1. 回顾es5：

1、开启严格模式的三个作用:

1)、变量要先定义后使用

2)、改变this指向

3)、使用eval()函数产生作用域

    <script>

        'use strict'; //开启严格模式

        //开启严格模式：

        //1、变量要先定义后使用

        //2、改变this指向

        //3、使用eval()函数产生作用域

        //说明eval()函数：执行字符串格式的语句

        // var str = "1+8";

        // console.log(str);

        // console.log(eval(str));

        //变量要先定义后使用

        var age;

        age = 20;

        // var  age = 20;

        // console.log(age);

        //改变this指向

        function demo(ages, usrs) {

            console.log(this);

            this.age = ages;

            this.usr = usrs;

            console.log(this.age, this.usr);

        }

        //  demo(18, 'lisi'); //相当于window.demo(18,'lisi')

        // new demo(18, 'lisi');

        //使用eval()函数产生作用域

        var email = 'lisi@qq.com';

        var str2 = "var email='zhangsan@163.com';alert(email);";

        eval(str2);

        alert(email);

    </script>

1. JSON对象：
2. 什么是json及json用途？

json(JavaScript Object Notation:javascript对象节点),它是一种轻量级的数据交换格式。

josn用途？

josn主要用来在不同编程语言之间交换数据。

2)Json的两种数据类型： json对象、json对象数组

<script>

       //json对象格式：{"属性名1":属性值1,"属性名2":属性值2...}

        let lisiObj = {

            "uname": "lisi",

            "age": 20,

            "email": "lisi@163.com"

        };

        // console.log(lisiObj.age);

        // for (var x in lisiObj) {

        //     console.log(lisiObj[x]);

        // }

        //josn对象数组格式：[{"属性名1":属性值1,"属性名2":属性值2...},{"属性名1":属性值1,"属性名2":属性值2...}...]

        var stuObj = [{

            "uname": "lisi",

            "age": 20,

            "email": "lisi@163.com"

        }, {

            "uname": "zhangsan",

            "age": 19,

            "email": "zhangsan@qq.com"

        }];

        // console.log(stuObj[1].uname);

        // for (var s = 0; s < stuObj.length; s++) {

        //     console.log(stuObj[s].uname, stuObj[s].age);

        // }

</script>

1. JSON.parse()、JSON.stringify()

 <script>

//JSON.parse():将字符串格式的json转换成真正的json对象

        var str = '{"bookname":"web开发","price":88}';

        var bkObj = JSON.parse(str);

        console.log(typeof bkObj, bkObj);

        //用途：$.get('url',{参数},function(d){},'json');

        //JSON.stringify():将json对象转换成字符串格式的json

        var bkStr = JSON.stringify(bkObj);

        console.log(bkStr, typeof bkStr);

    </script>

1. Object对象：
2. Object.create():以某个对象为原型创建新对象的同时可以扩展新的属性

2)Object.defineproperties():给指定对象扩展新的属性

    <script>

        // Object.create(prototype, [descriptors])：

        //       //\* 作用: 以指定对象为原型创建新的对象

        //       //\* 为新的对象指定新的属性, 并对属性进行描述

        //       //value : 指定值

        //       //writable : 标识当前属性值是否是可修改的, 默认为false

        //       //configurable: 标识当前属性是否可以被删除 默认为false

        //       //enumerable： 标识当前属性是否能用for in 枚举 默认为false  遍历

        var stuObj = {

            "usr": "lisi",

            "age": 20

        };

        // var stuObj2 = Object.create(stuObj);

        var stuObj2 = Object.create(stuObj, {

            "email": {

                value: "lisi@163.com",

                writable: true, //设置当前属性值是否可以被修改

                configurable: true, //设置当前属性是否可以被删除

                enumerable: true

            },

            "tel": {

                value: '110'

            }

        });

        stuObj2.email = 'lisi@qq.com';

        //删除属性：

        // delete stuObj2.email;

        console.log(stuObj2, stuObj2.email);

        //遍历对象

        for (var y in stuObj2) {

            //  console.log(y, 888);

        }

        //  Object.defineProperties(object, descriptors)

        //作用: 为指定对象定义扩展多个属性

        //get：用来获取当前属性值得回调函数

        //set：修改当前属性值得触发的回调函数，并且实参即为修改后的值

        var bookObj = {};

        // bookObj.price = 100;

        console.log(bookObj);

        Object.defineProperties(bookObj, {

            "price": {

                value: "68",

                writable: true

            }, //价格

            "setprice": {

                //获取当前属性值时调用的回调函数

                get: function() {

                    return this.price;

                    //console.log('get....');

                },

                //设置当前属性值时调用的回调函数

                set: function(val) {

                    //写业务代码

                    if (val >= 100) {

                        val = 99;

                    }

                    this.price = val;

                    // console.log(val, 'set....', this.price);

                }

            }

        });

        bookObj.setprice = 100;

        console.log(bookObj.setprice, 8888);

    </script>

1. 数组：

Array.filter()：遍历数组的同时过滤元素

Array.map():遍历数组的同时改变其中元素值

    <script>

        //分数

        var arr = [67, 89, 71, 90, 83];

        //需求：查询分数大于等于80分的

        var arr2 = arr.filter(function(item, ind) {

            // console.log(x, y);

            return item >= 80;

        });

        //  console.log(arr2);

        //需求：给分数小于80分的加5分

        var arr3 = arr.map(function(item, ind) {

            if (item < 80) {

                return item + 5;

            } else {

                return item;

            }

        });

        console.log(arr3);

    </script>

1. 回顾函数：

   1)、arguments

        2)、通常有return

        3)、改变this指向:  call()、apply()、bind()

    <script>

        //回顾函数：

        //1、arguments

        //2、通常有return

        //3、改变this指向:  call()、apply()、bind()

        function fn() {

            console.log(arguments, this);

        }

        //    fn(11, 2);

        document.onclick = function() {

            // fn(1, 3);

            //   fn.call(this, 1, 6);

            // fn.call(null, 1, 6);  //null不改为this指向

            //  fn.apply(this, [1, 6]);

            // var tst = fn.bind(this);

            // // console.log(tst);

            // tst(1, 2);

            //等价于：var tst = fn.bind(this);

            //       tst(1, 2);

            fn.bind(this)(1, 2);

        }

    </script>

1. Es6 简介：

        //ECMAScript与JavaScript关系 ？

        //ECMASceipt是规范或标准，而JavaScript是该规范的实现

        //es5与es6关系？

        //es6是在2015年6月推出的，在这之前称为es5，以后每年6月都会发布新的版本，但2015年这个版本是里程碑。

1. Es新特性--let与var:
2. let作用：用来声明变量：
3. Let与var的别：

1)、作用域不同：

let定义的变量是块级作用域 ；

var定义的变量是函数作用域[局部的]， 但在if、for等定义的变量是全局的。

2)、在同一个作用域中同一个变量let只能声明一次，而var可以多次声明，let声明的变量可以避免全局变量被污染。

3)、var声明的变量存在变量提升，而let则没有变量提升，let声明的变量要先定义后使用。

4)、let有暂时性死区：let声明的变量要先定义后使用。

5)、let声明的变量不会绑定到顶层对象(顶层对象可以理解为最大的全局变量，即window对象)上。

  <script>

        //let作用：声明变量

        //let 声明的变量是块级作用域 、而var声明的变量是函数作用域，但在if、for中声明的变量是全局的

        function fn() {

            var x = 21;

        }

        fn();

        //    console.log(x); // : x is not defined

        if (1) {

            var y = 22;

        }

        //  console.log(y);

        for (var i = 0; i < 10; i++) {

        }

        //  console.log(i);

        //let声明的变量是块级作用域

        let tst = 'hello';

        if (1) {

            console.log(tst);

            let tst2 = 'word';

        }

        //  console.log(tst2);

        for (let s = 0; s < 10; s++) {

        }

        // console.log(s); //s is not defined

        // for (var oop = 0; oop < 10; oop++) {

        //     console.log(oop, 666);

        //     var oop = 11;

        // }

        for (let oop = 0; oop < 10; oop++) {

            // let oop = 11;

            let oop = 11;

            console.log(oop, 777);

        }

    </script>

    <script>

        //let 在地同一作用域中只能声明一次，但var可以声明多次

        var age = 19;

        var age = 20;

        // console.log(age);

        let usr = 'lisi';

        //  let usr = 'zhangsan';

        usr = 'zhangsan';

        console.log(usr);

        if (1) {

            let email = 'lisi@qq.com';

            //var email = 'zhangsan@qq.com';  //'email' has already been declared

        }

    </script>

    <script>

        //let声明的变量没有预解析、但var有变量预解析

        // console.log(age);  //undefined

        // var age = 20;

        //let声明的变量没有预解析

        // console.log(x); //x is not defined

        // let x = 2;

        // var age = 20;

        // function demo() {

        //     console.log(age);

        // }

        // demo();

    </script>

    <script>

        // function demo() {

        //     console.log(x); //undefined

        //     var x = 2;

        // }

        //let有暂时性死区：也就是let声明的变量要先定义后使用

        function demo() {

            console.log(x); //x is not defined

            let x = 2;

        }

        demo();

    </script>

1. let应用：

<body>

    <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>

        var arr = [];

        //   for (var x = 0; x < 10; x++) {

        for (let x = 0; x < 10; x++) {

            arr[x] = function() {

                alert(x);

            }

        }

        //需求：调用arr[2]()之后输出结果为：alert(2);

        //   console.log(arr);

        arr[2]();

        //需求：单击第一个按钮alert(1)，后面的按钮功能类推：

        var btns = document.getElementsByTagName('button');

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

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

            btns[i].onclick = function() {

                alert(i + 1);

            }

        }

        //   console.log(i, 888);

    </script>

</body>

作业：

使用let 改写前面的”选项卡”案例

let的总结：暂时忘记var，尽量使用let

1. Es6的常量：

1、常量也是块级作用域、不存在声明提升、同一个常量只能声明一次；

2、常量一旦声明其值不能改变，但要注意：如果常量的值为数组、对象时其值是可以被改变的；

    <script>

        //声明常量：const  常量名

        //    注意：常量名习惯性大写

        //为什么要使用常量 ：

        //只要在整个程序运行期间其值是固定不变的都可以声明成常量，因为常量一旦声明其值不能被修改

        //常量的特点：

        //1、常量是没有声明提升、块级作用域、同一作用域只能同一个常量

        //2、常量一旦声明其值是不能被修改的，但如果常量的值是对象或数组则可以修改

        const PI = 3.1415;

        // PI = 3.14; //常量一旦声明其值是不能被修改的

        // const PI = 3.14;  // 'PI' has already been declared

        console.log(PI);

        //常量是没有声明提升

        // console.log(PORTS); // PORTS is not defined

        // const PORTS = 3306; //mysql端口

        //块级作用域

        // if (1) {

        //     const HOSTS = "www.baidu.com";

        // }

        // console.log(HOSTS); // HOSTS is not defined

        // 如果常量的值是对象或数组则可以修改

        const OBJ = {

            "x": 1,

            "y": 21

        };

        OBJ.x = 33;

        console.log(OBJ);

    </script>

总结：常量的使用场景：

1. 如果在开发程序时遇到其值是相对固定的则可以声明为常量
2. 开发项目时项目配置(如：数据库端口、主机名)则可以声明为常量
3. 什么是解构赋值：

在es6中将数组、对象中的数据按照一定模式提取出来并赋值给变量，这一过程称为解构赋值。

1. 数组解构赋值：

1)数组完全解构赋值

2)数组不完全解构赋值

3)默认值

4)数组嵌套解构赋值

**注意**：在数组解析赋值时要注意左边模式中的变量先后顺序，因为数组是一个有序的集合。

    <script>

                //数组完全解构赋值

        let arr = [1, 3, 5, 8];

        //es5写法：

        // let x = arr[0];

        // let y = arr[1];

        // let m = arr[2];

        // let [x, y, m, n] = arr;

        // console.log(x, y, m, n);

        // let [i, p, t] = [33, 22, 6];

        // console.log(i, p, t);

        //数组不完全解构赋值

        // let [x, y] = [1, 2, 3];

        // console.log(x, y);

        // let [x, , y] = [1, 2, 3];

        // console.log(x, y);

        // let [x, y, m, n] = [1, 2, 3];

        // console.log(x, y, m, n);

        //数组解构赋值的默认值：需要注意的是解析失败时默认值才生效

        // let [x, y = 77, z = 88] = [1, 6];

        // console.log(x, y, z);

        //数组嵌套解构赋值：

        let arr2 = [1, 3, [21, 31], 5];

        // console.log(arr2);

        // let [, x, [, y], z] = arr2;

        let [, y, [, x], z] = arr2;

        console.log(x, y, z);

    </script>

1. 对象解析赋值：

1)对象完全解构赋值

2)对象不完全解构赋值

3)默认值

4)对象嵌套解构赋值

    <script>

        // 1)对象完全解构赋值

        let stuObj = {

            "usr": "demo",

            "age": 20,

            "email": "demo@qq.com"

        };

        // //es5写法：

        // let usr = stuObj.usr;

        // let age = stuObj.age;

        //完整形式：

        // let {

        //     "usr": x,

        //     "age": y,

        //     "email": m

        // } = stuObj;

        // console.log(x, y, m);

        // let {

        //     "usr": usr,

        //     "age": age,

        //     "email": email

        // } = stuObj;

        //上面的代码可以简写为：

        //简写形式：

        // let {

        //     usr,

        //     age,

        //     email

        // } = stuObj;

        //\*\*注意：不需要考虑变量的先后顺序:

        let {

            age,

            email,

            usr

        } = stuObj;

        // console.log(usr, age, email, 999);

        // 2)对象不完全解构赋值

        // let {

        //     x,

        //     z

        // } = {

        //     "x": 11,

        //     "y": 2,

        //     "z": 9

        // };

        // console.log(x, z);

        // let {

        //     x,

        //     y,

        //     z

        // } = {

        //     "x": 11,

        //     "y": 33

        // };

        // console.log(x, y, z);

        // 3)默认值

        // let {

        //     m,

        //     n = 88,

        //     z = 9

        // } = {

        //     "m": 1,

        //     "n": 33

        // };

        // console.log(m, n, z);

        // 4)对象嵌套解构赋值

        let obj = {

            'tst': [11, 2, {

                "x": 21,

                "y": 33

            }, 9]

        };

        let {

            'tst': [, m, {

                y

            }, z]

        } = obj;

        console.log(m, y, z);

    </script>