# JavaScript-中阶-正则表达式和事件进阶-DAY03

## 课程介绍

* **正则表达式（了解）**
* **正则表达式案例**
* **注册验证应用**
* **事件对象（了解）（什么是事件对象 事件对象属性 事件对象方法）**
* **DOM事件流（了解）【三个阶段】**
* **事件委派**

1. 正则表达式（了解）
   1. 什么是正则表达式

**正则表达式：是描述字符模式的对象，以一个用简单方式描述或者匹配一系列符合某个句法规则的字符串。**

**正则表达式通常被用来验证、检索、替换那些符合某个模式的文本。**

<https://www.runoob.com/js/js-regexp.html>

* 1. 正则表达式能做什么

1. 检查字符串是否合法性
   * 1. 验证用户名 （a-z，0-9，不能全是数字，不能全是字母）
     2. 验证邮箱格式 (xxx@qq.com)
     3. 验证电话号码 (11位数字)
     4. 验证身份证 (18位 )
     5. 验证QQ号码格式（5-12纯数字，第一位不能为0）；
2. 提取字符串中信息
   * 1. 提取一条短信中数字；
     2. 提取文件名的后缀；
3. 替换字符串
   * 1. 替换字符串中的非法字符；
     2. 对电话号码进行屏蔽；（1852\*\*\*\*0102）
     3. 替换占位符 “hello {{name}} ” hello 王老二 （模板框架）
   1. 正则表达式使用

正则表达式是一种字符模式**对象（RegExp）**；

* + 1. 定义方式

* **使用new字符实例化正则表达式对象**

**let reg = new RegExp("字符串",[模式修饰符])**

正则表达式(regular expression)

* **使用字面量方式实例化正则表达式对象**

**let reg = /字符串/[模式修饰符]**

* + 1. 正则相关的方法

RegExp 对象方法

|  |  |  |  |
| --- | --- | --- | --- |
| **方法** | **描述** | **FF** | **IE** |
| exec | 检索字符串中指定的值。返回找到的值，并确定其位置。 | 1 | 4 |
| test | 检索字符串中指定的值。返回 true 或 false。 | 1 | 4 |

* + 1. 案例

案例一

   /\*

      //需求： 验证字符串中是否含有数字 7

      结果为Boolean值

  \*/

    //定义字符串

*let* str = "wwwwwww7wwwwwwww";

    //定义验证规则

    //let reg = new RegExp("7");

*let* reg = /7/;

    //验证是否存在

*let* result = reg.test(str);

    //访问结果

    console.log(result);//true

案例二

    //需求： 验证字符串str中是否含有指定字符  box

    //定义str字符串

*let* str = "this is a box!!!";

    //定义验证规则

*let* reg = /BOX/i;

    //let reg = new RegExp("BOX","i");

    //i模式修饰符，不区分大小写

    //验证

*let* result = reg.test(str);

    //访问结果

    console.log(result);

案例三

//验证手机号码是否合法

*let* phone="15608495584";

*let* reg=/^1[23456789][\d]{9}$/;

    if(reg.test(phone)){

        console.log("手机号码合法")

    }else{

        console.log("手机号码不合法")

    }

1. 注册界面布局
   1. 界面效果图



* 1. 布局样式

  \*{

*margin*: 0;

*padding*: 0;

        }

        /\* 页面注册按钮 \*/

        #reg-btn{

*background-color*: #f60;

*width*: 100px;

*height*: 35px;

*color*: #fff;

*outline*: none;

*border*: none;

*cursor*: pointer;

        }

        /\* 模态框 遮罩层 \*/

        .modal{

*background-color*:rgba(0, 0, 0, 0.5);

*position*: fixed;

*top*:0;

*right*: 0;

*bottom*: 0;

*left*: 0;

*margin*: auto;

*display*: none;

        }

        /\* 模态框 注册页面 \*/

        .modal #reg{

*width*: 400px;

*height*: 360px;

*background-color*: #fff;

*position*: fixed;

*top*:50%;

*left*: 50%;

*margin-top*: -180px;

*margin-left*: -200px;

        }

        #reg h3{

*text-align*: center;

*padding*: 10px 0;

*border-bottom*: 2px solid #f60;

*color*: #f60;

        }

       #reg div{

*margin*: 20px 0;

*padding*: 0 20px;

       }

       #reg div input{

*display*: block;

*width*: 100%;

*height*: 35px;

*border*: 1px solid #ddd;

*outline*: none;

*box-sizing*: border-box;

*text-indent*: 10px;

*border-radius*: 5px;

       }

       #reg div span{

*font-size*: 12px;

*text-indent*: 10px;

*height*: 16px;

*color*: #666;

       }

       /\* 验证不通过span的样式 \*/

       #reg div span.error{

*color*: red;

       }

       #reg button{

*border*: none;

*outline*: none;

*background-color*: #999;

*width*: 100px;

*height*: 35px;

*color*: #fff;

*cursor*: pointer;

       }

       #reg button:last-child{

*margin-left*: 150px;

*background-color*: #f60;

       }

1. 交互效果
   1. 显示隐藏注册模态框

   //获取页面注册按钮

*let* reg\_btn=document.querySelector("#reg-btn");

    //给页面注册按钮添加点击事件  显示模态框

    reg\_btn.onclick=*function*(){

        //显示模态框

        document.querySelector(".modal").style.display="block";

    }

    //给模态框的关闭按钮添加点击事件

*let* close\_btn=document.querySelector("#reg button");//获取关闭按钮

    close\_btn.onclick=*function*(){

        //隐藏模态框

        document.querySelector(".modal").style.display="none";

    }

* 1. 注册验证
     1. 验证用户名

     // 给用户名添加失去焦点事件 验证用户名是否合法

*let* uname = document.querySelector("#uname");

    uname.onblur = *function* () {

        //获取用户名

*let* userVal = this.value;

        //获取用户名下面的span

*let* span = this.nextElementSibling;

*let* reg = /^[\w\u4e00-\u9fa5]{2,12}$/;//中文、字母、数字、下划线长度2-12位

*let* result = reg.test(userVal);

        //根据验证结果显示提示信息

        if (result) {

            //正确

            span.innerHTML = "";//清除提示内容

            span.classList.remove("error");//删除错误className

            this.style.border = "1px solid green";//把input的边框改成绿色

        } else {

            //错误

            span.innerHTML = "中文、字母、数字、下划线长度2-12位";//写入提示内容

            span.classList.add("error");//给span节点添加className error

            this.style.border = "1px solid red";//把input的边框改成红色

        }

    }

* + 1. 验证邮箱

    // 给邮箱添加失去焦点事件 验证邮箱是否合法

*let* email = document.querySelector("#email");

    email.onblur = *function* () {

        //获取邮箱

*let* emailVal = this.value;

        //获取邮箱下面的span

*let* span = this.nextElementSibling;

*let* reg = /^([a-zA-Z]|[0-9])(\w|\-)+@[a-zA-Z0-9]+\.([a-zA-Z]{2,4})$/;//邮箱正则表达式

*let* result = reg.test(emailVal);

        //根据验证结果显示提示信息

        if (result) {

            //正确

            span.innerHTML = "";//清除提示内容

            span.classList.remove("error");//删除错误className

            this.style.border = "1px solid green";//把input的边框改成绿色

        } else {

            //错误

            span.innerHTML = "有效的邮箱";//写入提示内容

            span.classList.add("error");//给span节点添加className error

            this.style.border = "1px solid red";//把input的边框改成红色

        }

    }

* + 1. 验证密码

   // 给密码添加失去焦点事件 验证密码是否合法

*let* pwd = document.querySelector("#pwd");

    pwd.onblur = *function* () {

        //获取密码

*let* pwdVal = this.value;

        //获取邮箱下面的span

*let* span = this.nextElementSibling;

*let* reg = /^[\w]{6,12}$/;//字母、数字、下划线长度6-12位

*let* result = reg.test(pwdVal);

        //根据验证结果显示提示信息

        if (result) {

            //正确

            span.innerHTML = "";//清除提示内容

            span.classList.remove("error");//删除错误className

            this.style.border = "1px solid green";//把input的边框改成绿色

        } else {

            //错误

            span.innerHTML = "字母、数字、下划线长度6-12位";//写入提示内容

            span.classList.add("error");//给span节点添加className error

            this.style.border = "1px solid red";//把input的边框改成红色

        }

    }

* + 1. 设置标杆和表单提交事件

//定义3个全局标杆 用户名 邮箱 密码 默认false。 验证通过了值改为true.没有通过值改为false

*let*  uFlag=false,eFlag=false,pFlag=false;

 //给表单添加提交事件函数

*function* checkFlied(){

        if(!(pFlag&&uFlag&&eFlag)){//上面的正则表达式没有通过

            //自动触发表单的失去焦点事件

            uname.onblur();

            pwd.onblur();

            email.onblur();

            return false;

        }

        alert("恭喜您，注册成功！");

        //刷新页面

        history.go(0);

    }

1. 事件对象（了解）
   1. 什么是事件对象

概念：就是当你触发了一个事件以后，对该事件的一些描述信息。event 对象代表事件的状态，比如事件在其中发生的元素、键盘按键的状态、鼠标的位置、鼠标按钮的状态。

小结：

1. 事件对象是伴随事件而产生的一个对象
2. 事件对象相当于一个目击者( 旁观者 )，目睹了整个事件发生的过程
   1. 如何使用

标准浏览器(w3c浏览器)：作为事件处理函数中的第一个参数

* + 1. 语法：

**dom节点.addEventListener(“事件类型”,function(event){**

**//event：事件对象**

**})**

**dom节点.onclick=function(event){**

**//event:事件对象**

**}**

* + 1. 属性：

event.target ： 获取事件源

event.type : 获取事件类型

event.clientX：水平坐标

event.clientY：垂直坐标

event.keyCode 键盘码

* + 1. 方法：

event.preventDefault(); //阻止浏览器的默认行为

event.stopPropagation(); //阻止事件传播 === 阻止事件冒泡

1. DOM事件流
   1. 事件流概念：

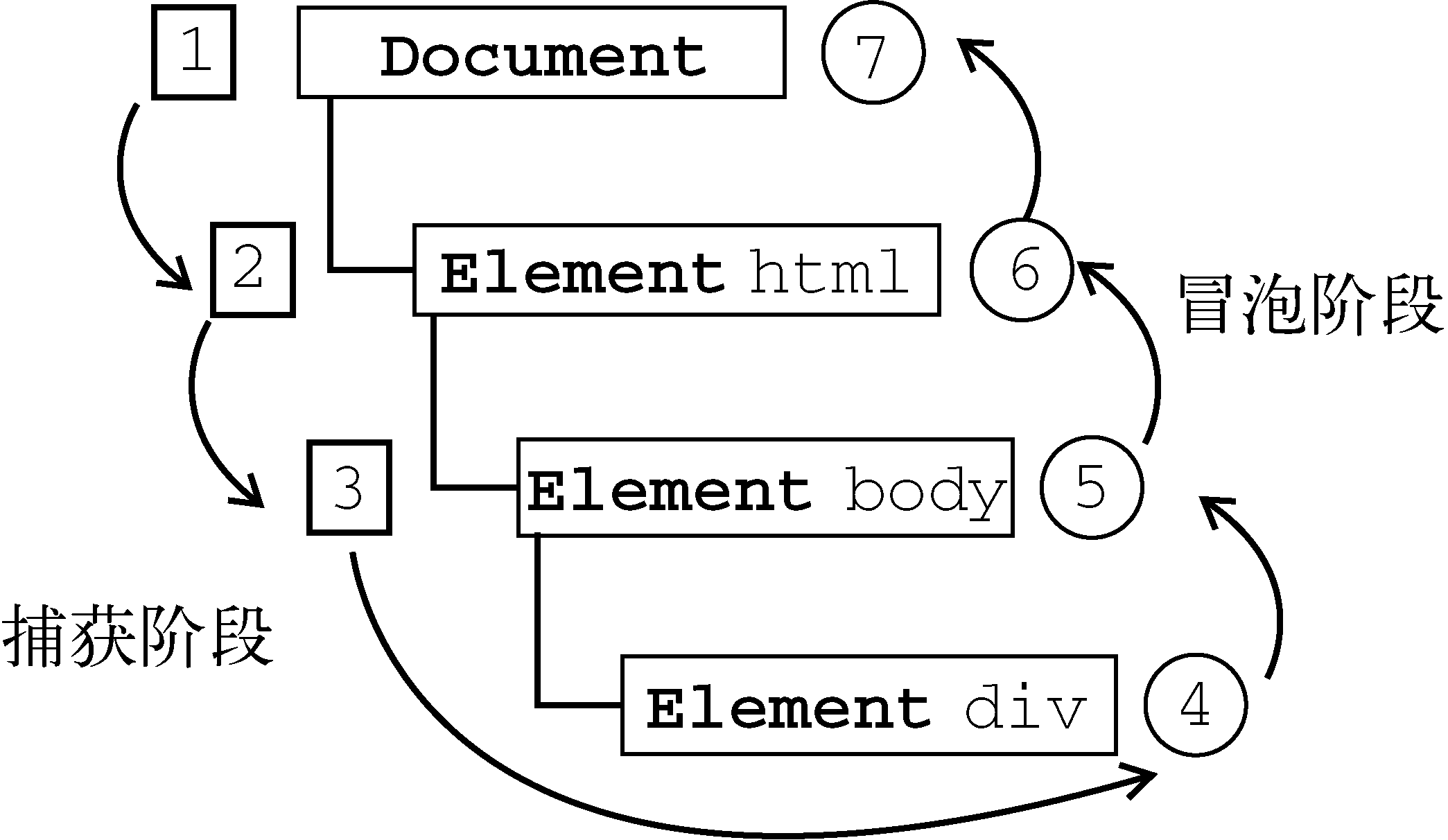
事件流描述元素接收事件的顺序。包括3个阶段：事件捕获阶段、当前目标阶段、冒泡阶段。

当前目标阶段：实际的目标接收到事件

* 1. 事件的捕捉和冒泡(了解)

（1）冒泡：事件从最具体的元素接收，逐级向上传到不具体的节点（window）

（2）捕获：不太具体的节点应该最早接收到事件，最具体的节点最后接收到事件。



向下是捕获1-2-3-4,向上是冒泡5,6,7,8

事件的捕获和冒泡只能够通过注册事件的方式来完成

 元素.addEventListener('事件类型'， 事件处理函数， 冒泡还是捕获)

第三个参数默认是false(冒泡)，如果是true就是捕获

1. 事件委派
   1. 什么是事件的委派？

        事件的委派也叫事件代理，简单理解就是：原本是某个元素需要绑定的事件，现在将这个事件交给别的元素去做，正常我们把它交给父级元素，比如：原本是给li绑定点击事件，现在交给它父级ul绑定，利用冒泡原理，点击li的时候会触发ul的事件；

* 1. 事件委派有什么好处？

        1.提升性能： 比如要给100个li安装点击事件，正常我们会用for循环去个每个li安装，这样与访问dom的次数就多了，性能也就比较低；利用冒泡原理，我们给ul安装点击事件（就是委派ul代理这个事件），点击li的事件由ul去执行，这样我们就只访问一次dom，性能就提升了；

        2.能绑定新添加标签的事件：什么意思？ 比如用for给10个li绑定了事件，然后现在又添加了新的li，这样新添加的li就没有绑定到了；此时我们委派ul去绑定事件，点击新添加li也是会触发这个事件的；

1. 作业
   1. 完成注册案例；

**（难度系数：☆☆☆）**

* 1. 事件对象的属性和方法？

**（难度系数：☆☆☆）**

* 1. 事件委派的好处？

**（难度系数：☆☆☆）**