ES6 PPT1511262: 箭头函数

Thursday, November 5, 2015 10:15 AM

箭头函数 =>：

1. 可直接返回值的函数：

   典型应用：`var fnArrowDoubleMe = n => n * 2;`

   function解析：`function fnFunctionDoubleMe(n) { return n * 2; }`

   函数模型：`var fnDirectToRes = (...paramsIn) => (resultOut);`

2. 需要执行一些代码段然后返回值的函数：

   典型应用：`var fnArrowAddMe = (arr, v) => { arr.push(v); return arr; }`

   function解析：`function fnFunctionAddMe(arr, v) { arr.push(v); return arr; }`

   函数模型：`var fnCodeToRes = (...paramsIn) => { myfunc1(); myfunc2(); return resultOut; };`

3. 注意事项：某种程度上讲箭头函数并非函数，其仅为一块可传参回值复用的代码段，与 function 不同

   a. 箭头函数内的 this 指向其所在父级函数体的 this 对象：

   ```
   var id = 'global';
   function foo1() {
     setTimeout(() => { console.log('使用箭头函数: this.id=' + this.id); }); // 使用箭头函数: this.id=local
     setTimeout(function () { console.log('使用function: this.id=' + this.id); }); // 使用function: this.id=global
   }
   foo1.call({ id: 'local' });
   ```

   b. 箭头函数不可用于构造函数即不可使用 new 命令：否则会搞乱 this 的指向

   ```
   function foo2() {
     console.log(this.id); // idfoo2
     var fnArrow = () => { this.id = 'idarrow'; }; console.log((new fnArrow()).id); // undefined
     var fnFunc = function () { this.id = 'idfunc'; }; console.log((new fnFunc()).id); // idfunc
     console.log(this.id); // idarrow
   }
   foo2.call({ id: 'idfoo2' });
   ```

   c. 不能使用 arguments 获取参数值，需使用...REST代替：

   ```
   function foo3(a, b, c) {
     var fnPushRest = (...rest) => { var arr = []; arr.push(...rest); return arr; };
     var fnPushArguments = (d, e, f) => { var arr = []; arr.push(...arguments); return arr; };
     console.log(fnPushRest(4, 5, 6)); // [4,5,6]
     console.log(fnPushArguments(4, 5, 6)); // [1,2,3]
   }
   foo3(1, 2, 3);
   ```

   d. 无法使用 yield 命令，所以箭头函数无法用作 Generator 函数

   e. 嵌套箭头函数：

   ```
   function foo4(a) {
     return (b) => {
   ```

```
    return (c) => {
      console.log("this.id=" + this.id); // this.id=idfoo4
      console.log("arguments=" + [...arguments]); // arguments=A
    };
  };
}
foo4.call({ id: 'idfoo4' }, 'A')('B')('C');
```

f. 以下关键字/变量于箭头函数使用时作用于父级函数体：this, arguments, super, new.target, call(), apply(), bind()

各类应用：

1. 简化回调 callback 函数： [1,2,3].map(x=>x*2); // [2,4,6]
2. 定义内联函数： (n=>n*n)(16); // 256
3. 一些情况下可代替 call(this), apply(this), bind(this)