技术调研选型

1. 需求背景

2. 技术选型

3. 生产环境场景整理

4. 功能测试，性能测试

5. 结论

天空配音软件

window版下载：

https://wwp.lanzouq.com/b02upfdxe

密码8888

备用 链接：https://pan.baidu.com/s/1O0Sk-AA1fmtlvBNUz9gk8A

提取码：tk99

关注UP主更新

技术调研编写

由于某次需求的需要，我进行了一次技术调研，内容是调研前端将 pdf 文件转为图片的解决方案，我接到这个需求的第一时间，立马打开搜索引擎，翻看了十分钟后，很快啊得出了一个口头结论但这肯定是不行的，十分钟就能整明白的事情就不叫技术调研了，也无需技术调研，然而如何摆好一个技术调研的正确姿势，也没有啥标准模板，让开发人员写文档本来就够痛了，再加上一个没有标准的场景，痛上加痛，既然我想做好这次技术调研，就必须解决这个痛点，那就顺便把这个问题也调研一下吧网上关于如何做好技术调研的文章也有一些，本文主要是贴合自身，从前端的角度进行解读

了解需求

首先你肯定要足够了解需求的，然后才能确定一个技术调研方向

比如需要你实现一个环绕地球的3D显示效果，你一看到 3D 立马就想到 three.js 甚至是 webgl，然后二话不说开始闷头研究起来，结果研究了两天后，在开始做需求的时候，发现需求的重点并不是那个3D地球，而是环绕地球展示的数据点，实际上这是个可视化展示的需求而不是3D效果需求，echarts 才是最佳解决方案

那么这个过程中你固然是可以了解到一些跟 webgl 相关的知识，但毕竟跟需求产生了偏差，对于当前需求来说可能是无用功所以一定要确定好要求，准确分析出需要准备的技术点，再进入下一步当然，不仅是技术调研，平常的技术开发也是需要这一步的，即确定需求的要求然后你才能从技术的角度跟PM讨价还价什么时候需要技术调研

就像文章开头提到的那样，你得先确定一件事情需要调研你才能开始调研，如果十分钟就能完全确定的事情就没必要大费周折了比如，你新启动一个项目，在 vue 和 react 中犹豫，不知道到底用哪个好，如果这个问题放到5年前，你可能确实需要调研一番，但放到当下这个时间点，显然就没必要了，十分钟足以判断为什么5年前需要呢？因为那个时候，无论是 react 还是 vue，都不够成熟，特别是 vue 在 2014 年才起步，没有现在那么普及，对于当时的前端圈来说，这两个东西都还算是比较新颖的事务，有经验的人不多，可搜集到的资料也没有那么全，为了保证线上的稳定性，就必须先对它们仔细调研一番才能决定是否启用

有些技术存在的时间已经足够久了，资料也比较齐全，但也不代表就能拿来就用大多数前端可能都涉及不到可视化方面的开发，但可能突然某一天你就接到了一个3D环绕地球的可视化需求，准确分析了需求的意图后，你去网上搜了下，找到了最火的 echarts，但是从效果上来看，明显不可能随便三两下就能实现的了，可能需要考虑很多问题，例如需要哪些配置？是否需要UI出图？用的canvas还是webgl？是否有兼容性上的问题？这些细节性的东西，可能就需要你亲自去实践一番了当这些细节都已经确定了之后，你发现还需要在3D地球的周围加一些飞线之类的东西，或者是需要具备点击地球上某个点实现地图放大/缩小的能力，那么你就还得看下 echarts 是否支持这种功能，如果不支持是否有其他替代方案等那么，综合上述，需要技术调研的场景包括但不限于以下三个方面：

新技术，资料较少，社区不完备足够成熟，但不确定细节实现想做 xxx 功能，但不确定能不能实现 调研方向 现存方案

得益于前端生态的百花齐放，对于同一个问题可能存在很多种解决方案，抛开那些重复的轮子以外，剩下的方案既然能够存在下去就说明它们有存在的理由，必然都有各自的优缺点，也都有各自最适合使用的场景你需要先尽可能地罗列出市面上已存的较为流行的方案，然后再对这些方案进行各方面对比，选出一个最适合你当前需求需要的方案

对于3D环绕地球效果这个需求来说，echarts、three.js、antdv、d3、chart.js 等都是潜在的可选方案，但是你不可能闭着眼睛随便选一个就行了，要去一一了解它们的各自优缺点，找出一个最适合你自己的当然，有些需求的解决方案可能就唯一的一个，例如前端操作PDF，线上可用的可能就只有 pdf.js 了，其他的可能都只是玩具，那么就只需要专注分析这一个即可对比环节了解了需求，列举了所有可用方案后，下面就进入最重要的选优环节了，方案对比的方向不要求能够覆盖所有方面，但最起码应该覆盖一些关键节点

对比不应当仅是客观地描述各个解决方案的优劣，更主要的是结合你当前的实际需求，从不同的方向上给各个解决方案进行打分，以解释明白为什么从 A 功能上看，要选 α 方案，而从 B 功能上看，β 方案更好

原理：实现原理基本上决定了具体方案的方方面面，了解了原理，才能更好地进行分析

例如 echarts 是 svg/canvas 双引擎，而 three.js 更多的是基于 webgl，那么如果想要更好地控制它们，前者要求开发者更熟悉 svg/canvas，而后者可能需要开发者具备一定的 webgl 知识；例如，pdf.js 是依据pdf文件标准，纯js进行二进制文件解析，不依赖特定浏览器API/特性实现的知道了原理之后，对于其优缺点就能有进一步的认知，同时可以结合自己对于其底层原理相关知识的经验，得出更多的结论

活跃度主要从 github star 数、代码更新频率、issue响应速度、文档完整度、在线示例、官方团队和社区的规模等方面进行判断

一个低于 1k star、超过半年没有更新、issue很少或者响应速度很慢，低于 3 个 contributor、文档只有几段话的项目一般而言是无法用于线上环境的

例如，echarts 由业内知名公司开源，有专门团队维护、有专门的社区、几乎每天都有commit，显然是一个可选方案

生产环境可用性

主要考虑的是，市面上是否已经存在使用这个解决方案的案例了，如果有其他规模较大的产品线上使用了这个方案，那么在一定程度上可以证明，这个方案是可以放在线上的

比如，对于 echarts 和 antv 来说，市面上使用它们的产品比比皆是，毫无疑问是可以线上化的方案；再比如，对于 web在线编辑器来说，ACE 和 CodeMirror 都是很好的开源产品，但从目前使用广泛度来说，CodeMirror 的受欢迎程度更高，羽雀、github都是基于其打造自己的在线编辑器，那么从这个方面相对来说，CodeMirror 可能比 ACE 更好

如果有内部团队曾经有过这方面的使用案例，那么就更需要去沟通一番了，可能他们的使用场景跟你的不一样（完全一样的话可能就没必要重复调研了），但肯定有可以借鉴的地方，了解他们的使用场景、使用过程中遇到的坑、是否有踩坑文档、是否推荐使用等

功能

技术方案是为实际业务需求所服务的，选出的技术方案必须能够满足需求所要求的所有功能

对于3D环绕地球效果来说，echarts、three.js 都能实现这个效果，而 antv、chart.js 则没有直接提供这个选项；而如果你想要可视化是关系数据（树状图、脑图、流程图）并且配色更专业协调，那么antv-G6 可能就是最佳选择

兼容性

前端必然需要考虑兼容性，比如浏览器的最低兼容版本、是否涉及pc端/移动端等，这其实也算是一种功能，用户需要能使用你所提供的功能才行

echarts、antv基本上都支持到 IE9，但是 antv 对于移动端有更佳的优化版本，所以如果你是在移动端使用，那么在其他主要功能都能满足的前提下，应该优先考虑 antv

性能

可以从包体积、渲染速度方面进行考量

包体积过大，一方面会导致页面加载速度变慢，其次是太大的体积意味着在一般情况下其性能也不会好到哪里去，例如，对于移动端gzip之后超过200k，pc端gzip之后超过 500k，都可以认为是体积有点大了（数字只是凭经验给出的）

渲染太慢导致页面空白时间过长或者浏览器失去响应，都是很影响用户体验的事情，为了加入一个功能而导致另外一个功能效果变差，那么还不如不加

除了这两个通用的之外，对于特定的技术方案可能还有特定的衡量指标，比如对于前端pdf转图片这个需求，需要衡量的指标应该还有转换过程需要耗费的时间，如果转换一个10页的 pdf需要5s以上，这就太慢了，如果再考虑到这个功能可能会存在几十乃至是上百页的pdf文件，那么显然用户是无法接受的

另外，你可以亲自对关键性能指标进行测试，列出详细的数据，会更有说服力，比如你需要在移动端引入一个可视化库，那么你就可以在移动端分别测试 antv 和 echarts 从加载到渲染完毕所需耗费的时间，得出一个耗时结果

可维护性

主要从工作量、学习/维护成本、对于业务的侵入度、最佳实践等方面考量

一般情况下，开箱即用的肯定比需要一大堆配置项的要好，没有额外学习成本的肯定比需要专业知识的要好（比如 webgl 就是专业知识），业务侵入度越低越好，如果能有官方/社区的最佳实践可参考那就最好不过了

缺陷及隐患

关注缺点的优先级高于关注优点的优先级，优点再多，也可能因为一个缺点而不能被应用

比如对于 antv，缺乏对于3D地球的直接支持，那么其他方便做的再好，对于你需求都是于事无补

不过也不是所有缺陷都不能容忍的

比如对于前端pdf转图片，pdf.js 直到目前为止依旧存在很多缺陷，还有一些issue创建几年了都没关，但这些问题如果并不影响你需求的实现，并且以后也不太可能涉及到这些，那么就是没问题的

你的项目是pc端项目，那么pdf.js在移动端的缩放、兼容等问题就不是问题；你不可能加载超过100页的复杂内容pdf，那么pdf.js处理大文件时可能遇到的问题你就无需担心

就算是可能与你需求相关的问题，如果其在可容忍范围内，那么也是可以接受的

比如pdf.js将原pdf文件转为图片后，清晰度会降低，但如果这并不明显影响体验，那么也是可以忽略的

其他针对具体的技术方案，可能还有其他一些比较重要的环节，需要具体需求具体对待

调研一个表单组件，你可能需要考虑到其安全性（是否默认防范xss攻击）；调研一个UI库，你可能需要考虑到到其是否具备跨端能力

产出文档

基本上上述信息足以支撑起得出一个调研结论了，但这个结论不能只存在于你自己的脑海中，你应当将这个过程记录下来，可以就按照上面的步骤作为模板，形成一份调研文档进行输出 这份调研文档应当包括以下四个方面：

1、需求背景

你的调研文档可能会被其他不熟悉你所做需求的人查看，对于一个做业务的技术人员来说，脱离具体业务谈技术就是耍流氓，你好不容易调研了一番然后又产出一篇文档，那么当然想要更多的人能够看得懂得到更多的认同

2、一句话结论

为了能快速给出一个定调，作为详细内容的“太长不看版”

不是所有人都想先完整地看完所有调研内容然后才得到一个结论的，你的详细调研内容都属于过程，而结论可能才是很多看你调研文档的人最先关心的东西，所以你应该提供一句简短的断言结论

3、现存方案对比记录

详细的对比过程是为了调研结论的细节和说服力，让别人更加认同你的结论

这个对比记录的内容主要应当围绕你当前面临的实际业务需求展开，除此之外，还可以描述一些需求可能涉及不到的点，比如你想调研pdf.js在pc端切割pdf文件转为图片的支持情况，那么除了这方面之外，你还可以额外描述一下其在移动端的支持度，给出一个更全面的参考，可能会对其他查看你调研报告的人产生启发当然还是要注意主次关系，大部分内容应当都是围绕你所面临的实际需求，额外的东西应当放在次要位置

4、参考文档链接

作用和现存方案对比记录差不多，都是你调研结果的支撑论据，也方便其他参考你报告的人自行去获取更多的内容

参考

当我们在做技术调研的时候，到底需要做什么？怎么做？

技术调研的模式

如何做好技术调研

技术调研流程分享

地址：[理清思路，前端技术调研到底应该怎么做？ - 掘金 (juejin.cn)](https://juejin.cn/post/6901845776880795662)

GPT标准模板：

调研 "FISH-SPEECH" 技术文档时，你需要遵循一般的技术文档调研方法，并专注于这一特定技术的各个方面。以下是一个针对 "FISH-SPEECH" 技术文档调研的指导步骤：

确定调研目标：

明确你希望通过调研 "FISH-SPEECH" 技术了解到什么，例如其原理、应用场景、优缺点、与其他技术的比较等。

信息收集：

技术论文和文章：查找有关 "FISH-SPEECH" 技术的学术论文、技术文章或白皮书。

在线资源：搜索相关的技术博客、论坛讨论、教程或演示视频。

官方文档：如果 "FISH-SPEECH" 是一个特定的产品或框架，请查阅其官方文档。

专家访谈：如果可能，与该技术领域的专家或实践者进行交流。

分析和整理：

技术原理：描述 "FISH-SPEECH" 的工作原理和核心技术。

应用实例：探讨 "FISH-SPEECH" 在实际中的应用案例。

优缺点分析：基于调研资料，分析该技术的优势和潜在的局限性。

技术比较：与其他类似技术对比，突出 "FISH-SPEECH" 的独特之处。

撰写报告：

引言：介绍调研的背景、目的和 "FISH-SPEECH" 的概念。

方法论：说明调研方法和使用的资源。

主体内容：详细介绍你的发现，包括技术详解、应用案例、优缺点、比较分析等。

结论和建议：基于调研结果，提出你的见解和对未来发展的预测。

参考文献：列出所有引用的资料。

审阅和修改：

确保信息的准确性和报告的清晰度。

报告呈现：

将调研成果以适当的形式展现给目标读者。

由于 "FISH-SPEECH" 这一术语不是广为人知的通用技术术语，确保在调研过程中准确理解和使用该术语，同时确认它的定义和应用范围。如果 "FISH-SPEECH" 是一个较新或较少被讨论的概念，可能需要更多地依赖于直接的专家访谈和最新的技术报告