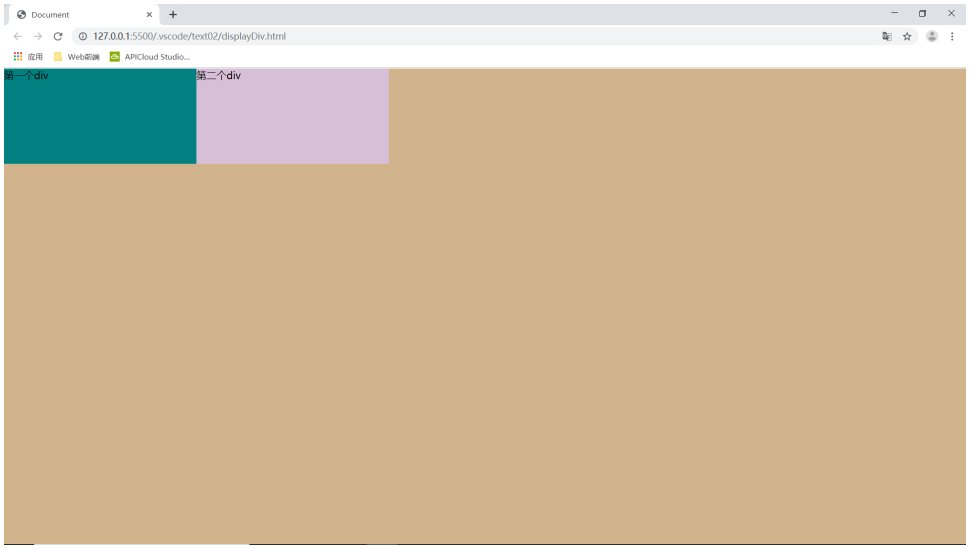
css基础学习

1.弹性布局(线性布局)

```
<div style="background: tan;width: 100%;height: 100%;  display: flex;flex-direction: row; ">
        <div style="background: teal;width: 20%;height: 20%;">
            <span>第一个div</span>
        </div>
        <div style="background:thistle;width: 20%;height: 20%;">
                <span>第二个div</span>
        </div>
    </div>
```



1.display:flex----->设置为弹性布局(线性布局)

flex-direction:row----->设置为水平对齐

flex-direction:column----->设置为垂直对齐

2.设置称重------>同Android

flex:1----->设置称重比为1(一般设置在子类)

3.一个剧中的小功能------>一般实现垂直剧中

当我们在父容器中加入了<弹性布局(线性布局)>-------->在子类中加入 margin: auto;他就会自动水平/垂直剧中

剧中我们还可能使用这些方法:

```
1. position: relative; top: 40%;left: 40%: 决定地址剧中
```

> 2. align-items: center:一般只能水平剧中
> 3. text-align: center：文本或内容剧中(也是水平)

2.网页与手机端的适配

```
<style>
    *{
        margin: 0;
        padding: 0;
    }
    html{
        width: 100%;
        height: 100%;
    }
    body{
        width: 100%;
        height: 100%;
    }
</style>
```

> 加上此代码才能在页面中使用100%比才能更好的适配与手机页面.

javascript基础学习

1.使用XMLHttpRequest原生访问网络数据

- get请求

```
    var xmlHttp = new XMLHttpRequest();
     xmlHttp.open("get","https://v1.hitokoto.cn/",false);
     xmlHttp.send();
     var obj = JSON.parse(xmlHttp.responseText);
     console.log(obj.id);
```

- post请求

```
   var xmlHttp = new XMLHttpRequest();
  xmlHttp.open("post","http://192.168.1.103:8088/transportservice/action/GetCarAccountBalance.do",false);
        xmlHttp.send(JSON.stringify({"CarId":1, "UserName":"user1"}));
        var obj = JSON.parse(xmlHttp.responseText);
         console.log(obj);
```

2.使用Jqury的ajax框架获取网络数据

```
    for (let index = 1; index < 5; index++) {
            //调用框架
              $.ajax({
                  //调用类型
                type:"post",
                  //获取的url

url:"http://192.168.1.103:8088/transportservice/action/GetCarAccountBalance.do ",
                  //发送数据类型
                dataType:'json',
                  //返回数据类型
                contentType:"application/json",
                  //发送的参数
                data:JSON.stringify({"CarId":index,"UserName":"user1"}),
                  //是否异步
                async:false,
                  //返回的回调:res返回结果对象
                success:function(res){
                    //如果使用:res.属性名
                    //打印toSting
                    console.log(index+JSON.stringify(res))
                },
                  //访问错误的回调
                error:function(){
                     console.log("失败")
                }
            });

        }
```

3.js与安卓的交互

准备环境

- android

准备一个布局

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".WebActivity">

    <WebView
        android:id="@+id/web_wv_view"
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent"/>

</LinearLayout>
```

java文件设置一些布局

```

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.webkit.JavascriptInterface;
import android.webkit.WebView;
import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class WebActivity extends AppCompatActivity {

    private WebView web_wv_view;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web);
        initView();
        setView();
    }
    //实例化
    private void setView() {
        Intent intent = getIntent();
        if (intent != null) {
            //获取相关信息
            String webName = intent.getStringExtra("webView");  //页面名字
            String pIndex = intent.getStringExtra("pView");     //题目id
            web_wv_view.loadUrl("file:///android_asset/" + webName); //加载网页
            web_wv_view.getSettings().setJavaScriptEnabled(true); //设置可以使用javaScript
            web_wv_view.addJavascriptInterface(new webJs(),"webJs"); //添加一个关联的接口
            Toast.makeText(getApplicationContext(), "" + pIndex, Toast.LENGTH_SHORT).show();
        } else {
            Log.d("WebView", "没有找到页面");
        }

    }

    private void initView() {

```

```
        web_wv_view = (WebView) findViewById(R.id.web_wv_view);
    }
    //准备一个与js共调的接口
    class webJs {
        //准备了一个方法(退出页面)
        @JavascriptInterface
       public void closeView() {
            finish();
        }
    }

}

```

1.Android调用js中的方法

- 在js里准备一个方法并且返回一个字符串(webView端)

```
  function tost(){
            alert("调用了tost方法");
            $("#txt").text("调用了tost方法");
            return "从tost返回的值";
        }
```

- 在Android调用方法(android端)

```
//一定已经加载过页面了而且（web_wv_view.getSettings().setJavaScriptEnabled(true);）允许使用javaScript
//方法一
// 第一种方法(通用)：tost()是js里的方法
mWebWvView.loadUrl("javascript:tost()");
//方法二:可以返回值
webView.evaluateJavascript("javascript:tost()", new ValueCallback<String>() {
                    @Override
                    public void onReceiveValue(String value) {
                        //value是调用方法返回的值
                        Toast.makeText(getApplicationContext(),""+value,   Toast.LENGTH_SHORT).show();
                    }
                });
```

- Android中拦截js里的弹窗

```
 //拦截js里的弹窗
 web_wv_view.setWebChromeClient(new WebChromeClient(){
     @Override
    public boolean onJsAlert(WebView view, String url, String message,
JsResult result) {
       AlertDialog alertDialog = new
AlertDialog.Builder(WebActivity.this)
               .setTitle("提示")
              .setMessage(message)      //提示内容
                .show();
        return true;
         }
    });
```

2.js调用安卓里的方法

- Android中准备一个js接口并且准备一个方法(退出页面)

```
//设置接口
web_wv_view.addJavascriptInterface(new webJs(), "webJs");
//准备接口
    class webJs {
        @JavascriptInterface
        public void closeView() {
            finish();
        }
    }
```

- js中调用:javaScript:window.webJs.closeView()

```
    <div>
        <!--javaScript:window:固定格式
        webJs:接口类
        closeView():方法
        -->
        <<span onclick="javaScript:window.webJs.closeView()"  id="txt">
文本</span>
    </div>
```

4.使用Jquri切换页面

```
<body>
    <div class="div_bt">
        <!--准备一些跳转的按钮-->
        <button onclick="bt1()">饼图</button>
        <button onclick="bt2()">折线</button>
        <button onclick="bt3()">柱形</button>
```

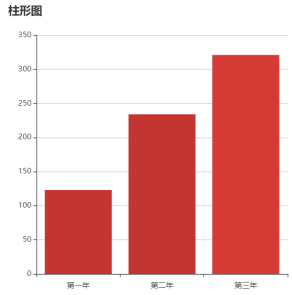
```
        <button onclick="bt4()">曲线</button>
        <button onclick="bt5()">横向柱形</button>
        <button onclick="bt6()">双折折线</button>
    </div>
    <!--准备的一个区域-->
    <div id="map" style="width: 500px;height: 500px;"></div>
    <script>
        <!--调用此方法
            $('#准备区域的id')
            .load('显示的html')
            -->
        function bt1(){
            $('#map').load("pieChart.html");
        }
        function bt2(){
            $('#map').load("lineChart.html");
        }
        function bt3(){
            $('#map').load("barChart.html");
        }
        function bt4(){
            $('#map').load("lineChartQ.html");
        }
        function bt5(){
            $('#map').load("barChartH.html");
        }
        function bt6(){
            $('#map').load("lineChart2.html");
        }
    </script>
</body>
```

5.使用ECharst

1.柱形图

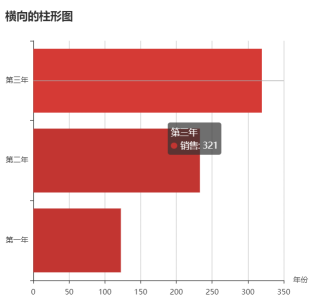
```
    var myChart = echarts.init(document.getElementById("map"));
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption({
            title:{
                text:'柱形图'
            },
            legend:{
                data:['第一年','第二年','第三年']
            },
            xAxis:{
                data:['第一年','第二年','第三年']
            },
            yAxis:{},
            series:[{
```

```
                name:'销量',
                type:'bar',
                data:[123, 234, 321]
            }]
        });
```



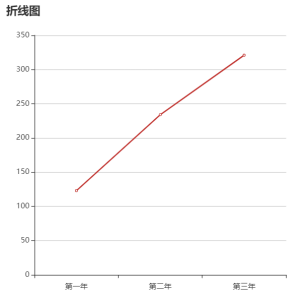
2.柱形图(横向)

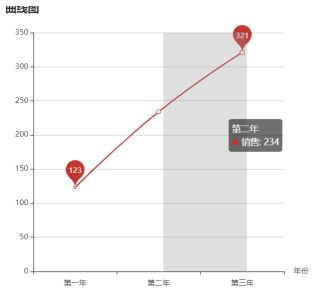
```
    var myChart = echarts.init(document.getElementById("map"));
      // 使用刚指定的配置项和数据显示图表。
      myChart.setOption({
          title:{
              text:'横向的柱形图',
          },
          tooltip:{
              trigger:'axis'
          },
          legend:{
              data:['第一年','第二年','第三年']
          },
          xAxis:{
              type:'value',
              name:'年份',
          },
          yAxis:{
              type:'category',
              data:['第一年','第二年','第三年']
          },
          series:{
              name:'销售',
              type:'bar',
              data:[123, 234, 321]
          }
      });
```



3.折线图/曲线图

```
var myChart = echarts.init(document.getElementById("map"));
      // 使用刚指定的配置项和数据显示图表。
      myChart.setOption({
          title:{
              text:'曲线图'
          },
          //鼠标放入点提示语句
          tooltip:{
              trigger:'axis',
              axisPointer:{
                  type:'shadow'
              }
          },
          legend:{
              data:['第一年','第二年','第三年']
          },
          xAxis:{
              name:'年份',
              data:['第一年','第二年','第三年']
          },
          yAxis:{},
          series:{
              name:'销售',
              type:'line',
              //决定是否为折线(false)或曲线(true)
              smooth:false,
              data:[123, 234, 321],
              //添加一个最大值的logo
              markPoint:{
                  data:[{
                      type:'max',
                      name:'最大值'
                  },
              //添加一个最小值的logo
                  {
                      type:'min',
                      name:'最小值'
                  }]
              }
          }
      });
```

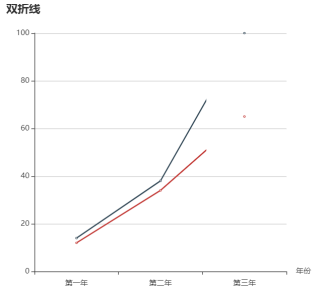




4.dataSet属性的图(折线图)

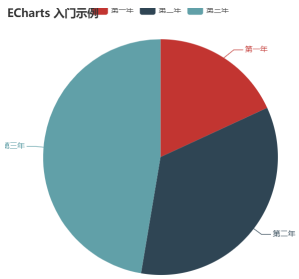
```
        var myChart = echarts.init(document.getElementById("map"));
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption({
            title:{
                text:'双折线'
            },
            tooltip:{
                tirgger:'axis',
                axisPointer:{
                    type:'line'
                }
            },
            legend:{
                data:['第一年','第二年','第三年']
            },
            xAxis:{
                name:'年份',
                data:['第一年','第二年','第三年']
            },
            yAxis:{},
            series:[
                {
                    type:'line',
                    data:[12,34,65]
                },{
                    type:'line',
                    data:[14,38,100]
                },
            ]

        });
```



5.饼图

```
        var myChart = echarts.init(document.getElementById("map"));
        // 使用刚指定的配置项和数据显示图表。
        myChart.setOption({
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip:{},
            legend:{
                data:['第一年','第二年','第三年']
            },
            series: [{
                name: '销量',
                type: 'pie',
                data: [
                    {name:'第一年',value:'123'},
                    {name:'第二年',value:'234'},
                    {name:'第三年',value:'321'}
                ]
            }]
        });
```



JQury的基础学习

1.基础操作

```
    <span id="txt" class="txtClass"> 字体 </span>
    <div class="div1">
        <!-- tiltle: 鼠标移动到时弹出得框 -->
        <span id="txt2" title="标题">空</span>
    </div>
    <button id="bt">点击展开/收起动画</button>
    <button id="bt2">点击停止动画</button>
    <div id="div2" style="background: red; width: 100px;height: 100px;">
    </div>
```

```
  //在页面加载完才执行
```

```
        $(document).ready(function(){
            //开始写Jqury
            $("#txt").text("修改后的字体"); //通过id获取修改属性
            $("#txt").text("修改后的字体").addClass("class1");  //通过id获取连"."设置属性并且添加一个css样式
            $("#txt").css({"font-size":"20px"});    //通过id获取设置他的css样式
            $(".txtClass").css({"font-size":"100px"});  //通过类获取设置他的css样式
            $("span").css({"font-size":"1px"});         //通过标签后去设置他的css样式
            //---------------------------高级设置属性
            $(".div1 #txt2").text("高级选择器").css({"font-size":"20px"});
            //---------------------------事件处理
            $("#txt2").click(function(){            //点击事件
                console.log("点击了"+$("#txt2").text());
            });
            $("#txt2").dblclick(function(){         //双击事件
                //-------------------小功能
                console.log("双击了"+$("#txt2").html());    //html返回网页
            });
            $("#txt2").mouseenter(function(){           //鼠标穿过事件
                //-------------------小功能
                document.getElementById("txt2").title = $("#txt2").text();       //修改title必须得使用document获取
                console.log("鼠标穿过"+$("#txt2").attr("title"));//attr("属性名")//获取属性里得值
            });
            $("#txt2").mouseleave(function(){       //鼠标离开事件
                console.log("鼠标离开"+$("#txt2").text());
            });
            $("#txt2").hover(function(){            //鼠标悬浮事件
                console.log("鼠标悬浮"+$("#txt2").text());
            });
            $(window).keydown(function(res){
                console.log(res);
                $("#txt2").text(String.fromCharCode(res.keyCode));
            });
            //---------------------------动画显示
            // $("#div2").fadeOut().fadeIn(3000);       //通过fade显示延迟来控制动画
            // $("#div2").fadeTo("show",0.15);          //颜色变淡
            // $("#div2").css({"display":"none"}).slideDown("slow");    //动画形式展开(但先设置不显示display)
            // $("#div2").slideUp("slow");         //动画形式收起(但先设置显示display)
            $("#bt").click(function(){
                $("#div2").slideToggle("slow");         //动画形式展开/展开(自动判断)
            });
```

```
            $("#div2").css({"position":"relative"}).animate({
                left:"123px",
                width:"200px",
                opacity:"0.5",
                hight:"+=100px"     //可以积加
                });         //开启一个向右移动的 动画(一定得设置position):该动画是一定执行完才会执行下一行
            });
            $("#bt2").click(function(){
                $("#div2").stop();              //获取id强制定制动画
            });
            $("#div2").hide(3000,function(){
                alert("动画执行完成");          //设置动画完成得回调
            })
```

2.动态页面

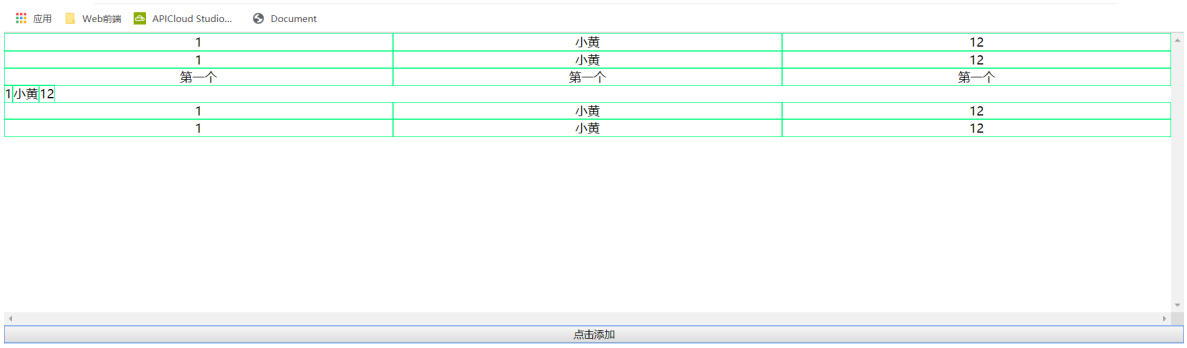
```
 <div id="div1" style="width: 100%;height: 50%; overflow: scroll;">
        <div style="width:100%; display: flex; flex-direction: row;text-align:center">
            <span style="flex: 1;border: 1px solid springgreen;">第一个</span>
            <span style="flex: 1;border: 1px solid springgreen;">第一个</span>
            <span style="flex: 1;border: 1px solid springgreen;">第一个</span>
        </div>
    </div>
    <button id="bt1" style="width: 100%;align-content: center;">点击添加</button>
```

```
var div = document.getElementById("div1");
        $(document).ready(function(){
            let obj = [
                    {index:1,name:'小黄',age:'12'},
                    {index:2,name:'小绿',age:'32'},
                    {index:3,name:'小且',age:'123'},
                    {index:4,name:'小属',age:'43'},
                ]
             //append：在div1里得内容后面补添加
             $("#div1").append('<div style="width:100%; class="q1" display: flex; flex-direction: row;text-align:center">'+
                                    '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].index+'</span>'+
                                    '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].name+'</span>'+
                                    '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].age+'</span>'+
                                    '</div>');
```

```
            $("#bt1").click(function(){

                //append：在div1里得内容后面补添加
                $("#div1").append('<div style="width:100%; display: flex; flex-direction: row;text-align:center">'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].index+'</span>'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].name+'</span>'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].age+'</span>'+
                                        '</div>');
                 //prepend：在div1里得内容头部补添加
                $("#div1").prepend('<div style="width:100%; display: flex; flex-direction: row;text-align:center">'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].index+'</span>'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].name+'</span>'+
                                        '<span style="flex: 1;border: 1px solid springgreen;">'+obj[0].age+'</span>'+
                                        '</div>');

                $("#div1").remove();        //删除div1
                // $("#div1").empty();        //清空div1内容（留着div）
            })
        })
```



3.获取父容器或资源

```
<body>
    <di style="width: 100%;height: 100%; display: flex;">
        <div class="div2" style="width: 50%;height: 50%;display: flex;margin: auto;">
            <div class="div3" style="width: 50%;height: 50%;display: flex;margin: auto;">
                <span id="tvItem" style="margin: auto;">子</span>
                <span id="tvItem2" style="margin: auto;">子</span>
            </div>
        </div>
    </di>
```

```
    <script>
        //-------------得到他的父容器
        // $("#tvItem").parent().css({"border":"1px solid red"});
        // //得到他的所有 父容器
        // $("#tvItem").parents().css({"border":"1px solid red"});
        //指定获取父容器
        // $("#tvItem").parents(".div2").css({"border":"1px solid red"});

        //---------------得到他的子元素
        // $(".div2").children().css({"font-size":"10px","color":"#ff0000"});
        //----------------得到同胞的元素
        // $("div").siblings().css({"border":"1px solid red"});
        //----------------过滤
        // $(".div3 span").first().css({"color":"#00ff00"});       //选择第一个
        // $(".div3 span").last().css({"color":"#00ff00"});       //选择最后一个
        // $(".div3 span").eq("0").css({"color":"#00ff00"});       //以index选择从0开始
        // $(".div3 span").filter("#tvItem2").css({"color":"#00ff00"});       //匹配选择（选择的）
        // $(".div3 span").not("#tvItem2").css({"color":"#00ff00"});    //匹配选择（去掉）

    </script
```

4.获取form表单里的值

```
<body>
    <form id="form1">
        <div>
            <div style="width: 100%;">
                名字:<input type="input" name="user">
            </div>
            <div style="width: 100%;">
                密码:<input type="input" name="pwd">
            </div>
            <div style="width: 100%;">
                <input id="bt_ok" type="submit" value="提交">
                <input type="reset" value="重置">
            </div>
        </div>
    </form>
    <script>
        $(document).ready(function(){   //页面加载后执行
        var data = {};  //准备一个对象
        $("#bt_ok").click(function(){           //点击提交
            var json = $("form").serializeArray();      //获取表单填入的所有值并给json对象
```

```
            $.each(json,function(){                 //循环json对象
                data[this.name] = this.value;       //根据name的key值给入值
            })
            alert(JSON.stringify(data));            //弹窗
        })
        })
    </script>
</body>
```

> 示例图

5.数组的操作

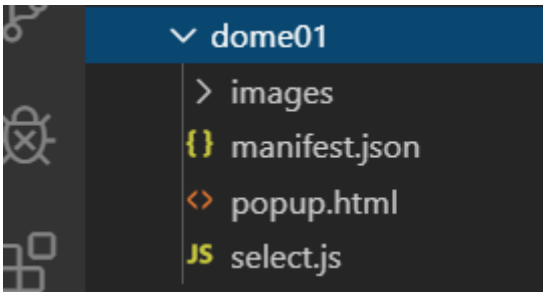
```
  var list = [
                {"name":"tt","age":55},
                {"name":"ww","age":66},
                {"name":"yy","age":32},
                {"name":"ii","age":54},
                {"name":"yy","age":32}
            ]

            //删除
            list.splice(1,1);
            //截取
            var res = list.slice(1,3);//不包括2,返回的是集合
            console.log(res);
            //排序
            list.sort(function(a,b){
                return a.age - b.age;
            })
            //循环集合
            list.forEach(function(res,index){
                console.log("name="+res.name+",age="+res.age);
                console.log("index="+index);
                console.log(list.indexOf(res)); //判断是否包括这个对象，有：索引，否：-1
            });
```

```
            //累计
            var counts = []  //准备一个数组
            list.forEach(function(res,index){        //拿到想要累计的数据
                  counts.push(list[index].name+"");
            });
            counts = counts.reduce(function(res,next){        //开始累计
参数res:计算后的数组,next：计算后数组的索引
                  res[next]?res[next]++:res[next]=1;  //如果算后的数组中重
复了，++，否则给1
                  return res;                  //一定得返回
            },{})            //一定得加一个空参数{}
            console.log(counts); //输出
```

chrome脚本基础

1.文件格式



> 文件
>
> 1.image资源已经
>
> 2.manifest.json配置文件
>
> 3.popup.html小窗口页面
>
> 4.js、css、html等一个自需文件

2.固定流程

- **manifest.json文件**

```
{
    "name": "csdn整理",        //插件名字
    "version": "1.1.1",          //插件版本
    "manifest_version": 2,
    "description": "开启代码旅程",     //插件解释
    "icons": {                   //插件icon
      "16": "images/icon.png",
      "48": "images/icon.png",
      "128": "images/icon.png"
    },

    "browser_action": {          //插件内部加载区块
```

```
      "default_title": "csdn整理",     //插件标题
      "default_icon": "images/icon.png",     //插件icon
      "default_popup": "popup.html"          //插件小窗口popup.html
    },
    "content_scripts": [            //逻辑文件
        {
            "js":["select.js"],     //后台植入的Js文件
            "matches":["https://blog.csdn.net/*"]       //拦截的地址"*"表示全部
        }
      ]
  }
```

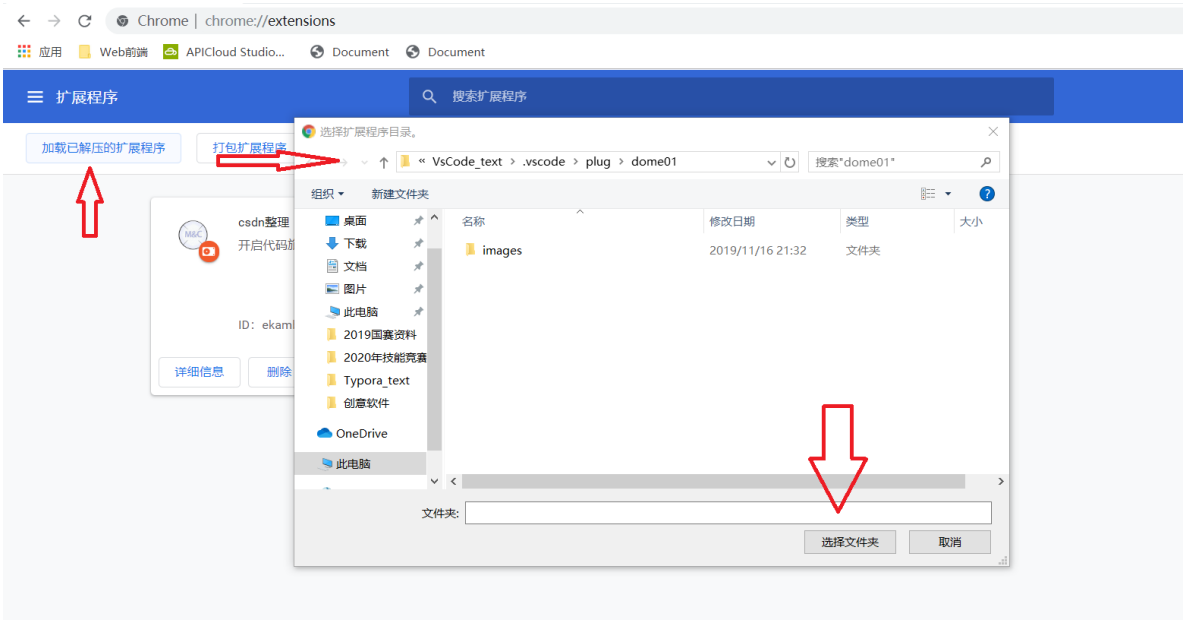
- **js文件样式**

```
document.querySelector(".btn-readmore").click();        //获取div容器默认点击
document.querySelector(".blog_container_aside").innerHTML="";    //取空div容器内容
```

- **popup.html文件样式**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <span>tmm:开始学习了</span>
</body>
</html>
```

- **操作流程**



> 创建一个插件就可以直接使用了