js二yu

《代码简介之道》

数据类型

【基本数据类型】

- 数字 nameber
- 字符串 string
- 布尔 boolean
- 空 null
- 未定义 undefined
  【引用数据类型】
- 对象 object
  - 普通对象
  - 数组对象 Array
  - 正则对象 RegEXP
  - 日期对象 date
  - 数学函数 math
- 函数 function
  真实项目中，根据需求，我们往往需要把数据类型之间进行转换

把其他数据类型转换为number类型

- 1、发生的情况：
  - issNaN检测的时候：当检测的值不是数据类型，浏览器会自己调用Number方法把它先转化为数字，然后在检测是否为有效数字
  - 基于parseInt/parseFloat去手动转换为数字类型
  - 数学运算：+ - * / % 但是+号不仅仅是数学运算，还可能是字符串拼接

```
var i = "3";
i = i + i;=>"31"
i += 1;=>"31";
i++;=>4  i++就是单纯的数学运算，已经摒弃掉字符串相加的规则
```

```
- 在基于==比较的时候，有时候也会把其他值转换为数字类型
```

- 2、转换规律：
  - 转换方法：Number（浏览器自行转换都是基于这个方法完成的）
  - 【把字符串转换为数字】
    - 只要遇到一个非有效数字字符，结果就是NaN
    - “” 空字符串为0
    - “ ”空格也是0
    - “\n”换行符也是0
    - “\t”制表符也是0
  - 【把布尔转为数字】
    - true ->1
    - false ->0
  - 【把没有转换为数字】
    - null ->0
    - undefined ->NaN
  - 【把引用类型值转为数字】
    - 首先都转换为字符串（toString），然后再转换为数字（Number）

把其他类型值转换为字符串

- 1、发生的情况：
  - 基于alert/confirm/prompt/document.write等方法输出内容的时候，会把输出的值转换为字符串，然后再输出

```
alert（1） =>"1"
```

- 2、转换规律：

关于js数组常用方法的剖析

- 数组也是对象类型的，也是由键值对组成的

```
var ary = [12,23,34];

结构：
0：12
1：23
2：34
length：3
```

- 特点：

    - 1.以数字作为索引（属性名），索引从0开始递增
    - 2.由一个length属性存储的是数组长度
- 获取：
    - art[0] 获取第一项
    - ary[ary.length] 获取最后一项

> 数组中每一项的值可以是任何数据类型的

数组中的常用方法

> 按照四个维度记忆：
>
> - 方法的作用
> - 方法的参数
> - 方法的返回值
> - 原有数组是否改变

push
作用：向数组**末尾**追加新的内容
参数：追加的内容(可以是一个，也可以是多个)
返回值：新增后数组的长度
原有数组改变

```
var ary = [12,23,34]

ary.push(100) =>返回结果4 ary：[12,23,34，100]
```

pop
作用：删除数组最后一项
参数：无
返回：被删除的那一项内容
原有数组改变

```
var ary = [12,23,34]
pop()
34
```

shift（第一项被删除后，后面每一项索引会向前提一位）
作用：删除数组中的第一项
参数：无

返回：被删除的那一项内容
原有数组改变
unshift
作用：
向数组开始位置追加新内容
参数：要新增的内容
返回：新增后数组的长度
原有数据改变

```
var ary = [12,13,14];
ary.unshift(100,true); ->返回5
```

splice
作用：基于splice可以对数组进行很多操作：删除指定位置的内容、向数组指定位置增加内容、还可以修改指定位置的信息
删除：ary.splice（n，m）
从索引n开始删除m个内容，把删除的部分以一个新的数组返回，原有数组改变（如果不指定m就是删除到末尾）
新增：ary.splice（n，0，x）
从索引n开始删除0项，把x或者更多需要插入的内容插入到n的前面,返回空数组（因为没有删除）
修改：ary.splice（n，m，x）
修改的原理就是把原有的m个内容删掉，然后用新的内容x替换这部分信息
利用splice会导致后面的索引向前提如果后面很多项会消耗性能

> 需求扩展：
> 1、删除数组最后一项，有几种办法？
>
> - ary.pop()
> - ary.splice(ary.length-1)
> - ary.length–

> 2、向数组末尾追加新内容，有几种办法？

- ary.posh(100)
- ary.splice(ary.length,0,100)
- ary[ary.length] = 300

slice
作用：在一个数组中，按照条件查找出其中的部分内容
参数：两个参数（ ）
yuanyou

数组查询和字符串转换

Math常用方法

abs：取绝对值
ceil/floor：向上、向下取整
round：四舍五入
sqrt：开平方
pow：取幂（n的m次方）
max、min：获取最大值和最小值
PI：获取圆周率
radom：获取0到1之间的随机小数
Math.round(Math.random()*(m-n)+n):获取n-m之间的随机整数

函数

- 实名函数：有函数名
- 匿名函数：没有函数名
  - 函数表达式：把函数当做值赋值给变量或元素的事件
    - var fn = function（ ）{}
    - oBox.onclick = function（ ）{}
  - 自执行函数：创建和执行一起完成
    - （function （ ）{}）（ ）
    - ~function（ ）{}（ ）
    - +function（ ）{}（ ）
    - ！function（ ）{}（ ）
  - ……

生成四位验证码

需求：生成一个四位随机验证码

- 数字+字母
- 照图片
- 滑动拼图
- 问答类
- 点击汉字拼成语
- 把倒着的文字或图片正过来
  方法1

```
<div id="codeBox">
ab4f
</div>
<a href="javascript:;"> id="link">看不清换一张</a>

var codeBox=document.getElementById("codeBox");
link = document.getElementById("link");
编写一个获取随机验证码的方法
function queryCode（）{
1、准备验证码获取的范围
var codeArea = “qwertyuiopasdfghjklzxcvbnm”+"QWERTYUIOPASDFGHJKLZXC
VBNM"+"1234567890";
2、准备四个索引，可在codeArea中通过char-AT方法获取到字符，把四个字符串拼接成一
个字符串就是验证码
var result = “”;
for (var i = 0;i < 4;i++){
var n = Math.round(Math.random()*(61);
char = codeArea.charAt(n);
if(result.indexOf(char)>-1){
i--;
continue;
}
result += char;
}
return result;
}
3、开始加载页面就需要生成一个验证码
codeBox.innerHTML = queryCode（）;
执行方法，把方法完成的返回值（四位验证码）赋值给codeBox的内容
link.onclick = function（）{
codeBox.innerHTML = queryCode（）;
}
```

方法2

```
function queryCode（）{
1、准备验证码获取的范围
var codeArea = “qwertyuiopasdfghjklzxcvbnm”+"QWERTYUIOPASDFGHJKLZXC
VBNM"+"1234567890";
var result = "";
while (result.length<4){
var n = Math.round(Math.random()*(61);
char = codeArea.charAt(n);
if(result.indexOf(char)===-1){
result += char;
        }
```

```
    }
    return result;
}
codeBox.innerHTML = queryCode();
link.onclick = function(){
codeBox.innerHTML = queryCode();
}
```

DOM树

> dom tree
> 当浏览器加载html页面的时候，首先就是dom结构的计算，计算出来的dom结构就是dom树（把页面中的html标签像树状结构一样，分析出之间的层级关系）
> DOM树描述了标签和标签之间的关系（节点见得关系），我们只要知道任何一个标签，都可以依据DOM中提供的属性和方法，获取到页面中任意一个标签或节点

在js中获得DOM元素的方法

getElemuntById

- 在ie67中浏览器会把表单的name当做id（以后使用表单的时候不要让name和id重名）

getElementsByTagName

- 语法：[context].getElementsByTagName()
- 1、获取的元素集合是类数组，不能直接用数组的方法
- 2、会把上下文中，后代层级内的标签都获取到
- 3、获得的结果永远都是个集合（不管有没有内容，也不管有几项内容，他是一个容器），要操作集合中的具体某一项，需要基于索引获取到

getElementsByClassName

- 语法：[context].getElementsByClassName()
- 在上下文中，基于元素的样式类名（class="xxx"）获取到一组元素集合
- 项目中，经常是基于样式类给元素设置样式，在js中也经常基于样式类来获得元素，但此方法ie678不兼容

getElementsByName

- 语法：document.getElementsByName()
- 上下文只能是document，在整个文档中，基于元素的name属性值获取一组节点集合（也是一个类数组）

- 在ie浏览器中，只对表单元素的name属性起作用（在项目中只会给表单元素设置name）
  querySelector
- 语法：[context].querySector() 在指定的上下文中基于选择器（类似于css选择器）获取到指定的元素对象（获取的是一个元素，哪怕选择器匹配了多个，我们只获取一个）
- 基于css选择器获取指定对象，只能获取一个

querySelectorAll

- 在querySelector的基础上，获取到选择器匹配的所有元素，结果是一个元素集合（Nodelist）
- querySelectorAll/querySelector都不兼容ie678（能用byid或其他方式获得的，尽量不用这两个方法，性能消耗较大）
  document.head
- 获取head元素对象
  document.body
- 获取body元素对象
  document.documentElement
- 获取html元素对象
- 需求：获取浏览器一屏幕的宽度和高度（兼容所有的浏览器）

```
document.documenElement.clientwidth||documenr.body.clientwidth

document.documenElement.clientheigth||documenr.body.clientheight
```

获取当前页面中所有id为haha的元素（兼容所有浏览器）

-思路：1、首先获取当前文档中所有的html标签
2、依次遍历这些元素标签对象，谁的id等于haha，就储存起来

```
function queryAllById(id) {
        //基于通配符拿到所有标签
        var nodeList = document.getElementsByTagName("*");
        //遍历集合中的每一项，把元素id和传递id相同的这一项存起来
        var ary = [];
        for (var i =0;i < nodeList.length;i++){
            var item = nodeList[i];
            item.id === id ? ary.push(item) : null;
        }
        return ary;
    }
```

```
    console.log(queryAllById("haha"));
```

简单方法

console.log（id名）

- 在js中，默认会把元素的id设置为变量（不需要在即获取设置），而且id重复，获取的结果就是一个集合，包含所有的id项，不重复就是一个元素对象（类似于byid获取的结果）

DOM中的节点（node）

> 在一个html文档中出现的所有东西都是节点
>
> - 元素节点（html标签）
> - 文本节点（文字内容）
> - 注释节点（注释内容）
> - 文档节点（document）

> 每一种节点都会有一些属性区分自己的特性和特征

- **nodeType：节点类型**
- nodeName：节点名称
- nodeValue：节点值

元素节点

- **nodeType：1**
- nodeName:大写标记名
- nodeValue：null

文本节点

- nodeType：3
- nodeName：#text
- nodeValue：文本内容

注释节点

- **nodeType：8**
- nodeName：#comment
- nodeValue：注释内容

文档节点

- **nodeType：9**
- nodeName：#document
- nodeValue：null

描述节点之间关系的属性

parentNode

> 获取当前节点唯一的父节点
> **childnodes**
> 获取当前元素的所有子节点
>
> - 子节点：只获得儿子级别的
> - 所有：包含元素节点、文本节点等
>   **children**
>   获取当前元素所有的元素子节点
>   在ie678中会把注释节点也当做元素节点获取到，所以兼容性不好
>   **previousSibling**
>   获取当前节点的上一个哥哥节点（获取的哥哥可能是元素或文本）
>   **previousElementSibling**
>   获取上一个哥哥元素节点(ie678比兼容)
>   **nextSibling**
>   获取当前节点的下一个弟弟节点
>   **nextElementSibling**
>   下一个弟弟元素节点（比兼容ie68）
>   **firstChild**
>   获取当前元素的第一个子节点（可能是元素或者文本等）
>   **firstElementChild**
>
> **lastChild**
> 获取当前元素的最后一个子节点
> **lastElementChild**

获取所有的元素子节点

> 基于children不兼容ie低版本浏览器（会把注释当元素）

```
//@parameter
```

```
    //curEle:[object]current element
    //@return
    //[array] all the element nodes
    function children(curEle) {
        //1.先获取当前元素下所有的子节点，遍历这些节点，筛选出元素的nodeType
=== 1，把筛选出来的结果单独存储起来即可
        var nodeList = curEle.childNodes;
            result = [];
        for(var i = 0;i < nodeList.length;i++){
            var item = nodeList[i];
            if(item.nodeType === 1){
                reult.push(item);
            }
        }
        return result;
    }
children(id1)
```

获取上一个哥哥元素节点

> previousSibling:上一个哥哥节点
> previousElementSibling：上一个哥哥元素节点，但是不兼容

```
function prev(curEle) {
        //1.先找当前元素的哥哥节点，看是否为元素节点，不是的话，基于哥哥，
        // 找哥哥的上一个哥哥节点···一直到找到元素节点或者已经没有哥哥了（说明
我就是老大）结束查找
        var pre = curEle.previousSibling;
        while(pre && pre.nodeType === 1){
            pre = pre.previousSibling;
        }
        return pre;
    }
    prev(li);
```

DOM元素的增删改

- createElement
  - 创建一个元素标签（对象）
- createTextNode
- appendChild

- insertBefore
- clineNode
- removeChild
- set/get/removeAttribute

while循环

- while（条件）{
  i++
  执行体
  }