Прокси из ES6: сахар для работы с DOM и не только

Алексей Швайка

WSD 01.11.2015

Front End инженер

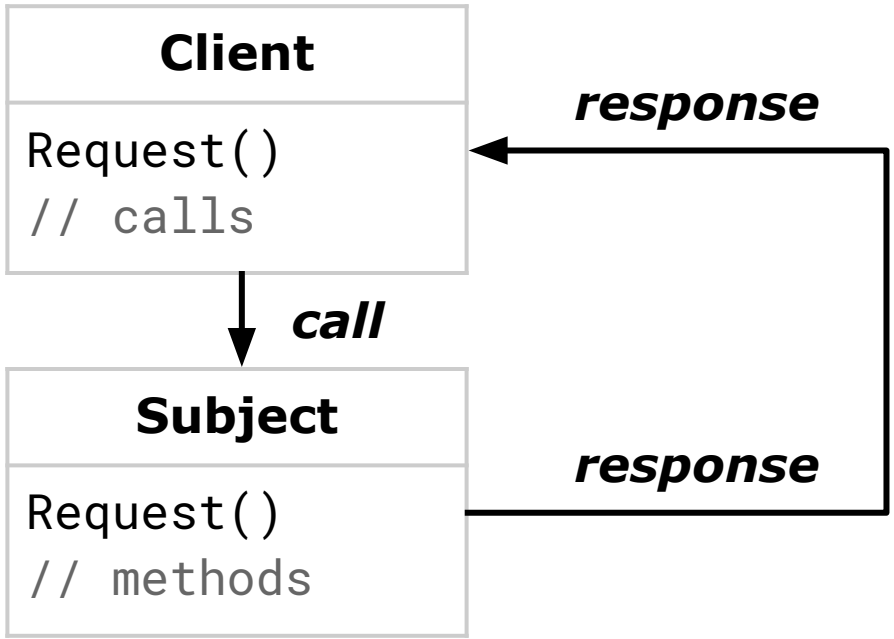
gitter.im/dev-ua

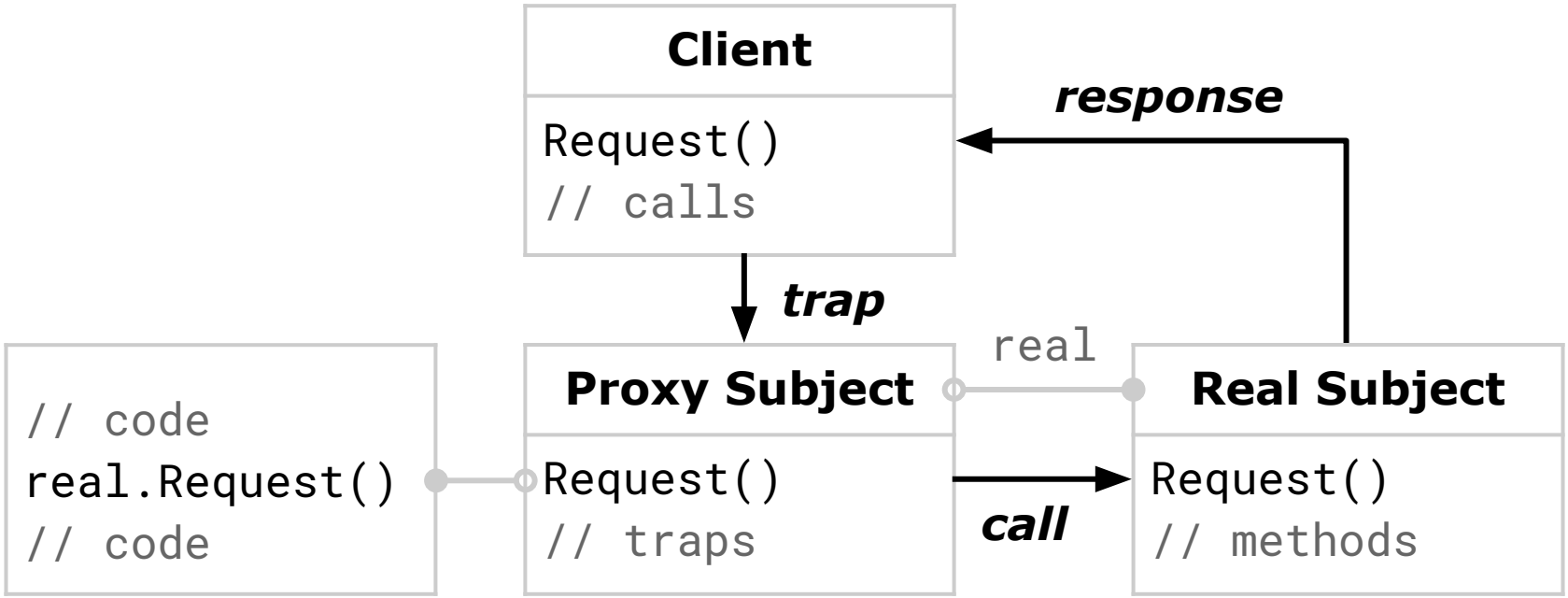
@shvaikalesh

Что такое ***прокси***?

```
$.proxy(callable, context /*, ...arguments */)
  $.proxy(context, name /*, ...arguments */)
```

```
$.proxy(callable, context /*, ...arguments */)
  $.proxy(context, name /*, ...arguments */)
```


Client
Request()
// calls
response
call
Subject
response
Request()
// methods


Client
Request()
// calls
response
trap
real
// code
real.Request()
// code
Proxy Subject
Request()
// traps
call
Real Subject
Request()
// methods

Зачем они нужны?

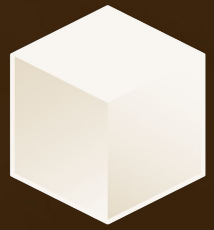
- логирование
- кэширование
- сборка мусора
- экономия памяти
- ограничение доступа
- синхронизация доступа

Другие шаблоны

- ***адаптер*** предоставляет *подходящий* интерфейс
- ***декоратор*** предоставляет *расширенный* интерфейс
- ***итератор*** использует ***прокси*** для деструкторизации

Прокси в JavaScript

git.io/vW6R1



Сахар №1:
работа с атрибутами

function attributesProxy()

```
return new Proxy(Object.create(null), {
    has: (map, name) => this.hasAttribute(name),
    get: (map, name) => this.getAttribute(name),
    set: (map, name, value) => this.setAttribute(name, value),
    deleteProperty: (map, name) =>
        this.hasAttribute(name) &&
        this.removeAttribute(name) ||
        true
})
```

```
Object.defineProperty(
    Element.prototype,
    "attributes", {
        get: attributesProxy
        set: function(value) {
            Object.assign(this.attributes, value)
        }
})
```

```
let matrix = document.query("feColorMatrix")
    matrix.type // => SVGAnimatedEnumeration { }
    matrix.values // => SVGAnimatedNumberList { }

let attributes = matrix.attributes
    attributes.type // => "saturate"
    attributes.values // => "0.3"

attributes.type = "hueRotate"
attributes.values = 50

<filter>
    <feColorMatrix type="hueRotate" values="50"/>
</filter>
```

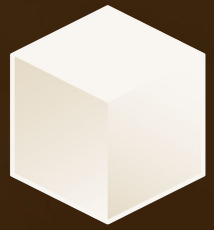
```
<input autocomplete="off" spellcheck="false">

///

let input = document.query("input").attributes

"autocapitalize" in input // => false
input.autocapitalize = "off"
input.autocorrect = "off"

"autocorrect" in input // => true
input.spellcheck // => "false"
delete input.autocomplete // => true
```



Сахар №2:
работа с селектами

```
function optionsProxy()

let _search = value => {
    return this
        .queryAll("option")
        .find($option => $option.value == value)
}

let _create = value => {
    let $option = document.createElement("option")
        $option.value = value

    return this.appendChild($option)
}
```

function optionsProxy()

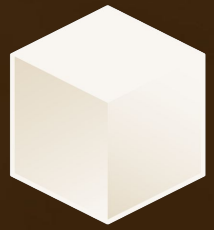
```
return new Proxy(Object.create(null), {
    has: (list, name) => _search(name) == null,
    get: (list, name) => _search(name),
    set: (list, name, value) => {
        let $option = _search(name) || _create(name)
            $option.textContent = value
    },
    deleteProperty: (list, name) => {
        let $option = _search(name)
        return $option && $option.remove() || true
    }
})
```

```
Object.defineProperty(
    HTMLSelectElement.prototype,
    "options", {
        get: optionsProxy,
        set: function(value) {
           Object.assign(this.options, value)
        }
})
```

```
let select = document.createElement("select")
    select.options = {
        eur: "Евро",
        rub: "Рубли",
        uah: "Гривны"
    }

select.options.eur // => "Евро"
select.options.usd = "Доллары"

delete select.options.usd // => true
"rub" in select.options // => true
```



Сахар №3:
работа с куками

function cookieProxy()

```
import { get, set } from "document-cookie.js"

return new Proxy(Object.create(null), {
    has: (map, name) => get(name) != "",
    get: (map, name) => get(name),
    set: (map, name, value) => set(name, value),
    deleteProperty: (map, name) => {
        set(name, "", { expires: new Date(0) })
        return true
    }
})
```

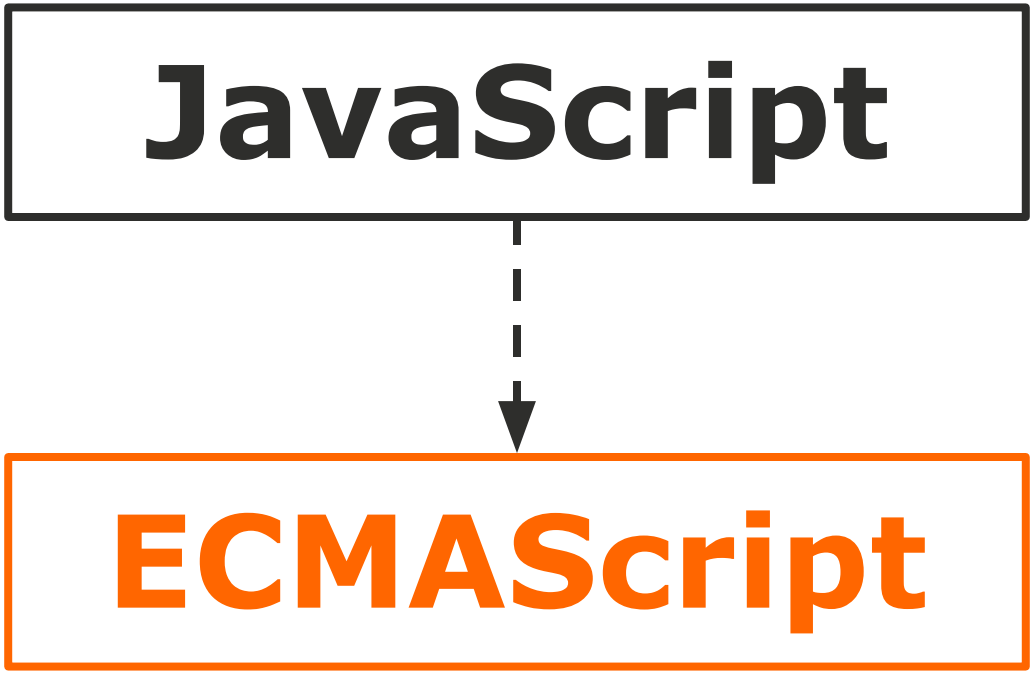
```
Object.defineProperty(
    Document.prototype,
    "cookie", {
        get: cookieProxy,
        set: function(value) {
            Object.assign(this.cookie, value)
        }
})
```

```
let cookie = document.cookie

if (!("session" in cookie)) {
    cookie.session_id = getSessionId()
    cookie.last_visit = Date.now()
}

if ("show_ad" in cookie) {
    showAd(cookie.banner_id)
    delete cookie.show_ad
}
```

Как ***прокси*** работают?


JavaScript
ECMAScript

```
// Установить прототип объекта
let user = Object.setPrototypeOf(
    { },
    { id: 4815, name: "Алексей" }
)

// Проверить наличие свойства
"name" in user

// Получить значение свойства
user.id


// Удалить собственное свойство
delete user.name
```

```
// LexicalDeclaration
let user = Object.setPrototypeOf(
    { },
    { id: 4815, name: "Алексей" }
)

// RelationalExpression
"name" in user

// MemberExpression
user.id

// UnaryExpression
delete user.name
```

LetOrConst	let
BindingIdentifier	user
CallExpression	MemberExpression
	Arguments

StringLiteral	Keyword	IdentifierName
"name"	in	user

PrimaryExpression	IdentifierName
user	id

Keyword
delete

MemberExpression	
PrimaryExpression	user
IdentifierName	name

```
%InitializeReferencedBinding%("user", { })
user.[[SetPrototypeOf]]({ id: 4815, name: "Алексей" })
// let user = Object.setPrototypeOf({ },
//     { id: 4815, name: "Алексей" }
// )

user.[[HasProperty]]("name")
// "name" in user

user.[[Get]]("id", user)
// user.id


user.[[Delete]]("name")
// delete user.name
```

astexplorer.net/#/WXr2zRWelu

Обычные внутренние **методы объектов**

function [[SetPrototypeOf]](object)

```
if (this.[[Prototype]] == object) return true
if (this.[[Extensible]] == false) return false

for (let prototype = object ;;) {
    if (prototype == object) return false
    if (prototype == null) break
    // если prototype не обычный объект, выйти из for
    prototype = Object.getPrototypeOf(prototype)
}

this.[[Prototype]] = object
return true
```

```
function [[HasProperty]](name)

let descriptor = Object.getOwnPropertyDescriptor(this, name)
if (descriptor != null) return true

let parent = Object.getPrototypeOf(this)
if (parent != null) return parent.[[HasProperty]](name)

return false
```

function [[Get]](name, receiver)

```
let descriptor = Object.getOwnPropertyDescriptor(this, name)
if (descriptor == null) {
    let parent = Object.getPrototypeOf(this)
    if (parent == null) return

    return parent.[[Get]](name, receiver)
}

if ("value" in descriptor) return descriptor.value
if ("get" in descriptor) return descriptor.get.call(receiver)
```

```
function [[Delete]](name)

let descriptor = Object.getOwnPropertyDescriptor(this, name)
if (descriptor == null) return true

if (descriptor.configurable) {
    // удалить собственное свойство name
    return true
}

return false
```

Прокси в **ECMAScript** 6

```
function Proxy(target, handler)

if (new.target != Proxy) throw new TypeError()
if (%HasPrimitive%(arguments)) throw new TypeError()

let proxy = Object.create(null, %ProxyInternalMethods%)
    proxy.[[ProxyTarget]] = target
    proxy.[[ProxyHandler]] = handler

"function" == typeof target || delete proxy.[[Call]]
"[[Construct]]" in target || delete proxy.[[Construct]]

return proxy
```

Внутренние **методы** ***прокси-****объектов*

function [[SetPrototypeOf]](object)

```
let handler = this.[[ProxyHandler]]
if (handler == null) throw new TypeError()
let target = this.[[ProxyTarget]]
let trap = handler.setPrototypeOf
if (trap == null) return target.[[SetPrototypeOf]](object)

let result = !! trap.call(handler, target, object)
if (Object.isExtensible(target)) return result
if (Object.getPrototypeOf(target) != object)
    if (result) throw new TypeError()

return result
```

```
function [[HasProperty]](name)

let trap = handler.has
if (trap == null) return target.[[HasProperty]](name)

let result = trap.call(handler, target, name)
if (result) return true

let descriptor = Object.getOwnPropertyDescriptor(target, name)
if (descriptor)
    if (!Object.isExtensible(target)) throw new TypeError()
    else if (!descriptor.configurable) throw new TypeError()

return false
```

```
function [[Get]](name, receiver)

let trap = handler.get
if (trap == null) return target.[[Get]](name, receiver)

let result = trap.call(handler, target, name, receiver)
let descriptor = Object.getOwnPropertyDescriptor(target, name)
if (descriptor && !descriptor.configurable)
    if ("value" in descriptor && !descriptor.writable)
        if (result != descriptor.value) throw new TypeError()
    else if ("set" in descriptor && descriptor.get == null)
        if (result != null) throw new TypeError()

return result
```

```
function [[Delete]](name)

let trap = handler.deleteProperty
if (trap == null) return target.[[Delete]](name)

let failed = !trap.call(handler, target, name)
if (failed) return false

let descriptor = Object.getOwnPropertyDescriptor(target, name)
if (descriptor && !descriptor.configurable)
    throw new TypeError()

return true
```

Поддержка браузерами **сегодня** (01.11.2015)

Edge	Firefox	Chrome	Safari	IE 11
21/21	**19/21**	**0/21**	**0/21**	**0/21**

шим не возможен	эти полифилятся
[[GetPrototypeOf]]	[[SetPrototypeOf]]
[[GetOwnProperty]]	[[IsExtensible]]
[[DefineOwnProperty]]	[[PreventExtensions]]
[[HasProperty]]	[[Call]]
[[Get]]	[[Construct]]
[[Enumerate]]	[[Set]]
[[OwnPropertyKeys]]	[[Delete]]

шим не возможен	эти полифилятся
[[GetPrototypeOf]]	**[[SetPrototypeOf]]**
[[GetOwnProperty]]	**[[IsExtensible]]**
[[DefineOwnProperty]]	**[[PreventExtensions]]**
[[HasProperty]]	[[Call]]
[[Get]]	[[Construct]]
[[Enumerate]]	[[Set]]
[[OwnPropertyKeys]]	[[Delete]]

шим не возможен	эти полифилятся
[[GetPrototypeOf]]	[[SetPrototypeOf]]
[[GetOwnProperty]]	[[IsExtensible]]
[[DefineOwnProperty]]	[[PreventExtensions]]
[[HasProperty]]	**[[Call]]**
[[Get]]	**[[Construct]]**
[[Enumerate]]	[[Set]]
[[OwnPropertyKeys]]	[[Delete]]

шим не возможен	эти полифилятся
[[GetPrototypeOf]]	[[SetPrototypeOf]]
[[GetOwnProperty]]	[[IsExtensible]]
[[DefineOwnProperty]]	[[PreventExtensions]]
[[HasProperty]]	[[Call]]
[[Get]]	[[Construct]]
[[Enumerate]]	**[[Set]]**
[[OwnPropertyKeys]]	**[[Delete]]**

StorageEvent

StorageEvent

- **localStorage** медленный
- *строка => строка*
- не работает с числовыми ключами в IE < 10
- срабатывает иначе в IE < 11
- **асинхронный**

Element.prototype.dataset

```
<figure id="map"
    data-zoom="16"
    data-marker-icon="/images/icon/marker"
    data-marker-href="about:blank">
</figure>

let map = document.query("#map")
    map.dataset.zoom // => "16"
    map.dataset.zoom = 13

map.dataset.markerIcon += ".png"
map.dataset.markerHref = "https://goo.gl/maps/WBy33"

delete map.dataset.markerHref // => true
"markerHref" in map // => false
```

Element.prototype.dataset

- медленный
- *строка => строка*
- не поддерживается IE < 11
- **синхронный**

MutationObserver

MutationObserver

- **асинхронный**
- это единственное что имеет значение

MutationEvents

MutationEvents

- **синхронные**
- устаревшие
- не полностью поддерживаются
- медленные
- **подходят**

DOMAttrModified

DOMCharacterDataModified

DOMNodeInserted

DOMNodeInsertedIntoDocument

DOMNodeRemoved

DOMNodeRemovedFromDocument

DOMSubtreeModified

DOMAttrModified

DOMCharacterDataModified

DOMNodeInserted

~~DOMNodeInsertedIntoDocument~~

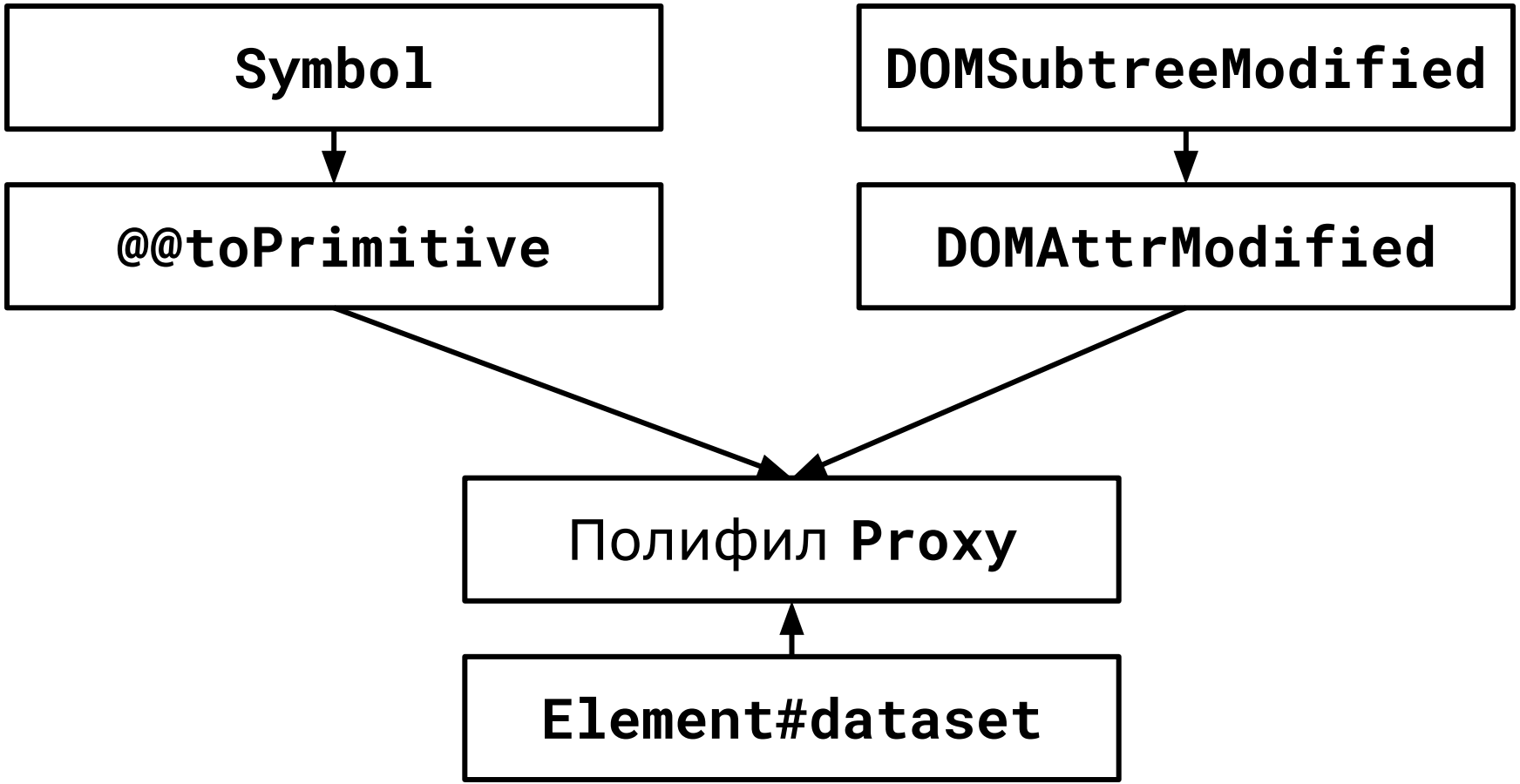
DOMNodeRemoved

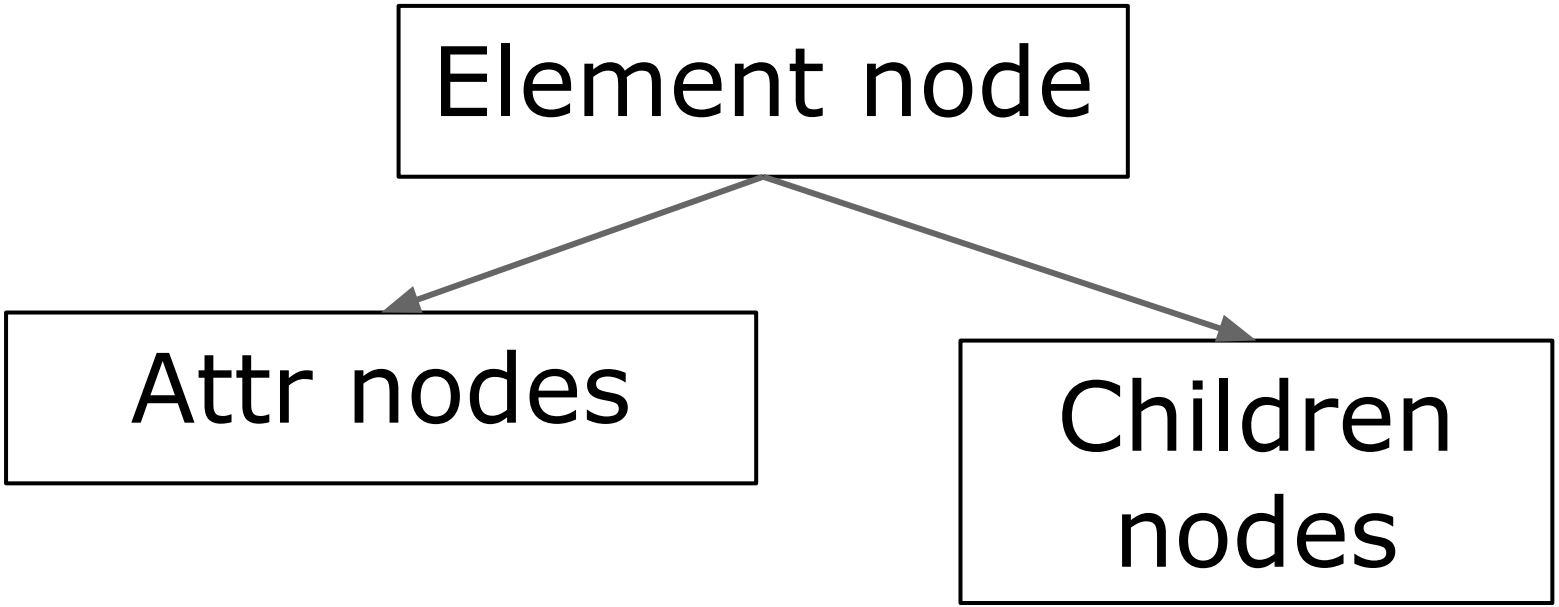
~~DOMNodeRemovedFromDocument~~

DOMSubtreeModified

DOMAttrModified
DOMSubtreeModified

В Chrome 45+, **`Attr`** больше не наследует интерфейс **`Node`**


Symbol
DOMSubtreeModified
@@toPrimitive
DOMAttrModified
Полифил Proxy
Element#dataset


Element node
Attr nodes
Children nodes

Edge	Firefox	Chrome	Safari	IE 11
21/21	**19/21**	**9/21**	**9/21**	**9/21**

github.com/shvaikalesh

Ссылки и материалы

- Спецификация ECMA-262
- Спецификация DOM Level 3 Events
- Спецификация DOM Level 4 (#dom-dataset)
- Исходный код Chromium
- Gang of Four: Design Patterns
- Brendan Eich: Proxies are Awesome!
- Ingvar Stepanyan: es6.concurrency()

Спасибо за внимание!