# 面向对象编程

## 声明静态类类：

**声明：**

1. **define**([ "dojo/dom"], **function**(dom) {
2. **return** {
3. setRed: **function**(id){
4. dom.byId(id).style.color = "red";
5. }
6. };
7. });

**使用：**

1. **<script>**
2. require(
3. [ "dojo/ready", "test/util" ],
4. function(ready, util) {
5. ready(function() {
6. var id = "selected\_text";
7. util.setRed(id);
8. });
9. });
10. **</script>**

## 声明普通类

使用**Declare 关键字**

1. **define**([ "dojo/\_base/declare", "dijit/Dialog", "dijit/\_WidgetBase",
2. "dijit/\_TemplatedMixin", "test/util" ], **function**(declare,
3. Dialog, \_WidgetBase, \_TemplatedMixin, util) {
4. **return** declare("test.RedTextDialog", [ Dialog, \_WidgetBase, \_TemplatedMixin ], {
5. title: "Dialog with Red Text",
7. onDownloadEnd : **function**() {
8. **var** id = "selected\_text";
9. util.setRed(id);
10. }
11. });
12. });

## 引用类

1：

1. require(
2. [ "dojo/ready", "test/util" ],
3. function(ready, util) {
4. ready(function() {
5. var id = "selected\_text";
6. util.setRed(id);
7. });
8. });

2：

var declare = require("dojo/\_base/declare");

## 继承类

MyCarBean = declare(BeanBase, {

//构造函数

constructor: function (options) {

this.status = options.status || 0;

this.speed = options.speed || 0;

this.carType = options.carType || "car";

this.carNo = options.carNo || "";

this.color = options.color || 1;

this.aspect = options.aspect || 0;

this.icon = options.icon || "";

});

})

# 基础功能

## 选择器

dom.byId

## dom操作

| **Module** | **Description** | **Contains** |
| --- | --- | --- |
| dojo/dom | Core DOM functions | byId() isDescendant() setSelectable() |
| dojo/dom-attr | DOM attribute functions | has() get() set() remove() getNodeProp() |
| dojo/dom-class | DOM class functions | contains() add() remove() replace() toggle() |
| dojo/dom-construct | DOM construction functions | toDom() place() create() empty() destroy() |
| dojo/dom-form | Form handling functions | fieldToObject() toObject() toQuery() toJson() |
| dojo/io-query | String processing functions | objectToQuery() queryToObject() |
| dojo/dom-geometry | DOM geometry related functions | position() getMarginBox() setMarginBox() getContentBox() setContentSize() getPadExtents() getBorderExtents() getPadBorderExtents() getMarginExtents() isBodyLtr() docScroll() fixIeBiDiScrollLeft() |
| dojo/dom-prop | DOM property functions | get() set() |
| dojo/dom-style | DOM style functions | getComputedStyle() get() set() |

## 动画

require([

'dojo/dom',

'dojo/fx',

'dojo/domReady!'

], function (dom, fx) {

// The piece we had before...

var greeting = dom.byId('greeting');

greeting.innerHTML += ' from Dojo!';

// ...but now, with an animation!

fx.slideTo({

node: greeting,

top: 100,

left: 200

}).play();

});

## 加载

require(["dojo/dom"], function(){

// some code

var dom = require("dojo/dom");

// some more code

});

# 高级功能

## 事件连接

dojo.connect(dojo.byId("greeting"),"onclick",function(){alert("clicked!")})

//dojo自己的事件

var on = require("dojo/on")

on(dojo.byId("greeting"),"click",function(){alert("clicked! 1111")})

## 订阅和发布事件

require(["dojo/topic"], function(topic){

// To publish a topic

topic.publish("some/topic", 1, 2, 3);

// To subscribe to a topic

var handle = topic.subscribe("some/topic", function(arg1, arg2, arg3){

// ...

});

// To unsubscribe from a topic

handle.remove();

});

## 延迟加载

require(["dojo/Deferred"], function(Deferred){

function createMyDeferred(){

var myDeferred = new Deferred();

setTimeout(function(){

myDeferred.resolve({ success: true });

}, 1000);

return myDeferred;

}

var deferred = createMyDeferred();

deferred.then(function(data){

console.log("Success: " + data);

}, function(err){

console.log("Error: " + err);

});

});

## 网络请求

require(["dojo/request"], function(request){

request.get("something.json", {

handleAs: "json"

}).then(function(response){

console.log("response:", response);

}, function(err){

console.log("error:", err);

});

});

示例：

var photos = esriRequest({

//"url": "1000-photos.json",

"url": "getTreeArrayData.json",

"handleAs": "json"

});

photos.then(addClusters, error);

# 实用功能

## Array循环

"dojo/\_base/array", ==== arrayUtils

var data = arrayUtils.map(resp, function(p) {

var car = new MyCarBean({

lat: p.LAST\_LATITUDE,

lng: p.LAST\_LONGITUDE,

status: "0",

speed: "80",

carType: "car",

carNo: p.CAR\_NO,

color: "1",

aspect: "5",

key: p.KEY\_ID,

icon: "http://202.102.101.74:8012/assets/icon/cars/greencar.png" //图片地址

});

return car;

});