Js中的Event Loop、Tasks和Micro-Tasks

爽

先看一道题目


```
setTimeout(function() {
        console.log(1)
}, 0);

new Promise(function executor(resolve) {
        console.log(2);
        for( var i=0 ; i<10000 ; i++ ) {
                i == 9999 && resolve();
        }
        console.log(3);
}).then(function() {
        console.log(4);
});

console.log(5);
```

输出顺序是什么？

答案


```
setTimeout(function() {
        console.log(1)                  //第五个输出
}, 0);

new Promise(function executor(resolve) {
        console.log(2);                 //第一个输出
        for( var i=0 ; i<10000 ; i++ ) {
                i == 9999 && resolve();
        }
        console.log(3);                 //第二个输出
}).then(function() {
        console.log(4);                 //第四个输出
});

console.log(5);                         //第三个输出
```

答案是2，3，5，4，1。为什么？

Event loop

简单来说，事件循环中有两种队列，tasks（macro-task）和micro- task；
这两个队列分两步执行，第一步先执行一个（仅一个）task任务，第二步会执行整个micro-task队列中的所有任务。
执行时，有下面的规定：

1、一个事件循环可以有多个tasks，按顺序执行
2、一个事件循环只有一个micro-task，在当前的task执行完毕后执行。并且，执行micro-task队列任务的时候，也允许加入新的micro-task任务，添加到micro-task队列结尾。所有Microtask任务全部执行完毕，才结束循环

macro-tasks一般包括: setTimeout，setInterval，setImmediate, I/O，UI rendering；
micro-tasks一般包括: process.nextTick，Promises，MutationObserver

解析


```
setTimeout(function() {
    console.log(1)
}, 0);

new Promise(function executor(resolve) {
    console.log(2);
    for( var i=0 ; i<10000 ; i++ ) {
        i == 9999 && resolve();
    }
    console.log(3);
}).then(function() {
    console.log(4);
});

console.log(5);
```

现在，再根据事件机制来解释这道题目

1、执行当前的js script（也就是一个task），按顺序执行

2、setTimeout的回调生成了一个新的task，添加到task队列中，待执行

3、实例化promise，里面的函数是直接执行的，所以先输出2；promise resolved；再输出 3;

4、promise.then，回调被添加到micro-tasks 队列中；

5、输出5

解析


```
setTimeout(function() {
    console.log(1)
}, 0);

new Promise(function executor(resolve) {
    console.log(2);
    for( var i=0 ; i<10000 ; i++ ) {
        i == 9999 && resolve();
    }
    console.log(3);
}).then(function() {
    console.log(4);
});

console.log(5);
```

/* 续.... */

6、当前task已经执行完毕了，然后micro-tasks队列里面的任务，即promise的回调，输出4

7、micro-tasks也执行完毕，执行下一个task，即setTimeout回调，输出1

所以，最终的输出顺序是2，3，5，4，1

THANKS

FOR YOUR WATCHING

02

OPEN ORIENTED

凹 凸 实 验 室