ECharts的入门教程-Getting started

# ECharts2的入门教程

<http://echarts.baidu.com/echarts2/doc/start.html>

## 模块化单文件引入（推荐）

步骤：

### 新建一个echarts.html文件，为ECharts准备一个具备大小（宽高）的Dom。



### 新建<script>标签引入模块化单文件echarts.js

<script src="http://echarts.baidu.com/**build/dist/echarts.js**"></script>



### 新建<script>标签中为模块加载器配置echarts和所需图表的路径（相对路径为从当前页面链接到echarts.js）

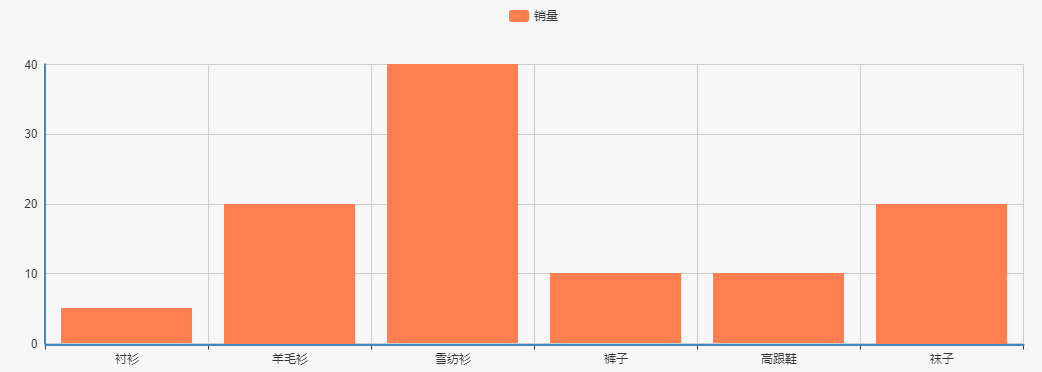


### <script>标签内动态加载echarts和所需图表，回调函数中可以初始化图表并驱动图表的生成





### 浏览器中打开echarts.html，就可看到以下效果



## 标签式单文件引入

步骤：

### 新建一个echarts.html文件。为ECharts准备一个具备大小（宽高）的Dom。（同上A）

### 新建<script>标签引入echarts-all.js（同上B，修改src）

<script src="http://echarts.baidu.com/**build/dist/echarts-all.js**"></script>

### 新建<script>标签，使用全局变量echarts初始化图表并驱动图表的生成





### 浏览器中打开echarts.html，就可看到效果（同上E）

## 模块化单文件引入与标签式单文件引入区别

模块化单文件引入的是**build/dist/echarts.js**文件，而标签式单文件引入的是**build/dist/echarts-all.js**文件。echarts-all.js文件中包含了各种图表，文件900多k，较大，而echarts.js文件中没有包含图表，文件大约500k，文件较小。这两个文件同级目录下有个**chart文件夹**，该文件夹存储各个图表js文件。非模块化引入是直接引入echarts-all.js文件，不需要require配置；模块化引入是引入echarts文件和对应的chart下的图表文件。

从代码上看：非模块化引入更简洁；

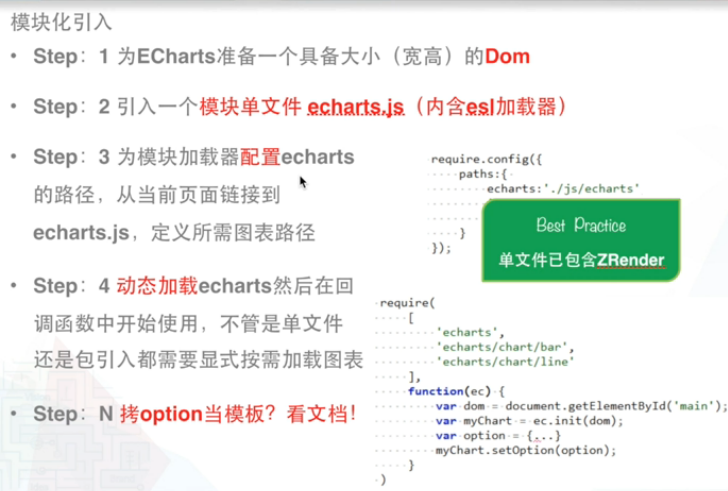
从性能上看：模块化引入更好，因此建议采用模块化引入。



非模块化引入：



模块化引入：



# 使用ECharts步骤总结

## 使用ECharts步骤总结：

首先创建一个DOM，然后**引入ECharts**，初始化一个ECharts对象，实现option变量，将option传入ECharts对象(为**ECharts**对象加载数据)。

## 具体实现示例

<!-- 为ECharts准备一个具备大小（宽高）的**Dom** ,一般用div-->

<div id="main" style="height:400px"></div>

// 基于准备好的**dom**，**初始化echarts图表**

var myChart = **echarts**.init(document.getElementById('main'));

//通过调用echarts的init方法，获取ECharts实例对象；echarts存在于**echarts.js**文件中。查看source/echarts.js文件。（注意**:build/dist/echarts.js**文件中没有注释和换行，不便于阅读）

init(dom,theme)函数的源代码见**源代码**部分。

//实现**option（这是使用ECharts最核心的工作）**

var option = {…}

// 为echarts对象**加载数据**

myChart.setOption(option);

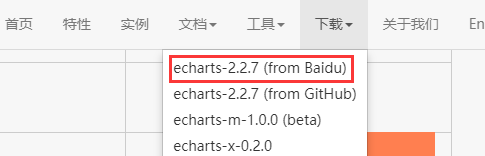
# 最好的参考资源：实例

**ECharts**可以说是**数据驱动的图表**，**大部分时候你关心的是那个option该如何实现**，官网的文档提供详细的说明，另外我们还有近100个实例，这些例子都为你展现了最核心的**option代码**，可以通过在线修改查看效果，ECharts是玩出来的，希望你玩得开心。

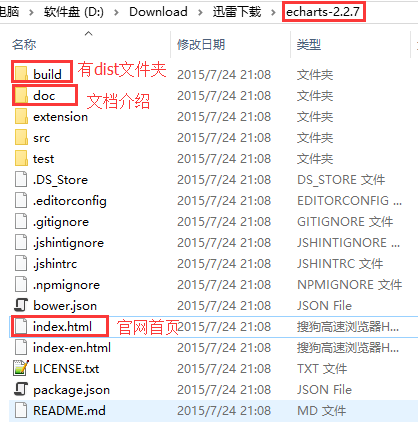
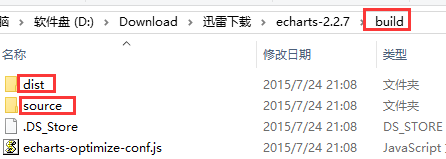
# echarts-2.2.7下载包介绍

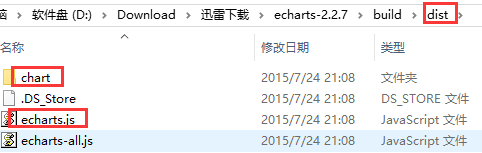
## ECharts下载包

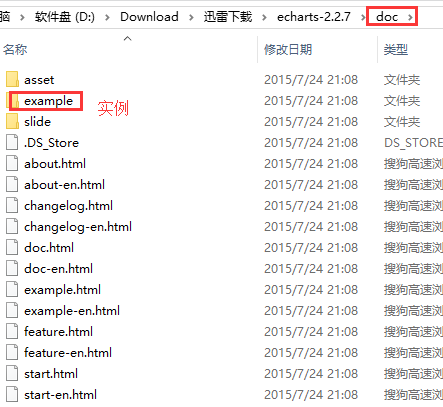
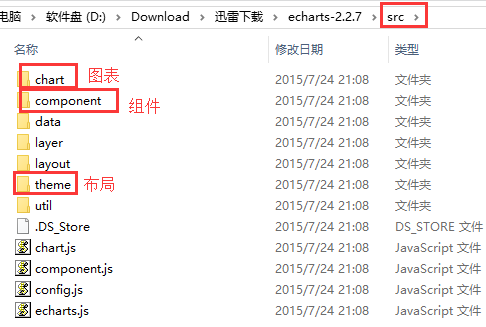
<http://echarts.baidu.com/echarts2/index.html>



## 下载包就是ECharts2整个官网一套文件



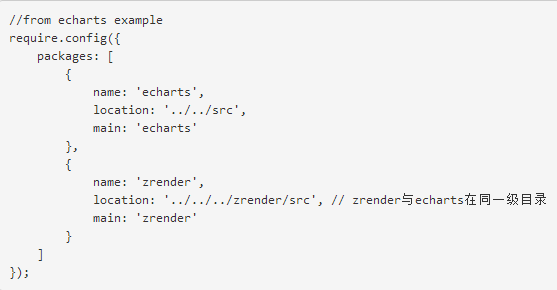
# 引入ECharts详细介绍

**echarts提供多种引入方式**，请根据你的项目类型选择合适的方式：

## 模块化包引入（packages）

如果你熟悉**模块化开发**，你的项目本身就是模块化且遵循AMD规范的，那引入echarts将很简单，使用一个符合AMD规范的模块加载器，如esl.js，只需要配置好packages路径指向src即可，你将享受到图表的按需加载等最大的灵活性，**由于echarts依赖底层zrender，你需要同时下载zrender到本地**，可参考demo，你需要配置如下。

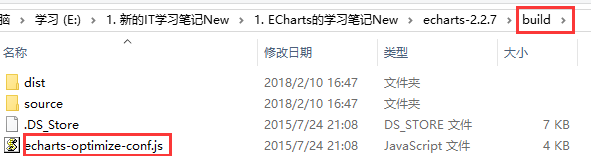
需要注意的是，包引入提供了开发阶段最大的灵活性，但并不适合直接上线，**减少请求的文件数量是前端性能优化中最基本但很重要的规则**，**务必在上线时做文件的连接压缩**。



## 模块化单文件引入（推荐）

如果你使用模块化开发但并没有自己的打包合并环境，或者说你不希望在你的项目里引入第三方库的源文件，我们建议你使用单文件引入，同模块化包引入一样，你需要**熟悉模块化开发**。

自2.1.8起，我们为echarts开发了**专门的合并压缩工具echarts-optimizer**。如你所发现的，**build文件夹下**已经包含了由**echarts-optimizer**生成的单文件：





**dist（文件夹）** : 经过合并、压缩的单文件

echarts.js : 这是包含AMD加载器的echarts主文件，需要通过script最先引入

**chart（文件夹）** : echarts-optimizer通过依赖关系分析同时去除与echarts.js的重复模块后为echarts的每一个图表类型单独打包生成一个独立文件，根据应用需求可实现图表类型按需加载

line.js : 折线图（如需折柱动态类型切换，require时还需要echarts/chart/bar）

bar.js : 柱形图（如需折柱动态类型切换，require时还需要echarts/chart/line）

scatter.js : 散点图

k.js : K线图

pie.js : 饼图（如需饼漏斗图动态类型切换，require时还需要echarts/chart/funnel）

radar.js : 雷达图

map.js : 地图

force.js : 力导向布局图（如需力导和弦动态类型切换，require时还需要echarts/chart/chord）

chord.js : 和弦图（如需力导和弦动态类型切换，require时还需要echarts/chart/force）

funnel.js : 漏斗图（如需饼漏斗图动态类型切换，require时还需要echarts/chart/pie）

gauge.js : 仪表盘

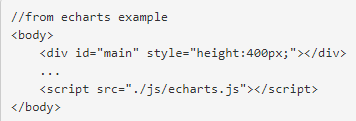
eventRiver.js : 事件河流图

treemap.js : 矩阵树图

venn.js : 韦恩图

**source（文件夹）** : 经过合并，但并没有压缩的单文件，内容同dist，可用于调试

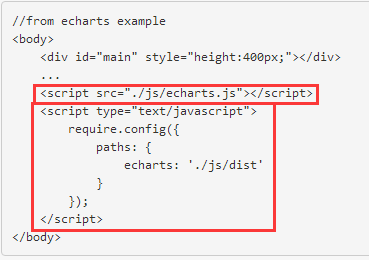
采用单一文件使用例子见**ECharts单一文件引入**，存放在example/www下，首先你需要通过script标签引入echarts主文件：



在主文件引入后你将获得一个**AMD环境**，**配置require.conifg**如下：



**require.config配置**后就可以通过**动态加载**使用echarts。





**总结来说**，模块化单文件引入ECharts，你需要如下4步：

（1）为ECharts准备一个具备大小（宽高）的Dom（当然可以是动态生成的）

（2）通过script标签引入echarts主文件

（3）为模块加载器配置echarts的路径，从当前页面链接到echarts.js所在目录，见上述说明；

（4）**动态加载echarts**及所需图表然后**在回调函数中开始使用**（容我罗嗦一句，当你确保同一页面已经加载过echarts，再使用时直接**require('echarts').init(dom)**就行）

## 标签式单文件引入

自1.3.5开始，**ECharts提供标签式引入**。如果你的项目本身并不是**基于模块化**开发的，或者是基于CMD规范（如使用的是seajs），那么引入基于AMD模块化的echarts可能并不方便，我们建议你采用srcipt标签式引入，忘掉require。**Srcipt标签引入echarts后将可以直接使用两个全局的命名空间：echarts，zrender，**可参考ECharts标签式引入，需要注意的是excanvas依赖body标签插入Canvas节点去判断Canvas的支持，如果你把引用echarts的script标签放置head内在IE8-的浏览器中会出现报错，解决的办法就是把标签移动到body内（后）。

标签式引入环境中，**常用模块的引用**可通过命名空间直取，**同模块化下的路径结构**，如：

echarts.config = require('echarts/config')，

zrender.tool.color = require('zrender/tool/color')



可以直接引入的**单文件**如下：

**dist/echarts-all.js** : 经过压缩，全图表，包含world，china以及34个省市级地图数据；

**source/echarts-all.js** : 未压缩，全图表，包含world，china以及34个省市级地图数据，可用于调试。

# 自定义构建echarts单文件

详见 echarts-optimizer 安装使用说明：**README.md**

# 初始化：init

通过require获得echarts接口（或者命名空间）后可实例化图表，echarts接口仅有一个方法**init**，执行init时传入一个具备大小的**dom节点**（width、height可被计算得到即可，不一定可见）后即可实例化出图表对象，图表库实现为多实例的，**同一页面可在多个dom上init出多个图表，同一个dom上多次init将自动释放已有实例**（1.4.0+）。init方法说明如下：

名称：{ECharts}init

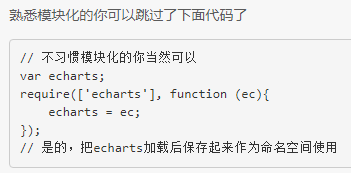
参数：{dom} dom, {string | Object =} theme

描述：**初始化接口，返回ECharts实例**，其中dom为图表所在节点，theme为可选的主题，内置主题（'macarons', 'infographic'）直接传入名称，自定义扩展主题可传入主题对象。如：

**var myCharts = echarts.init(document.getElementById('main'), 'macarons')；**

引入ECharts后的的初始化代码如下：





# 实例方法（ECharts的方法）

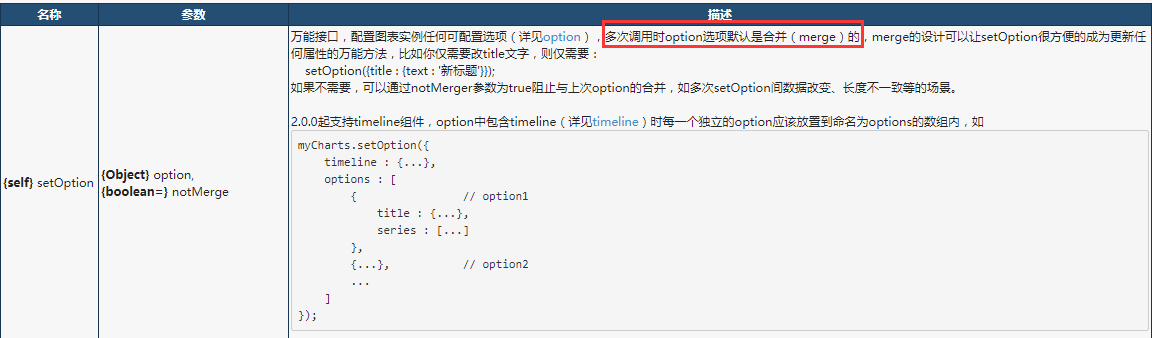
**实例**指的就是**接口init()返回的对象**，即上述代码中的“**myChart**”，非get接口均返回**自身self，支持链式调用。**

## setOption、getOption方法

**万能接口**，配置图表实例任何可配置选项（详见option），**多次调用时option选项默认是合并（merge）的**，merge的设计可以让setOption很方便的成为更新任何属性的万能方法，比如你仅需要改title文字，则仅需要：

**setOption({title : {text : '新标题'}});**

如果不需要，可以通过notMerger参数为true阻止与上次option的合并，如多次setOption间数据改变、长度不一致等的场景。

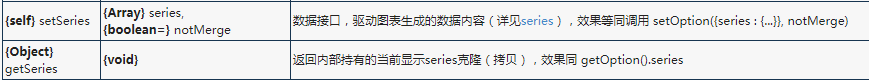


**getOption() : 返回内部持有的当前显示option克隆（拷贝）。**

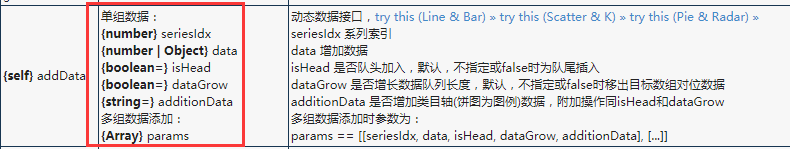


## setSeries、getSeries方法：

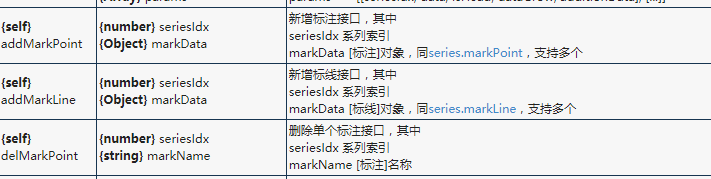
通过查看**源代码**，内部调用**setOption**和**getOption**实现的。



## addData



## addMarkPoint、addMarkLine、delMarkPoint、delMarkLine





**删除单个标线接口**，其中 seriesIdx 系列索引 ；markName [标线]名称，已构建的标线名称默认为markLine数据中起始点的name，如果同时终点也有name属性，如地图标线，则标线名称等于“nameStart > nameEnd”，如markLine的data为 [{name:'北京', value:100}, {name:'上海'}] ，则删除该标线时传入的markName为 "北京 > 上海"

## on

  
**事件绑定**，事件命名统一挂载到**require('echarts/config').EVENT（非模块化为echarts.config.EVENT）**命名空间下，建议使用此命名空间作为事件名引用，当前版本支持事件有：

### --基础事件-----------------------

REFRESH（刷新），

**RESTORE（还原），**

RESIZE（显示空间变化），

CLICK（点击），

DBLCLICK（双击），

HOVER（悬浮），

MOUSEOUT（鼠标离开数据图形），

### --交互逻辑事件--------------------

DATA\_CHANGED（数据修改，如拖拽重计算），

DATA\_VIEW\_CHANGED（数据视图修改），

MAGIC\_TYPE\_CHANGED（动态类型切换），

TIMELINE\_CHANGED（时间轴变化），

DATA\_ZOOM（数据区域缩放），

DATA\_RANGE（值域漫游），

DATA\_RANGE\_SELECTED（值域开关选择），

DATA\_RANGE\_HOVERLINK（值域漫游hover），

LEGEND\_SELECTED（图例开关选择），

LEGEND\_HOVERLINK（图例hover），

MAP\_ROAM（地图漫游），

MAP\_SELECTED（地图选择），

PIE\_SELECTED（饼图选择），

FORCE\_LAYOUT\_END（力导向布局结束）

## un：事件解绑定

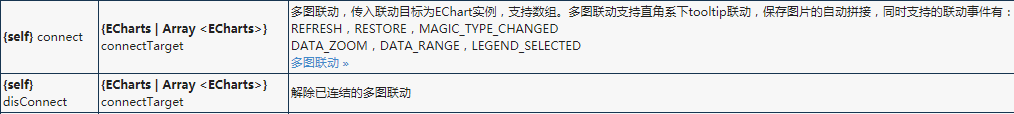


## setTheme：设置主题

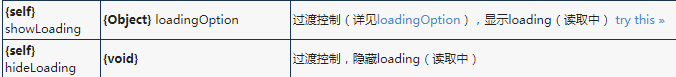


设置主题，内置主题（'macarons', 'infographic'）直接传入名称，自定义扩展主题可传入主题对象。

## connect、disConnect



## showLoading、hideLoading：显示或隐藏过渡控制

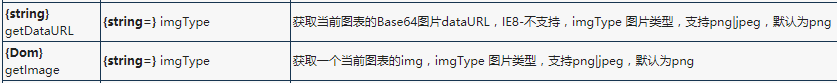


## getZrender

获取当前图表所用ZRender实例，可用于添加额外图形或进行深度定制



## getDataURL、getImage



## restore：还原图表



还原图表，各种状态均被清除，还原为最初展现时的状态。

## refresh：刷新图表



刷新图表，图例选择、数据区域缩放，拖拽状态均保持。

## clear：清空绘画内容



清空绘画内容，清空后实例可用。

## dispose:释放图表



释放图表实例，释放后实例不再可用。

# echarts的源代码分析

## self.init(dom,theme):入口方法，返回ECharts实例



## setOption方法：万能接口

/\*\*

\* **万能接口，配置图表实例任何可配置选项，多次调用时option选项做merge处理**

\* @param {Object} option

\* @param {boolean=} notMerge 多次调用时option选项是默认是合并（merge）的，

\* 如果不需求，可以通过notMerger参数为true阻止与上次option的合并

\*/

**setOption: function (option, notMerge)** {

if (!option.timeline) {

return this.**\_setOption**(option, notMerge);

}

else {

return this.**\_setTimelineOption**(option);

}

},

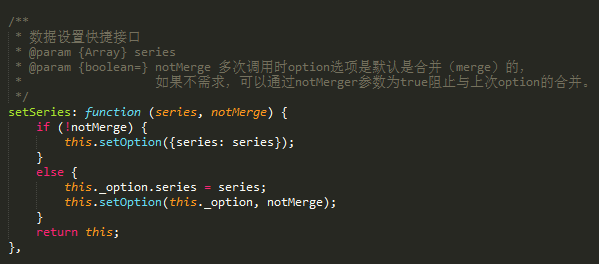
## getOption()：返回内部持有的当前显示option克隆

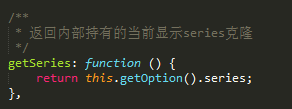


## setSeries(series,notMerge)、getSeries()

**setSeries**内部通过调用setOption方法实现;

**getSeries**内部通过调用getOption方法实现。





# 推荐加载ECharts方式：模块化单文件引入

模块化单文件引入，虽然引入时稍微麻烦一些，但是性能好，需要什么图，就加载什么图。

### 首先需要加载ECharts最基本的文件：

<script src="http://echarts.baidu.com/build/dist/echarts.js"></script>

### 其次就是js文件目录路径配置：

<script type="text/javascript">

// 路径配置

require.config({

**paths: {**

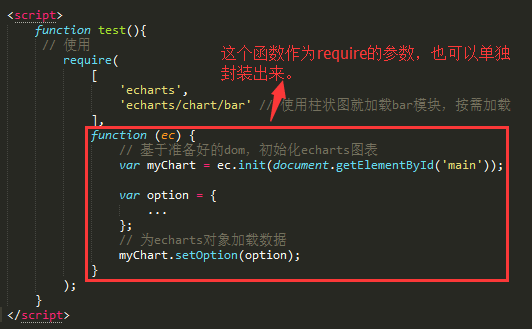
**echarts: 'http://echarts.baidu.com/build/dist'**

**}**

});

</script>

### 加载目录下所使用的文件和所有执行的函数：



### 测试：

如果想让页面一打开就显示图像，就不用封装到test函数里面，直接在<script>标签里写require()；如果想让点击按钮后，再触发显示图像，可以定义一个按钮，然后指定触发函数。

<!-- 为ECharts准备一个具备大小（宽高）的Dom -->

<**div id="main" style="height:400px"></div>**

<!-- 定义一个按钮 -->

**<button id = "btn" onclick="test()">按钮</button>**