目录

- 阻止默认行为
  - 阻止右键菜单

```
document.oncontextmenu = function(){ //调用菜单的事件
  return false;
}
```

  - 阻止超链接的默认行为（`e.preventDefault()`）

```
 e.preventDefault ? e.preventDefault() : e.returnValue = false;
```

- 获取当前页面的宽高
- 事件委托
- 事件监听器：
  - `node.addEventListener("click",F,布尔值)`
  - `node.removeEventListener("click",函数名)`
- 正则表达式
- `localStorage`
  - `setItem(name , value)`
  - `getItem(name)`
  - `removeItem(name)`

1.阻止默认行为

- 阻止右键菜单

```
document.oncontextmenu = function(){ //调用菜单的事件
  return false;
}
```

  - 实现自定义右键菜单，鼠标按下；按下的是右键，在右键这个位置显示菜单，如果按下的是别的键，让菜单消失
- 阻止超链接的默认行为
  - 简陋的阻止a链接的默认行为

```
// 缺点：运行到了return,后续的内容就执行不到了
a.onclick = function(){
  return false;
}
```

  - 规范的方法

```
e.preventDefault();   // 不考虑IE，直接使用此方法
e.returnValue = false; // IE
```

- 三目运算符 / if语句解决兼容性

```
e.preventDefault ? e.preventDefault() : e.returnValue = false;
// 后续的操作得以执行
```

- 不考虑IE使用`event.preventDefault()`阻止a链接的默认行为，放在作用域最上方

2.实现拖拽

- `node.onmousedown`
  - 按下记录鼠标和被拖拽物体的相对距离相对位置
    - 相对距离是不会改变的
- `node.onmouseup`
  - 鼠标抬起 让被拖拽物体停止移动
- `node.onmousemove`

3.当前页面的高和宽(为了限制拖拽出界面)

```
//获取当前页面的宽(body和html)  兼容写法
var windowWidth = document.documentElement.clientWidth || document.body.clientWidth;
var windowHeight = document.documentElement.clientHeight || document.body.clientHeight;
```

- `document.body`是DOM中`Document`对象里的`body`节点，`document.documentElement`是文档对象根节点(`html`)的引用。

- 扩展：
  - `node.offsetWidth`
    - 对象整体的实际宽度，包滚动条等边线，会随对象显示大小的变化而改变。
      - `content + padding + border` (包括滚动条)

  - `node.offsetTop/offsetLeft`
    - 获取元素和第一个有定位的父节点之间的距离;没有定位，则计算和浏览器的距离
      - 计算`node`的`border`距离定位元素之间的距离，`node`的`margin`也会被计算其中

  - `node.clientWidth`
    - 对象内容的可视区的宽度，不包滚动条等边线，会随对象显示大小的变化而改变。
      - `content + padding` 不包括滚动条

  - `node.clientTop/Left`

        - 边框的宽度

    - `node.scrollWidth`
        - 对象的实际内容的宽度，不包边线宽度，会随对象中内容超过可视区后而变大。
            - `content`(实际内容，包括隐藏区域) + `padding`(根据具体情况)
                - 计算实际内容，内容溢出后，会计算溢出区域的宽，不溢出就是`client`,内容溢出后的溢出边`padding`不计入其中

    - `node.scrollTop/Left`
        - `node.scrollTop()` 方法设置或返回被选元素的**垂直滚动条位置**。
            - 当滚动条位置位于最顶部时，位置是0；
            - 获取值时：该方法返回 元素的滚动条的垂直位置。
            - 设置位置时：该方法设置 元素的滚动条的垂直位置到指定区域。
        - 滚动条占用的是节点内部区域
        - 使用`localstorage`，记录滚动条的位置,赋值时调用`localstorage`。
- 获取浏览器窗口的宽和高(获取的就是 内容窗口高度，不包含页面距离)
  `window.innerHeight/innerWidth`

4.事件委托

- 添加点击事件时两个常见问题
    1. 给五个li标签添加了一模一样的函数，浪费资源
    2. 后添加的节点没有点击事件。(通过后续用户操作添加)

- 事件委托实现步骤：
    1. 找到当前节点中的父节点或祖先节点；
    2. 将事件添加到你找到的父节点或祖先节点上
    3. 找到触发对象，判断触发对象是否是我们想要的触发对象，进行后续操作
- 总结：事件委托就是通过**给祖先节点添加事件**，**找到目标子节点target**，**来对对应的子节点进行操作**

5.Event事件监听器

- 传统事件绑定不能满足的功能
    1. 重复添加，覆盖
    2. 不能精确的删除事件上的某一个函数

- 事件监听器（低版本IE浏览器不兼容）
    - `node.addEventListener()`
        - 格式：`node.addEventlistener("click",函数名/匿名函数,布尔值)`
            - 布尔值：`true` 事件捕获，`false` 事件冒泡(默认)
    - `node.removeEventListener()`

        - 格式：`node.removeEventListener("click",函数名)`
            - 删除事件对应的具体函数
        - 函数名不能带括号进行传参，带括号就是调用函数, 不带括号就是复制函数地址

    - 低版本IE事件监听器
        - `attachEvent()` `detachEvent()`
            - 兼容函数看(5_4事件监听器兼容函数封装.html)
- 事件监听器的作用：
    1. 给一个节点添加多个功能，又不想让功能互相干扰，可以通过事件监听器，给不同的功能添加事件函数。
    2. 可以精确的删除事件上的某一个函数。

- 事件捕获的细节：一个嵌套结构，有的层添加事件捕获(true)，有的层添加事件冒泡(false)，会先由外向里触发事件捕获，再由里向外触发事件冒泡。(具体看5_Event事件监听器 / 优先级测试)
- **事件监听器中的this指向** -> 添加事件的元素节点

6.动态生成表格

- 输入行列，生成对应行和列的表格
    - 思路：利用循环，先生成行，在行里面生成对应的单元格

7.放大镜案例

- 放大镜是反方向移动放大的图片，造成映射假象
    - 两个`div`容器, 移入正常图片时, 另一个放大版的`div`计算背景图位置

8.正则表达式

- 正则表达式也是一个对象
- 正则表达式(regular expression)是一个描述字符模式的对象，ECMAscript 的 RegExp 类表示正则表达式，而String 和 正则表达式都定义了进行强大的【模式匹配】和【文本检索】与【替换】的函数。

- 正则表达式的**创建**
    1. 通过`new`去声明正则表达式
        - 格式：`var box1 = new RegExp("hello","ig");`
            - 第一个参数：正则表达式主体 字符串
            - 第二个参数：修饰符 `i`忽略大小写 `g`全局匹配 `m`换行匹配
                - 修饰符没有顺序
    2. 省略`new`运算符去声明正则表达式
    3. 通过常量进行赋值 `/'hello'/ig`

- 正则表达式**方法**
  - `/reg/ig.test('str')`
    - 格式：`正则.test(字符串)`
    - 功能：**在字符串中匹配这个正则是否存在**
    - 返回值：匹配**成功**返回`true`，匹配**失败**返回`false`。类似于字符串查找
  - `/rrg/ig.exec('str')`
    - 格式：`正则.exec(字符串)`
    - 功能：**在字符串中匹配这个正则是否存在**
    - 返回值：匹配**成功**，**返回一个装有第一个匹配成功字符串的数组,匹配失败，返回null**
    - `exec()`执行匹配的时候，会先返回整体匹配值，再返回按照正则表达式中 由括号扩起来的小分组进行匹配的值，

```
/(a+)(b+)/.exec('aaabbb') // aaabbb , aaa , bbb
// ?:可以取消返回该分组的匹配值。
/(a+)(?:b+)/.exec('aaabbb') // aaabbb aaa
```

- **字符串**中**可以使用正则的方法**
  - `match()`
    - 格式：`字符串.match(正则);`
    - 功能：在字符串中匹配是否有符合的正则表达式
    - 返回值：匹配成功，返回装有匹配到子串的数组, 匹配失败 返回null
  - `replace()`
    - 格式：`字符串.replace(oldstr/正则,newstr);`
    - 功能：用新字符串将旧字符串替换掉 正则不加g修饰符也只能替换第一个
    - 返回值：替换成功的新字符串
  - `search()`
    - 格式：`字符串.search(字符串/正则)`
    - 功能：找到符合条件的子串第一次出现的位置
    - 返回值：找到，返回子串下标 没找到，返回-1。
  - `split()`
    - 格式：`字符串.split(str/正则 , length);`
    - 功能：将原有的字符串用传入的分隔符分割
    - 返回值：分割完毕的子串组成的数组
    - 参数：第一个是分隔符，第二个是要返回数组的长度，一般不传入。

9.正则表达式-元字符

- 元字符：在正则表达式中有特殊含义的字符

- 字符类：**单个字符**和**数字的元字符**

  `.`：匹配除换行符外的任意单个字符

`[]` ： 匹配单个范围内字符
`[0-9]` ： 匹配 `0-9`
`[a-cA-S_/]` ： 匹配 `a-c` 和 `A-S` 和 `_/`
`[^]` ： 匹配任意一个除括号范围内的字符
`[^a-z0-9]` ： 匹配除了 `0-9` 和 `a-z` 的单个字符

`\w` ： 匹配字母和数字及 `_` 等价于 `[a-zA-Z0-9_]`
`\W` ： 匹配非字母和数字及 `_` 等价于 `[^a-zA-Z0-9_]`
`\d` ： 匹配数字 等价于 `[0-9]`
`\D` ： 匹配非数字，同 `[^0-9]`

- 字符类：**重复字符** (x代表任意的单个字符)
  `x?` ： 匹配 `0` 个或 `1` 个 `x`
  `x+` ： 匹配至少 `1` 个 `x`
  `x*` ： 匹配 `0` 个或任意多个 `x`
  `x{m,n}` ： 匹配最少 `m` 个、最多 `n` 个 `x` ,包括 `n`
  `x{n}` ： 必须匹配 `n` 个 `x` 字符
  `(xyz)+` ： 小括号括起来的部分当做单个字符去处理
- 字符类：**空白字符**
  `\s` ： 匹配任意单个空白字符 空格、制表符和换行符
  `\S` ： 匹配任意单个非空白字符
  （以下了解、不用记忆）
  `\0` ： 匹配null字符
  `\b` ： 匹配空格字符
  `\f` ： 匹配进纸字符
  `\n` ： 匹配换行符
  `\r` ： 匹配回车字符
  `\t` ： 匹配制表符
  `\n` 是换行，是 ` ` 同类型的转义字符

- 字符类：**锚字符**
  `^` ： 行首匹配 必须以这个正则开头 `/^goo/`
  `$` ： 行尾匹配 必须以这个正则结尾 `/gle$/`
  `/^google$/` 一个 `google` 在行首，同一个 `google` 在行尾，匹配的必须是这个字符串

- 字符类：**替代字符**
  `this|where|logo` ： 匹配 `this` 或 `where` 或 `logo` 中任意一个

- 正则**修饰符**

- `i` ： 忽略大小写 `g` ： 全局匹配 `m` ： 换行匹配
  - 如果在字符串中，遇到换行，重新开始计算行首。

10.正则练习

1. 检查压缩包文件名
   - 转义字符：
     `\.` 代表本来 `.` 字符的意思
     `\*` 代表本来 `*` 字符的意思
     `\+` 代表本来 `+` 字符的意思
2. 删除多余空格
3. 删除首尾空格
4. 手机号码
5. 验证是否是纯中文
   `[\u4e00-\u9fa5]` 代表单个字符的中文范围

- 正则表单验证：用户名和密码强度的验证

11.localStorage(本地存储)

- 本地存储技术:
  - `localStorage` (IE8以下不兼容)
    1. 永久存储（除非主动删除）
    2. 最大存储 `5M` ，客户端的一个微型数据库
    3. 只能去存储字符串 `string` (放入什么类型的数据都会被自动转换为字符串)
       /下面了解，不重要的点/
    4. `localStorage` 在浏览器隐私模式下是不可读取的
    5. `localStorage` 本质上是对字符串的读取,如果存储内容多的话，会消耗内存空间，导致页面变卡。
    6. `localStorage` 不能被爬虫抓取到
  - `cookie`
    1. 可以设置过期时间
    2. 最大存储 `4KB`
    3. 每一个域名下最多可以存储 `50` 条
  - `sessionStorage` （结合后台使用）

- `localStorage` 的方法
  - `setItem(name , value)` ： 存储值

```
localStorage.setItem("a","1");
localStorage.b = "2";
localStorage["c"] = "3";
```

    - `value` 不管写什么值,都会被转换为字符串存储。
  - `getItem(name)`： 获取值

```
localStorage.getItem("a")
localStorage.b
localStorage['c']
```

  - `removeItem(name)`： 删除值
    - `localStorage.removeItem("a");`
- 通过 `localStorage` 记录滑动条的位置
  - 移动时记录位置上传，当页面刷新时，判断是否有 `localStorage` .记录值,来还原滑块位置
- 扩展: `for...in` 遍历 `localStorage` 时, 会遍历到 其 对象的方法,使用 `Object.keys( obj )`
  - `Object.keys( obj )`：将 `obj` 中 的属性,放入数组中返回

```
var keys = Object.keys( obj );
for(var i = 0; i < keys.length; i++){
  console.log( obj[ keys[i] ] )
}
```