目录

1.ECMA5严格模式

- 严格模式：写在那个作用域下，哪个作用域就能生效。
  - `"use strict"`
  - 尽量注意不要把严格模式写在全局。
- 用了严格模式，会有什么样的变化
  1. 全局变量声明时，必须加`var`。
  2. `this`无法指向全局变量。
  3. 函数内重名属性。
     - 在声明形参时，不能出现重名的形参。
  4. `arguments`对象
     - `arguments`对象不允许被动态改变（`arguments`只会获取传参时，形参的值，不会被后续的修改而影响）
  5. 新增保留字：`implements, interface, let, package, private, protected, public, static, yield`。
- 严格模式的目的：
  1. 消除javascript语法的一些不合理、不严谨之处，减少一些怪异行为。
  2. 消除代码运行的一些不安全之处，保证代码运行的安全
  3. 提高编译器效率，增加运行速度。
  4. 为未来新版本的javascript做好铺垫。

     【注】"严格模式"体现了javascrip更合理、更安全、更严谨的发展方向，包括IE10在内的主流浏览器，都已经支持它。

2.ECMA5新增数组的方法

- `arr.indexOf()`：查找元素

    - 在数组中查找 第一次出现 `item` 元素下标，从 `start` 开始去查找 (只会正向查找)

```
arr.indexOf(item, start);
```

    - `item` 任意的数据
    - `start` 下标 可以不传入，默认是0：
    - 返回值： `-1` 没有查找到; `>=0` 查找到元素的下标
- `lastIndexOf()`：从右往左查找元素

- `arr.forEach()`： [ECMA5新增]，只能遍历数组，不能遍历字符串

```
arr.forEach( function(item,index,arr) {
  // item 当前遍历找到的元素
  // index 当前遍历找到元素的下标
  // arr 数组本身
});
```

    - 没有返回值，只能用作遍历

```
var arr =[10,20,30,40,50,60];
arr.forEach(function(item,index){
    document.write(item + "," + index + "<br/>");
    //10,0  20,1  30,2  40,3  50,4  60,5
});
```

    - `forEach` 一旦开始, 无法在中途被停止(`return` 也不行)
    - `forEach` 可以改变 数组中 对象的属性值 `[{a:10}]`

```
var arr5 = [{ a:1 },{ a:2 }, { a:3 }]
arr5.forEach((item,index) => {
  item.a += 5
})
console.log(arr5)  6,7,8
```

- `arr.map()`：映射 **不会改变原数组**

```
arr.map( function(item,index,arr) {
  //遍历要做的事情    映射关系
  return item * 1.3;       //将数组中每一个元素都增加30%
});
```

    - 返回值：执行完操作后的数组 上面的返回值就是数组中每一个元素都×1.3后的新数组

- `arr.filter()`：过滤 **不会改变原数组**

```
arr.filter( function(item,index,arr){
  //过滤的条件
  return item > 30;   //复制大于30的元素
  return item > 30 && index > 4;  //查找大于30的元素，并且下标大于4
});
```

  - 返回值：数组中符合条件的元素 组成的新数组。

- `arr.some()`：查找数组是否有符合条件的值

```
arr.some( function(item,index,arr){
  //查找的条件
  return item > 30;   //有大于30的元素返回true
  return index > 4;   //如果有6个及以上元素返回true
  return item > 30 && index > 4;  //如果有大于30而且下标大于4的元素,返回true
});
```

  - 返回值：在数组中查找是否有符合条件的元素，有返回 `true`。
  - 短路操作：只要找到符合条件的元素，循环就停止了。

- `arr.every()`：查找数组中的值是否都符合条件

```
arr.filter( function(item,index,arr){
  //查找的条件
  return item < 30;   //元素全部小于30返回true
  return index < 4;   //元素的下标都小于4返回true
  return item < 30 && index < 4;  //元素全部小于30而且下标小于4,返回true
});
```

  - 返回值：在数组中查找每一个元素是否有符合条件，符合返回 `true`，不符合返回 `false`。
  - 短路操作：只要找到不符合条件的元素，终止循环，返回 `false`。

- `arr.reduce()`：累加器

```
arr.reduce( (prev,item,index,arr) => {}, initialValue )
```

  - `prev` 初始值 `initialValue` 或上一次遍历 `return` 的值
  - `item` 当前元素
  - `index` 当前元素的下标
  - `arr` 数组本身
  - `initialValue` (作为第一次调用 prev 的第一个参数)

3.认识字符串

- 字符串概念：所有带单引号(`''`)或者双引号(`""`)都叫字符串
- 字符串声明：
  1. 通过new运算符声明字符串

```
var str1 = new String(100);
```

2. 省略new声明字符串

```
var str1 = String(100);
```

3. 字符串常量赋值

```
var str1 = "100";
```

- `str.length`：访问字符串中字符的个数
  - 中文 `utf-8`(三个字符表示一个汉字) `gbk`编码(两个字符去表示一个汉字)，在计数时都是当做一个字符进行计数。
- `str.charAt(下标)` 下标从0开始，访问字符串中的单个字符
  - `str[0]` 也能访问，但是低版本浏览器不支持
- 【注】**字符串是只读的**，字符串一旦被声明，就**无法被修改**，如果要修改字符串，只能将原字符串销毁，重新声明新字符串。
- 字符串中的字符进行遍历 `for`遍历 / `for...i`n遍历

4.字符串方法(了解，后面基本用不到)

```
str.big();    //用大号字体显示字符串
str.blink();   //显示闪动字符串(ie下无效)
str.bold();    //使用粗体显示字符串
str.fixed();   //以打印机文本显示字符串
str.strke();  //使用删除线显示字符串
str.fontcolor("red"); //使用指定颜色显示字符串，颜色要加""
str.fontsize( "20px" );  //使用指定尺寸显示字符串，字体大小要加""
str.link("链接到的地址");    // 将字符串显示为链接,链接的地址要加""
str.sub();    //把字符串显示为下标
str.sup();     //把字符串显示为上标

// document.write()中使用，用特殊的样式输出该字符串，可以多个拼接
document.write( str1.bold().fontcolor("red").fontsize("20px") )
```

5.获取字符串中字符

- `str.charAt(3)` //获取下标为3的字符
- `str.charCodeAt(3)` //获取下标为3的字符的编码
  - 获取字符串对应下标字符的ASCII码值
  - 【注】上述两个方法使用字符串对象调用
- `String.fromCharCode(94,95)`; //编码转成字符
  - 将传入的`ASCII`码值转成对应的字符
  - 返回值: 组成的字符串

```
var str = String.fromCharCode(97,98,99,100);
```

6.字符串查找(重点)

- `str.indexOf(subStr,start)`：查找另一个字符串 第一次出现的位置
  - 从 `str` 中查找 `subStr` 第一次出现的位置，从 `start` 开始查找

```
str.indexOf(subStr,start);
```

  - `subStr`，要查找的字符串
  - `start` 从哪个下标开始查找，默认从下标0开始查找。
  - 返回值: -1 说明没有查找到

```
var str1 = "abcabcd";
var str2 = "abc";
alert( str1.indexOf(str2,1) );
```

- `str.lastIndexOf(subStr,start)`：查找另一个字符串 最后出现的位置

```
str.lastIndexOf(subStr,start);
```

  - 返回值: -1 说明没有查找到
  - 从右向左开始查找

- `str.search(subStr)`：从 `str` 中查找 `subStr` 第一次出现的位置

```
str.search(subStr);     //没有start参数
```

  - `subStr` 可填入 字符串 / 正则表达式
  - 返回值: -1 说明没有查找到

```
var supStr = "abcabcabc";
var sub = "abc";
alert(supStr.search(sub));  // 0
```

  - 正则表达式

```
var supStr = "Abcabcabc";
var sub = /abc/i;
alert(supStr.search(sub));  //0
```

    - 修饰符: `i` 忽略大小写 `g` 全局匹配
    - 正则表达式用两个斜杠 `/内容/修饰符1,修饰符2`

7.字符串提取

- `str.substring()`： 类似于 `数组.slice(start,end)`

```
str.substring(start,end);   // 不包含end
```

- 复制字符串中`[start,end)`部分字符，生成一个新字符串。
  - 省略`end`则从`start`提取到结尾
- 返回值: 新生成的字符串，原字符串不会改变(字符串不可能被修改)

- `str.substr()`： 类似于`数组.splice(start,length,参数)`

```
str.substr(start,length);
```

  - 从`start`开始的位置，`length`截取的长度
    - 省略`length`，默认从`start`位置提取到结束
  - 返回值: 新生成的字符串，原字符串不会改变(字符串不可能被修改)

- `str.slice(start,end)`：数组的方法,尽量少用
  - 数组的方法，字符串可以通过`字符串[下标]`访问对应的字符，所以字符串也可以用, 低版本浏览器不支持

8.字符串方法_重点

- `str.replace()`： 替换字符串 字符串/正则表达式

```
supStr.replace(oldStr, newStr);
```

  - 用`newStr`将`oldStr`，替换掉，生成新字符串
    - `oldStr`传入的是 **字符串**，*只能替换一次*
    - `oldStr`传入 **正则表达式 不加修饰符`g`也只能替换一个**
      - `/xxx/ig` `i`忽略大小写 `g`全局匹配
  - 返回值: 替换成的字符串

```
var str = "how are are you";
var newStr = str.replace("are", "old are");
alert(newStr);      //how old are are you 只有第一个被替换了
alert(str);         //how are are you

var str = "how are are you";
var newStr = str.replace(/are/g, "old are");
alert(newStr);      //how old are old are you
alert(str);         //how are are you
```

- `str.split()` 字符串分割

```
str.split('分隔符',length);
```

  - 用`'分割符'`对原字符串进行分割，将分割完毕的**子串放在一个数组中返回**
    - `'分隔符'`： 用这个分割符对原字符串进行分割
    - `length`： 控制返回数组的元素个数，一般情况下不用

- 返回值：装有分割完毕的数组

```
var str = "hello world";
var newStr = str.split(" ");
  //传入空格,在原字符串中有空格的地方就分割开
alert(newStr);  //hello,world
alert(str);     //hello word
```

  - 相邻的两个分隔符，会产生空字符串
  - 分隔符是空字符串""的时候，直接将每一个字符单独分割成子串，放在数组中返回(空格也会被拆开)

- `str.toLowerCase()` 转成全小写
- `str.toUpperCase()` 转成全大写

- `str.concat()`：字符串拼接，尽量使用 + 进行字符串拼接

```
var str = "hello";
var newStr = str.concat("world",100);
alert(newStr);      //helloword100
```

9.字符串练习(答案看8_字符串练习.html)

1. 将字符串`str = "When I was young, I love a girl in neighbor class."`中，从`young`提取到到`girl`生成新的字符串。
2. 将字符中单词用空格隔开
   - 已知传入的字符串只有字母，每个单词的首字母大写，请将每个单词用空格隔开，只保留一个单词的首字母大写
     - 传入:`"HelloMyWorld"`
     - 返回:`"Hello my world"`