链滴

浏览器 - 异步运行一个不阻塞 UI 的函数 (advanced)

作者：Vanessa

原文链接：https://ld246.com/article/1575540884547

来源网站：链滴

2019-12-05

描述

使用 Web Worker 在单独的线程中运行一个函数，在不阻塞 UI 的情况在允许长时间运行该函数。

提示

- 使用 Blob 对象 URL 创建一个 Worker，内容应为所提供函数的字符串版本
- 在回调函数执行完毕后，需马上将其返回值进行传递
- 返回一个 Promise，用来监听 worker 的 onmessage 和 onerror 事件。将 worker 中返回的数做为 Promise 成功的参数，将错误信息做为 Promise 失败的参数

代码

```
const runAsync = fn => {
  const worker = new Worker(
    URL.createObjectURL(new Blob([`postMessage((${fn})());`]), {
      type: 'application/javascript; charset=utf-8'
    })
  );
  return new Promise((res, rej) => {
    worker.onmessage = ({ data }) => {
      res(data), worker.terminate();
    };
    worker.onerror = err => {
      rej(err), worker.terminate();
    };
  });
};
```

示例

在单独的线程中运行函数：

```
const longRunningFunction = () => {
  let result = 0;
  for (let i = 0; i < 1000; i++)
    for (let j = 0; j < 700; j++) for (let k = 0; k < 300; k++) result = result + i + j + k;

  return result;
};
/*
  注意：由于函数运行在不同的上下文中，因此不支持闭包。
  提供给 `runAsync` 的函数将被转换为字符串，因此一切都变成来字符。
  所有函数和变量都必须定义在内部。
*/
runAsync(longRunningFunction).then(console.log); // 209685000000
runAsync(() => 10 ** 3).then(console.log); // 1000
let outsideVariable = 50;
```

```
runAsync(() => typeof outsideVariable).then(console.log); // 'undefined'
```

返回总目录

每天 30 秒系列之 JavaScript 代码