移动端网页特效

1.触屏事件

1.1触屏事件概述

移动端浏览器兼容性较好,我们不需要考虑以前JS的兼容性问题,可以放心的使用原生JS书写效果,但是移动端也有自己独特的地方。比如**触屏事件touch** (也称触摸事件) , Android和IOS都有。

touch对象代表一个触摸点，触摸点可能是一根手指,也可能是一根触摸笔。触屏事件可响应用户手指(或触控笔)对屏幕或者触控板操作。

常见的触屏事件如下:

触屏touch事件	说明
touchstart	手指触摸到一个DOM元素时触发
touchmove	手指在一个DOM元素上滑动时触发
touchend	手指从一个DOM元素上移开时触发

touchstart是手指点击DOM元素，touchmove是手指按住DOM元素在其上移动，touchend是离开DOM元素，如手指按下DOM元素后松开手指

示例:

```
var div = document.querySelector('div')
        // 手指触摸DOM元素事件
        div.addEventListener('touchstart', function () {
            console.log("手指触摸DOM元素");
        });
        // 手指在DOM元素上移动事件
        div.addEventListener('touchmove', function () {
            console.log("手指在DOM元素上移动");
        });
        // 手指离开DOM元素事件
        div.addEventListener('touchend', function () {
            console.log("手指离开DOM元素");
        })
```

1.2触摸事件对象(TouchEvent)

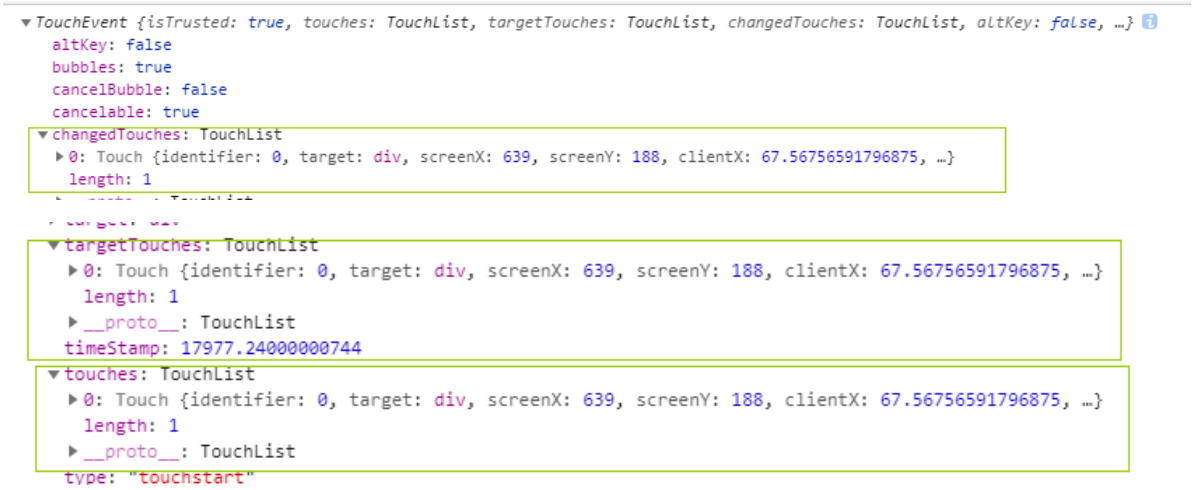
TouchEvent是一类描述手指在触摸平面(触摸屏、触摸板等)的状态变化的事件。这类事件用于描述一个或多个触点,使开发者可以检测触点的移动,触点的增加和减少,等等

touchstart. touchmove. touchend 三个事件都会各自有事件对象。

触摸列表	说明
touches	正在触摸屏幕的所有手指的一个列表
targetTouches	正在触摸当前DOM元素上的手指的一个列表
changedTouches	手指状态发生了改变的列表，从无到有,从有到无变化

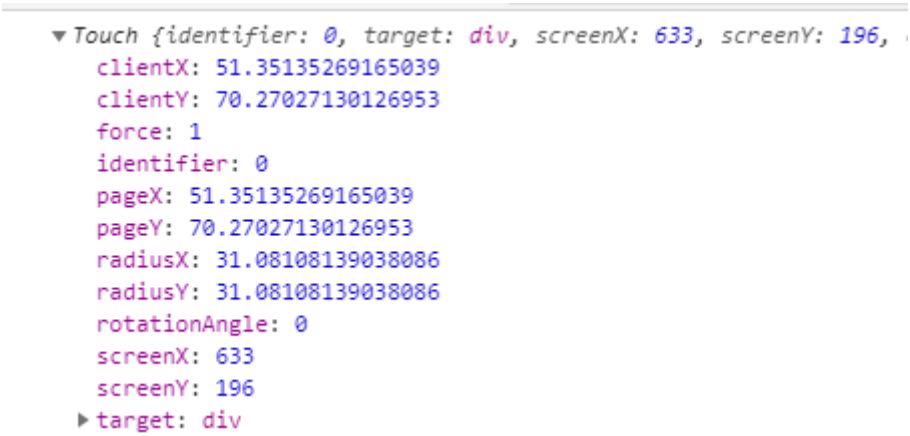
```
div.addEventListener('touchstart', function (e) {
            console.log(e);
        });
```

结果：

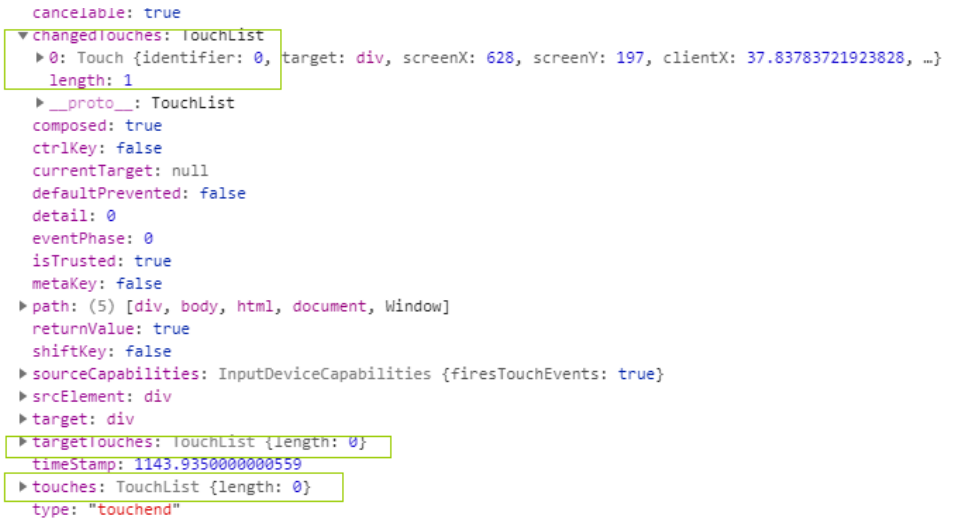


如果侦听的是一个DOM元素，touches和targetTouches两个返回结果一样

```
 console.log(e.targetTouches[0]);
// targetTouches[e] 就可以得到正在触摸dom元素的第个手指的相关信息,比如手指的坐标等等
```




```
// 手指离开DOM元素事件
        div.addEventListener('touchend', function (e) {
            console.log(e);
        })
```



手指离开DOM元素，手指状态发生了变化，因此只有changedTouches列表里有长度，touches和targetTouches列表里没有长度

1.3移动端拖动元素

1.touchstart. touchmove. touchend 可以实现拖动元素

2.但是拖动元素需要当前手指的坐标值我们可以使用targetTouche[0] 里面的pageX和pageY

3.移动端拖动的原理:手指移动中 ,计算出手指移动的距离。然后用盒子原来的位置+手指移动的距离

4.手指移动的距离:手指滑动中的位置 减去 手指刚开始触摸的位置

拖动元素三步曲:

(1)触摸元素touchstart: 获取**手指初始坐标**,同时获得**盒子原来的位置**

(2)移动手指touchmove:计算手指的滑动距离,并且移动盒子

(3)离开手指touchend:(可省略)

注意:手指移动也会触发滚动屏幕所以这里要阻止默认的屏幕滚动e.preventDefault();

```
var div = document.querySelector('div')
        // 手指初始坐标与盒子初始坐标定义为全局变量，两个事件内都需要使用到
        // 手指初始坐标
        var startX = 0;
        var startY = 0;
        // 盒子初始坐标
        var x = 0;
        var y = 0;
        // (1)触摸元素touchstart: 获取手指初始坐标,同时获得盒子原来的位置
        div.addEventListener('touchstart', function (e) {
            // 手指初始坐标
            startX = e.targetTouches[0].pageX;
            startY = e.targetTouches[0].pageY;
            // 盒子初始坐标
            x = this.offsetLeft;
            y = this.offsetTop;
        })
        //移动手指touchmove:计算手指的滑动距离,并且移动盒子
        div.addEventListener('touchmove', function (e) {
             // 手指滑动的距离：手指移动后的坐标-手指原始的坐标
            var movex = e.targetTouches[0].pageX - startX;
            var movey = e.targetTouches[0].pageY - startY;
```

```
            // 移动盒子 盒子距离=原来盒子的坐标+手指滑动的距离
            div.style.left = x + movex + 'px'
            div.style.top = y + movey + 'px'
        })
```

因为手指按住松开后离开元素，就不会触发触摸元素事件，无需再添加离开手指事件

1.4 classList属性

classList属性是HTML5新增的一个属性,返回元素的类名。但是ie10以上版本支持。

该属性用于在元素中添加,移除及切换CSS类。有以下方法

添加类:

```
element.classList.add('类名') ;
focus.classList. add( 'current') ;
```

移除类:

```
element.classList.remove('类名');
```

切换类:

```
element.classList.toggle('类名') ;
```

示例:

```
 <div class="one two"></div>
    <button>开关灯</button>

var div = document.querySelector('div')
        // classList 返回元素的类名，返回形式是伪数组的形式
        console.log(div.classList);
        // 1.添加类名  是在后面追加类名，不会覆盖以前的类名，注意前面不要加.
        div.classList.add('three');
        // 2.移除类名
        div.classList.remove('one')
        // 3.切换类 在添加类和移除类之间切换 可以简单实现开关灯效果
        var btn = document.querySelector('button')
        btn.addEventListener('click', function () {
            document.body.classList.toggle('bg')
        })
```

```
▼DOMTokenList(2) ["one", "two", value: "one two"]
   0: "two"
   1: "three"
   length: 2
   value: "two three"
  ▶__proto__: DOMTokenList
```

案例:移动端轮播图

移动端轮播图功能和基本PC端一致。

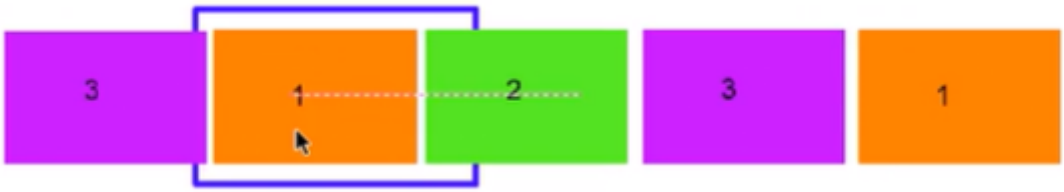
1.可以自动播放图片

2.手指可以拖动播放轮播图

在之前移动端携程案例上进行制作

与PC端不同的移动端小问题

1.PC端轮播图需要将第一张图片拷贝到最后，但移动端除此之外还需要将最后一张图片拷贝到第一张。移动端可以在第一张图片右滑到最后一张（如下图），无法通过直接跳转实现。



2.轮播图图片使用ul li装，与PC端一样，让li浮动，扩大ul的宽度，

此时注意：如和以前一样写下面的代码，会出现下图情况，图片变大了。

```
.focus img{
    width: 100%;
}

.focus ul{
    width: 500%;
}
```

原因：img图片宽度100%，而img的父级是li未指定宽度，li的父级是ul，而ul宽度是500%，img图片就会继承ul的宽度，图片变500%宽。

解决方法：我们有5个li，指定每个li的宽度是父级ul的20%即可（500% /5(5个li) = 100%），五分之一即20%

```
.focus img{
    width: 100%;
}

.focus ul{
    width: 500%;
}

.focus ul li{
    float: left;
    width: 20%;
}
```

3.由于将最后一张图片拷贝到了第一张，故默认显示的是最后一张图片，而我们需要显示第一张图片。

解决方法：使用marigin-left取负值**将ul整体左移100%距离**，使其显示第一张图片，**记得清除浮动，不影响后面的布局**

```
.focus ul{
    /* 清除浮动 */
    overflow: hidden;
    width: 500%;
    /* 让ul默认显示第一张图片 */
    margin-left: -100%;
}
```

注意：制作小圆点的ol自带边距，需要我们手动取消

自动播放

案例分析

①自动播放功能

②开启定时器

③移动端移动,可以使用**translate移动**，移动的距离=索引号*图片宽度，同样左移需要加负号

④想要图片优雅的移动,可以添加**过渡效果**

```
// 获取元素
    var focus = document.querySelector('.focus')
    var ul = focus.children[0];
    // 获得focus的宽度
    var w = focus.offsetWidth;
    // 1.自动播放图片
    // 1）利用定时器自动轮播图图片
    var index = 0;
    var timer = setInterval(function () {
        index++
        // 为了有动画效果，添加过渡
        ul.style.transition = 'all .3s'
        var translateX = -index * w
        ul.style.transform = 'translateX(' + translateX + 'px)'
    }, 2000)
```

自动播放功能-无缝滚动

案例分析:

①自动播放功能-无缝滚动

②注意,我们**判断条件是要等到图片滚动完毕再去判断,就是过渡完成后判断**

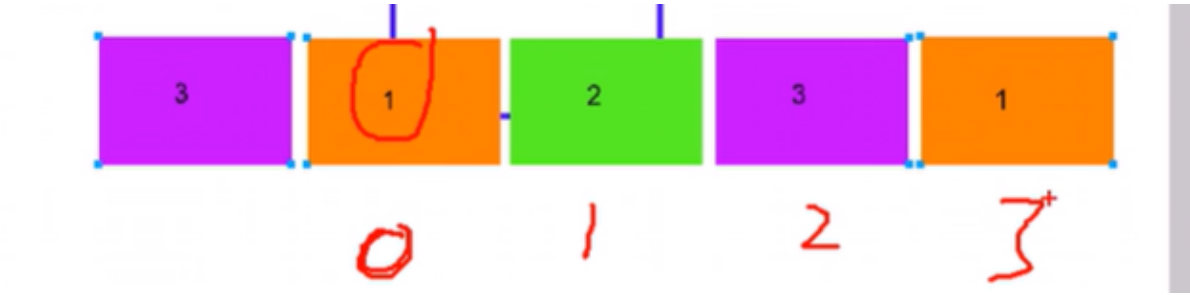
③此时需要添加检测**过渡完成事件transitionend**

④判断条件:如果索引号**大于等于3**说明走到最后一张图片,此时索引号要复原为0，快速跳到原第一张图片

⑤此时图片,去掉过渡效果，用新的索引号重新开始滚动

⑥**如果索引号小于0 ,说明是倒着走,索引号等于2**（手指移动让其倒着走情况下，与正走同理，让其快速跳到最后一张图片2）

注意：索引号是从原第一张图片开始数的，如下图。被拷贝过来的最后一张图没有地位，不记索引号内。




```
ul.addEventListener('transitionend', function () {
        // 无缝滚动
        if (index >= 3) {
            // 自动播放图片要到拷贝到第一张图片（现最后一张图），快速跳转到原第一张图片
            index = 0
            // 并去掉原有的过渡
            ul.style.transition = 'none'
            // 用新的索引号重新开始过渡，滚动图片
            var translateX = -index * w
            ul.style.transform = 'translateX(' + translateX + 'px)'
        } else if (index < 0) {
            index = 2
            // 去掉原有的过渡
            ul.style.transition = 'none'
            // 用新的索引号重新开始过渡，滚动图片
            var translateX = -index * w
            ul.style.transform = 'translateX(' + translateX + 'px)'
        }
    })
```

小圆点跟随变化效果

①小圆点跟随变化效果

②把ol里面li带有current类名的选出来去掉类名**remove**

③让当前索引号的小li加上current add

④但是,是等着过渡结束之后变化,所以这个写到transitionend事件里面

可以使用classList属性更简单完成

```
// 3.小圆点跟随变化效果
        // 把ol里面li带有current类名的选出来去掉类名remove
        ol.querySelector('.current').classList.remove('current')
        // 让当前索引号的小li加上current  add
        ol.children[index].classList.add('current')
```

为了动画更美观，可以给ol里li添加过渡效果

```
.focus ol li{
    /* 转为行内块，可以一行上显示，同时有空隙 */
    display: inline-block;
    width: 5px;
    height: 5px;
    background-color: red;
    border-radius: 2px;
    transition: all .3s;
}
```

手指滑动轮播图

①手指滑动轮播图

②**本质就是ul跟随手指移动,简单说就是移动端拖动元素**

③触摸元素touchstart:获取手指初始坐标**同时取消自动播放，清除定时器**

④移动手指touchmove:计算手指的滑动距离，并且移动盒子

⑤离开手指touchend:**根据滑动的距离分不同的情况**

⑥如果移动距离小于某个像素(50px)就回弹原来位置

⑦如果移动距离大于某个像素(50px)就上一张下一张滑动。

注意:

1.移动距离大于某个像素时判断是上一张还是下一张

解决方法：左滑，移动距离movex为负值；右滑，移动距离movex为正值。因此需要添加绝对值进行判断。左滑对应下一站，右滑对应上一张

2.严谨细节1

使用节流阀，用户手指长按但不移动时不做判断是否回弹还是上下一张。默认flag=false，在手指移动事件内添加flag=true，只有手指移动了才做判断，在手指离开事件内添加flag节流阀控制，**注意此时的定时器函数要在节流阀控制外面。**

3.严谨细节2

在手指移动事件内添加：e.preventDefault() //**阻止滚动屏幕行为**

4.**手指离开时，需要开启定时器自动播放**（为保证只有一个定时器启动，开启前先清除定时器）

5.手指滑动时，不需要动画效果，所以需要取消过渡效果

```
// 4.手指滑动轮播图
    var startx = 0;//手指左右滑动图片，不需要上下y轴
    var movex = 0; //定义全局变量，后面还需要使用到
    var flag = false;//使用节流阀，更严谨，用户手指长按但不移动时不做判断效果
    // 触摸元素touchstart:获取手指初始坐标
    ul.addEventListener('touchstart', function (e) {
        startx = e.targetTouches[0].pageX
```

```
        // 手指触摸时停止定时器，取消自动播放
        clearInterval(timer)
    })
    // 移动手指touchmove:计算手指的滑动距离，并且移动盒子
    ul.addEventListener('touchmove', function (e) {
        // 计算移动距离
        movex = e.targetTouches[0].pageX - startx
        // 移动盒子：盒子原来的位置+手指移动的距离
        var translateX = -index * w + movex
        // 手指滑动时，不需要动画效果，所以需要取消过渡效果
        ul.style.transition = 'none'
        ul.style.transform = 'translateX(' + translateX + 'px)'
        flag = true; // 用户手指移动时再触发手指离开判断上一张/下一张
        e.preventDefault() //阻止滚动屏幕行为
    })
    // 手指离开，根据移动距离判断是回弹还是播放上一张/下一张
    ul.addEventListener('touchend', function () {
        if (flag) {
            // 1)如果移动距离大于50px就播放上一张/下一张
            // 左滑，移动距离movex为负值；右滑，移动距离movex为正值。因此需要添加绝对值进行
判断
            if (Math.abs(movex) > 50) {
                if (movex > 0) {
                    // 右滑，上一张
                    index--
                } else {
                    // 左滑，下一站
                    index++
                }
                // 用新的索引号重新开始过渡，滚动图片
                var translateX = -index * w
                ul.style.transition = 'all .3s'
                ul.style.transform = 'translateX(' + translateX + 'px)'
            } else {
                // 2)如果移动距离小于等于50px就回弹,回弹过渡时间可以更短些，更自然
                var translateX = -index * w
                ul.style.transition = 'all .1s'
                ul.style.transform = 'translateX(' + translateX + 'px)'
            }
        }
        // 手指离开时，开启定时器自动播放
        clearInterval(timer);//清除定时器，保证待会只有一个定时器开启
        timer = setInterval(function () {
            index++
            // 为了有动画效果，添加过渡
            ul.style.transition = 'all .3s'
            var translateX = -index * w
            ul.style.transform = 'translateX(' + translateX + 'px)'
        }, 2000)
    })
```

案例:返回顶部

当页面滚动某个地方,就显示，否则隐藏

点击可以返回顶部

原理和移动端一样

案例分析

①滚动某个地方显示
②事件: scroll 页面滚动事件
③如果被卷去的头部 window pageYOffset ()大于某个数值
④点击，**window.scroll(0,0) 返回顶部**

```
var nav = document.querySelector('.local-nav')
    var goBack = document.querySelector('.goBack')
    window.addEventListener('scroll', function () {
        if (window.pageYOffset >= nav.offsetTop) {
            goBack.style.display = 'block'
        } else {
            goBack.style.display = 'none'
        }
    })
    goBack.addEventListener('click', function () {
        window.scroll(0, 0)
    })
```

卷过海外酒店头部再显示返回顶部按钮

做完轮播图的一个小问题

如下图，屏幕滚动时，轮播图里的图片盖住上面的搜索框

解决方法：提高搜索模块的层级 z-index: 999;

2.click延时解决方案

移动端click事件会有300ms的延时,原因是移动端屏幕双击会缩放(double tapto zoom)页面。

解决方案:

1.禁用缩放。浏览器禁用默认的双击缩放行为并且去掉 300ms的点击延迟。

```
<meta name="viewport" content="user-scalable=no">
```

2.*利用touch事件自己封装这个事件解决300ms延迟。*

原理就是:
1)当我们手指触摸屏幕,记录当前触摸时间
2)当我们手指离开屏幕,用离开的时间减去触摸的时间
3)如果时间小于150ms ,并且没有滑动过屏幕，那么我们就定义为点击

```
//封装tap,解决click 300ms延时
function tap (obj, callback){
    var isMove = false;
    var startTime= 0; //记录触摸时候的时间变量
    obj.addEventListener ('touchstart', function (e) {
    startTime = Date.now(); //记录触摸时间
    });
    obj.addEventListener ('touchmove', function (e) {
        isMove = true;//看看是否有滑动，有滑动算拖拽，不算点击
    });
    obj.addEventListener ('touchend', function (e) {
        if (!isMove && (Date.now() - startTime) < 150) { // 如果手指触摸和离开时间小
于150ms算点击
        callback && callback(); /执行回调函数
        }
        isMove = false; // 取反重置
        startTime = 0;
    });

    //调用
tap(div, function(){ // 执行代码 });
```

3.使用fastclick插件

该方法更常用

fastclick 插件解决300ms延迟。

GitHub官网地址: https://github.com/ftlabs/fastclick

使用方法：在html里引入fastclick插件，在script里放入下面代码块即可

```
if ('addEventListener' in document) {
    document.addEventListener('DOMContentLoaded', function() {
        FastClick.attach(document.body);
    }, false);
}
```

Usage

Include fastclick.js in your JavaScript bundle or add it to your HTML page like this:

```
<script type='application/javascript' src='/path/to/fastclick.js'></script>
```

The script must be loaded prior to instantiating FastClick on any element of the page.

To instantiate FastClick on the `body`, which is the recommended method of use:

```
if ('addEventListener' in document) {
        document.addEventListener('DOMContentLoaded', function() {
                FastClick.attach(document.body);
        }, false);
}
```

Or, if you're using jQuery:

```
$(function() {
        FastClick.attach(document.body);
});
```

3.移动端插件

3.1 Swiper插件的使用

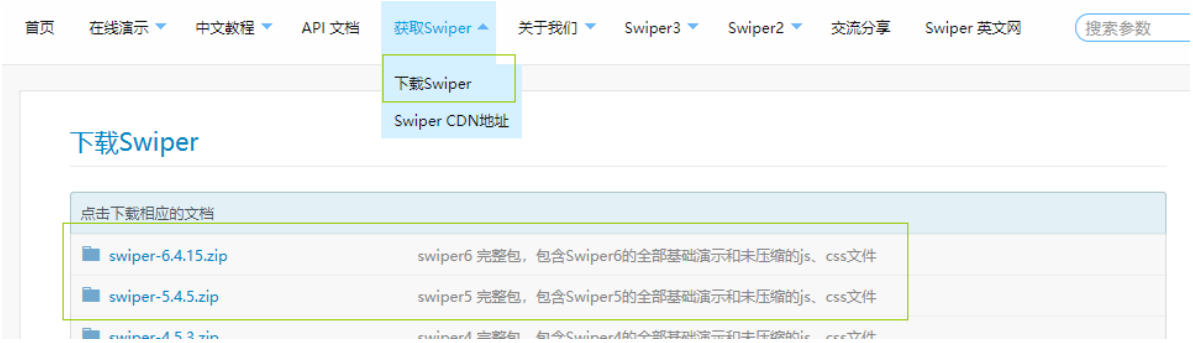
中文官网地址: https://www.swiper.com.cn/

官网的使用说明：https://www.swiper.com.cn/usage/index.html

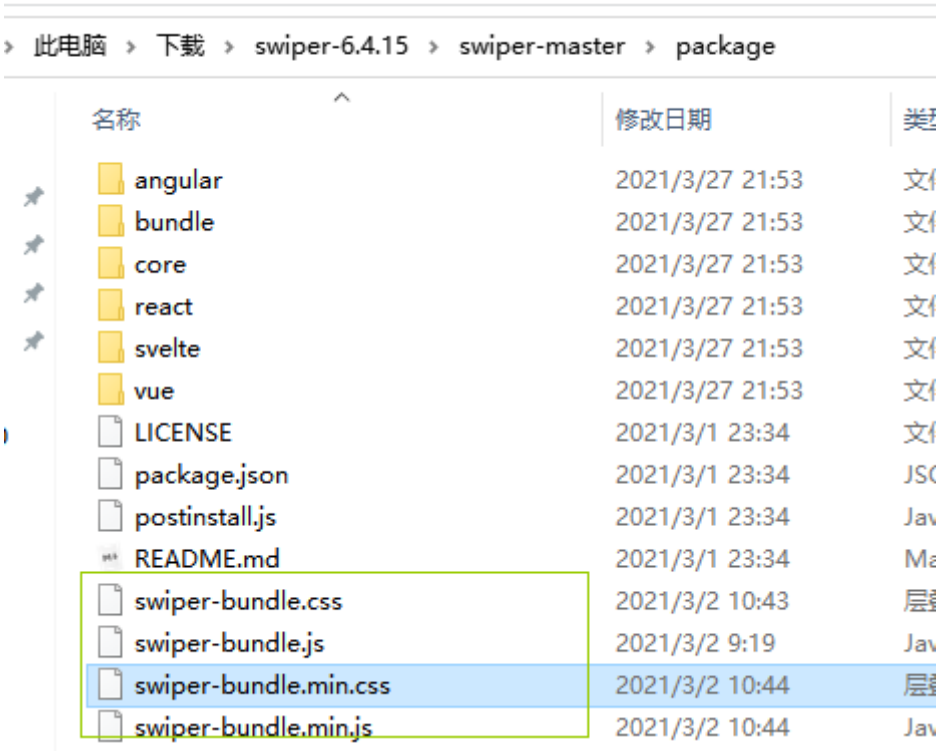
1.引入插件相关文件。需要用到的文件有swiper-bundle.min.js和swiper-bundle.min.css文件

2.按照规定语法使用。

官网下载



相关css /js文件在如下图里面，将相关文件直接拖入vscode即可使用



1.引入相关文件

```
<!-- 引入swiper css文件 -->
    <link rel="stylesheet" href="css/swiper-bundle.min.css">
    <!-- 引入swiper js文件 -->
    <script src="js/swiper-bundle.min.js"></script>
    <!-- 引入首页 js文件 -->
    <script src="js/index.js"></script>
```

2.复制相关的html/css/js代码

3.注意引用完后，轮播图图片会覆盖搜索框，解决方法：提高搜索模块的层级 z-index: 999;

结果：

3.2其他移动端常见插件

●superslide : http://www.superslide2.com/

●iscroll : https://github.com/cubiq/iscroll

3.3插件的使用总结

1.确认插件实现的功能
2.去官网查看使用说明
3.下载插件
4.打开demo实例文件,查看需要引入的相关文件,并且引入
5.复制demo实例文件中的结构html ,样式cssl以及js代码

3.4练习-移动端视频插件zy.media.js

H5给我们提供了video标签,但是浏览器的支持情况不同。

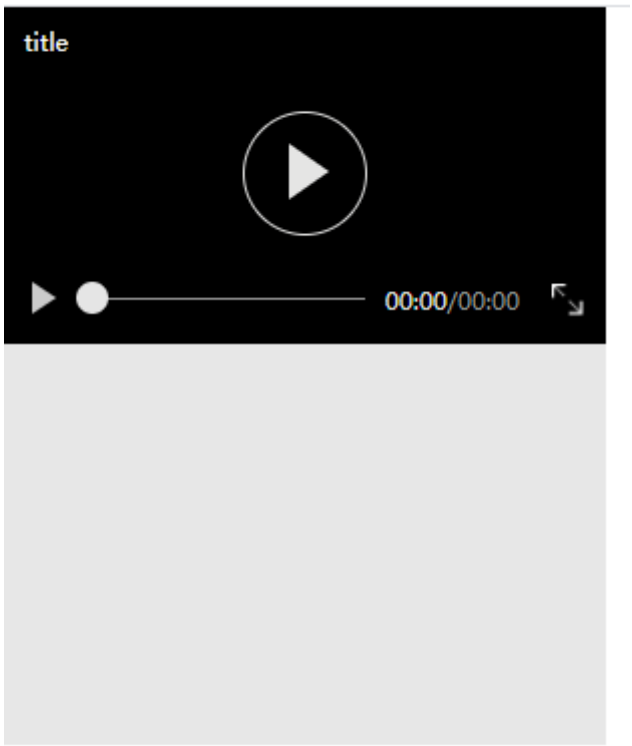
不同的视频格式文件,我们可以通过source解决。

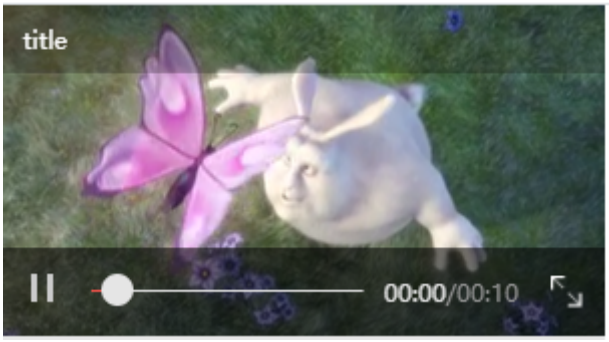
但是外观样式,还有暂停,播放,全屏等功能我们只能自己写代码解决。

这个时候我们可以使用插件方式来制作。

```
    <link rel="stylesheet" href="zy.media.min.css">
    <script src="zy.media.min.js"></script>
    <style>
        #modelView {
            background-color: #DDDDDD;
            z-index: 0;
            opacity: 0.7;
            height: 100%;
            width: 100%;
            position: relative;
        }

        .playvideo {
            padding-top: auto;
            z-index: 9999;
```

```
            position: relative;
            width: 300px;
            height: 200px;
        }

        .zy_media {
            z-index: 999999999
        }
    </style>
</head>

<body>
    <div class="playvideo">
        <div class="zy_media">
            <video data-config='{"mediaTitle": "title"}'>
                <source src="mov.mp4" type="video/mp4">
                您的浏览器不支持HTML5视频
            </video>

        </div>
        <div id="modelView"> </div>
    </div>
    <script src="zy.media.min.js"></script>
    <script>
        // 一般不自动播放
        zymedia('video', {
            autoplay: false
        });
    </script>
</body>

</html>
```





4.移动端常用开发框架

4.1框架概述

框架,顾名思义就是-套架构,它会基于自身的特点向用户提供-套较为完整的解决方案。 框架的控制权在框架本身,使用者要按照框架所规定的某种规范进行开发。
插件一般是为 了解决某个问题而专门存在,其功能单一, 并且比较小。
前端常用的框架有**Bootstrap. Vue. Angular. React**等. 既能开发PC端,也能开发移动端
前端常用的移动端插件有**swiper. superslide. iscroll**等。

框架:大而全, 一整套解决方案
插件:小而专一, 某个功能的解决方案

4.2 Bootstrap

Bootstrap是一个简洁、直观、强悍的前端开发框架,它让web开发更迅速、简单。
它能开发PC端,也能开发移动端

Bootstrap JS插件使用步骤:
1.引入相关js文件
2.复制HTML结构
3.修改对应样式
4.修改相应JS参数

本地存储

随着互联网的快速发展,基于网页的应用越来越普遍,同时也变的越来越复杂,为为满足各种各样的需求,会经常性在本地存储大量的数据, HTML 5规范提出了相关解决方案。

本地存储特性

1.数据存储在用户浏览器中

2.设置、读取方便、甚至页面刷新不丢失数据

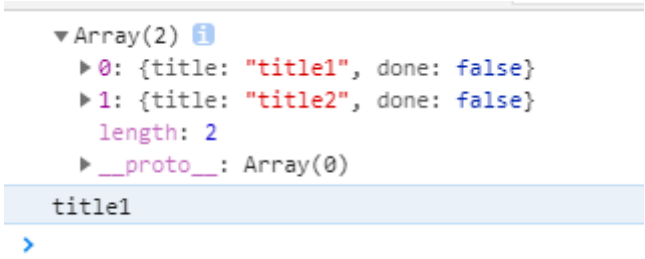
3.容量较大, sessionStorage约5M、localStorage约20M

4.**本地存储只能存储字符串，可以将对象JSON.stringify()编码后存储为字符串型，可再用JSON.parse()再转为对象，从而可以使用对象里的方法**

示例:

```
var todolist = [{
            title: 'title1',
```

```
            done: false
        }, {
            title: 'title2',
            done: false
        }];
        // 1.本地存储只能存储字符串数据格式，因此需要将数组对象转换为字符串格式
JSON.stringify()
        var data = JSON.stringify(todolist)
        localStorage.setItem('todo', data)
        // 2.为了能使用对象里的属性，我们需要将字符串转为为对象格式，使用JSON.parse()
        data = JSON.parse(data)
        console.log(data);
        console.log(data[0].title);
```



window.sessionStorage

1.生命周期为关闭浏览器窗口,页面刷新不丢失数据

2.在同一个窗口(页面)下数据可以共享

3.以键值对的形式存储使用

存储数据:

```
sessionStorage.setltem(key, value)
```

获取数据:

```
sessionStorage.getltem(key)
```

删除数据:

```
sessionStorage.removeltem(key)
```

删除所有数据:

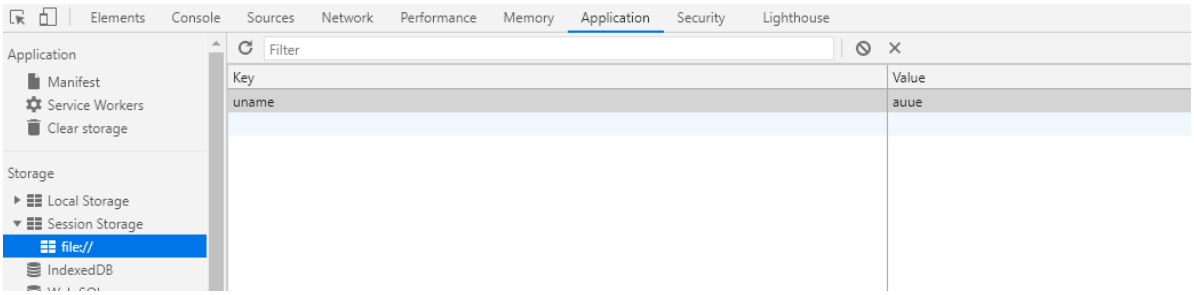
```
sessionStorage.clear()
```

示例:

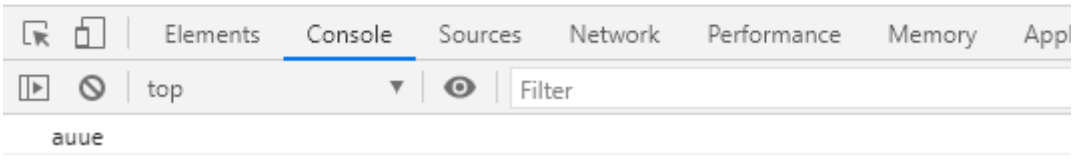
```
<input type="text">
    <button class="set">存储数据</button>
    <button class="get">获取数据</button>
    <button class="remove">删除数据</button>
    <button class="clear">删除所有数据</button>
    <script>
        var ipt = document.querySelector('input')
```

```
        var set = document.querySelector('.set')
        var get = document.querySelector('.get')
        var remove = document.querySelector('.remove')
        var clear = document.querySelector('.clear')
        set.addEventListener('click', function () {
            // 点击，存储表单里面的值
            var val = ipt.value
            sessionStorage.setItem('uname', val)
            // sessionStorage.setItem('pwd', val)
        })
        get.addEventListener('click', function () {
            // 点击，获取表单里面的值
            console.log(sessionStorage.getItem('uname'));
        })
        remove.addEventListener('click', function () {
            // 点击，删除表单里面第一个键值对
            sessionStorage.removeItem('uname')
        })
        clear.addEventListener('click', function () {
            // 点击，删除表单里面所有的值，慎用
            sessionStorage.clear()
        })
    </script>
```

存储数据：



获取数据：



删除数据：

存储两个数据：

Key	Value
uname	11
pwd	11

删除所有数据

Key	Value

window.localStorage

1.**生命周期永久生效,除非手动删除,否则关闭页面也会存在**

2.**可以多窗口(页面)共享(同浏览器可以共享)**

3.以键值对的形式存储使用

存储数据:

```
localStorage.setItem(key, value)
```

获取数据:

```
localStorage.getItem(key)
```

删除数据:

```
localStorage.removeItem(key)
```

删除所有数据:

```
localStorage.clear()
```

用法和sessionStorage一样，唯一不同在于生命周期

案例: 记住用户名

如果勾选记住用户名，下次用户打开浏览器,就在文本框里面自动显示上次登录的用户名

案例分析:

①把数据存起来,用到本地存储

②关闭页面,也可以显示用户名,所以用到localStorage

③打开页面,先判断是否有这个用户名,如果有,就在表单里面显示用户名,并且勾选复选框

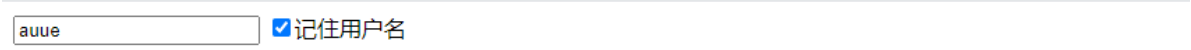
④当复选框发生改变的时候change事件

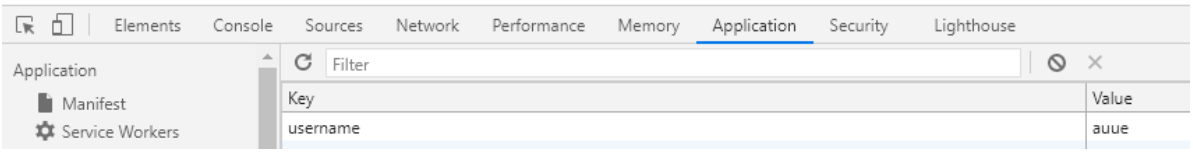
⑤如果勾选,就存储,否则就移除

```
var username = document.querySelector('#username')
        var remember = document.querySelector('#remember')
        // 打开页面,先判断是否有这个用户名,如果有,就在表单里面显示用户名,并且勾选复选框
        if (localStorage.getItem('username')) {
            username.value = localStorage.getItem('username')
            remember.checked = true
```

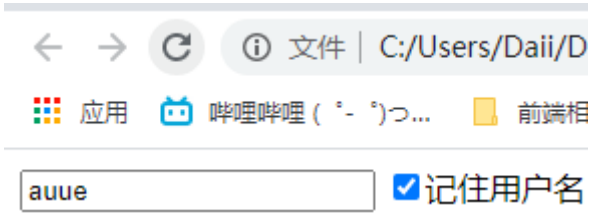
```
        }
        // 当复选框发生改变的时候change事件
        remember.addEventListener('change', function () {
            // 如果勾选,就存储,否则就移除
            if (this.checked) {
                localStorage.setItem('username', username.value)
            } else {
                localStorage.removeItem('username')
            }
        })
```

输入用户名，勾选复选框，数据存入LocalStorage：





页面刷新后默认显示之前的用户名，并勾选复选框：



不勾选，则删除数据

再刷新时，表单数据为空