# Nodejs

## http

http模块是node.js中非常重要的一个核心模块。

可以使用http.createServer方法创建一个http服务器；

可以使用http.reuqest方法来创建一个http客户端；

HTTP是一个用来传送数据的应用层协议，其底层使用TCP传输协议；

HTTP是互联网数据通信的基础，要相应客户端请求（如浏览器访问），首先需要部署HTTP服务器。

<https://itbilu.com/nodejs/core/N1okQ7Eh.html>

Nodejs如何创建HTTP服务器？

https://itbilu.com/nodejs/core/N1okQ7Eh.html

var http = require('http');

// 创建server对象，并添加request事件监听器

var server = http.createServer(function(req, res) {

    res.writeHeader(200,{'Content-Type':'text/plain'});

    res.end('http://erp.office.51fanli.com/');

});

server.listen(3000)

Nodejs对客户端的支持

var http = require('http');

var options = {

    host: '10.200.137.53',

    port: '8080',

    method: 'GET',

    path: '/Okr/OKpi/getProjectList?page=1&name=&type='

}

var req = http.request(options);

req.on('response', function(res){

    res.setEncoding('utf8');

    res.on('data', function(chunk){

        console.log('收到数据：%s', chunk);

    });

});

req.end();

返回结果：

# Vue3

## Composition （组合式）API

Vue2中，随着功能的增加，组件变得越来越复杂，越来越难维护，而难以维护的根本原因是Vue的API设计迫使开发者使用watch、computed、methods选项组织代码，而不是实际的业务逻辑。

另外vue2缺少一种较为简洁的低成本的机制来完成逻辑复用，虽然可以minxis完成逻辑复用，但是当minxi变多的时候，会使得难以找到对应的data、computed或者methods来源于哪个mixin，使得类型推断难以进行。

所以composition API的出现，主要是为了解决options API带来的问题，第一个是代码组织问题，composition API可以让开发者根据业务逻辑组织自己的代码，让代码具备更好的可读性和可拓展性。第二个是实现代码的逻辑提取和复用。

Composition API在setup函数中执行。

Mixins的最大的问题是命名冲突。



## 编译方面的优化

标记和提升所有的静态节点，diff的时候只需要对比动态节点的内容。

* **静态提升**

所有的静态节点都被提升到render方法之外，只会在应用启动的时候被创建一次，只会使用只需要应用提取的静态节点，随着每次的渲染被不断的复用。

## Patch flag

在动态标签末尾加上响应的标记，只有带patchFlag的节点才被认为是动态的元素，会被追踪属性的修改，能快速的找到动态节点，而不用逐个逐层遍历，提高了虚拟dom diff的性能。

# Vue性能优化

* **代码层面的优化**

1、v-if和v-show区分使用场景

v-if适用于在运行时很少改变条件，不需要频繁切换条件的场景；

v-show则适用于需要非常频繁切换条件的场景。

2、computed和watch区分使用场景

3、v-for遍历必须为item添加key，且避免同时使用v-if

4、长列表性能优化

纯粹的数据展示，不会有任何改变，我们不需要vue来劫持我们的数据，在大量数据展示的情况下，这能够明显的减少组件初始化的时间。使用Object.freeze来冻结一个对象。

5、事件销毁

Vue组件销毁时，会自动清理它与其它实例的连接，解绑它的全部指令及事件监听器，但是仅限于组件本身的事件。如果在js内使用addEventListene等方式是不会自动销毁的，所以我们在组件销毁时需要手动移除这些事件的监听，以免造成内存泄露。

6、图片懒加载

7、路由懒加载

Vue是单页面应用，可能会有很多的路由引入，这样使用webpack打包后的文件很大，当进入首页时，加载的资源过多，页面会出现白屏的情况，不利于用户体验。

如果我们能把不同路由对应的组件分割成不同的代码块，然后当路由被访问的时候才加载对应的组件，这样就更加高效了。

1. 第三方插件按需引入
2. 优化无限列表性能

如果是非常长或者无限滚动的列表，那么需要采用窗口化的技术来优化性能，只需要渲染少部分区域dom节点的时间。

可参考以下开源项目：vue-virtual-scroll-list和vue-virtual-scroller

10、服务器渲染SSR或者预渲染

* **Webpack配置层面的优化**
* **基础的web技术层面的优化**

# 微信小程序weapp

<https://www.cnblogs.com/teahouse/p/11504361.html>

<https://developers.weixin.qq.com/community/develop/doc/d1421cd729a51548672430e544c458b2>

## weapp的运行环境

根据微信官方说明，weapp的运行环境有3个平台，IOS的webkit（苹果开源的浏览器内核）、android的X5（QQ浏览器内核），开发时用的nw.js（c++实现的web转桌面应用）。

## Weapp为什么不直接运行在浏览器（webview）中，而要绕过浏览器直接调用内核

因为运行在浏览器中的weapp是做不了监控的，而weapp的表现是半native app，半web app，而native app和web app的一个很重要的区别就是native app有自己的生命周期，在这之中，我们可以根据生命周期的不同时间段做出不同的调整，比如常驻内存，防止被系统杀掉、系统后台保存活度等等，而web app是没有这回事的，

# 项目相关问题

## 项目中开发的组件

1. 上传文件展示上传文件的组件

可传入的参数：

1. accept 字符串，input应该接受的文件类型；
2. 已上传的文件列表，如编辑的情况下
3. 上传完成后的文件列表
4. 保存的位置

支持的功能

1. 支持多文件上传
2. 支持显示每个文件的上传进度 progress
3. 支持大文件分段上传、断点续传
4. 支持删除
5. 支持使用钉钉自带的图片上传功能
6. 支持图片缩略图预览
7. 其他类型文件支持链接跳转
8. 专门用于后台列表页面的组件；组件包含表格导入、表格展示、分页、编辑框
9. 调用该组件需要传入的参数为 模块名和类名
10. 根据模块名和类名找到对应的后端类
11. 后端类提供固定的方法getCols，返回表格表头的中文和英文映射关系、

表格操作列的按钮名称、编辑框中需要修改的项。

1. 后端类提供固定的方法getList，返回表格需要展示的数据列表和数据总数。
2. 编辑框通过判断编辑项的类型来展示输入框、下拉框等不同的表单。

后期考虑将上面需要后端返回的getCols里面的内容通过后台页面配置的方式获取；

1. 加载loading组件
2. 日期组件
3. 上传表格展示表格的组件
4. 选项卡组件
5. list组件
6. 搜索组件

参数：提示词

好处：统一样式，提高开发进度

## 业务组件库

前端团队建设业务组件库来解决

1. 业务组件跨项目复用的问题
2. 同时统一代码实现，统一代码质量

从而提高业务的开发效率。

## 组件设计原则

* 高效，性能上
* 易用：

引用组件的开发人员短时间内是否更快的学习上手；

用户层面上不需要用户进行深度思考，一看就知道怎么操作。

* 美观：B端产品是不应该有限考虑好看的，不管从用户价值还是商业价值。相对而言，美观的侧重点是简洁好用，它也是美观的范畴范围。
* 一致性、标准性
* 独立性

描述了组件的颗粒度，遵循单一职责原则，保持组件的纯粹性；

属性配置等API对外开放，组件内部状态对外封闭，尽可能的少与业务耦合。

* 无环依赖原则
* 入口处检查参数的有效性，出口处检查返回的正确性
* 扁平化参数

除了数据，避免复杂的对象，尽量只接受原始类型的值

## 大文件切割

1、分段数 = 文件总大小 / 切割大小 向上取整

Math.ceil(10/3) = 4 向上取整

Math.round(10/3) 四舍五入

Math.floor(10/3) 想下取整

2、进度极端：将上传进度的90%除以分段数，上传完一个将进度修改

余下的10%用于最终的合并请求。

3、Blob [blɒb]

Blob对象表示一个不可变、原始数据的类文件对象。它的数据可以按文本或二进制的格式进行读取，也可以转换成ReadableStream来用于数据操作。

Blob表示的不一定是JavaScript原生格式的数据。File接口基于Blob，继承了blob的功能并将其扩展使其支持用户系统上的文件。

4、File

（1）File对象是来自用户在一个<input>元素上选择文件后返回的FileList对象，

（2）File对象是特殊类型的Blab

（3）File的属性：

File.lastModifiedDate 返回当前File对象所引用文件最后修改时间的Date对象；

File.name 返回当前File对象所引用文件的名字

File.size 返回文件的大小

File.webkitRelativePath 返回File相关的path或URL

File.type 返回文件的类型

（4）File的方法

File接口没有定义任何的方法，但是它从Blob接口继承以下方法：

File.slice()

1. 上传文件请求头的content-type的值是**Content-Type:**

multipart/form-data; boundary=----WebKitFormBoundaryH1NDUHOs6FoCr2rZ

# 项目介绍

<https://www.v2ex.com/t/672672>

<https://www.v2ex.com/t/645335>

## 人事管理系统

**项目难点**

* SKU
* 骨架屏（Skeleton Screen）
* 长列表优化
* Vue大文件上传和断点续传
* 前端监控系统
* 异常问题排查

**遇到的难点、问题及解决办法**

|  |  |
| --- | --- |
| **1** | * 大文件上传，如视频200M   <1>上传文件类型的判断；不是通过后缀名判断；而是转化为二进制流，判断头信息  <2>文件切片 Blob.slice  <3>断点续传  <4>文件算hash，用hash做唯一标识  方式1：webwork 防止卡顿线程  this.worker = new Worker(‘/’)  方法二：浏览器每一帧都有空闲时间  流畅的动画或者网页60FPS，也就是1秒渲染60次，1帧大概16.6ms  卡顿造成的原因：同步任务的耗时远大于16ms  我们可以利用空闲来计算，一旦有一个优先级更高的同步任务，返回浏览器控制权，等待下一次空闲，所谓浏览器的fiber架构，就是这个理念。  Fiber：   1. 虚拟dom从树拆成链表 易于中断 2. Diff过程用空闲时间来做   Time-slice  requestIdleCallback  方法三、  布隆过滤器 牺牲一点进度 获取更高的效率  <5>时间切片  <6>控制并发数，异步任务并发 + 报错重试  <7> |
| **2** | * 动态生成页面，页面input、select太多，直接传参数（组件类型，id、方法等），通过if判断生成，无需重复编写html代码 |
| **3** | * 页面使用easyUI的输入框、下拉框、按钮太多了，导致页面渲染非常的慢。   这个页面是一个员工信息管理页面，主要包含几十列的table列表、几十个筛选项、几十个编辑项；  解决办法：  方法一：按需加载  easyui是通过class来标识的，首先将所有的对应的easyui class换成与easyui无关的class；  其次，在每次需要使用的时候，再转化为easyui对应的class，比如：点击编辑以后，再将编辑框里面的所有项的class加上easyui对应的class；  方法二：能不是使用easyui的地方就不要使用，这样会导致整体的样式不一致，但我们可以将样式修改成一样的。 |
| **4** | * 我们这个系统很多的页面涉及到上传的功能，一般有图片、pdf、excel的上传   它存在的问题是，每一处的上传都由各自的代码，很多处的代码都是复制粘贴过去的；  我需要解决的是，去除重复代码，将上传功能提取到公共文件里面，前前后后有做过大约3次大的变动；  第一次就是将以前的上传代码保留一份，放入公共模块，供使用的地方调取；  第二次改动是因为发现前面写的上传公共方法不支持一个页面有多处上传的情况  第三次改动是因为ios的钉钉不支持图片的上传  再后面是写了vue版本的公共上传组件。 |
| **5** | * 一个vue页面，它引用了三个子组件；其中两个子组件的接口需要依赖另外一个子组件接口的返回值。   B组件需要依赖A组件里面的接口返回值；  A组件创建（created）和B组件挂载（mounted）时间间隔，比A组件接口返回时间短，  所以在A组件创建阶段请求接口，B组件挂载阶段请求接口不行。  Vuex设置一个uid，默认值为0，A组件接口接口返回后给uid赋值，  B组件接口判断uid是否为0，如果是0的话等待100ms，再判断，如此一直循环，直到uid不为0，再继续请求B接口 |
| **6** | * 异常问题排查   北京分公司的一位同事访问erp.office.51fanli.com的广告排期相关的页面返回空白页； |
| **7** | * 使用jOrgChart实现组织架构树 * Dom节点生成png，支持将架构树转化为图片下载，要熟悉下原理   <https://github.com/tsayen/dom-to-image/blob/master/src/dom-to-image.js> |

## 爱予亲子阅读微信小程序

在这个项目是我自己独立开发的，我做过的事情有：

购买服务器、域名、

搭建运行环境，linux系统，使用nginx部署，配置https，

在这个项目中使用到微信接口有：

1. 小程序更新
2. 获取用户信息
3. 微信支付功能
4. 微信模板发送信息

遇到的难点、问题及解决办法

* ios8卡顿问题

scroll-view的bindscrolltolower下拉刷新确实会存在异常卡顿

解决方式：使用view再使用onReachBottom（页面上拉触底事件的处理函数）

* 登录保持

## 广告排期系统

这个系统里面涉及到了一个排期面板，这个是比较复杂的

## 同事圈

详情页返回列表页保持

keep-alive

路由的地方增加meta

实现滚动行为的代码：router/index.js

|  |
| --- |
| scrollBehavior(to, from, savePosition){  if(savePosition){  return savePosition  }else{  return {x:0,y:0}  }  } |

表情插件

1. 使用css 雪碧图（精灵图）将所有的小表情合并为一张图片，使用background-position显示指定的小表情；
2. 数据库存储小表情英文别名，取出来后，替换为dom
3. 使用div的contenteditable=”true”来实现小表情的输入

# 网上学习链接

<https://blog.csdn.net/weixin_42623421/article/details/106856141>

# 场景题

## 登录

第一步：服务器配置

在php.ini文件中配置（session.name可以自定义）

|  |
| --- |
| Session.name =PHPSESSID22  Session.auto\_start=0 |

第二步：浏览器发送第一个http请求时，服务器返回一个sessionID给浏览器

具体形式为：http请求的response中设置set-cookie

则sessionID保存在了浏览器的cookie中

|  |
| --- |
| Set-Cookie: PHPSESSID22=aqu7hrh85rhsrc4qe821behvm6; path=/ |

第三步：浏览器发送第二个http请求，http请求的request中携带cookie

|  |
| --- |
| Cookie: PHPSESSID22=aqu7hrh85rhsrc4qe821behvm6 |

第四步：服务器验证是否登录，没有登录跳转到登录页

第五步：浏览器post参数携带用户名和MD5加密的密码，服务器通过用户名、密码及其他参数生成一个key存储在服务器的session中，验证是否登录就是判断这个session是否存在

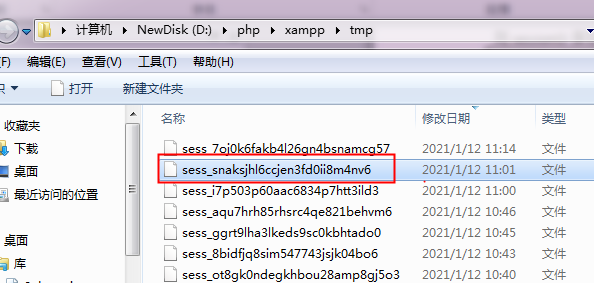
## PHP的session机制《PHPSESSID》

1. 默认机制，用磁盘文件来实现PHP会话。Php.ini配置如下

|  |
| --- |
| session.save\_handler=files  session.save\_path="D:\php\xampp\tmp" |

1. session\_start()

A、session\_start()是session机制的开始，



## JAVA的session机制 《JSESSIONID》

<https://www.cnblogs.com/barneywill/p/10116996.html>

## 单点登录（SSO）

上图是CAS官网上的标准流程，具体流程如下：

1. 用户访问app系统，app系统是需要登录的，但用户现在没有登录。
2. 跳转到CAS server，即SSO登录系统，**以后图中的CAS Server我们统一叫做SSO系统。** SSO系统也没有登录，弹出用户登录页。
3. 用户填写用户名、密码，SSO系统进行认证后，将登录状态写入SSO的session，浏览器（Browser）中写入SSO域下的Cookie。
4. SSO系统登录完成后会生成一个ST（Service Ticket），然后跳转到app系统，同时将ST作为参数传递给app系统。
5. app系统拿到ST后，从后台向SSO发送请求，验证ST是否有效。
6. 验证通过后，app系统将登录状态写入session并设置app域下的Cookie。

至此，跨域单点登录就完成了。以后我们再访问app系统时，app就是登录的。接下来，我们再看看访问app2系统时的流程。

1. 用户访问app2系统，app2系统没有登录，跳转到SSO。
2. 由于SSO已经登录了，不需要重新登录认证。
3. SSO生成ST，浏览器跳转到app2系统，并将ST作为参数传递给app2。
4. app2拿到ST，后台访问SSO，验证ST是否有效。
5. 验证成功后，app2将登录状态写入session，并在app2域下写入Cookie。

这样，app2系统不需要走登录流程，就已经是登录了。SSO，app和app2在不同的域，它们之间的session不共享也是没问题的。

**有的同学问我，SSO系统登录后，跳回原业务系统时，带了个参数ST，业务系统还要拿ST再次访问SSO进行验证，觉得这个步骤有点多余。他想SSO登录认证通过后，通过回调地址将用户信息返回给原业务系统，原业务系统直接设置登录状态，这样流程简单，也完成了登录，不是很好吗？**

**其实这样问题时很严重的，如果我在SSO没有登录，而是直接在浏览器中敲入回调的地址，并带上伪造的用户信息，是不是业务系统也认为登录了呢？这是很可怕的。**

总结

单点登录（SSO）的所有流程都介绍完了，原理大家都清楚了。总结一下单点登录要做的事情：

* **单点登录（SSO系统）是保障各业务系统的用户资源的安全 。**
* **各个业务系统获得的信息是，这个用户能不能访问我的资源。**
* **单点登录，资源都在各个业务系统这边，不在SSO那一方。 用户在给SSO服务器提供了用户名密码后，作为业务系统并不知道这件事。 SSO随便给业务系统一个ST，那么业务系统是不能确定这个ST是用户伪造的，还是真的有效，所以要拿着这个ST去SSO服务器再问一下，这个用户给我的ST是否有效，是有效的我才能让这个用户访问。**

## **微信扫描登录**

* 1. 用户打开网站https://wx.qq.com/ 时，浏览器向服务器发送登录请求，服务器收到请求后，随机生成一个uuid返给浏览器。
* 同时服务器会将这个id作为key值存入redis服务器，同时设置一个过期时间，在过期后，用户登录二维码需要进行刷新重新获取；

2、浏览器将uuid作为参数请求服务器，服务器收到请求后生成一个二维码返给浏览器。

* 二维码其实是一个包含uuid的链接

3、浏览器拿到二维码和uuid后，会每隔一秒向浏览器发送一次登录是否成功的请求。请求中携带uuid作为当前页面的标识。

4、用户用已经登录的手机微信扫描二维码，拿到一个带有uuid的url。

5、手机端微信携带uuid和自身已登录的token向手机服务器发送用户登录请求，服务器收到请求后从token中解析出用户信息。并将解析处理的用户信息和uuid进行绑定。

6、当浏览器再一次请求是否登录时，

## 微前端

<https://m.imooc.com/article/304768>

1. 什么是微前端？

微前端是一种类似于微服务的架构，它将微服务的理念应用于浏览器端，即将Web应用由单一的单体应用转变为多个小型前端应用聚合为一的应用。各个前端应用还可以独立运行、独立开发、独立部署。

微前端不是单纯的前端框架或者工具，而是一套架构体系。

二、为什么会有微前端？

任何新技术的产生都是为了解决现有场景和需求下的技术痛点，微前端也不例外：

* 1. 拆分和细化：当前前端领域，单页面应用（SPA）是非常流行的项目形态之一，而随着时间的推移以及应用功能的丰富，单页应用变得不再单一，而是越来越庞大也越来越难以维护，往往是改一处而动全身，由此带来的发版成本也越来越高。微前端的意义就是将这些庞大应用进行拆分，并随之解耦，每个部分可以单独进行维护和部署，提升效率。
  2. 整合历史系统：在不少的业务中，或多或少会存在一些历史项目，这些项目大多以采用老框架类似（Backbone.js/Angular.js 1）的B端管理系统为主，介于日常运营，这些系统需要结合到新框架中来使用还不能抛弃，对此我们也没有理由浪费时间和精力重写旧的逻辑。而微前端可以将这些系统进行整合，在基本不修改逻辑的同时兼容新老两套系统并行运行。

1. 实现微前端有哪些方案

实现微前端的重要思想就是将应用进行拆解和整合，通常是一个父应用加上一些子应用。

|  |  |  |  |
| --- | --- | --- | --- |
| 方案 | 描述 | 优点 | 缺点 |
| Nginx路由转化 | 通过Nginx配置反向代理来实现不同路径映射到不同应用，例如[www.abc.com/app1](http://www.abc.com/app1)对应app1，[www.abc.com/app2](http://www.abc.com/app2)对应app2，这种方案本身并不属于前端层面的改造，更多的是运维的配置。 | 简单、快速、易配置 |  |
|  |  |  |  |
|  |  |  |  |

# 智力题

**1、25匹马，5匹一组跑，跑几次可以找到前三名**

答：7 次

第一次： 5组各自跑一次，找到每组的第一名；

第二次： 5个第一名跑一次，找到前三名，此时已经知道最终的第1名

第三次：第1名组的第2和第3、第2组的第1和第2，第3组的第1 ，跑一次；

就可以知道最终的第2名和第3名

所以是5+1+1=7次

**2、5点15分，时针和分针的夹角是多少度？**

答：12个小时，每个小时的占360/12=30度，5小时=5\*30=150度

分针走一圈是1小时，1小时=60分钟，则1分钟占360/60=6度，15分钟=15\*6=90度

150度和90度的夹角是60度；

但分针走到15分钟时时针也有移动，1小时等于60分钟， 那么15分钟等于15/60=0.25小时；

1小时等于30度，那么0.25小时等于30\*0.25=7.5度；

所以5点15分，时针和分针的夹角是67.5度。

**3、如果你有两个桶，一个装红颜料，另一个装蓝颜料，两个桶的颜料一样多。你从蓝颜料里舀一杯，倒入红色颜料桶，再从红色颜料桶里舀一杯倒入蓝色颜料桶，假设红桶中的红色/蓝色 为 A，蓝桶中的蓝色/红色 为 B，A 和 B 哪个大？**

一样大；

假设红色和蓝色桶里面的颜料都为100毫升

蓝色往红色桶里倒入10毫升，则红色桶中有100毫升红色和10毫升蓝色

再从红色桶倒入蓝色桶10毫升，此时设置这10毫升颜料里面有x毫升红色和y毫升蓝色，则有以下公式：

X+y=10

X/Y = 10/100 = 1/10

计算出X=10/11 ; y=100/11

则现在蓝色桶有90 + 10/11= 1000/11的蓝色 ， 有100/11的红色

红色桶有100-100/11=1000/11的红色，有10-10/11=100/11的蓝色

蓝色桶 1000/11和100/11的比

红色桶1000/11和100/11的比

# 算法题

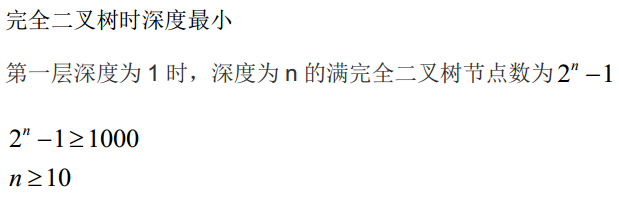
**1、一个二叉树有1000个节点，最小深度？**

答：最小深度为10

**2、一个树形结构，如果输入一个id找到他所在测children**

3、**二分查找算法**

二分查找也称折半查找，它是一种效率较高的查找方法。但是，折半查找要求线性表必须采用顺序存储结构，而且表中元素按关键字有序排列。



1. 爬楼梯、青蛙跳
2. 斐波那契数列

斐波那契数列（Fibonacci sequence），又称[黄金分割](https://baike.baidu.com/item/%E9%BB%84%E9%87%91%E5%88%86%E5%89%B2/115896)数列，因数学家莱昂纳多·斐波那契（Leonardoda Fibonacci）以兔子繁殖为例子而引入，故又称为“[兔子数列](https://baike.baidu.com/item/%E5%85%94%E5%AD%90%E6%95%B0%E5%88%97/6849441)”，指的是这样一个数列：0、1、1、2、3、5、8、13、21、34、……在数学上，斐波那契数列以如下被以递推的方法定义：*F*(0)=0，*F*(1)=1, *F*(n)=*F*(n - 1)+*F*(n - 2)（*n*≥ 2，*n*∈ N\*）在现代物理、准[晶体结构](https://baike.baidu.com/item/%E6%99%B6%E4%BD%93%E7%BB%93%E6%9E%84/10401467)、化学等领域，斐波纳契数列都有直接的应用，为此，美国数学会从 1963 年起出版了以《斐波纳契数列季刊》为名的一份数学杂志，用于专门刊载这方面的研究成果。

1. js 将金额每隔3加逗号 数字千分位

# 面试总结

面试可以问的问题：

**你们公司（团队）目前面临的最大挑战是什么？**

**什么新技术（编程语言）是你们未来希望采用的？**

众安：前后端开发 三面通过，薪资没谈拢

永辉 ：测试 一面挂，很惨

河姆渡：前端 两面通过，

掌门1对1：前端 1面挂

饿了吗：前端 1面挂

小红书：前端 1面挂

鱼泡泡/比心陪练： 1面挂

票易通：二面挂 2021-02-02~2021-02-09

众安

锅圈

酷家乐

唯链