

1. 从最近的项目开始说起吧，

最近是学校在实训，然后要求做一个实训设计，我和小组队员们选择做了一个无人停车场的停车管理系统，客户在小程序预定车位，扫码进场，然后就是后台管理系统；

然后再早期7月份的时候就是做了一个个人简历的静态网页，自己通过实现了swiper的翻页功能，然后其余的就是一些css的动画鸭什么的设计实现了。

然后就是在字节跳动训练营做的一个简易版今日头条搜索，是用react做的一个移动端app，然后就是大致与今日头条一样，有搜索框，有结果页面，详情界面，还有滑动加载用swiper去实现的，然后还有一些反向代理的思路的实现，节流防抖的应用，FCP白屏时间和LCP最大块内容时间的认识，然后还有一些对于路由按需加载的一些知识点，便于部署在服务器的时候的性能优化等等，

然后就是在绿灯公司实习参与的两个项目的开发，阳光课程和CIO协会的系统开发，都是小程序和后台管理系统，然后用的都是vue全家桶和小程序的知识。

知识点：节流，防抖，

防抖：即短时间内大量触发同一事件，只会执行一次函数，实现原理就是设置一个定时器，然后约定在xx毫秒后再触发事件处理，每次触发事件都会重新设置定时器。

    function debounce(func, wait) {

            var timeout = null;

            return function() {

                let context = this;

                let args = arguments;

                if (timeout) clearTimeout(timeout);

                timeout = setTimeout(() => {

                    func.apply(context, args)

                }, wait);

            }

        }

        function a() {

            console.log("1111111111111")

        }

        let b = debounce(a, 1000)

        b()

        b()

节流：防抖是延迟执行，而节流是限制执行。节流函数每个一段时间执行一次，实现原理为一个定时器，约定xx毫秒后执行事件，如果时间到了，执行函数并重置定时器，和防抖的区别在于，防抖每次触发事件都重置定时器，而节流在于定时器时间到之后再清除定时器。

    <script>

        function throttle(func, wait) {

            let timeout = null;

            return function() {

                let context = this

                let args = arguments

                if (!timeout) {

                    timeout = setTimeout(() => {

                        timeout = null

                        func.apply(context, args)

                    }, wait)

                }

            }

        }

        let b = throttle(() => {

            console.log("sdsds")

        }, 2000);

        b();

        b();

    </script>

知识点：FCP白屏时间和LCP最大块内容时间的认识

（讲讲为什么浏览器加载时为什么会出现白屏现象）：

这得从发送一次完整的http请求之后浏览器的解析过程：

浏览器先构建DOM,这个时候相应的css和js就会发起请求，然后在解析css时，构建DOM的过程是继续的不会停的，然后解析到js的时候，就会停止构建DOM，于是在render之前，也就是请求的时候，就会出现了这个白屏现象，可以通过插件lighthouse进行测试，然后也可以在开发者工具里面的performence里面进行查看。

1. 低版本支持新的语言特性：

ES6的话，可以用babel进行转换成ES5然后进行编译。

1. Js为什么是单线程？有什么好处？

Js的单线程和他的用途有关吧，作为脚本语言，它的用途是与用户互动，操作DOM等，这些就决定了它只能是单线程，假定js同时又两个线程，一个线程在DOM节点上添加内容，然后另一个线程删除这个节点，那么这个时候浏览器应该以哪个为准呢。。

Js在这个基础上面就衍生了一个事件循环机制的任务列队，任务列队可以分为同步仁务和异步任务，异步编程也就是在这个基础上进行的。

1. react和vue 的优劣：
2. template与jsx；
3. 虚拟DOM中的diff算法；
4. 双向绑定与setstate;
5. props传参的不同;
6. vue是怎么实现双向绑定的：

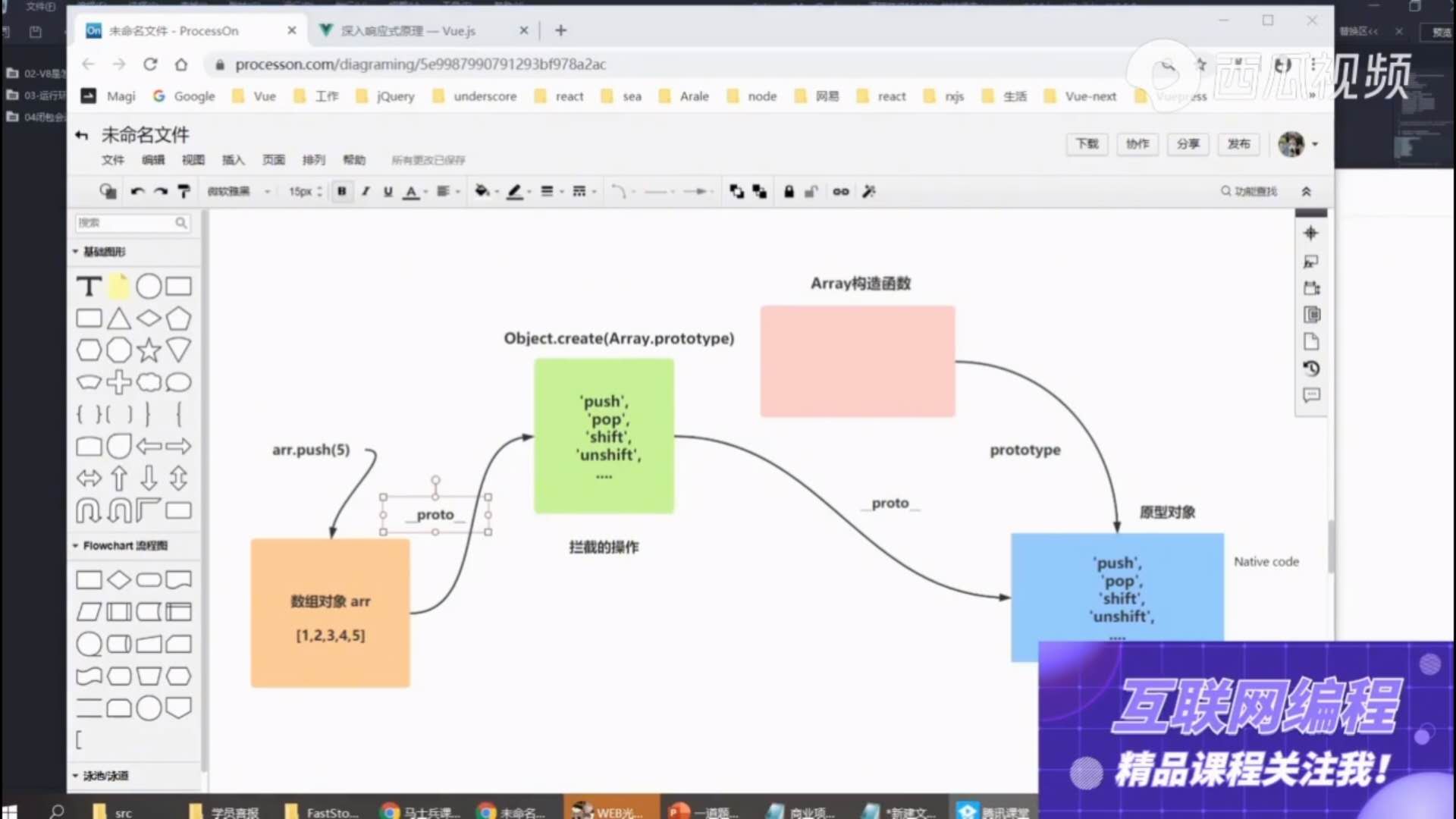
数据劫持Obect.defineProperty(),

Observer监听器，watcher订阅者，订阅者管理器，Compile解析器;

Vue2.0的数据劫持有个缺点就是Object.defineProperty没办法监听数据的内容变化，

但是可以通过原型链的方法去改进，

比如现在这里有个数组实例，然后我们知道数组构造函数Array.prototype指向我们的原型对象Array嘛，然后原型对象里面就有我们的push，pop等等的方法；



然后的话就是我们可以先用Object的create方法创建一个Object对象，然后把我的Object实例对象的\_\_proto\_\_指向我们的数组的原型对象，然后数组的实例对象的\_\_proto\_\_指向我们的Object对象，这样子就可以每次数组改变的时候，我们可以在object对象中监听push、pop的状态改变，然后再调用原型对象的方法进行push or pop这样子。

1. React单向数据流：数据主要是从父节点传递到子节点，通过props，如果父级某个props改变了，react会重新渲染所有的子节点。Props不可改变，（vue和小程序的可以改）

单向数据流：数据流动方向可以跟踪，流动单一

React从子组件传回给父组件数据该怎么做：父组件传递一个回调函数给子组件中，然后子组件调用这个函数，从而使父组件得到更新。

Vue的呢：$emit传入一个父组件方法，然后回调一下。

1. react向很深的一个地方传递数据，不用props还有什么方法：

Context:组件自己定义了一个context，然后子组件都可以访问到这个状态，不需要中间的组件传递，而这个组件的父组件是没有办法访问的。

        const ThemeContext = React.createContext('light');

        <ThemeContext.Provider value="dark">

        <Toolbar />

      </ThemeContext.Provider>

      class ThemedButton extends React.Component {

  // 指定 contextType 读取当前的 theme context。

  // React 会往上找到最近的 theme Provider，然后使用它的值。

  // 在这个例子中，当前的 theme 值为 “dark”。

  static contextType = ThemeContext;

  render() {

    return <Button theme={this.context} />;

  }

}

1. ES6中箭头函数的理解：

()=>{}箭头函数

左边 一个参数时可以省略括号，没有参数或者多个参数不能省略，

右边 返回一个对象时需要加（）；只有一句语句时不用{}

pairs = evens.map(v => ({ even: v, odd: v + 1 }))

没有自己的 this arguments super new.target, 不能用作构造函数，没有 prototype属性，不能用作生成器

箭头函数没有自己的 this 指针， call() apply() bind() 调用的时候， **第一个参数会被忽略**。

1. 跨域问题：

解决方式：

1. 通过proxy配置本地代理；
2. Jsonp;
3. CORS;

 Access-Control-Allow-Credential: true,

            Access-Control-Allow-Headers: origin, content - type, accept, x - request - with,

            Access-Control-Allow-Methods: GET, POST, PUT, DELETE, OPTIONS,

            Access-Control-Allow-Origin: http: localhost: 8000

1. 三次握手 && url到显示的过程：

三次握手：

四次挥手：

Url到显示的过程：

1. -（5）DNS解析的过程
2. 、输入url，然后浏览器进行DNS解析：浏览器会先看本地硬盘host文件，看看其中有没有和这个域名对应的规则， 有的话直接使用hosts文件里面的ip地址，如果没有的话，就会发出一个DNS请求到本地DNS服务器(服务商)，
3. 本地DNS服务器(服务商)：接收到请求后，本地查看这个ip有没有缓存，如果有缓存，那么就可以直接返回结果，此过程是递归查询；如果没有，则向本地DNS服务器(服务商)说去DNS根服务器进行查询。
4. DNS根服务器：没有记录具体的域名和ip地址的对应关系，则向本地DNS服务器(服务商)说去域服务器进行请求，迭代的过程。
5. 域服务器：查看请求的对象是.com（假设），那么就会返回所对应的可以解析这个域名的地址，
6. 最后本地DNS服务器(服务商)再向解析服务器发出请求，然后本地DNS服务器(服务商)就会缓存这个ip与域名的关系。
7. 浏览器向web服务器发送一个http请求：（三次挥手建立tcp连接）

三次挥手：