1.vue 父传子，子传父emit

父传子：props[] 接受
    子组件在props中创建一个属性，用以接收父组件传过来的值
    父组件中注册子组件
    在子组件标签中添加子组件props中创建的属性
    把需要传给子组件的值赋给该属性
子传父：this.$emit("自定义事件" ， "数据")
    子组件中需要以某种方式例如点击事件的方法来触发一个自定义事件
    将需要传的值作为$emit的第二个参数，该值将作为实参传给响应自定义事件的方法
    在父组件中注册子组件并在子组件标签上绑定对自定义事件的监听
兄弟传值：
    1）子传父 父传子
    2）通过eventBus传递数据
    3）在标签中使用<router-link>标签

```
this.$router.push({
    name:'routePage',
    query/params:{routerParams:params}
})
```

需要注意的是，实用params去传值的时候，在页面刷新时，参数会消失，用query则不会有这个问题。
这样使用起来很方便，但url会变得很长，而且如果不是使用路由跳转的界面无法使用。
**在通信中，无论是子组件向父组件传值还是父组件向子组件传值，他们都有一个共同点就是有中间介质，子向父的介质是自定义事件，父向子的介质是props中的属性。抓准这两点对于父子通信就好理解了

2.vue 深度监听watch

https://www.cnblogs.com/yesu/p/9546458.html
watch:{} 对象，可监听数据，数据发生变化， 处理函数
目的： watch虽可监听，但只是浅监听，只监听数据第一层或者第二层，
何为第二层?

```
let obj = {name: 'xx', child: {age: 11}};
```

child之后的值就为第二层或者深层
实现目标： 如果 要监听一个对象中的属性，属性最高也是第二层了，watch可能监听不到，
所有要使用深度监听，代码如下：

```
watch: {
    firstName : {
        handler:function() {    //特别注意，不能用箭头函数，箭头函数，this指向全局
            处理函数
        },
        // 代表在wacth里声明了firstName这个方法之后立即先去执行handler方法
        immediate: true,
        deep: true    //深度监听
    }
}
```

3.vue 路由传参，params和query哪个显示在地址栏上

1、用法
    A、query要用path来引入,接收参数都是this.$route.query.name。
    B、params要用name来引入，接收参数都是this.$route.params.name。
2、效果
    A、query类似于ajax中get传参，即在浏览器地址栏中显示参数。
    B、params则类似于post，即在浏览器地址栏中不显示参数。
3、个人建议
    在路由传参上建议使用params，以隐藏参数，做好安全保密。
不过如果使用params传值的话，页面一刷新params 的值就消失了。
https://www.cnblogs.com/lulianlian/p/7682790.html

4.斐波那契数列（递归，非递归）

```
//斐波那契数列(递归方法)
function fn(n){
    if(n==1|n==2){
        return 1;
    }
    //因为斐波那契数列格式为：1、1、2、3、5、8、13、21、34、.....,n=1和n=2的时候都是输出1
    return fn(n-1)+fn(n-2);
    //不断调用自身函数，n-1是穿进去的参数的前一次，就是最后n的前一个数字。所以n-2是最后传入参数的前两个数字。
}
//斐波那契数列(非递归方法)
function fib2(n){
    if(n==1 || n==2){
        return 1;
    }else{
        var arr =[];//用一个数组作为辅助的空间
        arr[0]=1, arr[1]=1;
        for(var i=2;i<n;i++){
            var temp = arr[0]+arr[1];
            arr[1] =arr[0];
            arr[0] = temp;
        }
        return arr[0];
    }
}
```

5.webpack打包配置项（https://blog.csdn.net/qq_36581955/article/details/80393563）

1、在工程目录下建个webpack.config.js
2、Webpack.config.js配置文件内容，如下：

```
const path = require('path');
const webpack = require('webpack');
module.exports = {
    //页面入口文件配置
    entry: './app.js',
    //入口文件输出配置
    output: {
        path:__dirname+'/build',
        filename:'[name]-[chunkhash].js'
    },
    resolve: {
        //自动扩展文件后缀名，意味着我们require模块可以省略不写后缀名
        extensions: ['.js','.json']
    },
    target: 'node',
    context: __dirname,
    node: {
        __filename: true,
        __dirname: true
    },
    //加载器配置
    module: {
        rules: [
            {
                test: /\.js$/,
                loader: 'babel-loader',
                query: {
                    presets: ['es2015','stage-0']
                },
                exclude:/node_modules/
            }
        ]
    },
    plugins: [
        //newwebpack.optimize.UglifyJsPlugin()
    ]
};
```

3、在当前项目目录下，用命令行工具输入webpack，进行打包。（命令webpack）

4、打包完成后，当前工程目录下的build文件夹里会多出一个js文件，在build目录下使用命令行，输入 node + js文件名，只有光标显示就是启动成功了！

6.Promise函数（https://www.jb51.net/article/111741.htm）

```
function fun1(res){
    return new Promise((resolve, reject)=>{
        //函数，代码，接口都行
        tyr{
            resolve('返回成功数据')
        }.catch(){
            reject('返回错误信息')
        }
    })
}
//fun2 , fun3 同fun1

//使用promise
   Promise.resolve().then((res=>{
       return fun1
   })).then((res=>{
       return fun2
   })).then((res=>{
       return fun3
   })).catch(error=>{
       console.log('-----错误信息----')
    })
    或:
    Promise.resolve().then(fun1).then(fun2).then(fun3)
Promise.all([fun1,fun2,fun3]).then((res)=>{
    ....当fun1 fun2 fun3执行完之后才执行此时代码
}).catch((error=>{...报错执行}))
```

7.js继承（https://www.jianshu.com/p/b76ddb68df0e）

1.属性拷贝（就是将对象的成员复制一份给需要继承的对象）
2.原型式继承（借用构造函数的原型对象实现继承）
3.原型链继承（即 子构造函数.prototype = new 父构造函数()）
4.借用构造函数（使用call和apply借用其他构造函数的成员，可以解决给父构造函数传递参数的问题，但是获取不到父构造函数原型上的成员.也不存在共享问题）
5.混合继承（借用构造函数 + 原型式继承）
6.es6 class类用extend继承

8.基本数据类型和引用数据类型区别

1.基本数据类型是单独操作（独用一块内存来存储，修改其中一个，另一个不会受影响）
    Number、String 、Boolean、Null和Undefined，这五种基本数据类型可以直接访问，他们是按照值进行分配的，存放在栈（stack）内存中的简单数据段，数据大小确定，内存空间大小可以分配。
2.引用数据类型是公用一块区域（公用一块内存来存储，修改其中一个，就会全部修改）
    也就是对象类型Object type，比如：Object 、Array 、Function 、Data等。即存放在堆（heap）内存中的对象，变量实际保存的是个指针，这个指针指向另一个位置。
3.什么是堆栈？在计算机领域中，堆栈是两种数据结构，它们只能在一端（层位栈顶top），对数据进行插入和删除。
    堆：队列先进先出；由操作系统自动分配释放，存放函数的参数值，局部变量的值等，其操作方式类似于数据结构中的栈。
    栈：先进后出；动态分配的空间一般由程序员分配释放，若程序员不释放，程序结束时可能由OS收回，分配方式倒是类似于链表。
***：简单数据类型存储在栈中，复杂数据类型存储在堆中，但是它的引用存储在栈中。

9.数组常用方法（https://blog.csdn.net/qq_35652217/article/details/78470118）

push：向数组的末尾增加一项 返回值是数组的新长度
unshift：向数组开头增加一项 返回值是数组的新长度
pop:删除数组的末尾项 返回值是删除的数组项
shift:删除数组开头项 返回被删除的开头项目
splice：删除数组中的任意项 返回值是被删除的数组项
slice:复制数组 返回值是复制到的新数组 写上数值之后 不包含被复制的最后一项
Array.length：返回或设置一个数组中的元素个数
Array.from() :对伪数组或可迭代对象(包括arguments Array,Map,Set,String…)转换成数组对象
Array.isArray()：用于确定传递的值是否是一个 Array
Array.of()：设置数组元素 Array.of(7) //[7]
copyWithin(target, start, end):浅复制数组的一部分到同一数组中的另一个位置
every(callback):方法测试数组的所有元素是否都通过了指定函数的测试
fill():用一个固定值填充一个数组中从起始索引到终止索引内的全部元素
filter():创建一个新数组，其包含通过所提供函数实现的测试的所有元素
find():返回数组中满足提供的测试函数的第一个元素的值
findIndex():返回数组中满足提供的测试函数的第一个元素的索引
includes():用来判断一个数组是否包含一个指定的值，如果是，酌情返回 true或 false
reduce():累加器和数组中的每个元素（从左到右）应用一个函数
reduceRight():接受一个函数作为累加器（accumulator）和数组的每个值（从右到左）将其减少为单个值
some():测试数组中的某些元素是否通过由提供的函数实现的测试。
toLocaleString():返回一个字符串表示数组中的元素。数组中的元素将使用各自的 toLocaleString 方法转成字符串，这些字符串将使用一个特定语言环境的字符串（例如一个逗号 “,”）隔开
toString():返回一个字符串，表示指定的数组及其元素
拼接：
concat:把一个数组和另一个数组拼接在一起 返回拼接好的数组
join:把数组中的每一项 按照指定的分隔符拼接成字符串

排序：
reverse:倒序数组 返回值倒序数组 原有数组改变
sort:根据匿名函数进行冒泡排序 b-a倒序 a-b升序

兼容性不好：
indexOf:返回获取项在数组中的索引
lastIndexOf:返回获取项在数组中出现的最后一次索引
forEach: 循环遍历数组 参数是一个匿名函数 默认返回为undefined
map：循环遍历数组 参数是一个匿名函数

10.创建，删除，移动，复制，查找节点

1）创建新节点
createDocumentFragment() //创建一个DOM片段
createElement_x() //创建一个具体的元素
createTextNode() //创建一个文本节点
2）添加、移除、替换、插入
appendChild() //添加
removeChild() //移除
replaceChild() //替换
insertBefore() //插入
3）查找
getElementsByTagName_r() //通过标签名称
getElementsByName() //通过元素的Name属性的值
getElementById() //通过元素Id，唯一性

11.html5新增属性和标签，h5语义化（https://blog.csdn.net/youpeng505/article/details/80621579）

1.新增属性：placehoder、required、pattern、min和max、step、height和width、autofocus、multiple等
2.新增标签：hrader、footer、nav、section、article、aside、detailes、summary、dialog、datalist、keygen、output、audio、video、canvas、figcaption、figure、mark、main、time等
3.增强表单类型：color、date、datetime、time、datetime-local、email、month、number、range、search、tel、url、week
3.语义元素清楚地向浏览器和开发者描述其意义，优点如下：
1）易于用户阅读，样式丢失的时候能让页面呈现清晰的结构。
2）有利于SEO，搜索引擎根据标签来确定上下文和各个关键字的权重。
3）方便其他设备解析，如盲人阅读器根据语义渲染网页
4）有利于开发和维护，语义化更具可读性，代码更好维护，与CSS3关系更和谐。

12.事件委托，事件冒泡原理（https://www.cnblogs.com/alsy/p/4915912.html）

事件委托：其实是使用了冒泡的原理，从点击的元素开始，递归方式的向父元素传播事件，这样做的好处是对于大量要处理的元素，不必为每个元素都绑定事件，只需要在他们的父元素上绑定一次即可，提高性能。 还有一个好处就是可以处理动态插入dom中的元素，直接绑定的方式是不行的。
事件冒泡：所谓的事件冒泡就是子级元素的某个事件被触发，它的上级元素的该事件也被递归执行
阻止事件冒泡：
    (http://caibaojian.com/javascript-stoppropagation-preventdefault.html)

```
1.window.event? window.event.cancelBubble = true : e.stopPropagation();
2.function stop(event){
    var event = event || window.event;
```

```
        if(event || event.stopPropagation){
            event.stopPropagation()
        }else{
            event.cancelBubble = true
        }
    }
```

13.元素垂直居中（https://www.cnblogs.com/hutuzhu/p/4450850.html）

```
方法一：table-cell
    display: table-cell;
    vertical-align: middle;
    text-align: center;
方法二：display:flex
    display: flex;
    justify-content:center;
    align-items:Center;
方法三：绝对定位和负边距
    position: absolute;
    width:100px;
    height: 50px;
    top:50%;
    left:50%;
    margin-left:-50px;
    margin-top:-25px;
    text-align: center;
方法四：绝对定位和0
    margin: auto;
    position: absolute;
    top: 0; left: 0; bottom: 0; right: 0;
方法五：translate
    position: absolute;
    top:50%;
    left:50%;
    width:100%;
    transform:translate(-50%,-50%);
    text-align: center;
方法六：...打开连接看
```

14.rem布局，rem和em的区别

```
rem(font size of the root element)是指相对于根元素字体大小的单位。简单的说它就是一个相对单位。
em(fontsize of the element)是指相对于自己的字体大小的单位。
区别：rem计算的规则是依赖于根元素，em是依赖于自己字体大小计算。
rem布局原理：先按定高宽设计出来页面，然后转换为rem单位，配合js查询屏幕大小来改变html的font-size，最终做出所谓的完美自适应。
```

15.标签选择器有哪几种，以及权重（https://blog.csdn.net/m0_37285193/article/details/81567522）

```
标签选择器、类选择器、id选择器、通配符选择器、分组选择器、后代选择器、伪类选择器、子选择器、属性选择器、相邻兄弟选择器、兄弟选择器、指定选择器、交集选择器
权重：!important > 行内样式 > id > 类 > 标签 > 继承样式 > 默认样式
```

16.数组去重方法

```
1.Array.from(new Set([1,2,2,3,4,4,5]));
2.let arr=[1,2,3,3,4,4,5,6,6];
  let temp=[]
  for(let i=0;i<arr.length;i++){
      if(temp.indexOf(arr[i]) == -1){
          temp.push(arr[i])
      }
  }  等
```

17.深拷贝，浅拷贝

```
1、基础类型：像Number、String、Boolean等这种为基本类型
2、引用类型：Object和Array

浅拷贝只是复制了对象的引用地址，两个对象指向同一个内存地址，所以修改其中任意的值，另一个值都会随之变化，这就是浅拷贝（例：assign()）

深拷贝是将对象及值复制过来，两个对象修改其中任意的值另一个值不会改变，这就是深拷贝（例：JSON.parse()和JSON.stringify()，但是此方法无法复制函数类型）
当你需要深拷贝对象中的方法时是可以用lodash.js(提高JS原生方法性能的JS库)中的cloneDeep()方法
 <script type="text/javascript">
       var objA = { "name": "戈德斯文" };
       var objB =lodash.cloneDeep(objA);
 </script>
```

18.盒模型:标准盒模型和怪异盒模型的区别（https://www.cnblogs.com/yky-iris/p/7719895.html）

```
在标准模式下，一个块的总宽度= width + margin(左右) + padding(左右) + border(左右)
在IE盒模型下：一个块的总宽度= width + margin(左右)（即width已经包含了padding和border值）
```

19.如何提高页面加载速度？（https://blog.csdn.net/qiqi_77_/article/details/79423111）

```
1.减少Http的请求（合并脚本 和样式表、字体图标、雪碧图、图片地图）
2.使用CDN
3.使用缓存Expires和Cache-Control
4.对HTTP传输进行压缩
5.样式表放头部
6.脚本放底部
7.合理使用外部js/css
8.减少DNS查找
9.懒加载
10.预加载
11.按需加载
```

20.浏览器页面加载页面的过程(https://blog.csdn.net/qq_39795538/article/details/82764250
https://blog.csdn.net/qq_22313585/article/details/78926141)

```
浏览器根据DNS服务器得到域名的IP地址
向这个IP的机器发送http请求
服务器收到、处理并返回请求
浏览器得到返回的内容

dom（html）文档加载步骤：
    解析html结构
    加载外脚本和样式表
    解析并执行脚本代码
    dom树构建完成 //DOMContentLoaded
    加载图片视频音频等文件
    页面加载完毕 //load
```

21.this指向，改变this指向的方法

```
this指向问题分为四种
1.函数调用模式中this指向window。如题中obj.getX()返回的是一个立即执行函数，而这个函数的调用者就是window
2.方法调用模式中this指向调用对象。如题中的obj.getY()
3.构造函数调用模式，this指向new出来的对象,也就是实例对象。
4.上下文调用模式，可以指定this的作用域。当第一个参数是null或者undefined时，this指向window

改变this指向call()，apply()，bind()，存储this指向到变量中
```

22.解释跨域，解决跨域的方法（https://www.cnblogs.com/2050/p/3191744.html）

跨域：不同源则跨域。同源策略是浏览器的一种安全策略，指不在http(协议)、域名、端口均相同。
解决方法：

1.通过jsonp跨域
2.通过CORS跨域
3.通过修改document.domain来跨子域
4.使用window.name来进行跨域
5.使用HTML5中新引进的window.postMessage方法来跨域传送数据

23.http常用状态码（http://tool.oschina.net/commons?type=5）

1XX 信息性状态码（Informational） 服务器正在处理请求
2XX 成功状态码（Success） 请求已正常处理完毕
3XX 重定向状态码（Redirection） 需要进行额外操作以完成请求
4XX 客户端错误状态码（Client Error） 客户端原因导致服务器无法处理请求
5XX 服务器错误状态码（Server Error） 服务器原因导致处理请求出错

24.数组和对象的常用方法

数组方法（9小题）
对象方法：（https://blog.csdn.net/zhanglir333/article/details/78714598）
Object.assign(target, …sources):把任意多个的源对象自身的可枚举属性拷贝（浅拷贝）给目标对象，然后返回目标对象。
Object.freeze(a);//此时不允许添加或修改a的任何属性
Object.is(a, b);//用于判断两个值是否相同
//注意，该函数与==运算符不同，不会强制转换任何类型，应该更加类似于===，但值得注意的是它会将+0和-0视作不同值
Object.keys(a);//用于返回对象可枚举的属性和方法的名称
Object.defineProperties(obj, props)：在一个对象上添加或修改一个或者多个自有属性，并返回该对象。
Object.prototype.hasOwnProperty():返回一个布尔值，表示某个对象是否含有指定的属性，而且此属性非原型链继承。
Object.prototype.isPrototypeOf():返回一个布尔值，表示指定的对象是否在本对象的原型链中。
Object.prototype.propertyIsEnumerable():判断指定属性是否可枚举。
Object.prototype.toString():返回对象的字符串表示。
Object.prototype.watch():给对象的某个属性增加监听。
Object.prototype.unwatch():移除对象某个属性的监听。
Object.prototype.valueOf():返回指定对象的原始值。
Object.create(proto,[propertiesobject]):创建一个拥有指定原型和若干个指定属性的对象。