Agoric

Dear Google, Let’s Harden JavaScript

Kris Kowal and Mark S. Miller

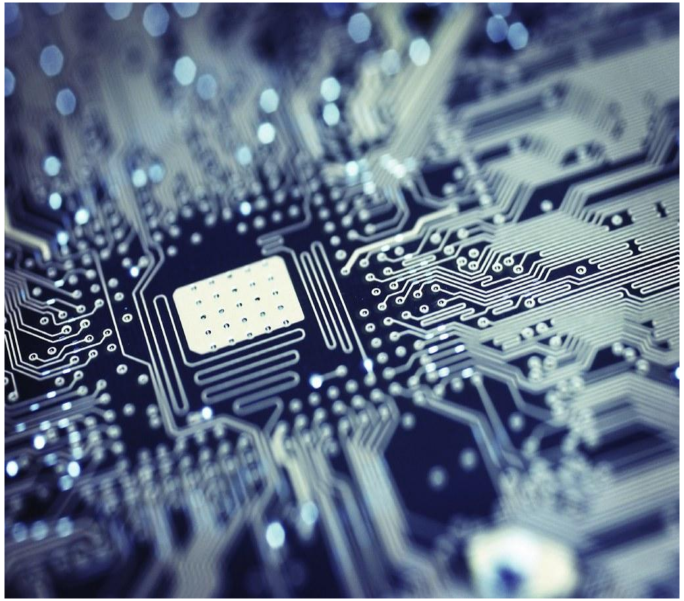
2024-02-12

How JavaScript can evolve to better support Hardened JavaScript

Agoric

Dear Google,

- What is **Hardened JavaScript**?
- Why do we need it?
- How do we emulate it?
- Limits to faithful emulation
- How can JavaScript evolve to better support Hardened JavaScript?



Hardened JavaScript

It’s just JavaScript, plus:

- Lockdown
- Harden
- Compartment

Does:

- Integrity
- Multi-tenant
- API defensibility

Doesn’t (alone):

- Availability (Denial of Service)
- Side-channel (Spectre/Meltdown) resistance

Agoric

Example

```
lockdown();

const compartment = new Compartment();
harden(compartment.globalThis);

const SafeFunction = compartment.globalThis.Function;

const search = harden(query => {
  const match = new SafeFunction('item', query);
  const matches = [];
  for (const item of database.items()) {
    if (match(harden(item))) {
      matches.push(item);
    };
  }
  return harden(matches);
});
```

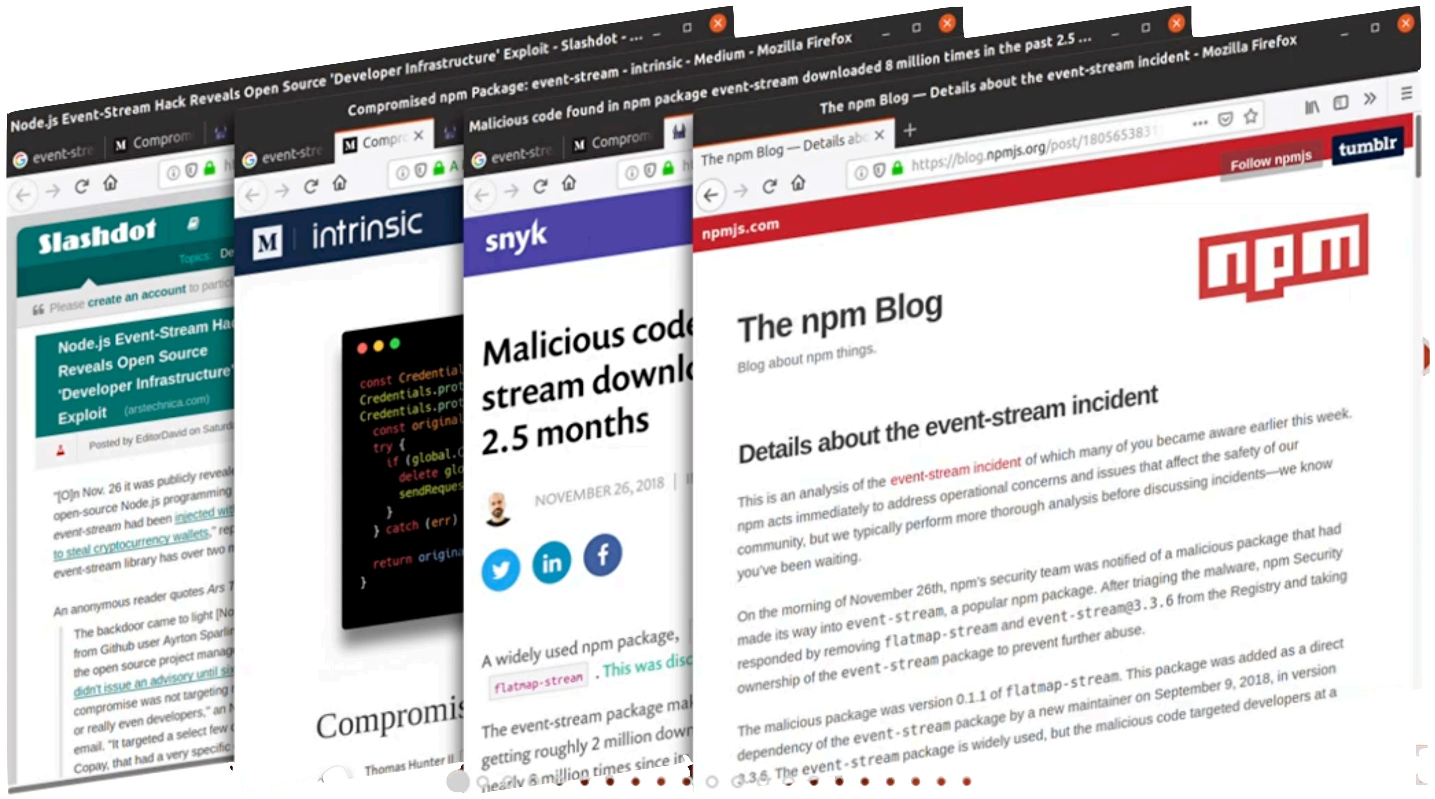
Why do we need Hardened JavaScript

That *any* JavaScript program consists of code that represents the interests of any single party is a rapidly deteriorating fiction.

Agoric

Why do we need HardenedJS

- Supply Chain Defense
- Decentralization / Mashups
- Smart Contracts
- Host Emulation



Node.js Event-Stream Hack Reveals Open Source 'Developer Infrastructure' Exploit - Slashdot - ...

Slashdot

Please create an account to partic

Node.js Event-Stream Hac
Reveals Open Source
'Developer Infrastructure'
Exploit (arstechnica.com)

Posted by EditorDavid on Saturd

"[O]n Nov. 26 it was publicly reveal
open-source Node.js programming
event-stream had been injected wit
to steal cryptocurrency wallets," rep
event-stream library has over two m

An anonymous reader quotes Ars T

The backdoor came to light [No
from Github user Ayrton Sparlin
the open source project manag
didn't issue an advisory until six
compromise was not targeting t
or really even developers," an N
email. "It targeted a select few
Copay, that had a very specific

Compromised npm Package: event-stream - intrinsic - Medium - Mozilla Firefox

intrinsic

```
const Credential
Credentials.prot
Credentials.prot
  const original
  try {
    if (global.c
      delete glo
      sendReques
    }
  } catch (err)

  return origina
}
```

Compromis

Thomas Hunter II

Malicious code found in npm package event-stream downloaded 8 million times in the past 2.5 ...

snyk

Malicious code
stream downl
2.5 months

NOVEMBER 26, 2018

A widely used npm package, flatmap-stream. This was disc

The event-stream package mal
getting roughly 2 million dow
nearly 8 million times since in

The npm Blog — Details about the event-stream incident - Mozilla Firefox

https://blog.npmjs.org/post/180565383

Follow npmjs tumblr

npmjs.com

npm

The npm Blog

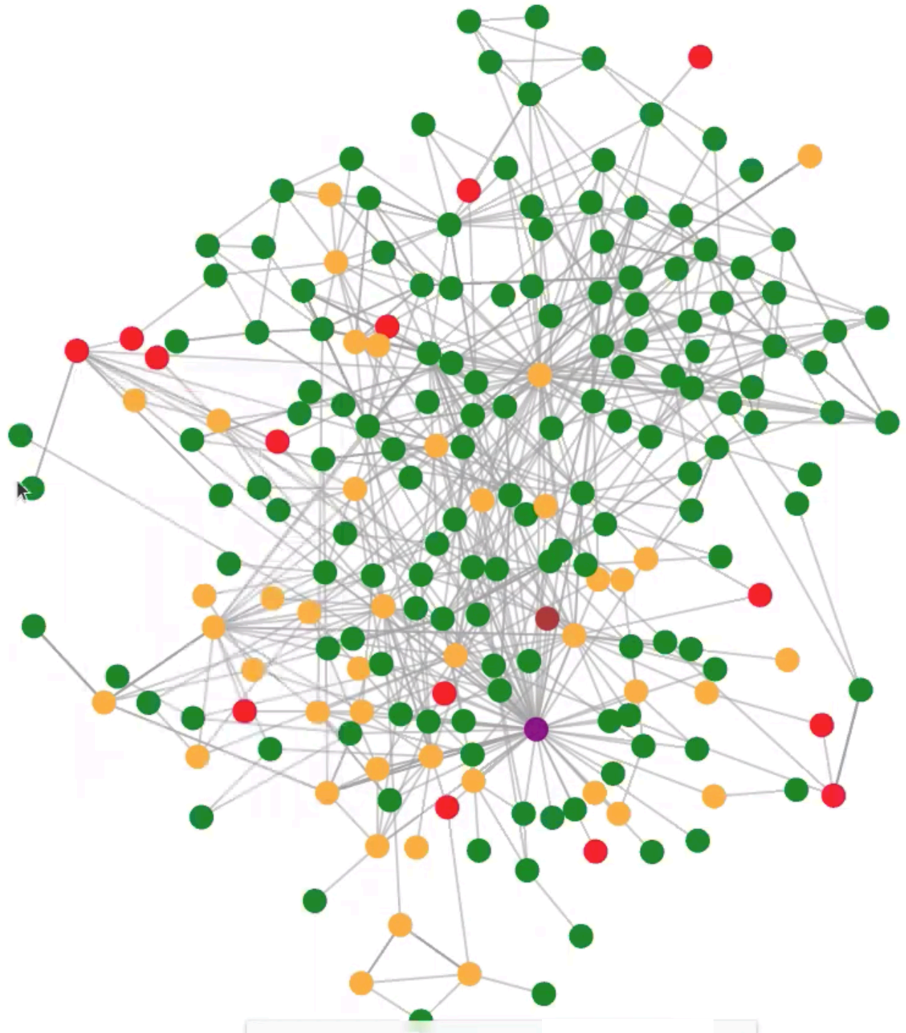
Blog about npm things.

Details about the event-stream incident

This is an analysis of the event-stream incident of which many of you became aware earlier this week. npm acts immediately to address operational concerns and issues that affect the safety of our community, but we typically perform more thorough analysis before discussing incidents—we know you've been waiting.

On the morning of November 26th, npm's security team was notified of a malicious package that had made its way into `event-stream`, a popular npm package. After triaging the malware, npm Security responded by removing `flatmap-stream` and `event-stream@3.3.6` from the Registry and taking ownership of the `event-stream` package to prevent further abuse.

The malicious package was version 0.1.1 of `flatmap-stream`. This package was added as a direct dependency of the `event-stream` package by a new maintainer on September 9, 2018, in version 3.3.6. The `event-stream` package is widely used, but the malicious code targeted developers at a



Emulating Hardened JavaScript Today

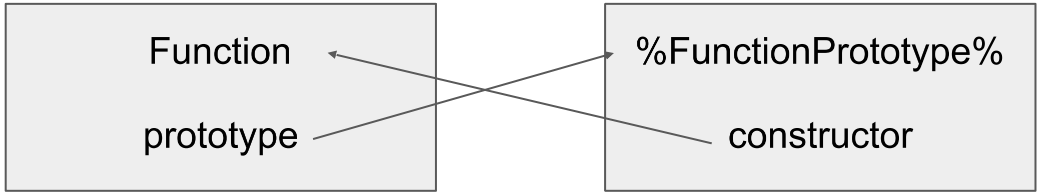
- `with`
- `eval`
- `Proxy`
- Sloppy mode

Agoric

The Device

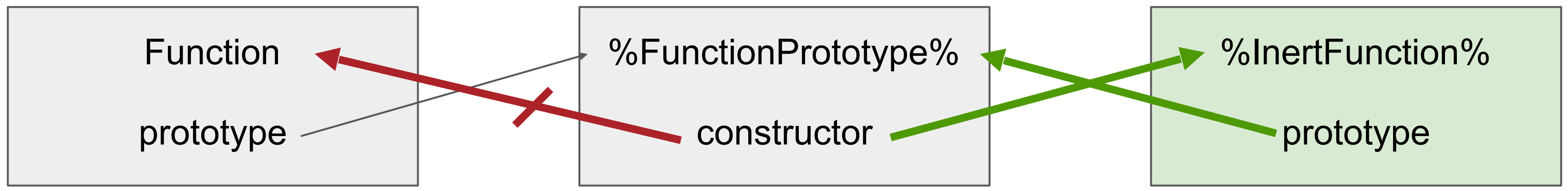
```
const makeEval = Function(`
  with (this.scopeTerminator) {
    with (this.globalObject) {
      with (this.moduleLexicals) {
        with (this.evalScope) {
          ${globalObjectOptimizer} // const {a,b,c} = this.globalObject;
          ${moduleLexicalOptimizer} // const {d,e,f} = this.moduleLexicals;
          return function(/* source */) {
            'use strict';
            return eval(arguments[0]); // censored: eval, import, HTML comments
          };
        }
      }
    }
  }
`);
```

Shared Intrinsics

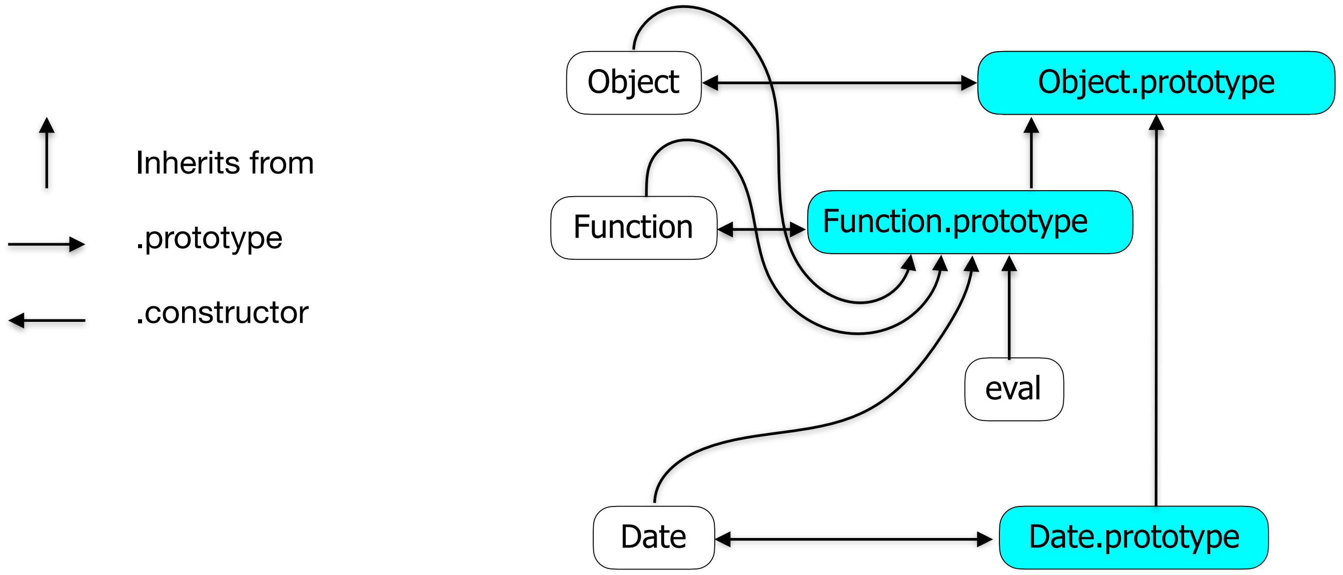


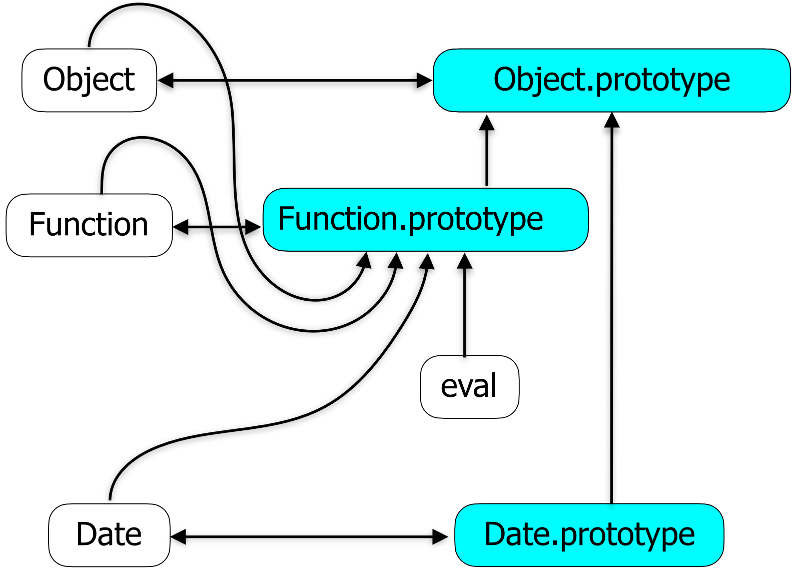
```
new Function("return 42") instanceof Function
    new Function.constructor("return 42")
```

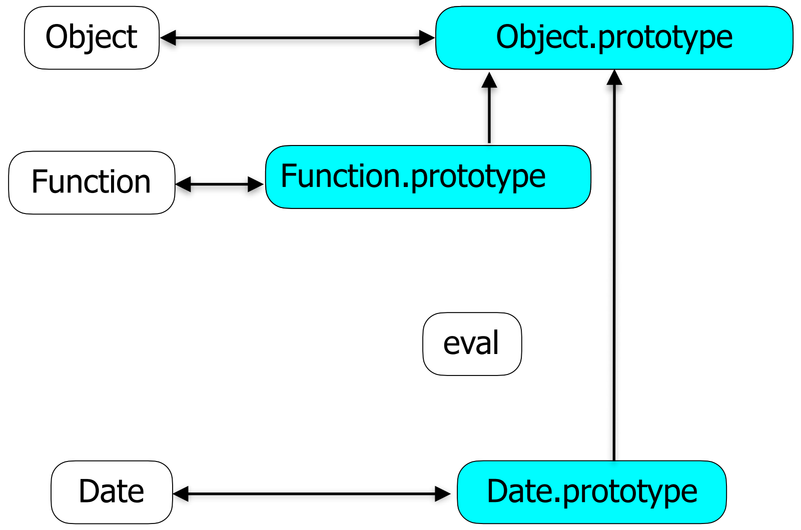
Lockdown Shared Intrinsics

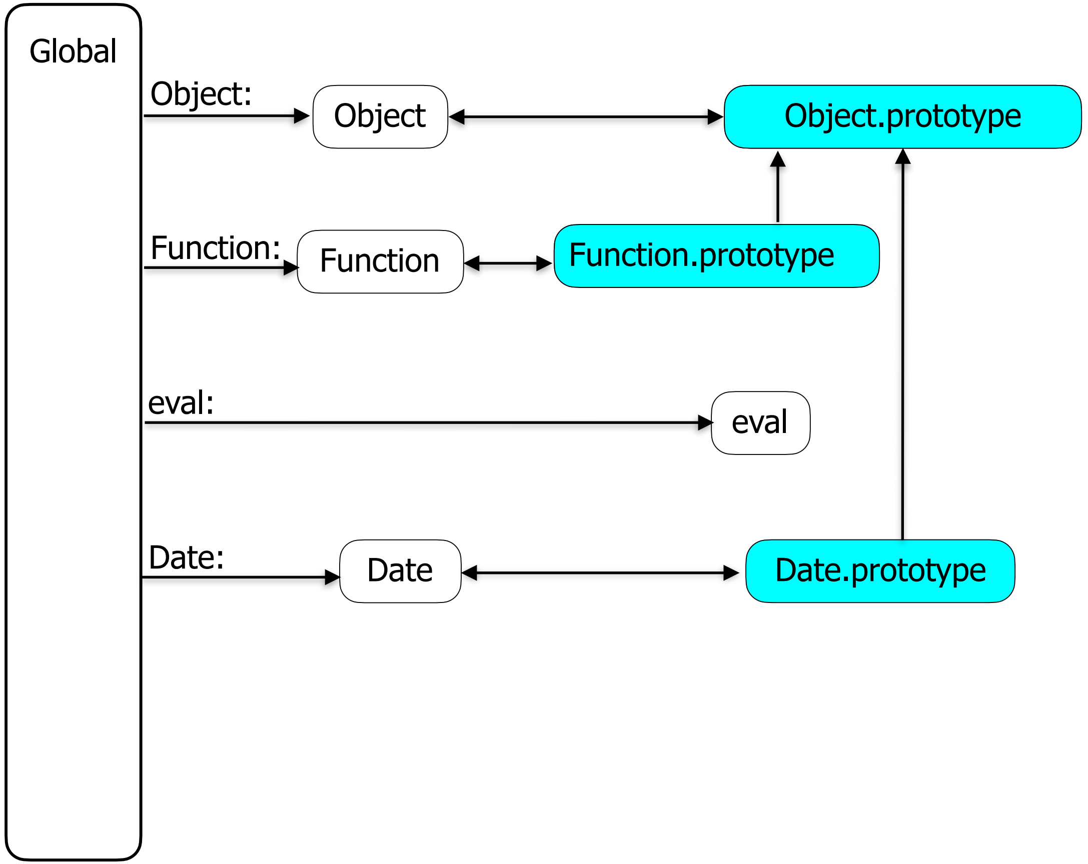


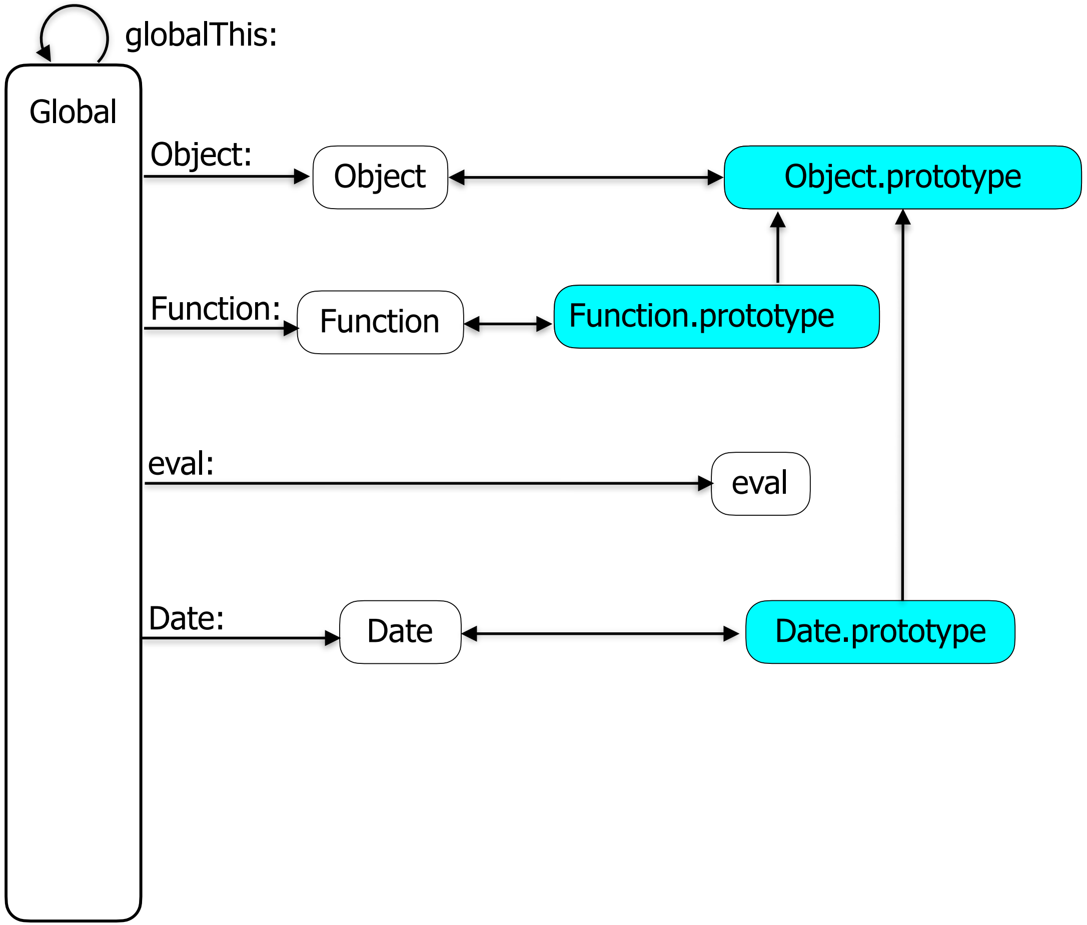
```
new Function("return 42") instanceof Function
    new Function.constructor("return 42")
```

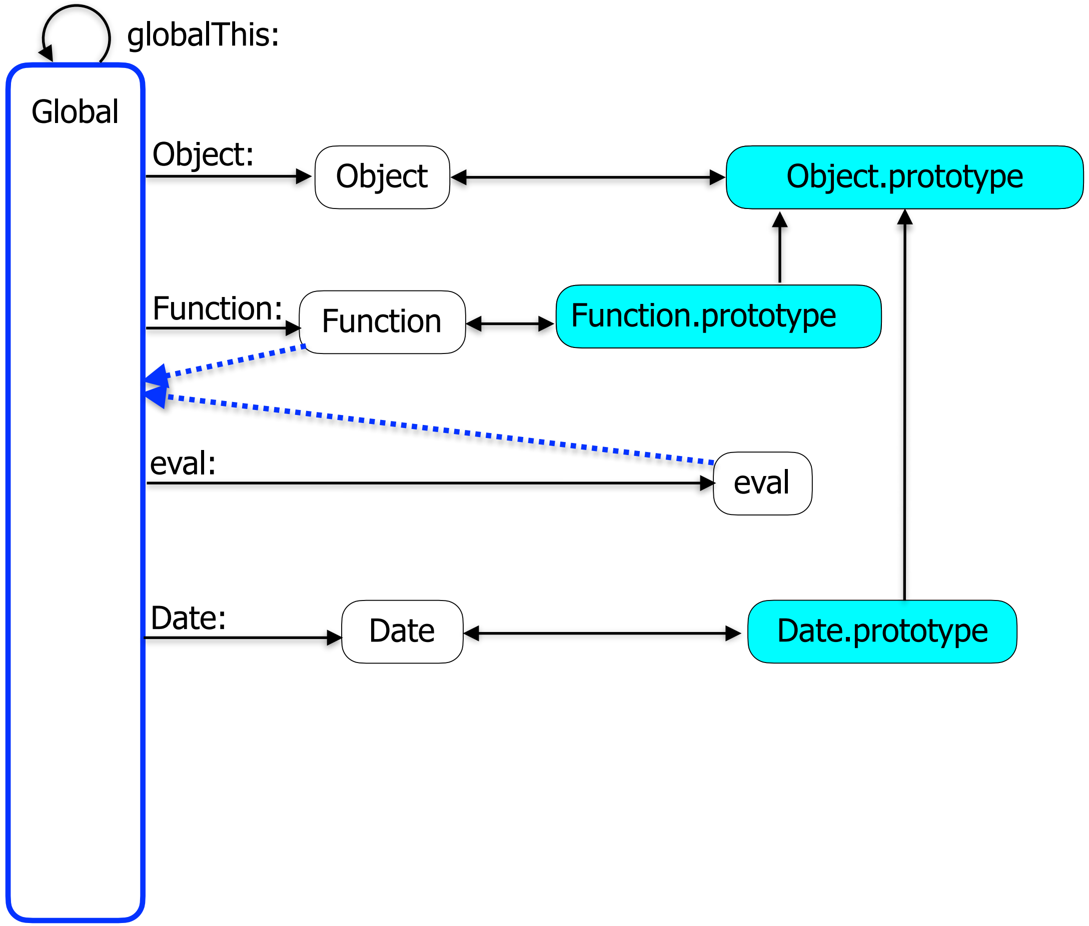

Inherits from
.prototype
.constructor
Object
Object.prototype
Function
Function.prototype
eval
Date
Date.prototype

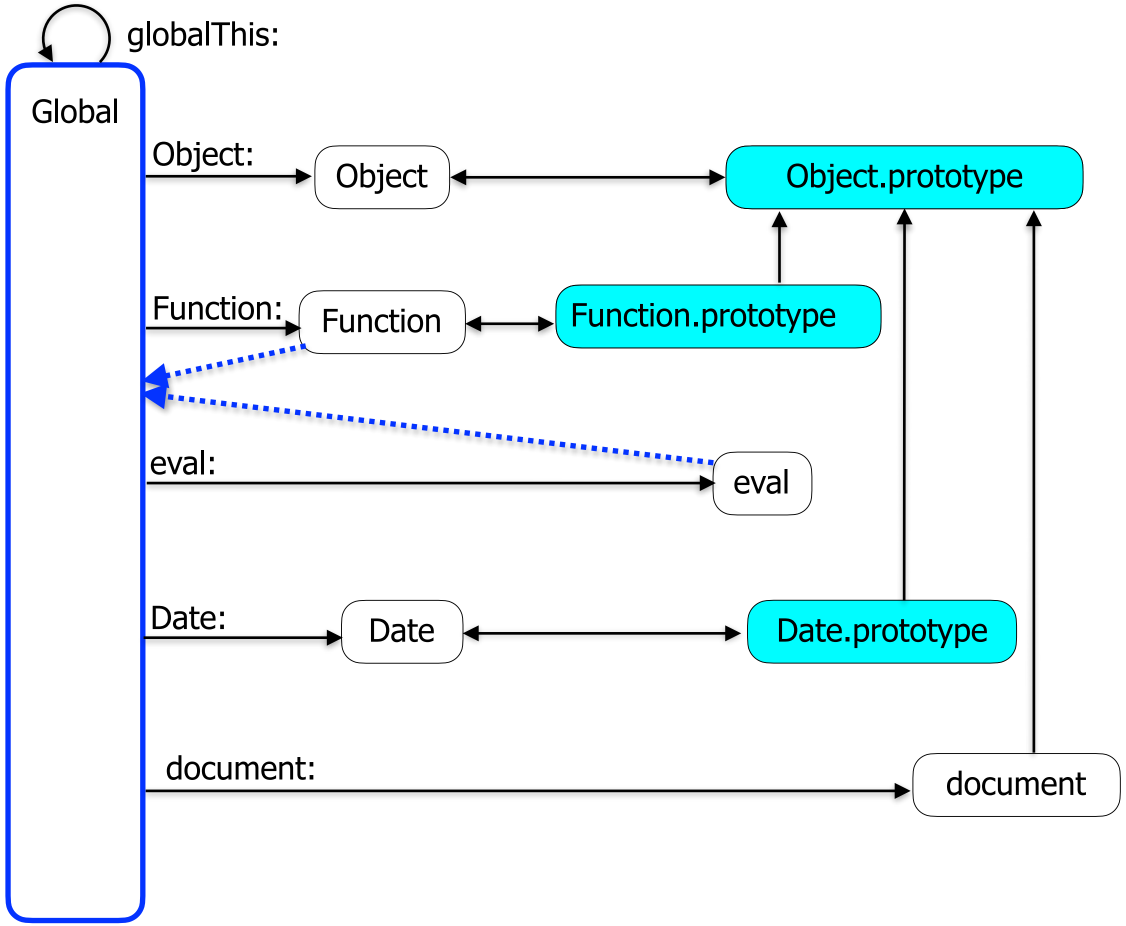

Object
Object.prototype
Function
Function.prototype
eval
Date
Date.prototype

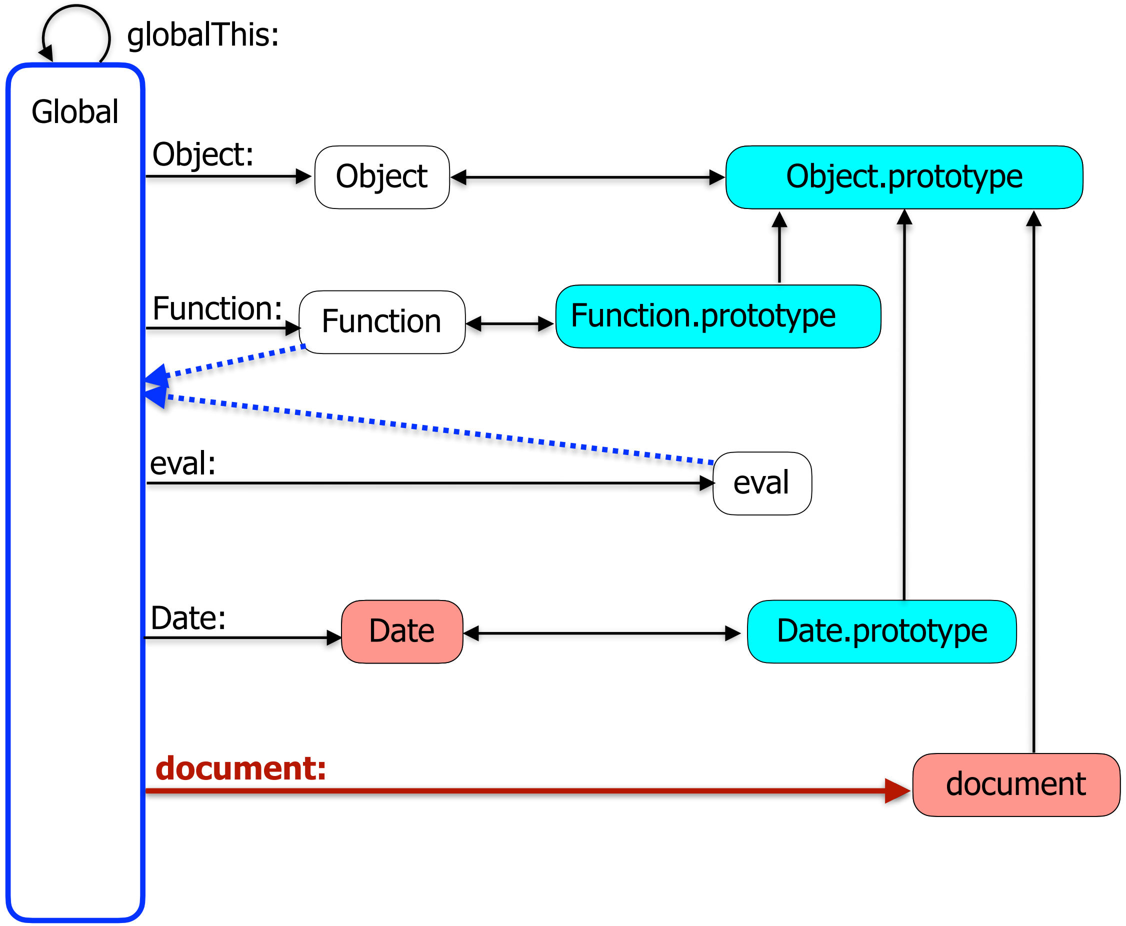

Object
Object.prototype
Function
Function.prototype
eval
Date
Date.prototype

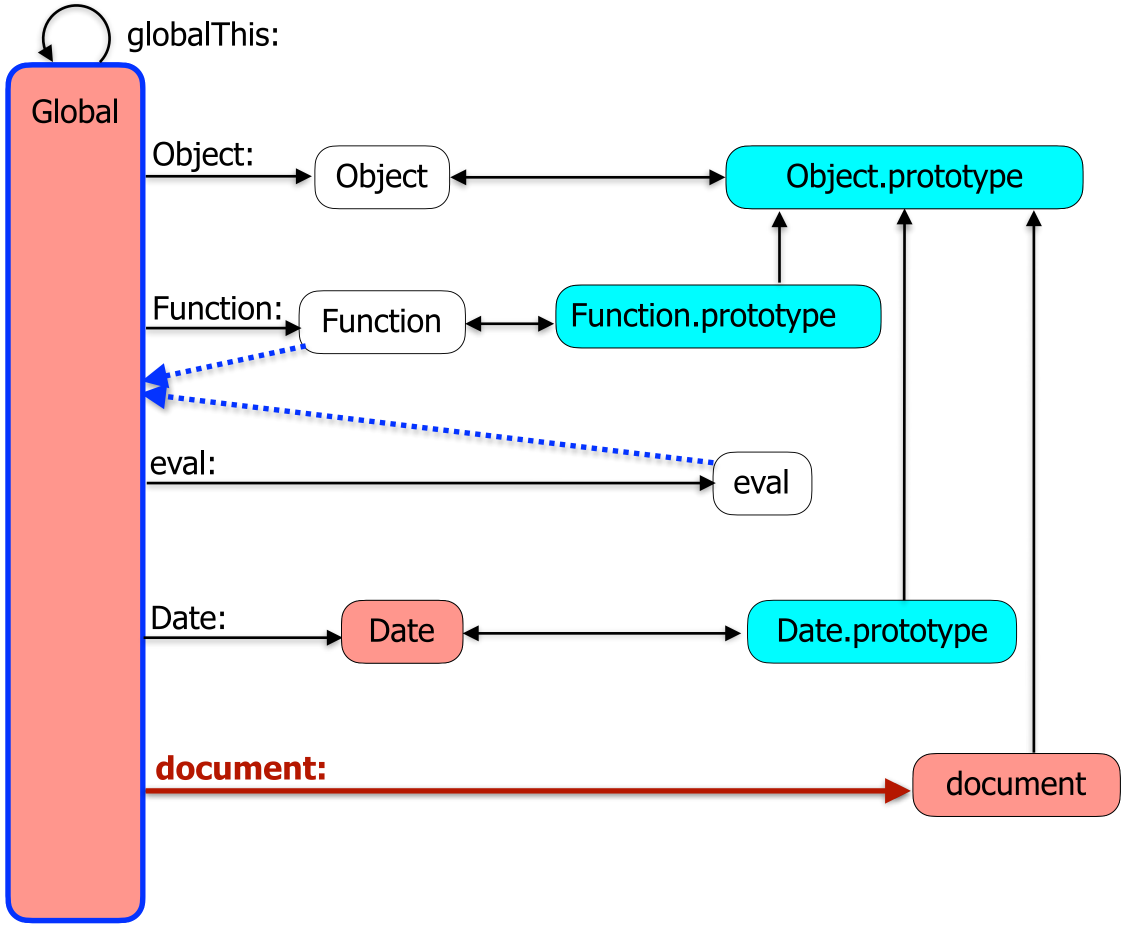

Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype

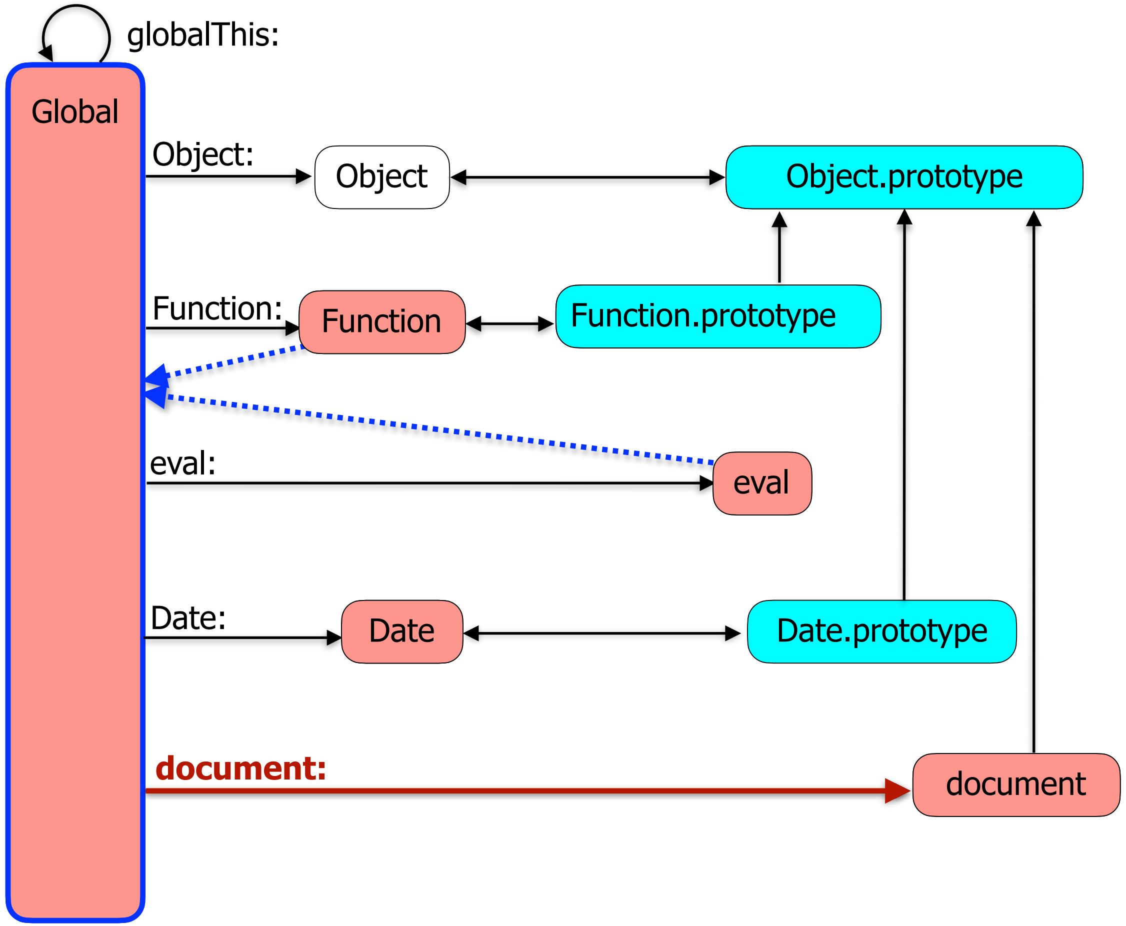

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype

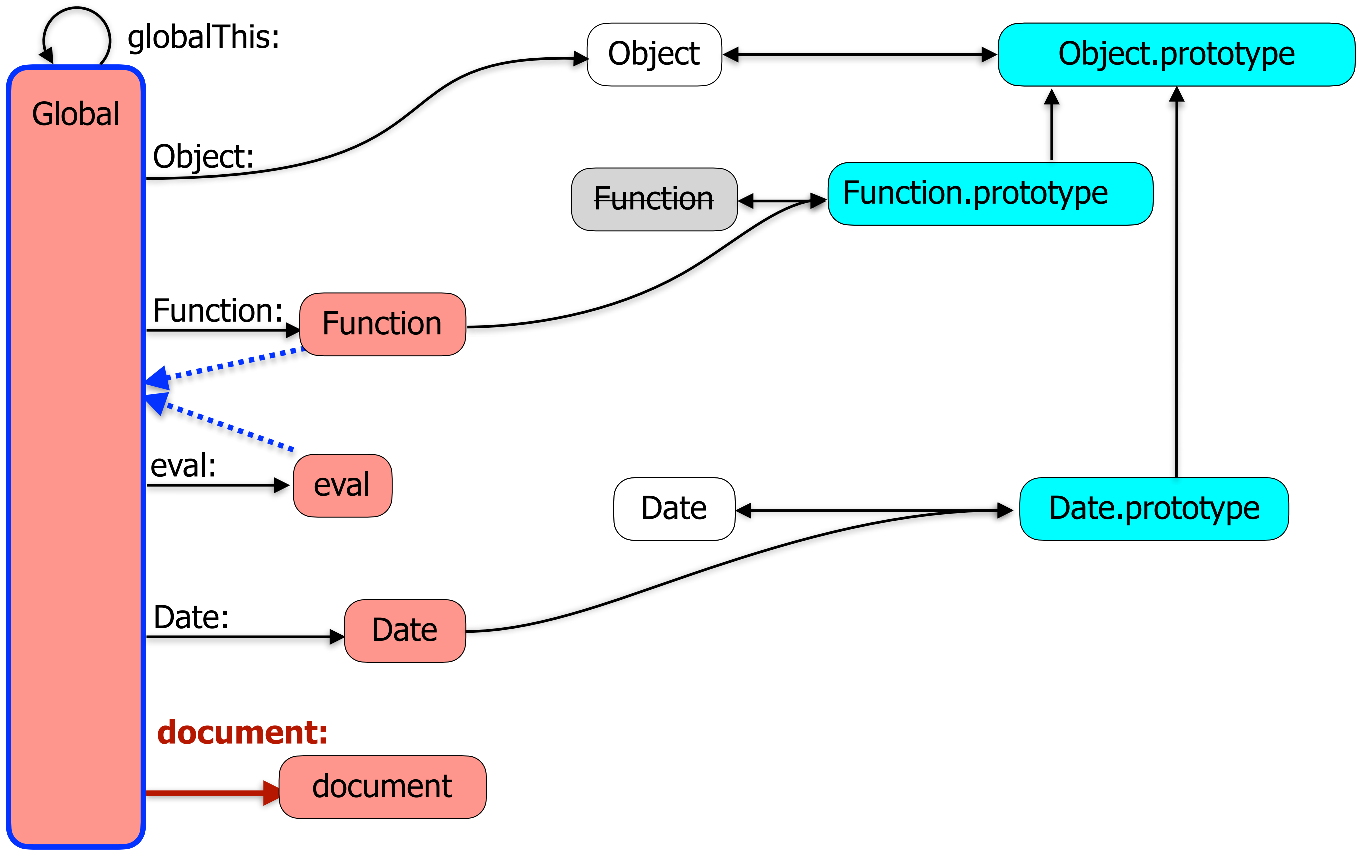

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype

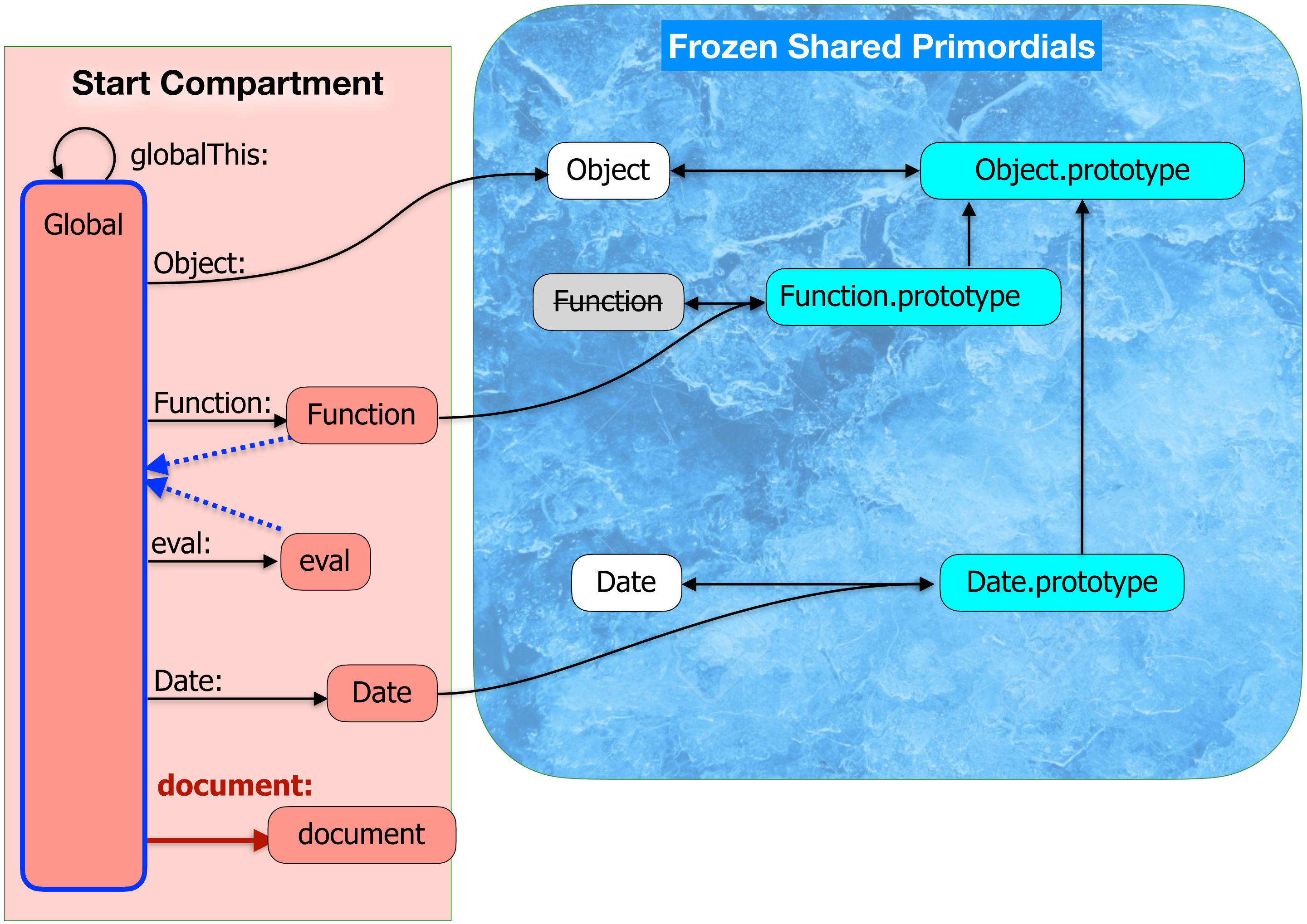

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype
document:
document

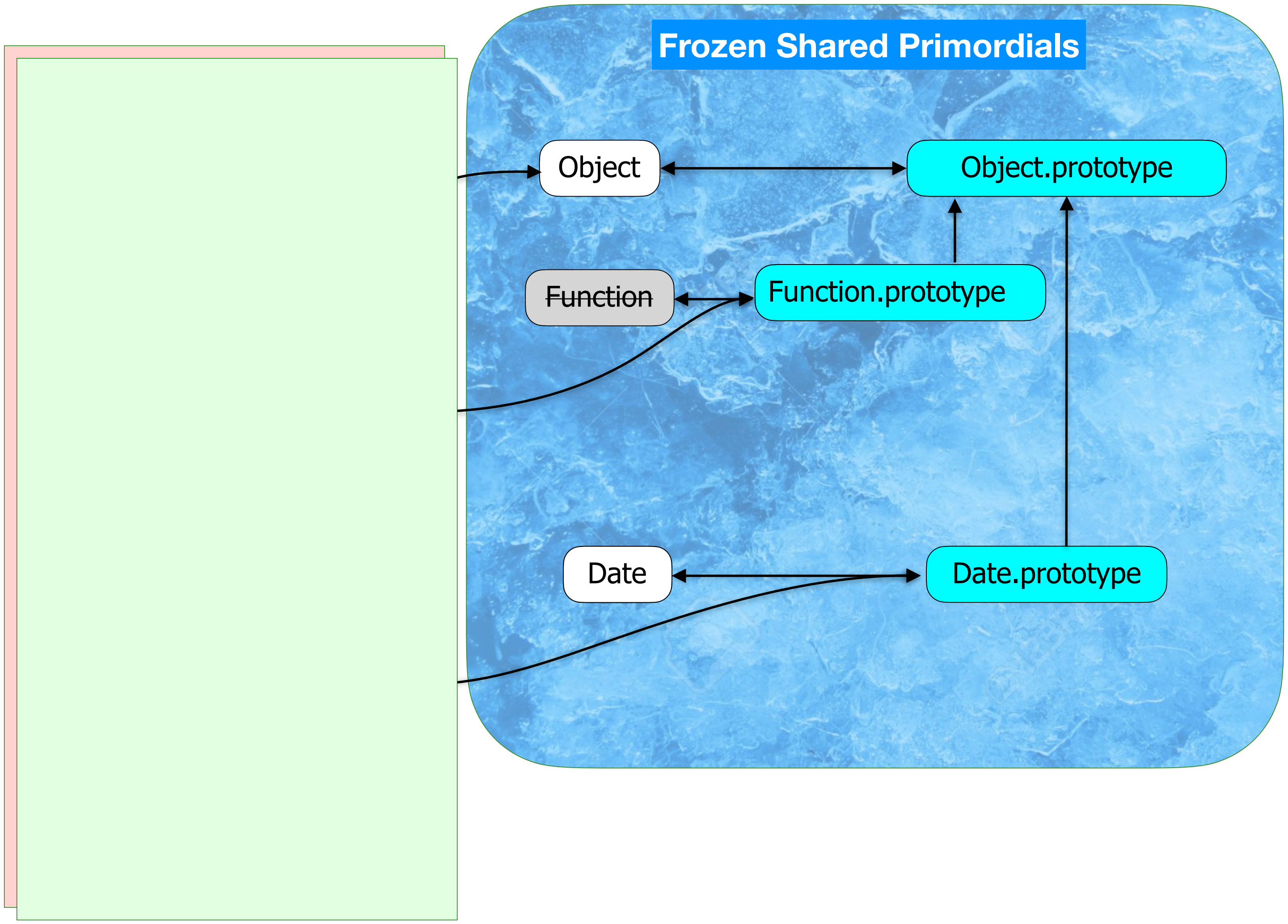

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype
document:
document

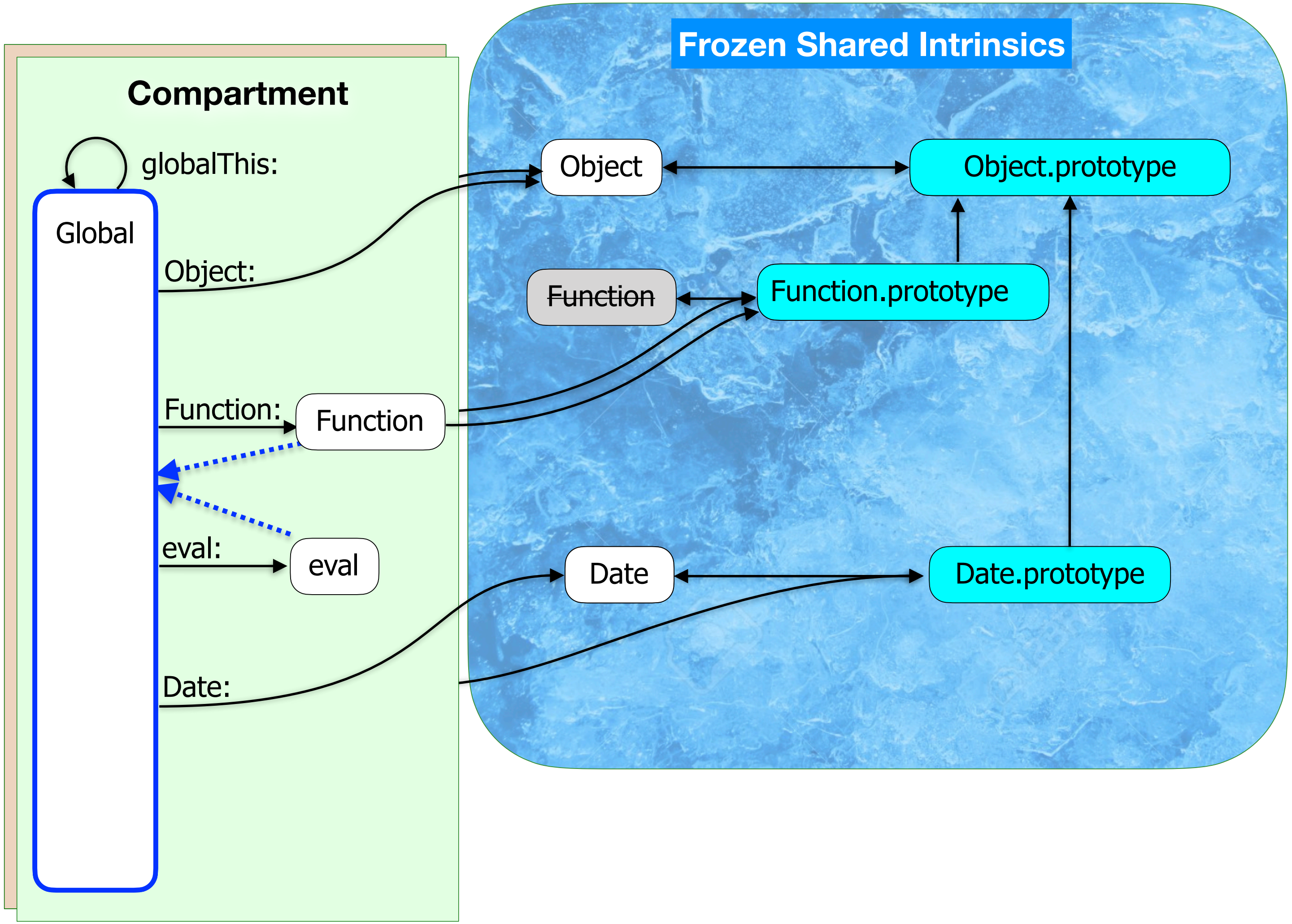

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype
document:
document

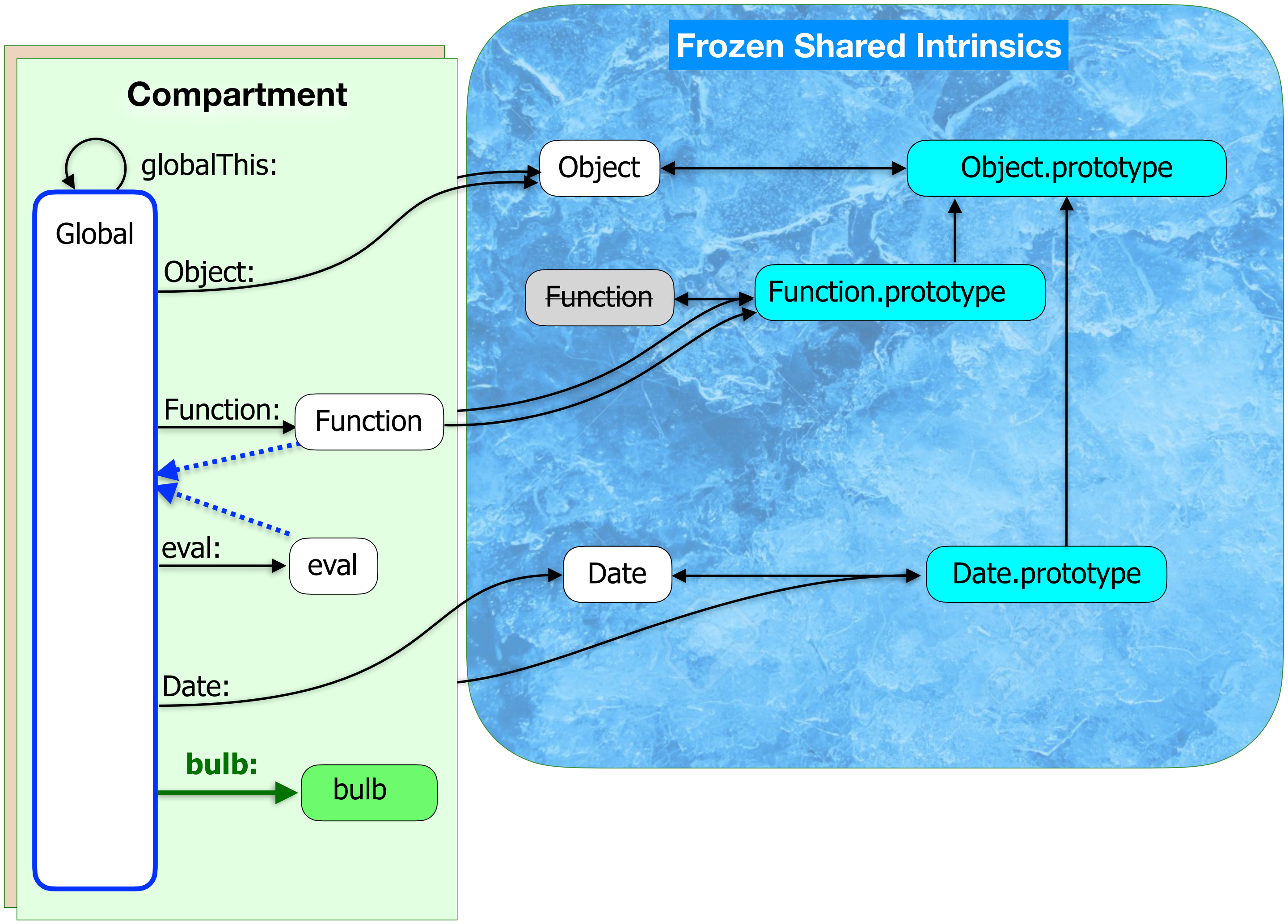

globalThis:
Global
Object:
Object
Object.prototype
Function:
Function
Function.prototype
eval:
eval
Date:
Date
Date.prototype
document:
document

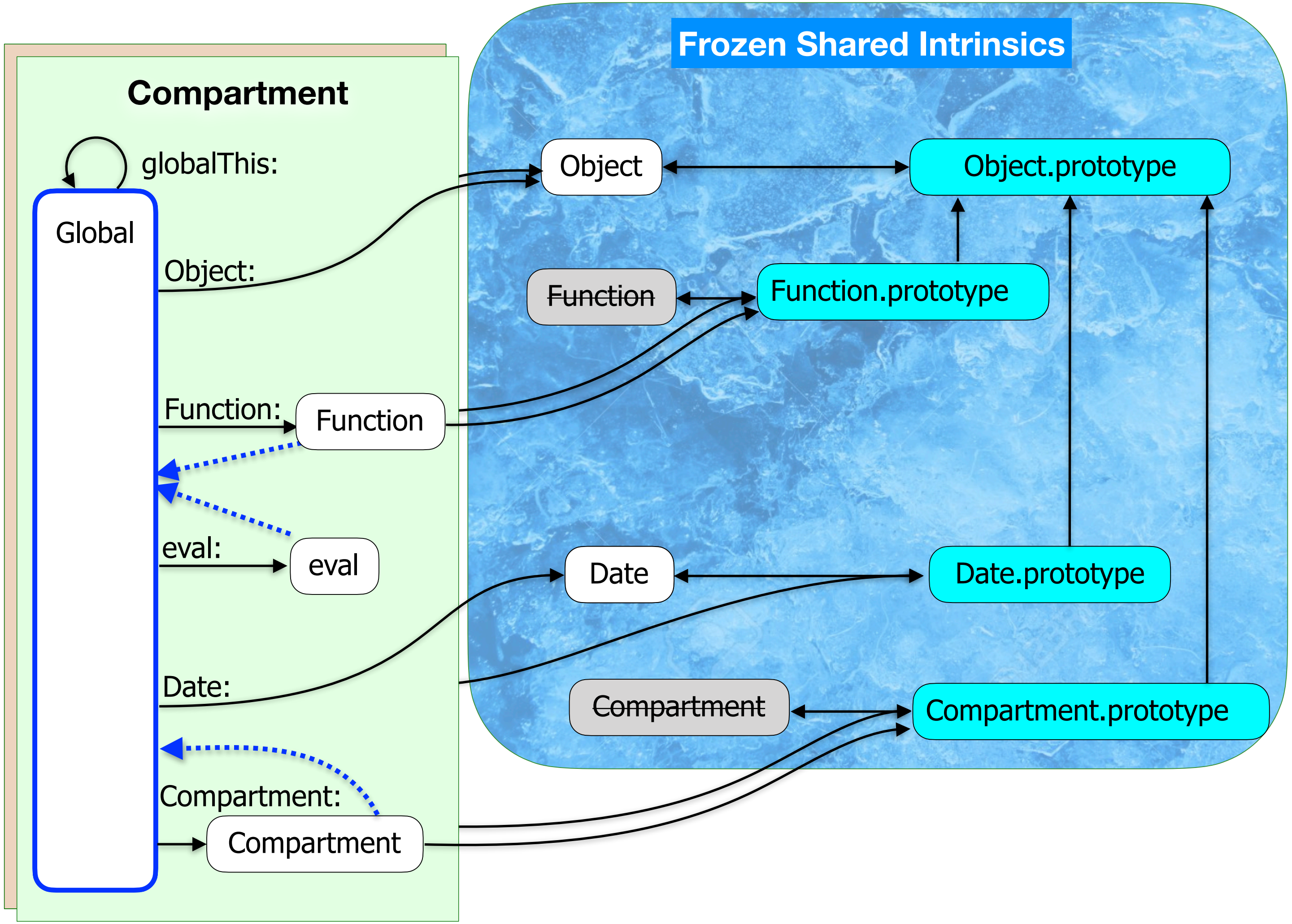

globalThis:
Global
Object:
Object
Object.prototype
~~Function~~
Function.prototype
Function:
Function
eval:
eval
Date
Date.prototype
Date:
Date
document:
document


Start Compartment
globalThis:
Global
Object:
Function:
Function
eval:
eval
Date:
Date
document:
document
Frozen Shared Primordials
Object
Object.prototype
~~Function~~
Function.prototype
Date
Date.prototype


Frozen Shared Primordials
Object
Object.prototype
~~Function~~
Function.prototype
Date
Date.prototype


Compartment
globalThis:
Global
Object:
Function:
Function
eval:
eval
Date:
Frozen Shared Intrinsics
Object
Object.prototype
~~Function~~
Function.prototype
Date
Date.prototype


Frozen Shared Intrinsics
Compartment
globalThis:
Global
Object:
Function:
Function
eval:
eval
Date:
bulb:
bulb
Object
Object.prototype
~~Function~~
Function.prototype
Date
Date.prototype


Frozen Shared Intrinsics
Compartment
globalThis:
Global
Object:
Function:
Function
eval:
eval
Date:
Compartment:
Compartment
Object
Object.prototype
~~Function~~
Function.prototype
Date
Date.prototype
~~Compartment~~
Compartment.prototype

Agoric

Taming of the Realm

- Tame RegExp constructor
- Tame Function constructors
- Disable Math.random
- Disable Date.now
- %SharedSymbol% snapshot
- &c

Caveats, Provisos, and Quid Pro Quos

A Hardened JavaScript shim admits an astonishing body of existing JavaScript, but a native implementation can admit even more.

Limitations of the emulation

- **Property override mistake**
- Censorship of eval, import, import.meta, and HTML comment **syntax**
- Revelation of module lexicals scope object
- Limitations to typeof

Agoric

Property Override Mistake

```
lockdown();

const x = {};
x.constructor = function () {};
x.toString = function () { return 'x'; };

Cannot assign to read only property 'constructor' of
object '[object Object]'
```

How can JavaScript Improve HardenedJS

Native implementations of Hardened JavaScript can confine nearly any JavaScript program.

Agoric

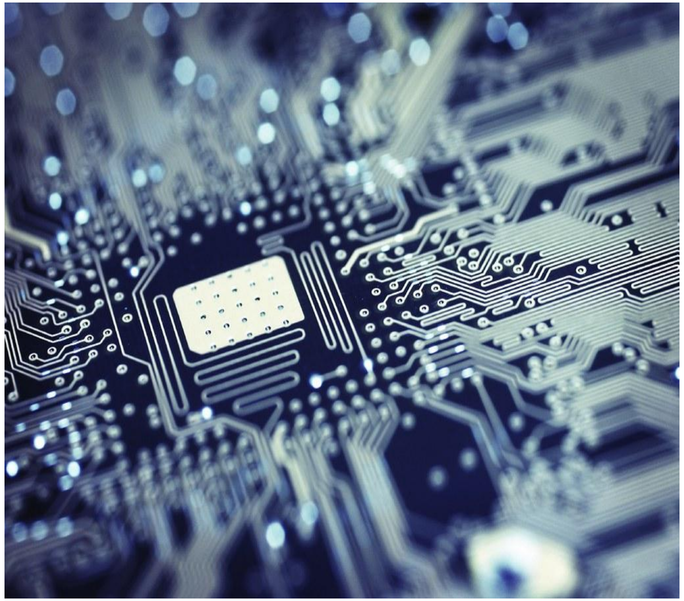
How JavaScript can grow to help

- Override integrity level
- Evaluators
- Module Harmony
- Please, move the stack back to Error prototype.

Agoric

Dear Google,

- What is **Hardened JavaScript**?
- Why do we need it?
- How do we emulate it?
- Limits to faithful emulation
- How can JavaScript evolve to better support Hardened JavaScript?



Agoric

References

- SES (Hardened JavaScript shim)
- TC39 Compartments Proposal

Agoric™