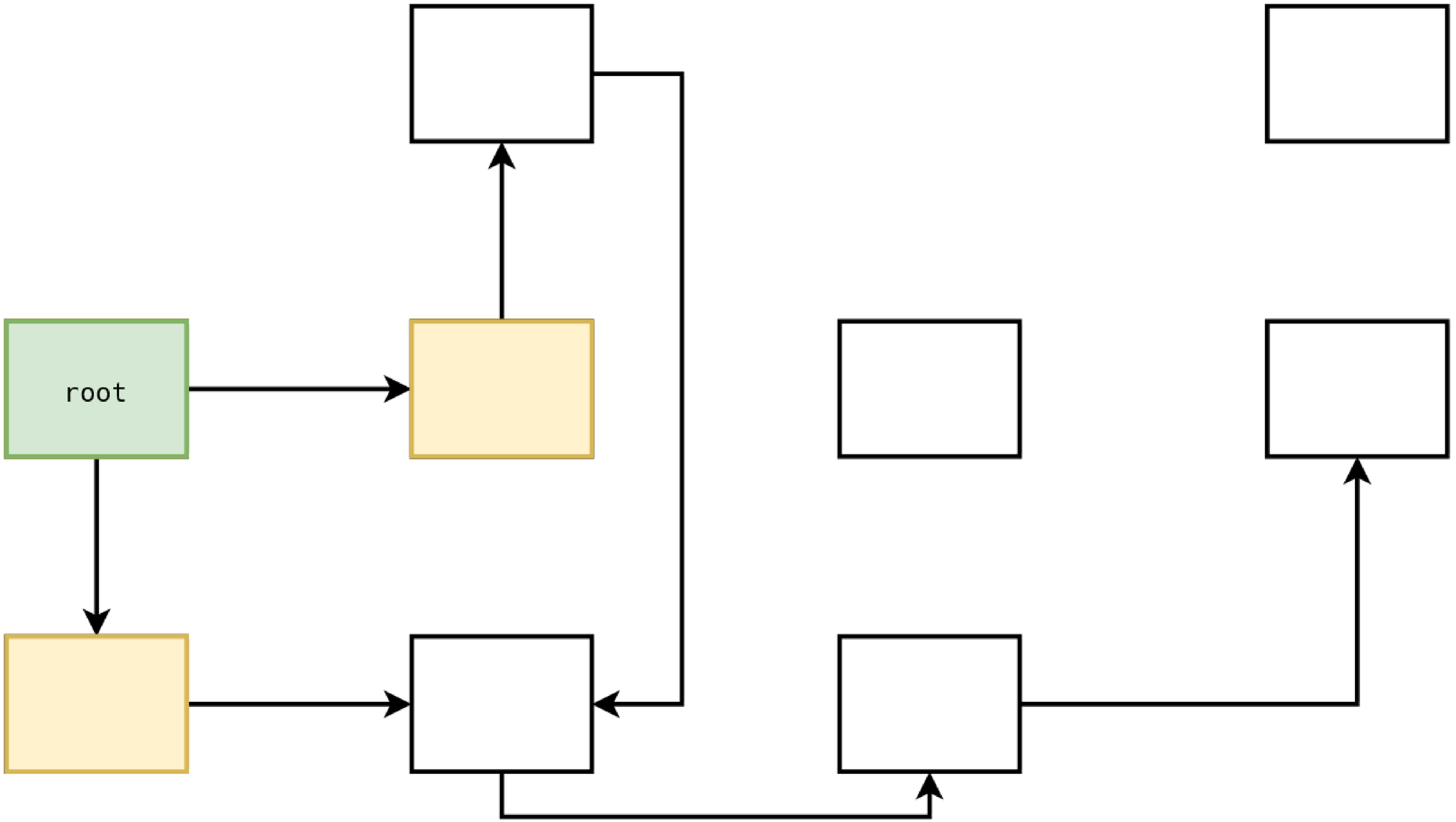
Что значит дойти по ссылкам?



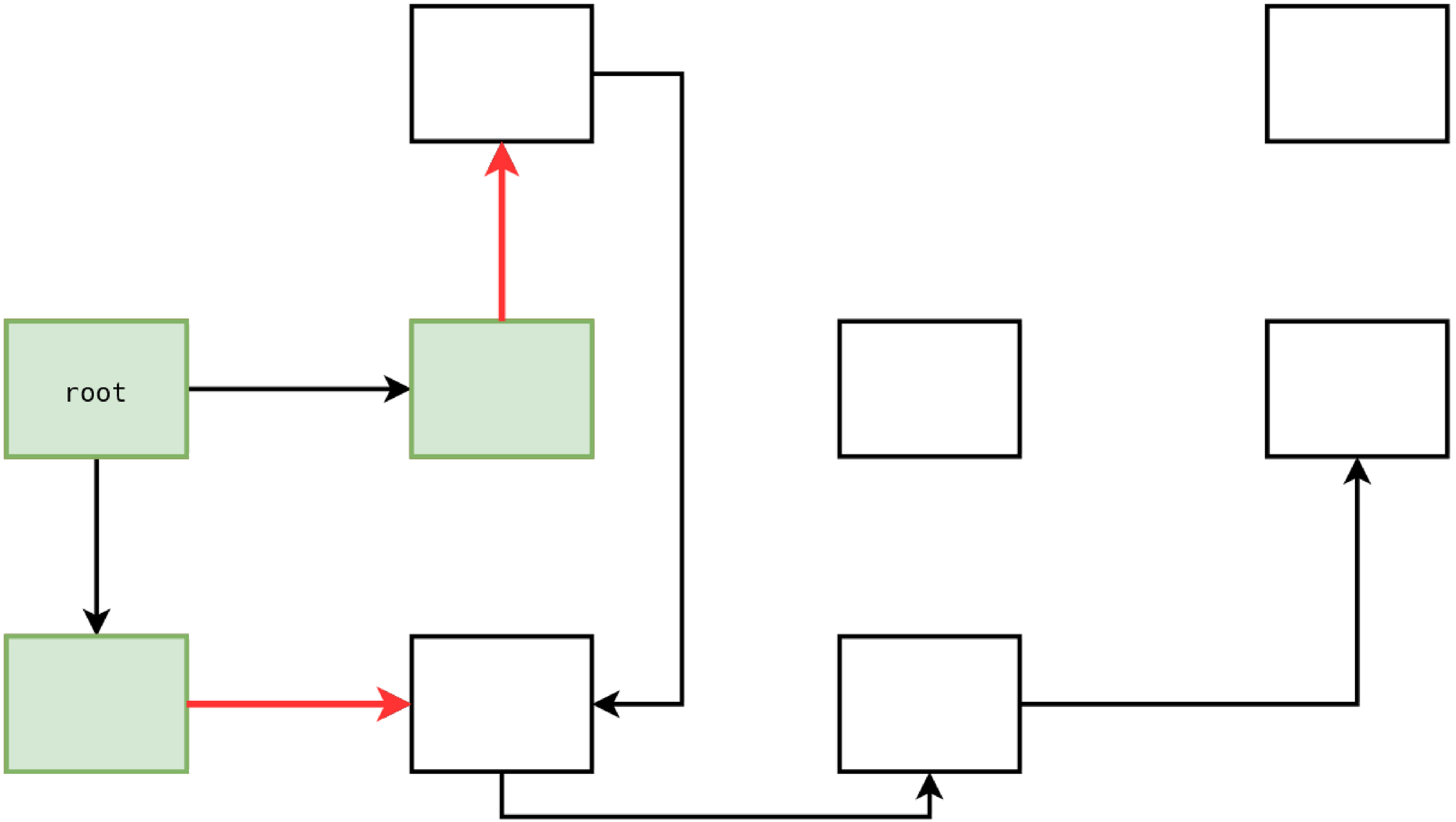
Что значит дойти по ссылкам?



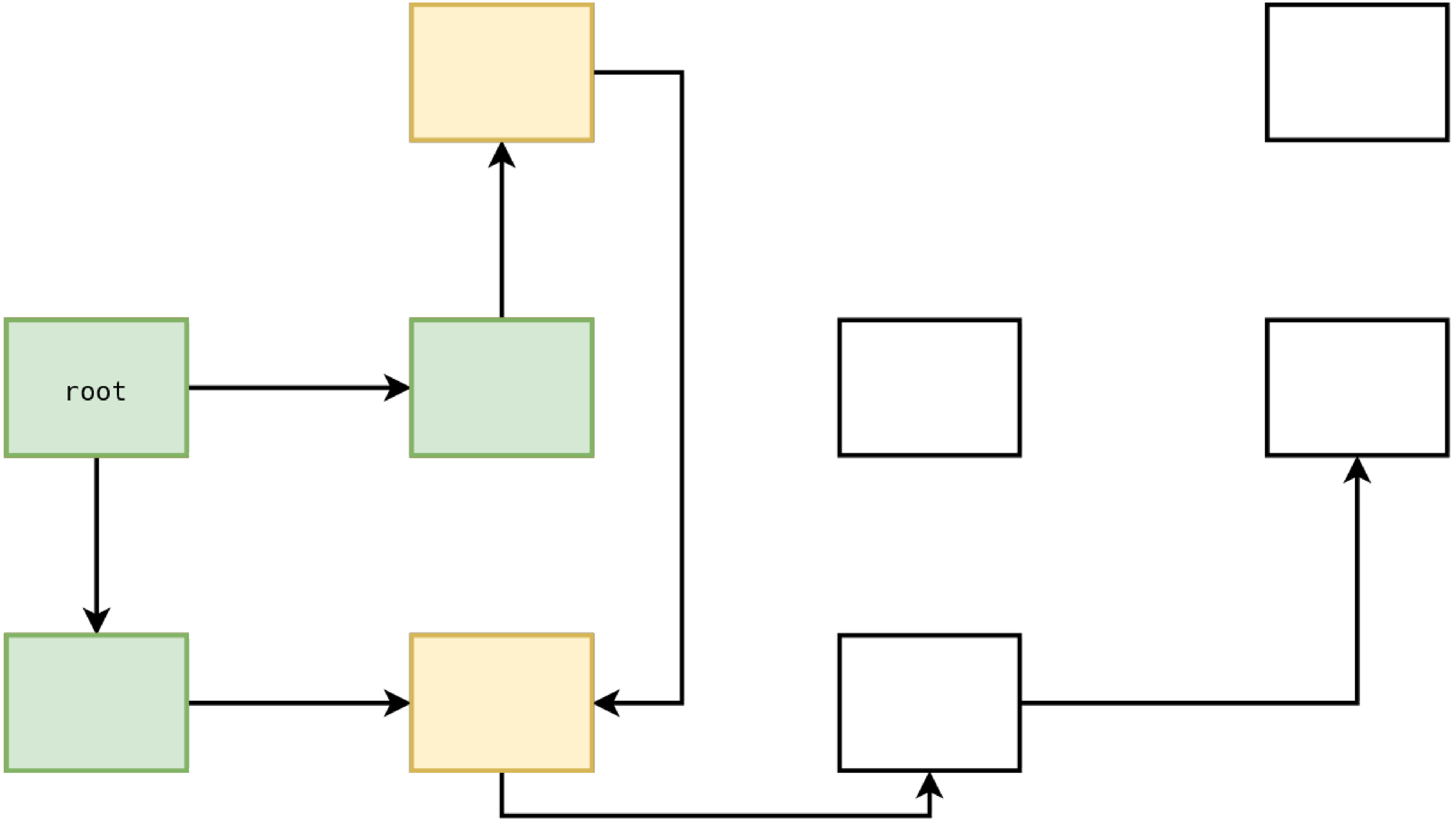
Что значит дойти по ссылкам?



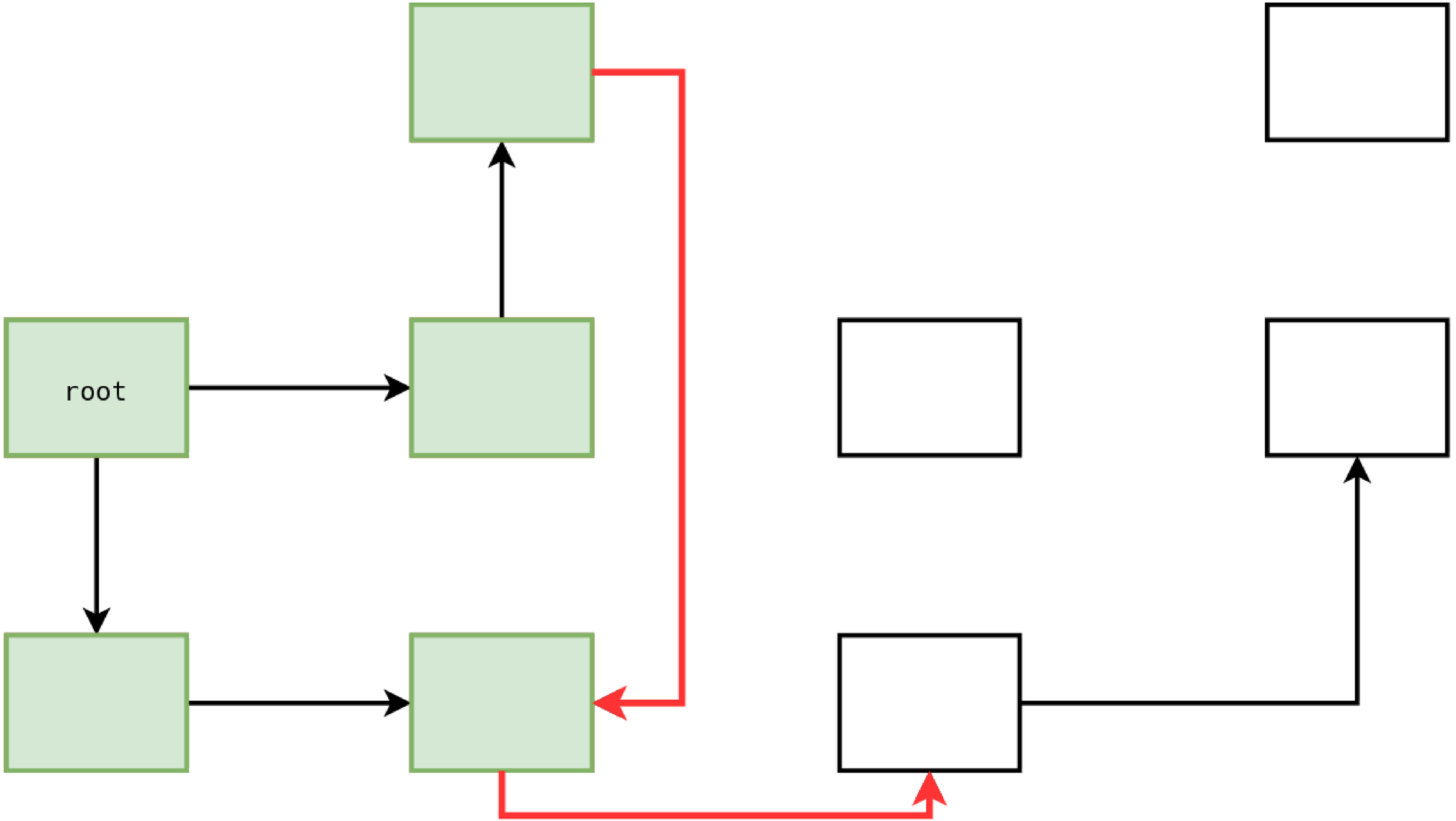
Что значит дойти по ссылкам?



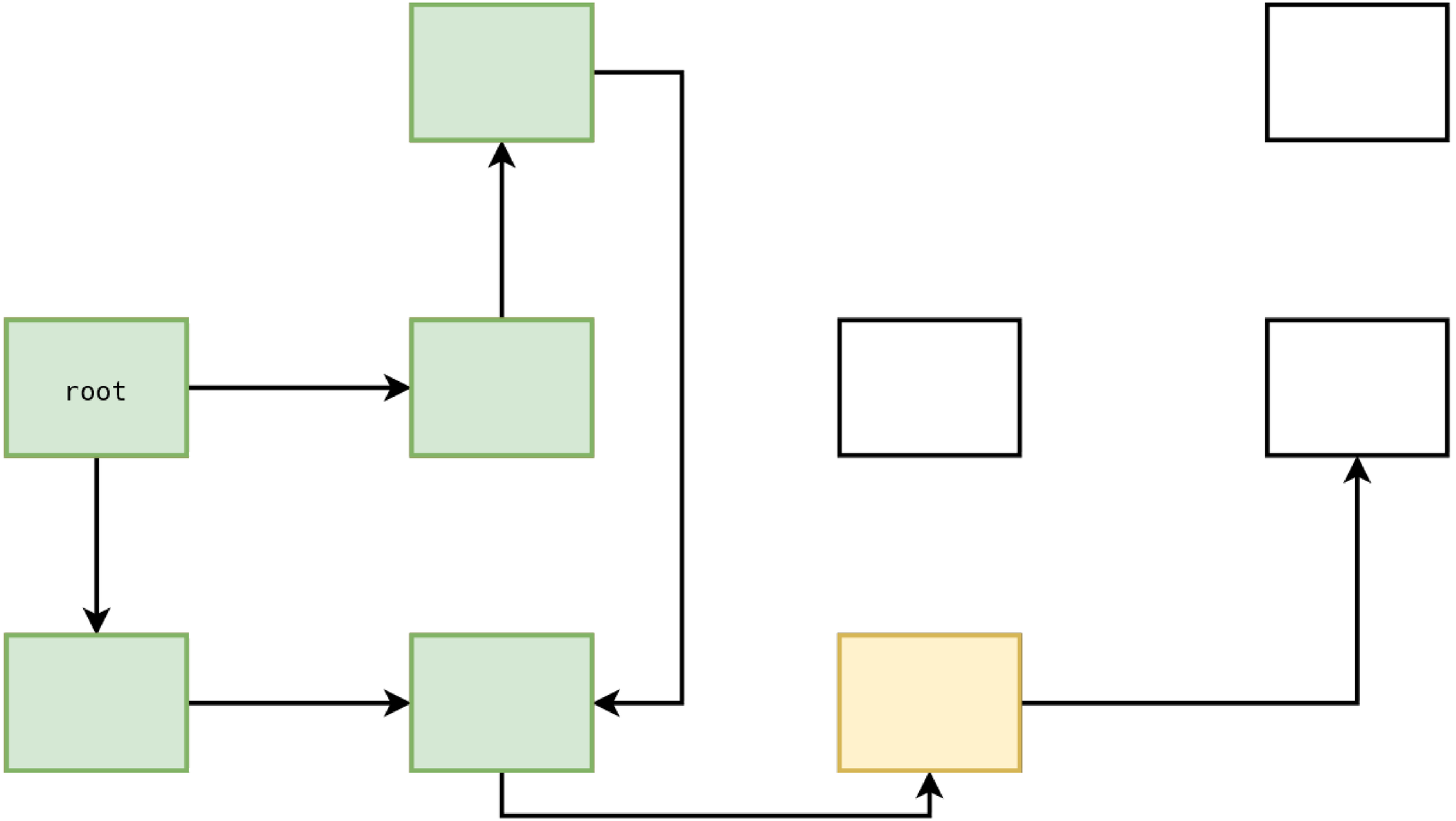
Что значит дойти по ссылкам?



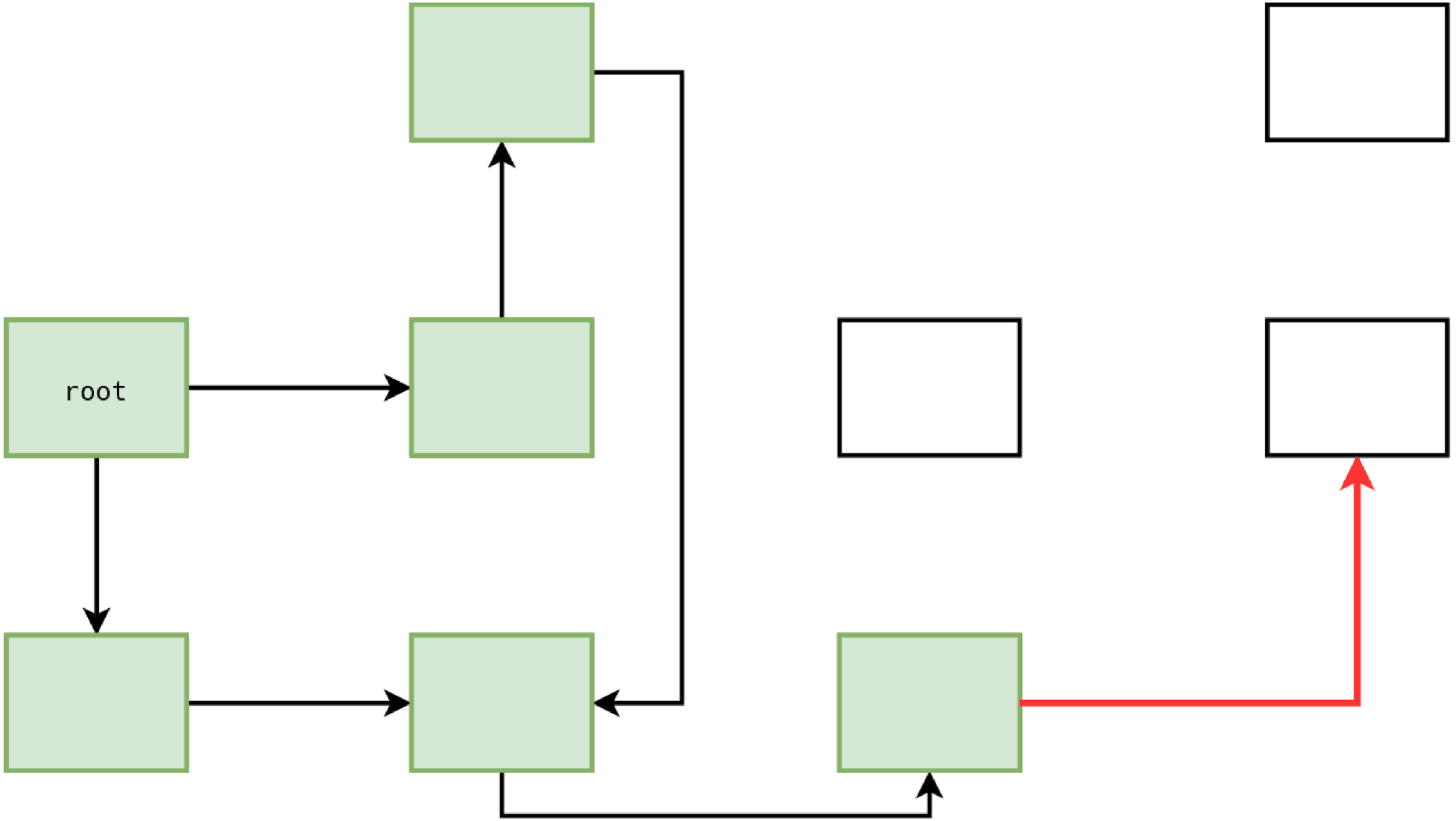
Что значит дойти по ссылкам?



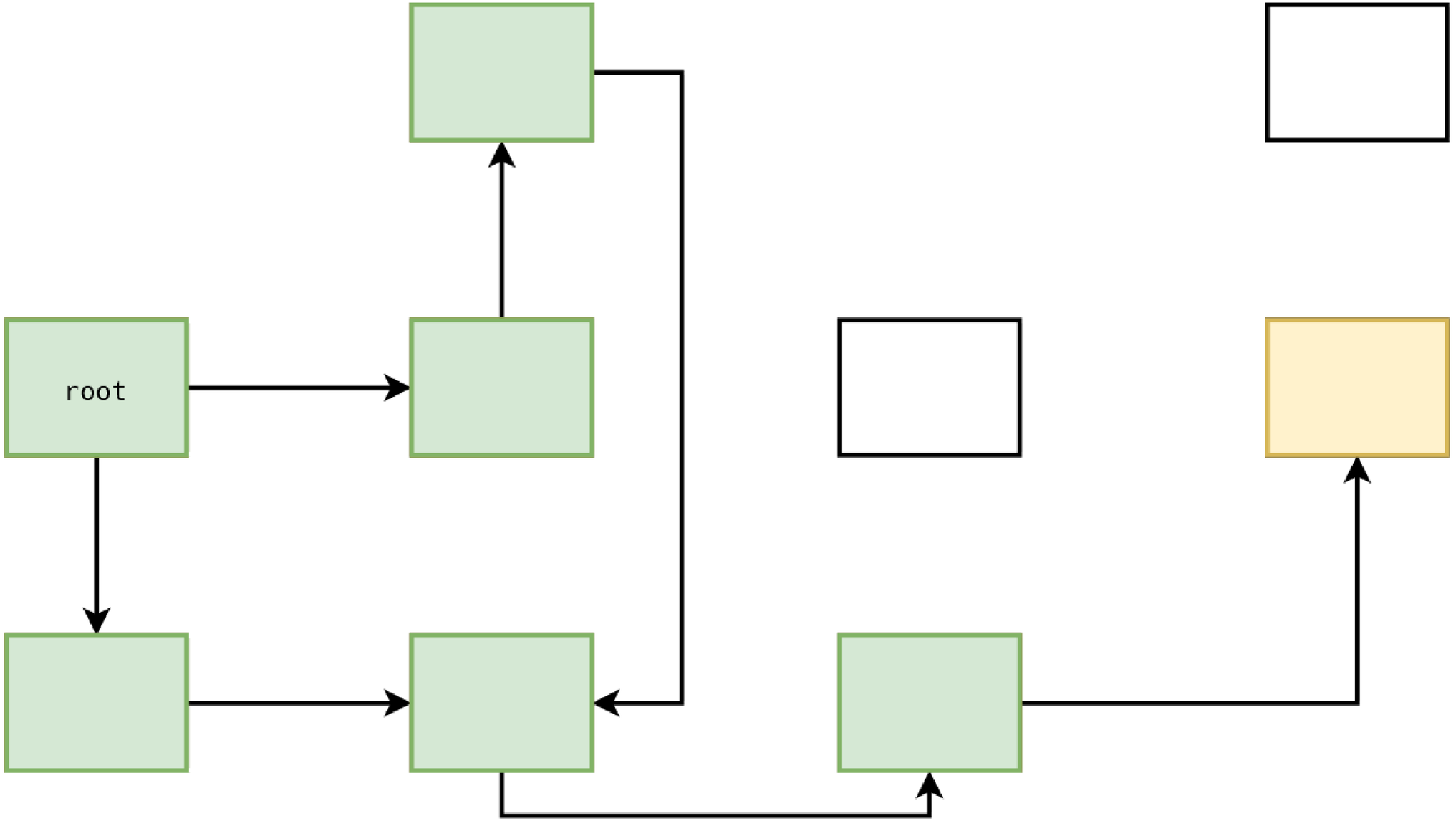
Что значит дойти по ссылкам?



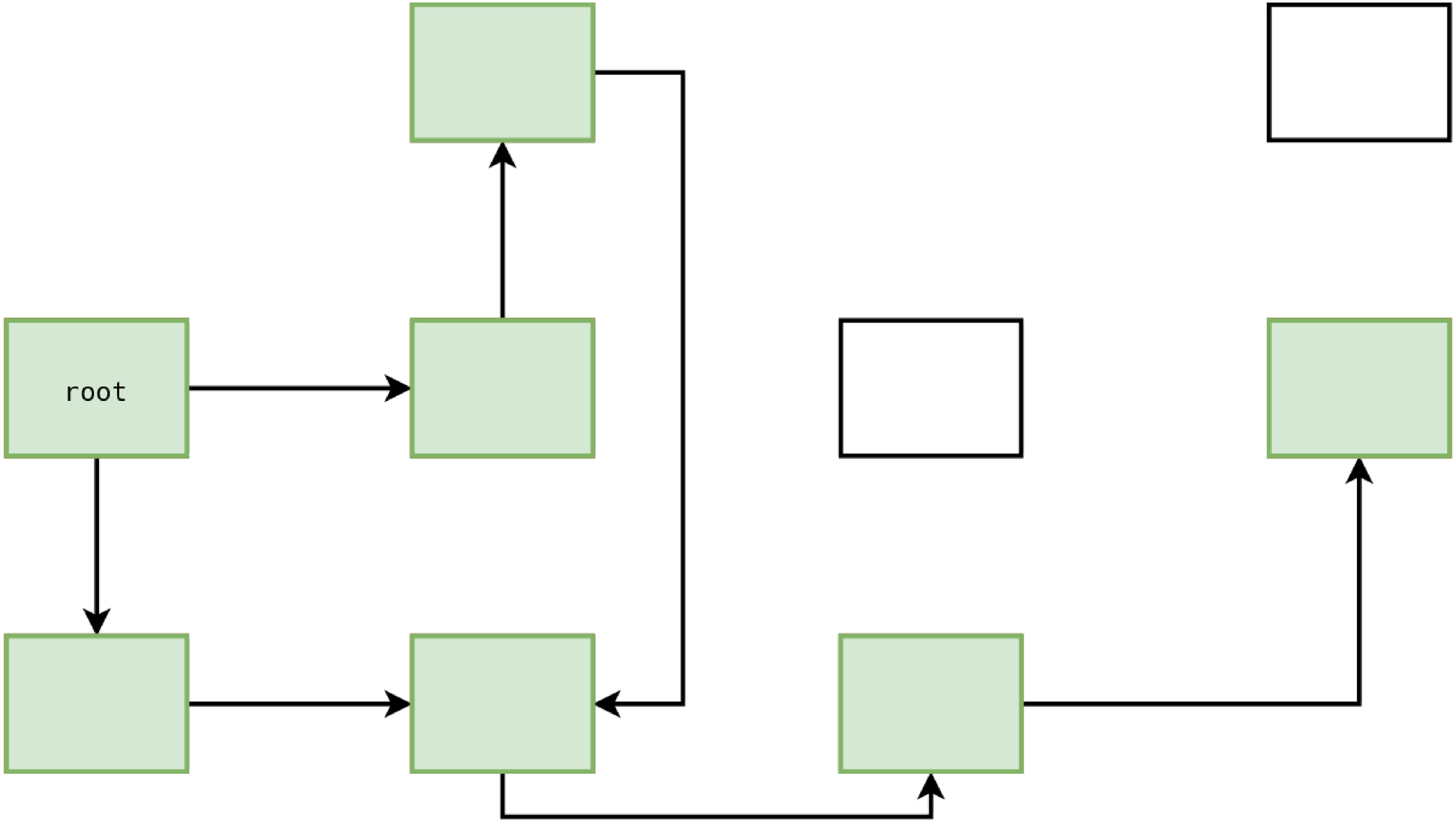
Что значит дойти по ссылкам?



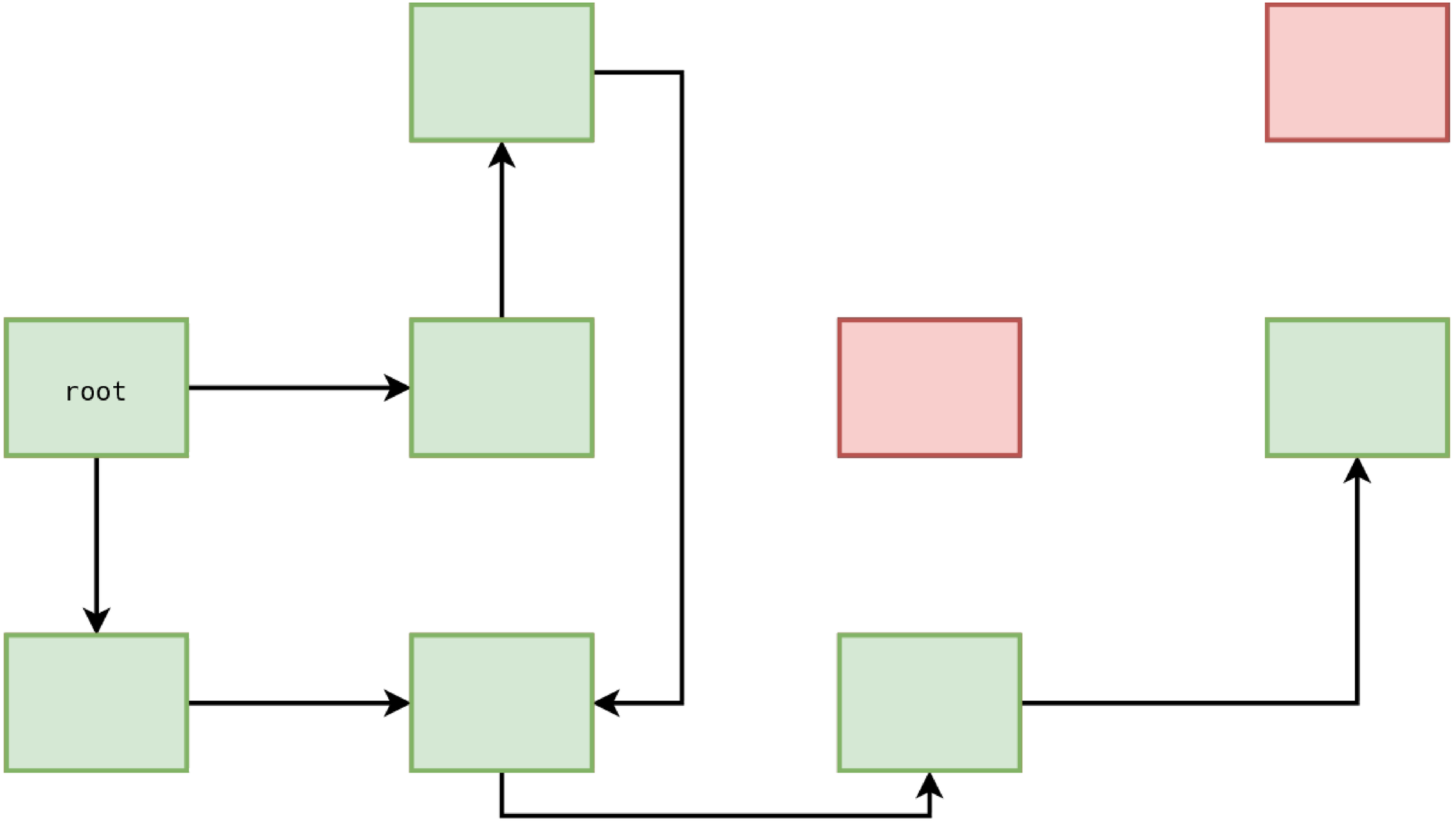
Что значит дойти по ссылкам?



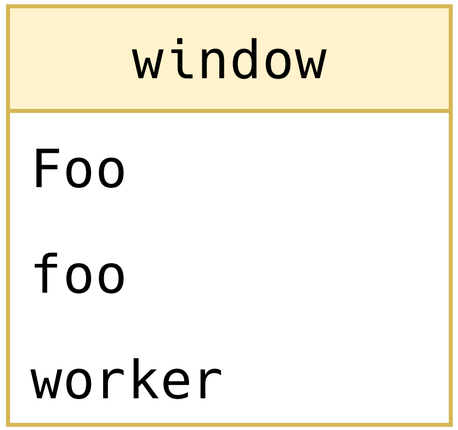
Что значит дойти по ссылкам?



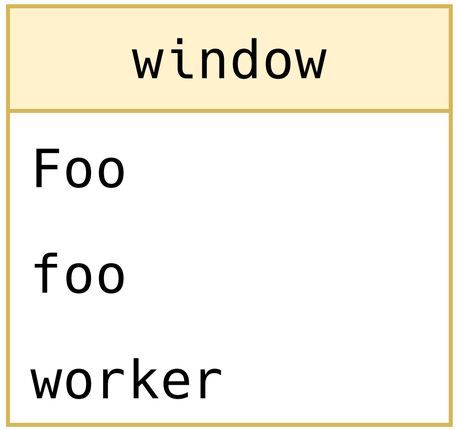
Что значит дойти по ссылкам?



Что значит корневой объект?

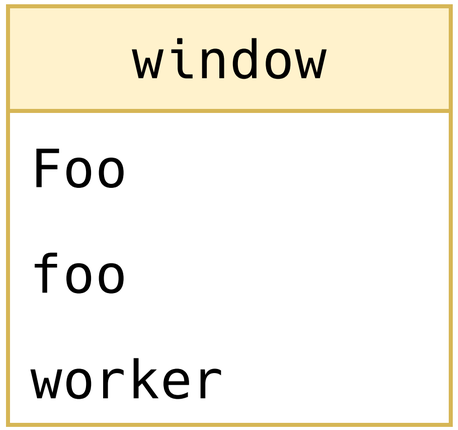


Что значит корневой объект?



DOM
Microtask
Macrotask
RAF
Idle

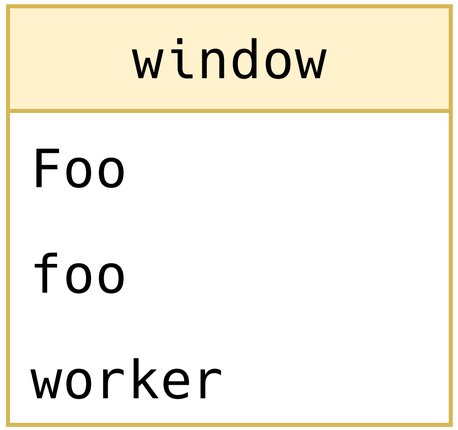
Что значит корневой объект?



DOM
Microtask
Macrotask
RAF
Idle

```
function foo (a, b, c) {
  function bar (x, y, z) {
    const x = {}; // nomem, run gc D:
    // ...
  }
  while (whatever()) bar();
}
```

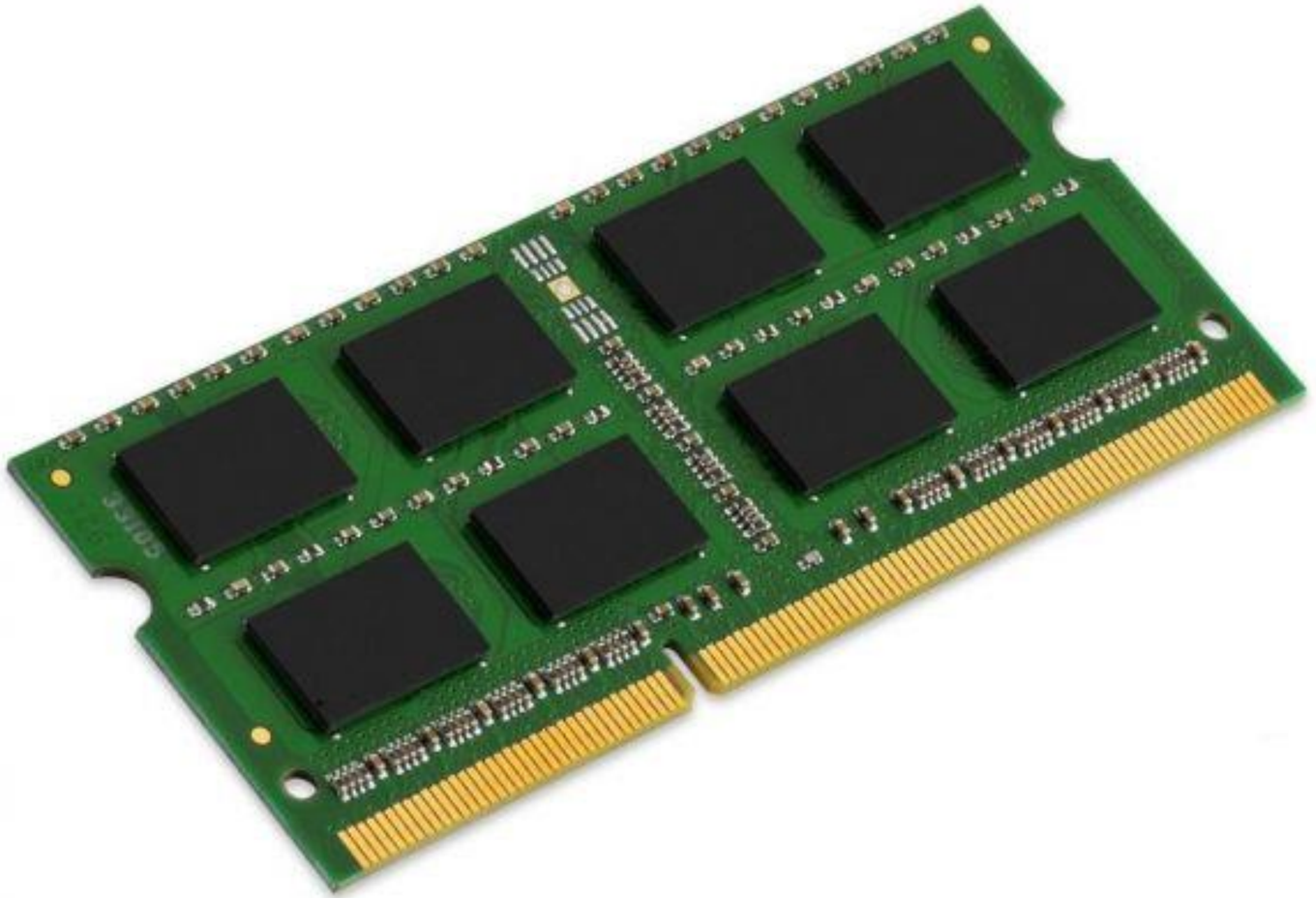
Что значит корневой объект?



DOM
Microtask
Macrotask
RAF
Idle

```
function foo (a, b, c) {
  function bar (x, y, z) {
    const x = {}; // nomem, run gc D:
    // ...
  }
  while (whatever()) bar();
}
```

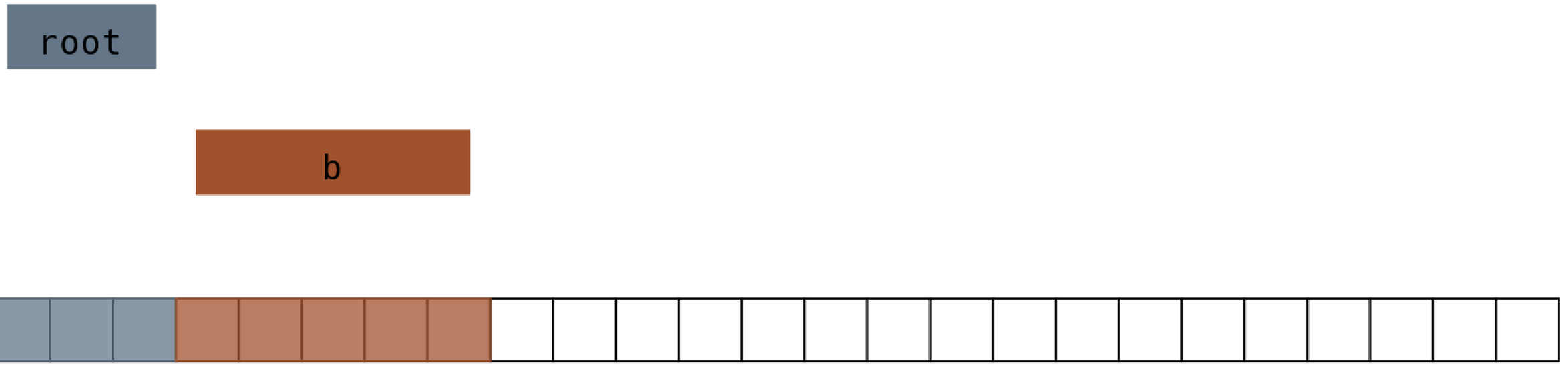
Суровая реальность

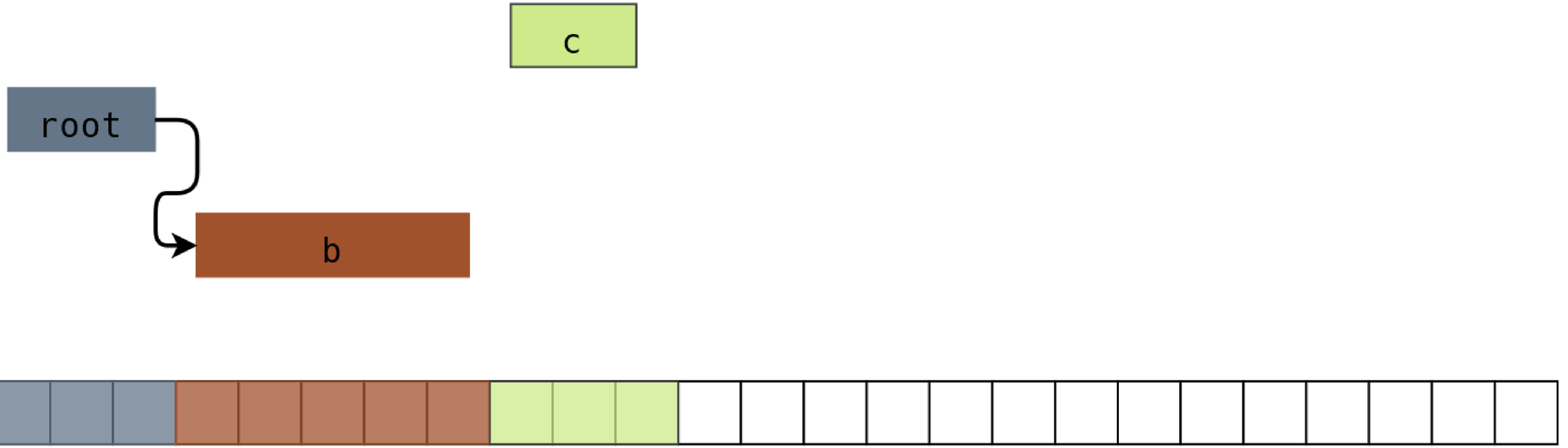


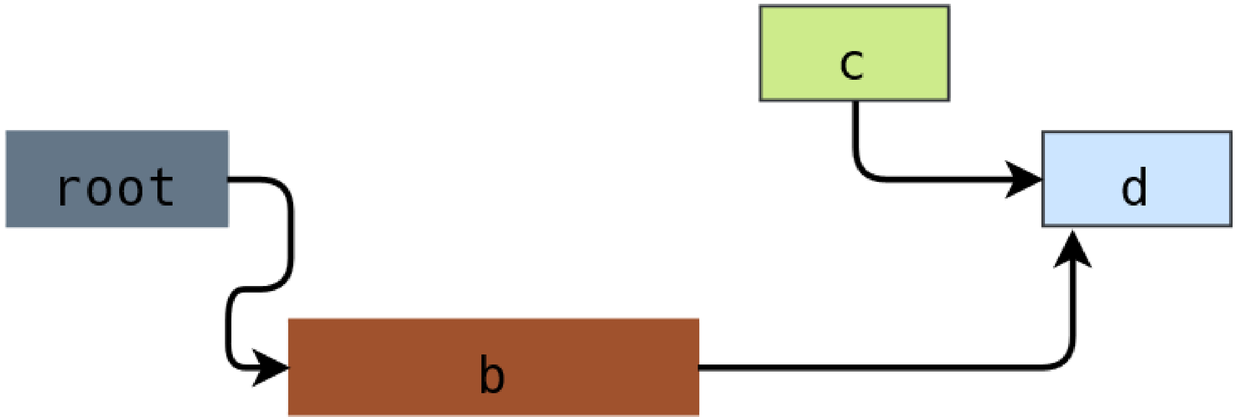

HYPER X
PREDATOR
Kingston

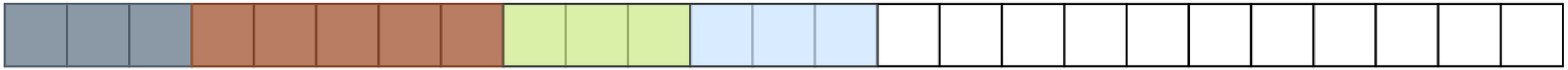
```
new Uint32Array(16 * 2 ** 30);
```

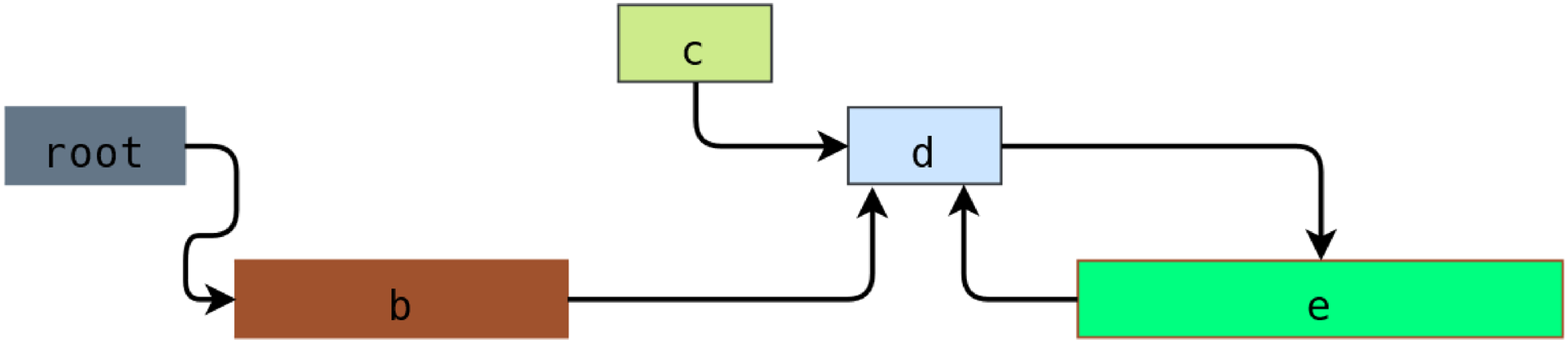
root

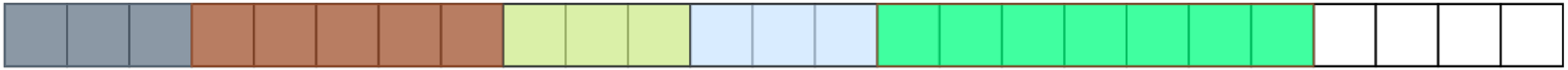

root
b

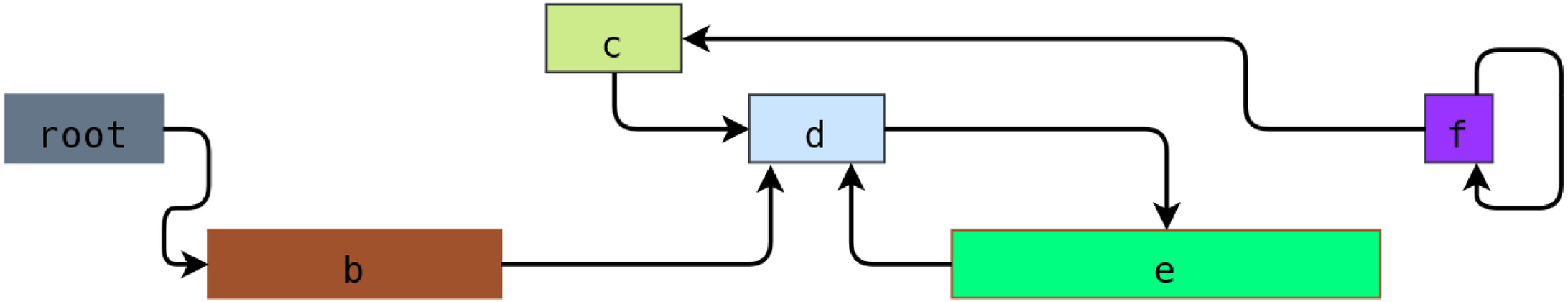

c
root
b

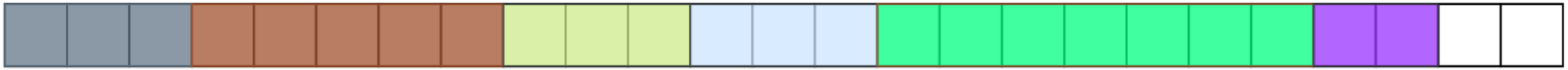

c
root
d
b

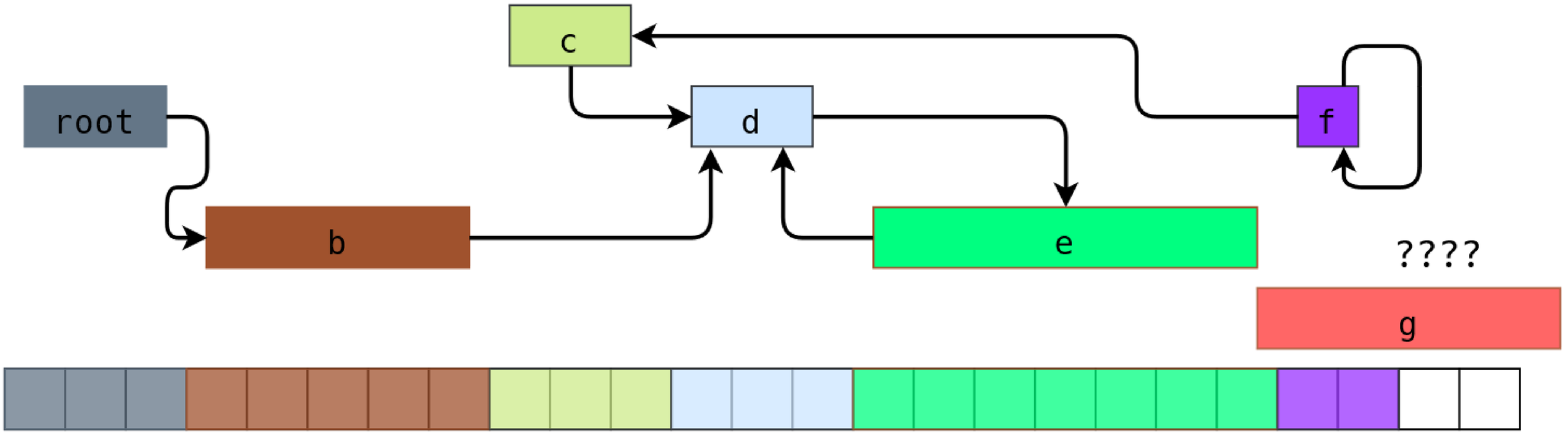



c
root
d
b
e




root
b
c
d
e
f




c
root
d
f
b
e
????
g

Aw, Snap!

Something went wrong while displaying this webpage.

Learn more

Reload

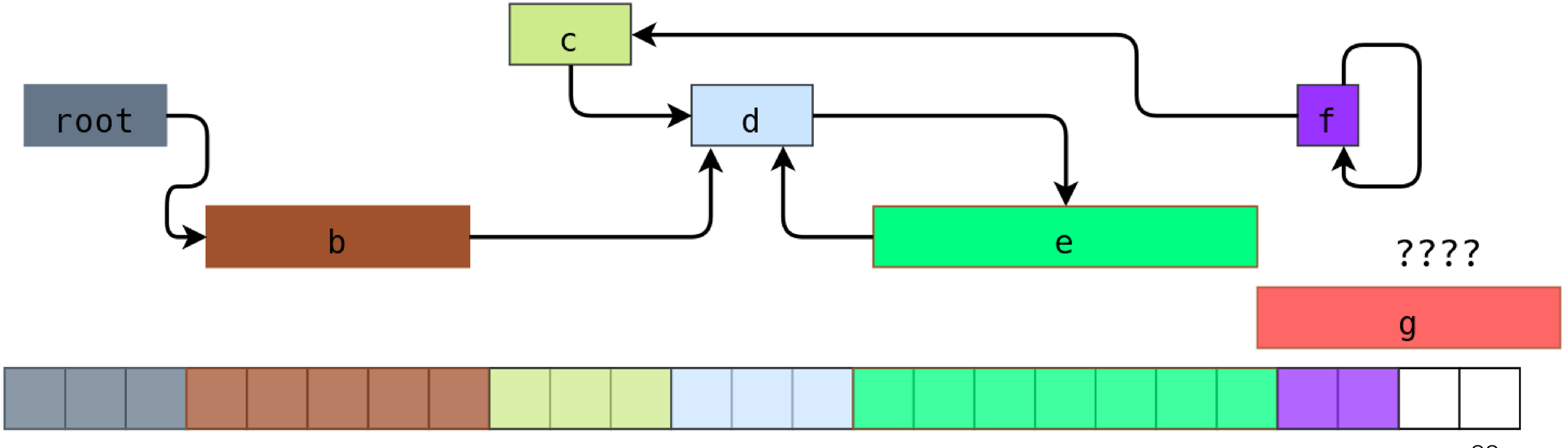
No-op collector

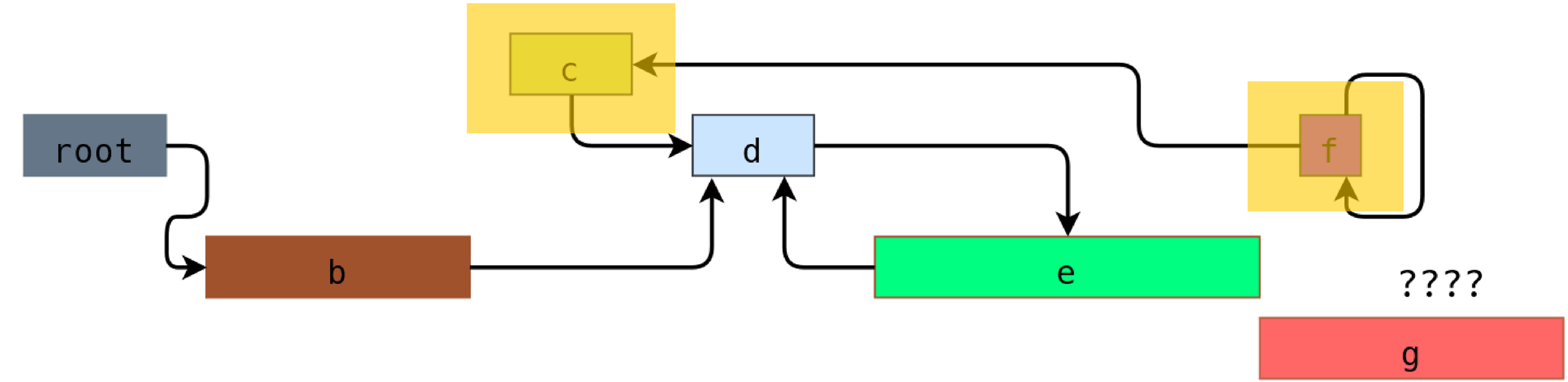
Плюсы

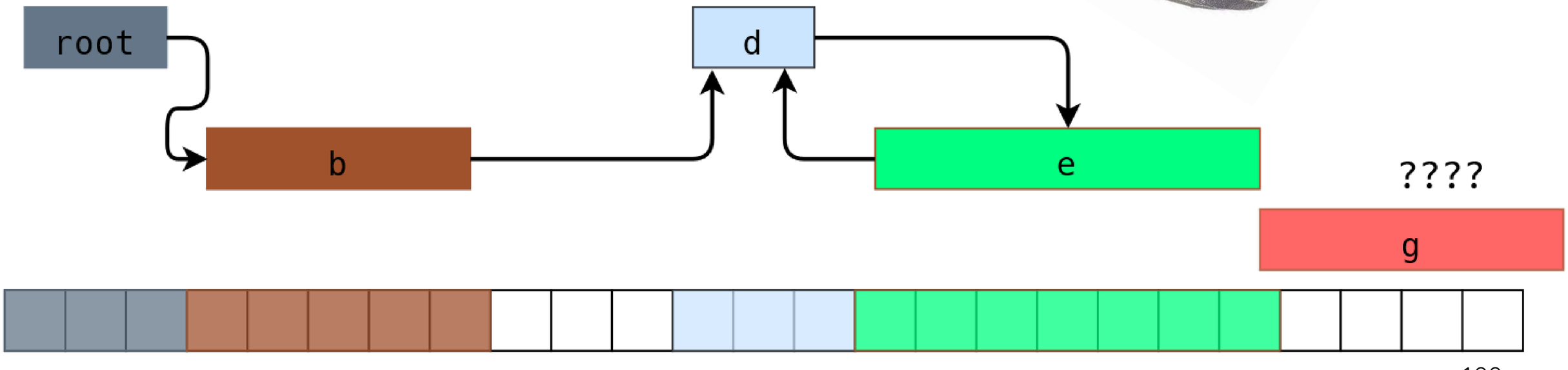
- Очень простой
- У вас просто нет сборки мусора

Минусы

- Ваше приложение падает


c
root
d
f
b
e
????
g


root
b
c
d
e
f
????
g


root
b
d
e
????
g

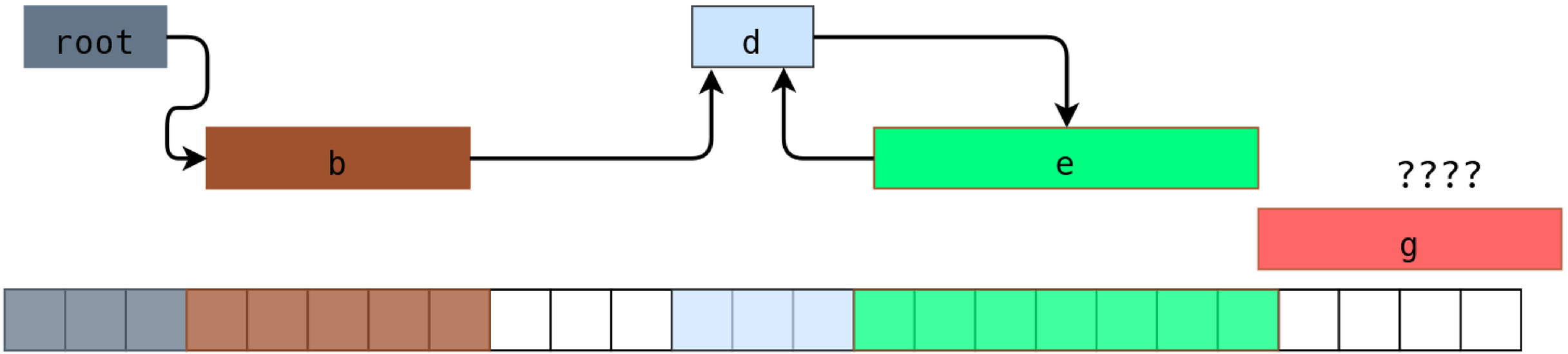
Mark and Sweep

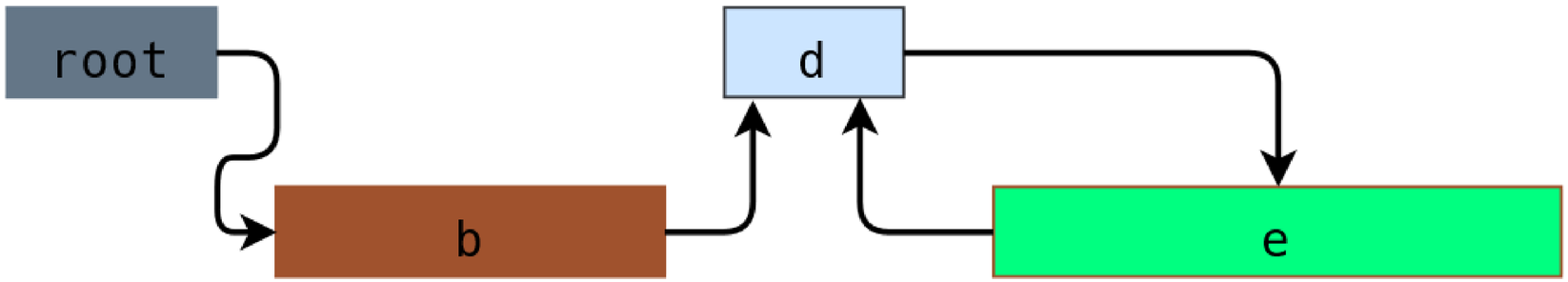
Плюсы

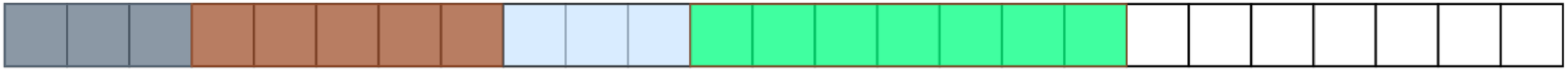
- Простой
- Работает пропорционально количеству мусора
- Хорошо работает, когда у вас мало мусора

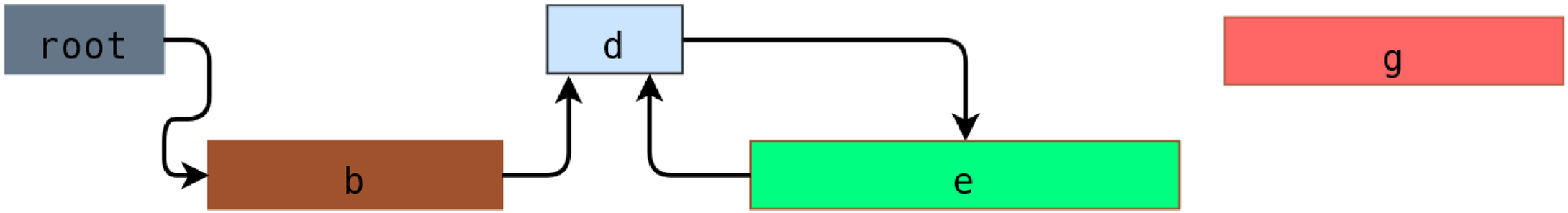
Минусы

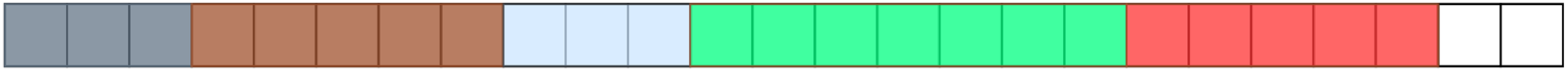
- Требует сложной логики поиска свободного места
- Фрагментирует память


root
b
d
e
????
g


root
d
b
e




root
d
g
b
e



Mark and Compact

Плюсы

- Дефрагментирует память

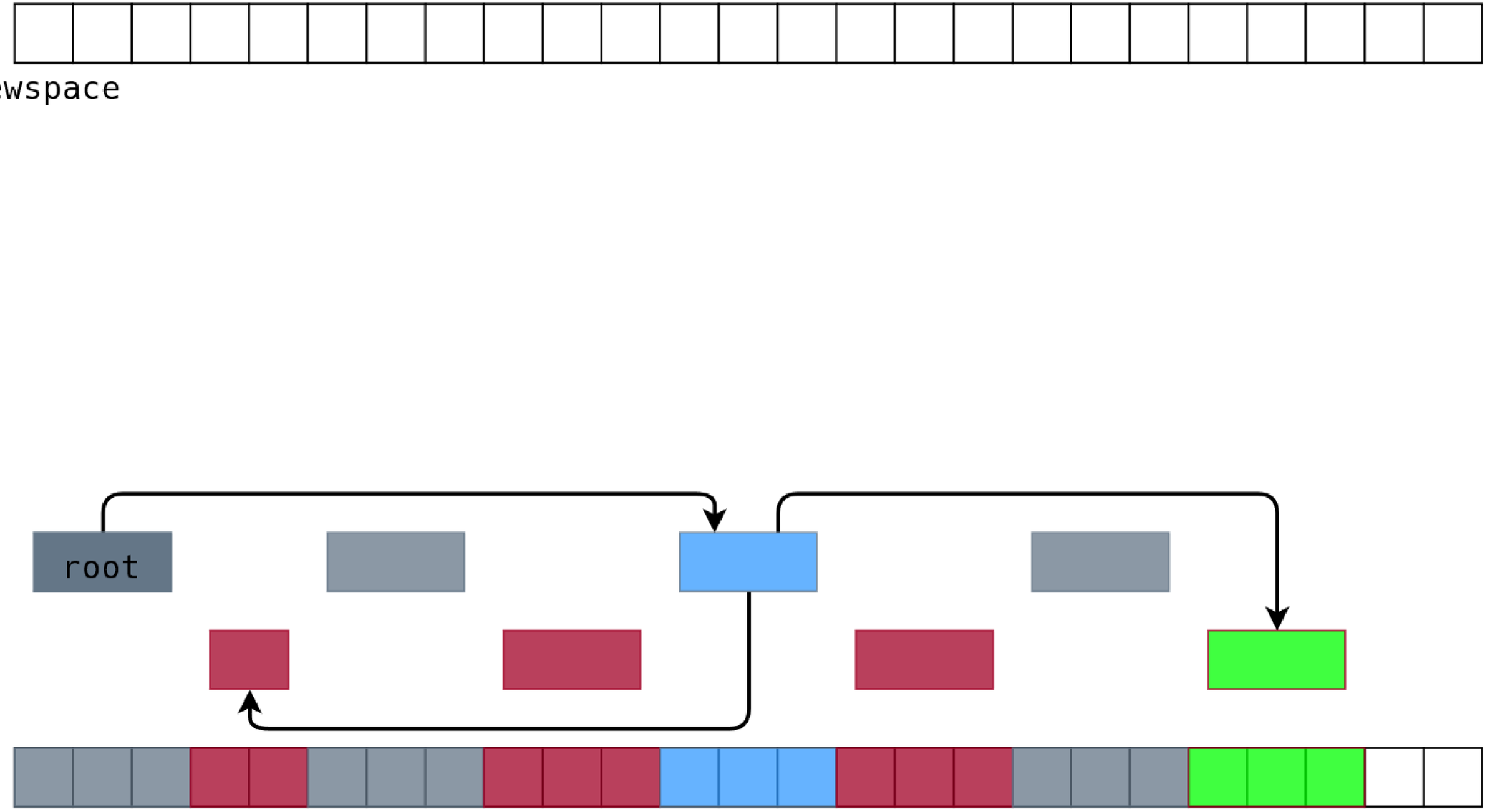
Минусы

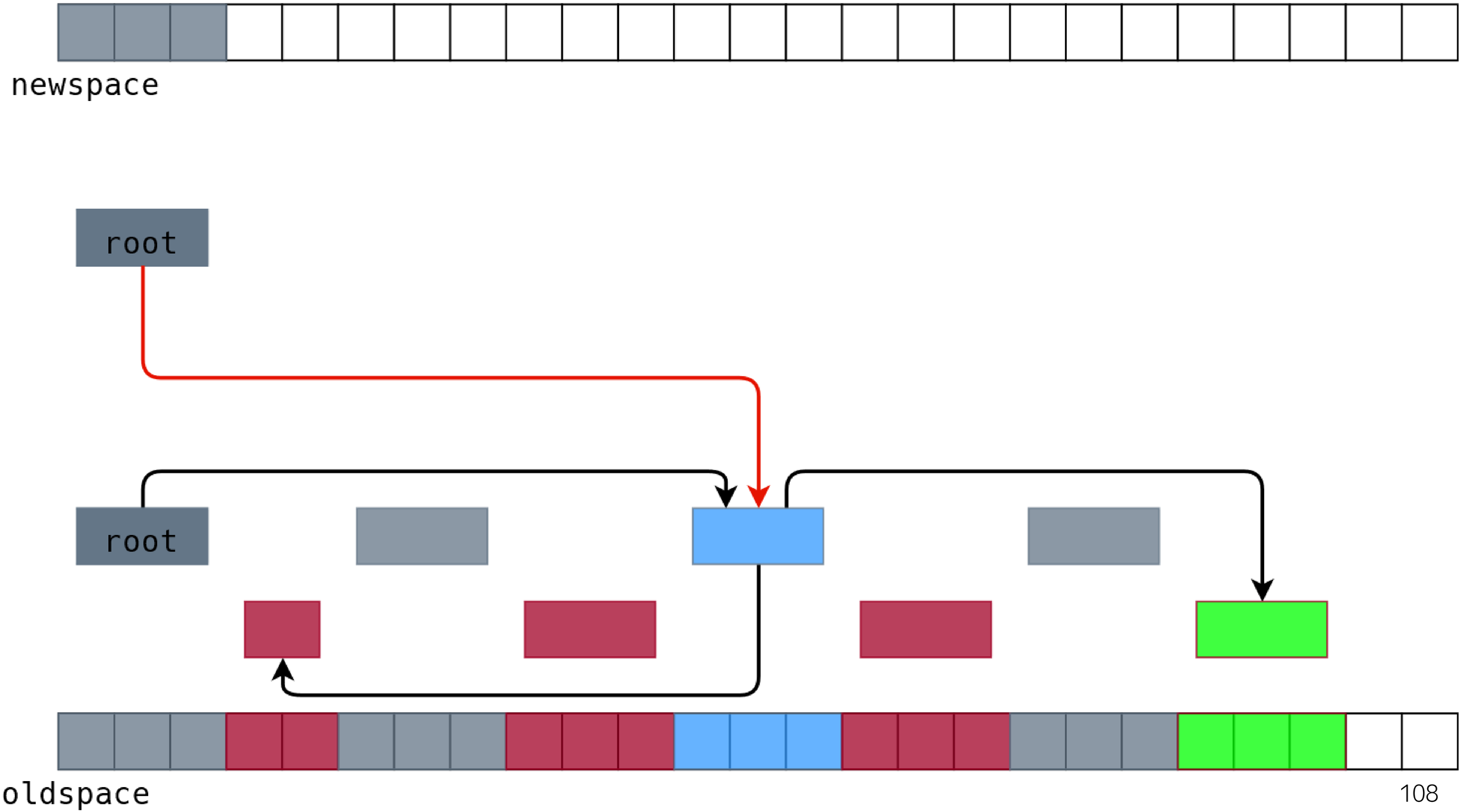
- Сложный
- Перемещает объекты
- Работает пропорционально количеству живых объектов
- Требует 2-3 прохода по всей памяти
- Медленный

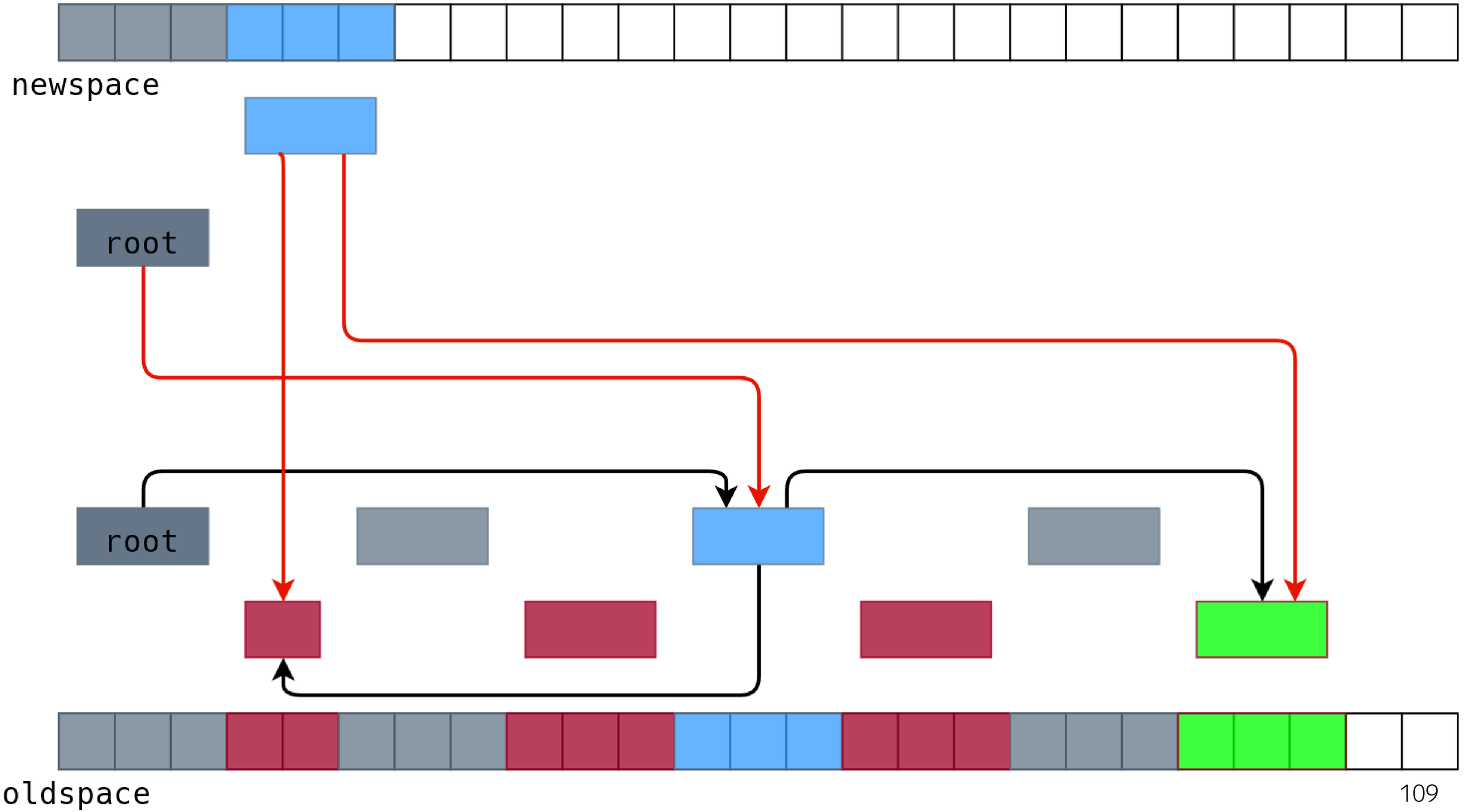
Сборка мусора не бесплатна

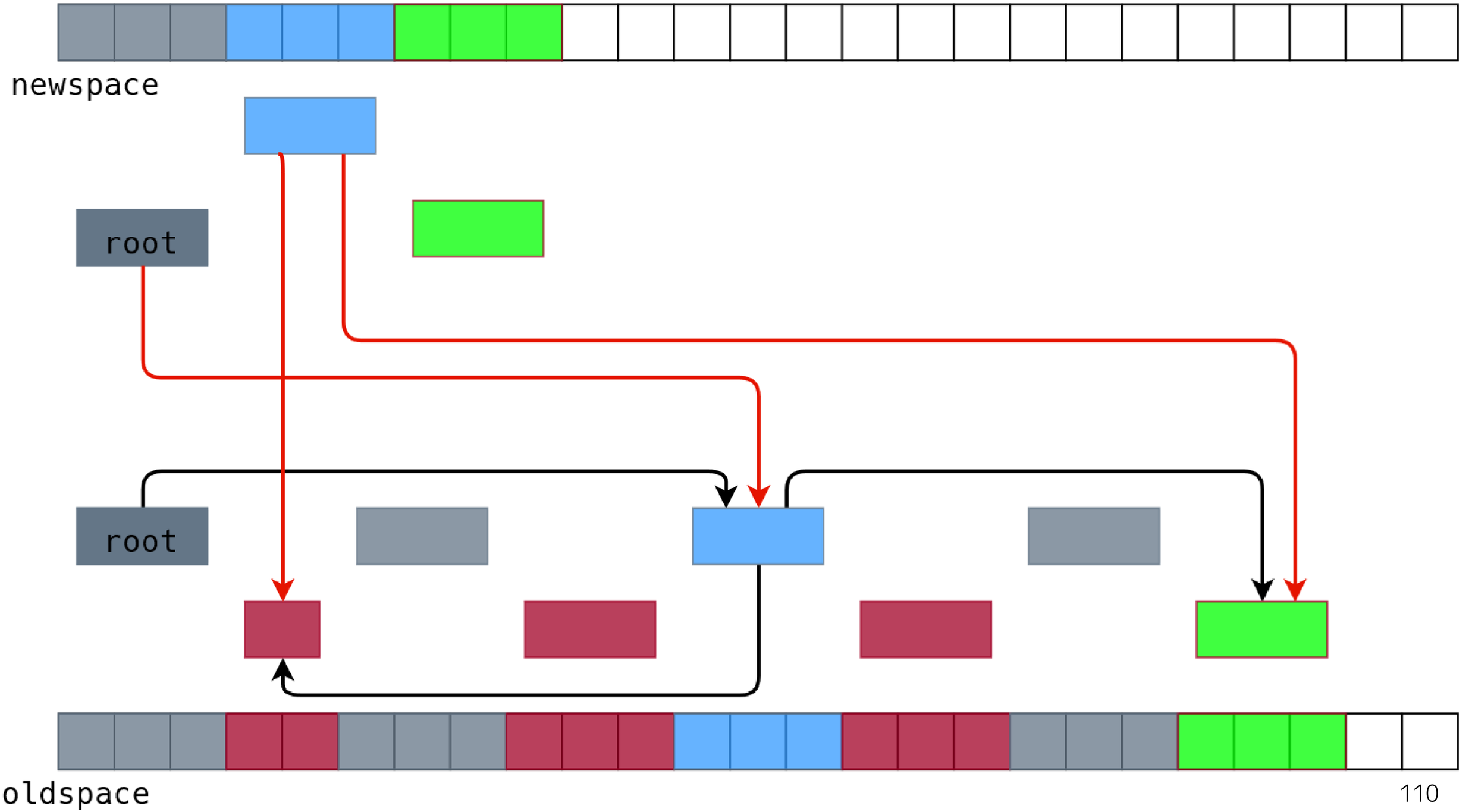
WebGL / WebAudio / WebGPU

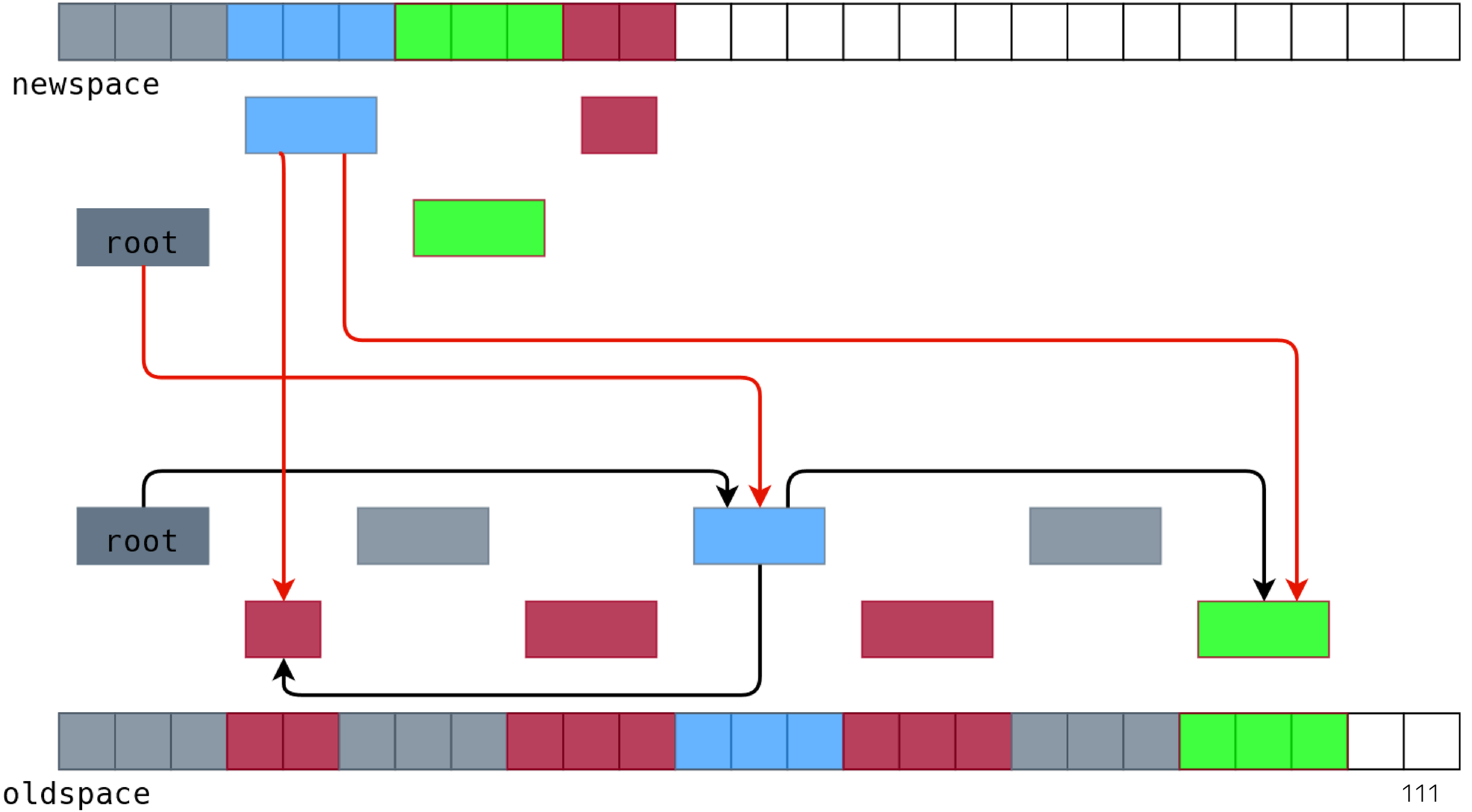
- Инициализация объектов
- pull() вместо Promise

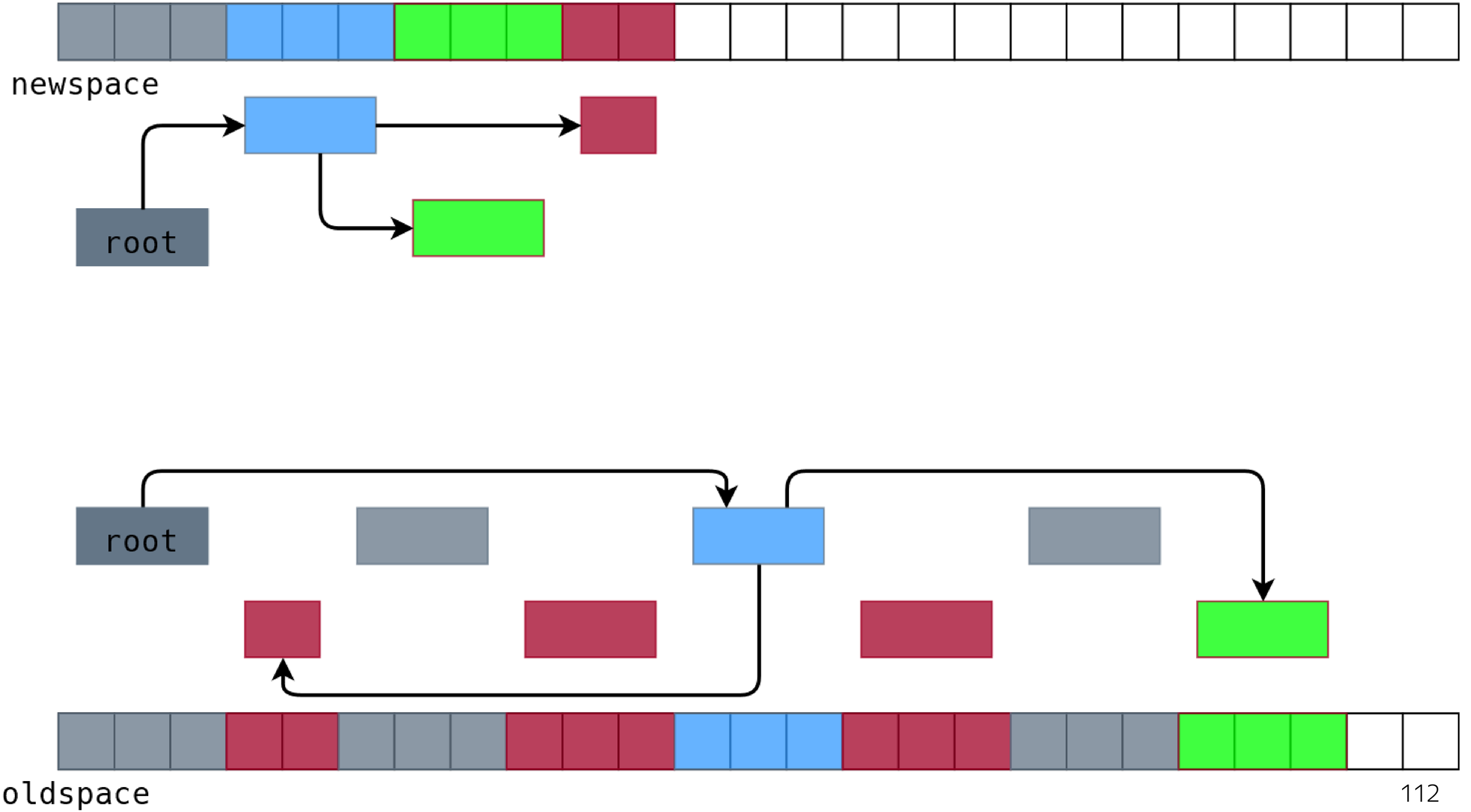

newspace
root
oldspace


newspace
root
root
oldspace

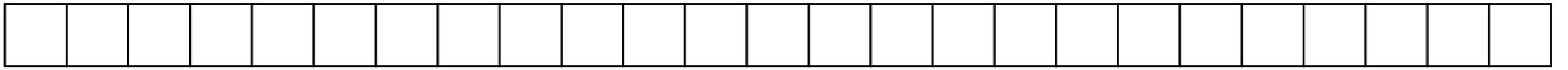

newspace
root
root
oldspace


newspace
root
root
oldspace


newspace
root
root
oldspace


newspace
root
root
oldspace


newspace
root


oldspace

Semispace aka Lisp 2

Плюсы

- Дефрагментирует память
- Простой
- Можно совместить с фазой обхода
- Работает пропорционально количеству живых объектов
- Хорошо работает, когда у вас много мусора

Минусы

- Двойной расход памяти
- Перемещает объекты

На заметку

- Сборщики мусора могут перемещать объекты

- Расширения используют двойные ссылки aka «хендлы»
- Например, v8::Local<v8::String>

Разные алгоритмы

	Mark and Compact	Mark and Sweep	Semispace	Eden
Сложность	~ все	~ мусор	~ живые	~ живые
Сборка	Мееедленно	Медленно	Быстро	Быстро
Создание	Быстро	Медленно	Быстро	Быстро
Фрагментация	нет	есть	нет	нет
Минус	Сложная реализация	Сложная аллокация	Память x2	Сбрасывается

Разные алгоритмы

	Mark and Compact	Mark and Sweep	Semispace	Eden
Сложность	~ все	~ мусор	~ живые	~ живые
Сборка	Мееедленно	Быстро	Быстро	Быстро
Создание	Быстро	Медленно	Быстро	Быстро
Фрагментация	нет	есть	нет	нет
Минус	Сложная реализация	Сложная аллокация	Память x2	Сбрасывается

Лучшее из всех миров

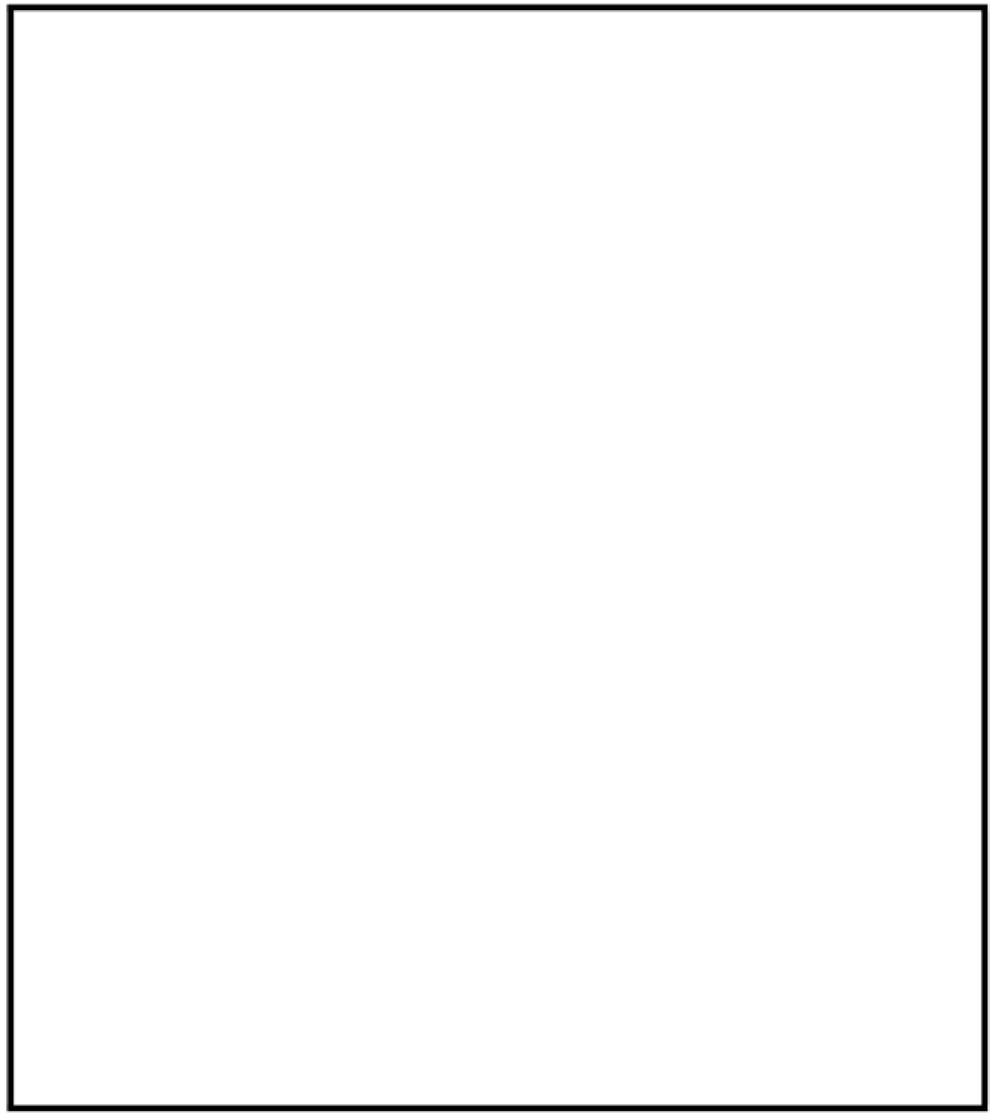
- Можно использовать несколько алгоритмов
- Но как?

Большинство объектов умирают молодыми

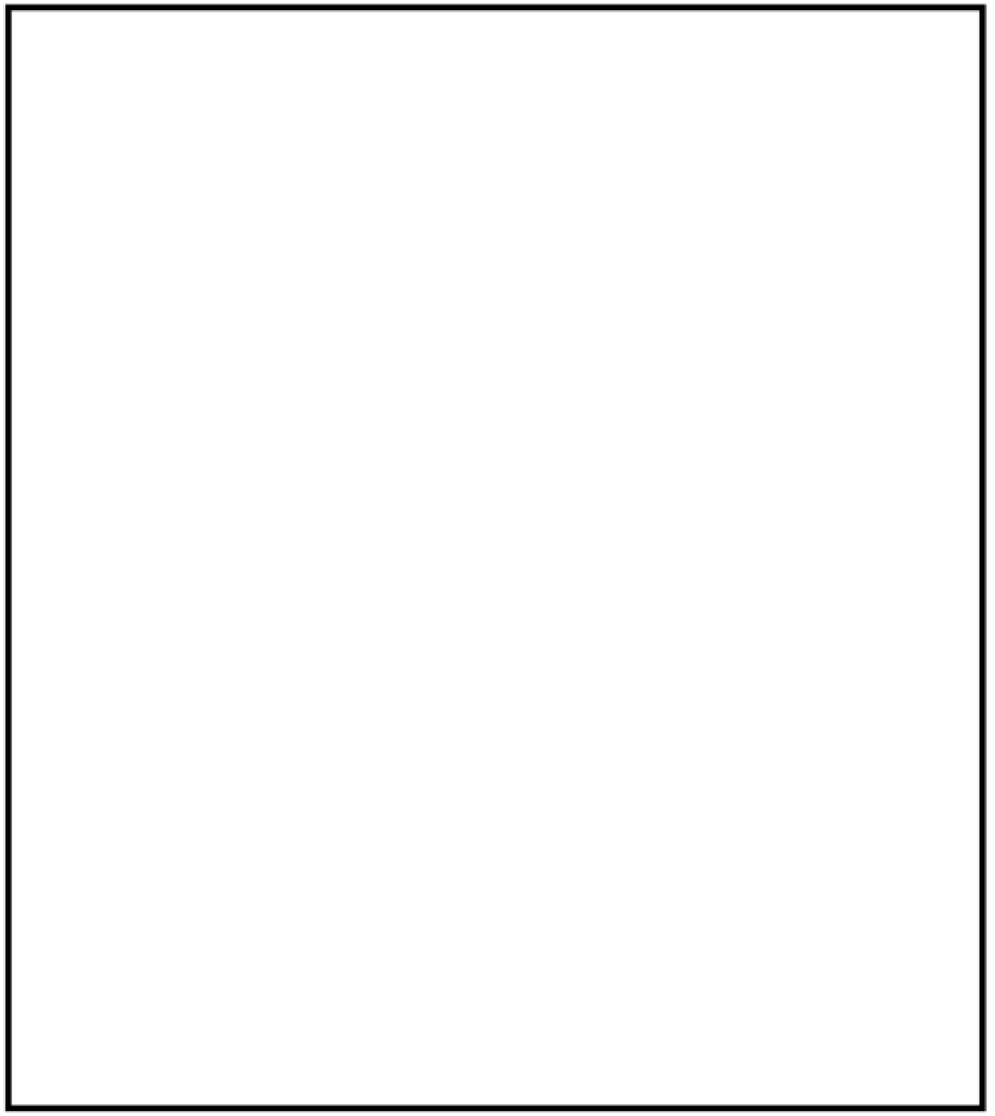
Слабая гипотеза о поколениях

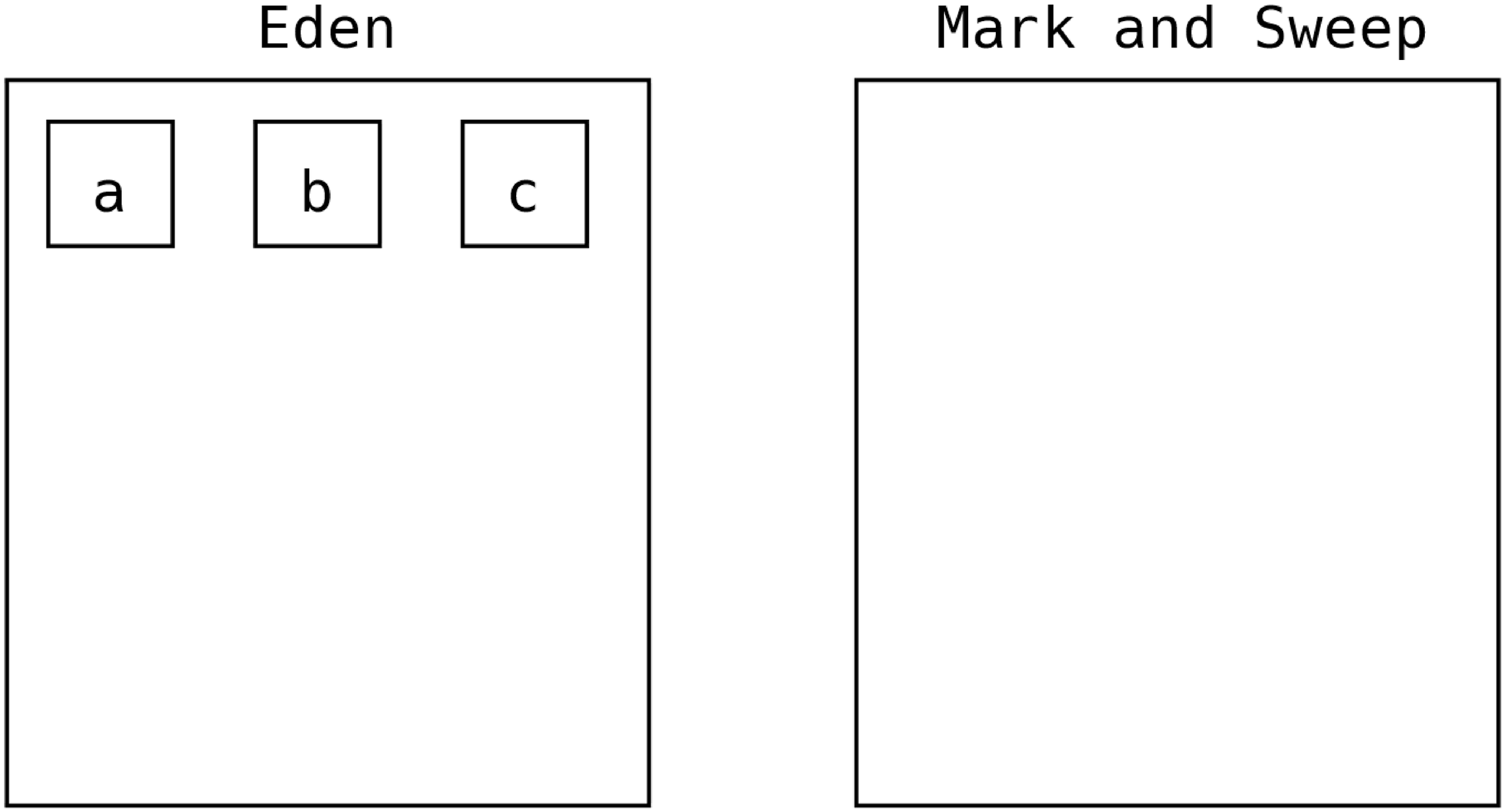
Сборка поколениями

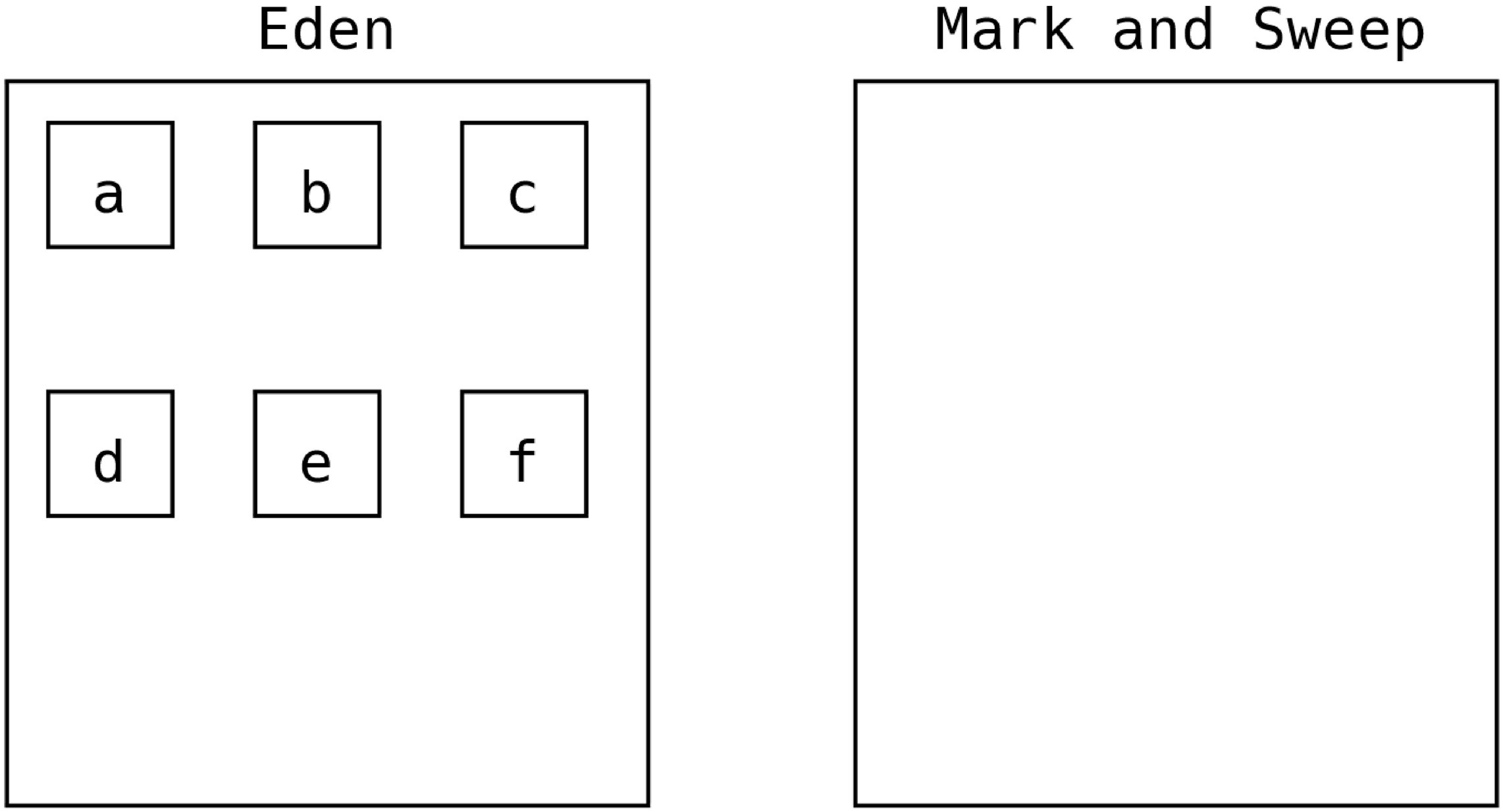
Eden

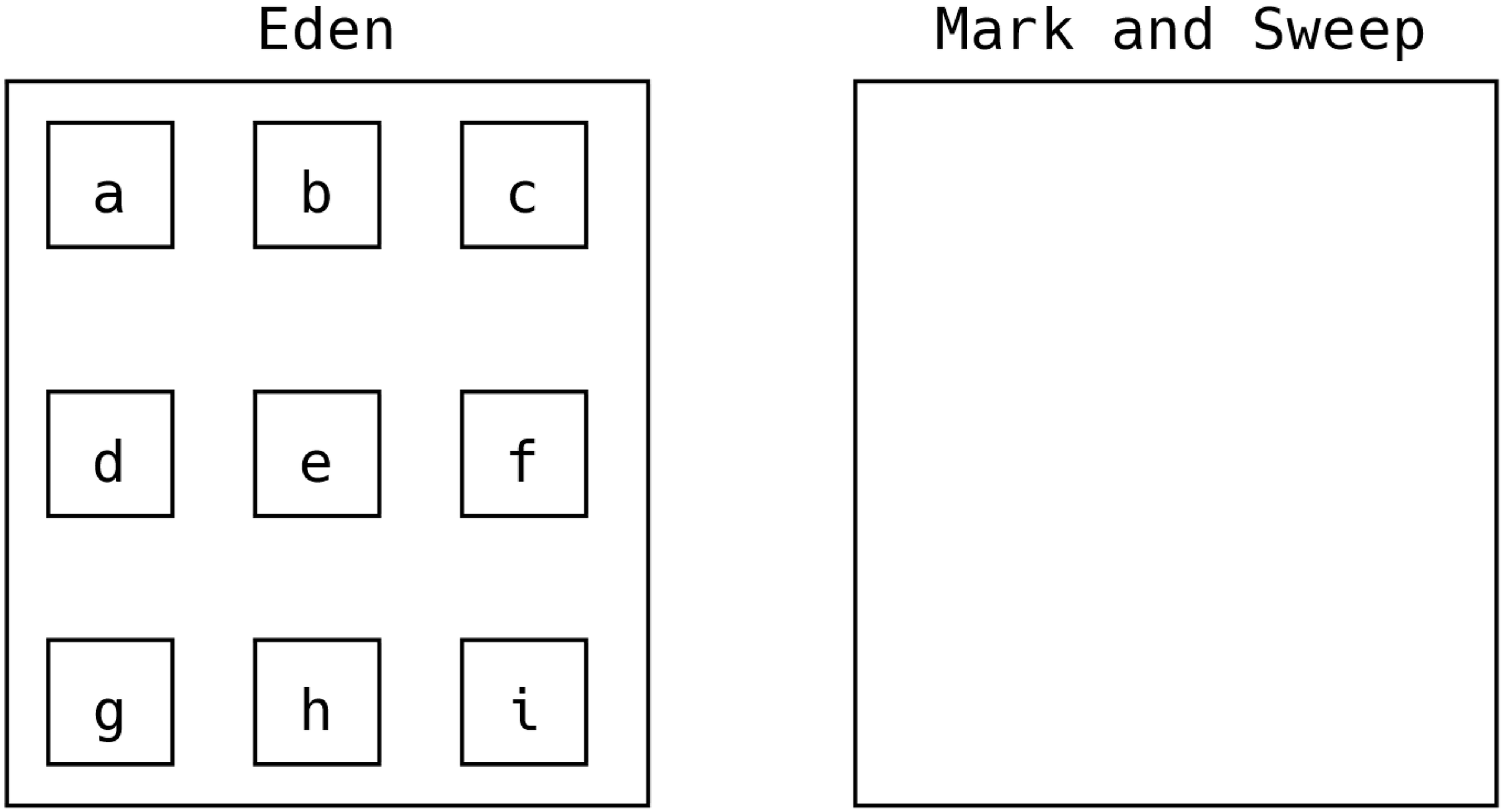


Mark and Sweep




Eden
a
b
c
Mark and Sweep

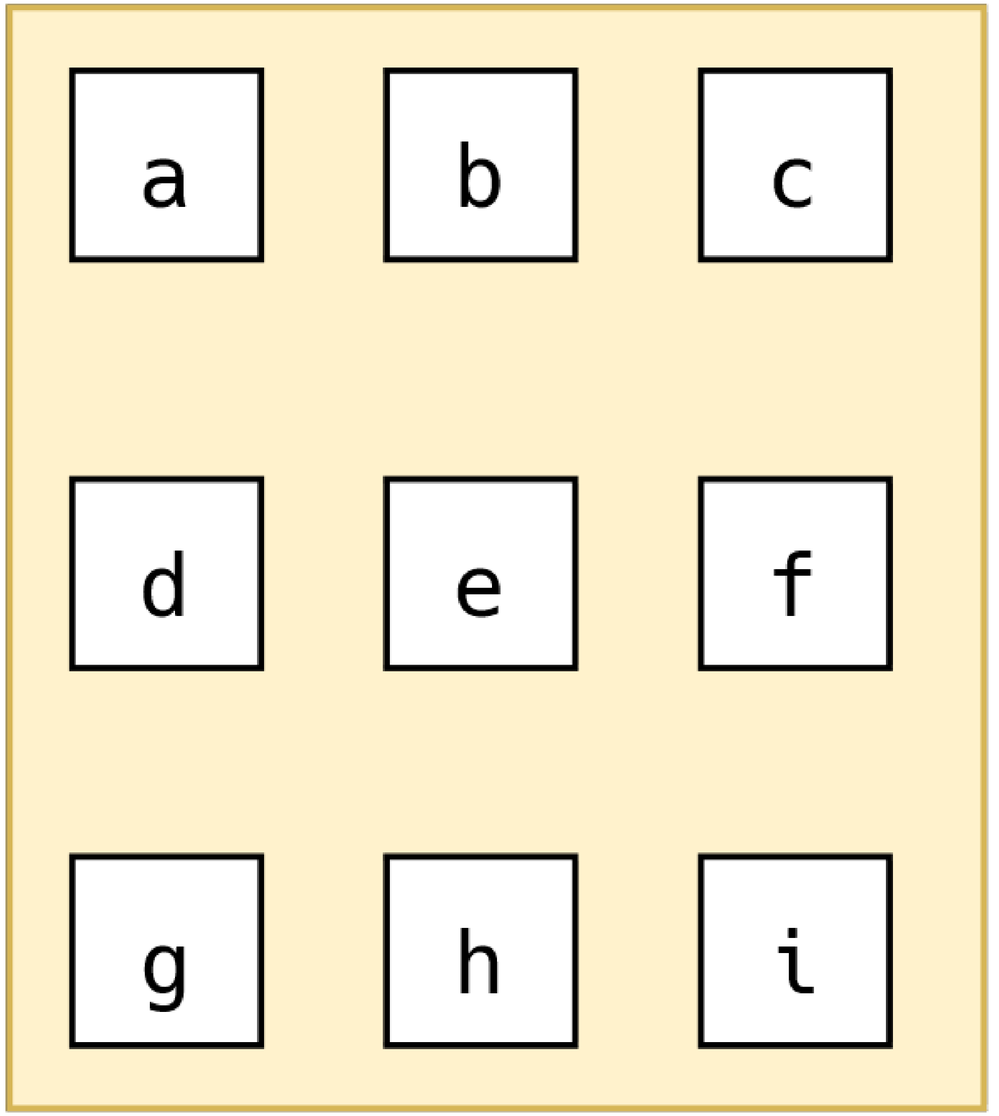

Eden
a
b
c
d
e
f
Mark and Sweep


Eden
a
b
c
d
e
f
g
h
i
Mark and Sweep

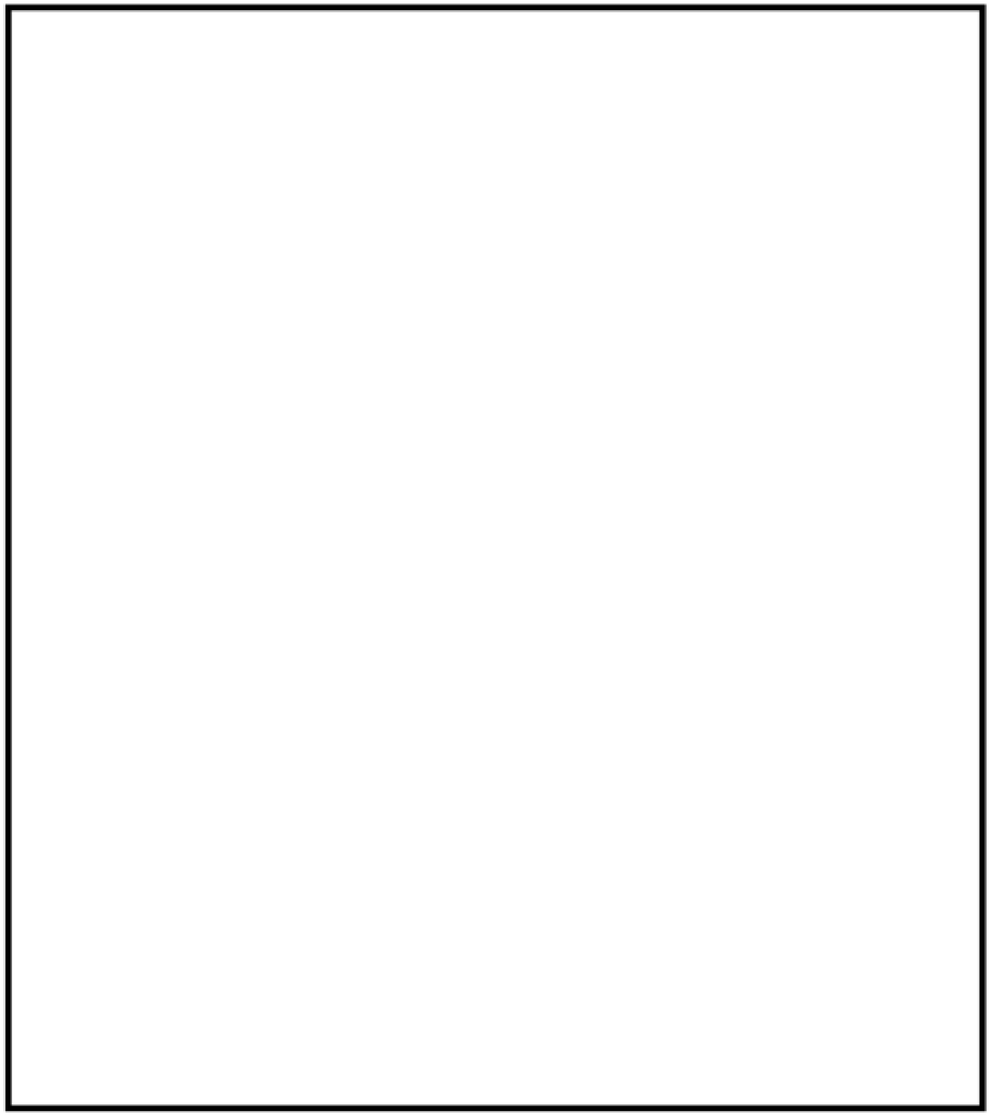
Eden

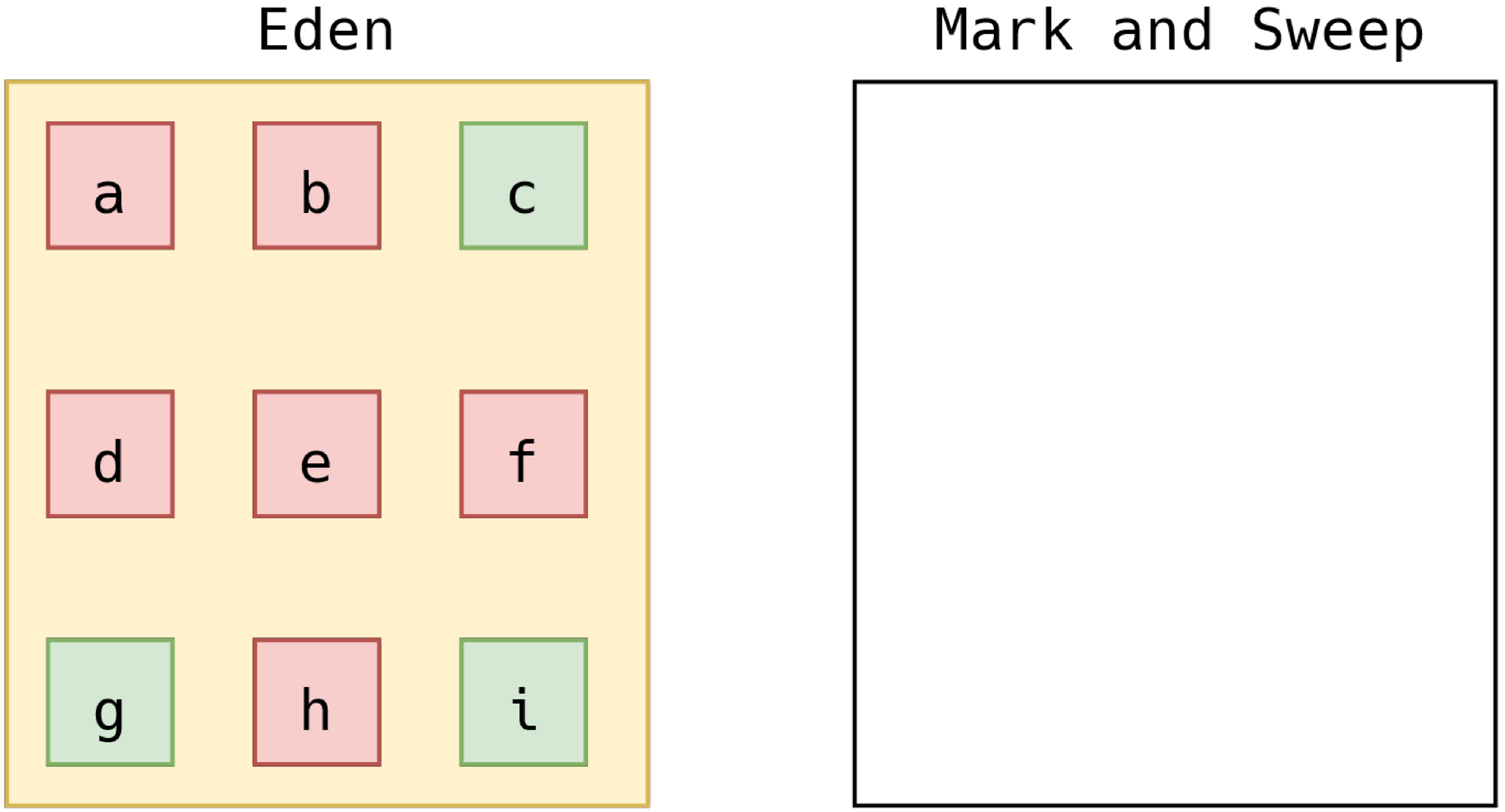
Mark and Sweep

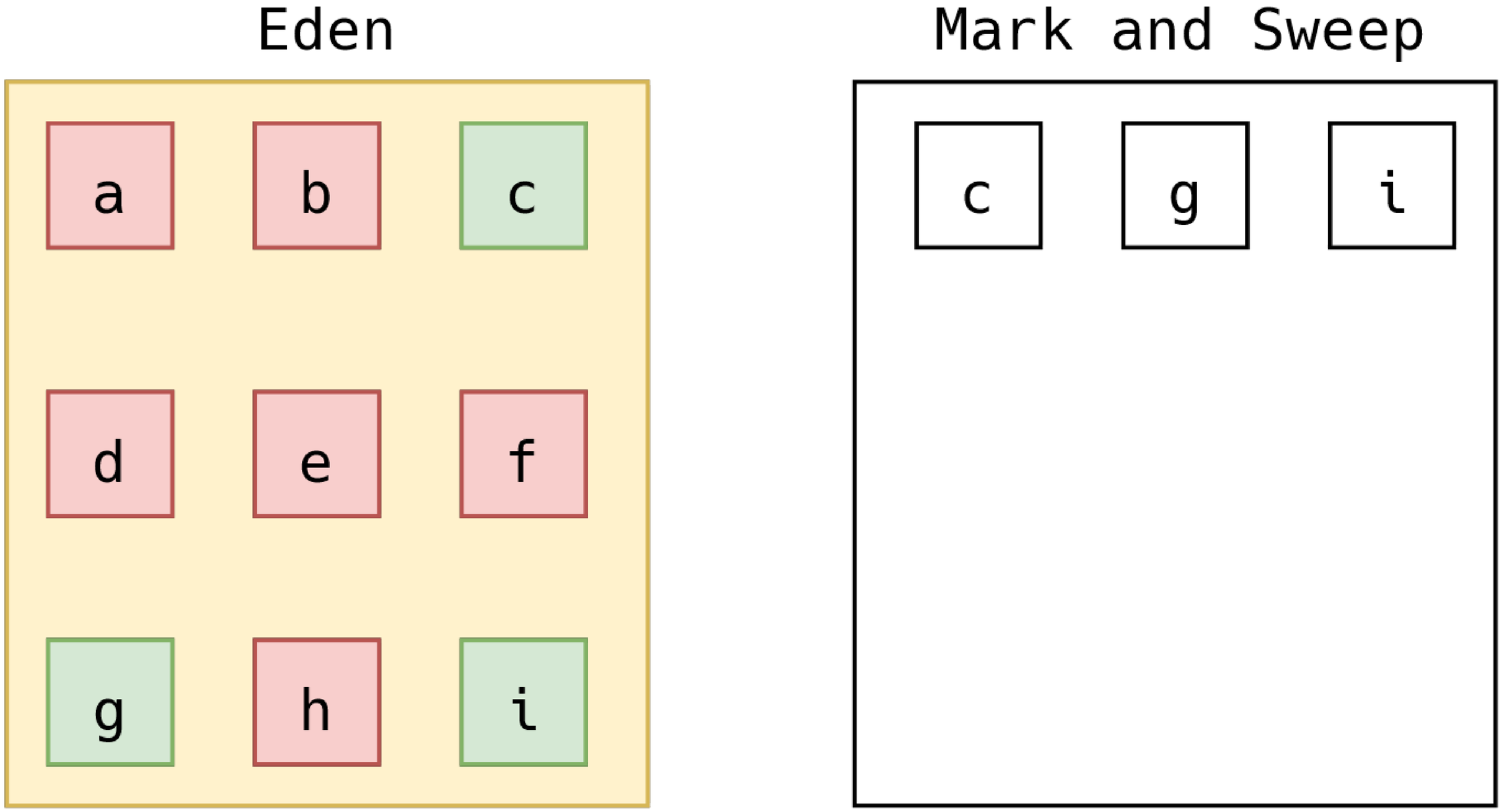
Eden



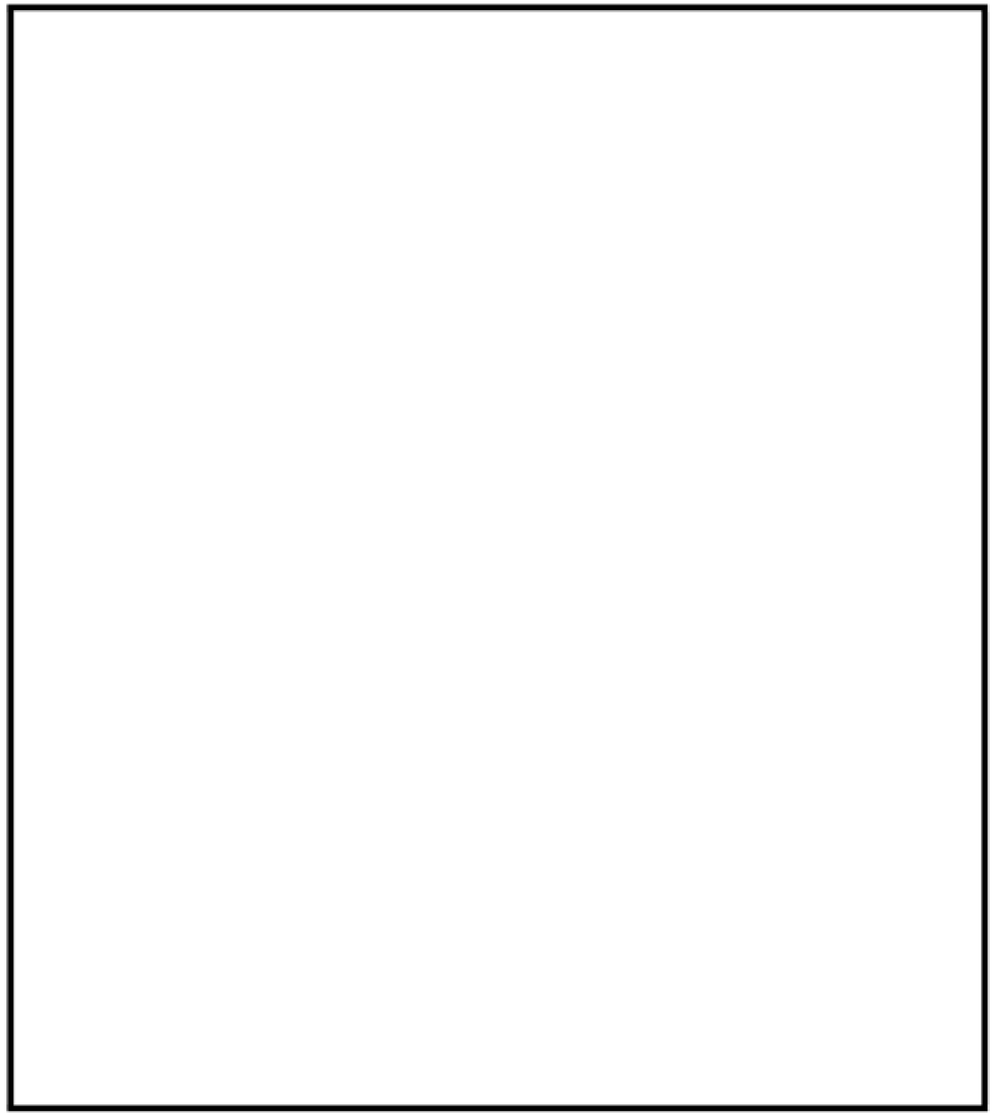
Mark and Sweep



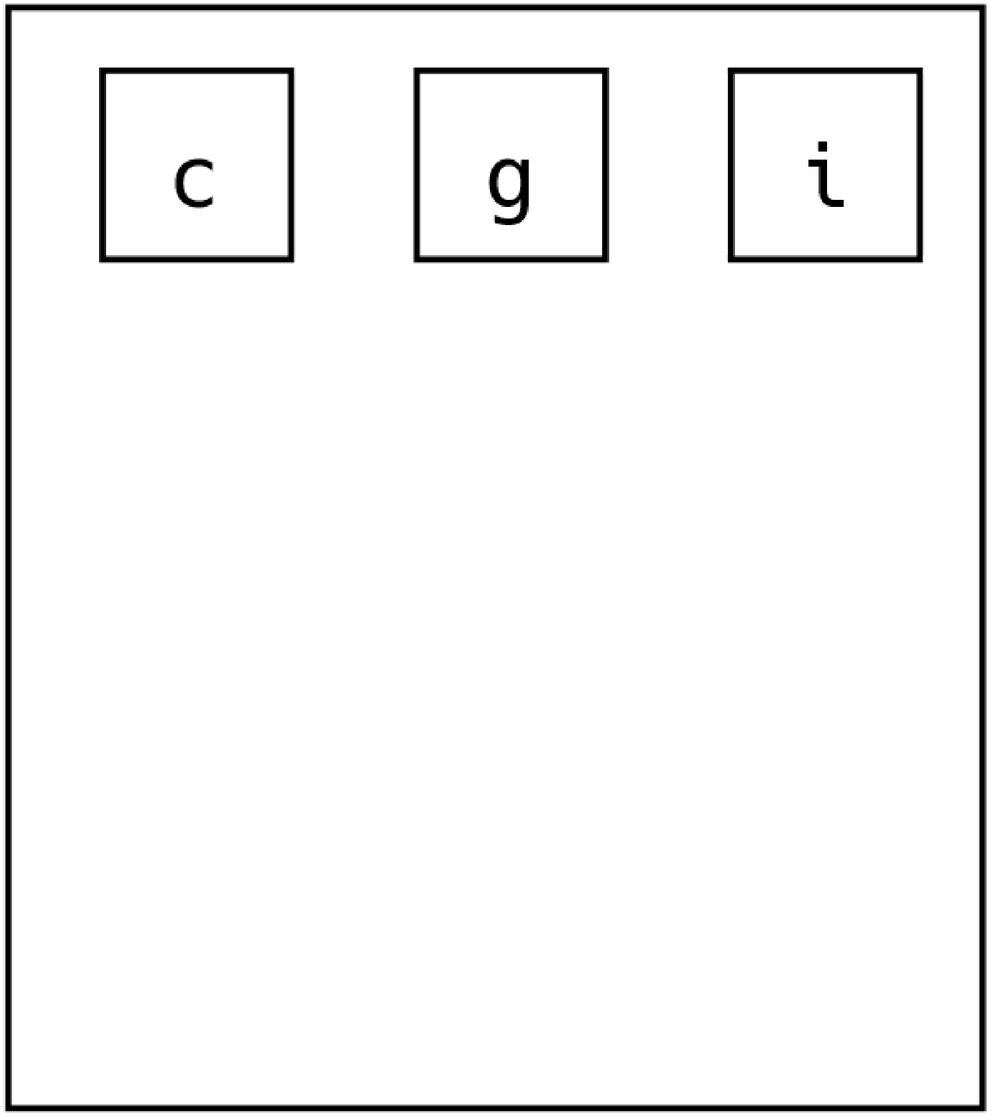

Eden
a
b
c
d
e
f
g
h
i
Mark and Sweep

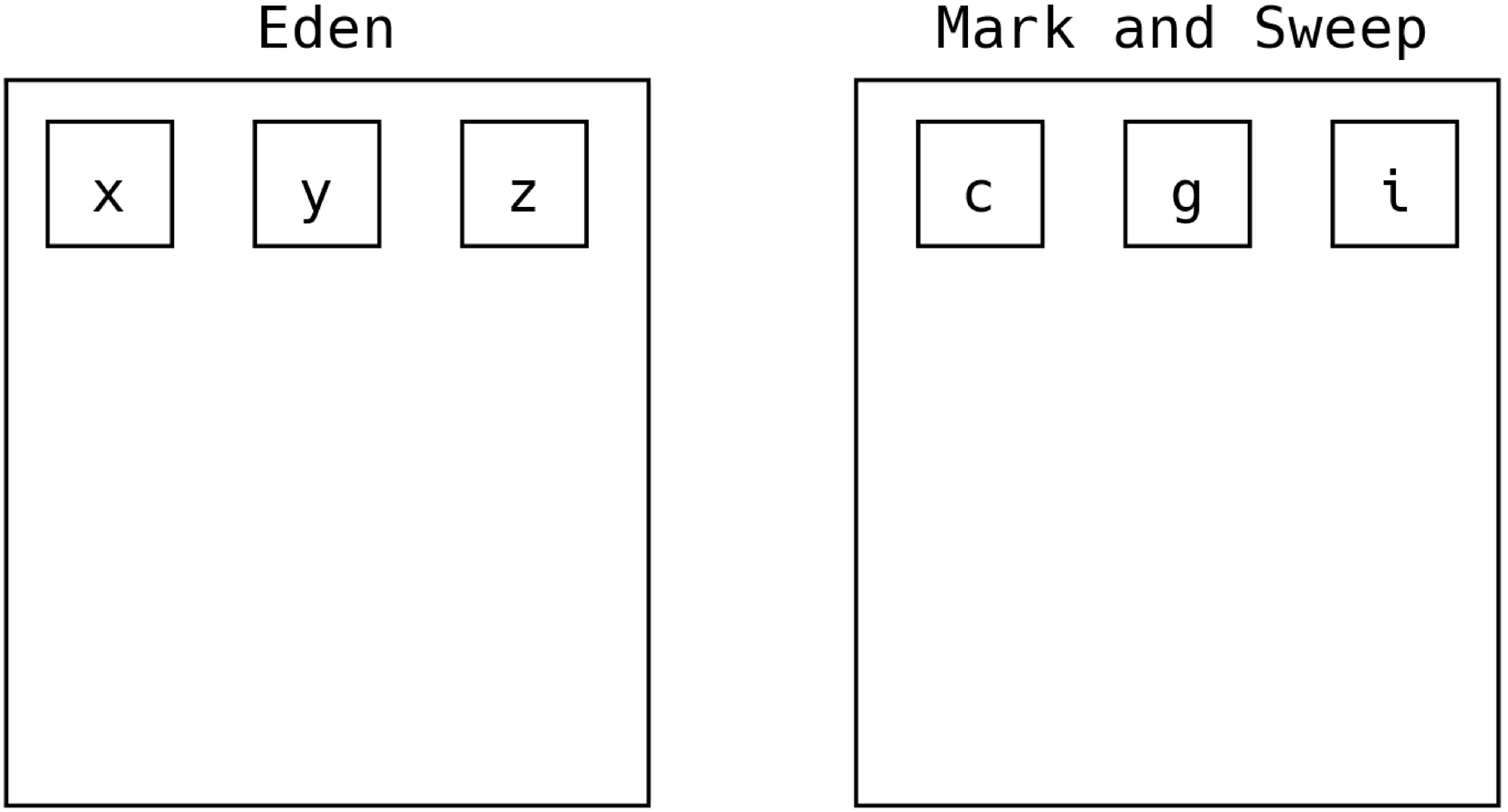

Eden
a
b
c
d
e
f
g
h
i
Mark and Sweep
c
g
i

Eden



Mark and Sweep




Eden
x
y
z
Mark and Sweep
c
g
i

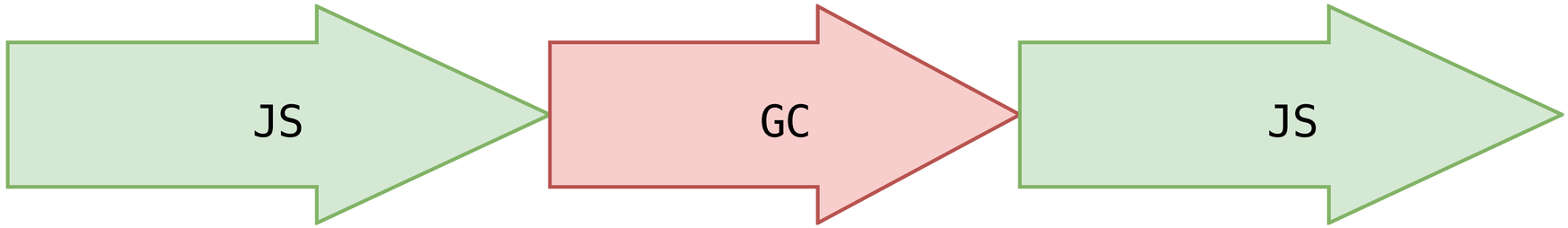
Ошибка: обман сборщика мусора

- Не создавайте долгоживущий мусор
- Сборщик начнет считать, что его не надо собирать

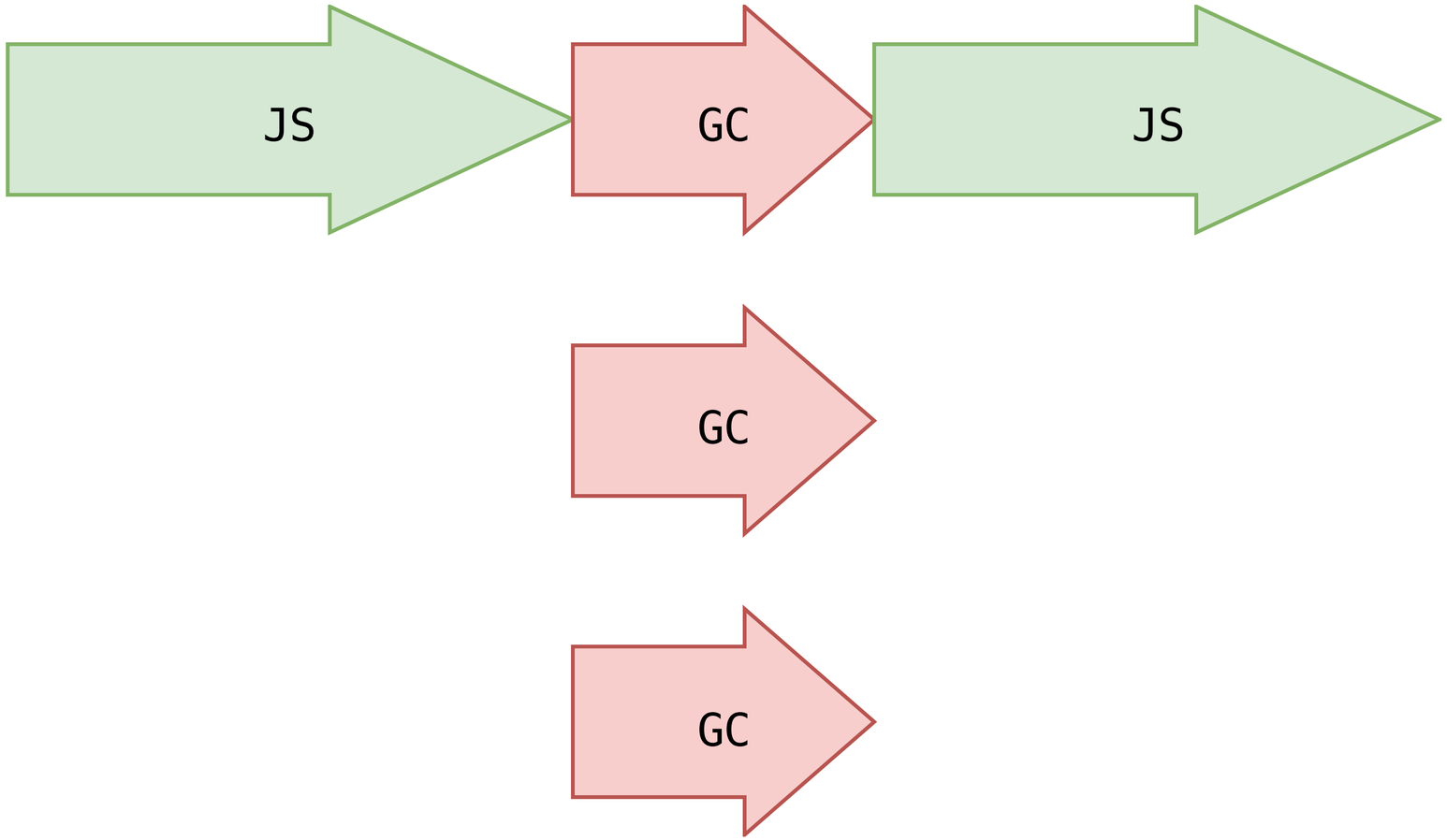
- Типичный пример обмана: LRU-cache

Когда собирать?

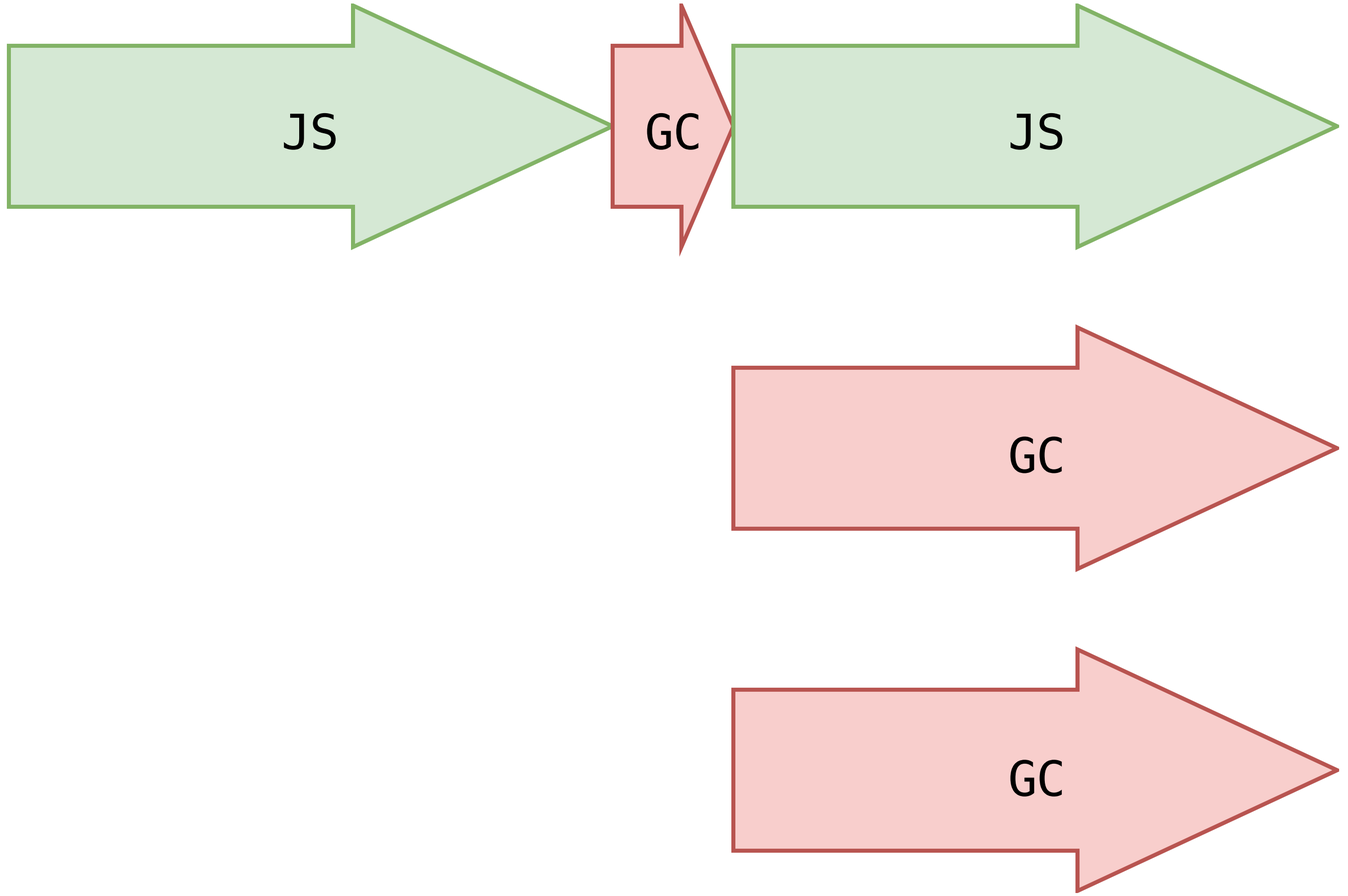
Stop the World сборщик



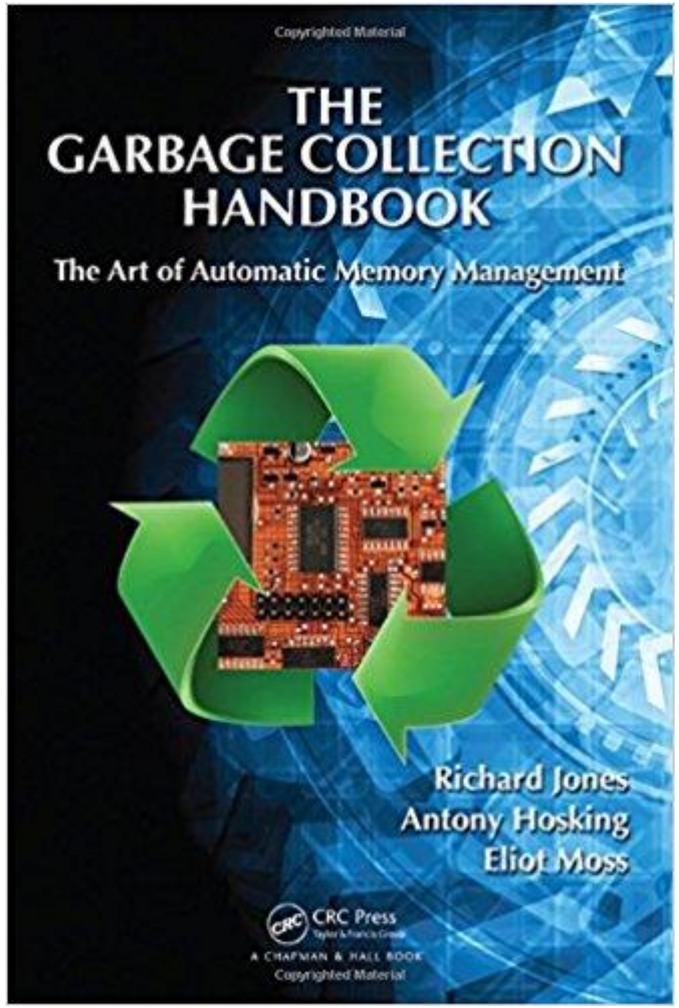
Parallel сборщик



Concurrent сборщик



Что почитать



wikipedia.org/Garbage_collection

memorymanagement.org

shipilev.net

(ну, и не только браузерная)

Браузерная реальность

IoT-движки

- Mark’n’sweep
- Stop the world
- ...
- На микроконтроллерах

JerryScript

A JavaScript engine for **Internet of Things**

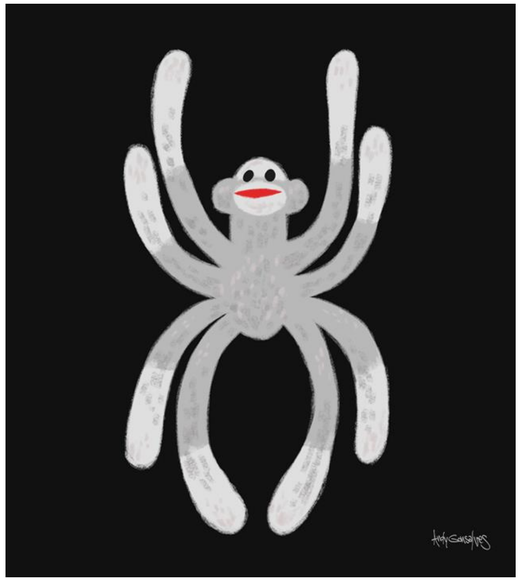
IoT-движки

- Медленный язык
- Секундные фризы
- Фрагментация
- Теперь и в вашем чайнике

JerryScript

A JavaScript engine for **Internet of Things**

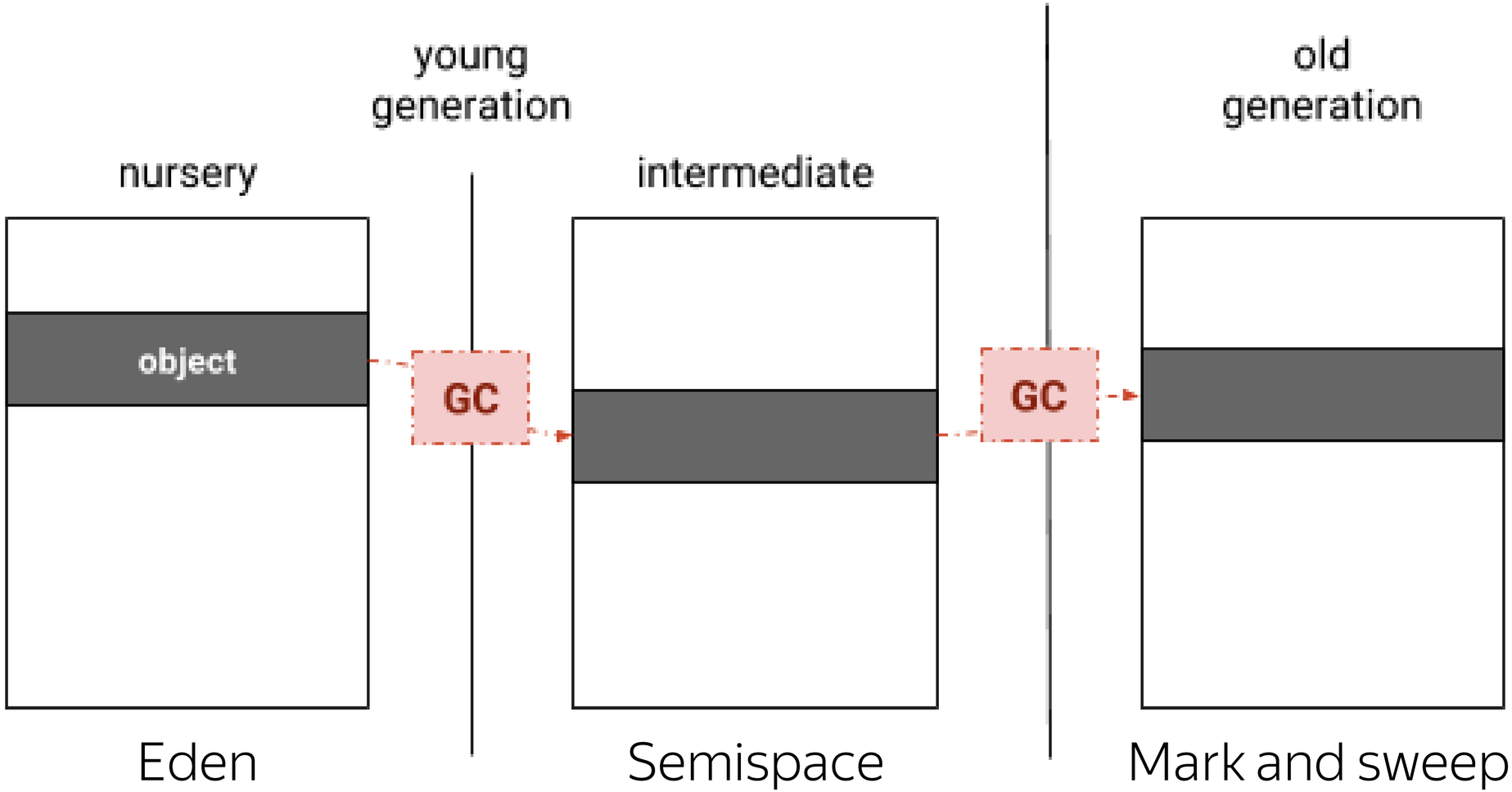
Вот эти ребята

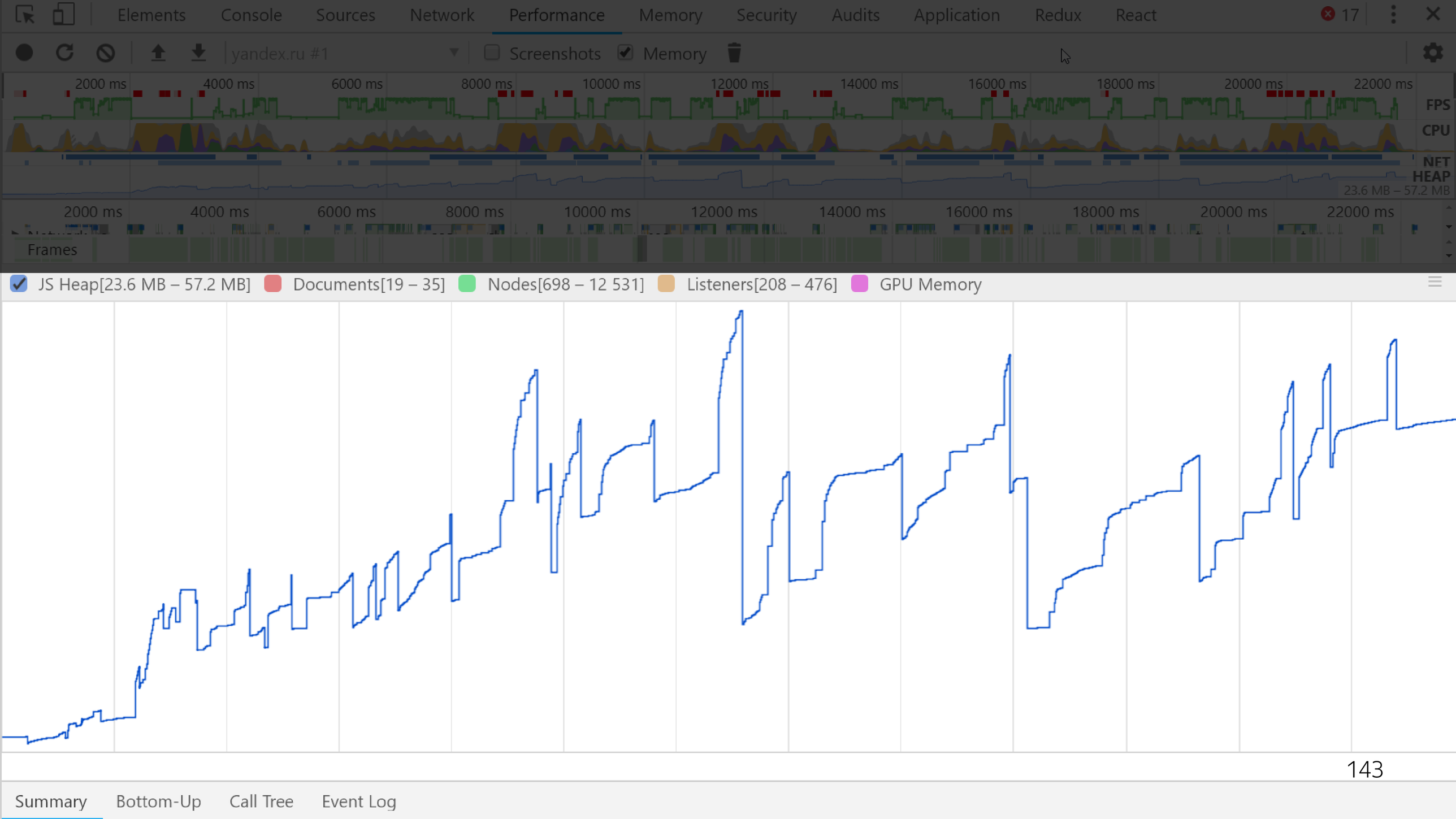


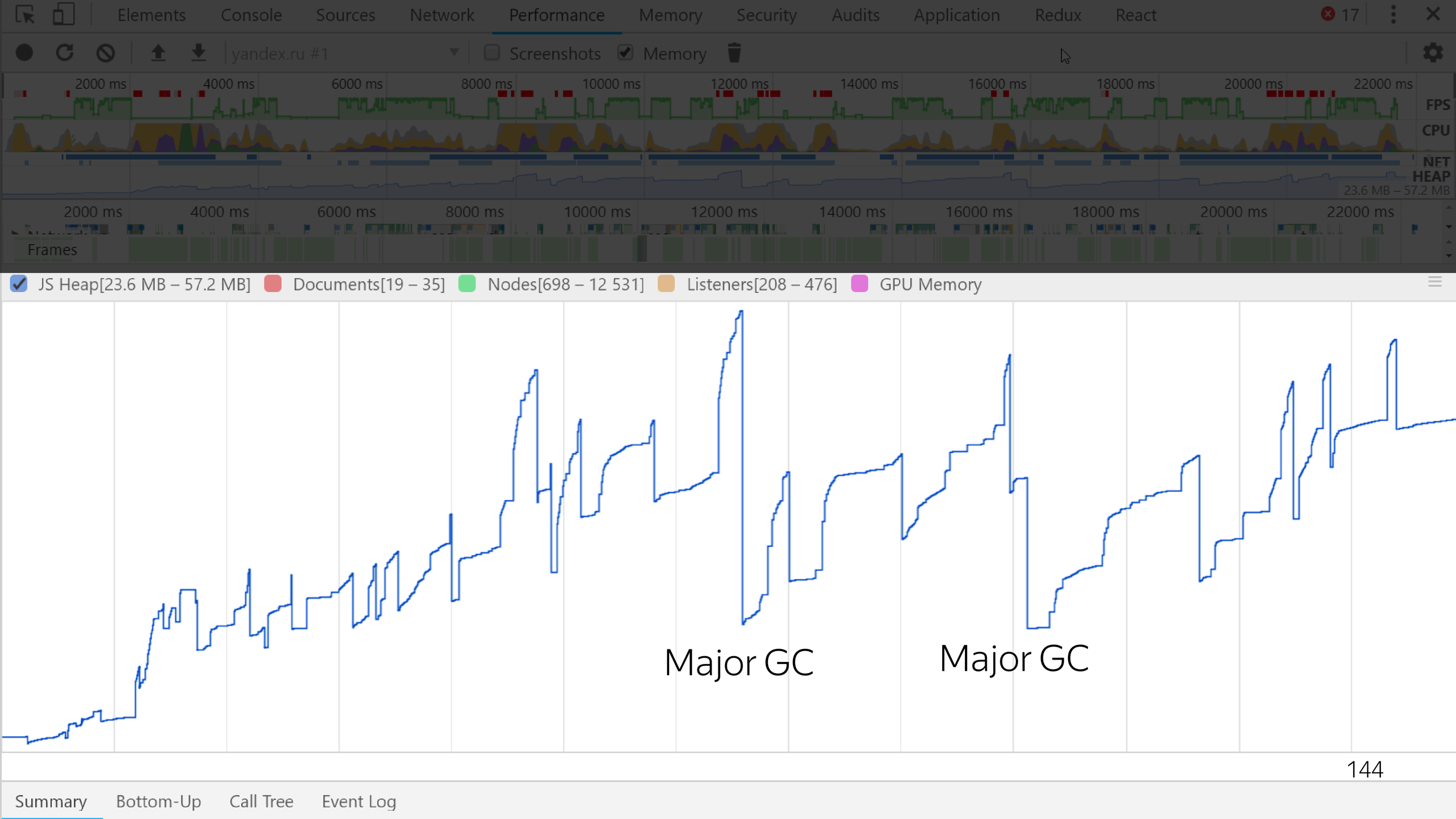
В основном, конечно, вот этот

- Почти весь серверный JavaScript
- ~80% клиентского JavaScript’а
- Больше всего информации
- Проще всего читать исходники

Сборка поколениями



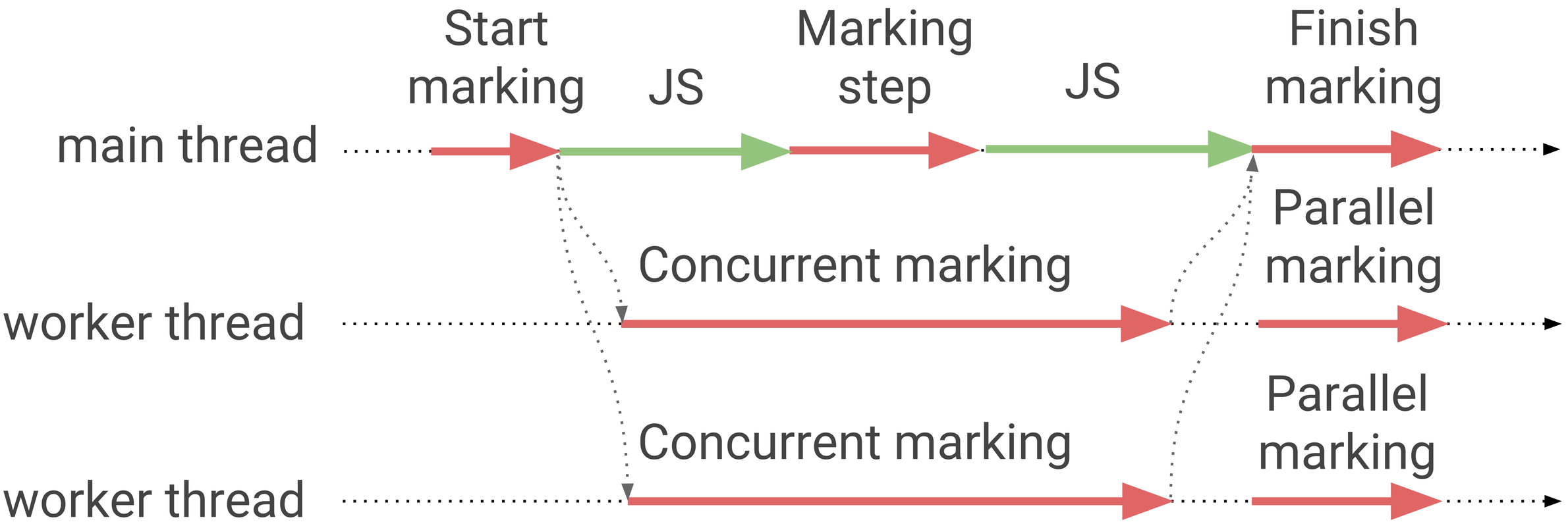

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
17
yandex.ru #1
Screenshots
Memory
2000 ms
4000 ms
6000 ms
8000 ms
10000 ms
12000 ms
14000 ms
16000 ms
18000 ms
20000 ms
22000 ms
FPS
CPU
NET
HEAP
23.6 MB – 57.2 MB
Frames
JS Heap[23.6 MB – 57.2 MB]
Documents[19 – 35]
Nodes[698 – 12 531]
Listeners[208 – 476]
GPU Memory
143
Summary
Bottom-Up
Call Tree
Event Log


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
17
yandex.ru #1
Screenshots
Memory
2000 ms
4000 ms
6000 ms
8000 ms
10000 ms
12000 ms
14000 ms
16000 ms
18000 ms
20000 ms
22000 ms
FPS
CPU
NET
HEAP
23.6 MB – 57.2 MB
Frames
JS Heap[23.6 MB – 57.2 MB]
Documents[19 – 35]
Nodes[698 – 12 531]
Listeners[208 – 476]
GPU Memory
Major GC
Major GC
144
Summary
Bottom-Up
Call Tree
Event Log

Ошибка: боязнь создания мусора

- Мусор можно создавать, если это действительно мусор
- Переиспользование объектов хуже, чем мусор

Параллельный mark



Сколько стоит все это удовольствие?

1-3%

Сколько стоит все это удовольствие?

1/100 – 1/33

Это много

Это много (для геймдева)

Это много (для геймдева)

```
const pool = [new Bullet(), new Bullet(), /* ... */];
function getFromPool() {
  const bullet = pool.find(x => !x.inUse);
  bullet.isUse = true;
  return bullet;
}
function returnToPool(bullet) { bullet.inUse = false; }

// Frame
const bullet = getFromPool();
// ...
returnToPool(bullet);
```

Это много (для геймдева)

```
const pool = [new Bullet(), new Bullet(), /* ... */];
function getFromPool() {
  const bullet = pool.find(x => !x.inUse);
  bullet.isUse = true;
  return bullet;
}
function returnToPool(bullet) { bullet.inUse = false; }

// Frame
const bullet = getFromPool();
// ...
returnToPool(bullet);
```

Статистика сборщика мусора: Chromium

```
> performance.memory

MemoryInfo {

  totalJSHeapSize: 10000000,

  usedJSHeapSize: 10000000,

  jsHeapSizeLimit: 2330000000

}
```

Статистика сборщика мусора: Node

```
> process.memoryUsage()

{ rss: 22839296,

  heapTotal: 10207232,

  heapUsed: 5967968,

  external: 12829 }
```

Будущее: WeakRef

Ссылки, которые могут быть
собраны в случае нехватки памяти

github.com/tc39/proposal-weakrefs

npmjs.com/package/weak

```
let cached = new WeakRef(myJson);

// 2 часа спустя

let json = cached.deref();
if (!json) {
  json = await fetchAgain();
}
```

Будущее: GC в WebAssembly

https://github.com/WebAssembly/gc

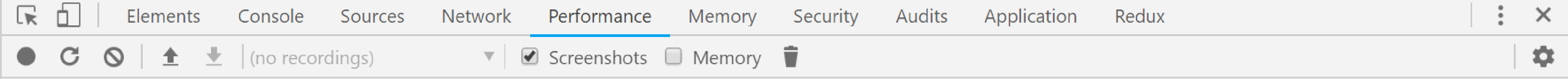
Что почитать

- v8.dev

Исходники

- github.com/v8/v8/tree/7.0.237/src/heap
- github.com/servo/mozjs/blob/master/mozjs/js/src/gc/
- github.com/WebKit/webkit/.../JavaScriptCore/heap/MarkedSpace.cpp
- github.com/Microsoft/ChakraCore/.../HeapAllocator.cpp
- github.com/svaarala/duktape/.../duk_heap_markandsweep.c
- github.com/jerryscript-project/jerryscript/.../ecma-gc.c

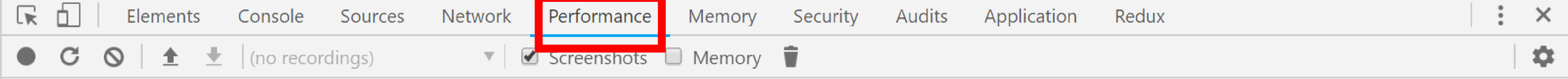
Повседневность



Click the record button ● or hit **Ctrl** + **E** to start a new recording.

Click the reload button ↻ or hit **Ctrl** + **Shift** + **E** to record the page load.

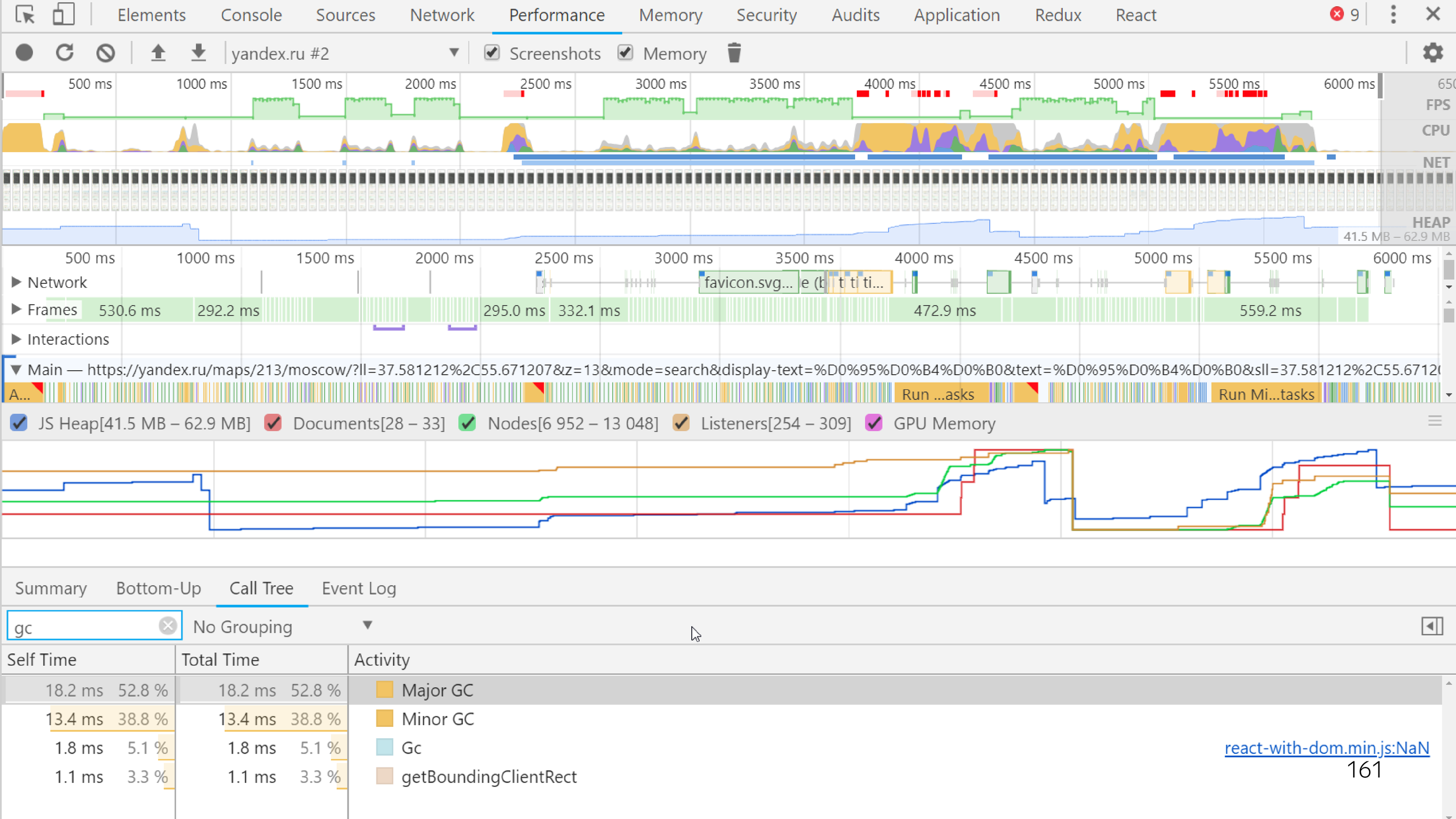
After recording, select an area of interest in the overview by dragging.
Then, zoom and pan the timeline with the mousewheel or **WASD** keys.
Learn more

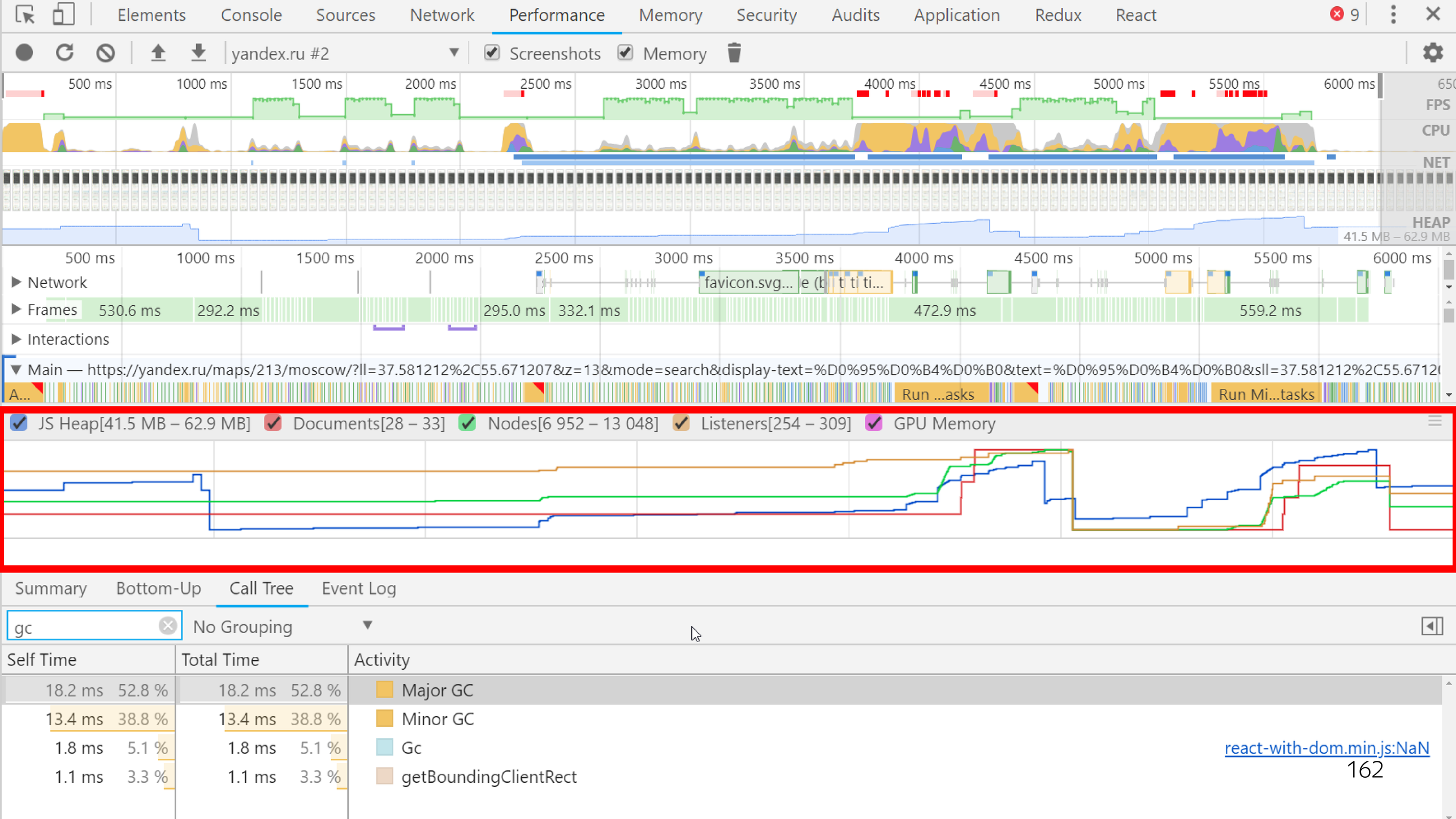


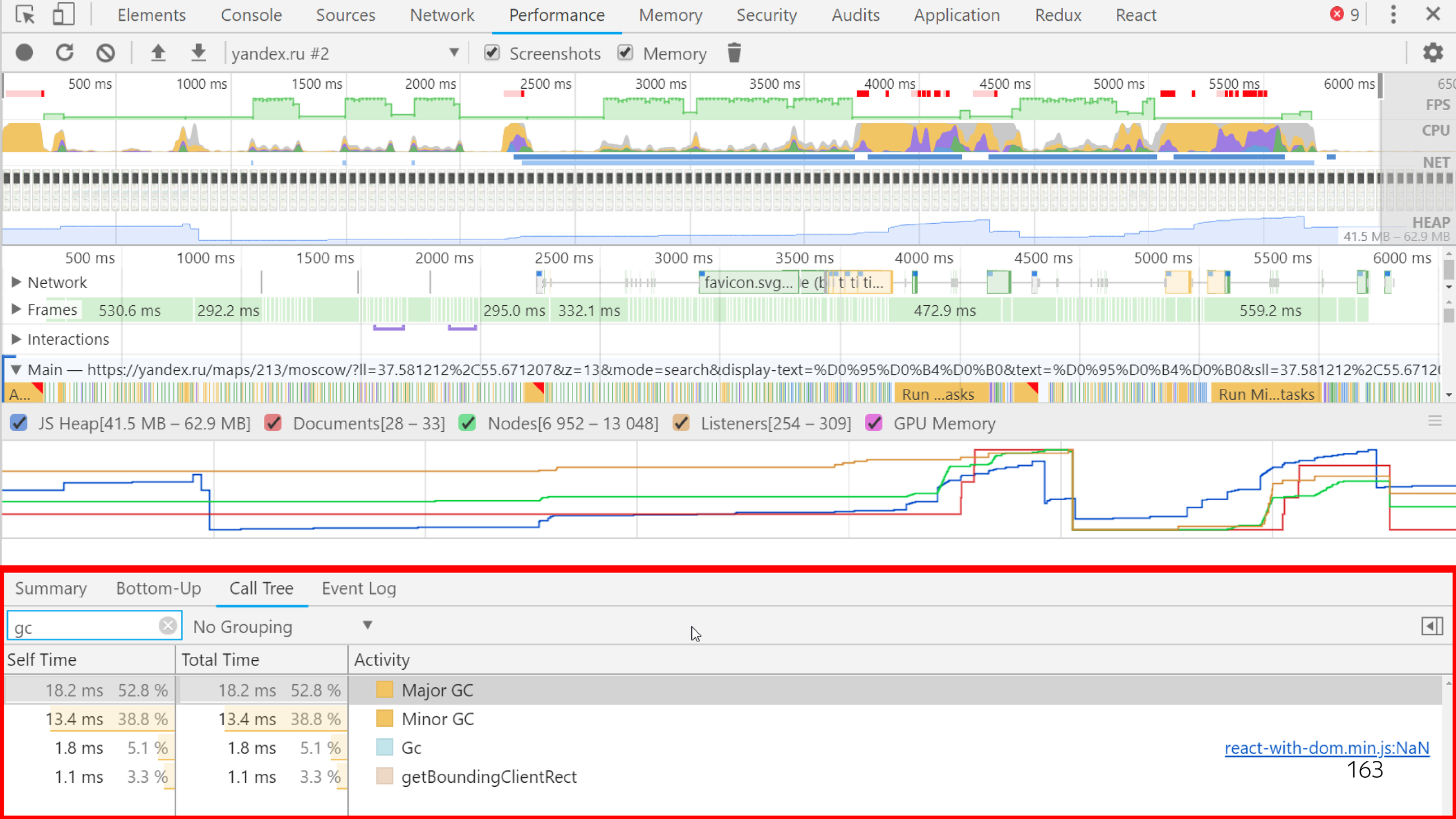
Click the record button ● or hit **Ctrl** + **E** to start a new recording.

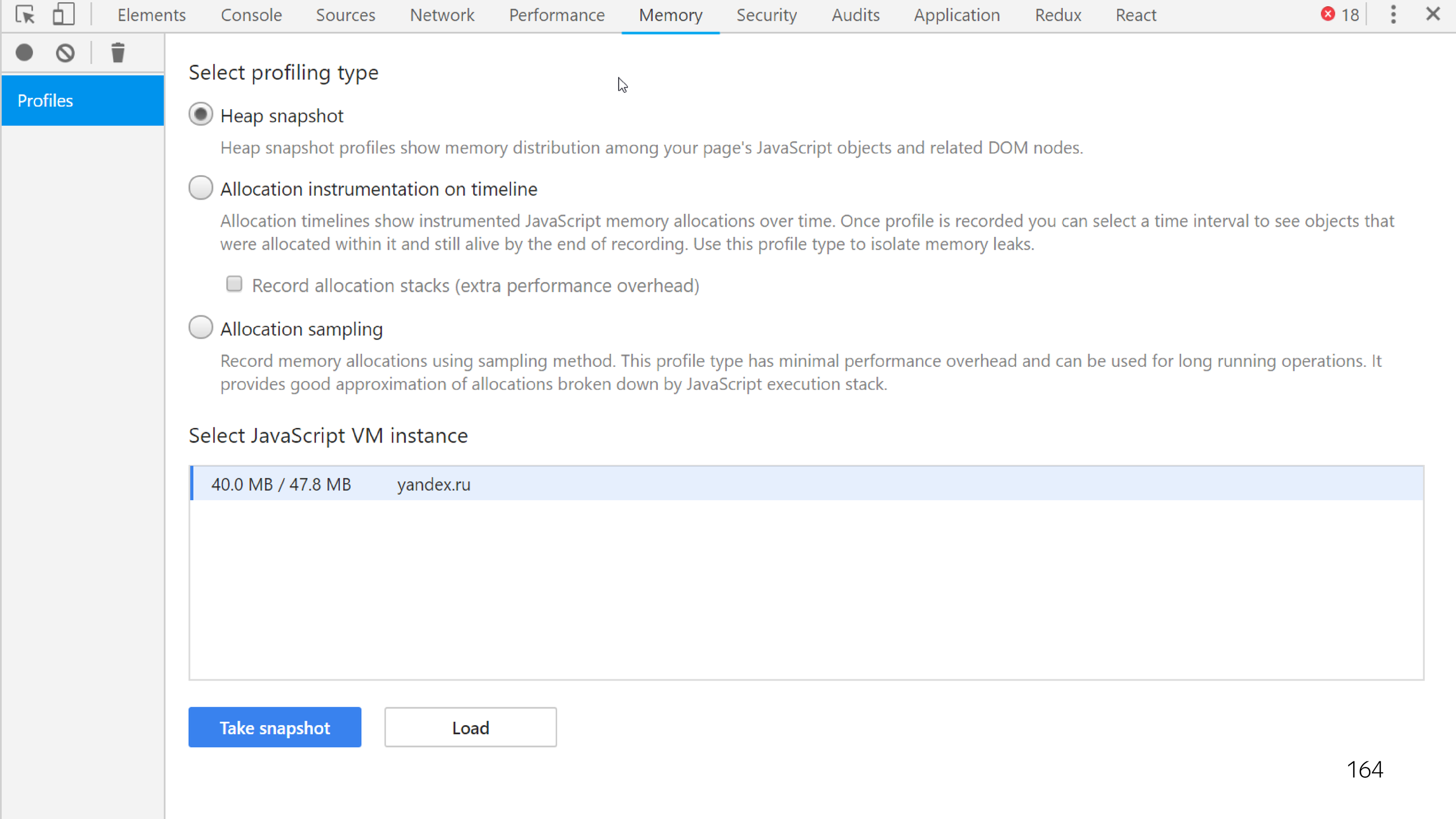
Click the reload button ↻ or hit **Ctrl** + **Shift** + **E** to record the page load.

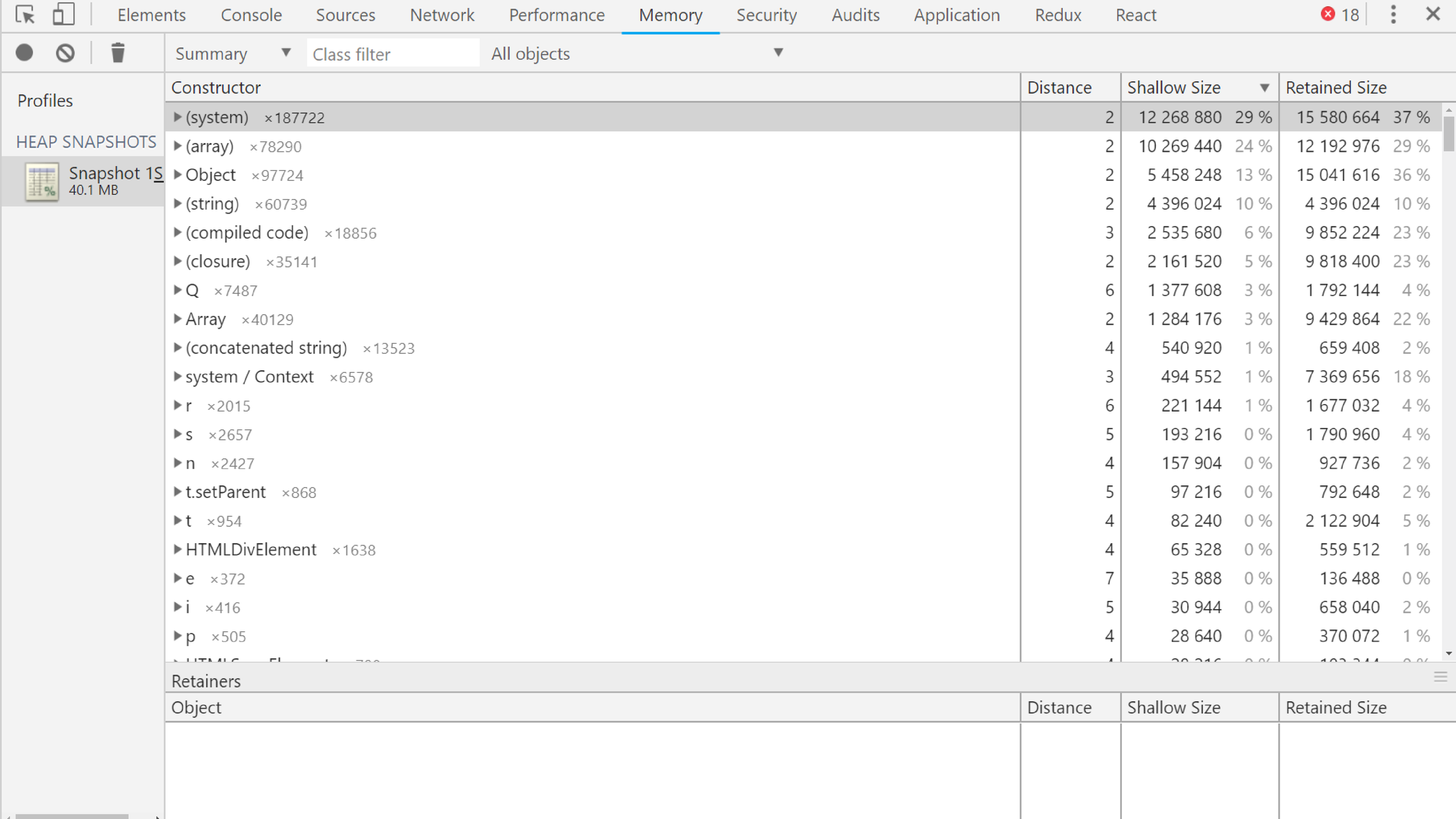
After recording, select an area of interest in the overview by dragging.
Then, zoom and pan the timeline with the mousewheel or **WASD** keys.
Learn more

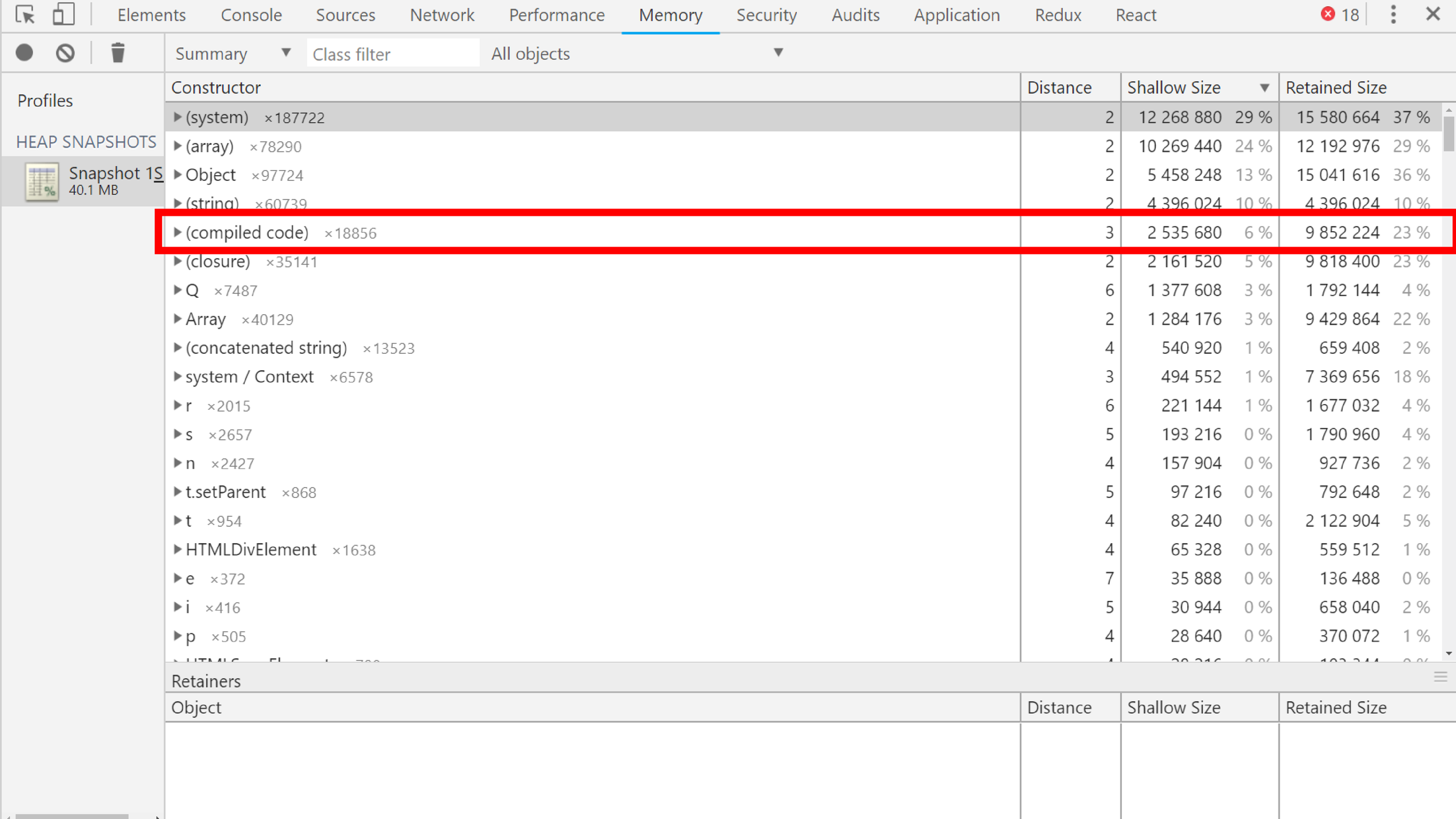

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
yandex.ru #2
Screenshots
Memory
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
5000 ms
5500 ms
6000 ms
FPS
CPU
NET
HEAP
41.5 MB – 62.9 MB
Network
favicon.svg...
Frames
530.6 ms
292.2 ms
295.0 ms
332.1 ms
472.9 ms
559.2 ms
Interactions
Main — https://yandex.ru/maps/213/moscow/?ll=37.581212%2C55.671207&z=13&mode=search&display-text=%D0%95%D0%B4%D0%B0&text=%D0%95%D0%B4%D0%B0&sll=37.581212%2C55.671
Run ...asks
Run Mi...tasks
JS Heap[41.5 MB – 62.9 MB]
Documents[28 – 33]
Nodes[6 952 – 13 048]
Listeners[254 – 309]
GPU Memory
Summary
Bottom-Up
Call Tree
Event Log
gc
No Grouping
Self Time
Total Time
Activity
18.2 ms 52.8 %
18.2 ms 52.8 %
Major GC
13.4 ms 38.8 %
13.4 ms 38.8 %
Minor GC
1.8 ms 5.1 %
1.8 ms 5.1 %
Gc
react-with-dom.min.js:NaN
1.1 ms 3.3 %
1.1 ms 3.3 %
getBoundingClientRect
161

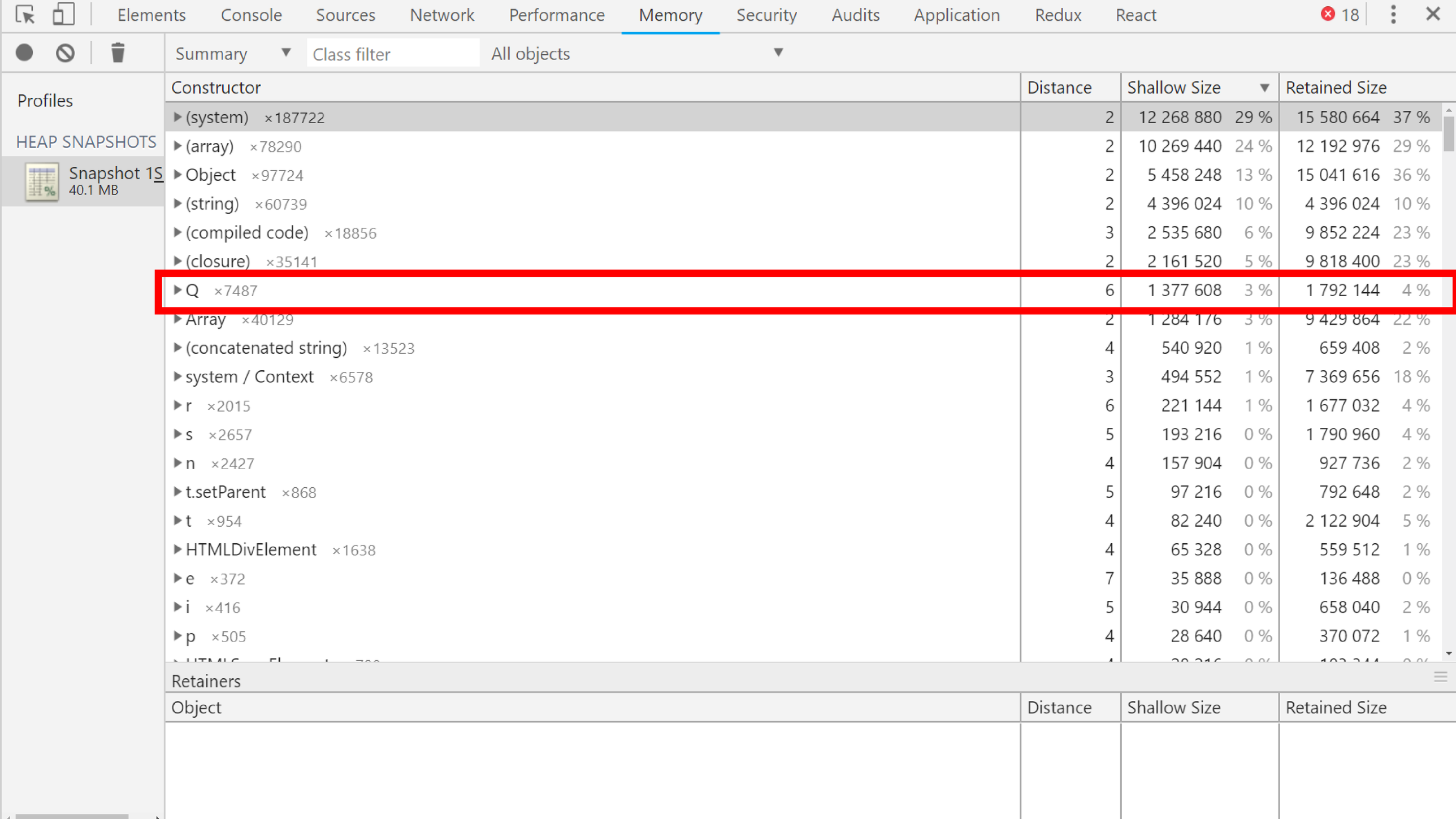

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
yandex.ru #2
Screenshots
Memory
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
5000 ms
5500 ms
6000 ms
FPS
CPU
NET
HEAP
41.5 MB – 62.9 MB
Network
favicon.svg...
Frames
530.6 ms
292.2 ms
295.0 ms
332.1 ms
472.9 ms
559.2 ms
Interactions
Main — https://yandex.ru/maps/213/moscow/?ll=37.581212%2C55.671207&z=13&mode=search&display-text=%D0%95%D0%B4%D0%B0&text=%D0%95%D0%B4%D0%B0&sll=37.581212%2C55.671
Run ...asks
Run Mi...tasks
JS Heap[41.5 MB – 62.9 MB]
Documents[28 – 33]
Nodes[6 952 – 13 048]
Listeners[254 – 309]
GPU Memory
Summary
Bottom-Up
Call Tree
Event Log
gc
No Grouping
Self Time
Total Time
Activity
18.2 ms 52.8 %
18.2 ms 52.8 %
Major GC
13.4 ms 38.8 %
13.4 ms 38.8 %
Minor GC
1.8 ms 5.1 %
1.8 ms 5.1 %
Gc
react-with-dom.min.js:NaN
1.1 ms 3.3 %
1.1 ms 3.3 %
getBoundingClientRect
162

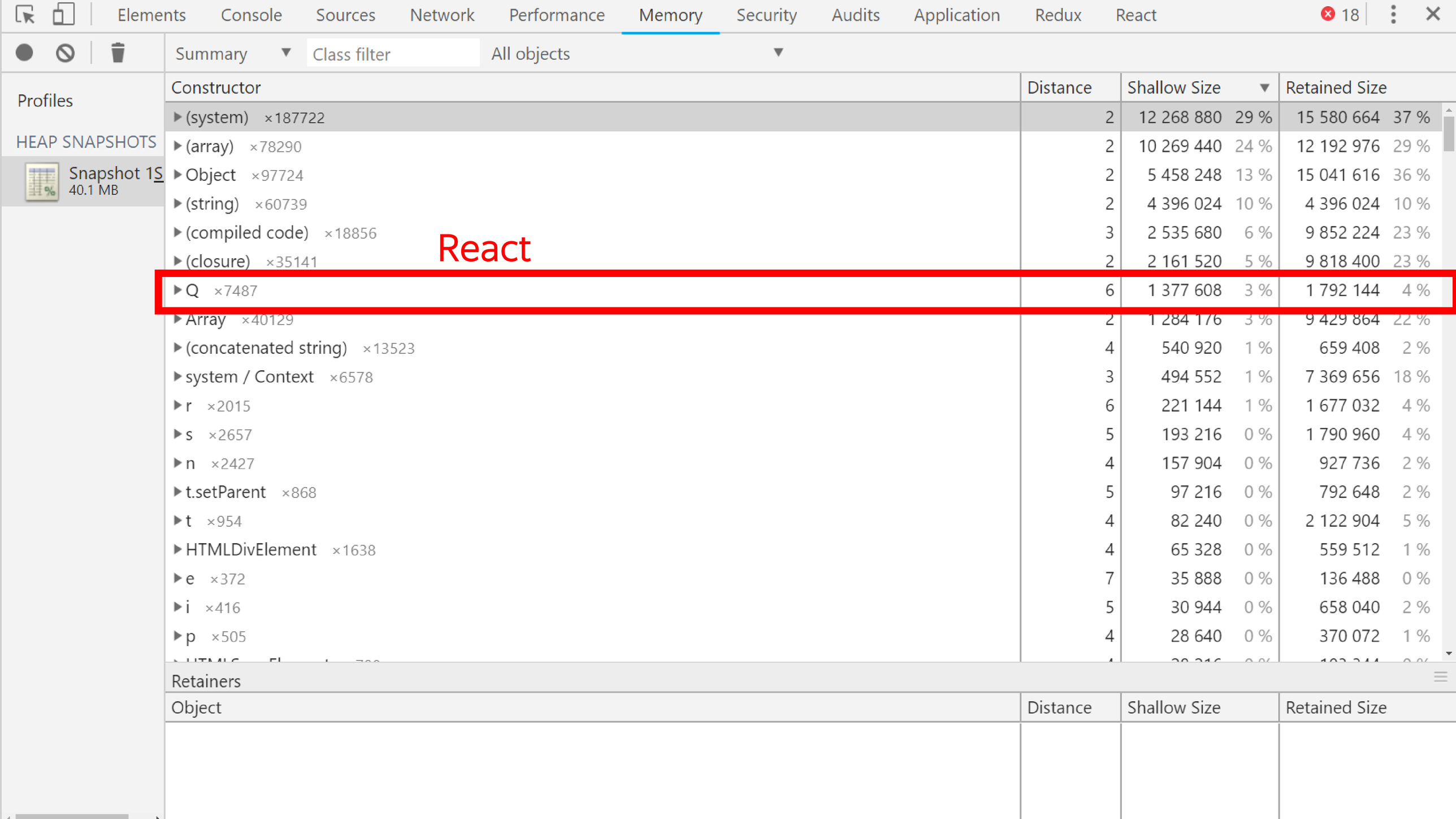

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
yandex.ru #2
Screenshots
Memory
500 ms
1000 ms
1500 ms
2000 ms
2500 ms
3000 ms
3500 ms
4000 ms
4500 ms
5000 ms
5500 ms
6000 ms
FPS
CPU
NET
HEAP
41.5 MB – 62.9 MB
Network
favicon.svg...
Frames
530.6 ms
292.2 ms
295.0 ms
332.1 ms
472.9 ms
559.2 ms
Interactions
Main — https://yandex.ru/maps/213/moscow/?ll=37.581212%2C55.671207&z=13&mode=search&display-text=%D0%95%D0%B4%D0%B0&text=%D0%95%D0%B4%D0%B0&sll=37.581212%2C55.671
Run ...asks
Run Mi...tasks
JS Heap[41.5 MB – 62.9 MB]
Documents[28 – 33]
Nodes[6 952 – 13 048]
Listeners[254 – 309]
GPU Memory
Summary
Bottom-Up
Call Tree
Event Log
gc
No Grouping
Self Time
Total Time
Activity
18.2 ms 52.8 %
18.2 ms 52.8 %
Major GC
13.4 ms 38.8 %
13.4 ms 38.8 %
Minor GC
1.8 ms 5.1 %
1.8 ms 5.1 %
Gc
react-with-dom.min.js:NaN
1.1 ms 3.3 %
1.1 ms 3.3 %
getBoundingClientRect
163

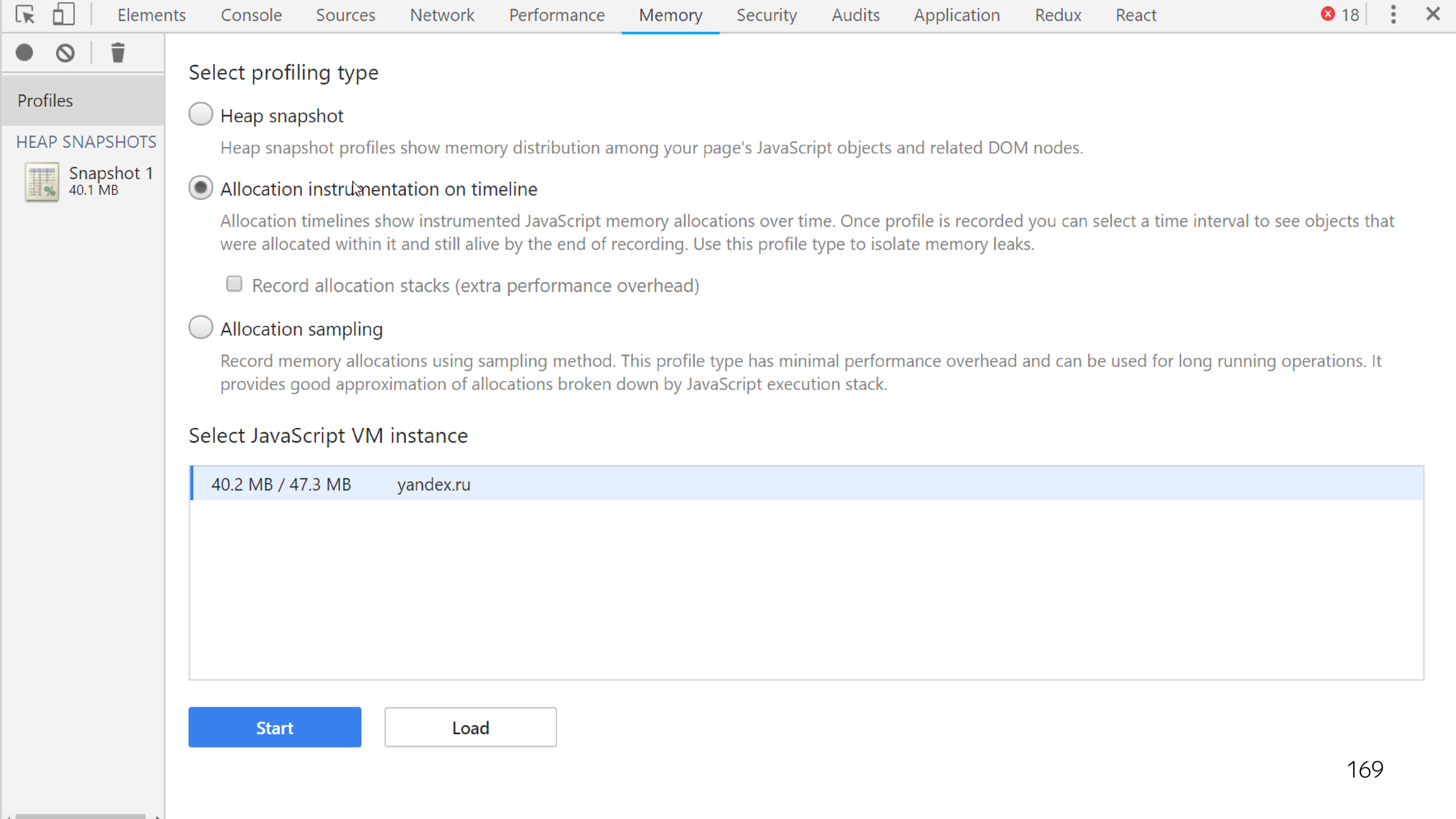

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Profiles
Select profiling type
Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
Record allocation stacks (extra performance overhead)
Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.
Select JavaScript VM instance
40.0 MB / 47.8 MB
yandex.ru
Take snapshot
Load

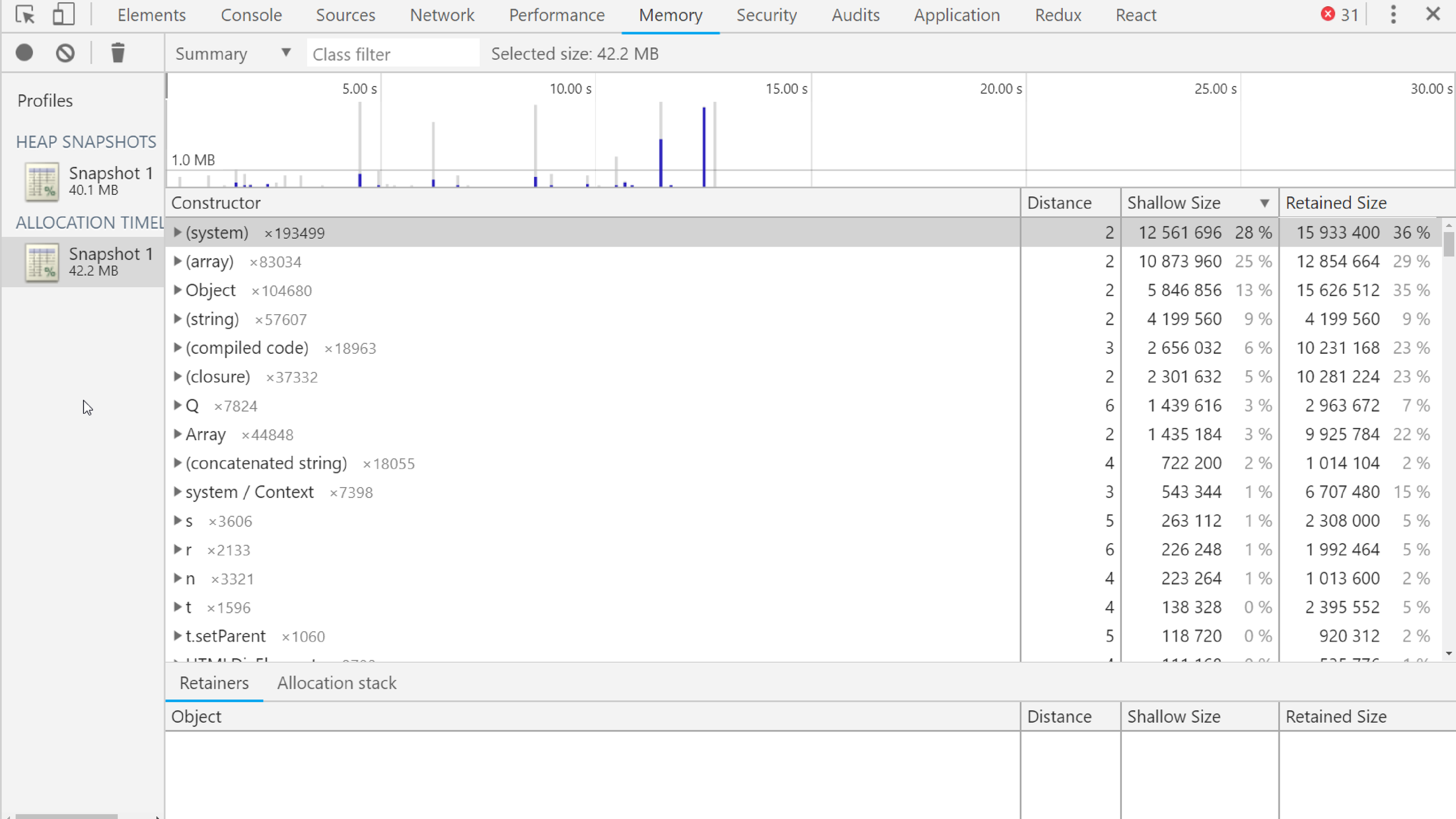

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1S
40.1 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×187722 2 12 268 880 29 % 15 580 664 37 %
(array) ×78290 2 10 269 440 24 % 12 192 976 29 %
Object ×97724 2 5 458 248 13 % 15 041 616 36 %
(string) ×60739 2 4 396 024 10 % 4 396 024 10 %
(compiled code) ×18856 3 2 535 680 6 % 9 852 224 23 %
(closure) ×35141 2 2 161 520 5 % 9 818 400 23 %
Q ×7487 6 1 377 608 3 % 1 792 144 4 %
Array ×40129 2 1 284 176 3 % 9 429 864 22 %
(concatenated string) ×13523 4 540 920 1 % 659 408 2 %
system / Context ×6578 3 494 552 1 % 7 369 656 18 %
r ×2015 6 221 144 1 % 1 677 032 4 %
s ×2657 5 193 216 0 % 1 790 960 4 %
n ×2427 4 157 904 0 % 927 736 2 %
t.setParent ×868 5 97 216 0 % 792 648 2 %
t ×954 4 82 240 0 % 2 122 904 5 %
HTMLDivElement ×1638 4 65 328 0 % 559 512 1 %
e ×372 7 35 888 0 % 136 488 0 %
i ×416 5 30 944 0 % 658 040 2 %
p ×505 4 28 640 0 % 370 072 1 %
Retainers
Object
Distance
Shallow Size
Retained Size

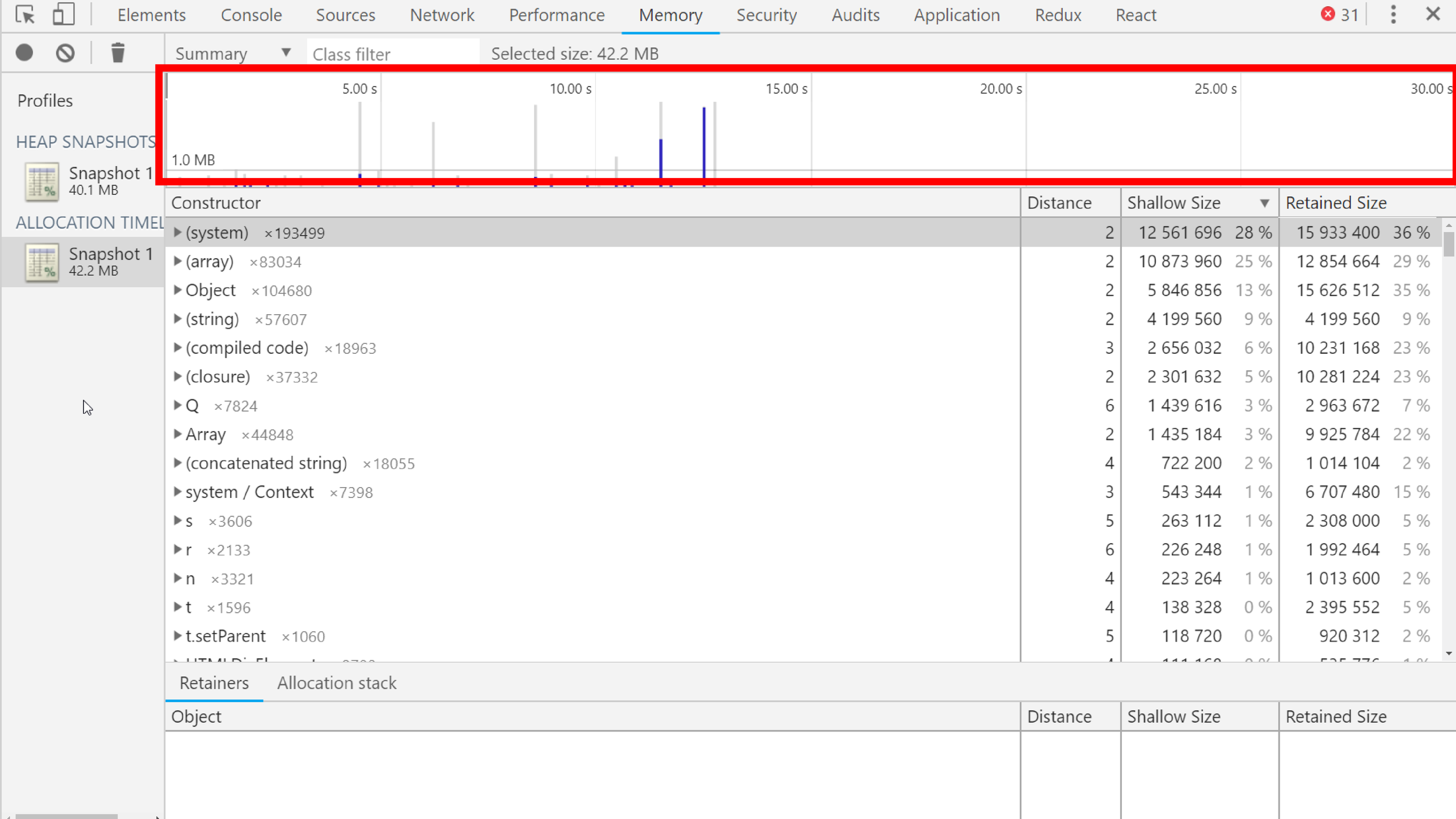

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1S
40.1 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×187722 2 12 268 880 29 % 15 580 664 37 %
(array) ×78290 2 10 269 440 24 % 12 192 976 29 %
Object ×97724 2 5 458 248 13 % 15 041 616 36 %
(string) ×60739 2 4 396 024 10 % 4 396 024 10 %
(compiled code) ×18856 3 2 535 680 6 % 9 852 224 23 %
(closure) ×35141 2 2 161 520 5 % 9 818 400 23 %
Q ×7487 6 1 377 608 3 % 1 792 144 4 %
Array ×40129 2 1 284 176 3 % 9 429 864 22 %
(concatenated string) ×13523 4 540 920 1 % 659 408 2 %
system / Context ×6578 3 494 552 1 % 7 369 656 18 %
r ×2015 6 221 144 1 % 1 677 032 4 %
s ×2657 5 193 216 0 % 1 790 960 4 %
n ×2427 4 157 904 0 % 927 736 2 %
t.setParent ×868 5 97 216 0 % 792 648 2 %
t ×954 4 82 240 0 % 2 122 904 5 %
HTMLDivElement ×1638 4 65 328 0 % 559 512 1 %
e ×372 7 35 888 0 % 136 488 0 %
i ×416 5 30 944 0 % 658 040 2 %
p ×505 4 28 640 0 % 370 072 1 %
Retainers
Object
Distance
Shallow Size
Retained Size

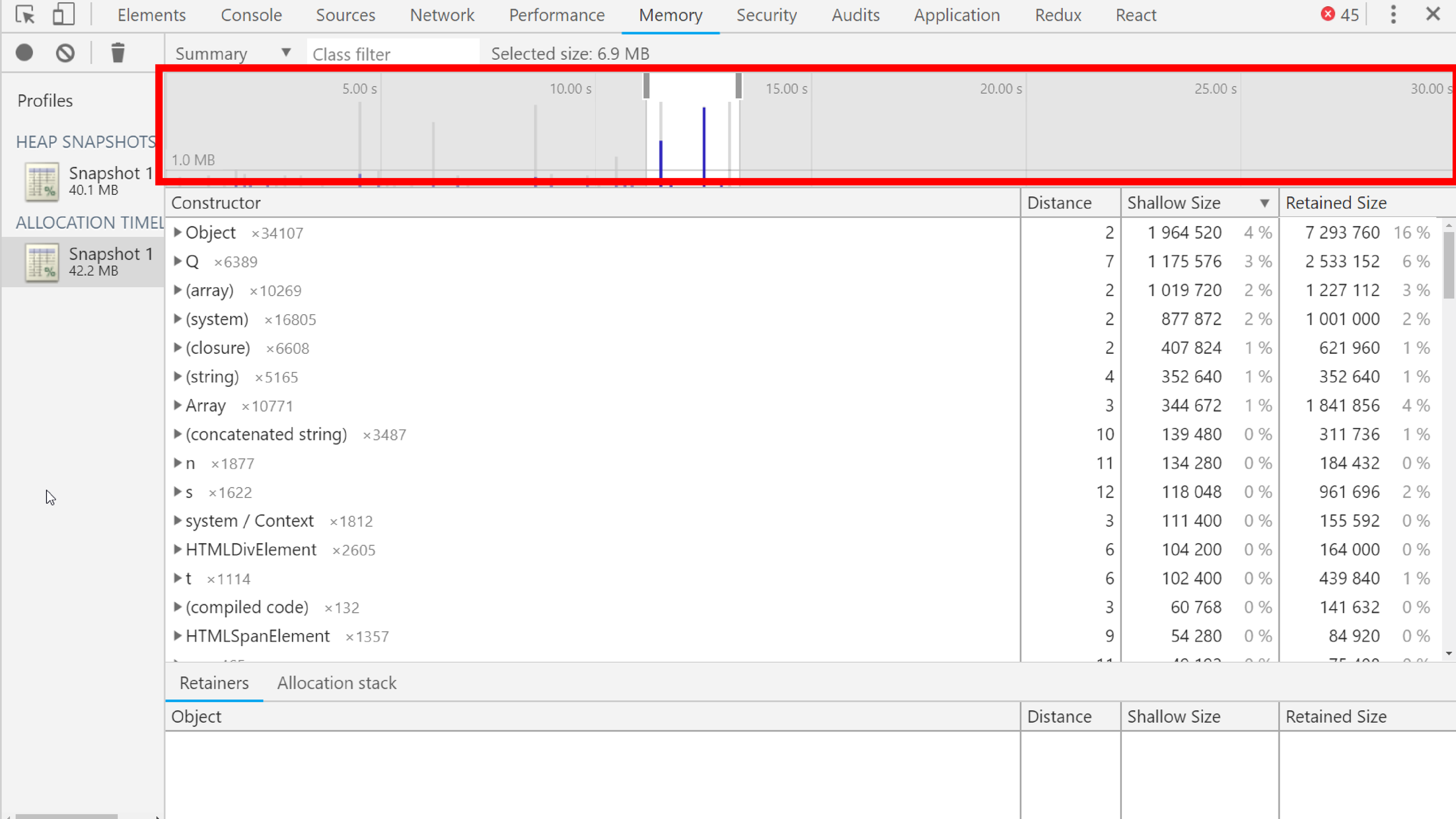

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1S
40.1 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×187722 2 12 268 880 29 % 15 580 664 37 %
(array) ×78290 2 10 269 440 24 % 12 192 976 29 %
Object ×97724 2 5 458 248 13 % 15 041 616 36 %
(string) ×60739 2 4 396 024 10 % 4 396 024 10 %
(compiled code) ×18856 3 2 535 680 6 % 9 852 224 23 %
(closure) ×35141 2 2 161 520 5 % 9 818 400 23 %
Q ×7487 6 1 377 608 3 % 1 792 144 4 %
Array ×40129 2 1 284 176 3 % 9 429 864 22 %
(concatenated string) ×13523 4 540 920 1 % 659 408 2 %
system / Context ×6578 3 494 552 1 % 7 369 656 18 %
r ×2015 6 221 144 1 % 1 677 032 4 %
s ×2657 5 193 216 0 % 1 790 960 4 %
n ×2427 4 157 904 0 % 927 736 2 %
t.setParent ×868 5 97 216 0 % 792 648 2 %
t ×954 4 82 240 0 % 2 122 904 5 %
HTMLDivElement ×1638 4 65 328 0 % 559 512 1 %
e ×372 7 35 888 0 % 136 488 0 %
i ×416 5 30 944 0 % 658 040 2 %
p ×505 4 28 640 0 % 370 072 1 %
Retainers
Object
Distance
Shallow Size
Retained Size

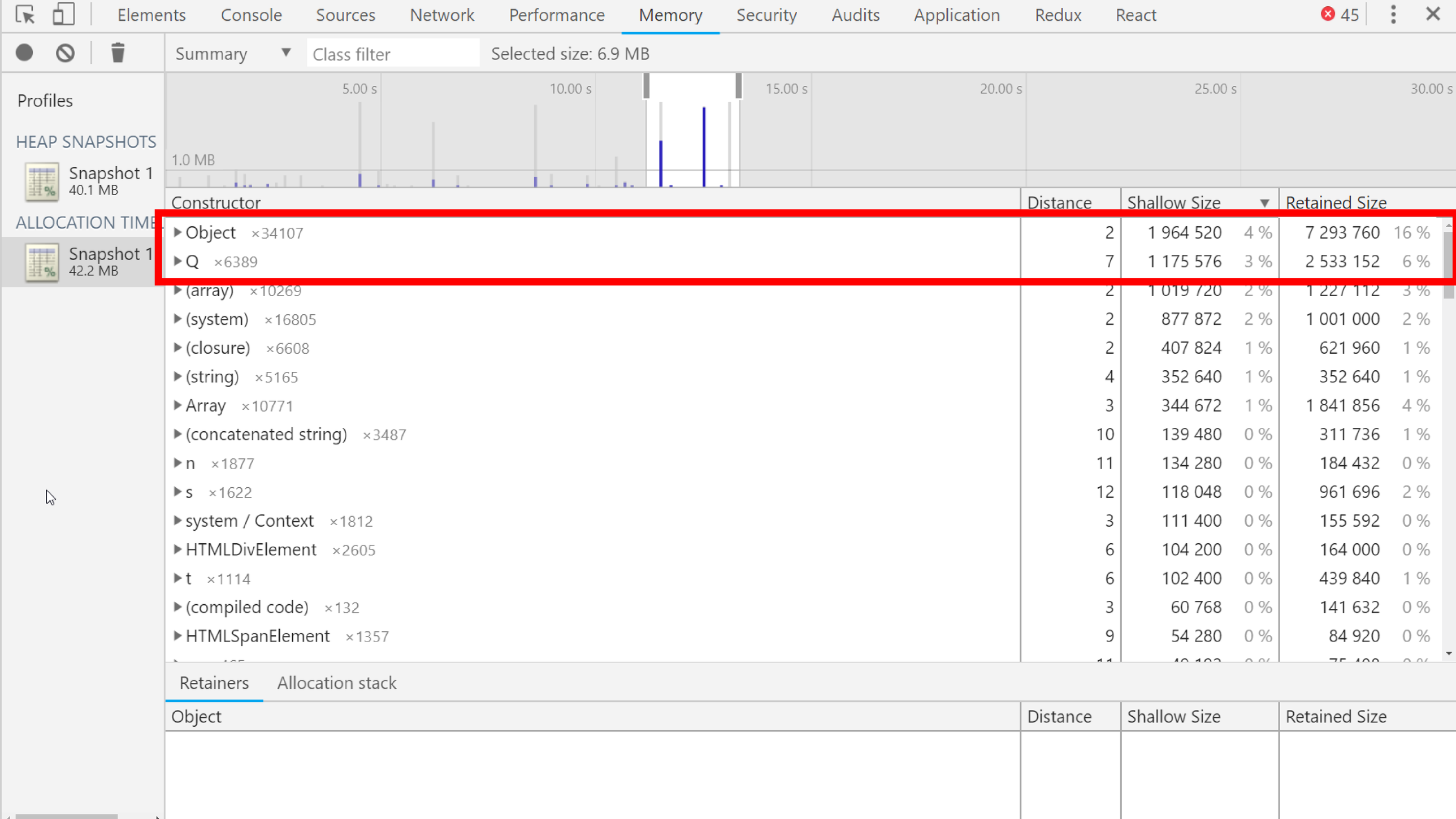

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Summary
Class filter
All objects
Profiles
HEAP SNAPSHOTS
Snapshot 1S
40.1 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×187722 2 12 268 880 29 % 15 580 664 37 %
(array) ×78290 2 10 269 440 24 % 12 192 976 29 %
Object ×97724 2 5 458 248 13 % 15 041 616 36 %
(string) ×60739 2 4 396 024 10 % 4 396 024 10 %
(compiled code) ×18856 3 2 535 680 6 % 9 852 224 23 %
(closure) ×35141 2 2 161 520 5 % 9 818 400 23 %
React
Q ×7487 6 1 377 608 3 % 1 792 144 4 %
Array ×40129 2 1 284 176 3 % 9 429 864 22 %
(concatenated string) ×13523 4 540 920 1 % 659 408 2 %
system / Context ×6578 3 494 552 1 % 7 369 656 18 %
r ×2015 6 221 144 1 % 1 677 032 4 %
s ×2657 5 193 216 0 % 1 790 960 4 %
n ×2427 4 157 904 0 % 927 736 2 %
t.setParent ×868 5 97 216 0 % 792 648 2 %
t ×954 4 82 240 0 % 2 122 904 5 %
HTMLDivElement ×1638 4 65 328 0 % 559 512 1 %
e ×372 7 35 888 0 % 136 488 0 %
i ×416 5 30 944 0 % 658 040 2 %
p ×505 4 28 640 0 % 370 072 1 %
Retainers
Object
Distance
Shallow Size
Retained Size

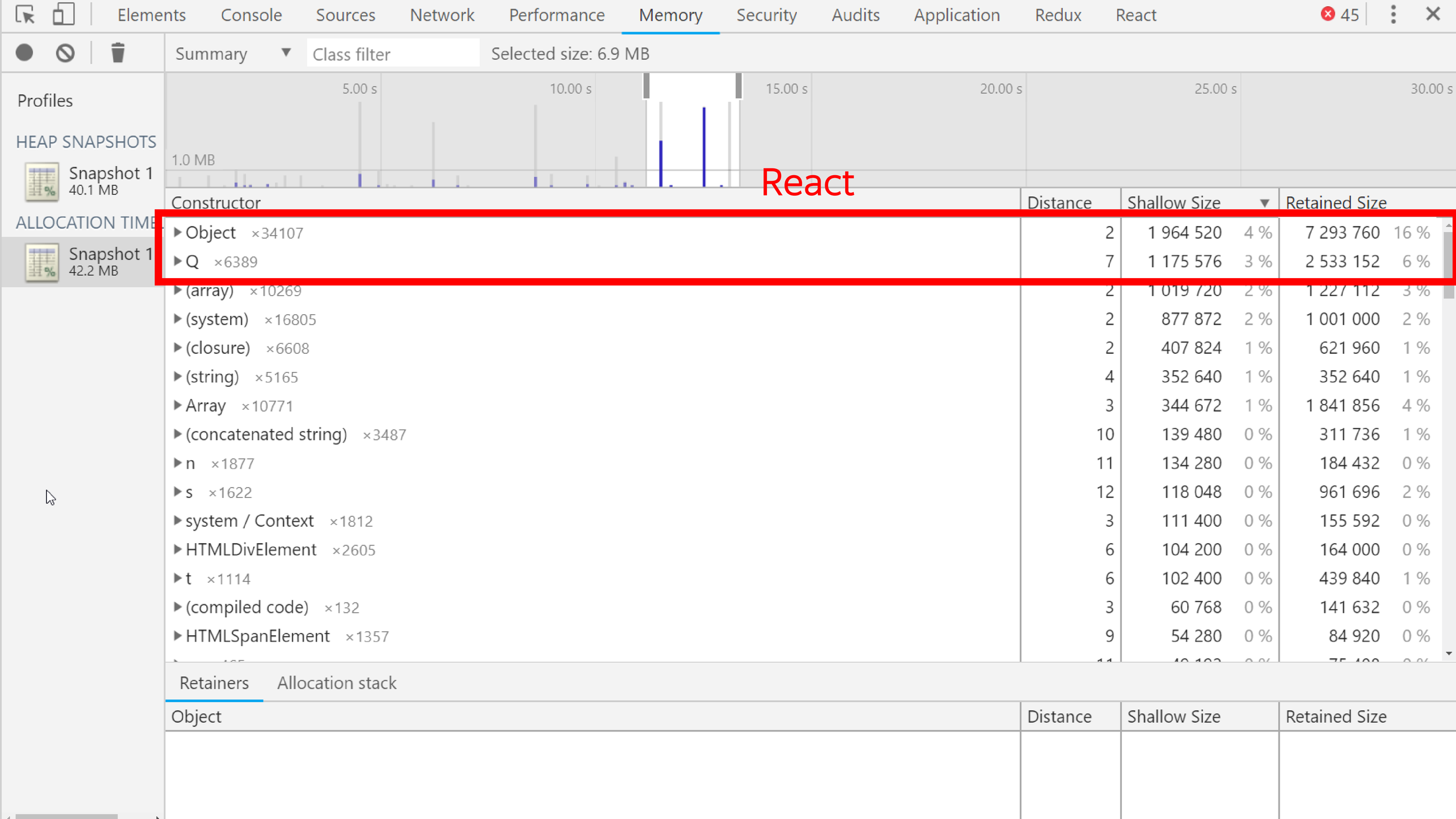

Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
18
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
Select profiling type
Heap snapshot
Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.
Allocation instrumentation on timeline
Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.
Record allocation stacks (extra performance overhead)
Allocation sampling
Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.
Select JavaScript VM instance
40.2 MB / 47.3 MB yandex.ru
Start
Load
169


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
31
Summary
Class filter
Selected size: 42.2 MB
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
ALLOCATION TIMEL
Snapshot 1
42.2 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
1.0 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×193499 2 12 561 696 28 % 15 933 400 36 %
(array) ×83034 2 10 873 960 25 % 12 854 664 29 %
Object ×104680 2 5 846 856 13 % 15 626 512 35 %
(string) ×57607 2 4 199 560 9 % 4 199 560 9 %
(compiled code) ×18963 3 2 656 032 6 % 10 231 168 23 %
(closure) ×37332 2 2 301 632 5 % 10 281 224 23 %
Q ×7824 6 1 439 616 3 % 2 963 672 7 %
Array ×44848 2 1 435 184 3 % 9 925 784 22 %
(concatenated string) ×18055 4 722 200 2 % 1 014 104 2 %
system / Context ×7398 3 543 344 1 % 6 707 480 15 %
s ×3606 5 263 112 1 % 2 308 000 5 %
r ×2133 6 226 248 1 % 1 992 464 5 %
n ×3321 4 223 264 1 % 1 013 600 2 %
t ×1596 4 138 328 0 % 2 395 552 5 %
t.setParent ×1060 5 118 720 0 % 920 312 2 %
Retainers
Allocation stack
Object
Distance
Shallow Size
Retained Size


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
31
Summary
Class filter
Selected size: 42.2 MB
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
ALLOCATION TIMEL
Snapshot 1
42.2 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
1.0 MB
Constructor
Distance
Shallow Size
Retained Size
(system) ×193499 2 12 561 696 28 % 15 933 400 36 %
(array) ×83034 2 10 873 960 25 % 12 854 664 29 %
Object ×104680 2 5 846 856 13 % 15 626 512 35 %
(string) ×57607 2 4 199 560 9 % 4 199 560 9 %
(compiled code) ×18963 3 2 656 032 6 % 10 231 168 23 %
(closure) ×37332 2 2 301 632 5 % 10 281 224 23 %
Q ×7824 6 1 439 616 3 % 2 963 672 7 %
Array ×44848 2 1 435 184 3 % 9 925 784 22 %
(concatenated string) ×18055 4 722 200 2 % 1 014 104 2 %
system / Context ×7398 3 543 344 1 % 6 707 480 15 %
s ×3606 5 263 112 1 % 2 308 000 5 %
r ×2133 6 226 248 1 % 1 992 464 5 %
n ×3321 4 223 264 1 % 1 013 600 2 %
t ×1596 4 138 328 0 % 2 395 552 5 %
t.setParent ×1060 5 118 720 0 % 920 312 2 %
Retainers
Allocation stack
Object
Distance
Shallow Size
Retained Size


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
45
Summary
Class filter
Selected size: 6.9 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
1.0 MB
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
ALLOCATION TIMEL
Snapshot 1
42.2 MB
Constructor
Distance
Shallow Size
Retained Size
Object ×34107 2 1 964 520 4 % 7 293 760 16 %
Q ×6389 7 1 175 576 3 % 2 533 152 6 %
(array) ×10269 2 1 019 720 2 % 1 227 112 3 %
(system) ×16805 2 877 872 2 % 1 001 000 2 %
(closure) ×6608 2 407 824 1 % 621 960 1 %
(string) ×5165 4 352 640 1 % 352 640 1 %
Array ×10771 3 344 672 1 % 1 841 856 4 %
(concatenated string) ×3487 10 139 480 0 % 311 736 1 %
n ×1877 11 134 280 0 % 184 432 0 %
s ×1622 12 118 048 0 % 961 696 2 %
system / Context ×1812 3 111 400 0 % 155 592 0 %
HTMLDivElement ×2605 6 104 200 0 % 164 000 0 %
t ×1114 6 102 400 0 % 439 840 1 %
(compiled code) ×132 3 60 768 0 % 141 632 0 %
HTMLSpanElement ×1357 9 54 280 0 % 84 920 0 %
Retainers
Allocation stack
Object
Distance
Shallow Size
Retained Size


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
45
Summary
Class filter
Selected size: 6.9 MB
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
ALLOCATION TIME
Snapshot 1
42.2 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
1.0 MB
Constructor
Distance
Shallow Size
Retained Size
Object ×34107 2 1 964 520 4 % 7 293 760 16 %
Q ×6389 7 1 175 576 3 % 2 533 152 6 %
(array) ×10269 2 1 019 720 2 % 1 227 112 3 %
(system) ×16805 2 877 872 2 % 1 001 000 2 %
(closure) ×6608 2 407 824 1 % 621 960 1 %
(string) ×5165 4 352 640 1 % 352 640 1 %
Array ×10771 3 344 672 1 % 1 841 856 4 %
(concatenated string) ×3487 10 139 480 0 % 311 736 1 %
n ×1877 11 134 280 0 % 184 432 0 %
s ×1622 12 118 048 0 % 961 696 2 %
system / Context ×1812 3 111 400 0 % 155 592 0 %
HTMLDivElement ×2605 6 104 200 0 % 164 000 0 %
t ×1114 6 102 400 0 % 439 840 1 %
(compiled code) ×132 3 60 768 0 % 141 632 0 %
HTMLSpanElement ×1357 9 54 280 0 % 84 920 0 %
Retainers
Allocation stack
Object
Distance
Shallow Size
Retained Size


Elements
Console
Sources
Network
Performance
Memory
Security
Audits
Application
Redux
React
45
Summary
Class filter
Selected size: 6.9 MB
Profiles
HEAP SNAPSHOTS
Snapshot 1
40.1 MB
ALLOCATION TIME
Snapshot 1
42.2 MB
5.00 s
10.00 s
15.00 s
20.00 s
25.00 s
30.00 s
1.0 MB
React
Constructor
Distance
Shallow Size
Retained Size
Object ×34107 2 1 964 520 4 % 7 293 760 16 %
Q ×6389 7 1 175 576 3 % 2 533 152 6 %
(array) ×10269 2 1 019 720 2 % 1 227 112 3 %
(system) ×16805 2 877 872 2 % 1 001 000 2 %
(closure) ×6608 2 407 824 1 % 621 960 1 %
(string) ×5165 4 352 640 1 % 352 640 1 %
Array ×10771 3 344 672 1 % 1 841 856 4 %
(concatenated string) ×3487 10 139 480 0 % 311 736 1 %
n ×1877 11 134 280 0 % 184 432 0 %
s ×1622 12 118 048 0 % 961 696 2 %
system / Context ×1812 3 111 400 0 % 155 592 0 %
HTMLDivElement ×2605 6 104 200 0 % 164 000 0 %
t ×1114 6 102 400 0 % 439 840 1 %
(compiled code) ×132 3 60 768 0 % 141 632 0 %
HTMLSpanElement ×1357 9 54 280 0 % 84 920 0 %
Retainers
Allocation stack
Object
Distance
Shallow Size
Retained Size

Если этого вдруг не хватило

- Chromium: `about:tracing`
- Firefox: `about:memory, about:performance`
- Node: `--trace-gc, --expose-gc, require('trace_events')`

Итого

Итого

- Сборщик мусора умный
- Никто вам не мешает выстрелить себе в ногу
- Не бойтесь создавать мусор
- Следите за перформансом
- Если у вас не SPA, то вы можете забить и ничего не делать
- Большая часть ошибок и сомнительных мест растет из незнания того, как работает ваш инструмент
- Вы теперь знаете

Команда JavaScript API Яндекс.Карт ищет разработчиков

- Возможность поучаствовать в работе над одним из крупнейших API рунета
- Нестандартные задачи для фронтенда
- Отсутствие верстки

- yandex.ru/jobs/vacancies/dev/intdev_api_maps
- yandex.ru/jobs/vacancies/dev/dev_graphics_api_maps

Слайды: flapenguin.me/talks/gc

Вопросы?

Роенко Андрей

flapenguin@yandex.ru

@flapenguin

@fla_penguin

flapenguin

flapenguin.me