JS第三周第二天

1.内存释放问题

```
<body>
<!--
1.堆内存 存储引用数据类型的
2.栈内存 作用域
堆内存的释放 一般需要手动清空 以后在项目中如果后面不再使用的引用数据类型地址 应该手动清空 赋值为null
obj=null;
栈内存的释放
1.全局作用域 只有关闭浏览器器才会销毁
2.私有作用域
销毁的
不销毁的:如果一个作用域执行之后返回一个新的函数地址 并且被外界占用了 此时这个作用就不销毁
-->
</body>
```

2.循环绑定事件问题

```
<body>
<ul id="list">
    <li>1</li>
    <li>2</li>
    <li>3</li>
    <li>4</li>
</ul>
</body>
```

```
<script>
  //输出每一个li的索引
  var list = document.getElementById("list");
  var oLis = list.getElementsByTagName("li");

  for (var i = 0; i < oLis.length; i++) {
    (function (i) {
      oLis[i].onclick = function () {
        console.log(i);
      }
    })(i)
  }

  for (var i = 0; i < oLis.length; i++) {
    oLis[i].onclick = (function (i) {
      return function () {
        console.log(i);
      }
    })(i)
  }
  //通过形成一个私有作用域来保护或者存储一些私有变量 这种形式成为闭包

  for (let i = 0; i < oLis.length; i++) {
    //这里的i是每一个块级作用域中私有的i
  }
  for (let i = 0; i < oLis.length; i++) {
    //这里的i是每一个块级作用域中私有的i
  }
</script>
```

3.函数问题

```
<script>
    //函数有length属性 没有默认值值的时候就是函数形参的个数
    function fn(a) {}
    console.log(fn.length);

    //形参的默认值 只有不传实参的时候才会走默认值
    function f1(x,y=(function () {console.log("珠峰"); return 100})
()) {
      x=10;
      console.log(x+y);
    }
    f1(1,2);
    function f2(x1,x2,x3=3) {}
```

```
    //一旦形参有默认值的时候length就不再表示形参的个数
    console.log(f2.length);

    var x=100;
    var s="hh";
    function f3(x=1,y=x,z=10,m=s) {
      console.log(y);
      var s="HH";
      console.log(m);
    }
    f3();
    //函数执行步骤:
    //1.形成私有作用域
    //2.形参赋值 x=1,y=1,z=10,m=s 此时他还没有私有变量s往上找 全局的s="hh"
    //3.变量提声
    //4.代码执行

    function f4(y1,y2,y3=m,m=10) {
      //形参赋值:y1=undefined,y2=undefined,y3=m 往前看自己有吗?没有上一级
有吗?没有 全局中都没有m此时就报错 m is not defined
    }
    f4()
</script>
```

4.箭头函数

```
<script>
  //function 函数名 (形参) {函数体}
  //箭头函数是匿名函数
  //(形参)=>{函数体}

  //简写方式
  //1.如果形参只有一个可以省略()
  //2.如果函数体只有一条语句而且是return XXX;可以省略{}和return
  function f(x) {
    return typeof x;
  }

  let ff = x => typeof x;
  //这俩是一个意思

  let fn = (x) => {
    x++;
    console.log(x);
  };
  fn(10);
```

```
  let ary = [1, 2, 3, 4, 5, 6, 7, 8];
  ary.forEach((item, index) => {
    console.log(item);
  });
  //将所有的奇数获取出来组成新的数组
  console.log(ary.filter(item => item % 2));

  ary.filter(function (item) {
    return item%2
  })
</script>
```

5.自定义属性

```
<body>
<ul id="list">
    <li></li>
    <li></li>
    <li></li>
</ul>
</body>

<script>
    let oLis=document.getElementById("list").getElementsByTagNam
e("li");

    for (var i=0;i<oLis.length;i++){
      oLis[i].i=i;
      oLis[i].onclick=function () {
        //this:给谁绑绑定的事件this就是谁
        console.log(this.i);
      }
    }
</script>
```

6.this

```
<body>
<!--
this:是执行主体
this是个对象,不可以赋值的
在全局中this是window
我们主要研究私有作用域下的this
当一个函数执行  函数里面的this是谁就看函数执行的时候前面有没有点  有点的话点前面是
```

```
谁this就是谁 没有点this就是window
-->
</body>
```

7.this的特殊情况

```
<script>
  //1.自执行函数中的this永远都是window
  (function () {
    console.log(this);
  })();
  var obj = {
    fn: (function () {//属性名fn的属性值是自执行函数执行后的结果
      console.log(this);
      return function () {
        console.log(this);
      }
    })()
  };
  obj.fn();
  //2.给元素绑定事件中 给谁绑的this就是谁
  box.onclick = function () {
    change("red");
  };

  function change(color) {
    box.style.backgroundColor = color;
  }

  //3.函数当做参数的时候 函数中的this是window
  var ary = [12, 13, 14];
  ary.forEach(function (item) {
    console.log(this);
  }, ary);
  //数组中的遍历方法 第二个参数就是用来改变第一个参数函数中的this的

  //4.箭头函数中没有指向 如果出现this指的是上一级的this
  let fn2 = () => {
    console.log(this);//window
  };
  var obj3 = {
    fn2: fn2
  };
  obj3.fn2();
</script>
```

8.对象扩展

```
<script>
    function fn1() {}
    let name="珠峰";
    //属性名和变量名重名了可以直接简写方式
    let obj={
      fn1,//fn1:fn1
      name,//name:"珠峰"
    };

    //数组解构赋值
    let [x,y]=[1,2];

    //对象的解构赋值 注意 左边一定不要忘了 写let
    let {name:n,age:a1}={name:"珠峰",age:10};
    //变量 n,a n对应的值就是name的属性值 a对应的值就是属性名age的值
    //变量名和属性名相同了可以简写
    let {age}={age:100};
    //age变量 对应右边属性名是age的值
    console.log(age);
    var aa="AA";
    var obj1={a:1,aa,bb:"BB"};
    var {aa:A,bb,a,aa}=obj1;
    // {aa:A,bb,a,aa}={a:1,aa:"AA",bb:"BB"}
    // A="AA"
    // bb="BB"
    // a=1
    // aa="AA"
    console.log(A);

    //数组的扩展运算符
    let ary1=[1,2];
    let ary2=[10,20];
    console.log([...ary1, ...ary2]);//[1,2,10,20]

    //对象的扩展运算符 ES7中提供的
    let o1={name:"珠峰"};
    let o2={age:10};
    let o={...o1,...o2};
    console.log(o);

    function f1() {}
    function f2() {}
    function f3() {}
    function f4() {}
    function f5() {}
```

```
        let obj0={f1,f2,f3,f4,f5};

        var {f4,f5}=obj0;
    </script>
```