**1.1.     YAHOO工具库提供的方法**

         **namespace**

用于创建一个全局的命名空间，使用YUI时，首先会自动创建widget,util,example三个命名空间，使用时也可以自定义命名空间。类似于在程序中建了了一个static变量。

         **lang**

javascript扩展的语言工具，用于判别对象的类型。

         **lang.extend**

用于从一个对象上扩展出另一个对象，模拟了类的继承的方式，但不同的是，在创建子对象时，父对象的构造函数不会自动调用。父对象的引用存放在了子对象的supperclass中，构成了一个链状继承关系。在2.2.2的版本中，YAHOO.lang.extend和YAHOO.extend指向同一函数对象。

         **lang.augment**

将一个对象的属性（部分或全部）复制到另一个对象，但并非真正意义上的复制，只是一种引用。YAHOO.augment=YAHOO.lang.augment。

         **log**

用来调试的一个工具，将信息显示到log控件。

         **env**

环境信息和YUI组件信息

         **YUI\_config.listener三**

可以定义自己的回调函数，当有新的YUI组件加载到页面时将会调用YUI\_config.listener指向的函数。

**2.            YUI提供的Dom操作**

**特点：**对于大部分DOM操作提供了批量操作的功能，而对用户只需使用统一的函数接口就能完成单个或批量的操作，主要得益于DOM内部的batch方法。

**2.1.     Element的查找**

**YAHOO.util.Dom.get(element)**

调用document.getElementById(element)，获取指定的页面元素。

**YAHOO.util.Dom.getElementsBy(method,tagName,rootNode)**

在rootNode的子节点中按照用户提供的method方法在所有标签为tagName的element中查找符合条件的节点。rootNode不指定则在整个Document中查找，method是一个method(elementID)类型的函数对象，该函数对象的返回值为Boolean值。

**YAHOO.util.Dom.getElementsByClassName(className, tagName, rootNode)**

返回指定根节点下所有标签为tagName,class为className的DOM节点数组。根节点为可选参数，不指定时在整个页面中查找

**YAHOO.util.Dom.inDocument (el)**

判断元素el是否在当前的DOM中，支持批量操作。

**2.2.     样式控制和访问**

**YAHOO.util.Dom.hasClass(element, className)**

判断element标签上是否指明了className的class，支持批量操作

**YAHOO.util.Dom.addClass(element, className)**

给指定标签增加名为className的class，支持批量操作.

**YAHOO.util.Dom.removeClass(element, className)**

删除element上的名为className的class，支持批量操作

**YAHOO.util.Dom.replaceClass(element, oldClassName, newClassName)**

替换element上的oldClassName样式为newClassName，支持批量操作

**YAHOO.util.Dom.getStyle(element, property)**

获取element的style中的property属性，支持批量操作

**YAHOO.util.Dom.setStyle(element,property,pValue)**

设置element的style的property属性为pValue，支持批量操作

**注：本节中的class指的是CSS中定义的class。**

**2.3.     位置控制和访问**

位置控制的相关函数

**YAHOO.util.Dom.setX**

**YAHOO.util.Dom.setY**

**YAHOO.util.Dom.setXY**

**YAHOO.util.Dom.getX**

**YAHOO.util.Dom.getXY** 返回元素坐标 [ left,top ]

**YAHOO.util.Dom.getRegion**

获取元素的坐标Region对象{left,top,right,bottom}

可支持批量操作

获取页面可视面积的高度和宽度

**YAHOO.util.Dom.getClientWidth**

**YAHOO.util.Dom.getClientHeight**

获取Document的高度和宽度

**YAHOO.util.Dom.getDocumentWidth**

**YAHOO.util.Dom.getDocumentHeight**

获取页面可视区域的高度和宽度（不包含滚动条）

**YAHOO.util.Dom.getViewportHeight**

**YAHOO.util.Dom.getViewportWidth**

**Region对象：{left,top,right,bottom}**

**YUI提供的一个对象，用于完成多个矩形区域间的计算（如相交，包含。**

**YAHOO.util.Region.contains(region)**

判断是否包含了region区域

**YAHOO.util.Region.getArea**

计算面积

**YAHOO.util.Region.intersect(region)**

计算与region区域的交迭区域

**YAHOO.util.Region.union(region)**

计算与region区域求并集（即包含两个区域的最小区域）

**Point对象：{x,y}**

**YUI提供的对象，用于定义坐标点。**

**3.            YUI提供的element工具**

YUI提供了一组操作页面element的工具，是对标准HTML elements的一种封装，能够直接通操作element的实例，使得增加监听器，操作DOM，设置/获取element的属性等工作变得很简单。部分方法是直接调用YUI的DOM工具集提供的方法，如对class操作的相关方法、获取element的相关方法等等，在此不再重复。

**YAHOO.util.Element(elementId)**

创建element，如果elementId在Document中还不存在，仍然可以通过YUI对他进行属性设置，增加监听器等操作， Element工具集会自动等到该elementId可用后执行这些操作，实际上真正的操作是等到contentReady事件发生后才进行的。

**YAHOO.util.Element.appendChild(child)**

在DOM结构中element下增加子节点

**YAHOO.util.Element.getElementsByTagName (tag)**

获取tagName为tag的所有页面元素

**YAHOO.util.Element.hasChildNodes**

判断是否具有子节点

**YAHOO.util.Element.insertBefore (element, before)**

在元素before前插入element

**YAHOO.util.Element.removeChild(child)**

删除DOM中元素的child子节点

**YAHOO.util.Element.replaceChild (newNode , oldNode)**

替换子节点oldNode为newNode

**4.            YUI提供的Event工具集**

YUI提供的Event工具集简化了浏览器中事件驱动程序的编写，提供了一种简单的接口来定制事件和检查浏览器中的event对象。YUI事件工具集提供了自定义事件对象(Custom Event)，通过自定义事件对象可以“发布”自己感兴趣的时刻或事件，页面中的YUI组件能够响应这些自定义的事件并做出回应。

**YUI对事件响应的顺序**：通过YUI Event工具集添加的事件，默认是在冒泡过程中执行事件处理函数的。从DOM节点上来说，是从子节点向根节点响应事件。

**Event需要的引入**

<!-- Dependency -->

build/yahoo/yahoo-min.js

<!-- Event source file -->

build/event/event-min.js

Event和Custom Event分别定义在YAHOO.util.Event和YAHOO.util.CustomEvent中

**Event工具集提供的方法**

**YAHOO.util.Event.addListener(element,eventType,fn,obj,override)**

**参数：**

element：为绑定事件的元素id，可以是一个数组，以支持批量操作

eventType：为事件类型

fn：为事件响应的回调函数

obj：当override为true时，为回调函数传入的参数对象；当override为false时，该参数被忽略。

override：

**返回值类型：**Boolean

**功能：**给指定的element绑定事件响应函数

**YAHOO.util.Event.removeListener:function(element,eventType,fn)**

**参数：**

element：为绑定事件的元素id,

eventType：事件类型

fn：为事件响应函数

**返回值类型：**Boolean

**功能：**给指定的element解除绑定事件

**YAHOO.util.Event.purgeElement ( el , recurse , sType )**

**参数：**

el：为绑定事件的元素id,

recurse：Boolean值，是否解除子节点的事件绑定

sType：事件类型

**返回值类型：**Boolean

**功能：**给指定的element解除绑定的同一类型的事件，也可以解除子节点上绑定的这一类型的事件

**YAHOO.util.on**

addListener的函数别名

**YAHOO.util.Event.onAvailable** **( p\_id , p\_fn , p\_obj , p\_override )**

**参数：**

p\_id：为绑定事件的元素id,

p\_fn：为事件响应函数

p\_obj：同addListener的obj参数

p\_override：同addListener的override参数

**返回值类型：**无

**功能：**当指定的element的p\_id出现时，执行事件响应函数。如果在页面初始化之前执行这一函数，当页面加载时（可能还未完成时），就会执行响应的事件响应函数；如果放在页面加载之后执行这一函数，将以固定的时间轮询，当element可用时响应这一事件。这个轮询的的时间是可以配置的，缺省的时间是10秒一次。

**YAHOO.util.Event.onContentReady** **( p\_id , p\_fn , p\_obj , p\_override )**

**参数：**

p\_id：为绑定事件的元素id,

p\_fn：为事件响应函数

p\_obj：同addListener的obj参数

p\_override：同addListener的override参数

**返回值类型：**无

**功能：**与onAvailable类似，但不同的是事件响应函数是等到element可以安全的修改的时候才响应。

**YAHOO.util.Event.onDOMReady ( p\_fn , p\_obj , p\_scope )**

**参数：**

p\_fn：为事件响应函数

p\_obj：同addListener的obj参数

p\_scope：同addListener的override参数

**返回值类型：**无

**功能：**当DOM第一次可用时执行响应函数。

**YAHOO.util.Event.preventDefault ( event)**

**参数：**

event：事件对象

**返回值类型：**无

**功能：**阻止事件的缺省行为发生。

**YAHOO.util.Event.getListeners ( el , sType )**

**参数：**

el：HTML element

sType：事件类型，String类型

**返回值类型：**Object{

type：事件类型

fn：addListener添加的事件响应函数

obj：提供给事件响应函数的参数对象

adjust：否获取缺省的事件监听器

index：UI事件监听器列表中的位置

}

**功能：**阻止事件的缺省行为发生。

**YAHOO.util.Event.getTime( event)**

**参数：**

event：事件对象

**返回值类型：**Date对象

**功能：**获取事件发生时的时间。

**YAHOO.util.Event.getTarget(ev , resolveTextNode)**

**参数：**

evt：事件对象

resolveTextNode：

**返回值类型：** HTML element

**功能：**获取事件发生时的页面标签。对于IE即window.event.srcElement

**5.            YUI提供的CSS样式**

**5.1.     Fonts**

字体的规范样式，需要引入build/fonts/fonts-min.css

**5.2.     Reset**

规范了所有HTML element的缺省样式，需要引入build/fonts/reset-min.css

**5.3.     Grids**

提供了用于页面排版的CSS样式，需要引入build/fonts/grids-min.css

**6.            YUI组件**

**6.1.     Connection Manager**

提供了访问XMLHttpRequest对象的一个简单接口

**对象定义：**

YAHOO.util.Connect.asyncRequest

**Connection的引入：**

<!-- Dependency -->

build/yahoo/yahoo-min.js

<!—可选项: 用到事件时引入 -->

build/event/event-min.js

build/connection/connection-min.js

**Connection的使用**

1.     创建对象

var transaction = YAHOO.util.Connect.asyncRequest('GET', sUrl, callback, null);

第一个参数：指明http请求的方式，可用的方式包括GET、POST、HEAD、PUT、DELETE，但PUT和DELETE可能在一些A级浏览器上不支持。

第二个参数：请求的URL

第三个参数：回调函数，用于服务器返回数据时调用的客户端处理程序

第四个参数：POST方式时，提供给URL的POST参数信息。

**[注]**：A级浏览器A-Grade browsers是YUI对浏览器的一个等级划分，具体划分原则详见http://developer.yahoo.com/yui/articles/gbs/index.html

2.     定义回调函数

       在异步事物中，可以创建回调函数处理服务器的响应和相关数据，如果你不关心服务器的返回信息，也可以忽略这些回调函数，所有这些回调函数对象都是可选的，然而在大多数情况下，应该至少提供以下三个回调函数：

success：服务器做出有效响应时的回调函数

failure：服务器响应了但提供了错误信息时的回调函数

argument：success和failure为了处理返回信息需要的参数，可以是对象、字符串、数字或者包含了数据的数组。

         在使用YAHOO.util.Connect.setForm上载文件时，需要定义upload回调函数代替success和failure

         在回调函数中this将失去作用范围，这种情况下需要通过一个指向父对象的引用的参数来访问对象的成员。为了能够使用对象的方法作为回调函数，并维持成员的作用范围，需要定义回调函数对象的成员scope，作为this的值。

**6.2.     Button**

**对象定义：YAHOO.widget.Button**

与传统HTML Form的按钮类似，不同的是它的label可以与 value不一致。还可以创建带菜单的按钮，或者radio button、checkbox

**分类：**

可以创建几种类型的按钮：

button:普通的下压式按钮，可以在按钮按下时执行用户指定的代码

link:按下时导航至相应的URL

submit:作用相当于form的提交按钮

reset:form的reset按钮

checkbox:

radio:

menubutton:按下时显示隐藏按钮

splitbutton:按下时执行命令或显示菜单的按钮

**使用Button必需的引入:**

<!-- CSS -->

build/fonts/fonts-min.css

build/button/assets/button.css

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

build/element/element-beta-min.js

<!—使用menubutton和splitbutton要用到的 -->

build/container/container\_core-min.js

build/menu/menu-min.js

<!-- Source file -->

build/button/button-beta-min.js

**初始化的方式：**

1． 使用替换<input>或<span>的方式

**constructor：**YAHOO.widget.Button(sourceElementId,{配置集})

Button的构造器首先根据sourceElementId 在DOM中查找<input>，一旦找到，就通过DOM的 replaceChild方法替换掉<input>

2． 使用新建的方式

constructor:YAHOO.widget.Button({配置集})

这种情况下根据配置集中指明的父id(Container)创建按钮，如果配置集中没有指明按钮则使用YAHOO.Dom.generateId生成button的ID,

**Button的配置集**

|  |  |  |
| --- | --- | --- |
| 属性名称 | 含义 | 备注 |
| id | 替换后的新的element id |  |
| label | 按钮上显示的文字 |  |
| check | 按钮选中/未选中的状态 | checkbox用到 |
| type | [指明button的类型](http://writeblog.csdn.net/Editor/FCKeditor/editor/fckeditor.html?InstanceName=ctl00_ContentPlaceHolder1_EntryEditor1_richTextEditor_richTextEditor&Toolbar=Default#_分类：) | 缺省为button |
| container | 按钮的父id | 使用新建方式时会用到 |
| srcelement | 使用替换方式时 |  |
| menu | 按钮对应的菜单YAHOO.widget.menu |  |
| title | 按钮title | 没有指定label时使用title |
| href | 按钮导航URL | 仅在按钮类型为link时有效 |
| target | <a>中的target | 仅在按钮类型为link时有效 |
| tabindex | 按tab切换焦点时的顺序号 |  |
| onclick | click事件的响应函数 | onclick:{  fn: Function, // 事件的响应函数. obj: Object, // An object to pass back to the handler. scope: Object // The object to use for the scope of the handler. } } |
|  |  |  |

**6.3.     ButtonGroup**

**对象定义：YAHOO.widget.ButtonGroup**

ButtonGroup是一组按钮，同组中只能有一个按钮被选中。根节点为<div>

**6.4.     AutoComplete**

用户在文本输入框中输入文字时，该组件通过输入的内容查找符合输入条件的内容，并显示出所有符合条件的内容，供用户能够很快的完成正确的输入。

**AutoComplete的引入**

<!-- 必须引入 -->

build/yahoo-dom-event/yahoo-dom-event.js

<!-- 可选: Connection(用到 XHR数据源时需要的引入) -->

build/connection/connection-min.js

<!--可选: 用到动画效果时需要的引入 -->

build/animation/animation-min.js

<!-- Source file -->

build/autocomplete/autocomplete-min.js

**控制AutoComplete的配置集**

|  |  |  |
| --- | --- | --- |
| 属性名称 | 含义 | 备注 |
| animVert | 控制下拉框向下展开的动画效果 |  |
| animHoriz | 控制下拉框水平方向展开的动画效果 |  |
| animSpeed | 控制动画的速度 |  |
| delimChar | 一行显示多条记录时的分隔符，可以是字符串数组或字符串 |  |
| maxResultsDisplayed | 结果集的最大显示行数 |  |
| minQueryLength | 进行查询前的输入的字符个数 |  |
| queryDelay | 用户键入字符后多少秒开始查询，默认0.5 |  |
| autoHighlight | 查询出来的结果集显示后，是否高亮显示第一条，默认为true |  |
| highlightClassName | 高亮显示的样式名，默认为yui-ac-highlight |  |
| prehightlightClassName | 鼠标移动到下拉框的一行上时，那一行的样式，默认为yui-ac-prehighlight |  |
| useShadow | 下拉框是否有阴影 |  |
| useIFrame | 下拉框欠套在一个iframe中，用于解决IE中覆盖<select>的问题 |  |
| forceSelection | 限制输入内容必须为查询结果中的内容，如果不是则输入内容被删除 |  |
| typeAhead | 是否自动根据查询结果的第一条补齐到autocomplete中 |  |
| allowBrowserAutocomplete | 是否允许浏览器提供的输入框记忆功能，默认为false |  |
| alwaysShowContainer | 是否始终显示查询结果的下拉框，默认为false |  |

**AutoComplete的使用**

**1          自定义事件(CustomEvent)**

AutoComplete除了默认的事件外，还有自定义的事件，以下为AutoComplete用到的各种CustomEvent

|  |  |  |
| --- | --- | --- |
| 事件回调函数 | 含义 | 备注 |
| textboxFocusEvent | 输入框获得焦点的事件 | textboxFocus |
| textboxKeyEvent | 按键弹起时的事件 | textboxKey |
| dataRequestEvent | 发出数据查询请求时的事件 | dataRequest |
| dataReturnEvent | 数据查询请求返回时的事件 | dataReturn |
| dataErrorEvent | 返回结果为空时的事件 | dataError |
| containerExpandEvent | 展开下拉框时的事件 | containerExpand |
| typeAheadEvent | 符合条件的结果自动用到第一条记录补齐时的事件 | typeAhead |
| itemMouseOverEvent | 鼠标移动到结果项上的事件 | itemMouseOver |
| itemMouseOutEvent | 鼠标移出结果项的事件 | itemMouseOut |
| itemArrowToEvent | 移动到某一选择项时的事件 | itemArrowTo |
| itemArrowFromEvent | 移出某一选择项时的事件 | itemArrowFrom |
| itemSelectEvent | 选中某一选择项的事件 | itemSelect |
| unmatchedItemSelectEvent | forceSelection为false时，如果输入了不符合查询结果的内容时触发本事件。 | unmatchedItemSelect |
| selectionEnforceEvent | 强制从结果集中选择输入项时触发的事件 | selectionEnforce |
| containerCollapseEvent | 下拉框收缩时的事件 | containerCollapse |
| textboxBlurEvent | 输入框失去焦点时的事件 | textboxBlur |

**可以根据需要重载自定义的事件**

例：输入框获得焦点时自动进行查询，可以重载textboxFocusEvent

myAutoComp.textboxFocusEvent.subscribe(

//发送一个查询条件为空的查询请求

function(){myAutoComp.sendQuery("");}

);

**2          格式化下拉框的输出**

如果返回的结果集中有多个属性，可以通过重载formatResult来格式化结果项的输出

oAutoComp.formatResult = function(oResultItem, sQuery) {

var sMarkup = oResultItem[0] + " (" + oResultItem[1] + ")" ;

return (sMarkup);

}

**3          AutoComplete的数据源**

         **DS\_JSArray**

data = ["AAA","BBB","BCC"];

var myDataSource = new YAHOO.widget.DS\_JSArray(data);

可以通过DWR的方式调用java方法获得一字符串数组作为查询的下拉框，并在回调函数中可以直接使用，如：

         **DS\_JSFunction**

         **DS\_XHR**

对于JSArray方式取数时，AutoComplete可以自动根据输入的内容查在JSArray中查找符合条件的结果项，但在使用XHR方式时，YUI并没有提供默认的查找功能，必须由用户通过控制URL参数方式来实现这一功能。

**JSON方式：**

使用JSON格式时，创建数据源YAHOO.widget.DS\_XHR时第一个参数为请求的URL，第二个参数为一数组，这一数组的第一条记录表示结果集在JSON对象的哪个对象属性下，如类似于下面结构的JSON对象

{

Result:{

data:[ {username:”Tom”,oldname:”Tom”,age:12”},

{username:”Jack”,oldname:”Jack”,age:16”}

]

}

}

如果是要查询username则结果集应该表示为”Result.data”，

第二条记录为搜索的主键，同上面的例子中如果是要查找出符合条件的username，则第二个数组应为”username”,如果返回的结果集中还需要返回oldname属性，则应该指名数组的第三条记录”oldname”,后面还可以指定更多的需要返回的属性。

因此第二个数组对象参数可以用以下方法表示：

[结果对象,搜索的主键,额外的返回属性1,…额外的返回属性N]

function initAutocomplete(){

var myDataSource1 = new YAHOO.widget.DS\_XHR(

"./queryUsername.action",

["data","username"]);

myDataSource1.responseType = YAHOO.widget.DS\_XHR.TYPE\_JSON;

//myDataSource1.scriptQueryAppend = "output=json&results=100";

var oAutoComp = new YAHOO.widget.AutoComplete("user\_username",

"usernameContainer", myDataSource1);

oAutoComp.dataRequestEvent.subscribe(

function(oSelf,sQuery){

myDataSource1.scriptQueryParam = "username";

});

}

**XML方式：**

应用方式与JSON方式类似，只是将reponseType指定为

myDataSource1.responseType = YAHOO.widget.DS\_XHR.TYPE\_XML

**FlatData方式：**

**6.5.     DataTable**

**DataTable的引入**

<!—需要的CSS -->

build/datatable/assets/datatable.css

<!--必须的引入 -->

build/yahoo-dom-event/yahoo-dom-event.js

<!-- 可选项，用到XHR取数方式时需要 -->

build/connection/connection-min.js

<!—可选项，用到调整列宽时需要 -->

build/dragdrop/dragdrop-min.js

<!-- Source files -->

build/datasource/datasource-beta-min.js

build/datatable/datatable-beta-min.js

**DataTable的使用**

**1.     定义表头**

var myColumnHeaders = [

{key:"name", text:"姓名"},

{key:"stdno", text:"学号"},

{key:"age", text:"年龄", type:"number"}

];

YAHOO.widget.ColumnSet的属性说明

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| key | 列的属性名 |
| text | 列的显示名称 |
| type | 列的类型，包括"string", "number", "date", "currency", "checkbox", "select", "email", "link"，缺省为”string” |
| resizeable | Boolean值，是否可通过拖动改变列宽度 |
| sortable | Boolean值，是否可排序 |
| abbr |  |
| children | 定义子表头，类型为YAHOO.widget.ColumnSet |
| width | 列宽度，单位为px |
| className | 定义本列单元格的样式名称 |
| formatter | 定义本列单元格的格式化函数  function(elCell, oRecord, oColumn, oData) |
| parse |  |
| editor | 定义可编辑的列，可用的值为”textbox”，”textarea” |
| descFunction | 递减排序的函数function(a,b) |
| ascFunction | 递增排序的函数function(a,b) |

**2.     创建datatable**

var myDataTable = new YAHOO.widget.DataTable(

"myContainer", //datatable绑定的页面element(DIV)

myColumnSet, //表头定义, YAHOO.widget.ColumnSet

myDataSource,  //数据源

   {caption:"My Caption",summary:"摘要"} //datatable的配置集

);

**DataTable的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| caption | 表格的表头文字 |
| summary | 表格摘要 |
| paginator | Boolean值，是否分页，缺省为false |
| paginatorOptions | {  containers:  rowsPerPage: 每页显示的记录数,  pageLinks: 最多显示的页面链接数，0为全部  currentPage:当前页  dropdownOptions:下拉框选择项，null表示不使用下拉框  } |
| initialRequest | XHR方式下，附加的请求参数 |
| fixedWidth | 固定表格宽度 |
| scrollable | 表格有滚动条，滚动时表格表头保持不动，Boolean值 |
| rowSingleSelect | 只允许选择一行，Boolean值 |
| contextMenu | 表格的上下文菜单，右键弹出，YAHOO.widget.ContextMenu |
| sortedBy | { colKey：排序的列,  dir: 排序的方向，"desc"和"asc" } |
| pageCurrent | 表格的当前页 |

**3.     DataTable的数据源**

**DS\_JSArray**

以对象数组的方式使用，可以是在客户端定义对象数组，也可以通过DWR的方式调用JAVA的方法获取一个JAVA类的List列表，在回调函数中以数组方式使用。

var commonDataSource = new YAHOO.util.DataSource(data);

//data可以为用javascript定义的对象数组，也可以是DWR方式下回调函数的参数

commonDataSource.responseType = YAHOO.util.DataSource.TYPE\_JSARRAY;

**JSON**

使用JSON对象时

var myDataSource = new YAHOO.util.DataSource("query.action");

myDataSource.responseType = YAHOO.util.DataSource.TYPE\_JSON;

myDataSource.responseSchema = {

resultsList: "result.data",// 结果集所在的JSON结构中的对象

fields: ["id","username","email","monicker","edit","del"]

};

**XML**

var myDataSource = new YAHOO.util.DataSource("query.action");

myDataSource.responseType = YAHOO.util.DataSource.TYPE\_XML;

myDataSource.responseSchema = {

resultNode: "Item", // 结果集所在的XML结构中的结点

fields: ["Company","Title","Name","Phone","Email"] //表格的列

};

**Plain Text Data**

var myDataSource = new YAHOO.util.DataSource("query.action");

myDataSource.responseType = YAHOO.util.DataSource.TYPE\_TEXT;

myDataSource.responseSchema = {

recordDelim: "\n", // 记录的分割符

fieldDelim: ",", // 字段的分割符

fields: ["Company","Title","Name","Phone","Email"] //表格的列

};

**4.     DataTable数据的访问**

这里主要指通过javascript访问DataTable数据时采用的方式。

var rs = YAHOO.Claim.commonDataTable.getRecordSet();

var obj = rs.getRecord(i); //获取第i行的数据对象

对象obj的属性就是定义表头时指定的各列的key值，可以通过obj[key值]访问该行上的各列属性值。

**5.     测试**

**6.6.     DragDrop**

**6.7.     TreeView**

**6.8.     Container**

一组模拟windows控件的组件

**Container下的控件继承关系**

|  |
| --- |
| **Overlay** |

|  |
| --- |
| **Module** |

|  |
| --- |
| **ToolTip** |

|  |
| --- |
| **Dialog** |

|  |
| --- |
| **SimpleDialog** |

|  |
| --- |
| **Panel** |

**Container的引入**

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

<!—可选项：如果不使用Dialog/SimpleDialog则不需要 -->

build/connection/connection-min.js

<!—可选项：使用动画效果时用到 -->

build/animation/animation-min.js

<!-- Source file -->

build/container/container-min.js

**6.8.1.        Container下的控件**

由于Container下的控件的创建方式和属性控制方式基本相同，可以在创建组件时通过config对象指定控件的各项属性，也可以在创建后通过myContainer.cfg.setAttribute来设置控件属性，通过对这些属性的控制来实现控件的不同动作表现。因此对于各个子控件的属性在下面采用配置集的方式用一个表格来说明。

**Module**

一种最基础的控件模块，通过Module可以创建一个从HTML中继承下来的javascript对象，你可以操作页面中的module,也可以动态创建新的module。module是没有预定义样式的，必须手动定义。

**使用Module必需的引入**

build/yahoo-demo/yahoo-dom-event.js

build/container/container\_core-min.js

**初始化的方式：**

**constructor：**YAHOO.widget.Module(ModuleId,{配置集});

moduleId如果页面中已经定义了moduleId,则可以直接调用render函数将设置的各种属性直接应用到module上，如果页面中没有定义这个id,则必须调用render(DOM中已经存在的element)将module插入到DOM中。

**显示/隐藏Module**

方式一：

myModule.show();

myModule.hide();

方式二：

myModule.cfg.setProperty("visible", true);

myModule.cfg.setProperty("visible", false);

**配置集的说明**

|  |  |  |
| --- | --- | --- |
| 属性名称 | 含义 | 备注 |
| visible | Module是否可见，Boolean值 |  |
| monitorresize | 设置是否创建一个不可见的屏幕元素用来监听DOM中文字大小的改变，Boolean值 |  |

**6.8.2.        Overlay**

对Module的一种扩展，但和Module不同的是Overlay可以通过绝对方式定位，并且是浮动窗口，不会影响页面中原有的内嵌标签的布局。与Module相同的是：也没有预先定义样式，必须手动设置。

**初始化的方式**

与Module相同

**位置设定**

1． 绝对定位：

myOverlay.cfg.setProperty("x", 100);

myOverlay.cfg.setProperty("y", 200);

myOverlay.cfg.setProperty("xy", [100,200]);

2． 居中

窗口中居中：myOverlay.center();

屏幕居中：myOverlay.cfg.setProperty("fixedcenter", true);

3． 对齐

// myOverlay's右上角对齐myContextEl的左上角

myOverlay.cfg.setProperty("context", ["myContextEl", "tr", "tl"]);

设置过对齐方式后可以直接使用align来改变对齐方式

// myOverlay's 右上角对齐 "myContextEl"'s 左下角.

myOverlay.align("tr", "bl");

注：对齐方式的写法: top->t ,left ->l, ,bottom->,right->r

top-left 就表示为tl

**配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| x | 设置元素的left属性 |
| y | 设置元素的top属性 |
| xy | 设置元素的top,left属性 |
| context | 停靠位置，[element, Overlay的顶角, 被停靠的element顶角] |
| effect | 当Overlay显示或隐藏时的特效对象. |
| fixedcenter | 窗口大小改变或滚屏时保持居中 |
| width | 设置元素的width |
| height | 设置元素的height |
| zIndex | 设置Overlay的遮盖顺序 |
| constraintoviewport | 设置为true时保持在视窗边界以内 |
| iframe | 设置为true时在Overlay后将存在一个ifram,防止被Z序高的element覆盖 |

已知的特效：

YAHOO.widget.ContainerEffect.SLIDE

YAHOO.widget.ContainerEffect.FADE 逐步变淡

**OverlayManage**

用于管理多个Overlay的组件，能够模拟类似于多窗口管理系统的交互，使得各个Overlay之间的能够正确的覆盖、获得焦点/失去焦点。

使用这一功能需要的代码很简单，只需要通过register方法登记所有需要纳入窗口管理的Overlay就可以了。

//示例：

YAHOO.namespage(“demo”);

YAHOO.demo.manager = new YAHOO.widget.OverlayManager();

YAHOO.demo.manager.register( [Overlay的数组列表] );

**6.8.3.        Tooltip**

**对象定义：**YAHOO.widget.Tooltip

从overlay上扩展出来的控件，小的浮动窗口，动态生成，提供了预定义的样式。

**Tooltip的引入**

<!-- CSS -->

build/container/assets/container.css

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

<!—如果要改变页面元素的属性需要引入-->

build/animation/animation-min.js

<!-- Source file -->

build/container/container-min.js

**Tooltip初始化方式**

**constructor：**YAHOO.widget.Tooltip(tooltipId , { 配置集 } );

tooltipId可以是页面中未声明的控件。

**Tooltip配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| text | 提示文字 |
| context | 鼠标停靠element ID |
| container | 缺省为document.body，指明容器元素 |
| preventoverlap | 缺省为true,指明是否使用覆盖方式 |
| showdelay | 显示tooltip的延迟时间，单位毫秒 |
| hidedelay | 隐藏tooltip的延迟时间，单位毫秒 |
| autodismissdelay | The number of milliseconds to wait before automatically dismissing a Tooltip after the mouse has been resting on the context element. |

同时还继承了Module,Overlay的所有属性

**6.8.4.        Panel**

从overlay扩展出来的一个组件，类似于操作系统的窗口。他不同于浏览器的弹出窗口，是内嵌在页面文件中的一种动态Html的element。他在overlay的基础上扩展了一些窗口的功能，如关闭、拖放等。Panel使用了预定义的样式表

**Panel的引入**

<!-- CSS -->

build/container/assets/container.css

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

<!—可选项：如果用到动画时需要引入 -->

build/animation/animation-min.js

<!—可选项：如果用到拖放时需要引入-->

build/dragdrop/dragdrop-min.js

<!-- Source file -->

build/container/container-min.js

**Panel的初始化**

**constructor：**

YAHOO.widget.Panel(panelId); //使用缺省配置选项

YAHOO.widget.Panel(panelId,{自定义的配置项}); //使用自定义的配置项

**Panel的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| close | 是否显示关闭按钮，true/false |
| draggable | 是否可以拖动 |
| underlay | 阴影显示方式 shadow,none,matte,缺省为shadow，none为无阴影，matte为白色边界 |
| modal | Panel获得焦点时，是否允许使用Panel以外的控件获得焦点，类似于MFC的模态/无模态对话框 |
| keylisteners | 响应按键事件的监听器列表 |

Panel同时还继承了Module,Overlay的属性

**6.8.5.        Dialog**

从Panel上扩展出来的组件，类似于Windows的窗口，在浏览器中提供了一种不用页面跳转就能使用交互来获取用户输入的交互方式，用户输入的数据都是通过一个标准的HTML Form获取的，并支持多种的获取输入数据的方式：普通的Form的提交，XMLHttpRequest，或者完全通过脚本来读取。

**Dialog的引入**

<!-- CSS -->

build/container/assets/container.css

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

<!—可选项: Animation (only required if enabling Animation) -->

build/animation/animation-min.js

<!—可选项: Drag & Drop (only required if enabling Drag & Drop) -->

build/dragdrop/dragdrop-min.js

<!--可选项: Connection (only required if performing asynchronous submission) -->

build/connection/connection-min.js

<!-- Source file -->

build/container/container-min.js

**Dialog的初始化**

**constructor：**YAHOO.widget.Dialog(dialogId);

dialogId为页面中已经申明的element

**Dialog的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| postmethod | 窗口提交的方式，async：异步，form：标准的submit，none：不提交 |
| buttons | dialog中的button数组列表， |

同时继承了Overlay，Panel的属性

单个按钮对象的定义方式：

{ text:按钮上的文本,

handler:按钮的click事件响应函数

isDefault:是否缺省按钮},

**6.8.6.        SimpleDialog**

从Dialog上扩展出来的一个组件,主要用于处理用户选择Yes/NO，OK/Cancel的交互

**SimpleDialog的引入**

<!-- CSS -->

build/container/assets/container.css

<!-- Dependencies -->

build/yahoo-dom-event/yahoo-dom-event.js

<!—可选项：用到动画效果时使用 -->

build/animation/animation-min.js

<!—可选项：用到拖放功能时使用-->

build/dragdrop/dragdrop-min.js

<!—可选项：用到异步提交方式时使用) -->

build/connection/connection-min.js

<!-- Source file -->

build/container/container-min.js

**SimpleDialog的初始化**

**constructor：**YAHOO.widget.SimpleDialog(dialogId,{配置集});

dialogId为页面中已经申明的element

**SimpleDialog的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| text | 对话框中显示的提示信息 |
| icon | 按钮图标：  ICON\_BLOCK,  ICON\_WARN,  ICON\_HELP,  ICON\_INFO,  ICON\_ALARM,  ICON\_TIP. |

**6.9.     Grids**

支持不规定宽度的设计

一组用于定义界面版式的样式表(CSS文件)

**6.10.            TabView**

用来创建导航tab栏组件

可以从HTML中的已经存在的标签来创建，也可以完全通过javascript创建，该组件的最上层元素为<div>，每一个tab页面项使用<li>

**TableView的引入**

<!-- Dependencies -->

<!-- core CSS -->

/build/tabview/assets/tabview.css

<!-- optional skin for border tabs -->

build/tabview/assets/border\_tabs.css

build/yahoo-dom-event/yahoo-dom-event.js

build/element/element-beta-min.js

<!—可选项: 使用动态加载时使用-->

build/connection/connection-min.js

<!-- Source file -->

build/tabview/tabview-min.js

**TableView的初始化**

**constructor：**YAHOO.widget.TabView(tableViewId);

**创建的方式：**

1． 页面中存在id

<div id="demo" class="yui-navset">

<ul class="yui-nav">

<li class="selected"><a href="#tab1"><em>Tab One Label</em></a></li>

<li><a href="#tab2"><em>Tab Two Label</em></a></li>

<li><a href="#tab3"><em>Tab Three Label</em></a></li>

</ul>

<div class="yui-content">

<div><p>Tab One Content</p></div>

<div><p>Tab Two Content</p></div>

<div><p>Tab Three Content</p></div>

</div>

</div>

注：

yui-navset为css文件中定义的TableView的样式

yui-nav为css文件中定义的Tab页标签的样式

2． 完全使用JavaScript创建TabView

<script type="text/javascript">

var myTabs = new YAHOO.widget.TabView("demo");

myTabs.addTab( new YAHOO.widget.Tab({

label: 'Tab One Label',

content: '<p>Tab One Content</p>',

active: true

}));

myTabs.addTab( new YAHOO.widget.Tab({

label: 'Tab Two Label',

content: '<p>Tab Two Content</p>'

}));

myTabs.appendTo(document.body);

</script>

**TableView的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| activeIndex | 当前激活的Tab页面编号，第一页为0 |
| activeTab | YAHOO.widget.Tab，指向被激活的Tab页， |
| element | 被绑定的HTML元素 |
| orientation | 页面标签所在的位置，top，bottom，left，right 默认为top |
| tabs | 标签页(Tab)的数组列表，只读 |

**Tab的配置集**

|  |  |
| --- | --- |
| 属性名称 | 含义 |
| active | 当前Tab页面是否激活，Boolean值 |
| cacheData | Tab页加载数据的时候是否隐藏，只有dataSrc可用的时候有效 |
| content | 激活页面的文字标签 |
| contentEl |  |
| dataLoaded | 数据是否加载完成 |
| dataSrc | 如果设置了这一属性，当Tab页被激活时，页面数据从这个URL处加载，需要用到Connection Manager |
| dataTimeout | Tab页加载的超时时间，单位微秒，只有在设置了dataSrc才有效 |
| disabled | 是否禁止激活本Tab页，缺省为false |
| label | Tab页的标签 |
| labelEl | Tab页的HTML容器元素 |
| loadMethod | 数据请求的方式，缺省为GET，只有在设置了dataSrc才有效 |

**TabView用到的CSS定义**

|  |  |
| --- | --- |
| Class | 描述 |
| yui-navset | TabView的<div> |
| yui-nav | TabView的<ul> |
| yui-content | TabView的<div>的下一级<div> |
| disabled | Tab页面不可用时的样式，应用在<ul>上 |
| selected | Tab页面激活时的样式，应用在<ul>上 |
| loading | Tab页面加载时的样式，TabView的<div>的下一级<div>的样式 |

**7.            常见问题**

**7.1.     YUI提供的javascript的文件引入**

**关于引入的版本：**

YUI对于需要引入的js文件都提供了多个版本：

1          min是去掉空格后的版本，在正式项目中建议使用min版。

2          debug是调试的版本，主要结合YUI的log组件使用。

3          beta版

4          不带后缀的为便于阅读的版本

yahoo-dom-event.js是包含了yahoo.js，dom.js，event.js的内容，使用了前者后就无需再引入后面三个文件。

**关于引入的顺序：**

1． 大部分组件都是依赖于事件驱动，并通过YUI提供的工具集进行DOM操作的，因此yahoo.js，dom.js，event.js或者这三个文件的合成文件yahoo-dom-event.js必须早于其他YUI的js文件引入

2． Autocomplete如果需要从服务器取数据时，要用到YUI提供的datasource.js，因此datasource.js必须在autocomplete.js之前引入；如果是以XHR（JSON，XML等）的方式取数，还要用到YUI提供的connection，因此connection.js也必须在autocomplete.js之前引入；如果用到了动画方式展开下拉框，则需要保证animation.js在autocomplete.js之前引入。

3． Datatable中列宽度的调整，Container组件中的Overlay以及从Overlay继承下来的Dialog，SimpleDialog的窗口拖动，需要用到dragdrop.js，因此dragdrop.js必须在datatable.js和container.js之前引入；如果要用到Dialog的动画显示效果，则需要先引入animation.js。