JS第三周第一天

1.变量提升

```
<script>
    //变量提升:变量的提前声明
    //作用域形成之后，代码执行之前，先将带var和function 关键字的提前声明或定义
    //作用域:代码执行的环境 栈内存
    //1.全局作用域:打开浏览器就形成全局作用域
    //2.私有作用域:一个函数执行就会形成一个私有作用
    //3.块级作用域:使用{}包起来的就是块级作用域 {},for(){},if(){},while(){};

    //声明或定义
    //使用var只声明不定义
    //function既声明又定义

    console.log(a);//undefined
    var a=10;
    console.log(a);
    //1.形成全局作用域
    //2.变量提声 将所有带var关键字的提前的声明 此时只声明不定义 a的值是undefined
    //3.代码执行 从上到下依次执行 23行 console.log(a);是undefined 24行给a赋值为10,25行console.log(a) a是10


   f();
   function f() {
     console.log("我是function");
   }
</script>
<script>
```

```
    f1();
    var b=1;
    function f1() {
      b++;
      console.log(b);
    }
    console.log(b);
    //1.全局作用域形成
    //2.变量提声:var b (undefined) function f1=xxxfff000
    //3.代码执行:
    //38:  f1();//xxxfff000 ()
    //41:  b++;此时b是undefined 所以b是NaN
    //42:  console.log(b); NaN
    //39:  b=1;
    //44:  console.log(b);1

</script>
```

2.没有关键字的变量

```
<script>
  //如果遇到使用变量的时候发现只声明未定义是undefined,如果没有声明过就报错了
  console.log(a);//undefined
  //console.log(b);//报错
  var a = 1;
  b = 1;

  console.log(x1);//undefined
  //console.log(x2);//报错
  var x1 = x2 = x3 = 10;//这里只有一个var 只能声明一个 x1


  console.log(y1, y2, y3);
  var y1 = 10, y2 = 20, y3 = 30;//这种使用逗号隔开的形式 是分别声明


  //fn();//报错:fn is not a function 此时fn是undefined
  var fn = function () {};//这种情况属于var声明的
</script>
```

3.声明变量var和function重名

```
<script>
    //遇到var 声明关键字和function声明的关键字重名了,在变量提声阶段 此时是一个变量 值是函数的地址值;但是代码执行的时候如果又给变量赋值了.而且赋的值不是函数再让变量当做函数执行的时候就会报错
    f();
    var f=10;
    function f() {
      console.log("珠峰");
    }
    f();


    console.log(x1);
    //console.log(x2);
    var x1=x2=function () {
      console.log("哈");
    };
    x1=10;
    console.log(x2);
    console.log(x1);

</script>
```

4.ES6中定义变量let和const

```
<script>
  //ES6中没有变量提声 不可以使用未声明的变量
  //console.log(a);//a is not defined
  let a = 1;

  //let
  /*
  * 1.不可以重复声明   'a' has already been declared
  * 2.没有变量提声
  * 3.声明的变量不会给window增加一个属性
  * */
  //const 定义常量
  /*
  * 1.不可以重复声明
  * 2.没有变量提声
  * 3.声明的常量不会给window增加一个属性
  * 4.一旦声明必须赋值
  * 5.赋值之后不可以修改
  * */
  // var a=9;
  // let a=1;
```

```
  // let a=10;
  let b = 10;
  console.log(window.b);
  console.log("b" in window);
</script>
```

5.变量声明的特殊情况

```
<body>
<div id="box">box</div>
</body>

<script>
    //注意:=右边如果赋值是一个函数 直接写一个匿名函数即可 如果写了函数名也没有任何作用,当前函数的名字仍然是=左边的变量
    //f();
    var fn=function f() {

    };
    //f();
    //A is not defined
    box.onclick=function A() {

    };

    //当函数当做参数的时候函数名字是没有任何意义的
    [1,3,2].forEach(function B(item) {
      console.log(item);
    });
    //console.log(B); B is not defined

    setInterval(function G() {
      console.log(1);
    },1000);
    //console.log(G);//G is not defined


    //1.综上 函数作为值的时候(=右边,回调函数(函数当做参数的时候)) 函数名字没有任何意义 所以说不进行变量提声
    //注意:是在作为值的时候定义的

    //2.自执行函数没有变量提升
    //其实第一个()里面的函数定义就是一个值(函数的地址值)
    //console.log(gg);
    (function gg (){console.log("我是自执行函数")})();
```

```
    //3.return后面的代码不执行但是会变量提声

    function f1() {
      console.log(a);//undefined
      f();//输出  "你好"
      return 1;
      var a=1;
      function f() {
        console.log("你好");
      }
    }
    f1();

    //4.判断
    // 在变量提声阶段 不管条件是否成立都进行变量提声 但是var 和function都是只声明不定义 当代码执行的时候 如果条件成立if的{}中先给函数赋值再执行代码
    var s="0";
    console.log(bb);
    console.log(ff);
    if(s){
      ff();
      var bb="bb";
      function ff() {
        console.log("ff");
      }
    }
</script>
```

6.私有作用域

```
<script>
    //私有作用域:一个函数执行会形成一个私有作用域 保护里面的私有变量不受外界的干扰
    //私有变量:
    //1.在私有作用域下声明过的变量
    //2.形参

    //函数执行的顺序
    //1.私有作用域形成
    //2.给形参赋值
    //3.变量提声
    //4.函数体中的代码执行

    //私有作用域下的私有变量外界获取不到
    //函数执行一次就形成一个私有作用域,如果执行多次每一次形从成的私有作用域没有
```

```
任何关系
    var a=2;
    function f1() {
      var a=1;
      var b=10;
    }
    f1();
    console.log(a);//2
    //console.log(b);//报错了


    //在私有作用域下变量提声的时候遇到已经有的私有变量(形参)就不用再声明了
    function f2(x1,x2) {
      console.log(x1);
      var x1=100;
      var a=1;
    }
    f2(10,20);
    //console.log(x1);
    f2(1,2);
    f2();


    //在私有作用域下遇到变量,先看是不是自己的私有变量(声明过的,形参),如果是就用
自己的 不是话往上一级作用域去找
    var s="1";
    var g=10;
    function fn1(g) {
      s+=2;
      g++;
      console.log(g);
    }
    fn1(g);
    console.log(s);
    console.log(g);
</script>
<script>
    var aa=100;
    var bb=200;
    function fn2(aa) {
      console.log(bb);
      var bb=1;
      aa=2;
      console.log(bb + aa);
    }
    fn2(bb);
    console.log(aa + bb);
</script>
```

7.块级作用域

```
<script>
    /*
    //{}
    //如果想要{}表示对象千万不要放在行首;
    var obj={name:"珠峰",age:10};
    ({name:"珠峰",age:10});

    //1.在块级作用域中使用ES5声明的变量依然是当前作用域的
    //但是比变量提声有点改变: var和function都是只声明不定义,进入到块级作用域中先给函数赋值再去执行块级作用域的代码
    //2.在块级作用域中使用ES6声明的变量是当前块级作用域的私有变量
    console.log(a);//undefined
    console.log(fn1);//undefined
    {
      console.log(a);//undefined
      console.log(fn1);//fn1这个函数本身
      var a=1;
      function fn1() {}
    }
    console.log(a);//1
    console.log(fn1);//fn1这个函数本身


    //暂时性死区:在块级作用域中不可以提前使用let和const定义的变量
    {
      //这里的x1属于当前块的私有变量
      //console.log(x1);//报错的
      let x1=10;
    }
    //console.log(x1);//x1 is not defined


    if(1){
      let aa=1;
    }

    for (let i=0;i<3;i++){
      //循环中如果使用let i,此时每一次循环得到一个块级作用域都是私有作用 i是他的私有变量
      console.log(i);
    }
    //console.log(i);i is not defined


    */
```

```
    //全局变量:a

    if(1){
      //a++;//报错 不可以提前使用let声明的变量
      let a=10;
      console.log(a);
    }
    var a=1;
    if(a){
      a++;
      var a=100;
      console.log(a);
    }
    console.log(a);
</script>
```