Yangfan Lv3
2019年01月25日 阅读 4908

关注

【译】 Promises/A+ 规范

> - 原文地址：promisesaplus.com/
> - Markdown 地址：github.com/Yangfan2016...
> - 译者：Yangfan2016

一个开放标准，对于开发人员可互操作的 JavaScript promise

一个 promise 代表一个异步操作的最终结果。主要的操作方式是通过调用 promise 的 `then` 方法，它接受的回调函数接受 promise 成功的结果或失败的原因

这个规范详细的描述了 `then` 方法的行为，提供一个互操作基础，所有符合 Promises/A+ 的都可以依赖这个标准实现。因此，该规范已经十分稳定。尽管 Promises/A+ 组织可能会偶尔修改以实现向后兼容，我们也会整合这些大的或不能向后兼容的改变，一起研究，讨论，测试。

最终，Promises/A+ 规范并没处理如何创建 fulfill，或 reject promise，而选择了可互操作的 `then` 方法替代。在今后的工作中可能会考虑。

1. 术语

1.1 ‘promise’ 是一个有符合此标准的 `then` 方法的 `object` 或 `function`

1.2 ‘thenable’ 是 `then` 方法定义的 `object` 或 `function`

1.3 ‘value’ 是一个 JavaScript 合法值（包括 `undefined`，thenable，promise）

1.4 ‘exception’ 是一个 `throw` 语句抛出错误的值

1.5 ‘reason’ 是一个表明 promise 失败的原因的值

2. 要求

2.1 Promise 状态

一个 promise 有且只有一个状态（pending，fulfilled，rejected 其中之一）

2.1.1 pending 状态时：

- 2.1.1.1 可能会转变为 fulfilled 或 rejected 状态

2.1.2 fulfilled 状态时：

- 2.1.2.1 不能再状态为任何其他状态
- 2.1.2.2 必须有一个 value，且不可改变

2.1.3 rejected 状态时：

- 2.1.3.1 不能再状态为任何其他状态
- 2.1.3.2 必须有一个 reason，且不可改变

注：这里 ‘不可改变’ 意思是不可变恒等（同理 `===`），但不意味永远不可变

一个 promise 必须提供一个 `then` 方法，用来获取当前或最终的 value 或 reason

一个 promise 的 `then` 方法接受两个参数：

```
promise.then(onFulfilled, onRejected)
```

2.2.1 onFulfilled 和 onRejected 都是可选参数：

- 2.2.1.1 如果 onFulfilled 不是函数，它会被忽略
- 2.2.1.2 如果 onRejected 不是函数，它会被忽略

2.2.2 如果 onFulfilled 是一个函数：

- 2.2.2.1 它一定是在 promise 是 fulfilled 状态后调用，并且接受一个参数 value
- 2.2.2.2 它一定是在 promise 是 fulfilled 状态后调用
- 2.2.2.3 它最多被调用一次

2.2.3 如果 onRejected 是一个函数：

- 2.2.3.1 它一定在 promise 是 rejected 状态后调用，并且接受一个参数 reason
- 2.2.3.2 它一定在 promise 是 rejected 状态后调用
- 2.2.3.3 它最多被调用一次

2.2.4 onFulfilled 或 onRejected 只在执行环境堆栈只包含平台代码之后调用 [3.1]

2.2.5 onFulfilled 和 onRejected 会作为函数形式调用 (也就是说，默认 `this` 指向 global，严格模式 `undefined`) [3.2]

2.2.6 promise 的 `then` 可以链式调用多次

- 2.2.6.1 如果或当 promise 转态是 fulfilled 时，所有的 onFulfilled 回调回以他们注册时的顺序依次执行
- 2.2.6.2 如果或当 promise 转态是 rejected 时，所有的 onRejected 回调回以他们注册时的顺序依次执行

`promise2 = promise1.then(onFulfilled, onRejected);`

- 2.2.7.1 如果 onFulfilled 或 onRejected 返回的是一个 x，那么它会以

`[[Resolve]](promise2, x)` 处理解析

- 2.2.7.2 如果 onFulfilled 或 onRejected 里抛出了一个异常，那么 promise2 必须捕获这个错误（接受一个 reason 参数）
- 2.2.7.3 如果 onFulfilled 不是一个函数，并且 promise1 状态是 fulfilled，那么 promise2 一定会接受到与 promse1 一样的值 value
- 2.2.7.4 如果 onRejected 不是一个函数，并且 promise1 状态是 rejected，promise2 一定会接受到与 promise1 一样的值 reason

2.3 Promise 处理程序

promise 处理程序是一个表现形式为 [[Resolve]](promise, x) 的抽象处理操作。如果 x 是 thenable 类型，它会尝试生成一个 promise 处理 x，否则它将直接 resolve x

只要 then 方法符合 Promises/A+ 规则，那么对 thenables 处理就允许实现可互操作（链式调用，层层传递下去）。它也允许对那些不符合 Promises/A+ 的 then 方法进行 “吸收”

`[[Resolve]](promise, x)` 的执行表现形式如下步骤：

2.3.1 如果返回的 promise1 和 x 是指向同一个引用（循环引用），则抛出错误

2.3.2 如果 x 是一个 promise 实例，则采用它的状态：

- 2.3.2.1 如果 x 是 pending 状态，那么保留它（递归执行这个 promise 处理程序），直到 pending 状态转为 fulfilled 或 rejected 状态
- 2.3.2.2 如果或当 x 状态是 fulfilled，resolve 它，并且传入和 promise1 一样的值 value
- 2.3.2.3 如果或当 x 状态是 rejected，reject 它，并且传入和 promise1 一样的值 reason

2.3.3 此外，如果 x 是个对象或函数类型

- 2.3.3.1 把 `x.then` 赋值给 `then` 变量

- 2.3.3.3 如果 `then` 是函数类型，那个用 x 调用它（将 `then` 的 `this` 指向 x）,第一个参数传 resolvePromise ，第二个参数传 rejectPromise：
    - 2.3.3.3.1 如果或当 resolvePromise 被调用并接受一个参数 y 时，执行 `[[Resolve]](promise, y)`
    - 2.3.3.3.2 如果或当 rejectPromise 被调用并接受一个参数 r 时，执行 reject(r)
    - 2.3.3.3.3 如果 resolvePromise 和 rejectPromise 已经被调用或以相同的参数多次调用的话吗，优先第一次的调用，并且之后的调用全部被忽略（避免多次调用）
    - 2.3.3.4 如果 `then` 执行过程中抛出了异常，
        - 2.3.3.3.4.1 如果 resolvePromise 或 rejectPromise 已经被调用，那么忽略异常
        - 2.3.3.3.4.2 否则，则 reject 这个异常
- 2.3.3.4 如果 `then` 不是函数类型，直接 resolve x（resolve(x)）

2.3.4 如果 x 即不是函数类型也不是对象类型，直接 resolve x（resolve(x)）

如果被 resolve 的 promise 参与了 thenable 的循环链中，那么可能会导致无限递归。我们鼓励实现检测这种无限递归的方法并且返回一个错误信息，但并不是必须的 [3.6]

3. 备注

3.1 这里的 “平台代码”是指引擎，环境，和 promise 实现代码。实际上，这个要求确保 onFulfilled 和 onRejected 都在下一轮的事件循环中（一个新的栈）被异步调用。可以用宏任务，例如：`setTimeout` ， `setImmediate` 或者微任务，例如： `MutationObsever` 或 `process.nextTick` 实现。由于 promise 的实现被当做平台代码，所以它本身可能包含一个任务队列或 “trampoline” 的处理程序

3.2 这个 `this` 在严格模式下是 `undefined` ，在宽松模式，指向 global 对象

3.3 具体的实现可以允许 promise2 和 promise1 绝对相等，要满足所有要求。每一个处理 promise2 和 promise1 绝对相等的实现都要写上文档标注

3.4 通常，只有它来自当前实现才可以判断 x 是一个真正的 promise。 此条款允许采取已知符合 promise 标准实现的状态

3.6 实现不应该武断地限制 thenable 链的深度，假设超出限制的无限递归。只有真正的循环引用才会导致一个 `TypeError` 错误，如果遇到一个不同的无限递归 thenable 链，一直递归永远是正确的行为

> - 本文仅代表原作者个人观点，译者不发表任何观点
> - Markdown 文件由译者手动整理，如有勘误，欢迎指正
> - 译文和原文采用一样协议，侵删

文章分类 阅读 文章标签 Promise

Yangfan Lv3
前端小学生 @ 达佳互联
获得点赞 1,010 · 获得阅读 85,598

关注

输入评论...

Cryptonym Lv2 0

2.3.2.2 如果或当 x 状态是 fulfilled，resolve 它，并且传入和 promise1 一样的值 value 这一段话完全不理解 可以解释一下吗

8月前 回复

Marszht Lv1

应该是promise2 一样的value , 类似下面
let p2 = p1.then(res => {
let p4 = new Promise((resolve, reject) => {
resolve(4);
})...

展开

3月前

千年虫25

promise值穿透总算搞明白了，感谢

10月前 回复