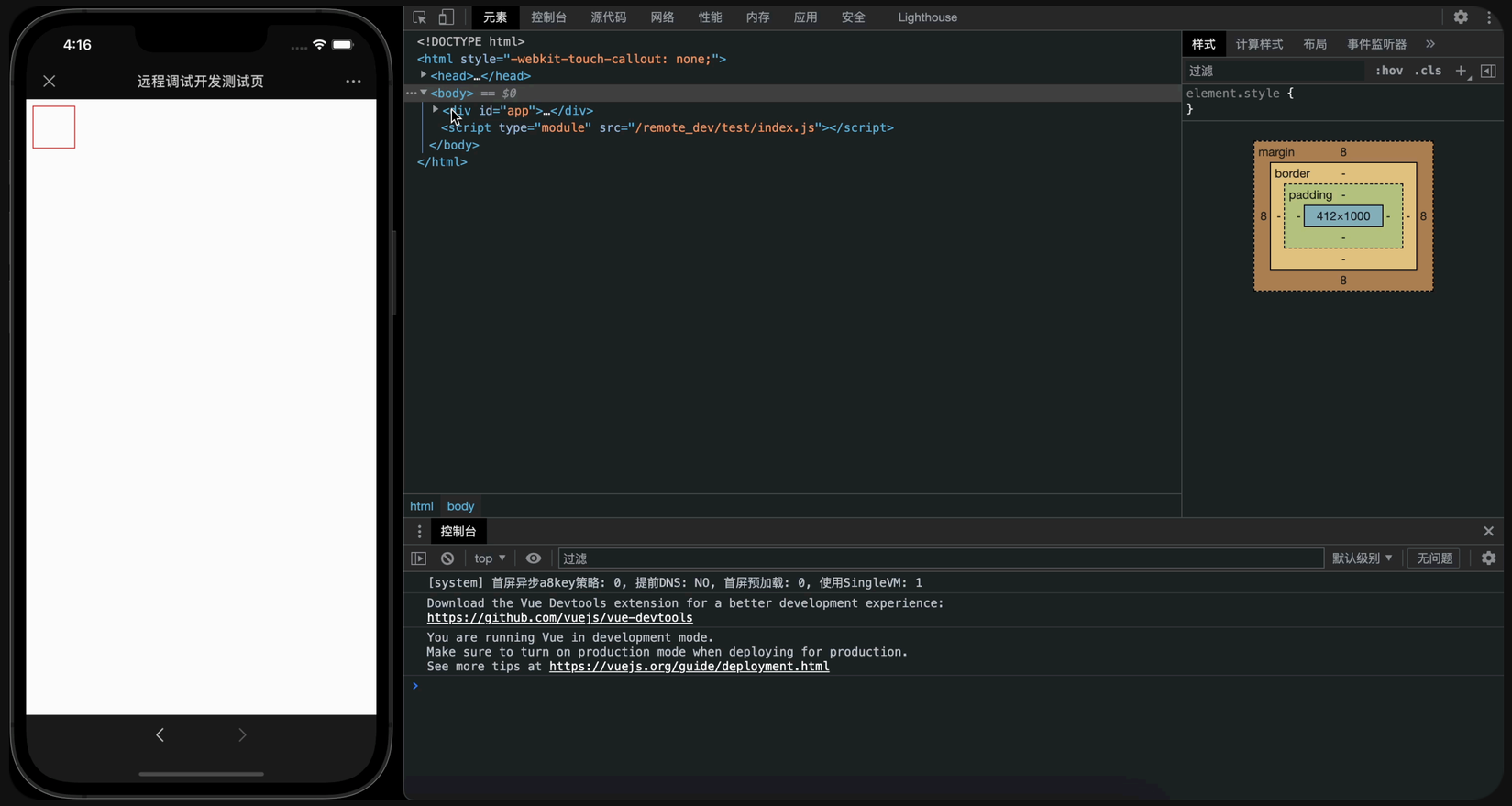
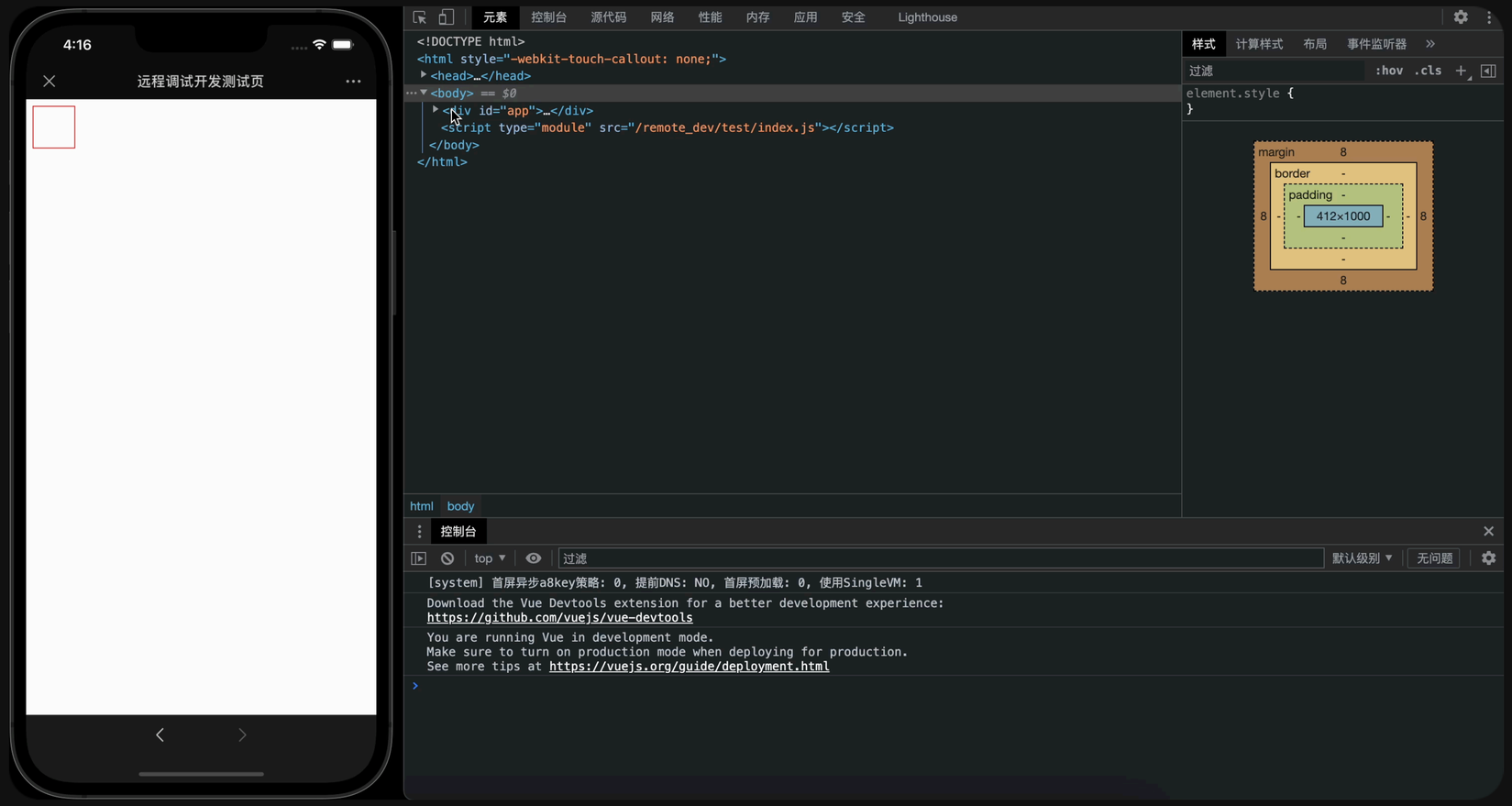
微信端内网页远程调试及断点原理

腾讯微信公众号团队　　邱焱坤 王熠弘

https://github.com/wechatjs/mprdev



https://github.com/wechatjs/mprdev




终端设备
SDK
WebSocket 转发服务
Chrome DevTools Protocol
Chrome
DevTools
调试面板

Chrome DevTools Protocol

https://chromedevtools.github.io/devtools-protocol

Console | Developer Resources | Protocol monitor | Coverage | Performance monitor

Filter

Type	Method	Request	Response	Elapsed time
	Page.captureScreenshot	{"format":"jpeg","clip":{"x":0,…	{"code":-32000,"message":"Canno…	2 ms
	Page.captureScreenshot	{"format":"jpeg","clip":{"x":0,…	(pending)	(pending)
	CSS.mediaQueryResultChanged		{}	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	Target.targetDestroyed		{"targetId":"CF650D04855F713908…	
	Target.targetDestroyed		{"targetId":"CF650D04855F713908…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	Network.getAllCookies	{}	{"cookies":[{"name":"SNID","val…	43 ms
	Target.targetCreated		{"targetInfo":{"targetId":"D0A4…	
	Target.targetCreated		{"targetInfo":{"targetId":"D0A4…	
	ServiceWorker.workerRegistrationUpdated		{"registrations":[{"registratio…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	
	Target.targetDestroyed		{"targetId":"D0A482276B6690513F…	
	Target.targetDestroyed		{"targetId":"D0A482276B6690513F…	
	ServiceWorker.workerVersionUpdated		{"versions":[{"versionId":"5","…	

Request | Response


```
▼ {cookies: [{name: "SNID",…},…]}
  ▼ cookies: [{name: "SNID",…},…]
    ▶ 0: {name: "SNID",…}
    ▶ 1: {name: "OTZ", value: "7112637_56_56__56_", domain:
    ▶ 2: {name: "gnubbyCookie", value: "true", domain: "logi
    ▶ 3: {name: "OTZ", value: "7112638_56_56__56_", domain:
    ▶ 4: {name: "OTZ", value: "7112704_56_56__56_", domain:
    ▶ 5: {name: "1P_JAR", value: "2023-07-13-08", domain: ".
    ▶ 6: {name: "AEC", value: "Ad49MVEsets2moc0kRAxmmbcgrSf0
    ▶ 7: {name: "NID",…}
    ▶ 8: {name: "_ga", value: "GA1.3.512623406.1689327323",
    ▶ 9: {name: "_octo", value: "GH1.1.76868201.1690276367",
    ▶ 10: {name: "_device_id", value: "35a6f62f77f8e19d6f9d3
    ▶ 11: {name: "user_session", value: "BowKpSblkFzdXFwhWnB
    ▶ 12: {name: "__Host-user_session_same_site", value: "Bo
    ▶ 13: {name: "logged_in", value: "yes", domain: ".github
    ▶ 14: {name: "dotcom_user", value: "hadrijau", domain: "
    ▶ 15: {name: "AEC", value: "Ad49MVF2uzwyUJofBqf4OMNjVGTb
    ▶ 16: {name: "VISITOR_INFO1_LIVE", value: "g7m1cq2opvQ",
    ▶ 17: {name: "1P_JAR", value: "2023-07-28-08", domain: "
    ▶ 18: {name: "_ga", value: "GA1.2.1646756824.1690533538"
    ▶ 19: {name: "__Host-GAPS", value: "1:HtZEtPrUPmPAyBKGQA
    ▶ 20: {name: "_gid", value: "GA1.3.751670330.1691399098"
    ▶ 21: {name: "NID",…}
    ▶ 22: {name: "S", value: "sso=Ipo54DDNDLz1iS2ltB2On5-Q-M
    ▶ 23: {name: "has_recent_activity", value: "1", domain:
    ▶ 24: {name: "preferred_color_mode", value: "light", dom
    ▶ 25: {name: "tz", value: "UTC", domain: ".github.com",
    ▶ 26: {name: "color_mode",…}
    ▶ 27: {name: "ASLBSA", value: "0003d087a9232136b2c757906
    ▶ 28: {name: "ASLBSACORS", value: "0003d087a9232136b2c75
    ▶ 29: {name: "_gh_sess",…}
```

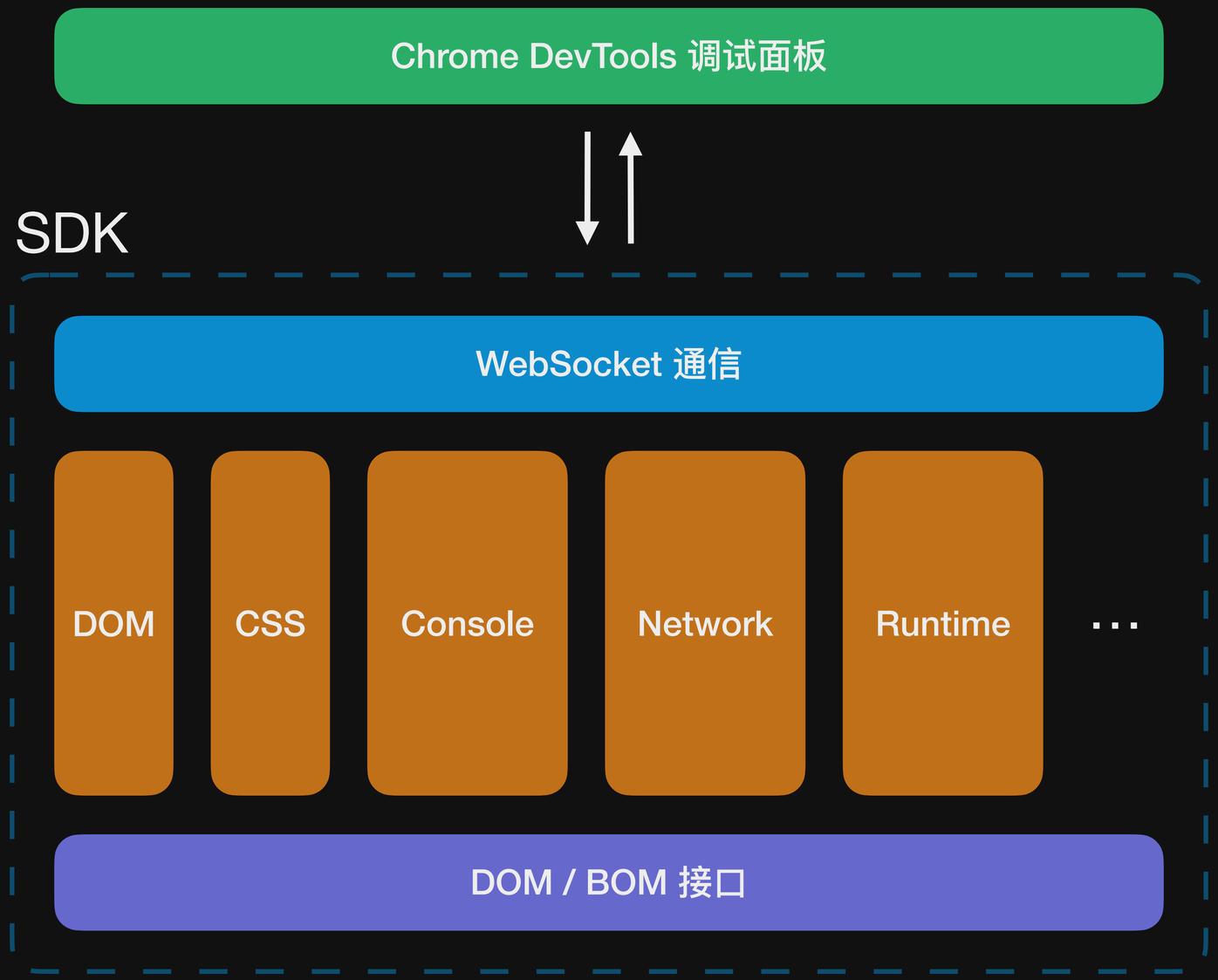

{"command":"Network.getAllCookies"}

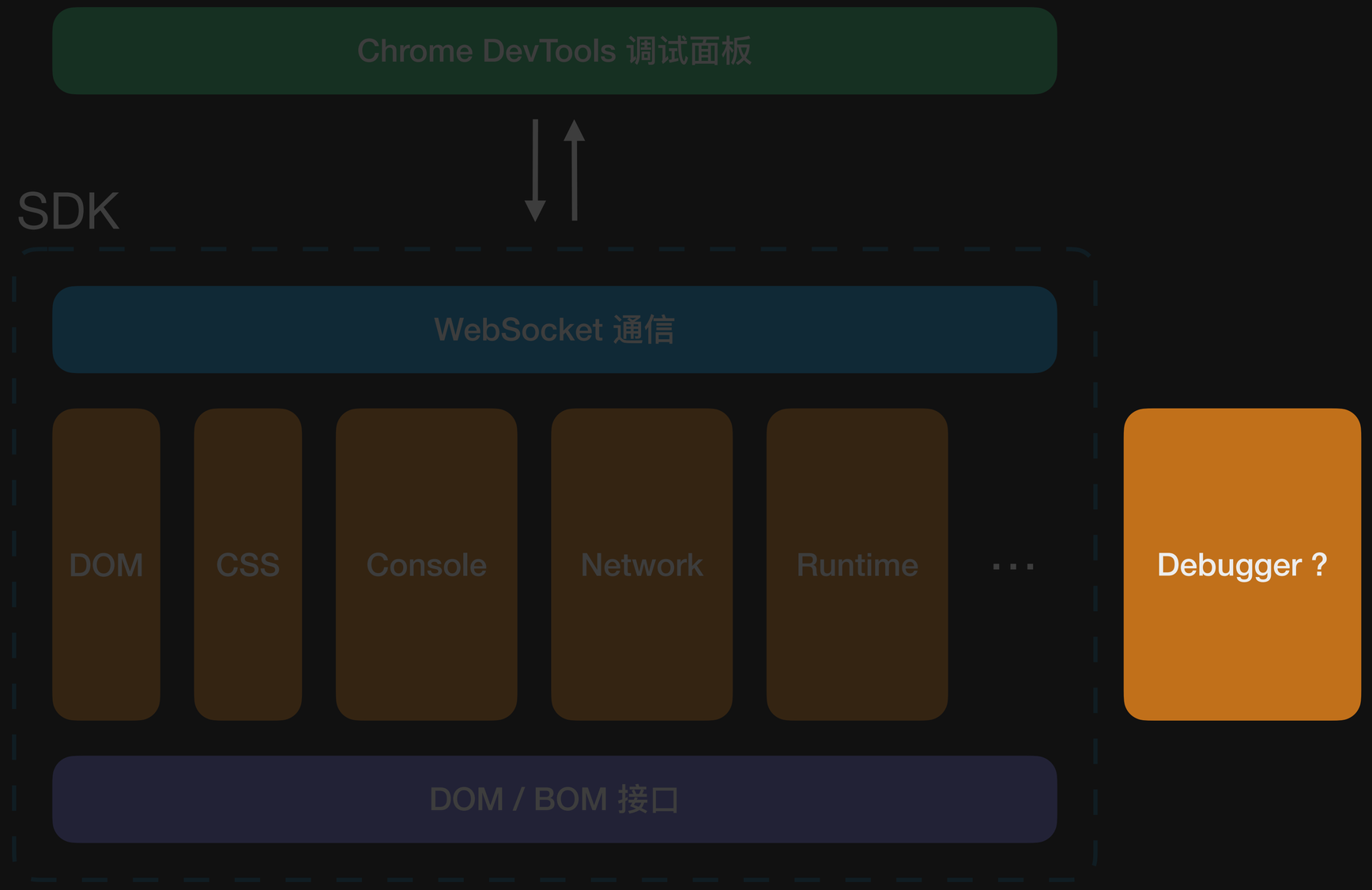
Tab ()

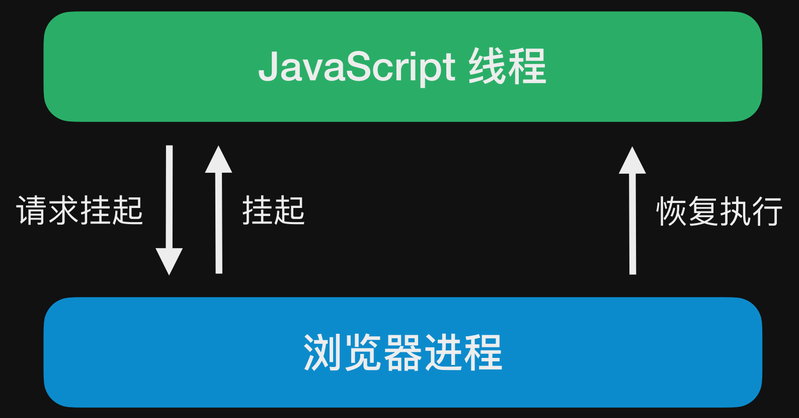

终端设备
SDK

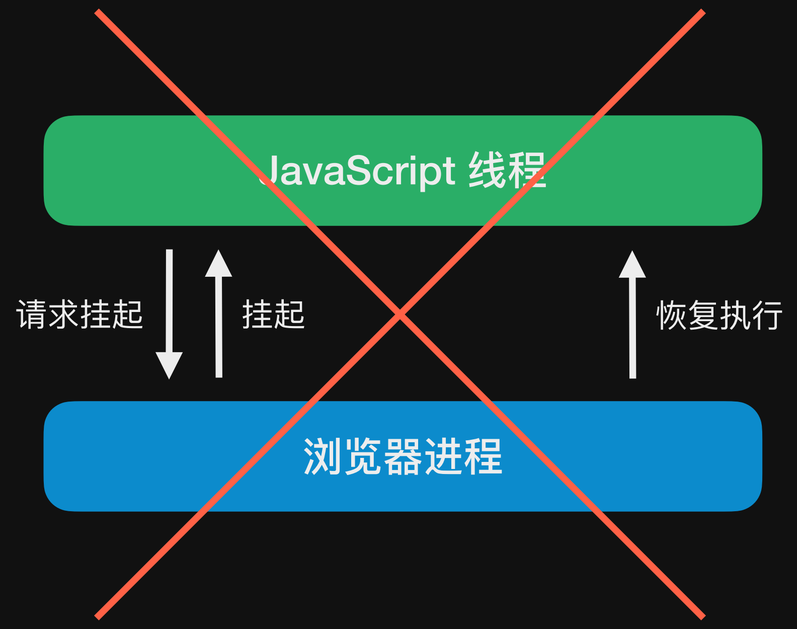
DOM接口 ✅

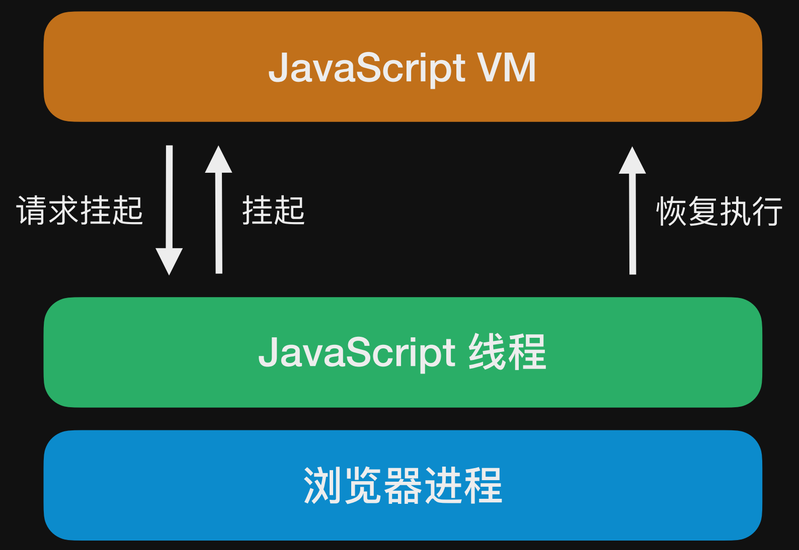
BOM接口 ✅


Chrome DevTools 调试面板
SDK
WebSocket 通信
DOM
CSS
Console
Network
Runtime
...
DOM / BOM 接口

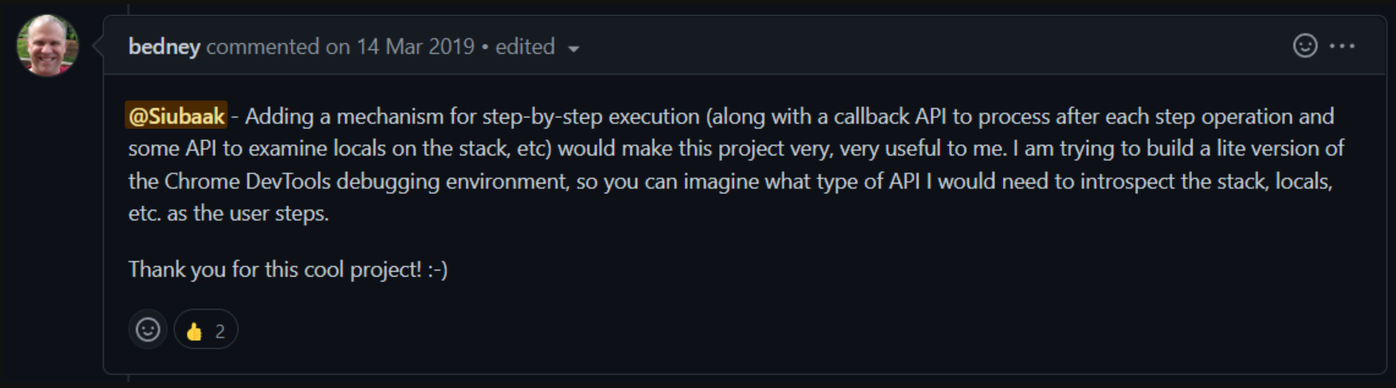

Chrome DevTools 调试面板
SDK
WebSocket 通信
DOM
CSS
Console
Network
Runtime
...
Debugger ?
DOM / BOM 接口


JavaScript 线程
请求挂起
挂起
恢复执行
浏览器进程


JavaScript 线程
请求挂起
挂起
恢复执行
浏览器进程


JavaScript VM
请求挂起
挂起
恢复执行
JavaScript 线程
浏览器进程

https://github.com/Siubaak/sval/tree/develop

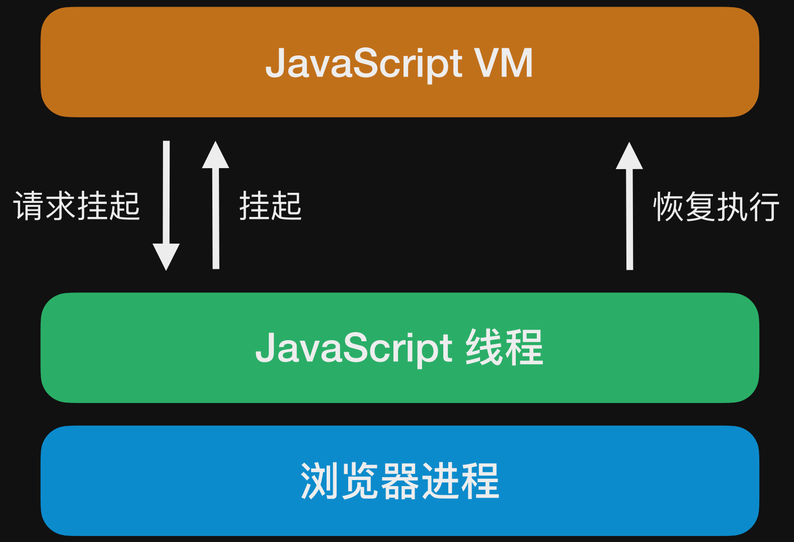


bedney commented on 14 Mar 2019 • edited

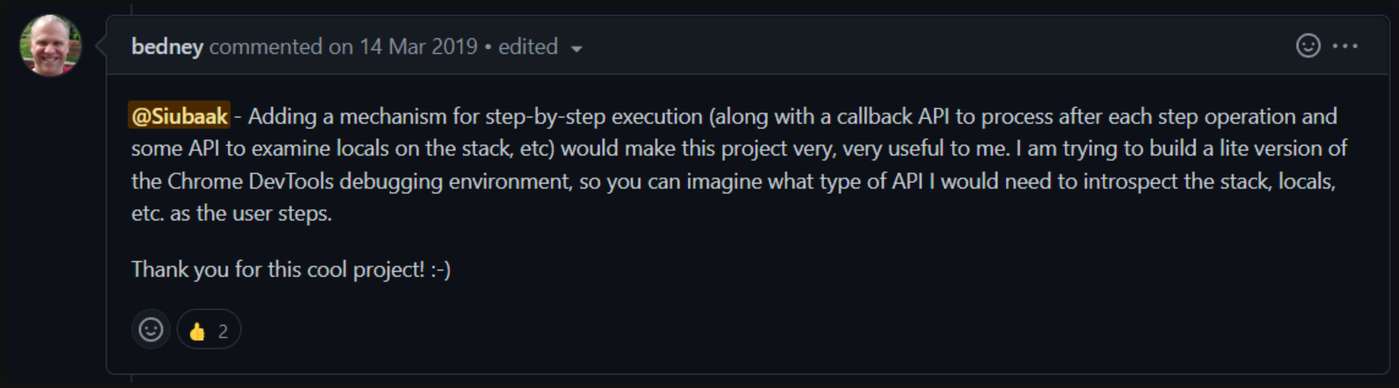
@Siubaak - Adding a mechanism for step-by-step execution (along with a callback API to process after each step operation and some API to examine locals on the stack, etc) would make this project very, very useful to me. I am trying to build a lite version of the Chrome DevTools debugging environment, so you can imagine what type of API I would need to introspect the stack, locals, etc. as the user steps.

Thank you for this cool project! :-)

2



https://github.com/Siubaak/sval/tree/develop

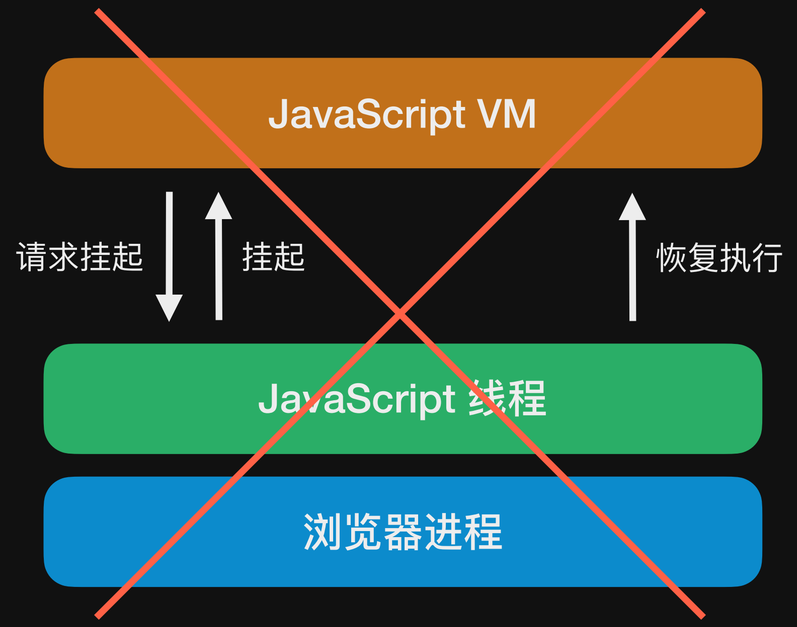


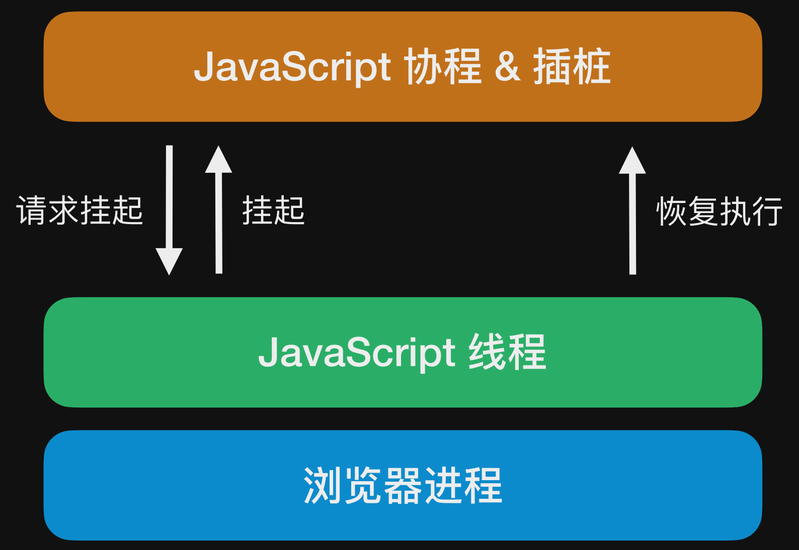
bedney commented on 14 Mar 2019 • edited

@Siubaak - Adding a mechanism for step-by-step execution (along with a callback API to process after each step operation and some API to examine locals on the stack, etc) would make this project very, very useful to me. I am trying to build a lite version of the Chrome DevTools debugging environment, so you can imagine what type of API I would need to introspect the stack, locals, etc. as the user steps.

Thank you for this cool project! :-)

2



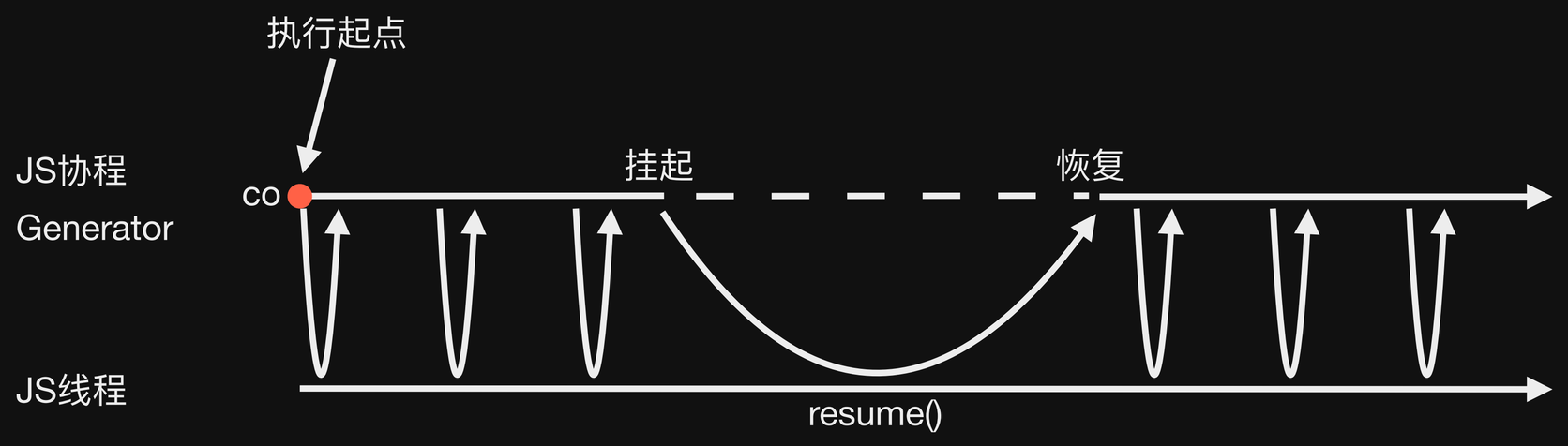

JavaScript 协程 & 插桩
请求挂起
挂起
恢复执行
JavaScript 线程
浏览器进程

Generator	Async Function
同步 ✅	异步 微任务堆积 时序问题
性能好 ✅	性能较差
实现较难 手动迭代	实现简单 ✅ 自动执行

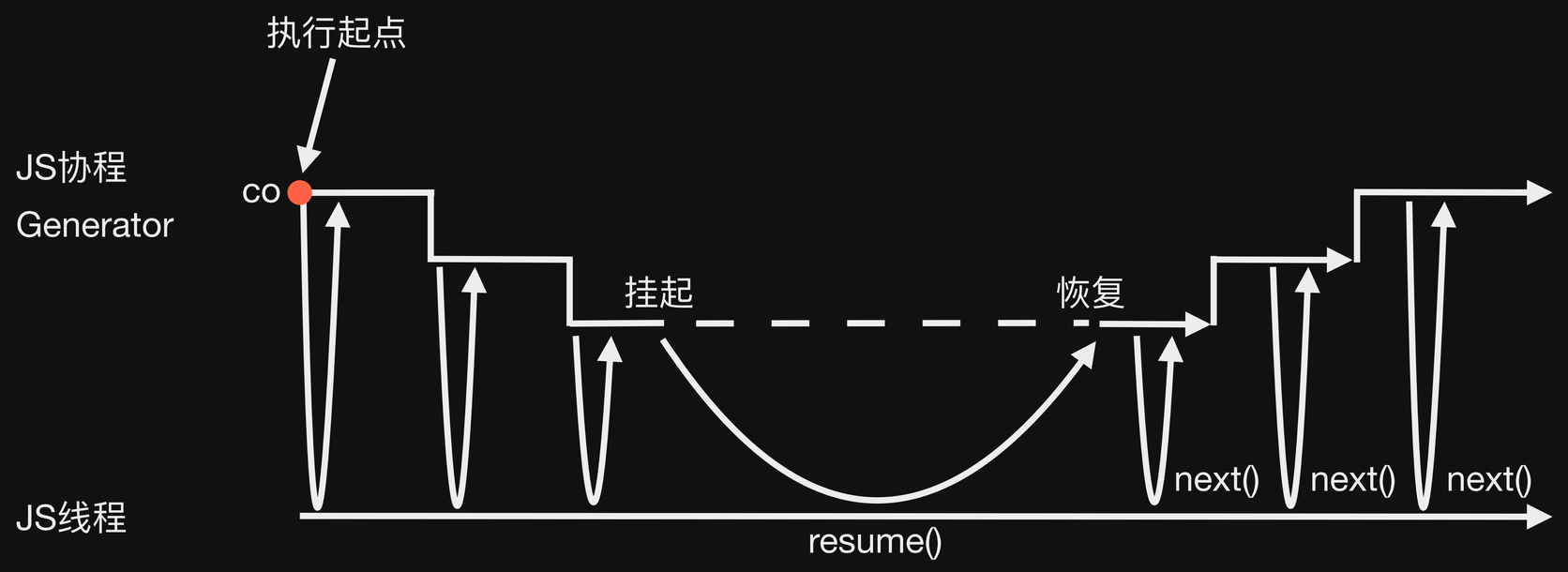
Generator	Async Function
同步 ✅	异步
	微任务堆积
	时序问题
性能好 ✅	性能较差
实现较难	实现简单 ✅
手动迭代	自动执行

```
function A() { co(...) }
const resume = A() // break
resume() // resume
```

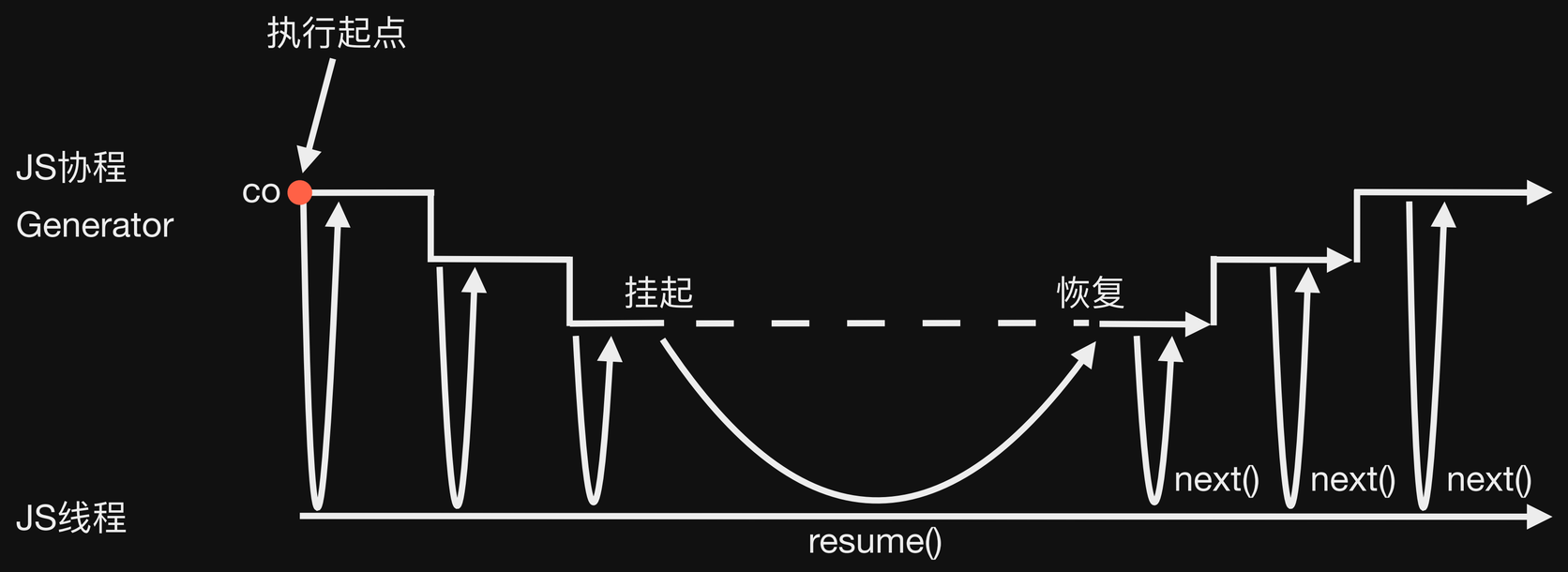
https://github.com/tj/co



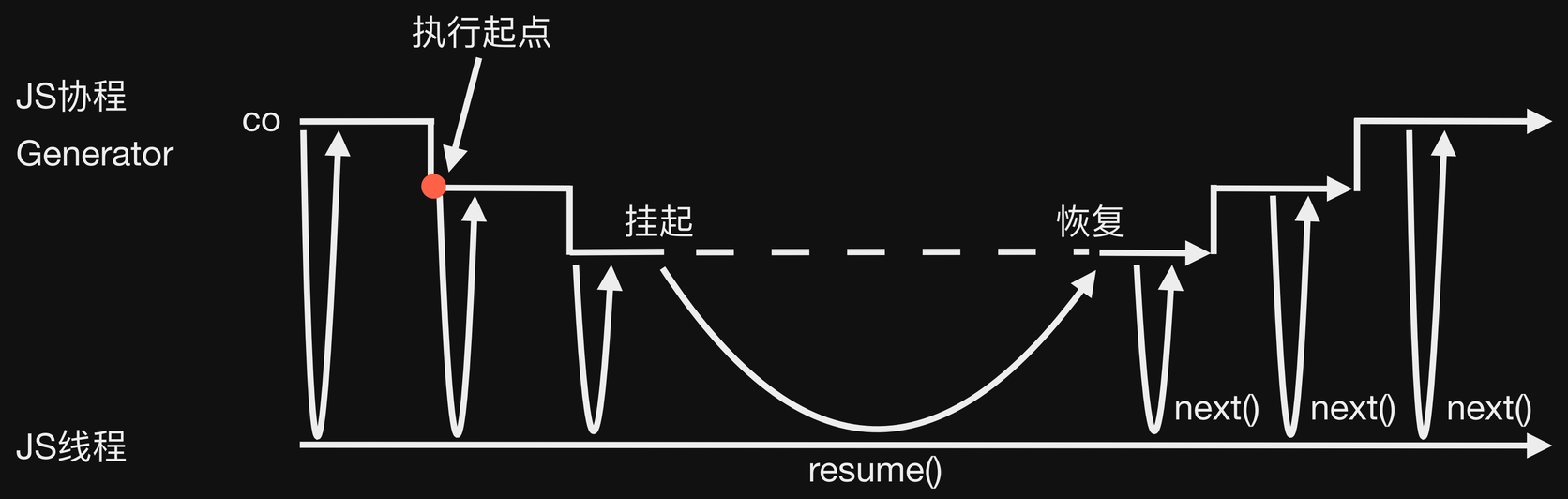
```
function A() { co(B) }
function* B() { yield* C() }
function* C() { yield ... }
const resume = A() // break
resume() // resume
```



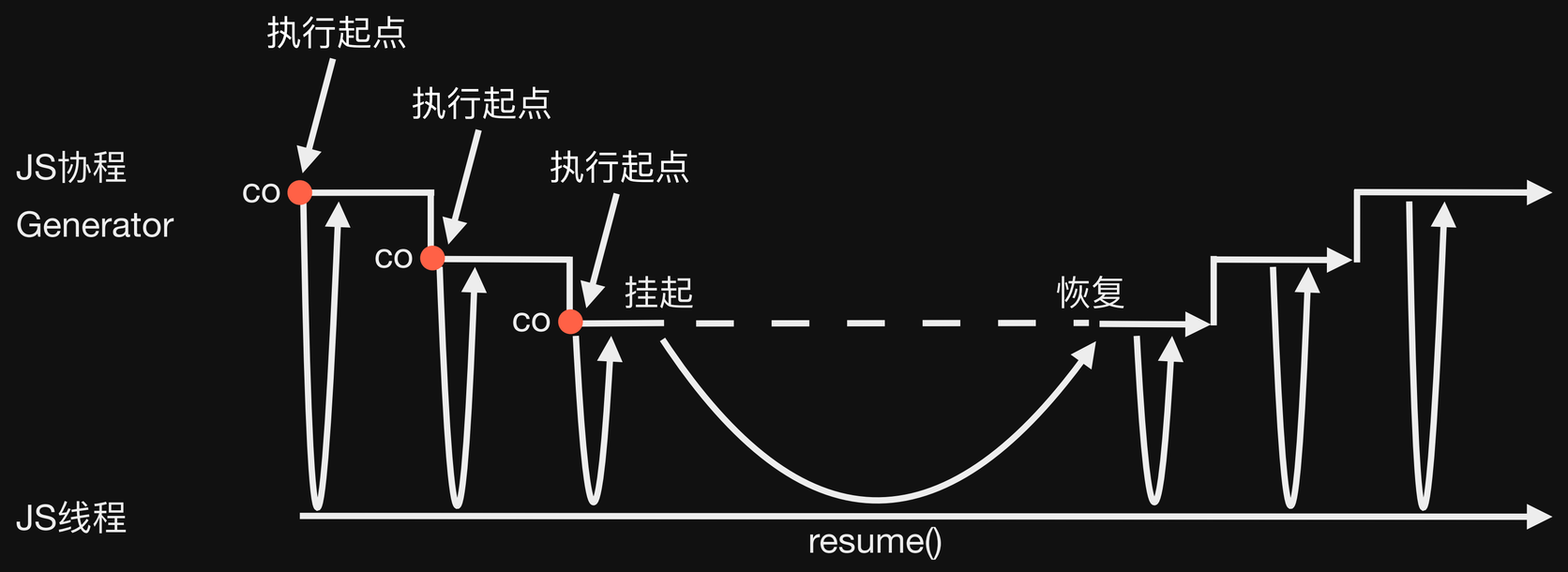
```
function A() { co(B) }
function* B() { yield* C() }
function* C() { yield ... }
const resume = A() // break
resume() // resume
```

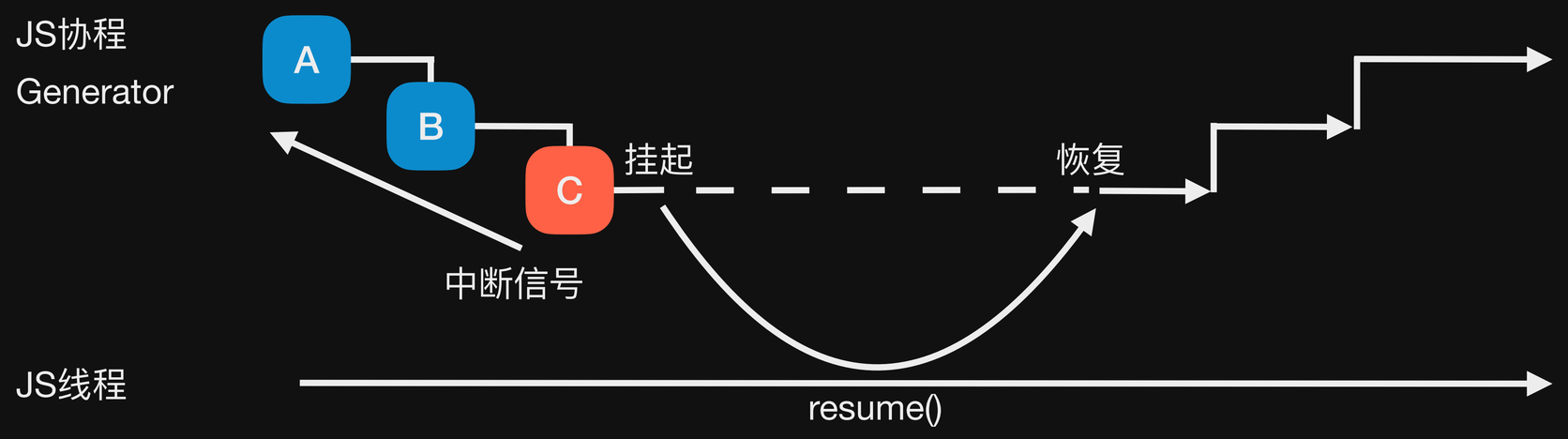


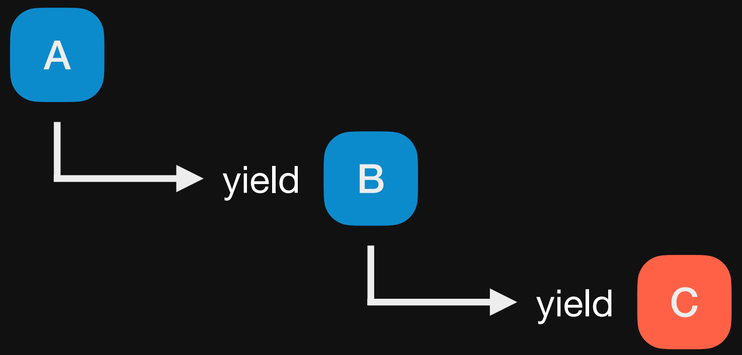
```
function A() { co(B) }
function* B() { yield* C() }
function* C() { yield ... }
const resume = A() // break
resume() // resume
```

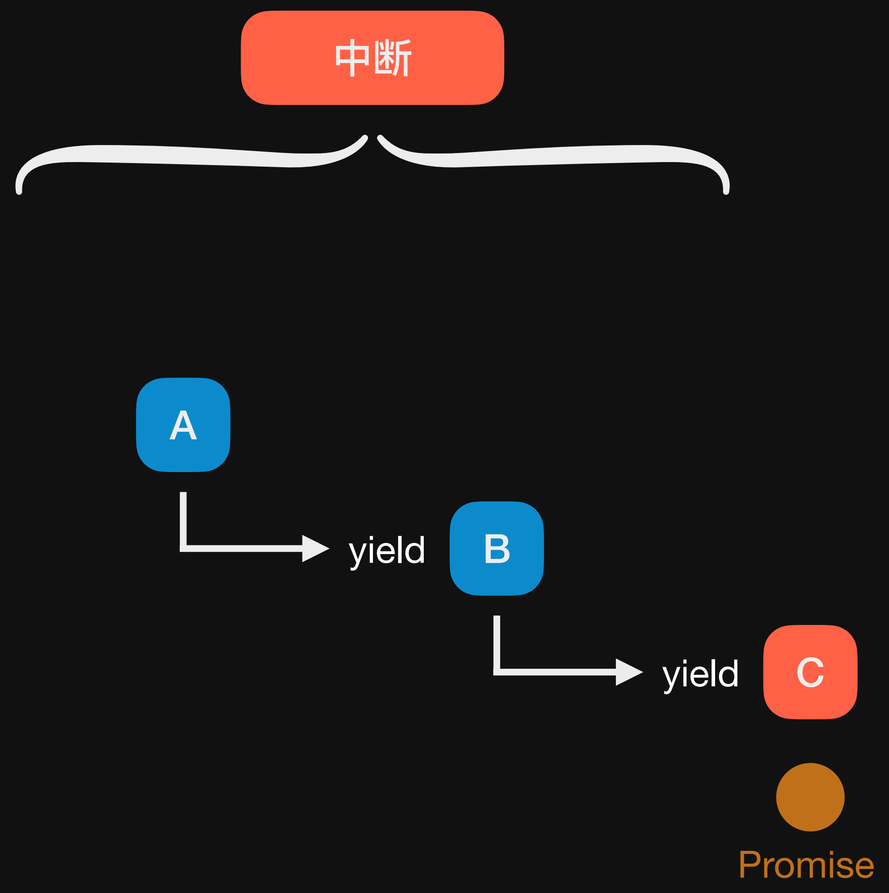


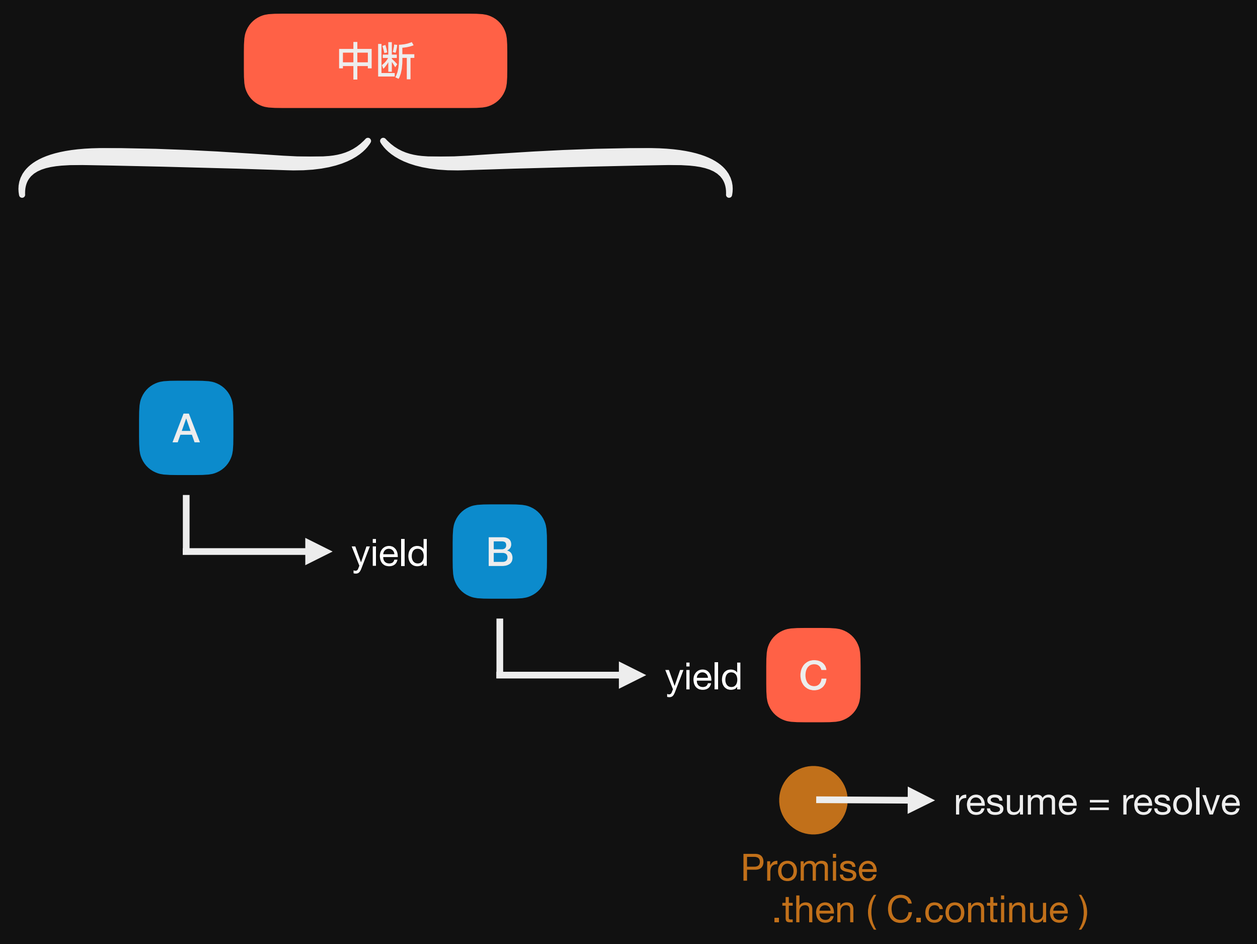
```
function A() { co(B) }
function B() { co(C) }
function C() { ... }
const resume = A() // break
resume() // resume
```

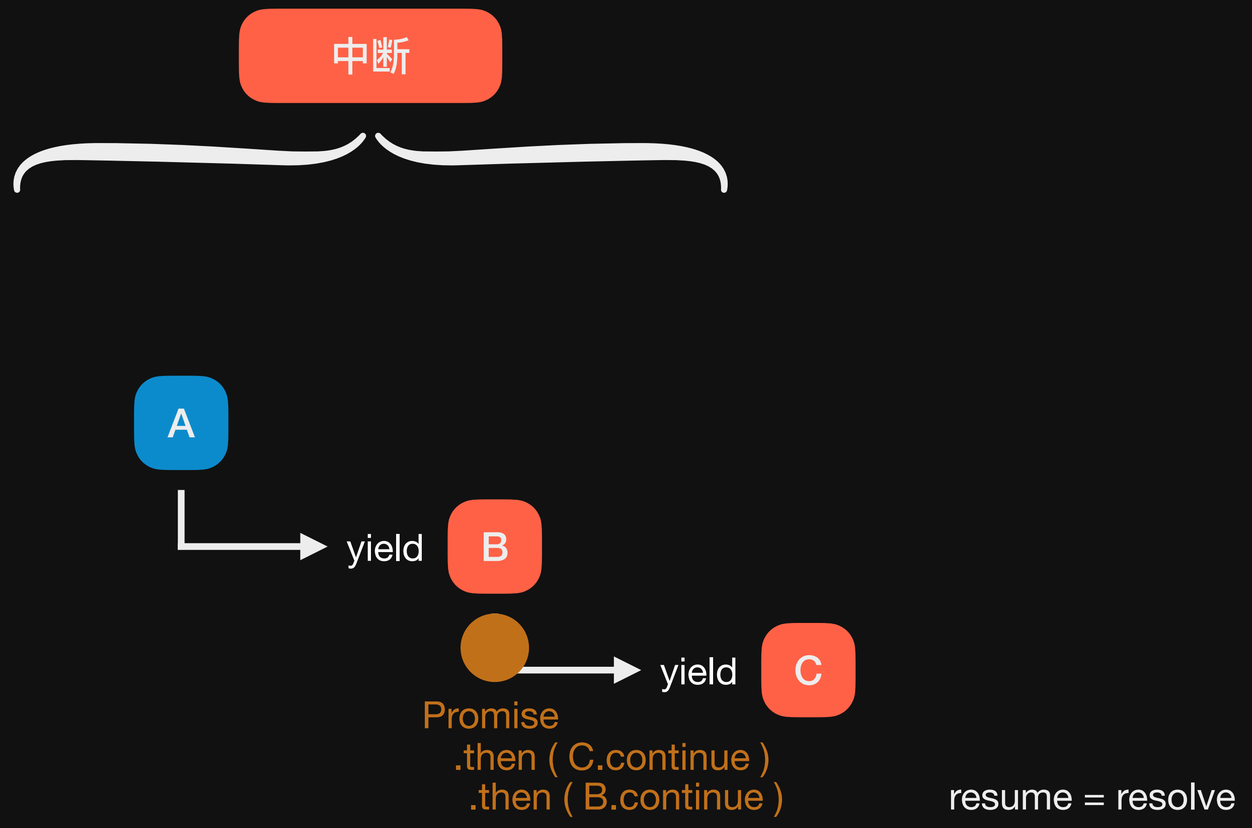



JS协程
Generator
A
B
C
挂起
恢复
中断信号
JS线程
resume()

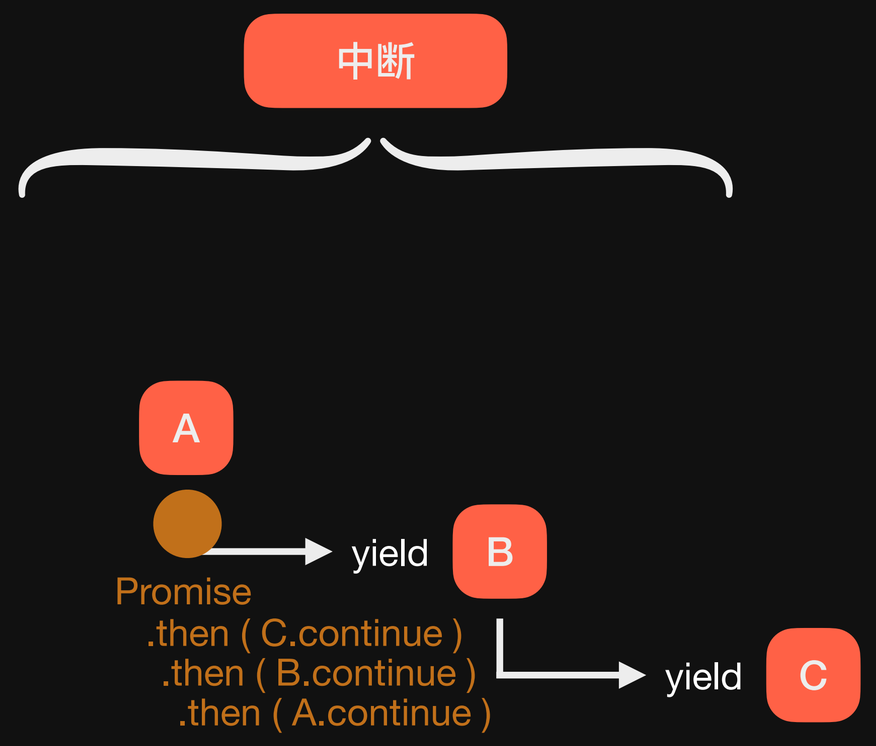

A
yield
B
yield
C


中断
A
yield
B
yield
C
Promise

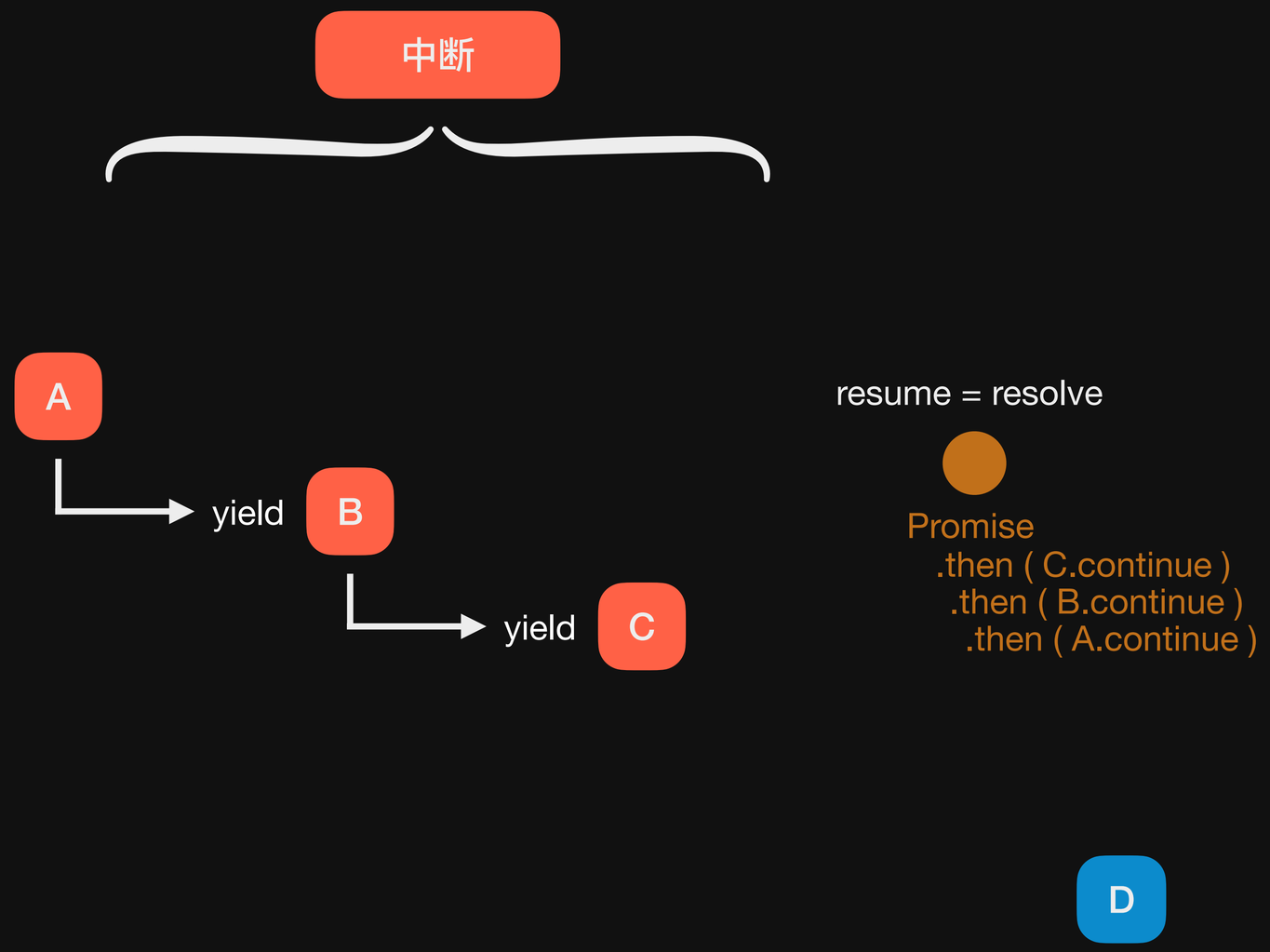

中断
A
yield
B
yield
C
resume = resolve
Promise
.then (C.continue)

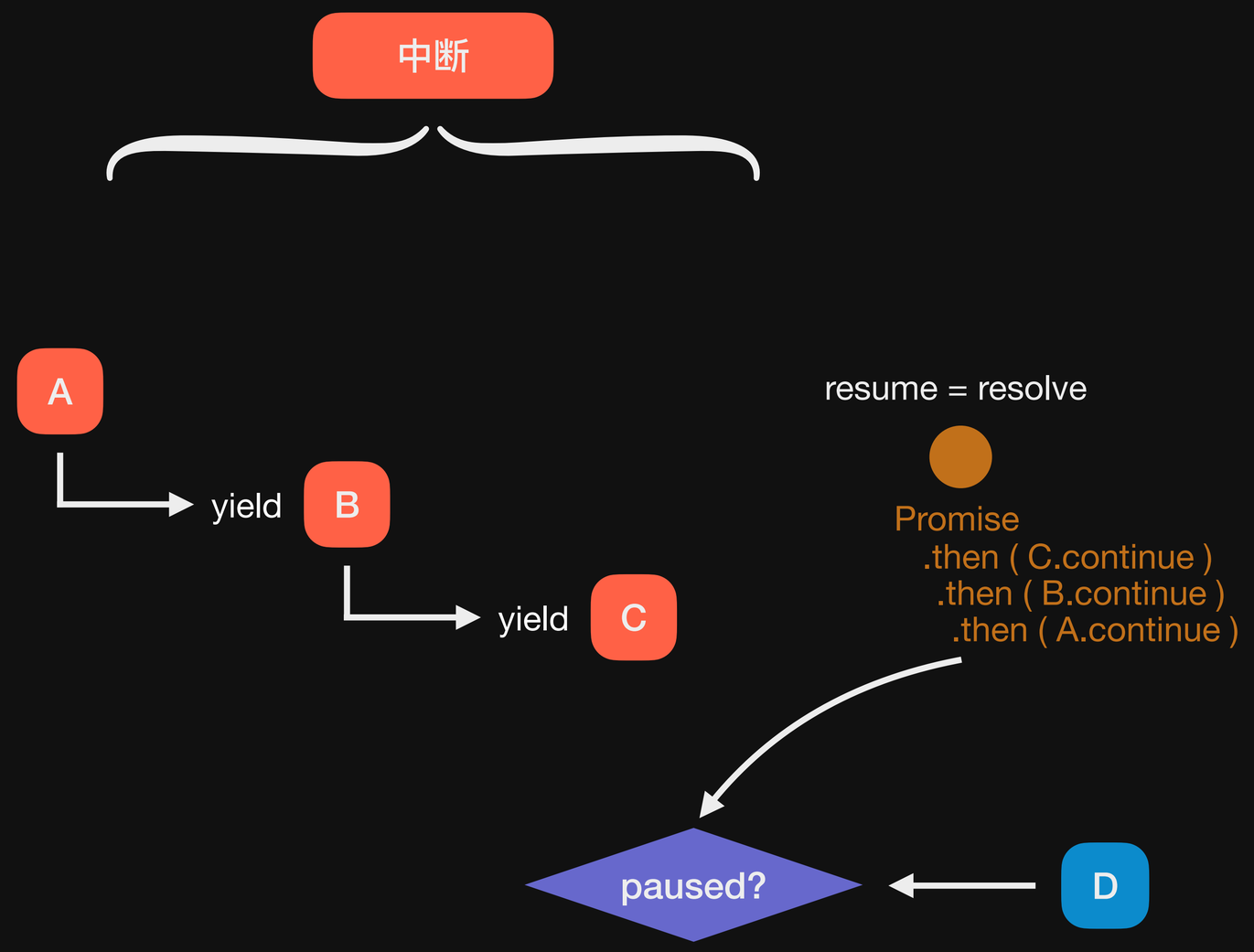


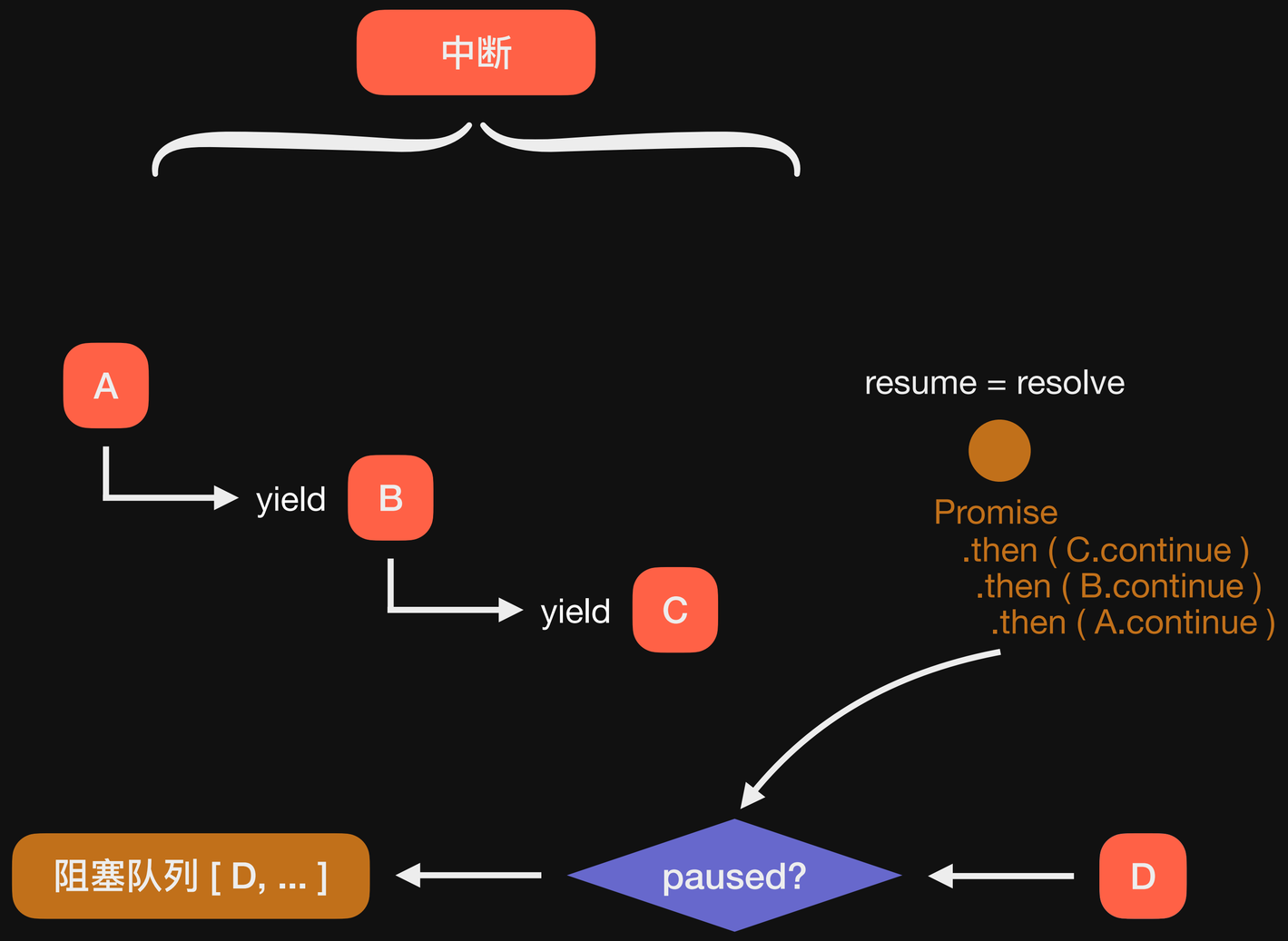
resume = resolve

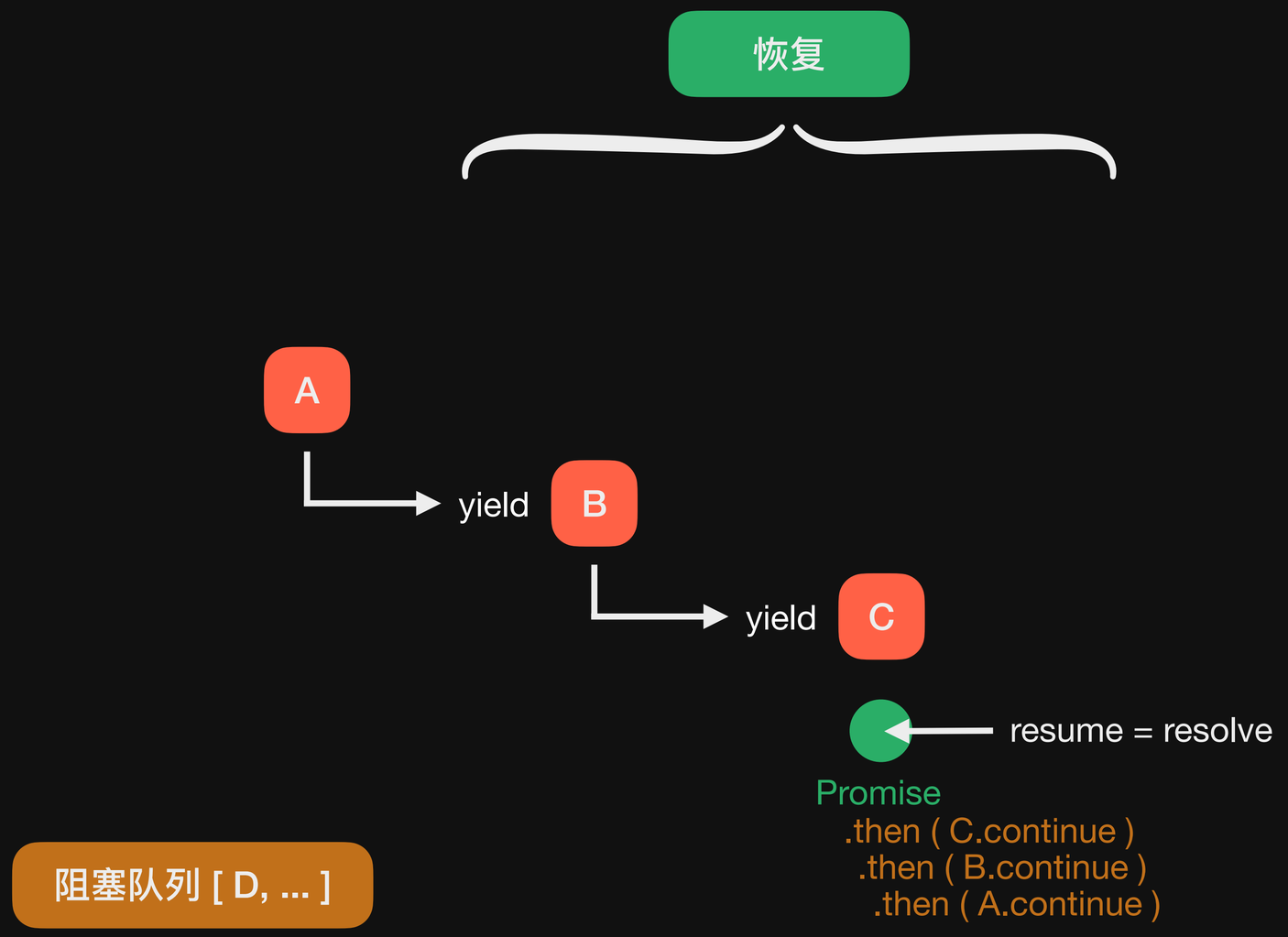


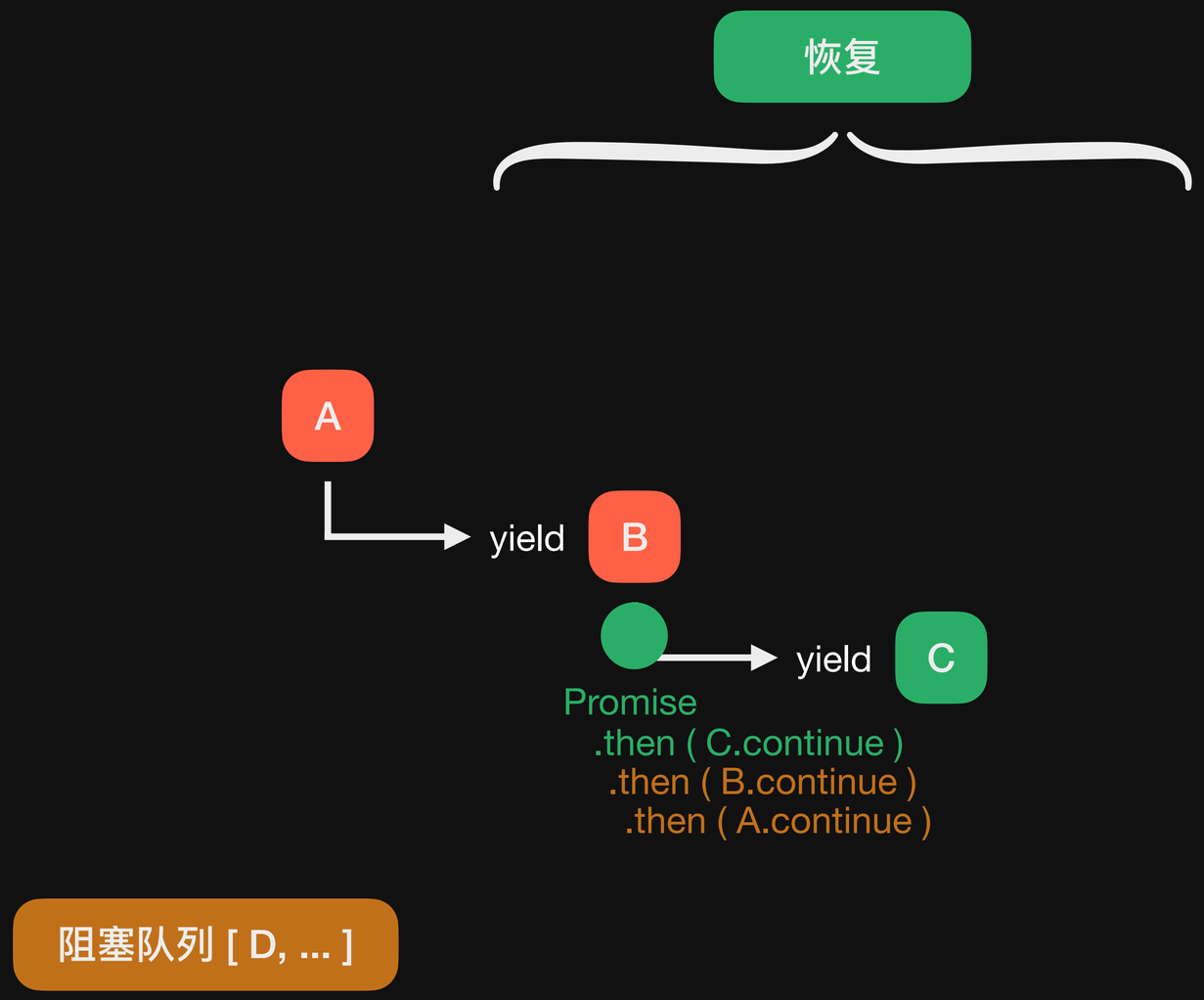
resume = resolve

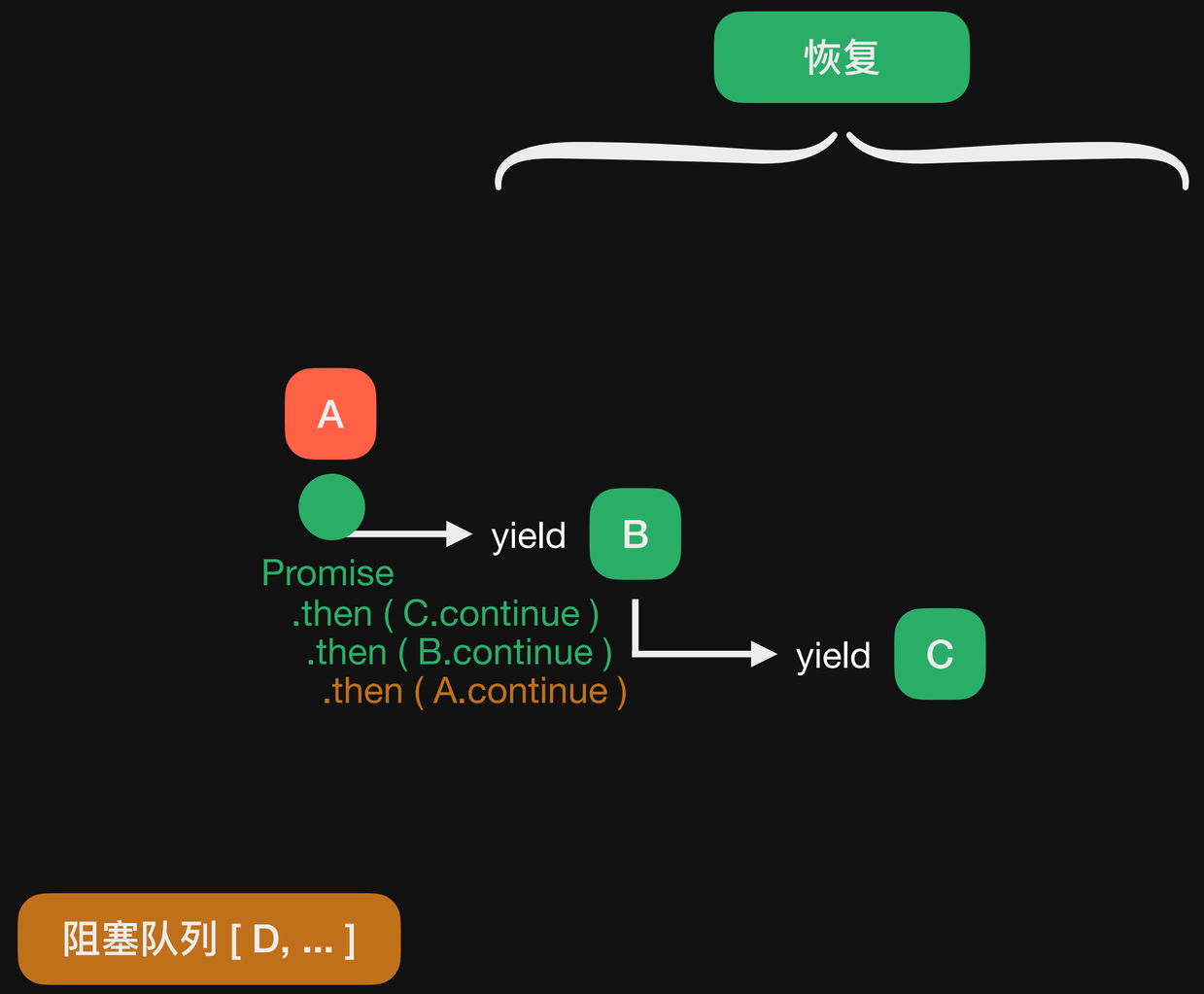

中断
A
yield
B
yield
C
resume = resolve
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
D

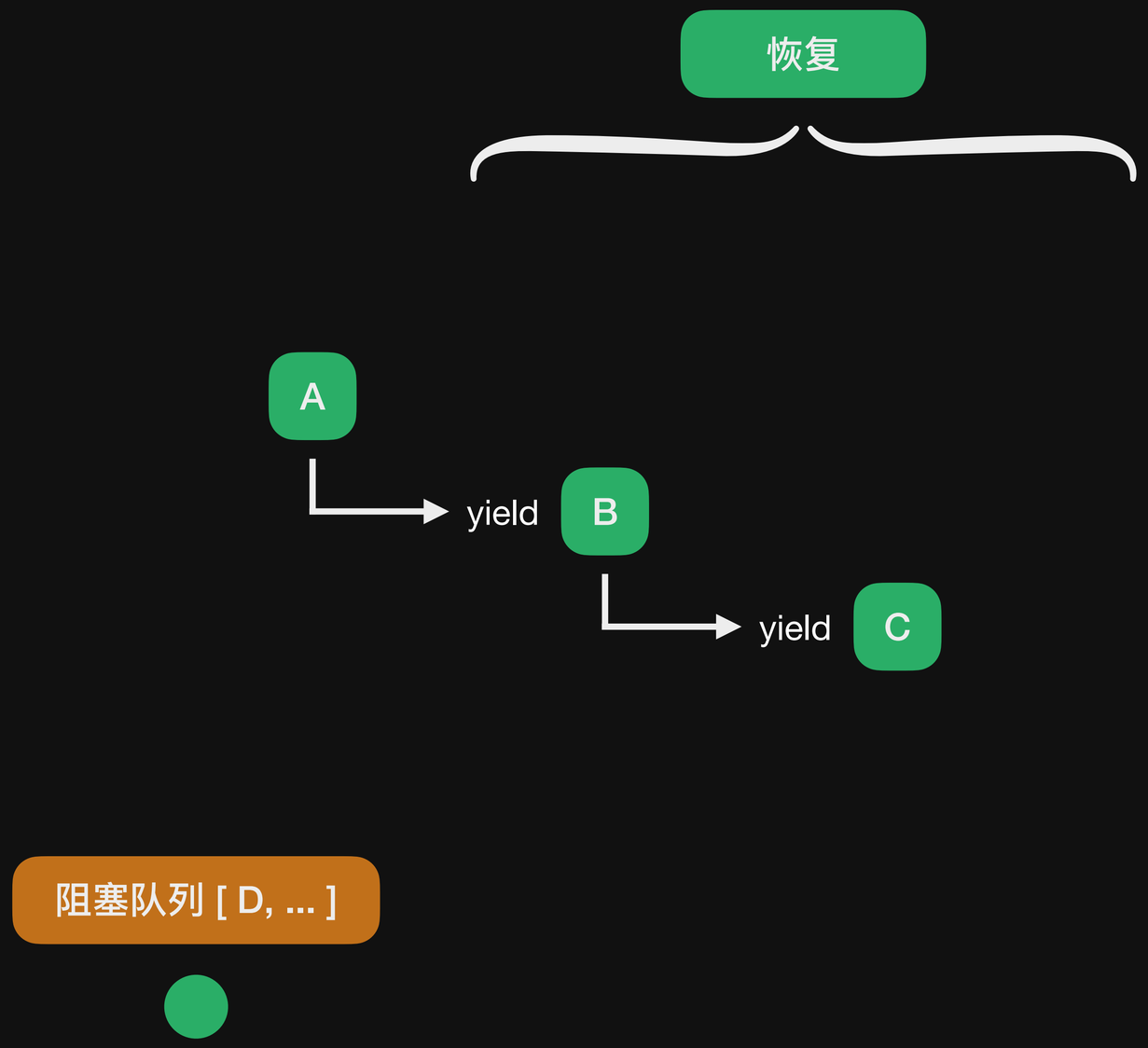

中断
A
yield
B
yield
C
resume = resolve
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
paused?
D

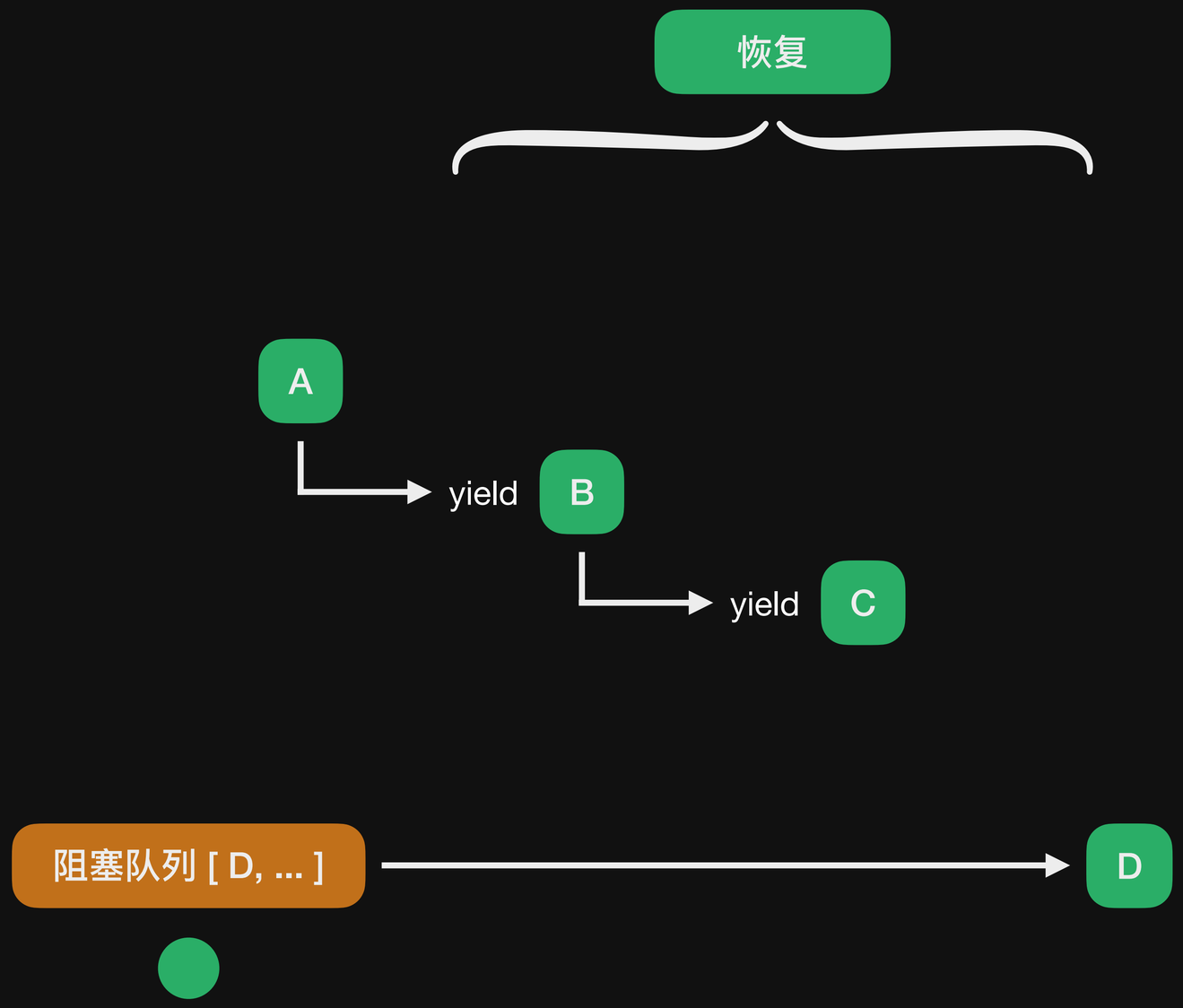

中断
A
yield
B
yield
C
resume = resolve
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
阻塞队列 [D, ...]
paused?
D

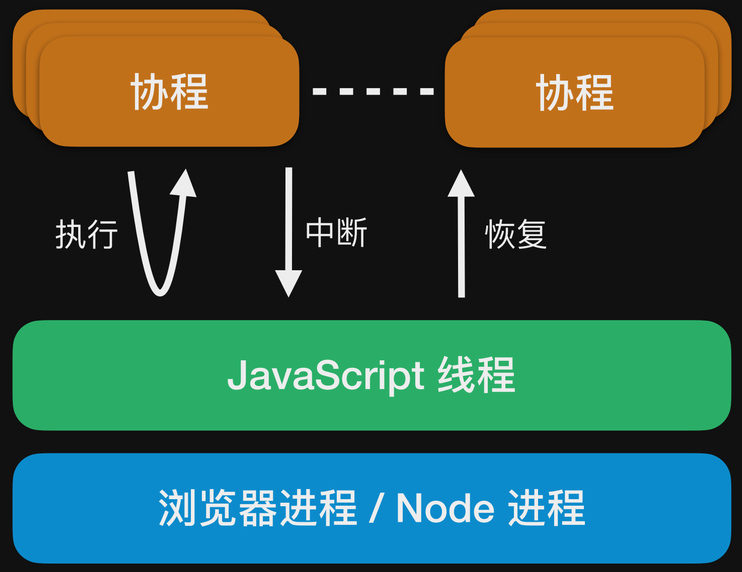

恢复
A
yield
B
yield
C
resume = resolve
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
阻塞队列 [D, ...]


恢复
A
yield
B
yield
C
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
阻塞队列 [D, ...]


恢复
A
yield
B
Promise
.then (C.continue)
.then (B.continue)
.then (A.continue)
yield
C
阻塞队列 [D, ...]


恢复
A
yield
B
yield
C
阻塞队列 [D, ...]


恢复
A
yield
B
yield
C
阻塞队列 [D, ...]
D


协程
协程
执行
中断
恢复
JavaScript 线程
浏览器进程 / Node 进程

getter/setter怎么转换?

模块系统如何转换?

哪些原生方法需要重写？如何重写?

对象创建过程如何中断?

Function怎么转换?

如何保存调用栈和上下文?

如何兼容Generator/Async Generator?

对象取值过程如何中断?

Class怎么转换?

getter/setter怎么转换?

模块系统如何转换?

哪些原生方法需要重写？如何重写?

对象创建过程如何中断?

Function怎么转换?

如何保存调用栈和上下文?

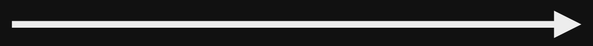
如何兼容Generator/Async Generator?

对象取值过程如何中断?

Class怎么转换?

Function怎么转换?

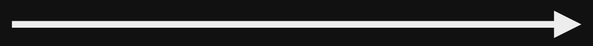
```
function a(b) {
    console.log(b)
}
```



```
function a(b) {
  return $$$_ec_(function* () {
    console.log(b)
  })
}
```

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```



```
function a(b) {
  return $$$_ec_(function* () {
    console.log(b)
  })
}
```

自动迭代，保证能自执行

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f
  return $$$_ec_(function* () {
    yield ($$$_tv_o = console,
    $$$_tv_f = yield $$$_tv_o.log,
    $$$_tv_f.bind($$$_tv_o)(b))
  })
}
```

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f
  return $$$_ec_(function* () {
    yield ($$$_tv_o = console,
    $$$_tv_f = yield $$$_tv_o.log,
    $$$_tv_f.bind($$$_tv_o)(b))
  })
}
```

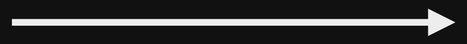
将嵌套调用结果yield出来

将对象属性获取yield出来

保证中断信号能正常传递

Function怎么转换?

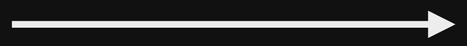
```
function a(b) {
    console.log(b)
}
```



```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f,
      $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    yield (
      $$$_tv_o = console,
      $$$_tv_f = yield $$$_tv_o.log,
      $$$_tv_f.bind($$$_tv_o)(b)
    )
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```



```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f,
      $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    yield (
      $$$_tv_o = console,
      $$$_tv_f = yield $$$_tv_o.log,
      $$$_tv_f.bind($$$_tv_o)(b)
    )
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

兼容通过函数创建对象

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f, $$$_tv_t = this,
      $$$_tv_a = arguments, $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    try {
      $$$_st_(true, x => eval(x), "a")
      yield (
        $$$_tv_o = console,
        $$$_tv_f = yield $$$_tv_o.log,
        $$$_tv_f.bind($$$_tv_o)(b)
      )
    } finally {
      $$$_st_(false)
    }
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f, $$$_tv_t = this,
      $$$_tv_a = arguments, $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    try {
      $$$_st_(true, x => eval(x), "a")
      yield (
        $$$_tv_o = console,
        $$$_tv_f = yield $$$_tv_o.log,
        $$$_tv_f.bind($$$_tv_o)(b)
      )
    } finally {
      $$$_st_(false)
    }
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

记录作用域、上下文，完成转换

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f, $$$_tv_t = this,
      $$$_tv_a = arguments, $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    try {
      $$$_st_(true, x => eval(x), "a")
      yield $$$_br_($$$_di_, 1663663129789, 2, 4)
      yield (
        $$$_tv_o = console,
        $$$_tv_f = yield $$$_tv_o.log,
        $$$_tv_f.bind($$$_tv_o)(b)
      )
    } finally {
      $$$_st_(false)
    }
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f, $$$_tv_t = this,
      $$$_tv_a = arguments, $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    try {
      $$$_st_(true, x => eval(x), "a")
      yield $$$_br_($$$_di_, 1663663129789, 2, 4)
      yield (
        $$$_tv_o = console,
        $$$_tv_f = yield $$$_tv_o.log,
        $$$_tv_f.bind($$$_tv_o)(b)
      )
    } finally {
      $$$_st_(false)
    }
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

行 列

插桩并判断此处是否命中断点

是否需要抛中断信号

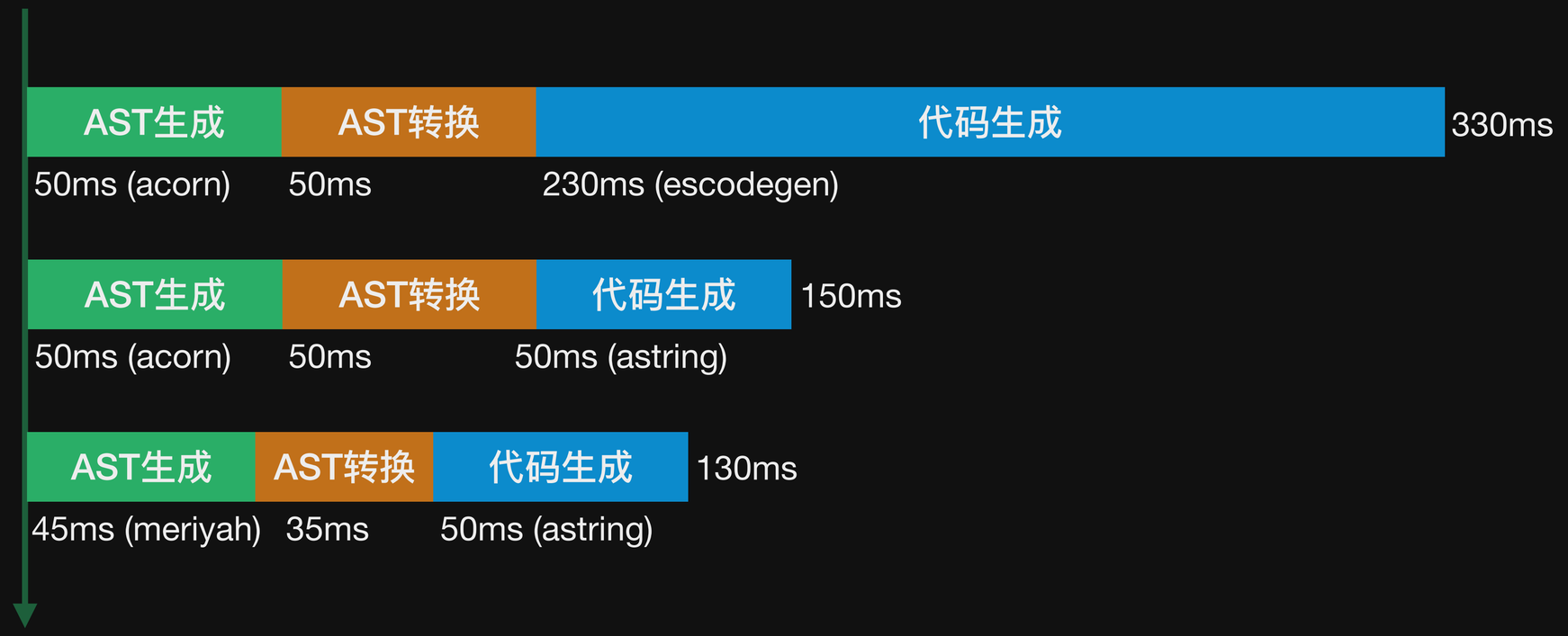
Function怎么转换?

```
function a(b) {
    console.log(b)
}
```

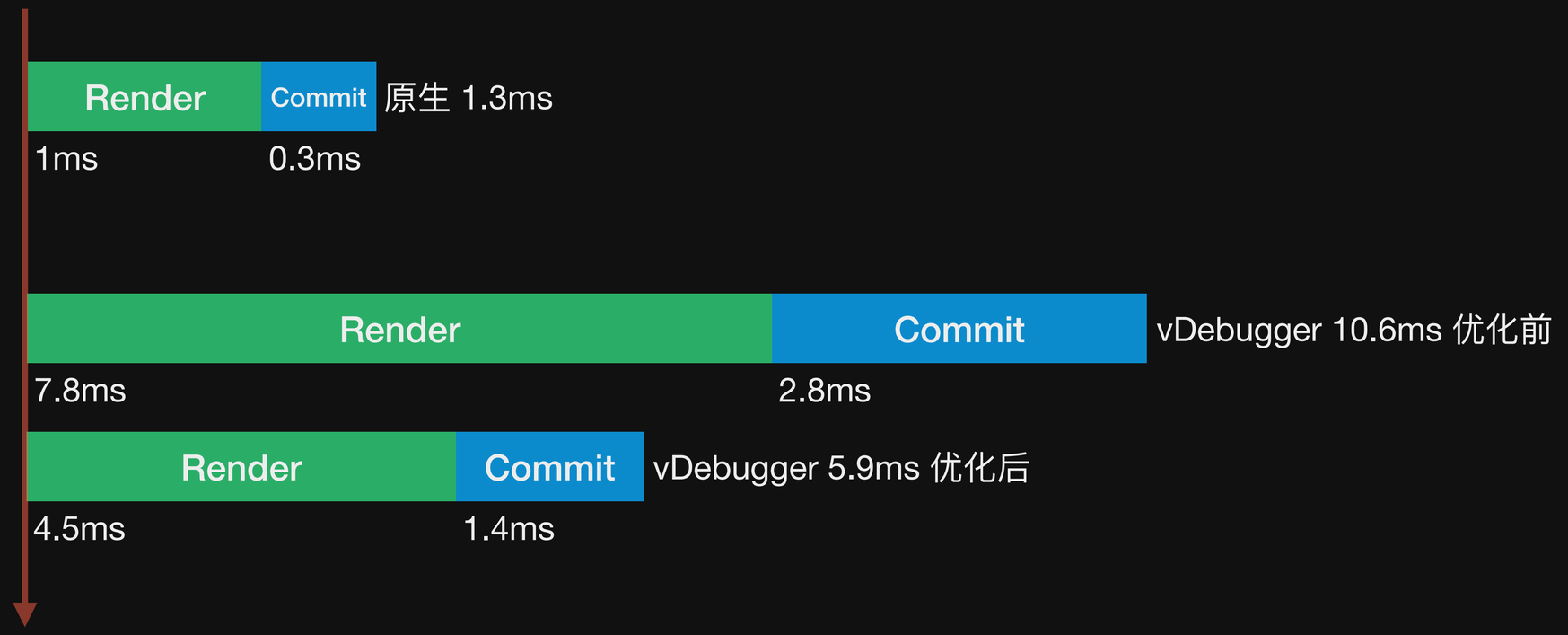
```
function a(b) {
  let $$$_tv_, $$$_tv_o, $$$_tv_f, $$$_tv_t = this,
      $$$_tv_a = arguments, $$$_nt_ = new.target
  const $$$_cr_ = function* () {
    try {
      $$$_st_(true, x => eval(x), "a")
      yield $$$_br_($$$_di_, 1663663129789, 2, 4)
      yield (
        $$$_tv_o = console,
        $$$_tv_f = yield $$$_tv_o.log,
        $$$_tv_f.bind($$$_tv_o)(b)
      )
    } finally {
      $$$_st_(false)
    }
  }
  return $$$_nt_
    ? (this.__proto__.$$$_cr_ = $$$_cr_, this)
    : $$$_ec_($$$_cr_.call(this))
}
```

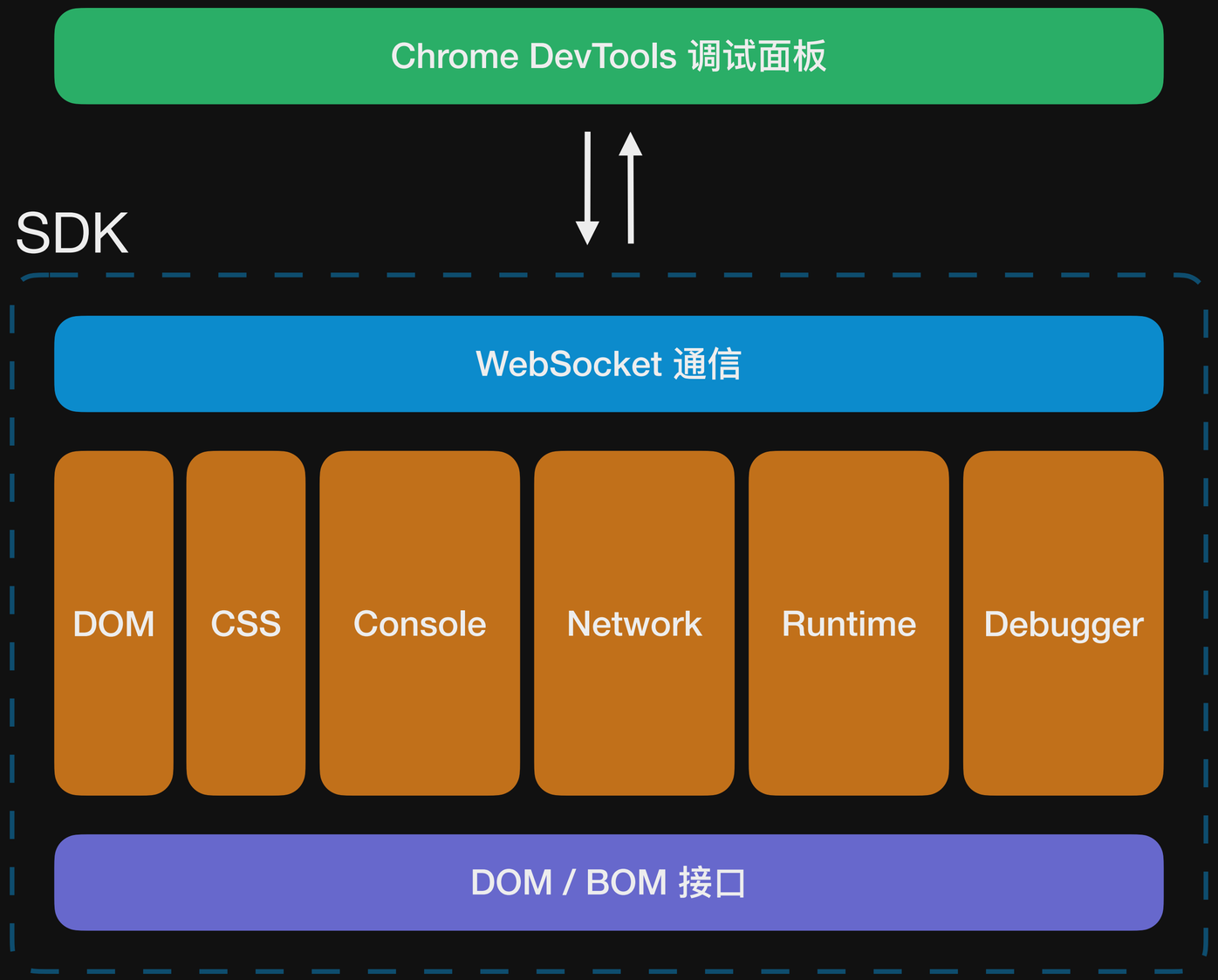
编译性能（ReactDOM为例，3w行代码）

AST规范: ESTree

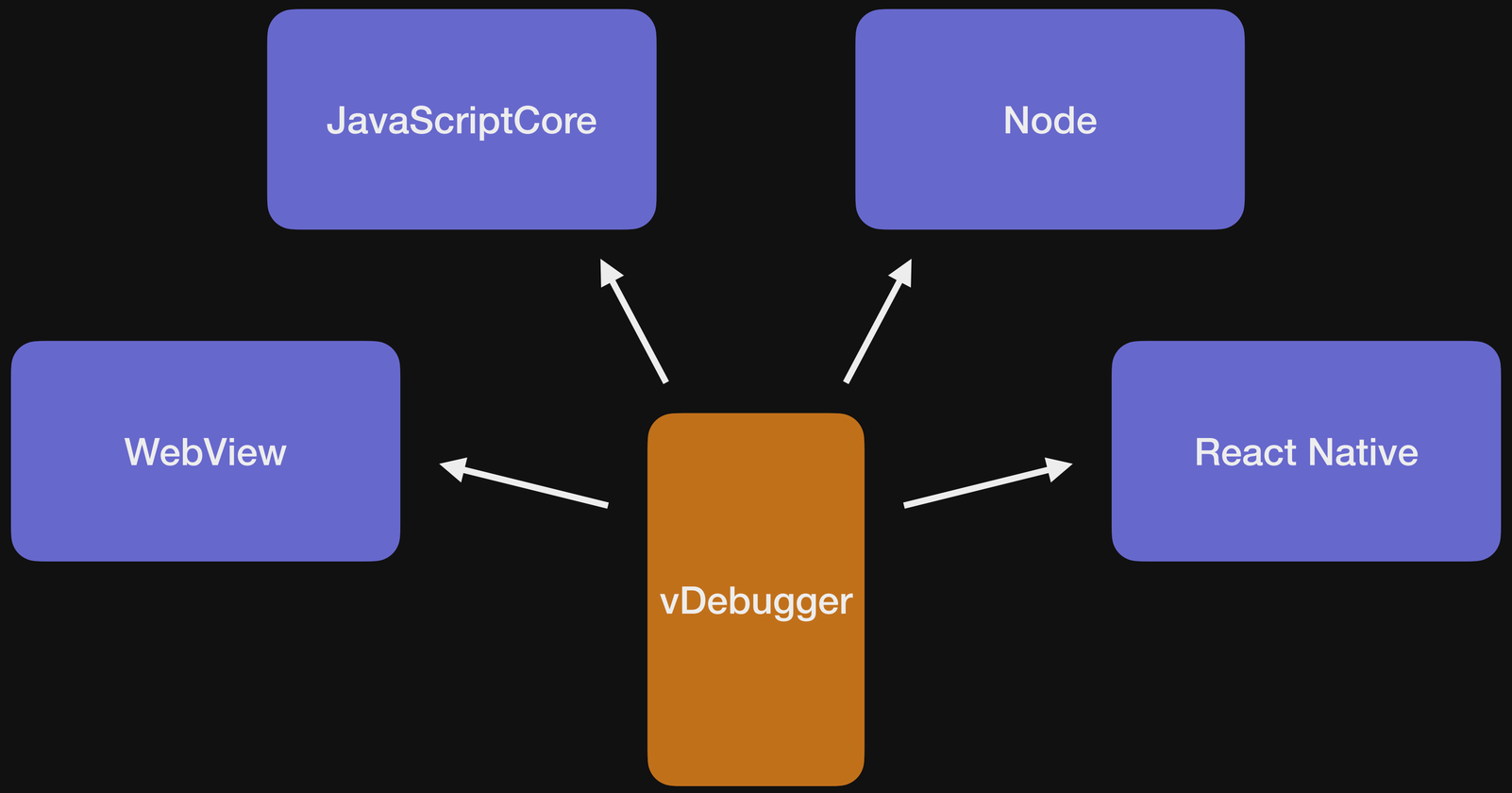


运行性能（React简单应用首次渲染为例）




Chrome DevTools 调试面板
SDK
WebSocket 通信
DOM
CSS
Console
Network
Runtime
Debugger
DOM / BOM 接口

Debugger



https://github.com/wechatjs/vdebugger

微信端内网页远程调试及断点原理

腾讯微信公众号团队　　邱焱坤 王熠弘