

20	React 和 Vue 两大框架之间的相爱相杀	学习时长: 11分20秒
21	Vue 常考基础知识	已学完 学习时长: 13分28秒
22	Vue 常考进阶知识点	已学完 学习时长: 26分37秒
23	React 常考基础知识	学习时长: 17分7秒
24	React 常考进阶知识点	学习时长: 5分25秒
25	监控	已学完 学习时长: 2分48秒
26	UDP	已学完 学习时长: 3分11秒
27	TCP	已学完 学习时长: 10分1秒
28	HTTP 及 TLS	已学完 学习时长: 10分38秒
29	HTTP/2 及 HTTP/3	

监控

前端监控一般分为三种，分别为页面埋点、性能监控以及异常监控。

这一章节我们将来学习这些监控相关的内容，但是基本不会涉及到代码，只是让大家了解下前端监控该用什么方式实现。毕竟大部分公司都只是使用到了第三方的监控工具，而不是选择自己造轮子。

页面埋点

页面埋点应该是大家最常写的监控了，一般起码会监控以下几个数据：

- PV / UV
- 停留时长
- 流量来源
- 用户交互

对于这几类统计，一般的实现思路大致可以分为两种，分别为手写埋点和无埋点的方式。

相信第一种方式也是大家最常用的方式，可以自主选择需要监控的数据然后在相应的地方写入代码。这种方式的灵活性很大，但是唯一的缺点就是工作量较大，每个需要监控的地方都得插入代码。

另一种无埋点的方式基本不需要开发者手写埋点了，而是统计所有的事件并且定时上报。这种方式虽然没有前一种方式繁琐了，但是因为统计的是所有事件，所以还需要后期过滤出需要的数据。

性能监控

性能监控可以很好的帮助开发者了解在各种真实环境下，页面的性能情况是如何的。

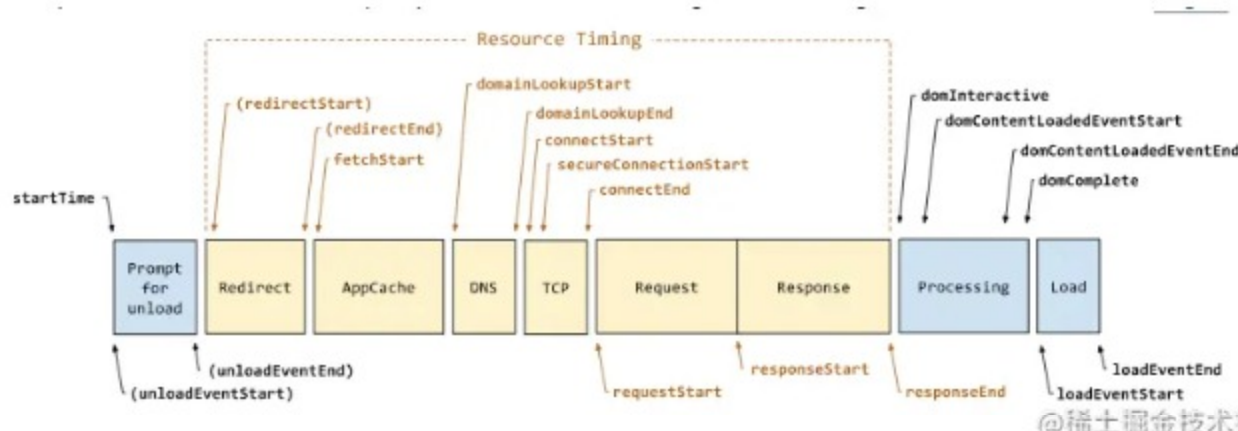
对于性能监控来说，我们可以直接使用浏览器自带的 [Performance API](#) 来实现这个功能。

对于性能监控来说，其实我们只需要调用 `performance.getEntriesByType('navigation')` 这行代码就行了。对，你没看错，一行代码我们就可以获得页面中各种详细的性能相关信息。

```
> performance.getEntriesByType('navigation')
< [PerformanceNavigationTiming] (1)
  0: PerformanceNavigationTiming
    connectEnd: 14.159999991534278
    connectStart: 14.159999991534278
    decodedBodySize: 17583
    domComplete: 3008.5649999964517
    domContentLoadedEventEnd: 1379.350000002887
    domContentLoadedEventStart: 1835.88499999982384
    domInteractive: 1835.8499999856576
    domainLookupEnd: 14.159999991534278
    domainLookupStart: 14.159999991534278
    duration: 3010.339999788962
    encodedBodySize: 11548
    entryType: "navigation"
    fetchStart: 14.159999991534278
    initiatorType: "navigation"
    loadEventEnd: 3010.339999788962
    loadEventStart: 3008.5850000032224
    name: "https://juejin.im/"
    nextHopProtocol: "h2"
    redirectCount: 0
    redirectEnd: 0
    redirectStart: 0
    requestStart: 25.439999997615814
    responseEnd: 102.87499998230487
    responseStart: 54.6449999652617
    secureConnectionStart: 0
    serverTiming: {}
    startTime: 0
    transferSize: 0
    type: "navigate"
    unloadEventEnd: 0
    unloadEventStart: 0
    workerStart: 0
```

@稀土掘金技术社区

我们可以发现这行代码返回了一个数组，内部包含了相当多的信息，从数据开始在网络中传输到页面加载完成都提供了相应的数据。



异常监控

对于异常监控来说，以下两种监控是必不可少的，分别是代码报错以及接口异常上报。

对于代码运行错误，通常的办法是使用 `window.onerror` 拦截报错。该方法能拦截到大部分的详细报错信息，但是也有例外

- 对于跨域的代码运行错误会显示 `Script error`。对于这种情况我们需要给 `script` 标签添加 `crossorigin` 属性
- 对于某些浏览器可能不会显示调用栈信息，这种情况可以通过 `arguments.callee.caller` 来做栈递归

对于异步代码来说，可以使用 `catch` 的方式捕获错误。比如 `Promise` 可以直接使用 `catch` 函数，`async await` 可以使用 `try catch`。

但是要注意线上运行的代码都是压缩过的，需要在打包时生成 sourceMap 文件便于 debug。

对于捕获的错误需要上传给服务器，通常可以通过 `img` 标签的 `src` 发起一个请求。

另外接口异常就相对来说简单了，可以列举出出错的状态码。一旦出现此类状态码就可以立即上报出错。接口异常上报可以让开发人员迅速知道有哪些接口出现了大面积的报错，以便迅速修复问题。

小结

这一章节内容虽然不多，但是这类监控的知识网上的资料确实不多，相信能给大家一个不错的思路。

留言



输入评论（Enter换行，`⌘` + Enter发送）

发表评论

全部评论 (23)

- 

getify

前端 @ 前端。 . . | 1月前

确实讲的有点少，像大纲

👍 点赞

🗨️ 回复
- 

用户1800602079...

前端开发 @ 阿里 | 3月前

退钱

👍 1

🗨️ 回复
- 

内小子

前端 | 1年前

setTimeout里面抛出异常怎么捕获

👍 点赞

🗨️ 1
- 

面向GPT4CV

1年前



👍 点赞

🗨️ 回复
- 

breeze

2年前

。 . . 挺敷衍的

👍 6

🗨️ 1
- 

想当敷衍，不是挺

👍 2

🗨️ 回复
- 

石头就是我54394

2年前

没有demo的文章，看着不怎么高大上，反而感觉烂烂的

👍 2

🗨️ 回复
- 

CregskIN

前端学徒 | 2年前

异常监控部分，是否有可能用ErrorBoundaries解决的呢？[zh-hans.reactjs.org](#)

👍 点赞

🗨️ 回复
- 

页面工具人

页面仔 @ 未知 | 3年前

有点短，怪不得阅读时间那么短。要是被问，哑口无言。

👍 3

🗨️ 回复
- 

console_man

Web前端 | 4年前

期待讲下无埋点内容

👍 点赞

🗨️ 回复
- 

转用冰西瓜

4年前

监控分类里面的“页面埋点”跟后面的两个不是同一个类型的东西，后面两个是从监控的需求，“页面埋点”是实践的一种，“页面埋点”也可以用来做性能监控。个人认为应该把“页面埋点”换成“用户行为监控”。

👍 点赞

🗨️ 回复
- 

wei

4年前

这属于赶进度啊。 . .

👍 17

🗨️ 2



千百度海

3年前

在过年的时候写的，有那么一丢丢赶进度的感觉

👍 点赞

🗨️ 回复



石头就是我...

2年前

写的什么啊，退钱了

👍 1

🗨️ 回复
- 

suhangdev

Node @ Ant | 4年前

能不能分享下前端无埋点方案的实现思路

👍 1

🗨️ 回复
- 

PerCy64890

4年前

没看够啊，这里内容不够仔细，只是知道个大概

👍 点赞

🗨️ 回复
- 

zexiplus

web前端 @ Coding | 4年前

越到后面就越想大纲，其实知识点很多，但需要自己搜集一下

👍 1

🗨️ 1



谏鲜花

4年前

的确大纲

👍 点赞

🗨️ 回复



MingYuan

公众号：一只前端 | 4年前

哇 到这都没人了么

👍 点赞

🗨️ 回复



566

4年前

这里的知识点也太少了吧

👍 点赞

🗨️ 回复



crackedlove

前端攻城狮 | 4年前

怎么没人？

👍 2

🗨️ 回复