shadowsocks wwpien vps framework

git bash里跳到最前面和最后面 ctrl+a/ctrl+e

如果一个包是给用户使用的就需要加 -D 给开发人员使用的就不需要

getBoundingClientRect() 可以获取元素的宽高和top left

this.$children 只能获取到子组件不能获取到子元素

**parcel:** 简洁版webpack npm i -D parcel-bundle

运行 .\\node\_modules\\.bin\\parcel index.html --no-cache (--no-cache)不使用缓存, 若不加index.html node会自动去找index.js文件, 则显示不出html

parcel打包index.js npx parcel build index.js --no-cache --no-minify

display: flex;有个属性为order可控制布局显示的顺序

子元素属性: flex-basis: 比width的优先级大

flex-grow: 1; 为元素分配父元素剩余的空间, 如设置了宽度则不分配

flex-shrink: 0; 不随着父元素的缩小而缩减自己的空间

**除去第一个子元素**



for in 用于遍历对象, 遍历数组时为下标

for of 用于遍历数组

**validator 校验器**, 可校验父元素传过来的值是否符合要求



**发布npm包**

在项目中运行npm adduser 输入账号密码 lijuancheng 1229432979

npm publish发布包时出现403错误no\_perms Private mode enable, only admin can publish this module:

或者使用yarn publish出现couldn't publish package:"https://registry.npm.taobao.org/包名:unauthorized"错误

出现原因：使用的是淘宝源cnpm,登陆到的是cnpm

解决方法：切换到npmjs的网址，代码如下npm config set registry http://registry.npmjs.org/

使用打包后的文件, package.json里面main改为dist/index.js且使用改npm包时需引入css文件

本地编写好轮子直接在本地测试, 在本地轮子运行npm link 在测试项目里运行npm link 轮子名

scoped属性的用法: 加上后每个组件元素会加上一个唯一标识[data-v-xxx] 以此区别其他相同的变量名

name的作用: 在vue tools用于调试

scoped: 有范围的, 在vue的style里加上后会在每个样式后加个唯一id

在svg中加入fill属性科修改icon的颜色

word-wrap: break-word 允许长单词换行到下一行

word-break: break-all 允许在单词内换行

keep-all 只能在半角空格或连字符处换行

如果父元素高度为min-height 子元素的高度100%不生效

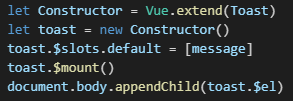
**input实现v-model双向绑定---语法糖**

给组件添加 v-model 属性时，默认会把 value 作为组件的属性，然后把 'input' 值作为给组件绑定事件时的事件名

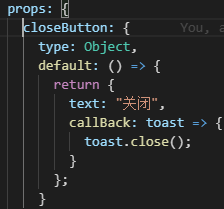


**defineProperty**

**动态创建vue实例**



如果props接收的类型为复杂类型需要用函数返回default值, 因为复杂类型是引用类型, 只会浅拷贝, 第一个接收的值会影响后面组件的值



**$nextTick的使用**

$nextTick 是在下次 DOM 更新循环结束之后执行延迟回调，在修改数据之后使用 $nextTick，则可以在回调中获取更新后的 DOM

**$emit和$on**

子组件用$emit传递一个方法



父组件$on监听该方法, close传递过来后执行后面的回调函数



**sync修饰符**

子组件变更父组件推荐以 update:myPropName 的模式触发事件

在子组件中用this.$emit('update:title', newTitle) 把值传给父组件

父组件可以监听那个事件并根据需要更新一个本地的数据 property



即 .sync 修饰符, 在父组件使用子组件时, 在子组件上需要更新的字段后加



用一个对象同时设置多个 prop 的时候，也可以将这个 .sync 修饰符和 v-bind 配合使用

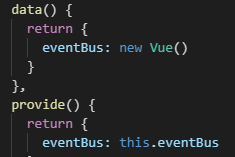
这样会把 doc 对象中的每一个 property (如 title) 都作为一个独立的 prop 传进去，然后各自添加用于更新的 v-on 监听器



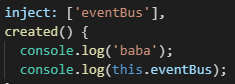
**provide和inject**

允许一个祖先组件向其所有子孙后代注入一个依赖

在祖先组件中注入一个eventbus



在后辈组件中注入inject, 即可拿到祖先传过来的provide里的值



用eventBus.$emit触发修改该属性值



用eventBus.$on监听, 回调拿到修改后的值



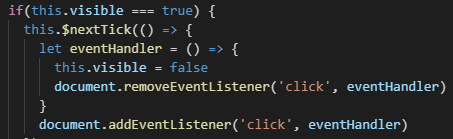
**监听document的点击事件**

1 点击页面关闭popover, 给document设置点击事件

2 点击事件添加在this.$nextTick里, 不然会冒泡, 点击按钮也会触发document的click

3 事件执行完后需要移除监听器, 不然会生成多个监听器同时监听

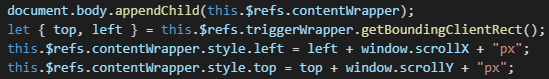
4 移除事件需要函数名, 用bind会产生一个新的函数, 所以需把函数提出来



**popover的几个bug**

1 若在popover外的元素加上了overflow: hidden; 内容会被遮盖, 应把内容元素添加到body中

2 内容显示在按钮上方, 把按钮的top和left赋值给内容要加上滚动的长度, 不然页面滚动后内容显示位置不正确



3 点击页面关闭应设置document的监听器, 因为body有时的高度没有页面高度高

4 设置监听器, 回调函数为事件监听执行的fn, 执行完fn后把监听器移除, 不然点击多次会产生多个监听器

5 如果点击按钮立刻设置监听器会产生冒泡, 一生成监听器就会自动关闭, 用setTimeout延迟设置监听器

6 轮子中并不能阻止冒泡, 不然组件外设置的点击事件不嫩执行, 所以应该用事件对象判断点击的是哪个元素, 点击按钮和页面关闭内容区域, 点击内容本身不关闭



7 点击按钮关闭时, 会关闭两次, 一次自身的关闭, 一次冒泡到document监听器的关闭, 所以当点击按钮, visible变为false后把页面监听器移除

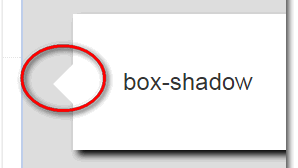
**box-shadow和filter: drop-shadow的区别**

box-shadow是盒子外的阴影

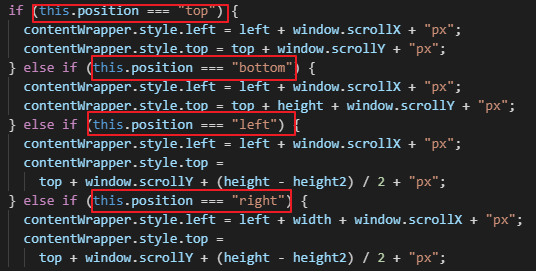
filter: drop-shadow贴近真实世界的阴影, 可以穿透盒子内的透明区域

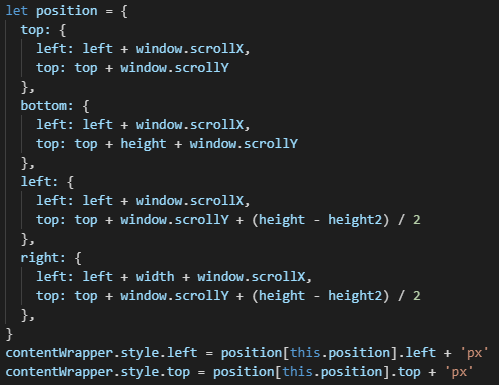
实际场景: 盒子外的三角形也需要阴影时使用

若中心的文字不需要阴影, 加上background: #fff;即可



**表驱动编程**





**设计模式(写代码的套路)**

1 发布订阅模式 emit/on/off

2 单向数据流

3 正交(props)

4 可测试的代码

5 不要让人思考

6 面向离职写代码

a 文档化

b 被观察感 ->严格要求

c 团队中表现优异

**vuepress**

<https://www.vuepress.cn/guide/getting-started.html>

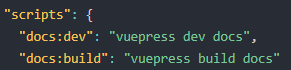
1 npm install -D vuepress

2 mkdir docs

3 新建一个 markdown 文件

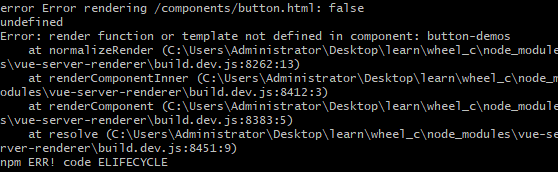
echo '# Hello VuePress!' > docs/README.md

4 在 package.json 里加一些脚本

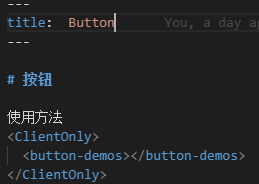


5 npm run docs:dev

6 若npm run docs:build 报错



则在components的组件md文件里用clientOnly标签把组件包起来



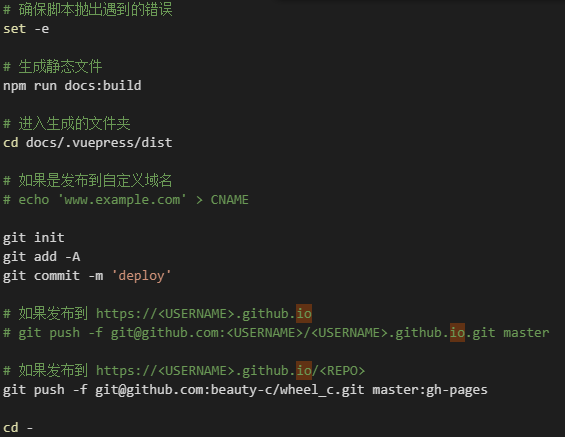
6 部署到github pages中

在config.js中加入base, 参数为项目名



在项目根目录创建deploy.sh文件

把最后一行的username和<REPO>改为自己的github名和项目名



7 运行./deploy.sh后即可在项目中settings看到自己的官网

