

# iDevUI

## Table of contents

---

Introduction .....	4
Features .....	6
Screenshots .....	7
Concept .....	8
Getting Started .....	9
Starting Coding .....	10
Definitions .....	14
App Definition .....	16
Page Definition .....	18
IDev Object .....	21
Page Manager .....	27
Preferences .....	29
Widgets .....	32
Button .....	34
Canvas .....	36
Chart .....	37
Checkbox .....	39
Combobox .....	41
Composite .....	44
DataView .....	44
Form Panel .....	47
Grid .....	49
Icon .....	51
IFrame .....	52
Image .....	54
Input .....	55
Label .....	57
List .....	58
ListBox .....	61
Object .....	63
Panel .....	65
Progress Bar .....	69
Radio Button .....	70
Slider .....	71
SVG Button .....	72
Spacer .....	76
Switch .....	76
Tab Panel .....	77
Text Area .....	79
Window .....	81
Window Form .....	84
Additional Widgets .....	85
Base Widget Methods .....	86
Data Store .....	91
Data Record .....	94
ColumnModel .....	96
Column .....	97
Utility Methods .....	99
Macros .....	103
Built-In Icons .....	105

How To Deploy .....	106
UX .....	107
Pictures .....	107
Date Picker .....	109
Gauge .....	111
Map .....	112
Rich Text .....	114
Signature .....	115
Treeview .....	117
Uploader .....	120
Local Seviles .....	122
Local Database .....	122
GEO Location .....	123
Local Storage .....	124
Advanced Topics .....	126
Using CSS3 .....	126
Page Animation .....	127
Languages .....	127
Special Features .....	128
Themes .....	129
iDev Studio .....	132
iDev Studio Deployment .....	133
Acknowledgement .....	135



# Welcome to iDev UI

(Internet **Dev**elopment **U**ser **I**nterface)

Version 1.1.1 build: 20120307-1

### *What is iDev UI?*

iDev UI is a Javascript framework sat on top of JQuery and designed to develop both mobile and desktop web applications across all popular browsers.



Phone Platforms:

- iPhone, iPad
- Android 3.2
- Blackberry
- Windows Mobile 7
- Palm WebOS



iDev UI was developed for a number of reasons:

1. There needed to be a single framework that could deliver applications for both desktop and all mobile platforms.
2. The majority of the existing mobile web development frameworks only focused on the iPhone and Android.
3. To develop at native code level means having knowledge of several different skills, each platform has its own development model and SDK.
4. The same framework needed to full extendible and device aware.
5. The aim of utilising more widely available web skills to create application.

Like all Javascript frameworks iDevUI is a mixture of programming and CSS. In simple terms the complete web interface is generated by the framework, controlled by the application code.





# Features

iDevUI delivers the following features:

- UI Widgets
- User defined extensions
- Themes
- Page navigation
- Geo-locations
- Event handling
- Animation
- Canvas (using Raphael)
- AJAX communications
- Drag'n'Drop
- Charting
- Dynamic Data

With iDevUI you can create virtually any web application, though some advanced features are subject to the target platform.



# Example Designs





# Concept

The basic concept behind iDevUI is the page, the application is built upon a collection of pages sitting within the main container. Each page, in fact the entire application UI is defined using JSON.



Pages are defined by your code but rendered as they are needed; they then act like a deck of cards with one of them at the front at all times.

The iDevUI is based on jQuery and a base style sheet. The creation of a theme style sheet allows the base styling to be overwritten and present the style you want.

All UI widgets are template driven and a base widget class.

The base widget class acts as a building block for all widgets and allows them to have a standard set of methods like show or hide.

Note: The navigation bar, status bar, header and footer are optional.





# Getting Started

## Basic Coding:

To start coding all you need is your favourite text/code editor.

- Download the latest version
- Unzip to a development folder where you want to build your application
- In the "js" folder you will see a file called "app.js" this is the main code file. Open it in your editor.
- To preview your open the index.html file in your browser.

## Where to start

The first place to start is to understand the structure of the application and then look at the available UI widgets.

From there it depends on the application you are intending to create.

You may want to download one of the examples on the website to see how they are done.



# Starting Coding

Before starting development you need to understand some fundamentals...

- The application starts from index.html

```
<!DOCTYPE html>
<html>
<head>
<meta name="language" content="English" />
<meta name="copyright" content="Trakware Solutions Ltd" />
<meta name="robots" content="index,follow" />
<meta name="revisit-after" content="7 Days" />
<meta http-equiv="Content-Language" content="en-gb" />
<meta http-equiv="imagetoolbar" content="false" />
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE">
<META HTTP-EQUIV="EXPIRES" CONTENT="-1">
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta name="apple-mobile-web-app-status-bar-style"
content="default" />
<meta name="viewport" content="width=480;height=640;">
  <title>Mobile App</title>
  <link rel="apple-touch-icon" href="../images/appicon.png"
/>
  <script type="text/javascript" src="../js/
preferences.js"></script>
  <script type="text/javascript" src="../js/idevui/
idevui.js"></script>
</head>
<body noscroll style='margin:0;'>
<div id='mask'>
</div>
<div id='container'>
<div style='position:absolute;left:46%;top:48%;'>
Loading please wait...
</div>
</div>
</body>
</html>
```

You should not need to edit all the above html, except to change the loading statement or mobile settings.

- The idevui.js already knows the dependencies and the app.js which is the main code file, it loads them automatically for you. Therefore you do not need to edit index.html at all other than to perhaps change the loading text.
- The other main file is preferences.js which control the UI framework and specifies where to find things (see section of preferences).

```

var _preferences = {
  title: 'Mobile App',
  blankimage: "../js/idevui/images/s.gif",
  libpath: "../js/idevui/",
  language: 'english',
  languagepath: '../js/',
  imagepath: '../images/',
  apppath: '../js/',
  app: 'app.js',
  theme: '../css/theme.css',
  styling: '../css/style.css',
  appicon: '../images/appicon.png',
  // Default button style.
  button: {
    startcolor: '#bbb',
    endcolor: '#fefefe:70-#ddd',
    fontcolor: '#000',
    fontsize: 14,
    fontweight: 'normal',
    iconcolor: '#000',
    radius: 6
  },
  // Framework user eXtensions (loaded from libpath).
  ux: [
    "gauge/gauge.js",
    "uploader/uploader.js",
    "richtext/richtext.js",
    "datepicker/datepicker.js",
    "signature/signature.js"
  ],
  // General application import files (loaded from apppath).
  imports: [
  ],
  config: {
    charts: true,
    mapping: true,
    norightclick: true,
    fitDocument: true,
    pageFit: true
  }
};

```

- the ux folder contains any addition UI plugins, which are extensions on the existing widget classes.

Folder Structure:

```

myApp
|___ css
|___ |___ styles.css
|___ images
|___ js
|___ |___ idevui
|___ |___ |___ themes
|___ |___ |___ ux
|___ |___ app.js
|___ |___ preferences.js
|

```

**Coding**

Now to coding you can open the app.js in your code editor.

To start you off we have created the app.js with the declaration of two page var's and a basic app structure which is the main object used by the framework.

When the web page is loaded it calls a idev.onReady function that start your application.

```
var page1...

var page2...

idev.app = {
  title:'Mobile App',
  toolbar: {
    widgets: [
      {
        wtype:'button',
        id:'idHome',
        width:90,
        text:'Home',
        iconAlign:'left',
        color:'blue',
        icon:'home',
        events : {
          click: function(page,btn)
          {
            idev.homePage();
          }
        }
      },
      {
        wtype:'label',
        width:180,
        text:'Mobile App',
        style:'color:#fff;padding-top:4;text-align:center;'
      },
      {
        wtype:'buttonrwd',
        id:'idBack',
        width:90,
        color:'blue',
        text:'Back',
        events : {
          click: function(page,btn)
          {
            idev.pageManager.prevPage();
          }
        }
      },
      {
        wtype:'spacer',
        width:2
      },
    ]
  }
}
```

```

        {
            wtype:'buttonfwd',
            id:'idNext',
            width:90,
            color:'blue',
            text:'Next',
            events : {
                click: function(page,btn)
                {
                    idev.pageManager.nextPage();
                }
            }
        }
    ],
    },
    pages : [ // Add pages here
        page1,
        page2
    ]
};
idev.onReady(function()
{
    idev.pageManager.showPage(0);
});

```

You will also notice in the above code, the object idev, this is the main framework object and the one you will use to access to features and objects created by the framework.

You should note at this stage that the framework uses the idev.app JSON config to define the complete application architecture, the pages and the widgets to be rendered on those pages.

To view your application simply load the index.html with you favourite browser.



# Definitions

### ***The App***

The app configuration is a JSON object with the following definition.

```
idev.app = {
  title: 'Mobile App',
  toolbar: { <----- The navigation bar at the
top.
    widgets: [
      <----- The UI Widgets go here.
    ]
  },
  pages : [
    <----- The pages that define your
app go here.
  ],
  statusbar: { <----- Optional status bar at the
bottom.
    widgets: [
      <----- The UI Widgets go here.
    ]
  }
};
```

### ***A Page***

A page is again configured as a JSON object with the following definition

```
var page = {
  name: 'Main Page',
  labelWidth: 120, <----- Main paging
options
  header: {
    height: 100,
    style: '', <----- Header options
    widgets: [
      <----- Widgets for the
header go here
    ]
  },
  content: {
    height: 448, <----- Content options
```

```

padding:5,
widgets: [
    <----- Main page Widgets
go here
    ],
    events : {
        <----- Container events
    },
    footer: {
        height:50,
        style:'', <----- Footer options
        widgets: [
            <----- Widgets for the
footer go here
        ]
    },
    events: {
        <----- Page events
    }
};

```

## ***The Widgets***

A UI Widget definition varies depending on the type of widget but in general with the following definition

```

{
    wtype:'image', <----- Widget type
    x:350,
    y:5, <----- Widget option
    width:96,
    height:96,
    src:'../images/earth.png',
    events: {
        click: function(img)
        {
            <----- Declaration of
widget events
        }
    }
}

```

Options not set will have their defaults and some like the style will be controlled by the theme.



# App Definition

The app definition is the starting point for defining your application and is part of the idev object.

An app is made up of four parts; title, pages, statusbar and events.

```
idev.app = {
  title: 'Mobile App',
  toolbar: {
    widgets: [
    ]
  },
  pages : [

    // Add pages here

  ],
  statusbar: {
    height:50,
    widgets:[
    ]
  },
  events: {
    afterRender : function()
    {
    },
    onOrientation : function(sOrientation)
    {
      alert(sOrientation)
    }
  }
};
```

### options

<i>title</i>	Application title and text to be display as page title
<i>width</i>	Sets the width of the main container DIV (optional)
<i>height</i>	Sets the height of the main container DIV (optional)
<i>toolbar</i>	A collection of widgets are permanently at the top of the app
<i>pages</i>	A collection of page definitions
<i>statusbar</i>	A collection of widgets are permanently at the bottom of the app



<i>events</i>	A set of event handlers
---------------	-------------------------

## ***toolbar***

<i>height</i>	Height of toolbar (default 40)
<i>widgets</i>	Collection of widgets to be rendered to the toolbar
<i>html</i>	Simple text or html to be show in the toolbar

## ***pages***

See Page Definition section of this document

## ***statusbar***

<i>height</i>	Height of statusbar (default 40)
<i>widgets</i>	Collection of widgets to be rendered to the statusbar
<i>html</i>	Simple text or html to be show in the statusbar

## ***events***

<i>afterRender</i>	Fired after the app has been created.  Parameters: none
<i>beforeRender</i>	Fired before the first page is rendered
<i>onOrientation</i>	Fired when the device screen changes orientation.  Note: Only on mobile devices
<i>onResize</i>	Fired if the main page is resized.  Parameters: event object



# Page Definition

The page is the main definition upon which all the widgets sit.

A page is made up of three parts; header, content and footer. But it also has its own configuration as well.

```
var page = {
  name: 'Main Page',
  labelWidth: 120, <----- Main paging config
  header: {

  },
  content: {

  },
  footer: {

  },
  events: {
    <----- Page events
  }
};
```

### options

<i>name</i>	User defined name that can be used when showing different pages by name.
<i>style</i>	CSS styling for page.
<i>cls</i>	CSS class to override the core page styling
<i>fn</i>	<p>A user defined object to extend the base page object with additional methods.</p> <pre>{   ...   fn: {     myfunc: function()     {       // Code     }   }   ... }</pre>

		This makes this function available from anywhere and any widget
<b>header</b>	<i>height</i>	Height of header (default 100).
	<i>style</i>	CSS styling for header.
	<i>cls</i>	CSS class to override the core header styling
	<i>hidden</i>	true/ false to hide or show header (default is false)
	<i>widgets</i>	A collection of widget configurations or objects.
	<i>html</i>	Simple text or html to be show in the header
<b>content</b>	<i>height</i>	Height of header (default 100).
	<i>style</i>	CSS styling for content.
	<i>css</i>	CSS class to override the core content styling
	<i>layout</i>	content widget layout, option for "form" which will use the label property of the widgets to render a form layout. (default none)
	<i>labelWidth</i>	Layout of a page, this property defines the width allocation to any widget labels when the content layout is "form".
	<i>widgets</i>	A collection of widget configurations or objects.
<b>footer</b>	<i>html</i>	Simple text or html to be show in the header
	<i>height</i>	Height of header (default 100).
	<i>style</i>	CSS styling for footer.
	<i>cls</i>	CSS class to override the core footer styling
	<i>hidden</i>	true/ false to hide or show header (default is false)
	<i>widgets</i>	A collection of widget configurations or objects.
	<i>html</i>	Simple text or html to be show in the header

## **events**

*afterRender*

Fired after the page has been created.

Parameters: page object

*beforeHide*

Fired before the page is hidden in favour of another

*show*

Fired after the page is shown.

## **methods**

*getHeaderWidget*

Return a header widget object or null based on the index (zero start).

Parameters: index number

*getContentWidget*

Return a content widget object or null based on the index (zero start).

Parameters: index number

*getFooterWidget*

Return a footer widget object or null based on the index (zero start).

Parameters: index number



# iDev Object

The main object in the framework is the idev object. It is this that allows you to create and control the application.

```
idev.app = {
  title: 'Mobile App',
  toolbar: {
    widgets: [
    ]
  },
  pages : [

    // Add pages here

  ],
  statusbar: {
    height: 50,
    widgets: [
    ]
  },
  events: {
    afterRender : function()
    {
    },
    onOrientation : function(sOrientation)
    {
      alert(sOrientation)
    }
  }
};
idev.onReady(function()
{
  idev.pageManager.showPage(0);
});
```

### ***app object***

This is the main object that describes your application.

*Config:*

<i>title</i>	User defined application title which will set the web page document title.
<i>toolbar</i>	Toolbar object. A collection of widgets that sits above every page.
<i>statusbar</i>	Statusbar object. A collection of widgets that sits below every page.
<i>pages</i>	The collection of pages that make up your applications

<i>events</i>	A set of event handlers.  afterRender: Call once the main application has been created.  onOrientation: Call everytime the framework detect a changing in orientation.
---------------	--

## **events**

OnReady

Called once the object has been created and all the dependencies have been loaded.

There is also a ***idev.events*** object that can be used to catch other internal events.

<i>onstartdrag</i>	Called once a widget starts to be dragged. This independent of the object events.  Parameters: widget, event
<i>ondrag</i>	called while the widget is being dragged.  Parameters: widget, event
<i>onenddrag</i>	Called at the end of widget drag.  Parameters: widget, event
<i>onmousemove</i>	Call as the mouse moves around the screen.  Parameters: event
<i>onmousedown</i>	Called on a mousedown anywhere on the app.
<i>onmouseup</i>	Called on a mouseup on the app.

Please note dragging a widget around is not the same as drag'n'drop.

## **properties**

<i>pgWidth</i>	Width of application document.
<i>pgHeight</i>	Height of application document.
<i>body</i>	Main document body object.
<i>pageManager</i>	Access the the iDev pageManager object
<i>agent</i>	Browser agent text
<i>url</i>	Web page URL less any parameters
<i>rawUrl</i>	Web page URL with and parameters
<i>parameters</i>	An array of the passed parameters
<i>fadeOutTime</i>	Current page fade out time in milliseconds
<i>fadeInTime</i>	Current page fade in time in milliseconds
<i>swipping</i>	Sets whether the user can use swipping left to right when changing pages on

	a touch screen device.  default false;
<i>geoposition</i>	Last GEO lat/long after the idev.local.currentLocation method was called.
<i>geoerror</i>	Last GEO error after the idev.local.currentLocation method was called and an error occurred. If the error exists geoposition will be null.
<i>mousedowntime</i>	A date object set when a mousedown event occurs
<i>mouseuptime</i>	A date object set when a mouseupevent occurs
<i>mousemoved</i>	A flag set to true or false to indicate the mouse moved between a mousedown and a mouseup.
<i>fn</i>	A user defined object to extend the base idev object with additional methods.  To add a function just...  idev.fn.myFunc = function() {  };  This makes this function available from anywhere and any widget
<i>dom</i>	This is a direct alternative for the JQuery object. Anything you can so with JQuery you can so via the <b>idev.dom</b> object.  This does not preclude you from using the jquery \$ object but you may wish to use this so that all your code is based around one core object.
<i>animationTime</i>	Time in milliseconds for page fading or sliding (default 500).
<i>focusHighLight</i>	This places a shadow around the input that has focus (default false).
<i>highlightCSS</i>	This is the CSS to add to the input in order to show the shadow.  Default: "0px 0px 8px #336699"  xoffset, yoffset, size, color

Note: The mousedowntime, mouseuptime and mousemove are also linked to a touch start and end.

## methods

<i>get</i>	Gets a widget by id regardless of the page it is on  Parameters: id string  Return: widget object
<i>remove</i>	Removes a widget by id regardless of the page it is on  Parameters: id string  Return: none
<i>homePage</i>	Show the first page in the collection

	Parameters: none Return: none
<i>gotoPage</i>	Show a page by index or name Parameters: index string ("0") is first page Return: none
<i>nextPage</i>	Show the next page in the collection Parameters: none Return: none
<i>prevPage</i>	Show the previous page in the collection Parameters: none Return: none
<i>currentPage</i>	Returns the current page object Parameters: none Return: none
<i>lastPage</i>	Returns the name of the last page shown Parameters: none Return: none
<i>currentPageName</i>	Returns the current page name Parameters: none Return: none
<i>removePageWidgets</i>	Removes a page widget by id Parameters: id Return: none
<i>fadeIn</i>	Fade in a widget Parameters: id, speed, easing, callback Return: none
<i>fadeOut</i>	Fade out a widget Parameters: id, speed, easing, callback Return: none
<i>hide</i>	Hide a widget



	Parameters: id Return: none
<i>show</i>	Show a widget Parameters: id Return: none
<i>getParameter</i>	Get a parameter passed as part of the url. http://www.mysite.com/myapp/index.html?name=fred Parameters: parameter,default Where: parameter can a number or the name of a parameter Returns parameter value or default
<i>getParameterKey</i>	Get a parameter key name by index number Parameters:index,default Returns parameter name or default
<i>isStandAlone</i>	Returns true or false if the application is being run as a web app on a mobile device.
<i>convertNulls</i>	Used to test an object is null and return either the object or a default if its null. Parameters:object,default Returns object or default
<i>isClass</i>	Test id the object passed is based on the baseWidget class. Parameters:object Returns true or false
<i>hideStatusbar</i>	Hide the main application statusbar
<i>showStatusbar</i>	Show the main application statusbar
<i>addTouchScroll</i>	Adds touch scroll capability to a DIV. Parameters:id,options id = the id of the DIV options = a JSON config <pre>{   vScroll:true,   hScroll:true,   bounce:true,</pre>

	<pre>momentum: true }</pre> <p>Returns new scroll class object</p>
<i>createWorker</i>	<p>Creates a HTML5 worker thread based on script file you pass</p> <p>Parameters:sScript,onmessage</p> <p>sScript = name of script to be found in _preferences.apppath onmessage = a function to be called when the thread posts a message back to the main app.</p> <p>see HTML5 worker threads in Internet.</p> <p>Returns worker object.</p>
<i>addWidgetToPageContent</i>	<p>Adds a widget config to a page's content widget collection.</p> <p>Parameters:pageID,widget</p> <p>Returns true or false</p> <p><b>Note: This does not render the widget</b></p>
<i>bodyHeight</i>	Returns the height in pixels of the main body of the application, less the toolbar and statusbar height.
<i>bodyWidth</i>	Returns the application width. (same as idev.pgWidth)
<i>isTouch</i>	Returns true or false if the device is a touch device.
<i>isIPhone</i>	Returns true or false if the device is a iPhone.
<i>isIPad</i>	Returns true or false if the device is a iPad.
<i>isAndroid</i>	Returns true or false if the device is running Android.
<i>isBlackberry</i>	Returns true or false if the device is a Blackberry.
<i>isPalm</i>	Returns true or false if the device is a Palm.
<i>isIEMobile</i>	Returns true or false if the device is a IE Mobile.
<i>isFF</i>	Returns true or false if the device is a Firefox.
<i>isWebkit</i>	Returns true or false if the device is running a webkit render engine.
<i>isIE</i>	Returns true or false if the device is running Internet Explorer.
<i>isMobile</i>	Returns true or false if the device is a mobile



# Page Manager

The idev object has a page manager object, it is this that controls the application pages.

A page is made up of three parts; header, content and footer, but it also has it's own configuration as well. Most of the common page changes can be accessed direct from the idev object there are some additional ones that are available from the pageManager object itself.

`idev.pageManager`

NOTE: Pages are created only the first time they are shown, up until that point only the page definition is available in the `idev.app.pages` array.

### methods

<i>add</i>	<p>Adds a new page definition to the application. This does not create the page just adds it to the page collection</p> <p>Parameters: JSON config</p> <p>Return: true</p>
<i>getPage</i>	<p>Returns a page object by name assuming it has been created.</p> <p>Parameters: name</p> <p>Return: page object or null</p>
<i>pageCount</i>	<p>Returns the number of created pages</p> <p>Parameters: count</p>
<i>remove</i>	<p>Removes a page from the page definitions collection and if create destroys the page and all created widgets.</p> <p>This should be followed by a <code>showPage</code> if page being removed is the current page.</p> <p>Parameters: name</p> <p>Return: true or false</p>
<i>showPage</i>	<p>Shows a page by name. If the page is not yet created to will render all the widgets. This will also perform any animation as the current page and new page change (see advanced topics).</p> <p>Parameters: name or index</p>

## Properties

	Return: true or false
<i>prevPage</i>	Shows the previous page in the collection
<i>nextPage</i>	Show the next page in the collection
<i>animationType</i>	Type of animation between page changes (default "fade") Options: "fade" or "slide"
<i>animation</i>	Determines if page changes are animated (default true)
<i>currentPage</i>	Current page object
<i>lastPage</i>	Last page object



# Preferences

To control the application iDevUI implements a preferences JSON object (found in the preferences.js file). This helps the framework to know where to find things and control some features.

```
var _preferences = {
  title: 'Mobile App',
  blankimage: "../js/idevui/images/s.gif",
  rootpath: '../',
  libpath: "../js/idevui/",
  language: 'french',
  languagepath: '../js/',
  imagepath: '../images/',
  apppath: '../js/',
  app: 'app.js',
  theme: 'default',
  startPage: 0,
  useCSS3: false,
  styling: '../css/style.css',
  // Default button style.
  button: {
    startcolor: '#bbb',
    endcolor: '#fefefe:70-#ddd',
    fontcolor: '#000',
    fontsize: 14,
    fontweight: 'normal',
    iconcolor: '#000',
    radius: 6
  },
  // Framework user eXtensions (loaded from libpath).
  ux: [
    "gauge/gauge.js",
    "uploader/uploader.js",
    "richtext/richtext.js",
    "datepicker/datepicker.js",
    "signature/signature.js",
    "mapping/mapping.js",
    "pictures/pictures.js",
    "treeview/treeview.js"
  ],
  // General application import files (loaded from apppath).
  imports: [

    // Add addition source code files here

  ],
  config: {
    charts: true,
  }
}
```

```

        norightclick:true,
        fitDocument:true,
        pageFit:true
    },
    events: {
    }
};

```

Every app MUST have a preferences object. Throughout the application the **\_preferences** object can be used to determine locations and settings

## properties

<i>title</i>	Title to be applied to application document.
<i>blankimage</i>	Some widget use this to create a space or instead of a graphic
<i>rootpath</i>	Path to root of all application files
<i>libpath</i>	Path to iDevUI framework
<i>language</i>	Selected language (see languages section) the default is English (optional)
<i>languagepath</i>	If you intend to use a language you can use this property to set the path
<i>imagepath</i>	Path to application images (these are separate to the framework images)
<i>apppath</i>	Path to main application script file
<i>app</i>	By default this is "app.js" but you can overwrite this. To do that add the beforeLoad event and change it before all is loaded.
<i>theme</i>	CSS style theme to use (default is "default")
<i>useCSS3</i>	Some browsers do not support SVG for the buttons this allow the framework to use CSS3 styling instead, though you will loose some features
<i>styling</i>	Path to application specific CSS stylesheet
<i>button</i>	A JSON object that set the default styling for all SVG buttons
<i>ux</i>	List of <b>U</b> ser <b>eX</b> tensions that get loaded automatically by the framework
<i>imports</i>	List of any additional script file the application may need
<i>forcereload</i>	Forces a full reload when reloading page (including cache)
<i>config</i>	<p>Some framework control settings</p> <p><b>charts:</b> set to true if you intend to use charts (if false the scripts are not loaded)</p> <p><b>noRightClick:</b> Disables the right click of the browser</p> <p><b>fitDocument:</b> Forces the framework to fit to the max screen size.</p> <p><b>pageFit:</b> Forces the pages to fit the screen less any application toolbar or statusbar. If the page has a header and footer the content section is calculated to the page height less these values.</p>

## events

	<p><b><i>trapUnload:</i></b> By default the framework will trap for any navigation away from the app and gives the user the opportunity to cancel. If this config property is set to false then no check is performed.</p> <p><b>unloadText:</b> Specifies the text to be shown in the trapUnload alert box. If not specified the default text will be shown on the alert.</p>
<i>beforeLoad</i>	Fired before the dependencies are loaded



# Widgets

Widgets are the **User Interface** items you use in the layout of your page. They range from simple text to the integration of a map.

All widgets are defined in JSON within the *widgets* section of the page, the framework will then take this list and render them for you.

### ***The Widgets***

Panel	An area of the page for the display of text or HTML.
Button	A trigger for the execution of script.
Input	Single line user input box
TextArea	Multi-line user input area
CheckBox	Standalone checkbox control widget
Combo Box	A standard combo box widget driven from a data store
Switch	A iPhone like, on/off switch
Radio	Radio button for option selection
Label	An input field label
Image	An image display control
Icon	A widget to display one of a built-in range of mobile icons
Spacer	A widget to create space when the page layout is a form
Element	A free format widget that can be used to render and user defined UI element
Canvas	A widget that supports SVG using Raphael.js
Adverts	A widget for fading in and out a series of images
ListBox	A mobile list box control like the iPhone
List	A HTML template driven list component
Data View	A HTML template driven view using a data store



Chart	A barchart, line or piechart widget
Slider	A slider widget
Progress Bar	A progress widget
IFrame	A widget to show a sub web page
Grid	A grid widget linked to a dataStore
Tab Panel	A tab widget for the display of multiple panels
Sound	A widget to play MP3 and WAV files
Composite	A widget to combine widget of different types into one layout

All widgets support properties, events and functions. Some require the definition of sub widgets like the list box and a knowledge of HTML.

***idev.ui object*** All widget classes are found in the idev.ui object.

ie.

```
var panel = new idev.ui.widgetPanel({ ..... });
```

***Method:***

<i>addEvent</i>	Added an event listener to a widget Parameter: id,eventName,function
<i>removeEvent</i>	Removes an event listener from a widget Parameter: id,eventName,function

ie.

```
function onMouseMove(e)
{
    ...
}
```

```
idev.ui.addEvent("myImage","mousemove",onMouseMove);
```

then

```
idev.ui.removeEvent("myImage","mousemove",onMouseMove);
```

Note: To remove an event the function must be the same one used when the event was added, so you need declare it separately.

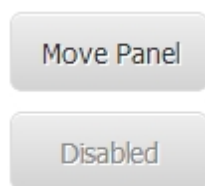
## Button



# Button

A button widget.

```
{
  wtype: 'button',
  x: 220,
  y: 5,
  width: 220,
  height: 280,
  color: 'blue',
  icon: 'arrow',
  text: "Click Me",
  events: {
  }
}
```



The buttons on iDevUI were created using SVG until version 1.1, when two separate widgets were created, one using SVG (keep backward compatability) and one using CSS.

Note: the default CSS for a button comes from the theme CSS.

### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of button.
<i>height</i>	Height of button.
<i>text</i>	Button text.
<i>textCls</i>	CSS class to be applied to button text
<i>transparent</i>	Button with no fill color but can have a border.
<i>icon</i>	icon to display on button (JSON object).

	<pre>icon: {   src:'path to image or standard icon name',   x:10,   y:10,   width:32,   height:32,   style:",   color:'white',   align:'right' }</pre> <p>Where</p> <p>style: CSS styling to be applied icon</p> <p>align: When no X or Y position supplied align the icon to the left (default) or right of text on button.</p> <p>color: When the name of a standard icon is supplied there are two possible colors, black (default) or white.</p>
label	Text used when rendered to a form layout.
enabled	This controls whether the button click events will function (default true)
title	Text shown when mouse hovers over button
toggle	Places the button into toggle mode (default false). Where the button can be shown as depressed.
down	Initial state of button in toggle mode (true/false)
cls	CSS class to be applied to main button body
toggleCls	CSS class to be applied when button is toggled (down)
disabledCls	CSS class to be applied when button is disabled
tooltip	Adds tool tip that displays on hover

## ***methods***

setText	<p>Set the button text.</p> <p>Parameters: text</p> <p>Return: none</p>
getText	<p>Get the current button text.</p> <p>Return: button text</p>
isDown	<p>Used to determine if button is down when in toggle mode.</p> <p>Parameters: none</p> <p>Return: true/false</p>

<code>enableButton</code>	Enables or disables button to react to click events  Parameters <code>enable</code> (true/false)
<code>toggleButton</code>	Shows button in toggle mode up or down.  {Parameters: <code>down</code> (true/false)  Return: none
<code>isEnabled</code>	Returns true/false if button is enabled.
<code>isDown</code>	Returns true/false if button is toggled down.

## events

<code>beforeRender</code>	Fired before the widget has been created.  Parameters: button widget
<code>afterRender</code>	Fired after the widget has been created.  Parameters: button widget
<code>click</code>	Fired when button clicked.  Parameters: button widget
<code>canToggle</code>	Fired before a button is toggled.  Parameters: button widget  Return true/false
<code>onToggle</code>	Fired after a button is toggled.  Parameters: button widget

---

Created with the Personal Edition of HelpNDoc: [Produce electronic books easily](#)

---

## Canvas



# Canvas

A canvas widget. Creates a SVG canvas using the Raphael JS framework ([www.dmitrybaranovskiy.github.io/raphael/](http://www.dmitrybaranovskiy.github.io/raphael/))

```
{
  wtype: 'canvas',
  x: 220,
  y: 5,
  width: 220,
```

```

        height:280,
        style:''
        events: {

        }
    }

```

## properties

<i>paper</i>	The Raphael paper object. With this you can draw on the canvas using the features found at <a href="http://www.dmitrybaranovskiy.github.io/raphael/">www.dmitrybaranovskiy.github.io/raphael/</a>
--------------	---

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>style</i>	CSS styling to be applied to canvas DIV container.
<i>label</i>	Text used when rendered to a form layout.

## events

<i>afterRender</i>	<p>Fired after the canvas has been rendered.</p> <p>Parameters: canvas widget</p> <p>Widget includes a paper object which can used the render onto the canvas.</p>
--------------------	--

---

Created with the Personal Edition of HelpNDoc: [Easily create iPhone documentation](http://www.helpndoc.com/)

---

## Chart



# Chart

A chart widget.

```

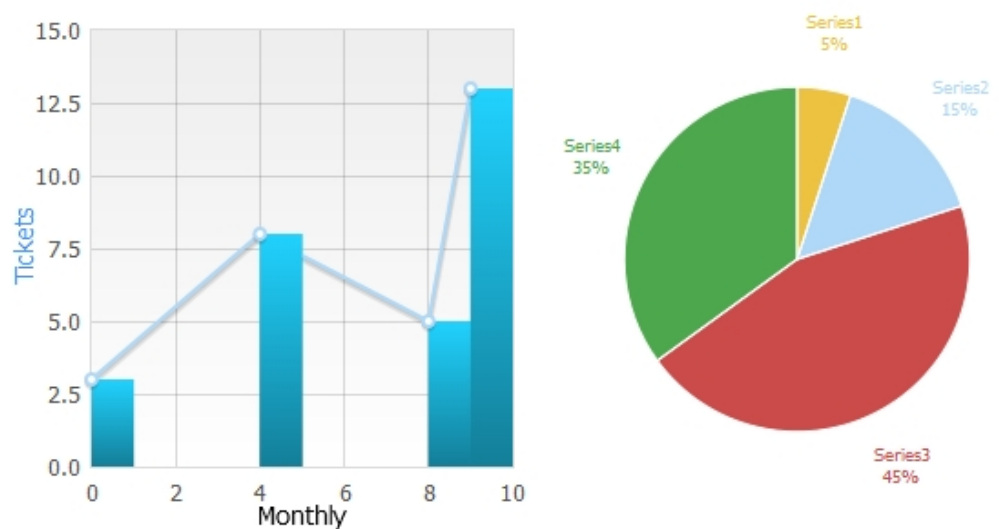
{
    wtype:'chart',
    x:220,
    y:5,
    width:220,

```

```

height:280,
series: [
  { data:[ [0, 3], [4, 8], [8, 5], [9, 13]] },
  { data:[ [0, 3], [4, 8], [8, 5], [9, 13]], points: { show: true }
},
options: {
  bars: {
    show: true,
    lineWidth:0,
    color:'#ddd',
    fill:true,
    fillColor: { colors: [ '#21D3FF', '#147F99' ] },
    colors: [{ opacity: 0.1 }]
  },
  grid: {
    backgroundColor: { colors: ["#eee", "#fff"] },
    borderColor:'#ddd',
    borderWidth:1
  },
  xaxis: {
    axisLabel: 'Monthly',
    axisLabelUseCanvas: true
  },
  yaxis: {
    axisLabel: 'Tickets',
    labelColor:'#4294FF',
    axisLabelUseCanvas: true
  },
  shadowSize:3
},
style:'background:#fff;',
events: {
}
}

```



## options

id	User defined id for widget, if not supplied one will be assigned.
----	---

<i>x</i>	Left position on page.						
<i>y</i>	Top position on page.						
<i>width</i>	Width of chart.						
<i>height</i>	Height of chart.						
<i>label</i>	Text used when rendered to a form layout.						
<i>series</i>	<p>An array data series. Each element in the array for a different data set.</p> <p>A data set has the following properties</p> <table> <tr> <td><i>data</i></td><td>A set on value pairs</td></tr> <tr> <td><i>label</i></td><td>A text label associate with the data set</td></tr> <tr> <td><i>lines</i></td><td>Parameters about the line if you want to see the data drawn as a line</td></tr> </table>	<i>data</i>	A set on value pairs	<i>label</i>	A text label associate with the data set	<i>lines</i>	Parameters about the line if you want to see the data drawn as a line
<i>data</i>	A set on value pairs						
<i>label</i>	A text label associate with the data set						
<i>lines</i>	Parameters about the line if you want to see the data drawn as a line						
<i>options</i>	A collection of properties for the chart						

With regards to the charting details we suggest you visit <http://code.google.com/p/flot/>

## methods

<i>setSeries</i>	<p>Updates chart with specified data series</p> <p>Parameters: series</p>
------------------	---

## events

<i>click</i>	<p>Fired when chart clicked.</p> <p>Parameters: chart widget</p>
<i>dblclick</i>	<p>Fired when chart double clicked.</p> <p>Parameters: chart widget</p>

---

Created with the Personal Edition of HelpNDoc: [Easily create HTML Help documents](#)

---

## Checkbox



# Checkbox

A checkbox widget.

```
{
    wtype: 'checkbox' ,
    x: 220 ,
```

```

    y:5,
    value: 'A',
    checked:true
}

```



## options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
y	Top position on page.
value	Checkbox value.
nvalue	Checkbox value when not checked
checked	Initial checked state (default false).
text	Text to be displayed next to the checkbox
textStyle	CSS style of text
align	Text align "left" or "right" default("left")
label	Text used when rendered to a form layout
lock	Prevents users from modifying Checkbox when set to true (default false).

## methods

check	<p>Changes the checked state of a checkbox.</p> <p>Parameters: checked (true/false)</p> <p>Return: none</p>
getValue	<p>Gets the value of the checkbox.</p> <p>Parameters: none</p> <p>Return: value</p>
setValue	<p>Sets the checkbox check depending if passed value matches value property</p> <p>Parameters: value</p>
isChecked	<p>Determines the checkbox is checked</p> <p>Parameters: none</p> <p>Return: true / false</p>

## events

afterRender	<p>Fired after the checkbox has been rendered.</p> <p>Parameters: checkbox widget</p>
change	Fired when then checkbox changes



## Combobox



# ComboBox

ComboBox widget

```
{  
    wtype: 'combo',  
    x: 220,  
    y: 5,  
    width: 110,  
    height: 30,  
    ds: myDataStore,  
    displayField: 'fdName',  
    valueField: 'fdUserID'  
}
```

## options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
width	Width of combobox.
height	Height of combobox
ds	Data store object (see Data Store section)
displayField	Field from DataStore to be displayed in dropdown list.
valueField	From form DataStore to be returned as the widget value.
inputType	This control the input part for typing (default 'text') can also be 'number' if the select is numeric.
selectColor	The colour the selection will be set to (default #eee).
label	Text used when rendered to a form layout
editable	Controls if the input part of the widget can be edited (default true)
inputDisabledCls	Specifies css class to be used for input filed if enable is set to false
buttonDisabledCls	Specifies css class to be used for the combo button if enable is set to false
listEntryHeight	Height of each entry in the dropdown list (optional)
visibleEntrys	Number of entries that the dropdown list display at any one time.
watermark	Text that is displayed in input until something is typed in. Useful for prompts
watermarkColor	Colour of watermark text (default #aaa).
dropWidth	Width of dropdown list (default width of widget)
listTpl	Alternative dropdown list template (optional) can be used to display multiple columns in the dropdown.
autoSelect	Forces the dropdown list to be select based on the combo value.

## events

click	Fired when the input is clicked  Parameters: combo  Return: none
dblclick	Fired when the input is double clicked  Parameters: combo  Return: none
lostfocus	Fired when the input has lost focus  Parameters: combo

## ***methods***

	Return: none
focus	Fired when the input get focus  Parameters: combo  Return: none
change	Fired when the input value changes  Parameters: combo  Return: none
keypress	Fired when a key is pressed  Parameters: combo, event  Return: none
selected	Fired when a entry in the list is selected   Parameters: combo  Return: none
setValue	Set the value of the widget.  Parameters: text  Return: none
setInputValue	Set the value of the input part of the widget.  Parameters: text  Return: none
getValue	Gets the value of the widget.  Parameters: none  Return: value
getInputValue	Gets the value of the input part of the widget.  Parameters: none  Return: value
getSelected	Returns the row number of the selected list item
getStore	Returns the datastore used

## Composite



# Composite

A composite widget is not a true widget more like a container.

```
{
    wtype: 'composite',
    x: 220,
    y: 5,
    layout: 'column',
    widgets: [
    ]
}
```

### options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
width	Width of composite.
height	Height of composite
layout	How the widgets should laid out (see <a href="#">panel</a> )
label	Text used when rendered to a form layout
cls	CSS class for listbox

---

Created with the Personal Edition of HelpNDoc: [Free iPhone documentation generator](#)

---

## DataView



# DataView

A dynamic data grid based on a template.

```

{
    wtype: 'dataview',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    tpl: "<div class='ui-listentry'>{field1}</div>",
    style: '',
    ds: myDataStore,
    entryWidth: 128,
    border: true,
    borderColor: '#999',
    autoScroll: true,
    events: {

    }
}

```

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page
<i>y</i>	Top position on page
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>tpl</i>	HTML template to be rendered for each record in the data store. Use {} brackets around the name of the field for inserting content into the template based on the record being rendered.
<i>data</i>	This an array of value that can used with the tpl. Normally the template would contain the fields names from the dataStore "{field1}". This allows you pass additional data then use "{0}" marker to indicate which entry in the data array you wish to display.
<i>style</i>	CSS styling for main list container
<i>ds</i>	Data store object (see Data Store section)
<i>dsFilter</i>	Function to call for each record in the DS.  Parameters: rec  Return true if the rec should be shown
<i>columns</i>	The number columns the data view should show before starting the next row.  You can also pass 'auto' and let the widget calculate it
<i>cls</i>	CSS class to override the core widget styling
<i>entryWidth</i>	Width of each cell in the grid, this will determine the number of cells you can get in a row.
<i>entryHeight</i>	Height of each row in the data view

<i>entryCls</i>	CSS class to be applied to each entry in the view
<i>border</i>	true / false as to when to show a border
<i>borderColor</i>	Colour of border if visible (default #ccc).
<i>autoScroll</i>	true / false to allow the list to auto scroll if the number of records in the data store means the entries exceed the height of the list
<i>offset</i>	Start position in data store (default 0) for first record.
<i>limit</i>	Max number of entries in the DS to show at any one time
<i>label</i>	Text used when rendered to a form layout.
<i>enabled</i>	This controls whether the click events will function (default true)
<i>roundCorners</i>	Render the panel with round corners
<i>radius</i>	Radius of round corners (default 4)
<i>selectColor</i>	Selection background colour.
<i>selectCls</i>	Selection CSS styling class. You can use this or selectColor or both
<i>autoSelect</i>	This is set to true if you pass a selectColor or selectCls but if want to keep a track of the selected entry without any change you must set this to true (default false);
<i>renderer</i>	<p>Function to be called for each entry.</p> <p>Parameters widget, sText, record, recno</p> <p>Where</p> <p>widget = the dataview widget object  sText = the HTML to be used for the entry.  record = the dataStore record  recno = index of record number in dataStore</p> <p>Return sText with any changes you may need.</p>

## ***properties***

<i>selected</i>	Currently selected item, if none selected this will be -1 otherwise zero to record count.
-----------------	---

## ***events***

<i>click</i>	<p>Fired when the dataview entry is clicked.</p> <p>Parameters: dataview widget,selected,oldSelection,event</p>
<i>dblclick</i>	<p>Fired when the dataview entry is double clicked.</p> <p>Parameters: dataview widget,selected,oldSelection,event</p>
<i>beforeRefresh</i>	Fired just before the dataview refreshes.

<i>afterRefresh</i>	Fired just after the dataview refreshes.
<i>hover</i>	<p>Fired when the mouse enters or leaves a dataview entry</p> <p>Parameters widget, in_out, recno, selected</p> <p>widget = the dataview widget object  in_out = true for in or false for out  recno = index of record number in datastore  selected = currently selected recno</p>

## methods

<i>getStore</i>	<p>Returns the DataView datastore object.</p> <p>Parameters: dataview widget,selected,oldSelection,event</p>
<i>getSelected</i>	<p>Returns the selected ds index.</p> <p>A return of -1 means no selection. This does however require autoSelect set to true.</p>
<i>refresh</i>	The forces a refresh of the dataview.
<i>select</i>	<p>Selects a entry in the dataview</p> <p>Parameters: index</p>

---

Created with the Personal Edition of HelpNDoc: [Easily create EPub books](#)

---

## Form Panel



# Form Panel

A panel that extends the panel widget but is data aware. By specifying a dataRecord the widget set the values of all appropriate child widgets from data held in the dataRecord fields.

Use a new field property with your widgets to the appropriate field name in dataRecord

```
{
    wtype: 'formpanel',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    title: 'My Form',
    dataRec: myDS.getAt(0),
    widgets: [
        {
            wtype: 'input',
```

```

        width:100,
        label:'Enter Name',
        field:'fdName'
    },
    ],
    events: {

    }
}

```

## options

All other options are the same as the panel widget

<i>dataRec</i>	User defined id for widget, if not supplied one will be assigned.
----------------	---

## methods

<i>getFieldWidget</i>	Returns the widget object for a given field name
-----------------------	--

Parameters: 'field name'

<i>getFieldConfig</i>	Returns the original widget JSON configuration object for a given field name
-----------------------	--

Parameters: 'field name'

<i>getValues</i>	Returns an array of the current form values where the key value in the array is the field name.
------------------	---

<i>clearValues</i>	Sets of the child widget values to default
--------------------	--

<i>setValues</i>	Sets the form widget values to a dataRecord.
------------------	--

Parameters: dataRecord

If one is not passed then the original dataRec property will be used

<i>setDataRec</i>	Sets the dataRec property to a new dataRecord and up dates all widgets
-------------------	--

<i>commit</i>	Writes all the widget values back to the dataRec property.
---------------	--

Parameters: dataRecord

Normally you would commit back to the dataRec property but you can pass an independent one and write the values to that one.

<i>setEditable</i>	Changes all inputs to editable or not.
--------------------	--

Parameters: canedit (true/false)

## Events

All event the same as a standard panel except:

<i>onCommit</i>	Fired after a call to the commit method.
-----------------	--

Parameters: panel widget

<i>onSetValues</i>	Fired after a call to the setValues method.
--------------------	---



## Grid



# Grid

A grid widget.

```
{
    wtype: 'grid',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    ds: myDS,
    cm: myCM
    events: {
    }
}
```

click				
	Name	Age		
1	Fred	[56]	●	●
2	Harry	[45]	●	●
3	Tom	[23]	●	●
4	Bob	[46]	●	●
5	John	[20]	●	●
click				

Pager

See section on columnModel for more details.

### options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.

<i>y</i>	Top position on page.
<i>width</i>	Width of grid.
<i>height</i>	Height of grid.
<i>headerHeight</i>	Height to use for column headers.
<i>rowHeight</i>	Height to use for each row.
<i>ds</i>	Data store object (see Data Store section).
<i>cm</i>	Column model object (see Data Store section).
<i>dsFilter</i>	Function to call for each record in the DS.  Parameters: rec  Return true if the rec should be shown
<i>offset</i>	Start position in data store (default 0) for first record.
<i>limit</i>	Max number of entries in the DS to show at any one time
<i>tbar</i>	Widget collection to be rendered to the grid toolbar.
<i>tbarHeight</i>	Height of grid toolbar.
<i>tbarStyle</i>	CSS Styling to apply to toolbar.
<i>tbarCls</i>	CSS class to apply to toolbar.
<i>bbar</i>	Widget collection to be render to the grid bottom bar
<i>bbarHeight</i>	Height of grid bottom bar.
<i>bbarStyle</i>	CSS Styling to apply to bottom bar
<i>bbarCls</i>	CSS class to apply to bottom bar.
<i>editClicks</i>	Number of mouse clicks to start editing of cell (default 1)
<i>label</i>	Text used when rendered to a form layout.
<i>roundCorners</i>	Render the grid with round corners
<i>radius</i>	Radius of round corners (default 4)
<i>showHeader</i>	Show or hide grid header (default true);
<i>altCls</i>	CSS Class to be used for alternating rows
<i>autoSelect</i>	Row to automatically select
<b>events</b>	
<i>click</i>	Fired when grid clicked.  Parameters: grid widget. event

<i>dblclick</i>	Fired when grid is double clicked.  Parameters: grid widget, event
<i>rowclick</i>	Fired when grid row clicked.  Parameters: grid widget, row, event
<i>cellclick</i>	Fired when grid cell clicked.  Parameters: grid widget, row, col, event
<i>beforeedit</i>	Fired when a grid cell is about to be edited.  Parameters: grid widget, row, col, rec  Return false to cancel editing
<i>afteredit</i>	Fired when a grid cell is about to be edited.  Parameters: grid widget, row, col, rec, fieldName, value
<i>contextmenu</i>	Fired on a right click of the grid.  Parameters: grid widget, event
<i>beforeRefresh</i>	Fired just before the grid refreshes.
<i>afterRefresh</i>	Fired just after the grid refreshes.

## methods

<i>selectRow</i>	Selects a row in the grid  Parameters: row number
<i>getStore</i>	Returns the grids Data Store
<i>getSelected</i>	Returns the data record of the selected row

---

Created with the Personal Edition of HelpNDoc: [Free help authoring tool](#)

---

## Icon



# Icon

A built-in icon widget.

```
{
    wtype:'icon',
    x:220,
```

```

        y:5,
        width:220,
        height:280,
        title:'Blue Arrow',
        color:'blue',
        icon:'arrow'
        events: {

    }
}

```

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>icon</i>	Name of icon to use (see built-in icons)
<i>title</i>	Tooltip for the icon to use (shown when you hover the mouse over the image)
<i>color</i>	Icon colour (default #000)
<i>background</i>	True or false to draw a circle behind the icon
<i>backgroundColor</i>	Colour of background circle (default #ddd)
<i>scale</i>	Scaling of icon (0 - 1) default is 0.7 so it can fit on a default toolbar height

## events

<i>afterrender</i>	Fired after the icon has been rendered.  Parameters: icon widget
<i>click</i>	Fired when the icon is clicked.  Parameters: icon widget
<i>dblclick</i>	Fired when the icon is double clicked.  Parameters: icon widget

## IFrame



# IFrame

A IFrame widget to display other web content within your application.

```
{
    wtype: 'iframe',
    x: 220,
    y: 5,
    width: 600,
    height: 400,
    src: 'http://www.bbc.co.uk',
    events: {
    }
}
```

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of iframe.
<i>height</i>	Height of iframe.
<i>src</i>	URL to be displayed in iframe.

## events

<i>loaded</i>	Called when the iframe has loaded its content.
---------------	--

## methods

<i>getSrc</i>	Returns the current src of iframe
<i>setSrc</i>	Changes the URL that the iframe displays Parameters: URL Return none
<i>getFrame</i>	Returns the DOM frame object for the iframe widget.
<i>getWindow</i>	Returns the window object of the iframe.
<i>getDocument</i>	Returns the document object of the iframe.

## Image



# Image

A image widget.

```
{
  wtype: 'image',
  x: 220,
  y: 5,
  width: 220,
  height: 280,
  src: 'images/myimage.png',
  events: {
  }
}
```

### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>src</i>	path or url to image file. You can also use the word "BLANK" to use the <code>_preferences.blankImage</code> url.
<i>style</i>	CSS styling for main image container
<i>imageWidth</i>	Width of image inside widget component
<i>imageHeight</i>	Height of image inside widget component
<i>title</i>	Set a popup title for image
<i>prehtml</i>	Renders additional html before the image. Useful to create text before the image.
<i>html</i>	Renders additional html after the image. Useful to create text below the image.

Note: If the image width and height is not passed the widget width and height is used.

## events

<i>click</i>	Fired when image clicked.
	Parameters: image widget

## methods

<i>setSrc</i>	Changes the image displayed.
	Parameters: image url

---

Created with the Personal Edition of HelpNDoc: [Free EPub and documentation generator](#)

---

## Input



# Input

A single text input widget.

```
{
    wtype:'input',    // or wtype:'textfield'
    x:220,
    y:5,
    width:220,
    height:280,
    value:'My text.'
    events: {
    }
}
```

## options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
Width	Width of widget.
Height	Height of widget.
value	Input value.
inputType	Used to set the input type (default text).  Other types "password" : Standard password input

	"number" : Allows only numeric keys "integer" : Allows only integers "currency" : Allows decimals but only to 2dp "custom" : Allows you to specify a custom regex for testing user input
inputStyle	CSS style for input control
editable	Controls whether the text in the input can be edited (default true).
watermark	Text that is displayed in input until something is typed in. Useful for prompts.
watermarkColor	Colour of watermark text (default #aaa).
enabled	This controls whether the button click events will function (default true)
label	Text used when rendered to a form layout.

### ***properties***

editable	Control where the widget can be edited (true/false)
lowercase	Displays all characters in lower case
uppercase	Displays all characters in upper case
capitalize	Displays the first letter of each word in upper case
maxlength	Specifies the maximum length of the entry string, will not allow any additional characters to be entered once the limit has been reached
invalidchars	Specifies a string of invalid characters that may not be entered

### ***events***

click	Fired when the input is clicked  Parameters: input object  Return: none
dblclick	Fired when the input is double clicked  Parameters: input object  Return: none
lostfocus	Fired when the input has lost focus  Parameters: input object  Return: none
focus	Fired when the input get focus  Parameters: input object  Return: none



change	Fired when the input value changes  Parameters: input object  Return: none
keypress	Fired when a key is pressed  Parameters: input object, event  Return: none
onenter	Fired when the ENTER key is pressed  Parameters: input object  Return: none

### **methods**

setValue	Set the value of the input.  Parameters: text  Return: none
getValue	Gets the value of the input.  Parameters: none  Return: value
setEditable	Set whether the input is editable.  Parameters: canedit (true/false)
getEditable	Returns the current editable state

---

Created with the Personal Edition of HelpNDoc: [Full-featured multi-format Help generator](#)

---

## **Label**



# Label

A form label.

```
{
    wtype:'label',
    x:220,
    y:5,
    width:220,
    height:280,
```

```

        text: "Hello World",
        events: {

        }
    }

```

### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>text</i>	Label text.
<i>label</i>	Text used when rendered to a form layout.
<i>title</i>	When included adds HTML title attribute.

### events

<i>click</i>	Fired when label clicked. Parameters: label widget
<i>dblclick</i>	Fired when label double clicked. Parameters: label widget

### methods

<i>getText</i>	Returns the label current text.
<i>setText</i>	Set the label text. Parameters stringText
<i>getValue</i>	Same as getText.
<i>setValue</i>	Same as setText.

## List



# List

A list widget is a scrollable list based on a template.

```
{
  wtype: 'list',
  x: 220,
  y: 5,
  width: 220,
  height: 280,
  tpl: new idev.wTemplate(
    "<div class='ui-listentry'>",
    "{field1}",
    "</div>"
  ),
  style: '',
  ds: myDataStore,
  offset: 0,
  limit: -1,
  autoScroll: true,
  events: {
  }
}
```

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page
<i>y</i>	Top position on page
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>tpl</i>	HTML template to be rendered for each record in the data store. Use {} brackets around the name of the field for inserting content into the template based on the record being rendered.
<i>metaData</i>	If the TPL property is a idev.wTemplate object instead of a string. The metaData (which must be an array) is applied first before the data store record fields.
<i>style</i>	CSS styling for main list container
<i>roundCorners</i>	Display round corners on list
<i>radius</i>	Radius of round corners.
<i>ds</i>	Data store object (see Data Store section)
<i>dfFilter</i>	Function to call for each record in the DS.

	Parameters: rec Return true if the rec should be shown
<i>offset</i>	Start position in data store (default 0) for first record.
<i>limit</i>	The max number of entries to include in list.
<i>autoScroll</i>	true/ false to allow the list to auto scroll if the number of records in the data store means the entries exceed the height of the list
<i>itemStyle</i>	CSS styling to be used with each list entry.
<i>titleStyle</i>	CSS styling for list title.
<i>backgroundStyle</i>	CSS styling to be used on the background to the list.
<i>autoSelect</i>	Auto show list entry selection
<i>selectColor</i>	Selection background color.
<i>selectCls</i>	Selection CSS styling class. You can use this or selectColor or both
<i>itemCls</i>	Item CSS styling class.
<i>renderer</i>	Function to be called for each entry.  Parameters widget, sText, record, recno  Where  widget = the list widget object sText = the HTML to be used for the entry. record = the dataStore record recno = index of record number in dataStore  Return sText with any changes you may need.
<i>data</i>	Array of custom data you can use in the template.
<b>events</b>	
<i>click</i>	Fired when the list item is clicked.  Parameters: list widget
<i>dblclick</i>	Fired when the list item is double clicked.  Parameters: list widget
<i>beforeRefresh</i>	Fired just before the list refreshes.
<i>afterRefresh</i>	Fired just after the list refreshes.
<b>methods</b>	
<i>getStore</i>	Returns the list data store
<i>getSelected</i>	Returns the current selected entry (-1 if none selected)

<i>getItemId</i>	Return the id of the nTh list entry  Parameters: index  Return id
<i>updateItem</i>	Update the CSS property of a give list entry.  Parameters: index, propertyName, value  Return none
<i>setScrollPos</i>	Sets the top position of the list  Parameters: pos (pixels)  Return none
<i>getScrollPos</i>	Returns the last scroll position set
<i>updateItem</i>	Updates the CSS of a given item in the list  Parameters: index,propertyName,value  Where propertyName is a standard CSS property
<i>refresh</i>	Used to re-render the list
<i>clearSelection</i>	Clear the list selecting index

---

Created with the Personal Edition of HelpNDoc: [Full-featured EPub generator](#)

---

## ListBox



# Listbox

A listbox widget. Creates a scrollable list, useful for menus.

```
{
    wtype: 'listbox',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    title: 'Main Menu',
    items: [
        {
            text: 'Option 1',
            icon: 'arrow',
            textStyle: ''
        }
    ]
}
```

```

        image:'',
        widget: null,
        widgetWidth:0,
        events: {
        }
    },
    ],
    events: {
    }
}

```



## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget. Use 'auto' if you want the list height to be controlled by number of items
<i>cls</i>	CSS class for listbox
<i>title</i>	Listbox title to be displayed at the top of the list.
<i>titleHeight</i>	Height of title bar
<i>items</i>	A collection of list box items. See listitem details
<i>background</i>	CSS class for list entry

## List Item options:

```

{
    text:'Option 1',
    textStyle:'',
    icon:'arrow',
    image:'',
    widget: null,
    widgetWidth:0,

```

```

        events: {
        }
    }

```

<i>icon</i>	Name of icon to be displayed on right of list entry
<i>image</i>	Image URL to be displayed on left of list entry
<i>text</i>	List entry text
<i>textstyle</i>	CSS styling for text
<i>widget</i>	Optional iDevUI widget to be display after text JSON definition for widget.
<i>widgetWidth</i>	Width of widget
<i>itemHeight</i>	Height of item
<i>selectable</i>	Allow item to be shown selected (default true)
<i>items</i>	A collect of list box items. See listitem details
<i>background</i>	CSS class for list entry

## events

<i>afterRender</i>	Fired after the listbox widget has been rendered.  Parameters: listbox widget
<i>click</i>	Fired when a list entry is clicked  Parameters: listbox, index, event
<i>dblclick</i>	Fired when a list entry is double clicked  Parameters: listbox, index, event

## methods

<i>add</i>	This method will add a new item to the listbox.  Parameters: item config (see Listitem for details)  Return: new widget object
<i>get</i>	This method returns a listitem widget base on a index number (starts at 0)  Parameters: index  Return: new widget object or null



# Object

A object widget to display other web content within your application.

```
{
    wtype: 'object',
    x: 220,
    y: 5,
    width: 600,
    height: 400,
    src: 'http://www.bbc.co.uk',
    events: {
    }
}
```

Typical use of this widget is to display older websites that do not have a doctype. Especially if you need to mix IE9 and IE8 content

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of iframe.
<i>height</i>	Height of iframe.
<i>src</i>	URL to be displayed in iframe.
<i>type</i>	Type of content (default text/html)

## events

<i>loaded</i>	Called when the iframe has loaded its content.
---------------	--

## methods

<i>getSrc</i>	Returns the current src of object
<i>setSrc</i>	Changes the URL that the object displays Parameters: URL Return none



## Panel



# Panel

This widget is used to render an area of text/html.

```
{
  wtype: 'panel',
  x: 220,
  y: 5,
  width: 220,
  height: 280,
  layout: 'table',
  roundCorners: false,
  autoScroll: true,
  style: 'font-size: 18px;',
  html: "My text panel",
  widgets: [

  ],
  events: {

  }
}
```

### wtype

panel

### options

id	optional, the framework will assign one automatically
x	The position from the left of the page section (header, content or footer), if omitted it will be relative to the last fixed widget or section on the page
y	The position from the top of the page, if omitted it will be relative to the last fixed widget or section on the page
width	The width of the panel
height	The height of the panel
title	Panel title. This creates a panel with a title bar.
titleHeight	Height of title bar
closable	If a title bar is created a close icon is display on the left
minisable	If a title bar is created a minimise icon is display on the left
maxisable	If a title bar is created a maximise icon is display on the left

expandable	<p>Allows the panel to collapse to the titlebar and back to full height.</p> <p>If the panel is part a frame the expandability will vary depending the frame area.</p>
collapsed	Sets the panel as collapsed on and expandable panel (default false).
draggable	Allows the panel to be moved around the screen.
roundCorners	Render the panel with round corners
radius	Radius of round corners (default 4)
autoScroll	Allow the text on the panel to scroll when a user swipes the screen
style	Standard CSS styling for the core panel
html	The text or HTML to be displayed in the panel
cls	CSS class to override the core widget styling
layout	<p>The layout determines how the panel widgets are laid out</p> <p><b>flow</b> (default) : as the widgets are created</p> <p><b>form</b>: 2 columns with the first column containing the label and the second the actual widget. If no label property then the widget spans both columns</p> <p><b>table</b>: widgets are laid out in a table format controlled by the columns property and the rowHeight property</p> <p><b>column</b>: widgets are laid out in a single table row.</p> <p><b>row</b>: widgets are laid out in a single table column a row for each widget.</p> <p><b>frame</b>: widgets are laid out in a north, south, west, center and east format, where the widgets must be panels and the area property set to the right location.</p> <p><b>fit</b> the first widget only is rendered to the same size as the panel. Any other widgets are made the same size as the panel but are hidden. You can use the show and hide methods of each widget to bring them into play.</p>
layoutConfig	<p>Additional layout properties.</p> <pre>{   fit:true. }</pre> <p>At present only "column" &amp; "row" use this property with a fit attribute so a column can match the panel height and row can match the panel width.</p>
area	<p>Location of panel when the parent panel is in "frame" layout.</p> <p>options: north, south, west, center and east</p>

columns	Works with the table layout to control the widget layout
columnAlign	Vertical alignment for each
rowHeight	Works with the table layout to control the row height. If not provided the widgets will control the height.
padding	Gap in pixels around the widgets
bodyStyle	CSS styling to be applied to the main body of the panel
backgroundStyle	CSS styling to be applied to the background of the panel
backgroundCls	CSS class to be applied to the background of the panel
icon	URL of icon to be displayed on left of panel
tbar	Widget collection to be rendered to the panel toolbar.
tbarHeight	Height of panel toolbar.
tbarStyle	CSS Styling to apply to toolbar.
tbarCls	CSS class to apply to toolbar.
bbar	Widget collection to be rendered to the panel bottom bar.
bbarHeight	Height of panel bottom bar.
bbarStyle	CSS Styling to apply to bottom bar.
bbarCls	CSS class to apply to bottom bar.
widgets	A collection of widget configurations or objects.
enabled	This controls whether the click events will function (default true)
center	Used on a row layout to center the panel widgets
shadow	Show a drop shadow around panel
shadowSize	Sets the size of the shadow in pixels
shadowColor	Sets the shadow colour (default #ccc)
autoFocus	Sets whether to autoFocus on the first input box or text area. (true/false)
focusID	ID of input or text area widget which to focus on first (once the page has rendered).
titleTextCls	CSS class to be applied to title text.

## events

<i>afterRender</i>	Fired after the widget has been created.  Parameters: panel object
<i>minimise</i>	Fired when minimised button clicked.  Parameters: panel object

<i>maximise</i>	Fired when maximised button clicked.  Parameters: panel object
<i>close</i>	Fired when close button clicked.  Parameters: panel object  Return false to cancel close
<i>click</i>	Fired when panel clicked.  Parameters: panel object
<i>dblclick</i>	Fired when panel double clicked.  Parameters: panel object
<i>beforeClose</i>	Fired before the widget closes. (Useful for Window widgets)  Parameters: panel object  Can return true to continue closing the widget or false to halt the close.
<i>afterClose</i>	Fired after the widget closes. (Useful for Window widgets)  Parameters: panel object

## ***methods***

<i>innerHTML</i>	Sets the inner html of the panel.  Parameters: html string  Returns: none
<i>close</i>	Close panel.  This will remove the panel and its widgets from the screen
<i>addWidget</i>	Adds a new widget to the panel  Parameters: widget (JSON)  Return true/false  Note: At present this is not designed for panels with "frame" or "accordion" layout.
<i>removeWidget</i>	Removes a widget from the panel  Parameters: id  Returns: none.
<i>setTitle</i>	Sets a new panel title if the panel was originally created with one.  Parameters: text  Return none

*getTitle*

Returns the panel title if the panel was originally created with one. If no title then an empty string is returned.

---

Created with the Personal Edition of HelpNDoc: [Produce electronic books easily](#)

---

## Progress Bar



# Progress Bar

A progress bar widget.

```
{
    wtype: 'progressbar',
    x:220,
    y:5,
    width:220,
    height:280,
    events: {

    }
}
```



### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>value</i>	Initial value of progress (0 - 1.0)
<i>roundCorners</i>	Render the widget with round corners
<i>radius</i>	Radius of round corners (default 4)
<i>label</i>	Text used when rendered to a form layout

### events

<i>click</i>	Fired when progressbar clicked.  Parameters: progressbar widget
--------------	---

<i>dblclick</i>	Fired when progressbar double clicked.  Parameters: progressbar widget
-----------------	--

## methods

<i>getValue</i>	Returns current value of progress bar
<i>setValue</i>	Set progress bar value  Parameters: value (between 0 - 1.0)  Return none

---

Created with the Personal Edition of HelpNDoc: [Easy EBook and documentation generator](#)

---

## Radio Button



# Radio Button

A radio button widget.

```
{
    wtype: 'radio',
    x: 220,
    y: 5,
    text: 'Apples',
    textStyle: '',
    value: 'A',
    group: 'fruit',
    checked: true
    events: {
    }
}
```

Apples ☒ Oranges ☐

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>Y</i>	Top position on page.
<i>text</i>	Label
<i>value</i>	Radio button value.
<i>group</i>	Group name that the button belongs to.

checked	Initial checked state (default false).
textStyle	CSS Styling for text label.
label	Text used when rendered to a form layout

## events

click	Fired when radio button clicked.  Parameters: widget
-------	--

## methods

check	Changes the checked state of a radio button.  Parameters: checked (true/false)  Return: none
getValue	Gets the value of the radio group based on the checked button. You can use any of the buttons to get the value.  Parameters: none  Return: value
isChecked	Determines the button is checked  Parameters: none  Return: true / false

---

Created with the Personal Edition of HelpNDoc: [Free EPub producer](#)

---

## Slider



# Slider

A slider widget.

```
{
    wtype: 'slider',
    x: 220,
    y: 5,
    value: 0.5,
    events: {

    }
}
```



### options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
value	Progress start value.
label	Text used when rendered to a form layout.

### events

change	Fired when slider changes position.  Parameters: widget,value
--------	---

### methods

getValue	Gets the value of the slider.  Parameters: none  Return: value
setValue	Set the new value of the slider.  Parameters: value  Return: none

---

Created with the Personal Edition of HelpNDoc: [Free iPhone documentation generator](#)

---

## SVG Button



# SVG Button

A SVG button widget.

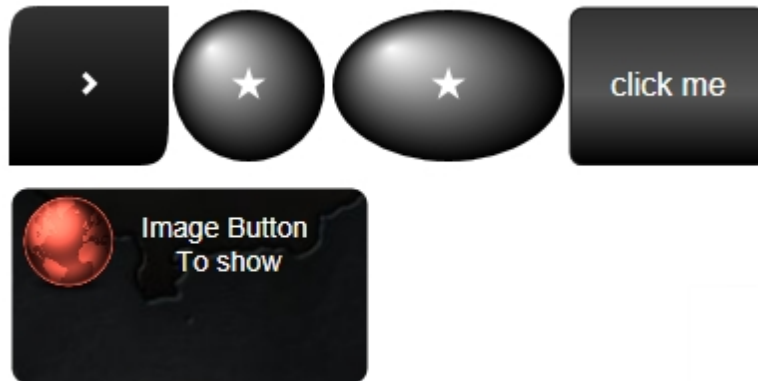
```
{  
    wtype: 'svgbutton',  
    x:220,  
    y:5,  
    width:220,  
    height:280,  
    color:'blue',  
    icon:'arrow',  
    iconAlign:'right'  
    text: "Click Me",  
}
```



```

        events: {
        }
    }

```



The buttons on iDevUI were created using SVG until version 1.1, when two separate widgets were created, one using SVG (keep backward compatability) and one using CSS.

Note: that some of the button default values are set in the ***\_preferences*** object.

## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of button.
<i>height</i>	Height of button.
<i>icon</i>	Name of button icon to use (see built-in icons)
<i>iconAlign</i>	Position of icon on button ("left" or "right").
<i>text</i>	Button text.
<i>textXOffset</i>	Offset from button centre for text.
<i>textYOffset</i>	Offset from button centre for text.
<i>textAlign</i>	Button text alignment ("left", "center", "right").
<i>textVAlign</i>	Button text vertical alignment ("top", "center", "bottom").
<i>color</i>	Button colour ("blue", "red", "green", "orange", "purple" or "black").
<i>transparent</i>	Button with no fill color but can have a border.
<i>startColor</i>	User defined colour range (ie #000). Works with endColor.
<i>endColor</i>	User defined colour range.
<i>border</i>	Whether to display the border (true/false).
<i>borderColor</i>	Colour of button border (default same as endColor).

fontColor	Colour of text on button (default #fff).
fontSize	Size of text on button (default 16).
iconColor	Colour of icon on button (default #fff).
iconRotation	Angle to draw icon.
roundCorners	Render the panel with round corners
radius	Button corner radius.
image	Image to display on button (JSON object).  <pre>image: {   src:'pathToImage',   x:10,   y:10,   width:32,   height:32 }</pre>
imageFill	Set this property to a image URL if you want a image to fill the button.
path	An SVG path string to set the shape on the button. (default "").
shape	By default a button is a rectangle, but you can also specify "circle" or "ellipse".
gradient	The colour gradient is by default "linear" but can also specify "radial".
rx	"radial" gradient centre (default is centre of button)
ry	"radial" gradient centre (default is centre of button)
label	Text used when rendered to a form layout.
enabled	This controls whether the button click events will function (default true)
title	Text shown when mouse hovers over button
toggle	Places the button into toggle mode (default false). Where the button can be shown as depressed.
down	Initial state of button in toggle mode (true/false)
startTColor	Toggle start color when button depressed (default is button endColor).
endTColor	Toggle end color when button depressed (default is button startColor).

## methods

setText	Set the button text.  Parameters: text  Return: none
getText	Get the current button text.  Return: button text

textColor	<p>Sets the color of the button text.</p> <p>Parameters: CSS color "ie #fff"</p> <p>Return: none</p>
setBorderColor	<p>Sets the border color of the buttont.</p> <p>Parameters: color, width</p> <p>CSS color "ie #fff"</p> <p>width in pixels</p> <p>Return: none</p>
isDown	<p>Used to determine if button is down when in toggle mode.</p> <p>Parameters: none</p> <p>Return: true/false</p>
reColor	<p>Used the recolor the button dynamically.</p> <p>Parameters: startColor, endColor</p> <p>Return: none</p>
enableButton	<p>Enables or disables button to react to click events</p> <p>Parameters enable (true/false)</p>
toggleButton	<p>Shows button in toggle mode up or down.</p> <p>{Parameters: down (true/false)}</p> <p>Return: none</p>

## events

<i>beforeRender</i>	<p>Fired before the widget has been created.</p> <p>Parameters: button widget</p>
<i>afterRender</i>	<p>Fired after the widget has been created.</p> <p>Parameters: button widget</p>
<i>click</i>	<p>Fired when button clicked.</p> <p>Parameters: button widget</p>
<i>canToggle</i>	<p>Fired before a button is toggled.</p> <p>Parameters: button widget</p> <p>Return true/false</p>
<i>onToggle</i>	<p>Fired after a button is toggled.</p> <p>Parameters: button widget</p>

## Spacer



# Spacer

Creates a form spacer designed to separate widgets on a page.

```
{
    wtype: 'spacer',
    width: 220,
    height: 280
}
```

You can specify either width or height depending on how you are using the space.

### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.

## Switch



# Switch

A switch widget.

```
{
    wtype: 'switch',
    x: 220,
    y: 5,
    value: 'A',
    on: true,
    events: {
    }
}
```



### options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
on	Initial switch state (default false).
label	Text used when rendered to a form layout.

### events

change	Fired when switch changes state. Parameters: widget
--------	--

### methods

setOn	Changes the state of a switch. Parameters: on (true/false) Return: none
isOn	Determines the switch is on. Parameters: none Return: true / false

---

Created with the Personal Edition of HelpNDoc: [Free Kindle producer](#)

---

## Tab Panel



# Tab Panel

A tab panel widget.

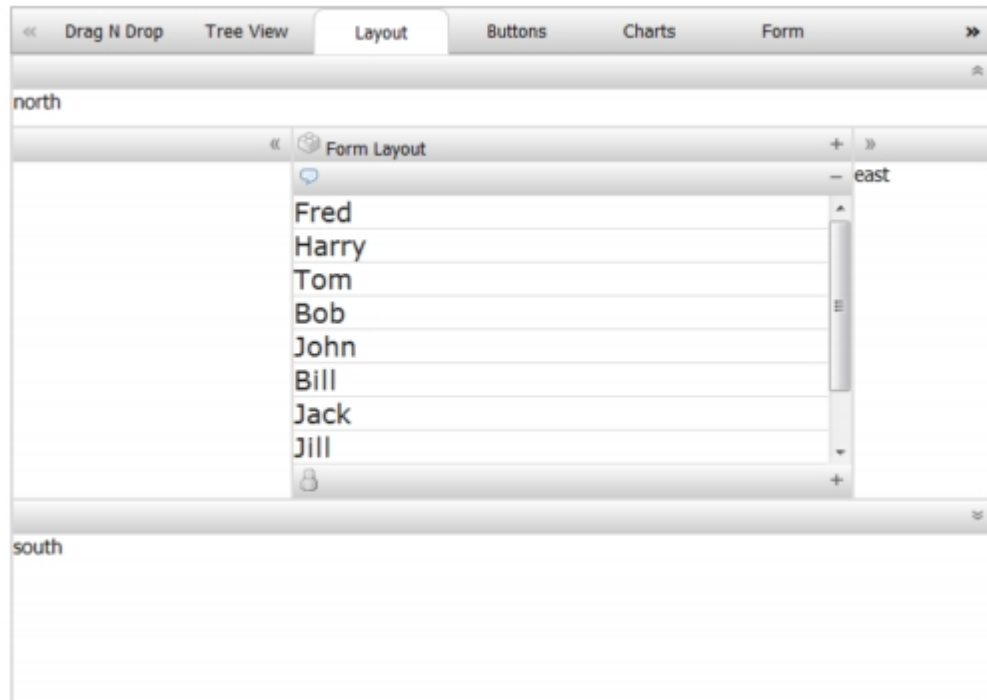
```
{  
    wtype: 'tabpanel',  
    x: 220,  
    y: 5,  
    width: 220,  
    height: 280,  
    tabWidth: 95,  
    tabHeight: 34,  
    activeTab: 5,  
    style: 'border: 1px solid #ccc;',  
}
```

```

        tabStyle:'font-size:10pt;',
        widgets:[
            // Add tabs here as panel widgets.
        ],
        events: {

        }
    }
}

```



Note all widgets MUST be panels.

### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>style</i>	CSS styling applied to core widget.
<i>tabWidth</i>	Width of each tab (default 90). If there are more widget panels than the width of the tab panel then it will automatically add scrolling controls at each end.
<i>tabHeight</i>	Height of tabs.
<i>tabScrollWidth</i>	Width of scroll control (Default 25).
<i>tabStyle</i>	CSS styling applied to each tab.
<i>activeTab</i>	First tab to display (default 0).

## events

<i>showTab</i>	Fired when a tab is shown.  Parameters: tab panel widget, active tab
<i>beforeCloseTab</i>	Fired before a tab is closed.  Parameters: tab panel widget, tab index  Return false if you want to abort closure.
<i>afterCloseTab</i>	Fired after a tab is closed  Parameters: tab panel widget, tab index  Return none
<i>beforeShowTab</i>	Fired before a tab is about to be shown.  Parameters tab panel widget, tab index  Return false to cancel show of tab

## methods

<i>showTab</i>	Show a particular tab.  Parameters: tab index  Return none
<i>closeTab</i>	Close a particular tab  Parameters: tab index  Return none
<i>addTab</i>	Add a new tab  Parameters: panel JSON widget  Tab will be added to the end of the tab bar.

---

Created with the Personal Edition of HelpNDoc: [Full-featured EPub generator](#)

---

## Text Area



# Text Area

A multi-line text input widget.

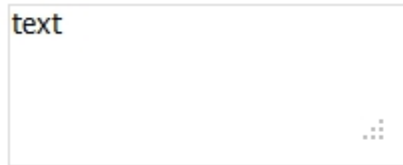
```
{
```

```

        wtype: 'textarea',
        x: 220,
        y: 5,
        width: 220,
        height: 280,
        value: 'My text area.'
        events: {

    }
}

```



### options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
Width	Width of widget.
Height	Height of widget.
value	Input value.
cls	CSS class to be applied to core widget
style	CSS styling to be applied to core widget
inputStyle	CSS style for input control
inputCls	CSS class to be applied to input element of the widget
editable	Controls whether the text in the input can be edited (default true).
watermark	Text that is displayed in input until something is typed in. Useful for prompts.
watermarkColor	Colour of watermark text (default #aaa).
enabled	This controls whether the button click events will function (default true)
label	Text used when rendered to a form layout.

### properties

editable	Control where the widget can be edited (true/false)
----------	---

### events

click	<p>Fired when the input is clicked</p> <p>Parameters: input</p> <p>Return: none</p>
dblclick	<p>Fired when the input is double clicked</p> <p>Parameters: input</p>



	Return: none
lostfocus	Fired when the input has lost focus  Parameters: input  Return: none
focus	Fired when the input get focus  Parameters: input  Return: none
change	Fired when the input value changes  Parameters: input  Return: none
keypress	Fired when a key is pressed  Parameters: input, event  Return: none

## **methods**

setValue	Set the value of the text area.  Parameters: text  Return: none
getValue	Gets the value of the textarea.  Parameters: none  Return: value
setEditable	Set whether the input is editable.  Parameters: canedit (true/false)
getEditable	Returns the current editable state

---

Created with the Personal Edition of HelpNDoc: [Easily create EBooks](#)

---

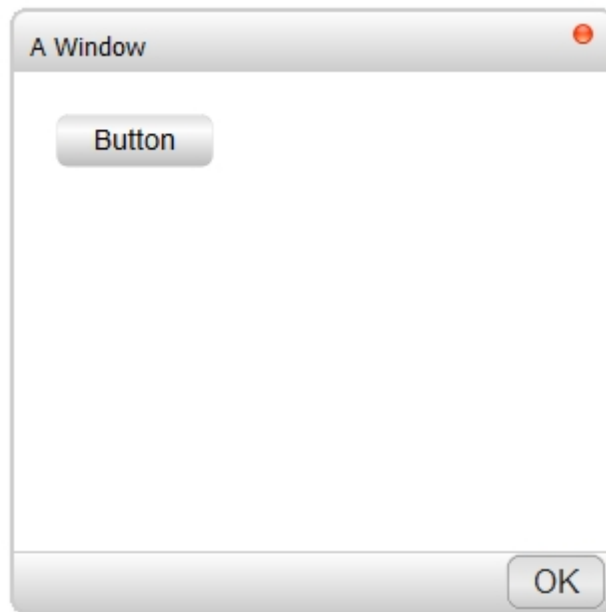
## **Window**



# Window

A window widget is basically a panel with some additional features.

```
var win = new idev.ui.widgetWindow({
  title: 'A Window',
  width: 300,
  height: 300,
  modal: true,
  autoScroll: true,
  padding: 20,
  widgets: [
    {
      wtype: 'button',
      width: 80,
      height: 28,
      text: 'Button',
      events: {
        click: function()
        {
          var win2 = new idev.ui.widgetWindow({
            title: 'Second Window',
            width: 200,
            height: 200,
            modal: true
          });
          win2.show();
        }
      }
    }
  ],
  bbar: [
    '>>',
    {
      wtype: 'button',
      width: 50,
      height: 28,
      text: 'OK',
      color: 'silver',
      events : {
        click: function(page, btn)
        {
          win.close();
        }
      }
    }
  ]
});
win.show();
```



## options

A window cannot be created as part of another widget. Can also open sub windows from a window.

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of button.
<i>height</i>	Height of button.
<i>modal</i>	The background widgets are unavailable while the window is showing
<i>roundCorners</i>	Always true
<i>closable</i>	Always true
<i>draggable</i>	Always true
<i>panelCls</i>	By default is "ui-window"
<i>bodyStyle</i>	By default #fff
<i>autoFocus</i>	Sets whether to automatically focus on the first input box or text area. (true/false)
<i>focusID</i>	ID of input or text area widget which to focus on first (once the page has rendered).
<i>autoClose</i>	Sets whether the window will be automatically closed if you click outside of it. (Default: false)  Note: This will automatically make your window modal.

## Events

The Window Widget uses the same events as the Panel widget. ([See Panel widget](#))



# Window Form

A window widget based on the form panel.

```
var win = new idev.ui.widgetFormWindow({
    title: 'A Window',
    width: 300,
    height: 300,
    modal: true,
    autoScroll: true,
    padding: 20,
    widgets: [
        {
            wtype: 'button',
            width: 80,
            height: 28,
            text: 'Button',
            events: {
                click: function()
                {
                }
            }
        }
    ],
    bbar: [
        '>>',
        {
            wtype: 'button',
            width: 50,
            height: 28,
            text: 'OK',
            color: 'silver',
            events : {
                click: function(page,btn)
                {
                    win.close();
                }
            }
        }
    ]
});
win.show();
```

All options, methods and events follow the underlying form panel.



# Additional Widgets

Additional Widgets are widgets that are not included in the idewui base download. You can download any of these separately to use with idewui.

Each widget has its own separate documentation in the form of a ReadMe file in the zip file download.

You can download Additional Widgets [here](#).



# Base widget methods

Each widget has a standard set of methods that allow you control how it appears or functions.

### ***Properties***

Base properties of all widgets

data	A user defined object for additional data. This used by the DataView widget for example.
fn	<p>A user defined object to extend the widget with additional methods.</p> <p>Add to config widget prior to creation .</p> <p>fn has parent property referring back to the widget</p> <pre>{   ...   fn: {     myfunc:function()     {       this.parent //reference back to the widget       // Code     }   }   ... }</pre>
anchor	<p>Enables the widget to be anchored from the top or left or both and the width or height to be extended the right or bottom of the parent panel.</p> <p>Values should be 'left' or 'top' or 'left top'</p> <p>This is applied to all widgets</p>
anchorMargin	<p>A JSON object with the right and bottom offsets</p> <pre>{   right:10   bottom:20 }</pre>

## Methods

	}  These offsets are taken off the calculated width or height.
<code>disabledCls</code>	specifies a css class to use is enable is set to false
<code>show</code>	Show a hidden widget.
<code>hide</code>	Hide a visible widget.
<code>fadeOut</code>	Hide the widget by slowly changing its transparency  Parameters: animSpeed,easing,callback  <i>animSpeed:</i>  Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast".  <i>easing:</i>  The way it is faded (see easing list)  <i>callback:</i>  Function to call when animation is complete
<code>fadeIn</code>	Show the widget by slowly changing its transparency  Parameters: animSpeed,easing,callback  <i>animSpeed:</i>  Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast".  <i>easing:</i>  The way it is faded (see easing list)  <i>callback:</i>  Function to call when animation is complete
<code>fadeToggle</code>	Toggle the transparency of a widget  Parameters: animSpeed,easing,callback  <i>animSpeed:</i>  Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast".  <i>easing:</i>  The way it is faded (see easing list)  <i>callback:</i>

	Function to call when animation is complete
<i>setDraggable</i>	Set where the widget can be dragged around the page  Parameters: true/false
<i>isDraggable</i>	Returns true/false as to whether the widget is draggable
<i>isVisible</i>	Returns true/false as to where the widget is visible
<i>isEnabled</i>	Returns true/false if a widget is enabled. Disabling a button prevents the click events.
<i>destroy</i>	Destroys the widget object and removes from page.
<i>getPosition</i>	Return the current top-left of the widget in a JSON object  <pre>{   x:150,   y:67 }</pre>
<i>getWidth</i>	Returns the width of the widget.
<i>getHeight</i>	Returns the height of the widget.
<i>moveTo</i>	Moves the widget to a specific location on the page.  Parameters: x,y,animSpeed,easing,callback)  <i>animSpeed: in milliseconds</i>  <i>easing:</i>  The way it is faded (see easing list)  <i>callback:</i>  Function to call when animation is complete
<i>moveBy</i>	Same as moveTo but the x & y parameters are relative to the widget current position
<i>animate</i>	Animates a specific CSS properties.  Parameters: properties,animSpeed,easing,callback)  properties: a JSON object of CSS properties with their target value  <i>animSpeed: in milliseconds</i>  <i>easing:</i>  The way it is faded (see easing list)  <i>callback:</i>  Function to call when animation is complete



<i>focus</i>	Sets focus on an input widget
<i>enable</i>	Designed to enable/disable a button widget.  Parameters: enable (true/false)
<i>getParent</i>	Returns the widgets parent widget
<i>getWidget</i>	Returns a widget child widget by index (zero indexed)  Parameter: index  Return widget object
<i>widgetCount</i>	Returns the number of child widgets.
<i>collapse</i>	Collapses a widget to a specific height  Parameters: height
<i>expand</i>	Expands a widget back to its original height
<i>collapseWidth</i>	Collapses a widget to a specific width  Parameters: width
<i>expandWidth</i>	Expands a widget back to its original width
<i>addEvent</i>	Adds an event callback function to a widget  Parameters: sEventName,callback  Event name should be a standard HTML event
<i>removeEvent</i>	Removes an event by name  Parameters: sEventName
<i>css</i>	Updated a CSS property  Parameters: property, css  ie.  \$get("mywidget").css("color", "#fff");  Pass only the property to get the value
<i>attr</i>	Updates a HTML object attribute value  Parameters: property, attr  ie.  \$get("mywidget").attr("cls", "newClass");  Pass only the property to get the value

<i>fireEvent</i>	<p>Allows the code to trigger a event code as if the event has occurred. Useful to fire the click event of button from the onEnter event of an input</p> <p>Parameters event, params</p> <p>event = an event name ('click')</p> <p>params = a parameters object</p>
------------------	---

## Events

<i>beforeRender</i>	<p>Fired before the widget has been rendered.</p> <p>Parameters: widget</p>
<i>afterRender</i>	<p>Fired after the widget has been rendered.</p> <p>Parameters: widget</p>
<i>onShow</i>	<p>This event is call when a widget is shown and cascades down the all the children below</p>
<i>onHide</i>	<p>This event is call when a widget is hidden and cascades down the all the children below</p>

The above events are common to all widgets

## Easing List

Each one of the names below gives a different effect to the animation

linear  
 swing  
 easeOutBounce  
 easeOutElastic  
 easeOutBack  
 easInBounce  
 easeInElastic  
 easeInBack



# Data store

A data store is a collection of records maintained with the application that can be bound to a widget.

```
var ds = new idev.data.dataStore({
    fields: ['field1', 'field2'],
    data: [
        { 'field1': 'Value 1', 'field2': 'Value 2' }
    ]
});
```

The data store can be primed with some static data or by specifying a URL to fetch its data from the server.

### *options*

fields	A user defined array of field names.
data	An array of data for each record (optional)
url	URL to server side script that will return some data
params	<p>A JSON object with a collection of parameters to be posted to the above URL.</p> <pre>params:{     comand:'mycommand' }</pre> <p>You can use this command parameter to send a command to the server side script. The server side script should then return a JSON structure like so:</p> <pre>{"success":true,"results":[{"columnName1":"Row1 - Column1","columnName2":"Row1 - Column2" etc..}</pre> <p>In PHP, you can get fields from the database and return the output using your command as a function in your API like so:</p> <pre>public function mycommand() {     global \$idev;      // Connect to the Database (example showing MySQL, can use any database)     \$conn = mysql_connect("example.com",, "username", "password");     if(\$conn)     {         \$db = mysql_select_db("database_name");     } }</pre>

	<pre>// Get Fields \$sql = "select top(10) * from table"; \$query = mysql_query(\$sql); \$data = array(); while(\$row = mysql_fetch_row(\$query)) {     \$data[] = \$row; }  // Return JSON result \$this-&gt;pushResult("results",\$data); }</pre> <p>Note: This function should be put into ajax.php on the server side. (You do not have to use PHP).</p>
--	--

autoDestroy	<p>Set to true if you want dataStore to be deleted when the widget bond to it is destroyed (default false)</p> <p>This is useful if the ds property of a dataview widget no longer needs the dataStore when it is destroyed. But the default setting ensure shared or global dataStore are not remove when a widget using it is destroyed.</p>
-------------	--

## events

load	<p>Called when new data is loaded</p> <p>Parameters: ds</p>
loadexception	<p>Called when the fetch method fails</p> <p>Parameters: ds, textStatus</p>

## methods

load	<p>Load the data store with data.</p> <p>Parameters: array of records</p> <p>Return: none</p>
onload	<p>Sets the load event callback</p> <p>Parameters: callback function</p>
getFieldName	<p>Gets the field name by index number (starts at 0).</p> <p>Parameters: index</p> <p>Return: name</p>
fieldCount	<p>Returns the number of fields in the data store.</p> <p>Parameters: none</p> <p>Return: count</p>

getAt	<p>Returns a give record by index number (starts at 0).</p> <p>Parameters: index</p> <p>Return: record</p>
add	<p>Adds a new record to data store</p> <p>Parameters: array of data for record fields or JSON record</p> <pre>[ 'Value 1', 'Value 2' ]</pre> <p>or</p> <pre>{   'field1', 'value 1',   'field2', 'value 2' }</pre> <p>Parameters: data,nobind,first (true/false)</p> <p>Return: none</p>
removeAll	<p>Removes all the records in a data store.</p> <p>Parameters: none</p> <p>Return: none</p>
removeAt	<p>Removes a record by index</p> <p>Parameters: index</p> <p>Returns true/false</p>
removeLast	<p>Remove last record in collection.</p>
getCount	<p>Get the record count of a data store.</p> <p>Parameters: none</p> <p>Return: count</p>
fetch	<p>Executes an AJAX call to the URL property and if successful loads the new data.</p> <p>Parameters: callback</p> <p>callback is an optional parameter which defines a function to be called upon a successful load. The callback function is passed the dataStore object.</p> <p>Also if the dataStore has been used by other widgets (like dataview) this will be refreshed.</p>
setParam	<p>Sets a URL parameter ready for the data fetch.</p> <p>Parameters: sParam,value</p>

getModified	Returns an array of the modified records.
commitAll	Commits all modified records and changes their status to not modified.
blankRec	Returns a blank dataRecord object.
asString	<p>Returns the dataStores as a string in JSON form or CSV</p> <p>Parameters: type,separator</p> <p>type: "csv" or "json" (default)</p> <p>separator: character used as a field data separator (default comma)</p>
sort	<p>Sorts the data array records in ascending or descending order</p> <p>Parameters: fieldname, ascending (true/false)</p> <p>This will also refresh any bindings.</p>
find	<p>Find a record that matches a value</p> <p>Parameters: sField,sValue,start,matchAny,caseSensitive</p> <p>sField: field name</p> <p>sValue: value to find</p> <p>start: start position on record collection (default 0)</p> <p>matchAny:(true/false)Look for text at any position in field value (default true)</p> <p>caseSensitive:(true/false) for case sensitivity (default true)</p>
bind	This method binds the dataStore to a widget so that when the dataStore changes the widgets refresh function will get called.
updateBinds	This method can be called at any time to update any widgets bound to the dataStore.

---

Created with the Personal Edition of HelpNDoc: [Full-featured multi-format Help generator](#)

---

## Data Record



# Data store record

A data store of records. Used to hold individual data entries in JSON format

```
var rec = new idev.data.jsonRecord({
    fields: ['field1','field2'],
```

```

        data: [
            { 'field1':'Value 1','field2':'Value 2' }
        ]
    });

```

## options

fields	A user defined array of field names.
data	An array of data for each record (optional)
store	DataStore the record is attached to.

## methods

getStore	Returns the datastore the record is attached to
load	<p>Load the data record with data.</p> <pre>[{ 'field2':'Value 1','field2':'Value 2' }]</pre> <p>Parameters: array of data for records, each record is a list of name and value pairs, where the name is the name of a field.</p> <p>Return: none</p>
loadText	<p>Loads the record from a string</p> <p>Parameters: record_data,separator</p> <p>record_data: string of field data or order of field declaration</p> <p>separator: separating character between field data (default comma)</p>
set	<p>Set a field by name.</p> <p>Parameters: name, value ,nobind</p> <p>Return: true / false</p> <p>if false name not found.</p> <p>The 'nobind' parameter stops any widgets bound to the datastore from being updated (default false). You may wish to do this before calling the updateBinds method on the main datastore.</p>
get	<p>Get the value of a field in the data record.</p> <p>Parameters: name</p> <p>Return: value</p>
isModified	Returns true/false if record has changed
commit	Clears the modified status
asString	<p>Returns the record as a string.</p> <p>Parameters: type,separator</p>

	type: "csv" or "json" (default)
	separator: character to use as a field data separator (default comma)
clone	Returns a copy of the record.  Note this record is not attached to any DataStore.
getRecNo	Returns the record number of the dataRecord store.

---

Created with the Personal Edition of HelpNDoc: [Easily create PDF Help documents](#)

---

## ColumnModel



# Column Model

Defines the columns in a grid widget

```
var dataCM = new idev.data.columnModel({
    columns: [

        new rowMarker(),
        { header:'Name', field:'fdName', width:180,
          editor: {
              wtype:'textfield'
          }
        },
        { header:'Age', field:'fdAge', width:50, renderer: function(value)
        { return '['+value+'']'; } },
        new rowAction({
            events: {
                afterRender: function(grid,col)
                {
                },
                click: function(grid,col)
                {
                    $debug("click");
                }
            }
        }),
        new rowAction({
            events: {
                afterRender: function(grid,col)
                {
                    $debug("afterRender");
                },
                click: function(grid,col)
                {

```



```

        $debug("click");
    }
    })
}
});

```

Note the "rowMarker" is a widget that is based on the column object

```

var rowMarker = idev.data.column.extend(
{
    init: function(config)
    {
        this._super(config);
        this.header = "";
        this.width = 20;
        this.cls = "ui-hgradient";
        this.noselect = true;
        this.renderer = function(value,rec,row,col)
        {
            return (row+1);
        };
    }
});

```

### options

columns	An array of column objects or column JSON definitions
---------	---

### methods

getCount	Returns the number of columns in the model.
getAt	Returns the column object at an index in the model.  Parameters index  Returns column object (zero indexed)

---

Created with the Personal Edition of HelpNDoc: [Full-featured EBook editor](#)

---

## Column



# Column

The definition of a column in a grid widget.

```

{
    header:'Name',
    field:'fdName',
    width:180,

```

```

        editor: {
            wtype:'textfield'
        }
    }

```

## options

header	Text to display in column header
field	field name from grid dataStore, it display in cells
width	Column width in pixels
editor	The JSON definition of a iDevUI widget to be used when editing.  The editor widget is created and then destroyed for each edit.
renderer	A function to be called before rendering the cell  Parameters value,rec,row,col  value: current field value rec: current dataStore record row: current row in grid col: current column index  This is a change to override the existing value or add some HTML formatting around the value.  example:  return "<b>" + value + "</b>";  Return new value
headerRenderer	Optional function to be called if you wish to overwrite the entering of the column header  Parameters: column object, column index  Return new HTML or text for header

## events

beforeEdit	Fired before the editing starts  Parameters: grid,row,col,rec  Return: false if you want to cancel the editing
afterEdit	Fired at the end of editing  Parameters: grid,row,col,field,oldValue,newValue  Return: none
headerClick	Fired when the header of a column is clicked  Parameters: grid widget , column object ,evt  Return none



# Utility Methods

This section covers some of the utility methods provided by the iDevUI to help with development.

accessed via the *idev.utils* object.

### methods

#### *ajax*

Basic AJAX call.

Parameters: config

```
config
{
  url:'ajax.php',
  dataType:'json' or 'text'
  params: {
    command:'myCommand'
  }
  success: function(response, textStatus, jqXHR)
  {
    // Some code
  },
  error: function(errorText)
  {
    // Error handling code
  },
  timeout:30000
}
```

See [jQuery ajax](#) for more details.

#### *loader*

Dynamically loads additional script into your application.

Parameters: url,callback,params

url: url to script you want to load

callback: function to be call after load

params: any parameters you want to pass to the url (in JSON format)

#### *json*

Return a JSON object from a string

Parameters: jsonString

Return jsonObject

<i>eval</i>	Return evaluation string as a result.  Parameters: string, onerror function
<i>cssOpacity</i>	Returns the cross browser CSS for changing the opacity of a HTML element  Parameters: opacity (0 - 1.0)  Return CSS styling
<i>showBusy</i>	Shows an animated busy graphic
<i>hideBusy</i>	Hides the animated busy graphic
<i>delay</i>	Allows you to setup a function call after X milliseconds  Parameters duration,callback,scope  duration: delay in milliseconds callback: function called after time-out scope: variable passed to callback
<i>replaceAll</i>	Replaces a sub string within a string  Parameters string,substr,replacestr  Return new string
<i>round</i>	Returns a number rounded to a specific number of decimal places  Parameters: number, decimals
<i>roundUp</i>	
<i>ucwords</i>	Returns a string the first letter of each word has been capitalised  Parameters string
<i>dateAdd</i>	Adds a date period to a date object  Parameters: objDate, strInterval, intIncrement  where strInterval can be...  "M" - month "D" - day "Y" - year "h" - hour "m" - minute
<i>fromUDateTime</i>	Converts a string in the format YYYYMMDDHHMMSS to a date object
<i>trim</i>	Returns a string trimmed of leading and trailing spaces.
<i>ltrim</i>	Returns a string trimmed of leading spaces.

<i>rtrim</i>	Returns a string trimmed of trailing spaces.
<i>urldecode</i>	return a string url decoded.  ie %20 is replaces by a space
<i>urlencode</i>	return a string url encoded.  ie a space is replaces by %20
<i>JSON2string</i>	Returns a JSON object as a string  Parameters objJSON  Return string
<i>mousePosition</i>	Returns the mouse position from a HTML event object  Parameters event  Return { x: 100, y: 100 }
<i>clone</i>	Create new JSON object as a copy of one you pass.  Parameters objJSON  Returns objJSON copy
<i>generatePassword</i>	Returns unique password based on the parameters passed.  Parameters: length, strength  Where:  length is the number of characters in password.  strength is a bitwise value to set options  1 = Add upper case characters 2 = Add vowels 4 = Add numbers 8 = Add special characters  Therefore a value of 10 would mean add vowels and specials
<i>parseJSON</i>	Returns a string in JSON format as a JSON object.
<i>guid</i>	Returns a GUID unique string.
<i>extractIndex</i>	Used to extract the index number of a target id with a click event occurs on a grid or list widget.  Parameters: target object from event object  Returns: -1 (no index) or the indexed number.
<i>loadCSS</i>	Loads a style sheet dynamically

	Parameters: css src, callback
<i>downloadURL</i>	Downloads the file at the specified url without creating a leave window alert Parameters: url
<i>scrypt</i>	Creates a searchable encrypted/decrypted text. Parameters: text, key, encrypt return: encrypted or decrypted text  This is not meant to be a strong cipher but can be used to make local stored data unreadable outside your app.  The function uses a random alphabet and a number key between 0 and 25. It was designed so you could store data but still search within it. (ONLY TEXT)  The random alphabet is generated upon loading of iDevUI and store in the idev property idev.rAlphabet  You may choose to store this on a server so you can retrieve at a later date for decryption
<i>switchTheme</i>	Allows the application theme to be switch dynamically Parameters: theme, callback  theme = the name of a loaded them under the iDevUI theme folder callback = function to be call once the new them has been loaded
<i>updateJSONObject</i>	Updates one JSON object with another (or array) where the names match Parameters: json, data  json = JSON object to be updated



# Macros

Quick function calls to help development time.

### Methods

<i>\$debug</i>	Echo's a message to the JavaScript debug console.  Parameter: variable or text
<i>\$msg</i>	Shows a simple message box.  Parameters: string message
<i>\$popup</i>	Popup's up a message for a given time  Parameters: t,i,x,y,a,ax,ay  Where:  t: text i: icon name ("error", "warning", "info") x: left start position of message box y: top start position of message box a: animation time ax: left end position of message box ay: top end position of message box
<i>\$error</i>	Shows an error message box.  Parameters: string message
<i>\$warning</i>	Shows a warning message box.  Parameters: string message
<i>\$yesno</i>	Shows a message box with yes/no buttons.  Parameters: string message, callback  The callback is passed a string parameter with "YES" or "NO"
<i>\$delay</i>	Performs the same function as the <code>idev.utils.delay</code> method  Parameters: timeout, callback function, scope
<i>\$get</i>	Returns a give widget object by id

	Parameters: widget id
<i>\$tr</i>	<p>Translates the passed text to the given load language.</p> <p>Parameters text</p> <p>Returns translated text or original text</p>
<i>\$ec</i>	<p>Encrypt the passed text based on two idev object properties</p> <p>rAlphabet &amp; rkey (default 13)</p> <p>You may overwrite these properties to perform your own encryption</p> <p>Parameters: text</p> <p>Returns: encrypted text</p>
<i>\$dc</i>	<p>Decrypt the passed encrypted text</p> <p>Parameters: text</p> <p>Returns: decrypted text</p>
<i>\$round</i>	<p>Rounds a number to the nearest decimal place</p> <p>Parameters: number, precision</p> <p>Returns: rounded number</p>
<i>\$int</i>	<p>Rounds a number up to the nearest whole number</p> <p>Parameters: number</p> <p>Returns: rounder number</p>



Within idevUI there is a set of icon available for buttons and listbox items.



temp, thunder, snow, hail, rain, cloudy, sun, merge, split, shuffle, refresh, smile2, smile, alarm, clock,

tshirt, page, page2, plugin, svg, bookmark, hammer, users, user, mail, picture, bubble, home, lock, clip

cloudUp, location, volume0, volume1, volume2, volume3, key, ruler, power, unlock, flag, tag, search, zoomout, zoomin

nomagnet, flip, flipv, connect, disconnect, folder, man, woman, notebook, diagram,  
barchart, piechart, linechart

Each icon is 20x20 you can use the scale property to change size or use css styling.



# How To Deploy

Once you have created your application all you need to do is upload it to a folder on your server and invoke the index.html.

<http://www.mysite.com/firstapp/index.html>.

If you call the above URL from a smartphone (iPhone/ Android etc) you can then book mark the application to the phones home screen.

Alternatively you can join PhoneGap and use its packaging service to embed your application in a native app.

<http://phonegap.com>

You need to subscribe to PhoneGap Build or download the libraries and build locally.

If you use the PhoneGap Build, you can simply zip your iDevUI application, upload and PhoneGap will generate the platform apps.

PhoneGap also gives your app access to other device features like the camera, but you need to include the phonegap.js (provided) in your preferences.js



# UX Widgets

User eXtension widgets are the widgets that are optional and are included by editing the application preferences.

## ***The Widgets***

Pictures	A widget for fading in and out a series of images
Date Picker	A widget for date selection
Gauge	A simple gauge widget
Map	A widget display Google maps
Rich Text	A widget for entering formatted text. With toolbar
Signature	A widget for capturing a signature or drawing
Tree View	A widget being a standard tree view
Uploader	A widget for uploading files

All widgets support properties, events and functions. Some require the definition of sub widgets like the list box and knowledge of HTML

---

Created with the Personal Edition of HelpNDoc: [Easily create EPub books](#)

---

## Pictures



# Pictures

A UX pictures widget creates a rolling sequence of images for the presenting of products and services etc. These images fade in and out as they swap over.

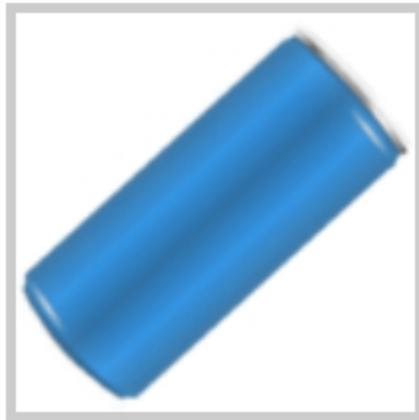
```
{  
    wtype: 'pictures',  
    x: 220,  
    y: 5,
```

```

width:220,
height:280,
animSpeed:2000,
pictures: [
    { src:'images/advert.jpg' }
],
events: {
}
}

```

When using a picture widget you can supply an array of pictures or a dataStore with links to the image



### options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>animSpeed</i>	Speed in milliseconds for fading
<i>delaySpeed</i>	Speed in milliseconds for switch images
<i>pictures</i>	A collect of image details including the url to the image and a link to the name of the page the user will be sent to if the click on it.
<i>ds</i>	Data store object (see Data Store section)
<i>imageField</i>	Field in the dataStore that contains the url of the image

### events

<i>afterRender</i>	Fired after the picture widget has been rendered.  Parameters: widget
<i>click</i>	Fired when a image is clicked  Parameters pictureWidget,index

## method

<i>pause</i>	Pause animation
<i>run</i>	Start animation

---

Created with the Personal Edition of HelpNDoc: [Produce electronic books easily](#)

---

## Date Picker



# Date Picker

A date picker widget.

```
{  
    wtype: 'datepicker',  
    x: 220,  
    y: 5,  
    width: 220,  
    height: 280,  
    events: {  
  
    }  
}
```



## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.

<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>inputHeight</i>	Height of date input element
<i>value</i>	Initial data value
<i>format</i>	Format to use when displaying data in input (mode 2)
<i>mode</i>	<p>The date picker widget has 2 modes (default 1)</p> <ol style="list-style-type: none"> <li>1. Open calendar, show the month in full</li> <li>2. An input box with trigger button to show calendar</li> </ol>
<i>iconColor</i>	Colour of icon on trigger button (default #000)
<i>colorScheme</i>	<p>Calendar colour scheme</p> <p>aqua armygreen bananasplit beige deepblue greenish lightgreen ocean_blue orange (default) peppermint pink purple torquoise</p>

## **events**

<i>click</i>	<p>Fired when date changes.</p> <p>Parameters: date picker widget</p>
--------------	---

## **method**

<i>getValue</i>	<p>Returns the selected date in JSON format.</p> <p>Returns...</p> <pre>{   year:2012,   month:1,   day:1 }</pre>
<i>getDate</i>	Returns the selected date as a standard date object

<i>getValue2String</i>	Returns the selected date in string format  String is format is "yyyymmdd"
<i>setValue</i>	Sets the widget to a particular date.  Parameters: date  Where date = { year:2012, month:1 day:1 }
<i>setDate</i>	Set the widget date.  Parameters: date  Where date is a standard data object
<i>setDateFromString</i>	Set the date from a string parameter  Parameters: sDate  Where  sDate : yyyymmdd

---

Created with the Personal Edition of HelpNDoc: [Easy EBook and documentation generator](#)

---

## Gauge



# Gauge

A gauge widget.

```
{
    wtype: 'gauge',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    events: {

    }
}
```

Draws a simple gauge.



## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>startcolor</i>	Colour of first arc section of gauge (default #33FF33)
<i>centercolor</i>	Colour of centre arc section of gauge (default #FFCC00)
<i>endcolor</i>	Colour of end arc section of gauge (default #FF0000)
<i>value</i>	Initial value (between 1 - 100)
<i>title</i>	Title to display on gauge

## events

<i>click</i>	Fired when gauge is clicked.  Parameters: gauge widget
--------------	--

## methods

<i>getValue</i>	Returns the current gauge value
<i>setValue</i>	Sets the gauge value  Parameters value  Value must be between 1 and 100

Created with the Personal Edition of HelpNDoc: [Free HTML Help documentation generator](#)

## Map



# Map

UX widget to add a google map widget.

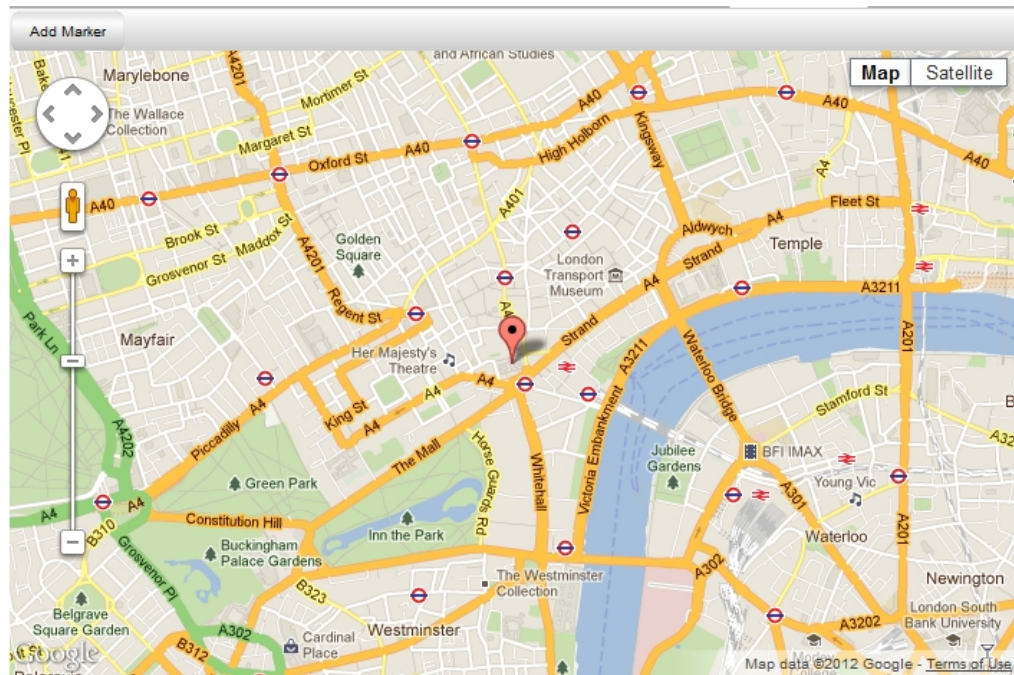


```

{
  wtype: 'map',
  x:220,
  y:5,
  width:220,
  height:280,
  location:'',
  events: {

  }
}

```



## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>location</i>	GEO location, place name, postal code or longitude and latitude (comma separated).

## events

<i>afterRender</i>	Fired after the map widget has been rendered.  Parameters: map widget
--------------------	---

## methods

<i>setCenter</i>	Set the centre of the map to a given place name, postal code or longitude and latitude.
------------------	---

	Parameters: location Returns: none
<i>addMarker</i>	Adds a marker to the map. Parameters: location,title Returns: none
<i>removeMarker</i>	Removes a marker from the map. Parameters: index (starts at zero) Returns: none

---

Created with the Personal Edition of HelpNDoc: [Free help authoring environment](#)

---

## Rich Text

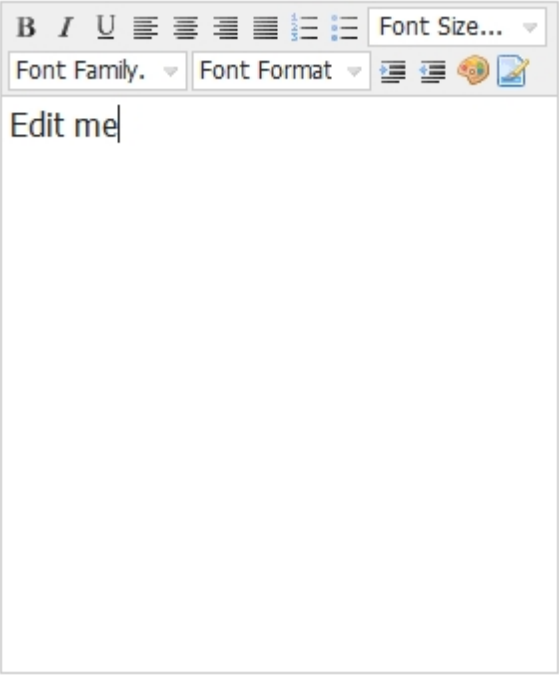


# Rich Text

A rich text widget. To create formatted text.

```
{
    wtype:'richtext',
    x:220,
    y:5,
    width:220,
    height:280,
    value:'Edit me',
    events: {

    }
}
```



**options**

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>value</i>	Initial text value

**events**

<i>afterRender</i>	Fired after the widget has been created.  Parameters: widget
--------------------	--

**methods**

<i>getValue</i>	Returns the current text
<i>setValue</i>	Sets the widget text

**Signature**



# Signature

A signature widget.

```
{
    wtype: 'signature',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    events: {
    }
}
```



## options

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>value</i>	Initial signature value
<i>displayOnly</i>	Make the widget display only (default false)
<i>bgColor</i>	Set background colour (default #FFFFFF)
<i>penColor</i>	Set colour of drawing pen (default #145394)
<i>lineColor</i>	Set colour of signature guide line (default #CCCCCC)
<i>lineWidth</i>	Set width of guide line (default 2)
<i>lineMargin</i>	Set the margin at each end of the guide line (default 2)

<i>lineTop</i>	Position of guide line (default 35)
<i>style</i>	CSS styling for core widget

## events

<i>afterRender</i>	Fired after the widget has been created.  Parameters: widget
<i>click</i>	Fired when the widget is clicked  Parameters: widget  Return: none

## methods

<i>clear</i>	Clear the signature  Parameters: widget
<i>getValue</i>	Returns the signature as a string
<i>getImage</i>	Returns the signature as a string that can be the src of a HTML image element
<i>setValue</i>	Sets a new signature  Parameters: signature (must be string)

---

Created with the Personal Edition of HelpNDoc: [Free help authoring environment](#)

---

## Treeview



# Treeview

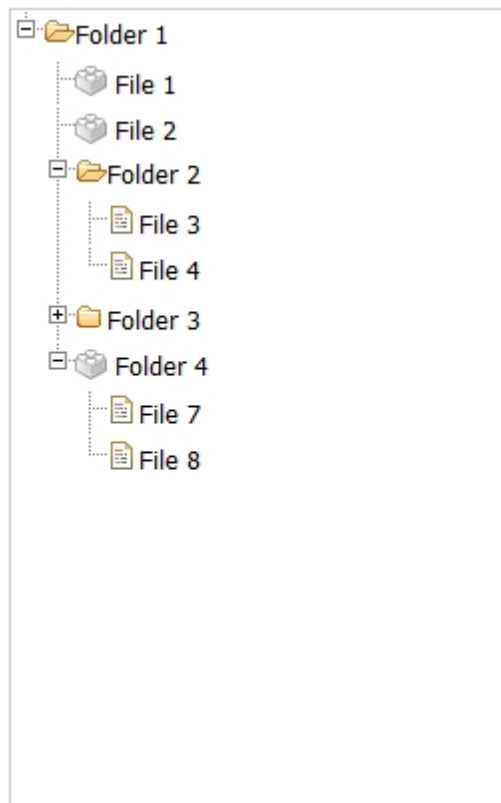
A treeview widget.

```
{
    wtype: 'treeview',
    x: 220,
    y: 5,
    width: 220,
    height: 280,
    style: 'border: 1px solid #ccc;',
    treeCls: 'filetree',
    collapsed: false,
    animated: "fast",
    persist: true,
    root: {
        text: 'Folder 1',
```

```

        cls:'folder',
        children:[
            { text:'File 1', cls:'file', icon:_preferences.imagepath
+'icons/brick.png' },
            { text:'File 2', cls:'brick' },
            {
                text:'Folder 2',
                expanded:false,
                cls:'folder',
                children:[
                    { text:'File 3', cls:'file' },
                    { text:'File 4', cls:'file' }
                ]
            },
            {
                text:'Folder 3',
                expanded:false,
                cls:'folder',
                children:[
                    { text:'File 3', cls:'file' },
                    { text:'File 4', cls:'file' }
                ]
            }
        ],
        events: {
    }
}

```



## options

id	User defined id for widget, if not supplied one will be assigned.
----	---

<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>style</i>	CSS styling applied to core widget
<i>treeCls</i>	CSS class to control the expandable elements of the treeview "filetree" etc (see treeview.css for examples and create your own)
<i>root</i>	A JSON definition that defines the tree structure.
<i>collapsed</i>	Shows the tree initially collapsed (default false)
<i>persist</i>	This controls whether the expanded branched are remembered for next time
<i>cookieId</i>	If you have more than one treeview you will need to provide a cookieId so they are remembered separately like "mytree1"
<i>animated</i>	Determines if the opening and closing of a branch is animated (default false)

## events

<i>afterRender</i>	Fired after the widget has been created.  Parameters: widget
<i>toggle</i>	Fired whenever a branch is expanded or collapsed  Parameters: widget,id,index,text,expanding
<i>click</i>	Fired whenever a leaf is clicked  Parameters: widget,id,index,text
<i>dblclick</i>	Fired whenever a leaf is double clicked  Parameters: widget,id,index,text

## method

<i>set</i>	Sets a completely new tree structure  Parameters: nodes  nodes: can be a string in JSON format or a JSON object
<i>getRoot</i>	Returns the root tree object
<i>getChild</i>	Returns a child node of the tree by index (zero start)  Parameters: node, index  Returns treenode

<i>getNodeId</i>	<p>Returns a tree node id</p> <p>All tree nodes have a unique id which is made up of widget.id + "-node-" + nodeCount;</p>
<i>findNodeById</i>	<p>Finds a tree child node by id</p> <p>Parameters: node,id</p> <p>Return node</p>
<i>find</i>	<p>Find a tree node by text</p> <p>Parameters: node,text</p> <p>Return node</p>
<i>add</i>	<p>Add a new branch to the tree</p> <p>Parameters: node,nodes</p> <p>node: tree node to add new nodes to nodes: JSON object of nodes</p>
<i>remove</i>	<p>Remove a node</p> <p>Parameters: node</p>
<i>clear</i>	<p>Clear the complete tree</p>
<i>getText</i>	<p>Return the text of a node</p> <p>Parameters: node</p>
<i>setText</i>	<p>Set the text of a node</p> <p>Parameters: node, text</p>
<i>toString</i>	<p>Returns the complete tree structure as a string in JSON format</p>

---

Created with the Personal Edition of HelpNDoc: [Full-featured multi-format Help generator](#)

---

## Uploader



# Uploader

A uploader widget.

```
{
    wtype: 'uploader',
```



```

        x:220,
        y:5,
        width:220,
        height:280,
        url:'upload.php',
        params: {
            acc:'tw0001',
            number:'ACC001'
        },
        events: {

        }
    }
}

```

Displays a button that will allow uploading of files

### **options**

<i>id</i>	User defined id for widget, if not supplied one will be assigned.
<i>x</i>	Left position on page.
<i>y</i>	Top position on page.
<i>width</i>	Width of widget.
<i>height</i>	Height of widget.
<i>text</i>	Text to be shown on the upload button (default 'Upload')
<i>url</i>	URL the file will be posted to
<i>params</i>	Parameters to be posted with file
<i>image</i>	URL of image to be used on the upload button.

### **events**

<i>afterRender</i>	Fired after the widget has been created.  Parameters: widget
<i>change</i>	Fired when file name has changed.  Parameters: widget

### **methods**

<i>getValue</i>	Return full file name and path
<i>setParam</i>	Sets a parameter
<i>clearParams</i>	Clear all parameters
<i>upload</i>	Execute the upload
<i>getFilename</i>	Return only the file name section



# Local Services

This section covers some of the features available locally on the target device.

- Local Storage
- Local Database
- GEO Positioning

---

Created with the Personal Edition of HelpNDoc: [Easily create HTML Help documents](#)

---

## Local Database



# Local Database

With modern browsers that support HTML5 a developer has the opportunity to store information in a local SQLite database.

To provide access to this service iDevUI provides the ***iddev.local*** object with methods designed to help the developer.

### ***methods***

*dbSupport*

This checks that local database storage is supported by the browser.

Return true or false

*dbOpen*

Opens a local database

Parameters: dbName, dbSize, dbVersion

dbName = the database name

dbSize = the max database size in bytes (optional) default 100k

dbVersion = SQLite version you want to use (optional) default "version 1.0)

	Returns a database object to be used with <i>dbExecute</i>
<i>dbExecute</i>	<p>Executes an SQL statement on a database.</p> <p>Parameters: db,sSQL,args,success,failed</p> <p>db = name to store the data under  sSQL = SQL statement to execute  args = array of values that replace ? characters in your sql.  success = method to call upon success  failed = method to call upon failure</p> <p>No return this is a asynchronous operation</p> <p>example:</p> <pre>idev.local.dbExecute(db,"select * from tbNames",function(tx,rx) {     // Success function     for(var i=0; i&lt;rs.rows.length; i++)     {         var row =rs.rows.item(i)          \$debug(row['fdName']);     } }, function(tx,error) {     // Failed function     \$debug(error.message); });</pre> <p>The args parameter is good for insert statements</p> <pre>INSERT INTO todo(todo, added_on) VALUES (?,?)</pre> <p>where the args[0] replaces the first ? and so on.</p>

## GEO Location



# GEO Location

With modern browsers that support HTML5 a developer has the opportunity to get the GEO positioning of the mobile device.

To provide access to this service iDevUI provides the ***idev.local*** object with methods designed to help the developer.

Note: Some desktop browsers also support the feature via the WiFi hub.

## methods

<i>currentLocation</i>	<p>This obtains the devices current location. The first time this is used the user will be asked if they want their location determined</p> <p>Parameters: success,failed</p> <p>Both the above parameters are optional and if not passed the built-in idev function is used. In which case you can then simply use the idev,<i>geoposition</i> property.</p> <pre>success : function(position) {     \$debug(position.coords.latitude);     \$debug(position.coords.longitude); }  failed : function(error) {     \$debug(error.code);     switch(error.code)     {         case error.TIMEOUT:             alert ('Timeout');             break;         case error.POSITION_UNAVAILABLE:             alert ('Position unavailable');             break;         case error.PERMISSION_DENIED:             alert ('Permission denied');             break;         case error.UNKNOWN_ERROR:             alert ('Unknown error');             break;     } }</pre> <p>No return</p>
------------------------	---

---

Created with the Personal Edition of HelpNDoc: [Full-featured Help generator](#)

---

## Local Storage



# Local Storage

With modern browsers that support HTML5 a developer has the opportunity to store information at a local level. Where this data can be retrieved from the next time the application is run.

To provide access to this service iDevUI provides the ***idev.local*** object with methods designed to help the developer.

Note: *The typical limit of this type of local storage is 5MB but it is dependant on the browser*

## ***methods***

<i>localSupport</i>	This checks that local storage is supported by the browser.  Return true or false
<i>storeLocal</i>	Allows data to be store against a key word.  Parameters: key,value  key = name to store the data under value = the data value you wish to store  Return true or false
<i>removeLocal</i>	This removes previously store data  Parameters: key  key = name of the stored data to remove  Return true or false
<i>countLocal</i>	Returns the number of local data values that have been stored
<i>clearLocal</i>	Clears all stored data
<i>getLocal</i>	Gets the previously store data.  Parameters: key,default  Returns the value stores or the default value



# Advanced Topics

This section covers some of the more advanced topics of iDevUI.

We recommend you read this section once you are comfortable with the fundamentals.

---

Created with the Personal Edition of HelpNDoc: [Easily create PDF Help documents](#)

---

## Using CSS3



# Using CSS3

If you want to use CSS3 style for the buttons you can set the `_preferences.useCSS3` property to true (default is false).

This means that iDevUI will create buttons based on CSS3 style classes to be found in the `theme.css` stylesheet.

The base class is...

```
.ui-button {  
    -webkit-border-radius: 6px;  
    -moz-border-radius: 6px;  
    border-radius: 6px;  
    font-family:tahoma;  
}
```

This should only need be changed for the button corner radius.

On top of the base class the framework will apply the default button styling

```
.ui-button-body {  
    ○  
    ○  
    ○  
}
```

or

```
.ui-button-body-???? {  
    ○  
    ○  
    ○  
}
```

where ???? is the colour you have select in the button properties.

---

Created with the Personal Edition of HelpNDoc: [Easy EBook and documentation generator](#)

---

## Page Animation



# Page Animation

Currently iDevUI supports two page animations

"fade"  
"slide"

or you can switch it off completely

To control the animation you need to change the follow idev properties:

idev.pageManager.animation

idev.pageManager.animationType

idev.fadeOutTime (in milliseconds default 1000)

idev.fadeInTime (in milliseconds default 1000)

idev.animationTime (used when sliding in milliseconds default 500)

*Note: Animation will apply a time lag between pages.*

---

Created with the Personal Edition of HelpNDoc: [Write eBooks for the Kindle](#)

---

## Languages



# Languages

iDevUI support a language mechanism

Using the preferences file you can set the language you want to use.

This will force the framework to load the language.js file ( you have to create the file yourself in line with your application)

Language file format:

```
_language.words = {  
  "data view": "voir les données",  
  "general": "générale",  
  "buttons": "des boutons",  
  "charts": "graphiques",  
  "form": "sous forme",  
  "grid": "la grille",  
  "map": "plan",  
  "iframe": "iframe",  
  "apples": "pommes",  
  "value": "valeur",  
  "image": "l'image",  
  "search": "Recherche",  
  "cancel": "Annuler",  
  "title": "titre",  
  "welcome": "bienvenue"  
}
```

You will notice this is in JSON format where the key is the English word and the value is new language word

Once this is loaded you may use the \$tr macro to convert words

\$tr("charts") would return "graphiques".

---

Created with the Personal Edition of HelpNDoc: [Free Kindle producer](#)

---

## Special Features



# Special Features

This section covers some of the less obvious features within iDevUI.

```
">>"
```

Use this text as a widget in a toolbar or statusbar as a spacer so all widgets after it are moved over to the right

example:

```
widgets:[  
  {  
    wtype:'button',  
    text:'Button A'  
  },  
]
```



```

">>",
{
    wtype: 'button'
    text: 'Button B'
}
]

```

In the above example button B would be moved to the right of the toolbar

## Date Formatting

As part of the iDevUI the basic JavaScript date object has been extended to include a date format function

```
var sTextDate = new Date().format("ddd dd mmm yyyy");
```

Output: Mon 26 Mar 2012

Available format characters

yyyy	Full 4 digit year
mm	Month 01 - 12
mmm	Short month jan,feb etc
mmmm	Full month January,February etc
hh	Hour 00-23
mm	Minute 00-59
ss	Second 00-59
TT	AM/PM
z	Timezone

dateFormat.masks

shortDate:	"m/d/yy",
mediumDate:	"mmm d, yyyy",
longDate:	"mmmm d, yyyy",
fullDate:	"dddd, mmmm d, yyyy",
shortTime:	"h:MM TT",
mediumTime:	"h:MM:ss TT",
longTime:	"h:MM:ss TT Z",
isoDate:	"yyyy-mm-dd",
isoTime:	"HH:MM:ss",
isoDateTime:	"yyyy-mm-dd'T'HH:MM:ss",
isoUtcDateTime:	"UTC:yyyy-mm-dd'T'HH:MM:ss'Z'",
uDateTime:	"yyyymmddHHMMss"

Usage:

```
var sTextDate =new Date().format(dateFormat.masks.mediumDate);
```



# Themes

This section covers the topic of CSS and themes.

## **Base CSS**

IDeVUI has a base CSS stylesheet "idevui.css" which can be found in the root framework folder. The file has the default styling for a number of the key widgets.

All base classes start with the word ui-??????

## **Themes**

There are two base themes that come with the download:

- Dark
- Default

Which theme your applications uses is controlled by your applications preferences files.

```
_preferences = {  
  
theme: 'default',  
  
}
```

In the above example the theme property has the name 'default' but you can specify your own application specific theme.

```
_preferences = {  
  
theme: '../css/theme.css',  
  
}
```

To create own theme all you need to do create your version of the .ui-????? class in your css file.

### **default:**

```
.ui-panel-tbar {  
    background: #e0e0e0;  
    border-top: 1px solid #ccc;  
    background-image: -moz-linear-gradient(top, #fefefe,  
#cccccc);  
    background-image: -webkit-gradient(linear, left top, left  
bottom, color-stop(0, #fefefe), color-stop(1, #cccccc));  
    background: -ms-linear-gradient(top, '#fefefe' 0%  
, '#fefefe' 31%, '#cccccc' 100%); /* IE10+ */  
}
```

```

        background: -o-linear-gradient(top, '#fefefe' 0%, '#fefefe'
31%, '#cccccc' 100%); /* Opera11.10+ */
        filter: progid:DXImageTransform.Microsoft.gradient
( startColorstr='#fdfdfd',
endColorstr='#dddddd',GradientType=0 ); /* IE6-9 */
    }

```

#### **my css:**

```

.ui-panel-tbar {
    background: rgb(254,254,254); /* Old browsers */
    background: -moz-linear-gradient(top,  rgba(254,254,254,1)
0%,  rgba(30,87,153,1) 58%,  rgba(204,204,204,1) 100%); /* FF3.6
+ */
    background: -webkit-gradient(linear, left top, left
bottom, color-stop(0%,rgba(254,254,254,1)), color-stop(58%
,rgba(30,87,153,1)), color-
                                stop(100%,rgba
(204,204,204,1))); /* Chrome,Safari4+ */
    background: -webkit-linear-gradient(top,  rgba
(254,254,254,1) 0%,rgba(30,87,153,1) 58%,rgba(204,204,204,1)
100%); /* Chrome10+,Safari5.1+ */
    background: -o-linear-gradient(top,  rgba(254,254,254,1)
0%,rgba(30,87,153,1) 58%,rgba(204,204,204,1) 100%); /* Opera
11.10+ */
    background: -ms-linear-gradient(top,  rgba(254,254,254,1)
0%,rgba(30,87,153,1) 58%,rgba(204,204,204,1) 100%); /* IE10+
*/
    background: linear-gradient(to bottom,  rgba
(254,254,254,1) 0%,rgba(30,87,153,1) 58%,rgba(204,204,204,1)
100%); /* W3C */
    filter: progid:DXImageTransform.Microsoft.gradient
( startColorstr='#fefefe',
endColorstr='#cccccc',GradientType=0 ); /* IE6-8 */
}

```


Once you have created your own theme, create a folder under themes of the name of your theme and copy the css file to it. Make sure you call it "theme.css" and then you can reference it by name in your preferences file.

Note: Make sure you include any graphics required by your theme in your theme folder.



# Getting Started With iDev Studio

**Starting:** To get started follow the steps below:

- Log into your copy of iDev
- The open screen will show your projects, we are going to add a new mobile project
- Select "Add New Project" on the toolbar 
- Enter in your project name and double click on either a desktop, tablet or mobile app icon to create your project. It's that simple.



- You should see a new icon in your list with the name you have just entered



- Double click the icon and your are ready to go.
- Once the new project is loaded you can click "Preview" on the top toolbar to see the new application.



Obviously not very exciting at this stage but that's where you come in.

## Where to start

The first place to start is to understand the structure of the application and then look at the available UI widgets.

Now to coding, you will need to select the scripts tab to view the code. Under scripts there are three secondary tabs to cover both the client and server side scripting.

We are going to focus on the client and the main web applications. Click the "Client Side" tab.

Here you will see a JavaScript file call app.js, double click to load it in for editing and you will instantly see we use JSON to define the pages.

(see [Starting Coding](#) for basic coding)

Then click the preview icon (top right) to see any changes.

---

Created with the Personal Edition of HelpNDoc: [Full-featured multi-format Help generator](#)

---

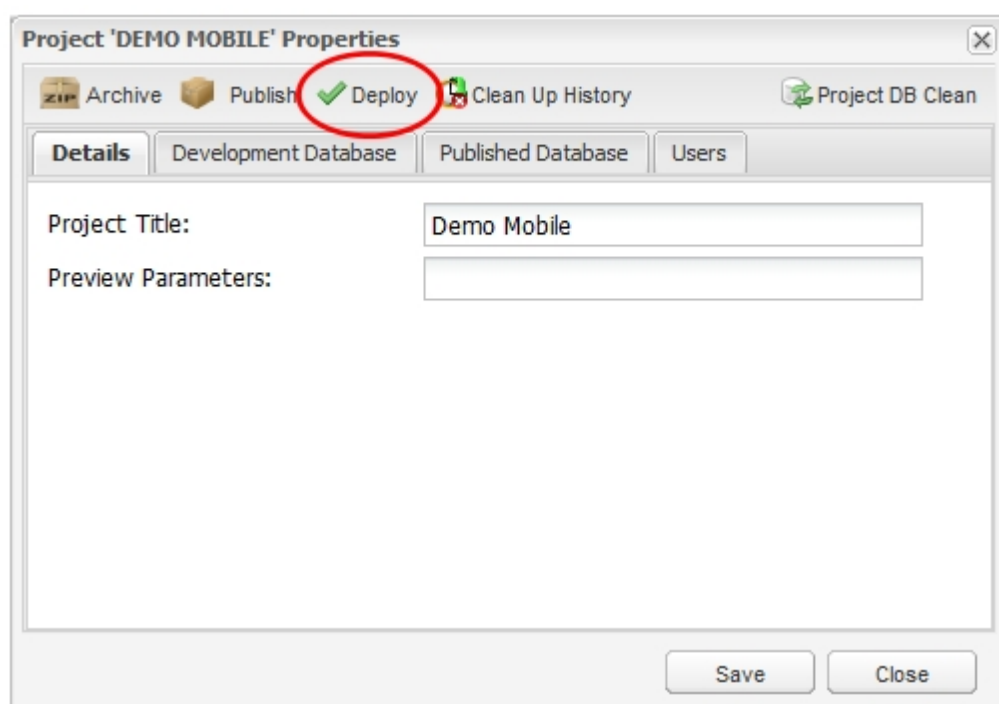
## iDev Studio Deployment



# How To Deploy From iDev Studio

To deploy your application you need to follow the steps below.

1. Select the projects tab.
2. Right click your project and select project properties.
3. On the toolbar of the properties dialog you will see a tick and the word "Deploy"



This will generate a zip file containing the files and folders for your application

You can then upload them to your web server.



# Acknowledgement

We would like to acknowledge the efforts of the other open source software that iDevUI has made use of

JQuery	<a href="http://www.jquery.com">http://www.jquery.com</a>	Basic HTML control
iScroll	<a href="http://cubiq.org/iscroll">http://cubiq.org/iscroll</a>	Touch scrolling
Flot	<a href="http://code.google.com/p/flot/">http://code.google.com/p/flot/</a>	Charting
John Resig	<a href="http://www.ejohn.org">http://www.ejohn.org</a>	Class inheritance
Thomas J Bradley	<a href="http://thomasjbradley.ca/lab/signature-pad">http://thomasjbradley.ca/lab/signature-pad</a>	Signature capabilities
Raphaeljs	<a href="http://dmitrybaranovskiy.github.io/raphael/">http://dmitrybaranovskiy.github.io/raphael/</a>	SVG graphics
Brian Kirchoff	<a href="http://nicedit.com">http://nicedit.com</a>	RichText editing
Jörn Zaefferer	<a href="http://bassistance.de/jquery-plugins/jquery-plugin-treeview">http://bassistance.de/jquery-plugins/jquery-plugin-treeview</a>	Tree view coding
Javascriptcalendar	<a href="http://javascriptcalendar.org">http://javascriptcalendar.org</a>	Date picker
Adam Shaw	<a href="http://arshaw.com/fullcalendar/">http://arshaw.com/fullcalendar/</a>	Full calendar

Please respect their licensing