排他思想

<!-- 如果有同一组元素，我们想要某一个元素实现某种样式，需要用到循环的排他思想算法  -->

    <!-- 1；所有元素全部清除样式（干掉其他人 -->

    <!-- 2；给当前元素设置样式（留下我自己 -->

    <!-- 3；注意顺序不能颠倒 首先干掉其他人，在设置自己  -->

  <button>按钮1</button>

    <button>按钮2</button>

    <button>按钮3</button>

    <button>按钮4</button>

    <button>按钮5</button>

 <script>

        //获取所有按钮

        var btns = document.querySelectorAll('button')

        // btn得到是伪数组 里面的每一个元素 btns[i]

        for(var i = 0; i < btns.length; i++){

            btns[i].onclick = function(){

                // console.log(11);

                // 1我们先让所有元素的背景颜色去掉

                for(var i = 0; i <btns.length; i++){

                    btns[i].style.backgroundColor = ''

                }

                // 2；然后才让当前背景颜色为pink

                this.style.backgroundColor = 'pink'

            }

        }

</script>

排他思想

百度换肤

<title>百度换肤案例</title>

    <!-- 案例分析 -->

    <!-- 这个案例联系的是给一组元素注册事件 -->

    <!-- 给4个小图片利用循环注册点击事件 -->

    <!-- 当我们点击了这个图片 让我们页面背景颜色改为当前的图片 -->

    <!-- 核心算法 把当前图片的src路劲取过来 给body作为背景即可 -->

    <style>

        \*{

            margin: 0;

            padding: 0;

        }

        body {

            background: url(./images/1.jpg)no-repeat center top;

        }

        li {

            list-style: none;

        }

        .baidu {

            overflow: hidden;

            margin: 100px auto;

            background-color: #fff;

            width: 410px;

            padding-top: 3px;

        }

        .baidu li {

            float: left;

            margin: 0 1px;

            cursor: pointer;

        }

        .baidu img {

            width: 100px;

        }

    </style>

</head>

<body>

    <ul class="baidu">

        <li><img src="./images/1.jpg" alt=""></li>

        <li><img src="./images/2.jpg" alt=""></li>

        <li><img src="./images/3.jpg" alt=""></li>

        <li><img src="./images/4.jpg" alt=""></li>

    </ul>

    <script>

        // 1获取元素 所有的img图片

        var imgs = document.querySelector('.baidu').querySelectorAll('img')

        // 2;循环注册事件

        for(var i = 0 ; i < imgs.length;i++){

            imgs[i].onclick = function(){

                // console.log(this.src);

                // 得到当前图片src属性的路径 添加给body

                document.body.style.backgroundImage = 'url('+this.src+')'

            }

        }</script>

百度换肤

表格隔行换色

<title>表格隔行换色</title>

    <!-- 1；用到新的鼠标事件鼠标经过onmouseover 鼠标离开 onmouseout -->

    <!-- 2；核心思路 鼠标经过tr行 当前的行变背景颜色 鼠标离开去掉当前的背景颜色 -->

    <!-- 注意：第一行（thead里面的行）不需要变换颜色 因此我们获取的是tbody里面的行  -->

    <style>

        table{

            width: 800px;

            margin: 100px auto;

            border-collapse: collapse;

            text-align: center;

            font-size: 14px;

        }

        thead tr {

            height: 30px;

            background-color: skyblue;

        }

        tbody tr {

            height: 30px;

        }

        tbody td {

            border-bottom: 1px solid #d7d7d7;

            font-size: 12px;

            color: blue;

        }

        .bg {

            background-color: pink;

        }

    </style>

</head>

<body>

    <table>

        <thead>

            <tr>

                <th>代码</th>

                <th>名称</th>

                <th>最新公布净值</th>

                <th>累计净值</th>

                <th>前单位净值</th>

                <th>净值增长率</th>

            </tr>

        </thead>

        <tbody>

            <tr>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

            </tr>

            <tr>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

            </tr>

            <tr>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

            </tr>

            <tr>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

            </tr>

            <tr>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

                <td>003526</td>

            </tr>

        </tbody>

    </table>

    <script>

        // 1; 获取元素 获取的是tbody里面的所有行

        var trs = document.querySelector('tbody').querySelectorAll('tr')

        // 2;利用循环绑定s鼠标经过注册事件

        for(var i = 0 ; i < trs.length; i++){

            trs[i].onmousemove = function(){

                // console.log(1);

                // 鼠标经过事件绑定成功 经过谁谁变化背景颜色

                // 这里通过添加类的方式去做 如果后期需要变化颜色 直接修改css样式 js不需要改动

                this.className = 'bg'

            }

            // 鼠标离开的时候移除类

            trs[i].onmouseout = function(){

                this.className = ''

            }

        }

</script>

表格隔行换色

全选取消反选

<title>全选取消反选</title>

    <!-- 1；全选和取消全选做法：让下面所有的复选框的checked属性（选择状态）跟随全选按钮即可 -->

    <style>

        \* {

            padding: 0;

            margin: 0;

        }

        .wrap {

            width: 300px;

            margin: 100px auto 0;

        }

        table {

            border-collapse: collapse;

            border-spacing: 0;

            border: 1px solid #c0c0c0;

            width: 300px;

        }

        th,

        td {

            border: 1px solid #d0d0d0;

            color: #404060;

            padding: 10px;

        }

        th {

            background-color: #09c;

            font: bold 16px "微软雅黑";

            color: #fff;

        }

        td {

            font: 14px "微软雅黑";

        }

        tbody tr {

            background-color: #f0f0f0;

        }

        tbody tr:hover {

            cursor: pointer;

            background-color: #fafafa;

        }

    </style>

</head>

<body>

    <div class="wrap">

        <table>

            <thead>

                <tr>

                    <th>

                        <input type="checkbox" id="j\_cbAll">

                    </th>

                    <th>商品</th>

                    <th>价钱</th>

                </tr>

            </thead>

            <tbody id="j\_tb">

                <tr>

                    <td>

                        <input type="checkbox">

                    </td>

                    <td>iPhone8</td>

                    <td>8000</td>

                </tr>

                <tr>

                    <td>

                        <input type="checkbox">

                    </td>

                    <td>iPad Pro</td>

                    <td>5000</td>

                </tr>

                <tr>

                    <td>

                        <input type="checkbox">

                    </td>

                    <td>iPad Air</td>

                    <td>2000</td>

                </tr>

                <tr>

                    <td>

                        <input type="checkbox">

                    </td>

                    <td>Apple Watch</td>

                    <td>2000</td>

                </tr>

            </tbody>

        </table>

    </div>

    <script>

        // 1；获取元素

        var j\_cbAll = document.getElementById('j\_cbAll')

        // tbody 下面所有的复选框

        var j\_tbs = document.getElementById('j\_tb').getElementsByTagName('input')

        // 2 全选注册事件

        j\_cbAll.onclick = function () {

            // 选择当前 复选框的 checked 属性

            // console.log(this.checked) true false

            for (var i = 0; i < j\_tbs.length; i++) {

                j\_tbs[i].checked = this.checked

            }

        }

          // 2；下面复选框需要全部选中 。上面的全选v才能选中做法

        // 给下面所有的复选框绑定点击事件，每次点击，都要循环查看下面所有的

        // 复选框是否有没选中的，如果有一个没选中的，上面全选就不选中

        // // 可以设置一个变量

        for (var i = 0; i < j\_tbs.length; i++) {

            j\_tbs[i].onclick = function () {

                //   每次点击下面的复选框都要循环检查4个小按钮是否全被选中

                for (var i = 0; i < j\_tbs.length; i++) {

                     // flag 控制全选按钮是否选中

                    falg = true

                    if (!j\_tbs[i].checked) {

                        falg = false

                         break  //退出for循环 这样可以提高执行效率 因为只要有一个没有选中，剩下的就无需循环判断了

                    }

                }

                j\_cbAll.checked = falg

            }

        }

全选取消反选

    </script>

</body> tab栏切换

 <title>tab栏切换</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        li {

            list-style-type: none;

        }

        .tab {

            width: 978px;

            margin: 100px auto;

        }

        .tab\_list {

            height: 39px;

            border: 1px solid #ccc;

            background-color: #f1f1f1;

        }

        .tab\_list li {

            float: left;

            height: 39px;

            line-height: 39px;

            padding: 0 20px;

            text-align: center;

            cursor: pointer;

        }

        .tab\_list .current {

            background-color: #c81623;

            color: #fff;

        }

        .item\_info {

            padding: 20px 0 0 20px;

        }

        .item {

            display: none;

        }

    </style>

</head>

<body>

    <div class="tab">

        <div class="tab\_list">

            <ul>

                <li>商品介绍</li>

                <li>规格与包装</li>

                <li>售后保障</li>

                <li>商品评价（50000）</li>

                <li class="current">手机社区</li>

            </ul>

        </div>

        <div class="tab\_con">

            <div class="item">

                商品介绍模块内容

            </div>

            <div class="item">

                规格与包装模块内容

            </div>

            <div class="item">

                售后保障模块内容

            </div>

            <div class="item">

                商品评价（50000）模块内容

            </div>

            <div class="item" style="display: block;">

                手机社区模块内容

            </div>

        </div>

    </div>

    <script>

        // 1;上面的模块选项卡 点击某一个 当前这一个底色会是红色。其余不变（排他思想）修改类名的方式

        // 获取元素 所有的xiaoli获取过来

        var tab\_list = document.querySelector('.tab\_list')

        var lis = document.querySelectorAll('li')

        var item =document.querySelectorAll('.item')

        // for 循环绑定点击事件

        for (var i = 0; i < lis.length; i++) {

            // 开始给5个li。设置索引号

            lis[i].setAttribute('index',i)

            lis[i].onclick = function () {

                console.log(1);

                // 排他思想 干掉所有人 其余的li清除 class

                for(var i = 0 ; i < lis.length; i++){

                    lis[i].className = ''

                }

                this.className = 'current'

                // 2；下面的模块内容 会跟随上面的选项卡变化，所以下面模块变化写到点击事件里面

                // 规律 下面的模块显示内容和上面的选项卡一一对应，相匹配

                // 核心思路 给上面的tab\_list里面的所有li添加自定义属性 属性值从0 开始

                //得到自定义属性的值

              var index =  this.getAttribute('index')

            //   console.log(index);

            // 当我们点击tab\_list里面的某个li,让tab\_con里面对应序号的内容显示，其他隐藏（排他思想

            for(var i = 0 ;i <item.length ; i++){

                item[i].style.display = 'none'

            }

            item[index].style.display = 'block'

                tab栏切换

            }

        }

    </script>

下拉菜单

<title>下拉菜单</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        li {

            list-style-type: none;

        }

        a {

            text-decoration: none;

            font-size: 14px;

        }

        .nav {

            margin: 100px;

        }

        .nav>li {

            position: relative;

            float: left;

            width: 80px;

            height: 41px;

            text-align: center;

        }

        .nav li a {

            display: block;

            width: 100%;

            height: 100%;

            line-height: 41px;

            color: #333;

        }

        .nav>li>a:hover {

            background-color: #eee;

        }

        .nav ul {

            display: none;

            position: absolute;

            top: 41px;

            left: 0;

            width: 100%;

            border-left: 1px solid #FECC5B;

            border-right: 1px solid #FECC5B;

        }

        .nav ul li {

            border-bottom: 1px solid #FECC5B;

        }

        .nav ul li a:hover {

            background-color: #FFF5DA;

        }

    </style>

</head>

<body>

    <ul class="nav">

        <li>

            <a href="#">微博</a>

            <ul >

                <li>

                    <a href="">私信</a>

                </li>

                <li>

                    <a href="">评论</a>

                </li>

                <li>

                    <a href="">@我</a>

                </li>

            </ul>

        </li>

        <li>

            <a href="#">微博</a>

            <ul >

                <li>

                    <a href="">私信</a>

                </li>

                <li>

                    <a href="">评论</a>

                </li>

                <li>

                    <a href="">@我</a>

                </li>

            </ul>

        </li>

        <li>

            <a href="#">微博</a>

            <ul >

                <li>

                    <a href="">私信</a>

                </li>

                <li>

                    <a href="">评论</a>

                </li>

                <li>

                    <a href="">@我</a>

                </li>

            </ul>

        </li>

        <li>

            <a href="#">微博</a>

            <ul >

                <li>

                    <a href="">私信</a>

                </li>

                <li>

                    <a href="">评论</a>

                </li>

                <li>

                    <a href="">@我</a>

                </li>

            </ul>

        </li>

    </ul>

    <script>

        // 导航栏厘米的li都有鼠标经过效果，所有需要循环注册鼠标事件

        // 核心原理：当鼠标经过li里面的第二个孩子ul显示 当鼠标离开 则ul隐藏

        // 获取元素

        var nav = document.querySelector('.nav')

        var lis = nav.children //得到4个小li

        for(var i = 0 ;i < lis.length;i++){

            // 鼠标经过显示

            lis[i].onmouseover = function(){

                // console.log(1);

                this.children[1].style.display = 'block'

            }

            // 鼠标离开隐藏

            lis[i].onmouseout = function(){

                // console.log(1);

                this.children[1].style.display = 'none'

            }

        }

</script>

下拉菜单

微博发布案例

<title>微博发布案例</title>

    <!-- 核心思路：点击按钮之后，就动态创建一个li 添加到ul里面 -->

    <!-- 创建li的同事，把文本域里面的值通过li.innerHTML赋值给li -->

    <!-- 如果想要新的留言后面显示就用appendCild如果想要显示在前面就是insertBefore -->

    <style>

        textarea {

            width: 200px;

            height: 200px;

            border: 1px solid pink;

            margin: auto 100px;

        }

    </style>

</head>

<body>

    <textarea name="" id="">123</textarea>

    <button>发布</button>

    <ul>

    </ul>

    <script>

        // 1;获取按钮

        var btn = document.querySelector('button')

        var test = document.querySelector('textarea')

        var ul = document.querySelector('ul')

        // 2;注册事件

        btn.onclick = function () {

            if (test.value == '') {

                alert('您没有输入内容')

                return false

            } else {

                // console.log(test.value)

                // (1)创建元素

                var li = document.createElement('li')

                // 先有li 才能赋值

                li.innerHTML = test.value

                //  (2)添加元素 添加给ul

                // ul.appendChild(li)

                // 指定添加元素

                ul.insertBefore(li,ul.children[0])

            }

        }

</script>

微博发布案例

动态生成表格

 <title>动态生成表格</title>

    <style>

        table {

            width: 500px;

            margin: 100px auto;

            border-collapse: collapse;

            text-align: center;

        }

        td,

        th {

            border: 1px solid #333;

        }

        thead tr {

            height: 40px;

            background-color: #ccc;

        }

    </style>

</head>

<body>

    <table cellspacing="0">

        <thead>

            <tr>

                <th>姓名</th>

                <th>科目</th>

                <th>成绩</th>

                <th>操作</th>

            </tr>

        </thead>

        <tbody>

        </tbody>

    </table>

    <script>

        // 动态生成表格 因为里面的学生数据都是动态的，我们需要js动态生成

        // 这里我们模拟数据，自己定义好数据。数据我们采取对象形式存储

        var data = [

            {

                name: '魏玲玲',

                subject: 'javaScript',

                score: 100

            },  {

                name: '宏利',

                subject: 'javaScript',

                score: 80

            },   {

                name: '傅恒',

                subject: 'javaScript',

                score: 90

            },  {

                name: '名域',

                subject: 'javaScript',

                score: 88

            }

        ]

        // 创建行 所有的数据都是放到tbody里面 因为行很多，我们需要循环创建多个行

        var tbody  = document.querySelector('tbody')

        for(var i = 0 ;i < data.length;i++){

            // 创建tr行

            var tr = document.createElement('tr')

            // 添加行

            tbody.appendChild(tr)

            // 创建单元格 放到tr里面 td 单元格的数量每个对象里面的属性个数

            // for循环遍历对象 data[i]就是对象

            for(var k in data[i] ){

                // k得到的是属性名 data[i]得到的是属性值

                // 把得到的属性值给td

                // 创建td单元格

                var td = document.createElement('td')

                td.innerHTML= data[i][k]

                // 添加到tr里面

                tr.appendChild(td)

            }

            // 创建删除单元格

            var td = document.createElement('td')

            //

            td.innerHTML = '<a href="javaScript:;">删除</a>'

            tr.appendChild(td)

        }

        // 删除操作

        var as =document.querySelectorAll('a')

        for(var i = 0 ;i < as.length; i++){

            as[i].onclick = function(){

                // 点击a 删除当前a 所在的行 node.removeChild(连接的爸爸的爸爸)

                tbody.removeChild(this.parentNode.parentNode)

            }

        }

    </script>

</body>

动态生成表格

禁止鼠标右键复制文字

    <title>禁止鼠标右键复制文字</title>

</head>

<body>

    <div>

        我是一段不愿意分享的文字

    </div>

    <script>

        var div = document.querySelector('div')

        // contextmenu 主要控制应该何时显示上下文菜单，主要用于程序员默认取消的上下文菜单

        div.addEventListener('contextmenu',function(e){

            // 阻止事件

            e.preventDefault()

        })

        // 禁止鼠标选中 给整个页面添加禁止选中

        document.addEventListener('selectstart',function(e){

            // 阻止事件

            e.preventDefault()

        })

</script>

禁止鼠标右键复制文字

天使跟随案例

<title>天使跟随案例</title>

    <style>

        img {

            position: absolute;

            top: 2px;

        }

    </style>

</head>

<body>

    <img src="./images/angel.gif" alt="">

    <script>

        var imgs = document.querySelector('img')

        // 给页面添加一个鼠标移动事件 mousemove

        document.addEventListener('mousemove',function(e){

            //获得最新的坐标 给图片

          var x = e.pageX

          var y = e.pageY

        //   这里千万注意 一定要跟上单位 px 不然不生效

          imgs.style.left = x - 50+'px'

          imgs.style.top = y -40 +'px'

        })

</script>

天使跟随案例

模拟京东按键输入内容

<title>模拟京东按键输入内容</title>

</head>

<body>

    <input type="text" >

    <script>

        // 1；核心思路：检测用户是否按下了s建，如果按下s建，就把光标定位到搜索栏里面

        // 2；使用键盘事件对象里面keyCode 判断用户按下的是否是s建

        // 3；搜索栏获得焦点：使用js里面的focus（）方法

        var search = document.querySelector('input')

       document.addEventListener('keyup',function(e){

            if(e.keyCode === 83){

                //    search.focus() 搜索框获得焦点

                search.focus()

            }

        })

</script>

模拟京东按键输入内容

模拟京东快递单号查询

<title>模拟京东快递单号查询</title>

    <style>

        \* {

            margin: 0;

            padding: 0;

        }

        .search {

            position: relative;

            width: 178px;

            margin: 100px;

        }

        .con {

            display: none;

            position: absolute;

            top: -40px;

            width: 171px;

            border: 1px solid rgba(0, 0, 0, .2);

            box-shadow: 0 2px 4px rgba(0, 0, 0, .2);

            padding: 5px 0;

            font-size: 18px;

            line-height: 20px;

            color: #333;

        }

        .con::before {

            content: '';

            width: 0;

            height: 0;

            position: absolute;

            top: 28px;

            left: 18px;

            border: 8px solid #000;

            border-style: solid dashed dashed;

            border-color: #fff transparent transparent;

        }

    </style>

</head>

<body>

    <!-- 要求：当我们在文本框中 输入内容是，文本框上面自动显示大字号的内容-->

    <div class="search">

        <div class="con">

            123

        </div>

        <input type="text" placeholder="请输入您的快递单号" class="jd">

    </div>

    <script>

        // 快递单号输入 内容时；上面的大号字体盒子（con）显示（这里面的字号更大）

        // 表单检测用户输入：给表单添加键盘事件

        // 同时把快递单号里面的值（value）获取过来赋值给（con）盒子（innerText）作为内容

        // 如果快递单号里面的内容为空，则隐藏大号盒子（con）

        var con = document.querySelector('.con')

        var input = document.querySelector('.jd')

        input.addEventListener('keyup',function(){

            if(this.value === ''){

                con.style.display = 'none'

            } else {

                con.style.display = 'block'

                con.innerHTML = this.value

            }

        })

    // 当我们失去焦点 就隐藏这个con盒子

          input.addEventListener('blur',function(){

            con.style.display = 'none'

          })

          // 当我们获得焦点 并且文本框内容不为空，就显示这个con盒子

          input.addEventListener('focus',function(){

            if(this.value !== ''){

                con.style.display = 'block'

            }

          })

        // 注意这里用keyup更合适 keydown和keypress在文本框里面的特点：他们俩个事件触发的时候，文字还没有落入文本框中

        // keyup事件触发的时候，文字已经落入文本框里面了

</script>

模拟京东快递单号查询

5秒钟自动关闭广告

<title>5秒钟自动关闭广告</title>

</head>

<body>

    <img src="./images/ad.jpg" alt="" class="ad">

    <script>

        var ad = document.querySelector('.ad')

        // 核心思路：5秒钟之后，就把这个广告隐藏起来

        setTimeout(function(){

            ad.style.display = 'none'

        },5000)

</script>

5秒钟自动关闭广告

清除 setInterva定时器

<title>清除 setInterva定时器</title>

</head>

<body>

    <button class="begin">开启定时器</button>

    <button class="stop">停止定时器</button>

    <script>

        var begin = document.querySelector('.begin')

        var stop = document.querySelector('.stop')

        var timer = null //把timer 定义成一个全部变量 其他函数调用都可以使用

        // null是一个空的对象 定时器也是一个对象 把定时器赋值给timer 就可以清除了

        // 如果在函数里面定义 就会变成局部变量 超过大括号以外的函数是访问不到的

        begin.addEventListener('click', function () {

            timer = setInterval(function () {

                console.log('停止定时器');

            }, 1000)

        })

        stop.addEventListener('click', function () {

            clearInterval(timer)

        })

</script>

清除 setInterva定时器

发送短信案例

<title>发送短信案例</title>

</head>

<body>

    <input type="number"> <button>发送</button>

    <!-- 1；按钮点击之后，会禁用 disabled 为true -->

    <!-- 2；同时按钮里面的内容会变化，注意button 里面的内容通过innerHTML修改 -->

    <!-- 3 ；里面的秒数是有变化的，因此需要用到定时器 -->

    <!-- 4；定义一个变量，在定时器里面，不断递减 -->

    <!-- 5；如果变量为0 说明时间到了，我们需要停止定时器，并且复原按钮初始状态 -->

    <script>

        var btn = document.querySelector('button')

        var time = 3 //定义剩下的秒数

        btn.addEventListener('click', function () {

            btn.disabled = true

            var timer =   setInterval(function () {

                if(time ==  0){

                    // 当剩下的秒数等于0 的时候

                    // 清除定时器跟复原按钮

                    clearInterval(timer)

                    btn.disabled = false

                    btn.innerHTML = '发送'

                    time = 3

                } else {

                    btn.innerHTML = '还剩下' + time + '秒'

                time--

                }

            }, 1000)

        })

</script>

发送短信案例

5秒钟之后跳转页面

<title>5秒钟之后跳转页面</title>

</head>

<body>

    <button>点击</button>

    <div></div>

    <script>

        var btn = document.querySelector('button')

        var div = document.querySelector('div')

        btn.addEventListener('click', function () {

            // console.log(window.location.href)

            location.href = 'http://www.itcast.cn'

        })

        var timer = 5

        setInterval(function () {

            if (timer === 0) {

                location.href = 'http://www.itcast.cn'

            } else {

                div.innerHTML = '您将在' + timer + '秒钟之后跳转到首页'

                timer--

            }

        }, 1000)

</script>

5秒钟之后跳转页面

获取url参数数据

   <form action="index.html">

提交的时候不要忘记带name 这里提交方式get提交

        用户名： <input type="text" name="uname">

        <input type="submit" value="登录">

    </form>

<title>获取url参数数据</title>

    <!-- 主要练习参数在不同页面中的传递 -->

</head>

<body>

    <div></div>

    <script>

        console.log(location.search) // ?uname=andy

        // 1.先去掉？  substr('起始的位置'，截取几个字符);

        var params = location.search.substr(1) // uname=andy

        console.log(params);

        // 2. 利用=把字符串分割为数组 split('=');

        var arr = params.split('=')

        console.log(arr); // ["uname", "ANDY"]

        var div = document.querySelector('div')

        // 3.把数据写入div中

        div.innerHTML = arr[1] + '欢迎您'

    </script>

</body>

获取url参数数据

获取鼠标在盒子的位置

<title>获取鼠标在盒子的位置</title>

    <style>

   \*{

            margin: 0;

            padding: 0;

        }

        .box {

            width: 200px;

            height: 200px;

            background-color: palegreen;

            margin: 200px;

        }

    </style>

</head>

<body>

    <div class="box"></div>

    <script>

        // 我们在盒子内点击，想要得到鼠标距离盒子左右的距离

        // 首先得到鼠标在页面中的坐标（e.pageX.e.pageX)

        // 其次得到盒子在页面中的距离（box.offsetTop  box.offsetLeft)

        // 用鼠标距离页面的坐标减去盒子在页面中的距离，得到鼠标在盒子内的坐标

        // 如果想要移动鼠标，就要获得最新的坐标，使用鼠标移动事件mousemove

        var box = document.querySelector('.box')

        box.addEventListener('mousemove',function(e){

            // console.log(e.pageX)

            // console.log(e.pageY)

            // console.log(box.offsetTop)

            // console.log(box.offsetLeft)

            var x = e.pageX - this.offsetTop

            var y = e.pageY - this.offsetLeft

            this.innerHTML = 'x坐标是'+ x +' y坐标是'+ y

        })

</script>

获取鼠标在盒子的位置

 模态框拖拽

<title>模态框拖拽</title>

    <style>

        .login-header {

            width: 100%;

            text-align: center;

            height: 30px;

            font-size: 24px;

            line-height: 30px;

        }

        ul,

        li,

        ol,

        dl,

        dt,

        dd,

        div,

        p,

        span,

        h1,

        h2,

        h3,

        h4,

        h5,

        h6,

        a {

            padding: 0px;

            margin: 0px;

        }

        .login {

            display: none;

            width: 512px;

            height: 280px;

            position: fixed;

            border: #ebebeb solid 1px;

            left: 50%;

            top: 50%;

            background: #ffffff;

            box-shadow: 0px 0px 20px #ddd;

            z-index: 9999;

            transform: translate(-50%, -50%);

        }

        .login-title {

            width: 100%;

            margin: 10px 0px 0px 0px;

            text-align: center;

            line-height: 40px;

            height: 40px;

            font-size: 18px;

            position: relative;

            cursor: move;

        }

        .login-input-content {

            margin-top: 20px;

        }

        .login-button {

            width: 50%;

            margin: 30px auto 0px auto;

            line-height: 40px;

            font-size: 14px;

            border: #ebebeb 1px solid;

            text-align: center;

        }

        .login-bg {

            display: none;

            width: 100%;

            height: 100%;

            position: fixed;

            top: 0px;

            left: 0px;

            background: rgba(0, 0, 0, .3);

        }

        a {

            text-decoration: none;

            color: #000000;

        }

        .login-button a {

            display: block;

        }

        .login-input input.list-input {

            float: left;

            line-height: 35px;

            height: 35px;

            width: 350px;

            border: #ebebeb 1px solid;

            text-indent: 5px;

        }

        .login-input {

            overflow: hidden;

            margin: 0px 0px 20px 0px;

        }

        .login-input label {

            float: left;

            width: 90px;

            padding-right: 10px;

            text-align: right;

            line-height: 35px;

            height: 35px;

            font-size: 14px;

        }

        .login-title span {

            position: absolute;

            font-size: 12px;

            right: -20px;

            top: -30px;

            background: #ffffff;

            border: #ebebeb solid 1px;

            width: 40px;

            height: 40px;

            border-radius: 20px;

        }

    </style>

</head>

<body>

    <div class="login-header"><a id="link" href="javascript:;">点击，弹出登录框</a></div>

    <div id="login" class="login">

        <div id="title" class="login-title">登录会员

            <span><a id="closeBtn" href="javascript:void(0);" class="close-login">关闭</a></span>

        </div>

        <div class="login-input-content">

            <div class="login-input">

                <label>用户名：</label>

                <input type="text" placeholder="请输入用户名" name="info[username]" id="username" class="list-input">

            </div>

            <div class="login-input">

                <label>登录密码：</label>

                <input type="password" placeholder="请输入登录密码" name="info[password]" id="password" class="list-input">

            </div>

        </div>

        <div id="loginBtn" class="login-button"><a href="javascript:void(0);" id="login-button-submit">登录会员</a></div>

    </div>

    <!-- 遮罩层 -->

    <div id="bg" class="login-bg"></div>

    <script>

        var login = document.querySelector('.login')

        var login\_bg = document.querySelector('.login-bg')

        var link = document.querySelector('#link')

        var closeBtn = document.querySelector('#closeBtn')

        var title = document.querySelector('#title')

    //   2;点击弹出层这个链接，link 让login\_bg遮罩层   login登录页面显示出来

        link.addEventListener('click',function(){

            login\_bg.style.display = 'block'

            login.style.display = 'block'

        })

        // 3;点击 关闭按钮   closeBtn  让login\_bg遮罩层   login登录页面隐藏出来

        closeBtn.addEventListener('click',function(){

            login\_bg.style.display = 'none'

            login.style.display = 'none'

        })

        // 4；在页面中拖拽的原理：鼠标按下mousedown 鼠标移动 mousemove 松开鼠标mouseup

        // 要三个事件一起搭配才能完成

        // 拖拽过程：鼠标移动过程中。获得最新的值赋值给模态框的left和top值，这样模态框就可以跟着鼠标走了

        // 鼠标按下触发的事件源是最上面一行，就是id为title

        // 鼠标的坐标减去鼠标在盒子内的坐标，才是模态框真正的位置

        // 鼠标按下的时候，得到鼠标在盒子内的坐标

        title.addEventListener('mousedown',function(e){

            var x = e.pageX - login.offsetLeft

            var y = e.pageY - login.offsetTop

            // 只有按下才会触发移动事件 一定要写在按下事件里面

            // 2;鼠标移动的时候，吧鼠标在页面中的坐标，减去

            // 鼠标在盒子内的坐标就是模态框的left和top

            document.addEventListener('mousemove',move)

            function move(e){

                login.style.left = e.pageX - x + 'px'

                login.style.top = e.pageY - y + 'px'

            }

             // 鼠标松开；就停止拖拽，就是可以让鼠标事件解除

             document.addEventListener('mouseup',function(){

                document.removeEventListener('mousemove',move)

             })

        })

    </script>

</body>

模态框拖拽

仿淘宝固定右侧侧边栏

 <title>仿淘宝固定右侧侧边栏</title>

    <!-- 原先是侧边栏是绝对定位 -->

    <!-- 当页面滚动到一定位置，侧边栏改为固定定位 -->

    <!-- 页面继续滚动，会让返回顶部显示出来 -->

    <style>

        .slider-bar {

            position: absolute;

            left: 50%;

            top: 300px;

            margin-left: 600px;

            width: 45px;

            height: 130px;

            background-color: pink;

        }

        .w {

            width: 1200px;

            margin: 10px auto;

        }

        .header {

            height: 150px;

            background-color: purple;

        }

        .banner {

            height: 250px;

            background-color: skyblue;

        }

        .main {

            height: 1000px;

            background-color: yellowgreen;

        }

        span {

            display: none;

            position: absolute;

            bottom: 0;

        }

    </style>

</head>

<body>

    <div class="slider-bar">

        <span class="goBack" >返回顶部</span>

    </div>

    <div class="header w">头部区域</div>

    <div class="banner w">banner区域</div>

    <div class="main w">主体部分</div>

    <script>

        //1. 获取元素

        var sliderbar = document.querySelector('.slider-bar');

        var banner = document.querySelector('.banner');

        // banner.offestTop 就是被卷去头部的大小 一定要写到滚动的外面

        var bannerTop = banner.offsetTop

            // 当我们侧边栏固定定位之后应该变化的数值

        var sliderbarTop = sliderbar.offsetTop - bannerTop;

        // 获取main 主体元素

        var main = document.querySelector('.main');

        var goBack = document.querySelector('.goBack');

        var mainTop = main.offsetTop;

        // 2. 页面滚动事件 scroll

        document.addEventListener('scroll', function() {

            // console.log(11);

            // window.pageYOffset 页面被卷去的头部

            // console.log(window.pageYOffset);

            // 3 .当我们页面被卷去的头部大于等于了 172 此时 侧边栏就要改为固定定位

            if (window.pageYOffset >= bannerTop) {

                sliderbar.style.position = 'fixed';

                sliderbar.style.top = sliderbarTop + 'px';

            } else {

                sliderbar.style.position = 'absolute';

                sliderbar.style.top = '300px';

            }

            // 4. 当我们页面滚动到main盒子，就显示 goback模块

            if (window.pageYOffset >= mainTop) {

                goBack.style.display = 'block';

            } else {

                goBack.style.display = 'none';

            }

        })

</script>

仿淘宝固定右侧侧边栏

动画函数封装

一定加定位

<  <button>点击夏雨荷才走</button>

    <div></div>

    <span>夏雨荷</span>

    <script>

        // obj是个对象

        // var obj = {

        //     obj.name= 'andy'

        // }

        // 简单动画函数封装 obj目标对象 target 目标位置

     function animate(obj,target){

        //当我们不断的点击按钮。这个元素的速度会越来越快，因为开启了太多了定时器

        // 先清除以前的定时器，只保留当前的一个定时器

        clearInterval(obj.timer)

    // 给不同的元素指定不同的定时器

        obj.timer = setInterval(function () {

            if (obj.offsetLeft >= target) {

                // 停止动画 本质是定时器

                clearInterval(obj.timer)

            }

            //获得当前位置

            // offsetLeft 是获取  style是修改元素

            obj.style.left = obj.offsetLeft + 5 + 'px'

        }, 30)

     }

     var div = document.querySelector('div')

     var span = document.querySelector('span')

    var btn = document.querySelector('button')

     animate(div,300)

    btn.addEventListener('click',function(){

        animate(span,600)

    })

    </script>

</script>

动画函数封装

一定加定位

动画函数封装 -缓动动画

<title>动画函数封装 -缓动动画</title>

    <!-- 封装动画函数：注意函数需要传递2个参数。动画对象和移动到的距离 -->

    <!-- 缓动动画就是让元素运动速度有所变化，最常见的是让速度慢慢停下来 -->

    <!-- 思路：1；让盒子每次移动的距离慢慢变小，速度就会慢慢落下来

         2；核心算法 （目标值-现在的位置）/ 10 作为每次移动的距离步长-->

         <!-- 停止的条件是：让当前盒子位置等于目标位置就停止定时器 -->

         <style>

            div {

                position: absolute;

                left: 0;

                width: 100px;

                height: 100px;

                background-color: plum;

            }

            span {

                position: absolute;

                left: 0;

                display: block;

                width: 150px;

                height: 150px;

                background-color: peru;

            }

        </style>

</head>

<body>

    <button class="btn500">点击夏雨荷到500</button>

    <button class="btn800">点击夏雨荷到800</button>

    <!-- <div></div> -->

    <span>夏雨荷</span>

    <script>

           function animate(obj, target) {

            //当我们不断的点击按钮。这个元素的速度会越来越快，因为开启了太多了定时器

            // 先清除以前的定时器，只保留当前的一个定时器

            clearInterval(obj.timer)

            // 给不同的元素指定不同的定时器

            obj.timer = setInterval(function () {

                // 步长值写在定时器的里面

                //把我们步长值 取整 往上取整

                // var step = Math.ceil((target - obj.offsetLeft) / 10)

                var step = (target - obj.offsetLeft) / 10

                // 判断如果你是大于0 就执行取整往大了取 小于0就往小了取

              step  =  step > 0 ? Math.ceil(step): Math.floor(step)

                if (obj.offsetLeft == target) {

                    // 停止动画 本质是定时器

                    clearInterval(obj.timer)

                }

                //获得当前位置

                // offsetLeft 是获取  style是修改元素

                // 把每次加1 这个步长值改为一个慢慢变小的值

                // （目标值-现在的位置）/ 10 作为每次移动的距离步长

                obj.style.left = obj.offsetLeft + step + 'px'

            }, 15)

        }

        var div = document.querySelector('div')

        var span = document.querySelector('span')

        var btn500 = document.querySelector('.btn500')

        var btn800 = document.querySelector('.btn800')

        // animate(div, 300)

        btn500.addEventListener('click', function () {

            animate(span, 500)

        })

        btn800.addEventListener('click', function () {

            animate(span, 800)

        })

        // 匀速动画 就是盒子是当前的位置 + 固定的值 10

        //缓动动画就是 盒子当前的位置 + 变化的值（目标值-现在的位置）/ 10

</script>

动画函数封装 -缓动动画

引入animate封装函数

<script src="./aninate.js"></script>

    <style>

        .sliderbar {

            position: fixed;

            right: 0;

            bottom: 100px;

            width: 40px;

            height: 40px;

            text-align: center;

            line-height: 40px;

            cursor: pointer;

            color: #fff;

        }

        .con {

            position: absolute;

            left: 0;

            top: 0;

            width: 200px;

            height: 40px;

            background-color: purple;

            z-index: -1;

        }

    </style>

</head>

<body>

    <div class="sliderbar">

        <span>←</span>

        <div class="con">问题反馈</div>

    </div>

    <script>

        // 当我们鼠标经过 sliderbar 就会让 con 这个盒子滑动到左侧

        // 当我们鼠标离开 sliderbar 就会让 con 这个盒子滑动到右侧

        var sliderbar = document.querySelector('.sliderbar')

        var con = document.querySelector('.con')

        sliderbar.addEventListener('mouseenter',function(){

            // animate(obj,target,callback)

            animate(con,-160 ,function(){

                // 当我们动画执行完毕 就把左箭头← 改为右 →

                sliderbar.children[0].innerHTML = '→'

            })

        })

        sliderbar.addEventListener('mouseleave',function(){

            // animate(obj,target,callback)

            animate(con,0,function(){

                sliderbar.children[0].innerHTML = '←'

            })

        })

</script>

引入animate封装函数

移动端拖动效果

 <title>移动端拖动效果</title>

    <!-- 移动端拖动的原理：手机移动中，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 -->

    <!-- 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 -->

    <style>

        div {

            position: absolute;

            left: 0;

            width: 100px;

            height: 100px;

            background-color: powderblue;

        }

    </style>

</head>

<body>

    <div></div>

    <script>

        // 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   计算手机的滑动距离，并且移动盒子

        // 3；离开手指 touchend

        var div = document.querySelector('div')

        var startX = 0  // 获取手指初始坐标

        var startY = 0

        var x = 0  //获取盒子原来的位置

        var y = 0

        div.addEventListener('touchstart',function(e){

            // 获取手指初始坐标

            startX = e.targetTouches[0].pageX

            startY = e.targetTouches[0].pageY

            // 获取盒子原来的位置

            x = this.offsetLeft

            y = this.offsetTop

        })

        div.addEventListener('touchmove',function(e){

            // 计算手指的移动距离 ：手指移动之后的坐标 减去 手指初始的坐标

          var moveX = e.targetTouches[0].pageX -  startX

          var moveY = e.targetTouches[0].pageY -  startY

        //   移动我们的盒子 盒子原来的位置+手指移动的距离

        this.style.left = x + moveX  + 'px'

        this.style.top = y + moveY + 'px'

        // 注意手指移动也会触发滚动屏幕所以这要阻止默认的屏幕滚动

        e.preventDefault()

        })

</script>

移动端拖动效果