执行系统

JavaScript 语言在引擎级别的执行过程

周爱民 / aimingoo
aiming@gmail.com
https://aimingoo.github.io
https://github.com/aimingoo

...X

resolving...

v

```
console.log(...x)
```

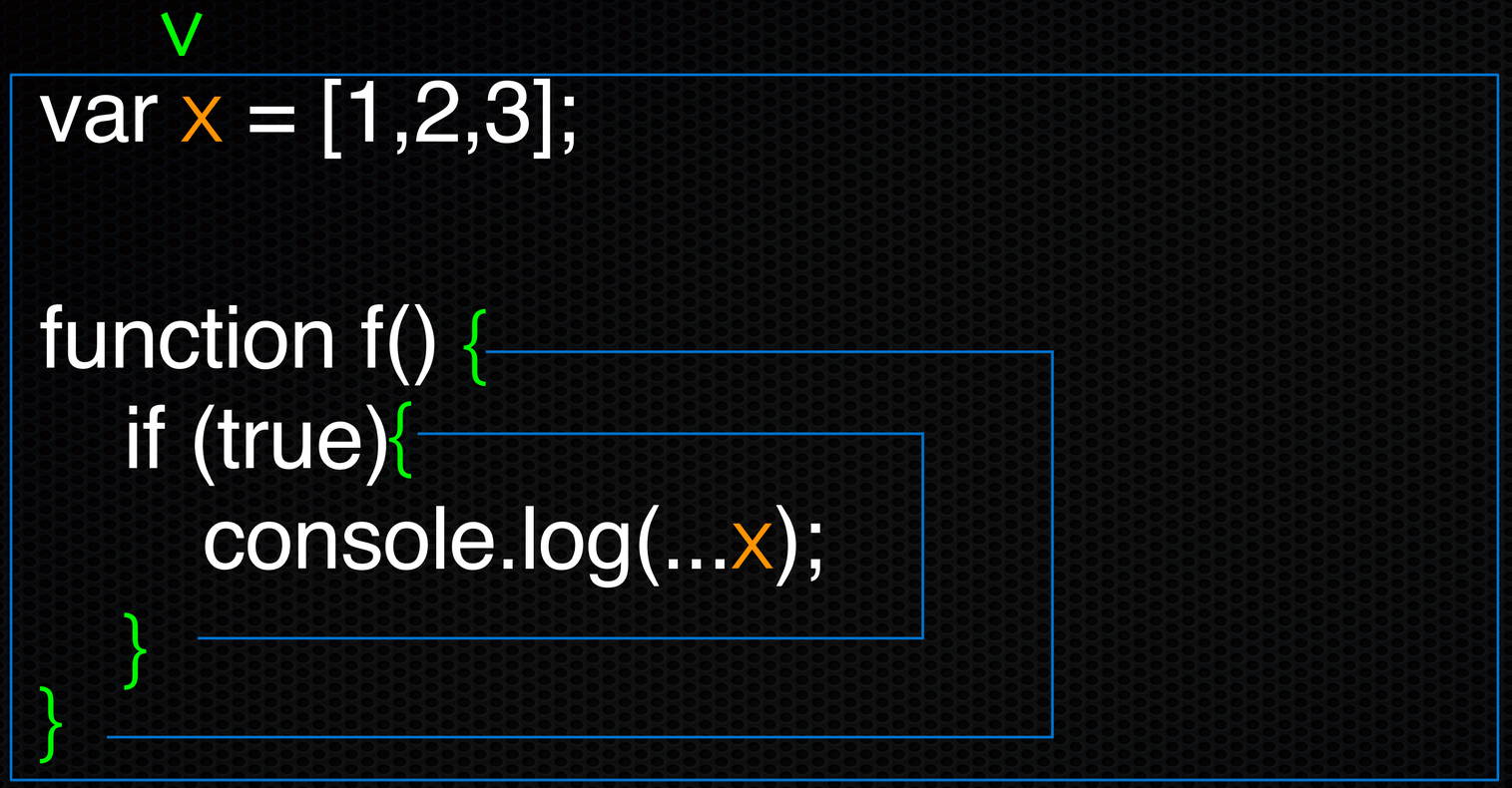
```
if (true)
  console.log(...x);
```

```
                v
if (true){
   console.log(...x);
}
```

```
                  v
function f() {
    if (true){
        console.log(...x);
    }
}
```

```
    v
var x = [1,2,3];

function f() {
    if (true){
        console.log(...x);
    }
}
```



Scope

Environment

now, it's all here...

作用域

Scope

```
object:  varDecls_and_funcDecls
parent:  OuterScope_or_null
```

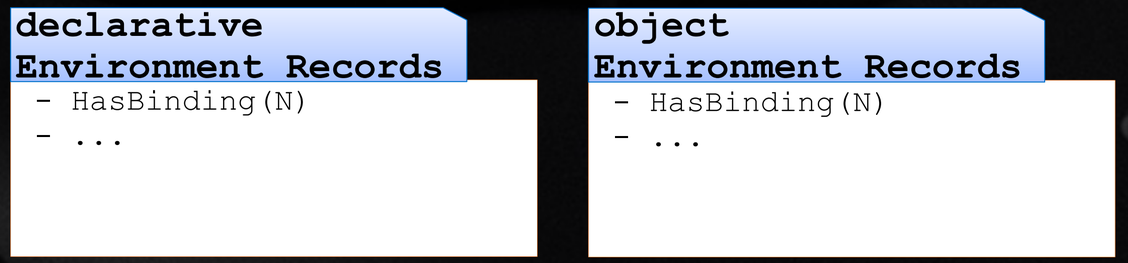
查找名字

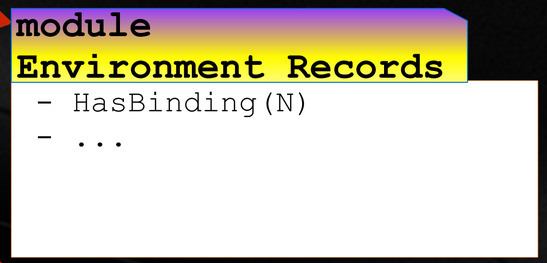
如果没有，查找parent

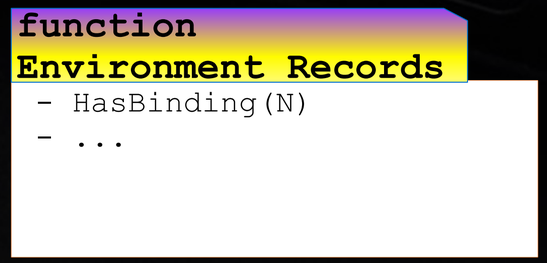
https://github.com/BrendanEich/narcissus

环境记录规范

词法环境规范







（属性描述符和）属性标识符规范

标识符引用

- 8 Executable Code and Execution Contexts
  - 8.1 Lexical Environments
    - 8.1.1 Environment Records
      - 8.1.1.1 Declarative Environment Records
      - 8.1.1.2 Object Environment Records
      - 8.1.1.3 Function Environment Records
      - 8.1.1.4 Global Environment Records
      - 8.1.1.5 Module Environment Records
    - 8.1.2 Lexical Environment Operations
      - 8.1.2.1 GetIdentifierReference (*lex*, *name*, *stric*...
      - 8.1.2.2 NewDeclarativeEnvironment (*E*)
      - 8.1.2.3 NewObjectEnvironment (*O*, *E*)
      - 8.1.2.4 NewFunctionEnvironment (*F*, *newTar*...
      - 8.1.2.5 NewGlobalEnvironment (*G*, *thisValue*)
      - 8.1.2.6 NewModuleEnvironment (*E*)


...x
base: envRec
name: 'x'
strict:
...
with (obj) ...x
base: objectEnvRec
name: 'x'
strict:
...
...obj.x
base: obj
name: 'x'
strict:
...
...super.x
base: super
name: 'x'
thisValue:
strict:
...

...*x*	with (obj) ...*x*	...obj.*x*	...super.*x*
base: *envRec*	base: *objectEnvRec*	base: obj	base: *super*
name: 'x'	name: 'x'	name: 'x'	name: 'x'
strict:	strict:	strict:	thisValue:
...	...	...	strict:
			...

Execution Context

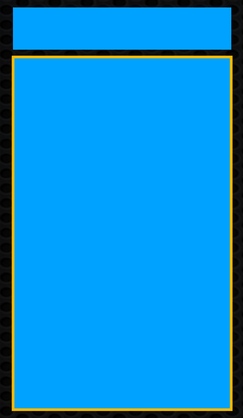
Call or evaluating ...


v
全局环境
var x =
function
函数环境
if
{
词法作用域
console.log(...
}
}

环境记录规范

属性描述符和
属性标识符规范

RunJobs()



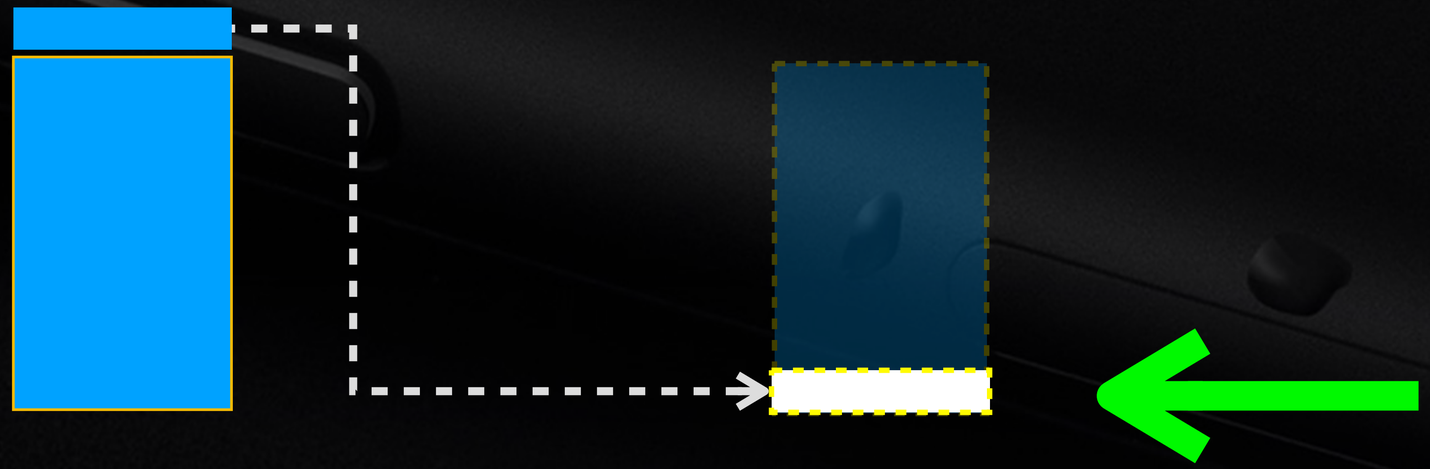
ScriptEvaluationJob

TopLevelModuleEvaluationJob

任务队列

PromiseJobs or ScriptJobs

RunJobs()



任务队列

PromiseJobs,
ScriptJob/TopModuleJob

执行栈

execution context stack

runnning execution context

执行上下文

全局环境

```
var x = [1,2,3];

function f() {
  if (true) {
    console.log(...x);
  }
}
```

(抄写)

ECMAScript-ExecutionContext

```
...
LexicalEnvironment: Global
VariableEnvironment: Global
```

ScriptEvaluation()

```
let scriptContext ...
let scriptBody ...
...
初始化全局(Instantiation)
```

ExecutionContext

```
Realm: from_Initialize_Host
Function: null
```

PendingJob

```
...
job: ScriptEvaluationJob()
callerContext: current...
```

执行栈

1 - *scriptContext*

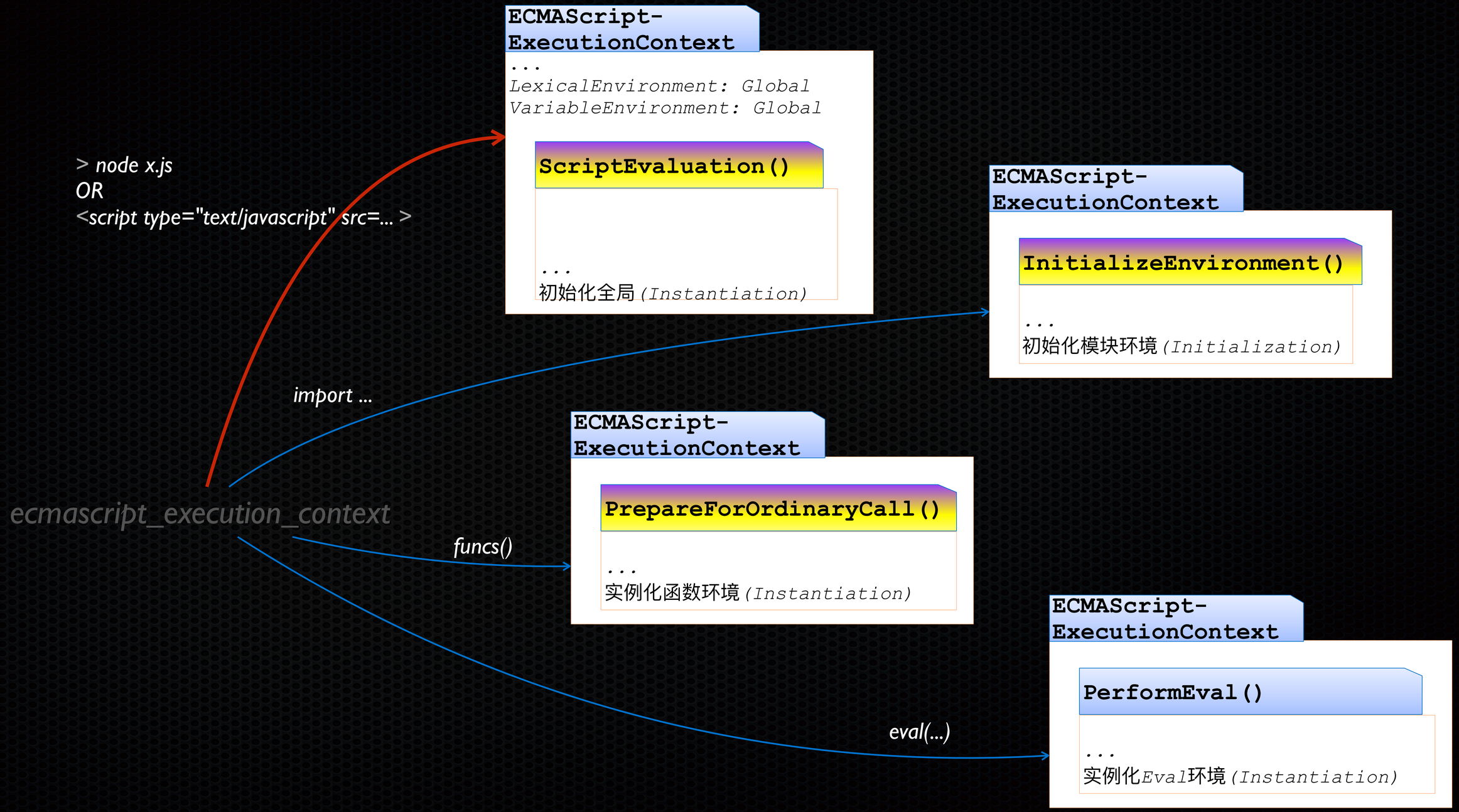
0 - *newContext* for job

-1 - for InitializeHost

runnning execution context

building-func.[[Call]]

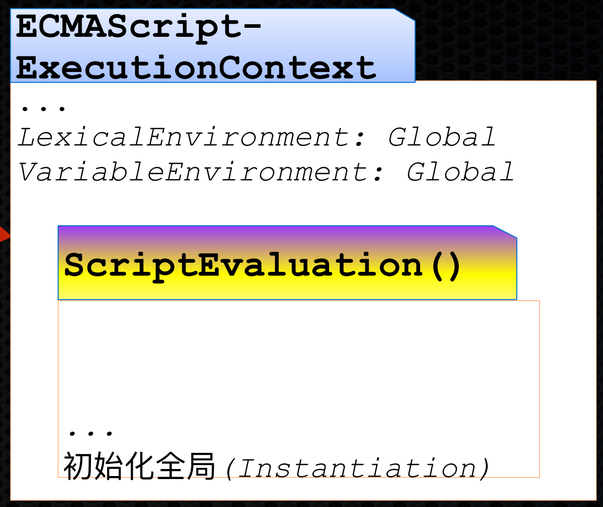
execution_context


ECMAScript-ExecutionContext
...
LexicalEnvironment: Global
VariableEnvironment: Global
ScriptEvaluation()
...
初始化全局 (Instantiation)
> node x.js
OR
<script type="text/javascript" src=... >
ecmascript_execution_context
import ...
ECMAScript-ExecutionContext
InitializeEnvironment()
...
初始化模块环境 (Initialization)
funcs()
ECMAScript-ExecutionContext
PrepareForOrdinaryCall()
...
实例化函数环境 (Instantiation)
eval(...)
ECMAScript-ExecutionContext
PerformEval()
...
实例化Eval环境 (Instantiation)

全局环境

```
var x = [1,2,3];

function f() {
  if (true) {
    console.log(...x);
  }
}
```



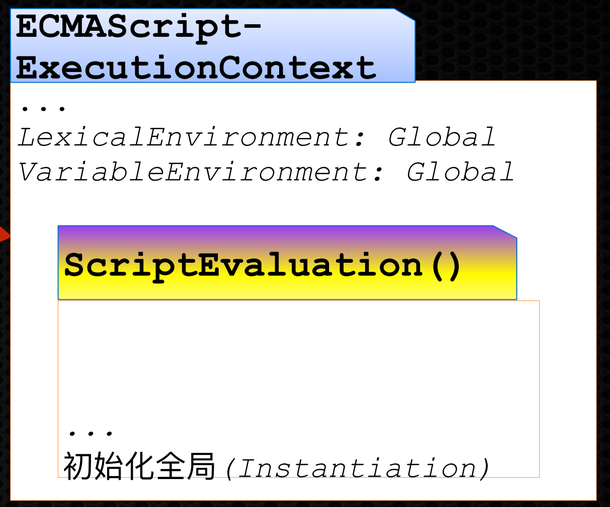
全局环境

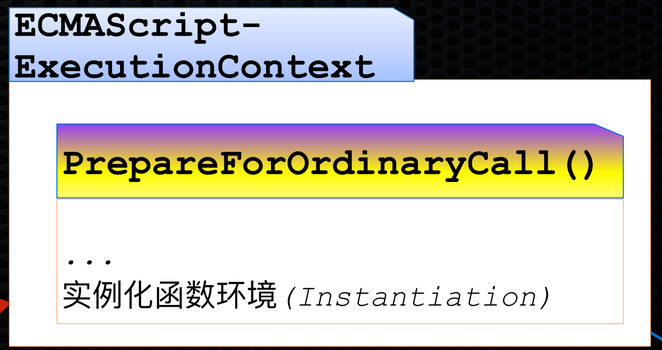
```
var x = [1,2,3];
```

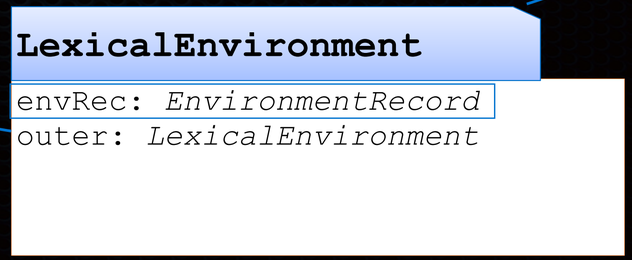
函数环境

```
function f() {
  if (true) {
    console.log(...x);
  }
}
```

```
f();
```



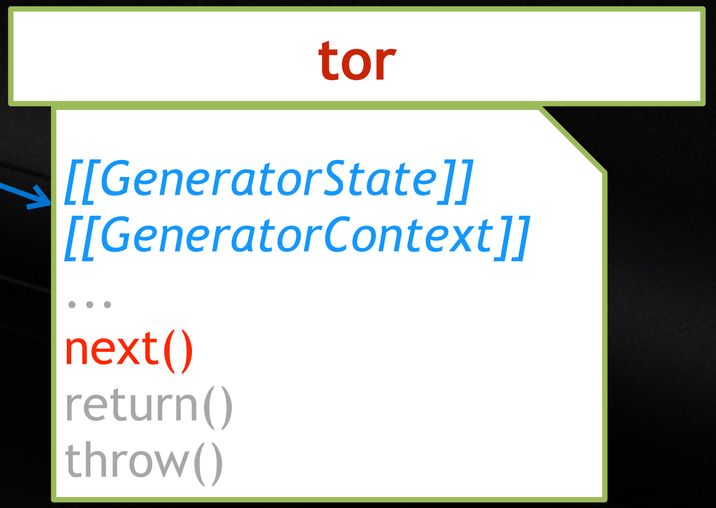




ecmascript_execution_context

```
function* genFunc() {
    ...
    yield ...
}

let tor = genFunc();
result = tor.next();
```

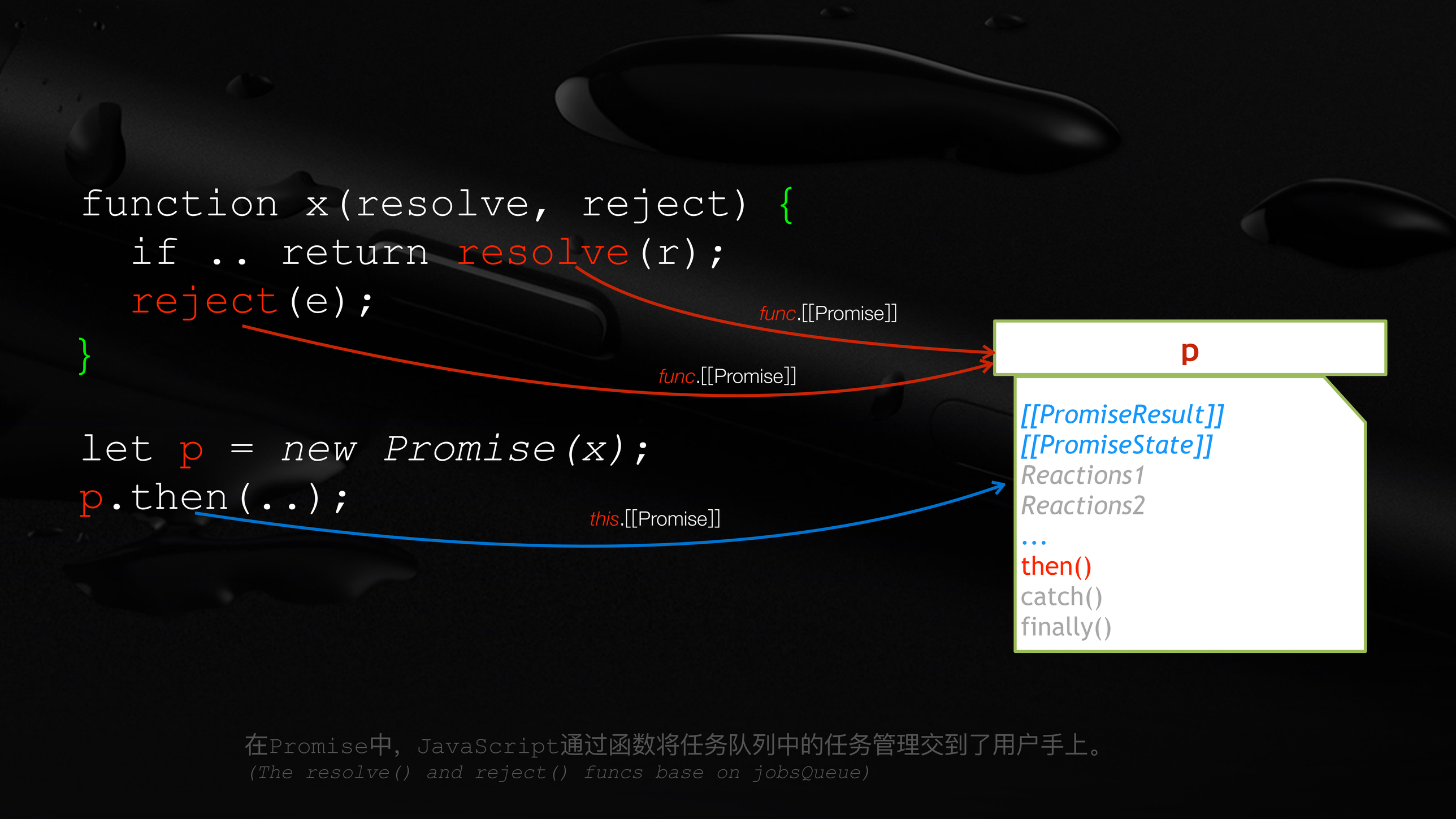


在生成器中，JavaScript通过函数将执行栈中的上下文调度交到了用户手上。

(The tor.next() and yield operator for execute_context)

```
function x(resolve, reject) {
   if .. return resolve(r);
   reject(e);
}

let p = new Promise(x);
p.then(..);
```



在Promise中，JavaScript通过函数将任务队列中的任务管理交到了用户手上。

(The resolve() and reject() funcs base on jobsQueue)

Result

Completion, reference, or values.

```
> obj = { foo() { return this } }

> (obj.foo)() === obj
true

> eval('obj.foo')() === obj
false
```

result of evaluating XXX, when xxx is statement

Completion Specification Type

type
vlaue
target

GetValue(*ref*)

result of evaluating XXX, when xxx is expression

Reference Specification Type

base
name
strict
thisValue

value is Language Types, include:
- Primitive values, or
- Object

or Empty

result is Language Types, include:
- Primitive values, or
- Object

or Reference Specification type.

```
eval('obj.foo')
```

```
(obj.foo)
```

...X

Summary

- 环境的准备
- 逻辑的执行
- 过程的控制
- 结果的返回

...X

Summary

[ ] / ( )

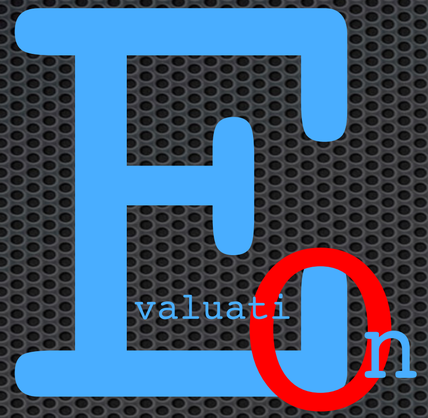
Summary

```
[...x]
(...x)
```

展开语法

... result of *performing ArrayAccumulation for ... // SpreadElement*

... argList be ? ArgumentListEvaluation of ... // a ArgumentList of arguments



github.com/aimingoo