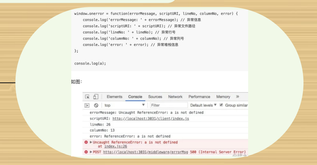
技术广度

sentry

不用sentry怎么捕获和上报异常？

捕获异常的方式：

- try/catch
- **window.onerror**
  - 监控JavaScript运行时错误（包括语法错误）和 资源加载错误
  - 提供了全局监听异常的功能
  - 可以获取到报错的位置
  - 缺点：window.onerror这样的异常捕获不能捕获promise的异常错误
- unhandledrejection
- 框架自身的捕获机制（优先级高于window.onerror） — Vue.config.errorHandler



```
window.onerror = function(errorMessage, scriptURL, lineNo, columnNo, error) {
  console.log('errorMessage: ' + errorMessage); // 异常信息
  console.log('scriptURL: ' + scriptURL); // 异常文件路径
  console.log('lineNo: ' + lineNo); // 异常行号
  console.log('columnNo: ' + columnNo); // 异常列号
  console.log('error: ' + error); // 异常堆栈信息
};

console.log(a);
```

如图：

```
Vue.config.errorHandler = function (err, vm, info) {
        let {
            message, // 异常信息
            name, // 异常名称
            script,  // 异常脚本url
            line,  // 异常行号
            column,  // 异常列号
            stack  // 异常堆栈信息
        } = err;

        // vm为抛出异常的 Vue 实例
        // info为 Vue 特定的错误信息，比如错误所在的生命周期钩子
}
```

上报机制

- 设置node中间层 — node安装'source-map'插件
- 我们通过前端传过来的异常文件路径获取服务器端map文件地址，然后将压缩后的行列号传递给sourceMap返回的promise对象进行解析，通过originalPositionFor方法我们能获取到原始的报错行列号和文件地址，最后通过ajax将需要的异常信息统一传递给后台存储，完成异常上报。

https://juejin.cn/post/6844903582672650253

sentry原理

异常捕获

全局捕获

- 通过全局的接口，将捕获代码集中写在一个地方，可以利用的接口有 — `window.addEventListener('error') / window.addEventListener("unhandledrejection") / document.addEventListener('click')`
- 框架级别的全局监听
  - vue、react都有自己的错误采集接口
  - 例如aixos中使用interceptor进行拦截，
- 对全局函数进行封装包裹，实现在在调用该函数时自动捕获异常 — 比如重写setTimeout：

```
const prevSetTimeout = window.setTimeout;

window.setTimeout = function(callback, timeout) {
  const self = this;
  return prevSetTimeout(function() {
    try {
      callback.call(this);
    } catch (e) {
      // 捕获到详细的错误，在这里处理日志上报等了逻辑
      // ...
      throw e;
    }
  }, timeout);
}
```

- 专门写一个函数来收集异常信息，在异常发生时，调用该函数

单点捕获

上报方式

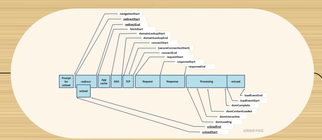
- 优先fetch,其次XHR，最后new Image
- 使用new Image上报

```
var REPORT_URL = 'xxx'  //数据上报接口
var img = new Image; //创建img标签
img.onload = img.onerror = function(){  //img加载完成或加载src失败时的处理
    img = null; //img置空，不会循环触发onload/onerror
};
img.src = REPORT_URL + Build._format(params); //数据上报接口地址拼接上报参数作为img的src
// 或者
<img src="images/bg.png" onerror="javascript:this.src='images/bgError.png';">
```

- 为什么不能用请求其他的文件资源（js/css/ttf）的方式进行上报？
  - 创建资源节点后只有将对象注入到浏览器DOM树后，浏览器才会实际发送资源请求。图片不仅不用插入DOM，而且只要在js中new出Image对象就能发起请求。
  - 载入js/css资源还会阻塞页面渲染，没有阻塞问题
  - 在没有js的浏览器环境中也能通过img标签正常打点

前端性能监控

window.preformance.timing



由于`window.preformance.timing`是一个在不同阶段，被不停修正的一个参数对象，所以，建议在window.onload中进行性能数据读取和上报。

工具

Google PageSpeed Insights

可分析网页内容并生成建议，还可以分别对移动端和PC端进行分析

https://pagespeed.web.dev/?utm_source=psi&utm_medium=redirect

LighHouse

Lighthouse 是一个开源的自动化工具，用于改进网络应用的质量。 您可以将其作为一个 Chrome 扩展程序运行，或从命令行运行。 您为 Lighthouse 提供一个您要审查的网址，它将针对此页面运行一连串的测试，然后生成一个有关页面性能的报告。

https://developers.google.com/web/tools/lighthouse

主题