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 64
- Progress Bar 68
- Radio Button 70
- Slider 71
- SVG Button 72
- Spacer 75
- 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 Sevices 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

Introduction

Welcome to iDev UI

(**i**nternet **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.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

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.

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

Screenshots

Example Designs

Created with the Personal Edition of HelpNDoc: Full-featured Help generator

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.

Created with the Personal Edition of HelpNDoc: Full-featured Documentation generator

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 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.

Created with the Personal Edition of HelpNDoc: Generate EPub eBooks with ease

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
  |
```

|____ index.html

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.

Created with the Personal Edition of HelpNDoc: Full-featured Kindle eBooks generator

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.

Created with the Personal Edition of HelpNDoc: Easy EBook and documentation generator

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

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

	events	A set of event handlers

toolbar

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

pages

See Page Definition section of this document

statusbar

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

events

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

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

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 it's own configuration as well.

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

       },
       content: {

       },
       footer: {

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

options

name	User defined name that can be used when showing different pages by name.
style	CSS styling for page.
cls	CSS class to override the core page styling

fn	A user defined object to extend the base page object with additional methods.
	
	{
	...
	fn: {
	myfunc:function()
	{
	// Code
	}
	}
	...
	}

	This makes this function available from anywhere and any widget

header

height	Height of header (default 100).
style	CSS styling for header.
cls	CSS class to override the core header styling
hidden	true/ false to hide or show header (default is false)
widgets	A collection of widget configurations or objects.
html	Simple text or html to be show in the header

content

height	Height of header (default 100).
style	CSS styling for content.
css	CSS class to override the core content styling
layout	content widget layout, option for "form" which will use the label property of the widgets to render a form layout. (default none)
labelWidth	Layout of a page, this property defines the width allocation to any widget labels when the content layout is "form".
widgets	A collection of widget configurations or objects.
html	Simple text or html to be show in the header

footer

height	Height of header (default 100).
style	CSS styling for footer.
cls	CSS class to override the core footer styling
hidden	true/ false to hide or show header (default is false)
widgets	A collection of widget configurations or objects.
html	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

Created with the Personal Edition of HelpNDoc: Free help authoring environment

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:

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

events	A set of event handers. afterRender: Call once the main application has been created. onOrientation: Call everytime the framework detect a chaning 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.

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

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

properties

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

	a touch screen device. default false;
geoposition	Last GEO lat/long after the idev.local.currentLocation method was called.
geoerror	Last GEO error after the idev.local.currentLocation method was called and an error occurred. If the error exists geoposition will be null.
mousedowntime	A date object set when a mousedown event occurs
mouseuptime	A date object set when a mouseupevent occurs
mousemoved	A flag set to true or false to indicate the mouse moved between a mousedown and a mouseup.

fn	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

dom	This is a direct alternative for the JQuery object. Anything you can so with JQuery you can so via the **idev.dom** 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.
animationTime	Time in milliseconds for page fading or sliding (default 500).
focusHighLight	This places a shadow around the input that has focus (default false).
highlightCSS	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

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

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

	Parameters: id Return: none
show	Show a widget Parameters: id Return: none
getParameter	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
getParameterKey	Get a parameter key name by index number Parameters:index,default Returns parameter name or default
isStandAlone	Returns true or false if the application is being run as a web app on a mobile device.
convertNulls	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
isClass	Test id the object passed is based on the baseWidget class. Parameters:object Returns true or false
hideStatusbar	Hide the main application statusbar
showStatusbar	Show the main application statusbar
addTouchScroll	Adds touch scroll capability to a DIV. Parameters:id,options id = the id of the DIV options = a JSON config { vScroll:true, hScroll:true, bounce:true,

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

Created with the Personal Edition of HelpNDoc: Write EPub books for the iPad

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

add	Adds a new page definition to the application. This does not create the page just adds it to the page collection Parameters: JSON config Return: true
getPage	Returns a page object by name assuming it has been created. Parameters: name Return: page object or null
pageCount	Returns the number of created pages Parameters: count
remove	Removes a page from the page definitions collection and if create destroys the page and all created widgets. This should be followed by a showPage if page being removed is the current page. Parameters: name Return: true or false
showPage	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). Parameters: name or index

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

Created with the Personal Edition of HelpNDoc: Free Kindle producer

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

title	Title to be applied to application document.
blankimage	Some widget use this to create a space or instead of a graphic
rootpath	Path to root of all application files
libpath	Path to iDevUI framework
language	Selected language (see languages section) the default is English (optional)
languagepath	If you intend to use a language you can use this property to set the path
imagepath	Path to application images (these are separate to the framework images)
apppath	Path to main application script file
app	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.
theme	CSS style theme to use (default is "default")
useCSS3	Some browsers do not support SVG for the buttons this allow the framework to use CSS3 styling instead, though you will loose some features
styling	Path to application specific CSS stylesheet
button	A JSON object that set the default styling for all SVG buttons
ux	List of **U**ser e**X**tensions that get loaded automatically by the framework
imports	List of any additional script file the application may need
forcereload	Forces a full reload when reloading page (including cache)
config	Some framework control settings ***charts:*** set to true if you intend to use charts (if false the scripts are not loaded) ***noRightClick:*** Disables the right click of the browser ***fitDocument:*** Forces the framework to fit to the max screen size. ***pageFit:*** 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.

		trapUnload: 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. **unloadText:** Specifies the text to be shown in the trapUnload alert box. If not specified the default text will be shown on the alert.
events		
	beforeLoad	Fired before the dependencies are loaded

Created with the Personal Edition of HelpNDoc: Full-featured multi-format Help generator

Widgets

Widgets are the **U**ser **I**nterface items you use in the layout of your page. They range from simple text t
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 th
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:

addEvent	Added an event listener to a widget Parameter: id,eventName,function
removeEvent	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.

Created with the Personal Edition of HelpNDoc: Produce electronic books easily

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

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 button.
height	Height of button.
text	Button text.
textCls	CSS class to be applied to button text
transparent	Button with no fill color but can have a border.
icon	icon to display on button (JSON object).

	icon: {
	src:'path to image or standard icon name',
	x:10,
	y:10,
	width:32,
	height:32,
	style:'',
	color:'white',
	align:'right'
	}
	
	Where
	
	style: CSS styling to be applied icon
	
	align: When no X or Y position supplied align the icon to the left (default) or right of text on button.
	
	color: When the name of a standard icon is supplied there are two possible colors, black (default) or white.

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

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

events

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

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

```
        height:280,
        style:''
        events: {

        }
}
```

properties

paper	The Raphael paper object. With this you can draw on the canvas using the features found at www.raphaeljs.com

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.
style	CSS styling to be applied to canvas DIV container.
label	Text used when rendered to a form layout.

events

afterRender	Fired after the canvas has been rendered. Parameters: canvas widget Widget includes a paper object which can used the render onto the canvas.

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

Chart

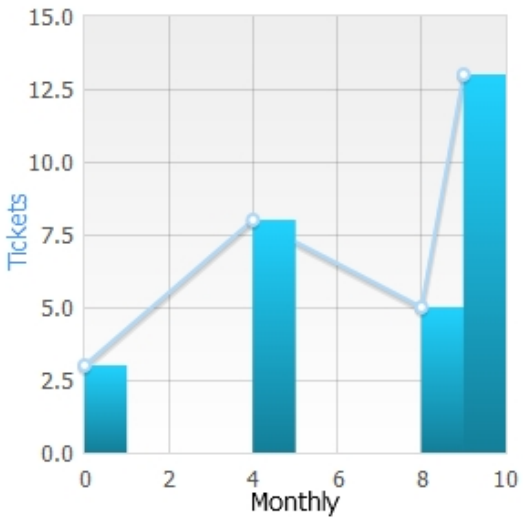
Chart

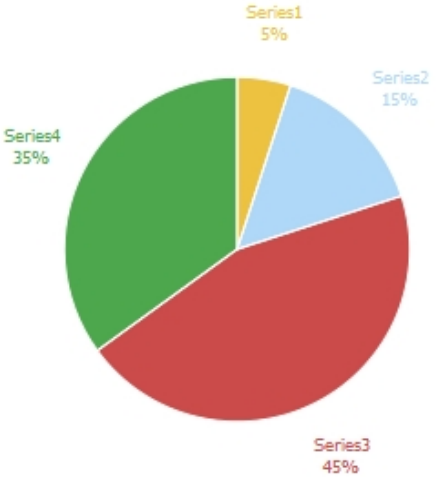
A chart widget.

```
{
        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.

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

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

methods

setSeries	Updates chart with specified data series Parameters: series

events

click	Fired when chart clicked. Parameters: chart widget
dblclick	Fired when chart double clicked. Parameters: chart widget

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

events

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

	Parameters: checkbox widget

Created with the Personal Edition of HelpNDoc: Easily create EPub books

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

		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
methods	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

Created with the Personal Edition of HelpNDoc: Free EBook and documentation generator

Composite

Composite

A composit 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 panel)
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

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.
tpl	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.
data	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.
style	CSS styling for main list container
ds	Data store object (see Data Store section)
dsFilter	Function to call for each record in the DS. Parameters: rec Return true if the rec should be shown
columns	The number columns the data view should show before starting the next row. You can also pass 'auto' and let the widget calculate it
cls	CSS class to override the core widget styling
entryWidth	Width of each cell in the grid, this will determine the number of cells you can get in a row.
entryHeight	Height of each row in the data view

entryCls	CSS class to be applied to each entry in the view
border	true / false as to when to show a border
borderColor	Colour of border if visible (default #ccc).
autoScroll	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
offset	Start position in data store (default 0) for first record.
limit	Max number of entries in the DS to show at any one time
label	Text used when rendered to a form layout.
enabled	This controls whether the click events will function (default true)
roundCorners	Render the panel with round corners
radius	Radius of round corners (default 4)
selectColor	Selection background colour.
selectCls	Selection CSS styling class. You can use this or selectColor or both
autoSelect	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);
renderer	Function to be called for each entry. Parameters widget, sText, record, recno Where 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 Return sText with any changes you may need.

properties

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

events

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

	afterRefresh	Fired just after the dataview refreshes.
	hover	Fired when the mouse enters or leaves a dataview entry Parameters widget, in_out, recno, selected 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
methods	*getStore*	Returns the DataView dataStore object. Parameters: dataview widget,selected,oldSelection,event
	getSelected	Returns the selected ds index. A return of -1 means no selection. This does however require autoSelect set to true.
	refresh	The forces a refresh of the dataview.
	select	Selects a entry in the dataview Parameters: index

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

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

methods

getFieldWidget	Returns the widget object for a given field name Parameters: 'field name'
getFieldConfig	Returns the original widget JSON configuration object for a given field name Parameters: 'field name'
getValues	Returns an array of the current form values where the key value in the array is the field name.
clearValues	Sets of the child widget values to default
setValues	Sets the form widget values to a dataRecord. Parameters: dataRecord If one is not passed then the original dataRec property will be used
setDataRec	Sets the dataRec property to a new dataRecord and up dates all widgets
commit	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.
setEditable	Changes all inputs to editable or not. Parameters: canedit (true/false)

Events All event the same as a standard panel except:

onCommit	Fired after a call to the commit method. Parameters: panel widget
onSetValues	Fired after a call to the setValues method.

	Parameters: panel widget

Created with the Personal Edition of HelpNDoc: Create iPhone web-based documentation

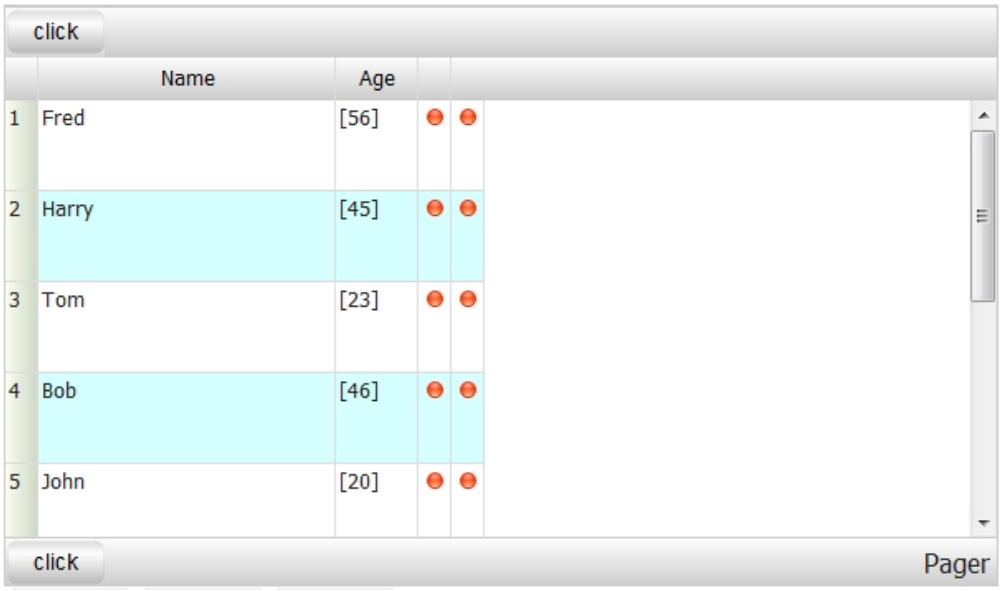
Grid

Grid

A grid widget.

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

	}
}
```



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.

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

events

click	Fired when grid clicked. Parameters: grid widget. event

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

methods

selectRow	Selects a row in the grid Parameters: row number
getStore	Returns the grids Data Store
getSelected	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

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.
icon	Name of icon to use (see built-in icons)
title	Tooltip for the icon to use (shown when you hover the mouse over the image)
color	Icon colour (default #000)
background	True or false to draw a circle behind the icon
backgroundColor	Colour of background circle (default #ddd)
scale	Scaling of icon (0 - 1) default is 0.7 so it can fit on a default toolbar height

events

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

Created with the Personal Edition of HelpNDoc: Easily create HTML Help documents

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

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 iframe.
height	Height of iframe.
src	URL to be displayed in iframe.

events

loaded	Called when the iframe has loaded its content.

methods

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

Created with the Personal Edition of HelpNDoc: Free Kindle producer

Image

Image

A image widget.

```
{
	wtype:'image',
	x:220,
	y:5,
	width:220,
	height:280,
	src:'images/myimage.png',
	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.
src	path or url to image file. You can also use the word "BLANK" to use the _preferences.blankImage url.
style	CSS styling for main image container
imageWidth	Width of image inside widget component
imageHeoght	Height of image inside widget component
title	Set a popup title for image
prehtml	Renders additional html beforer the image. Useful to create text before the image.
html	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

click	Fired when image clicked. Parameters: image widget

methods

setSrc	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: {

      }
}
```

12121

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
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

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.
text	Label text.
label	Text used when rendered to a form layout.
title	When included adds HTML title attribute.

events

click	Fired when label clicked. Parameters: label widget
dblclick	Fired when label double clicked. Parameters: label widget

methods

getText	Returns the label current text.
setText	Set the label text. Parameters stringText
getValue	Same as getText.
setValue	Same as setText.

Created with the Personal Edition of HelpNDoc: Produce Kindle eBooks easily

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

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.
tpl	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.
metaData	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.
style	CSS styling for main list container
roundCorners	Display round corners on list
radius	Radius of round corners.
ds	Data store object (see Data Store section)
dfFilter	Function to call for each record in the DS. Parameters: rec Return true if the rec should be shown
offset	Start position in data store (default 0) for first record.
limit	The max number of entries to include in list.
autoScroll	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

	itemStyle	CSS styling to be used with each list entry.
	titleStyle	CSS styling for list title.
	backgroundStyle	CSS stying to be used on the background to the list.
	autoSelect	Auto show list entry selection
	selectColor	Selection background color.
	selectCls	Selection CSS styling class. You can use this or selectColor or both
	itemCls	Item CSS styling class.
	renderer	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.
	data	Array of custom data you can use in the template.
events		
	click	Fired when the list item is clicked. Parameters: list widget
	dblclick	Fired when the list item is double clicked. Parameters: list widget
	beforeRefresh	Fired just before the list refreshes.
	afterRefresh	Fired just after the list refreshes.
methods		
	getStore	Returns the list data store
	getSelected	Returns the current selected entry (-1 if none selected)
	getItemId	Return the id of the nTh list entry Parameters: index Return id
	updateItem	Update the CSS property of a give list entry. Parameters: index, propertyName, value Return none

setScrollPos	Sets the top position of the list Parameters: pos (pixels) Return none
getScrollPos	Returns the last scroll position set
updateItem	Updates the CSS of a given item in the list Parameters: index,propertyName,value Where propertyName is a standard CSS property
refresh	Used to re-render the list
clearSelection	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

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. Use 'auto' if you want the list height to be controlled by number of items
cls	CSS class for listbox
title	Listbox title to be displayed at the top of the list.
titleHeight	Height of title bar
items	A collection of list box items. See listitem details
background	CSS class for list entry

List Item options:

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

      }
}
```

icon	Name of icon to be displayed on right of list entry
image	Image URL to be displayed on left of list entry
text	List entry text

	textstyle	CSS styling for text
	widget	Optional iDevUI widget to be display after text JSON definition for widget.
	widgetWidth	Width of widget
	itemHeight	Height of item
	selectable	Allow item to be shown selected (default true)
	items	A collect of list box items. See listitem details
	background	CSS class for list entry
events		
	afterRender	Fired after the listbox widget has been rendered. Parameters: listbox widget
	click	Fired when a list entry is clicked Parameters: listbox, index, event
	dblclick	Fired when a list entry is double clicked Parameters: listbox, index, event
methods		
	add	This method will add a new item to the listbox. Parameters: item config (see ListIem for details) Return: new widget object
	get	This method returns a listitem widget base on a index number (starts at 0) Parameters: index Return: new widget object or null

Created with the Personal Edition of HelpNDoc: Easily create Help documents

Object

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 nee to mix IE9 and IE8 content

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 iframe.
height	Height of iframe.
src	URL to be displayed in iframe.
type	Type of content (default text/html)

events

loaded	Called when the iframe has loaded its content.

methods

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

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

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	Allows the panel to collapse to the titlebar and back to full height. If the panel is part a frame the expandability will vary depending the frame area.
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	The layout determines how the panel widgets are laid out ***flow*** (default) : as the widgets are created ***form***: 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 ***table***: widgets are laid out in a table format controlled by the columns property and the rowHeight property ***column***: widgets are laid out in a single table row. ***row***: widgets are laid out in a single table column a row for each widget. ***frame***: 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. ***fit***: 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.

layoutConfig	Additional layout properties.
	
	{
	fit:true.
	}
	
	At present only "column" & "row" use this property with a fit attribute so a column can match the panel height and row can match the panel width.

area	Location of panel when the parent panel is in "frame" layout. options: north, south, west, center and east
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

afterRender	Fired after the widget has been created. Parameters: panel object
minimise	Fired when minimised button clicked. Parameters: panel object
maximise	Fired when maximised button clicked. Parameters: panel object
close	Fired when close button clicked. Parameters: panel object Return false to cancel close

click	Fired when panel clicked. Parameters: panel object
dblclick	Fired when panel double clicked. Parameters: panel object
beforeClose	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.
afterClose	Fired after the widget closes. (Useful for Window widgets) Parameters: panel object

methods

innerHTML	Sets the inner html of the panel. Parameters: html string Returns: none
close	Close panel. This will remove the panel and its widgets from the screen
addWidget	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.
removeWidget	Removes a widget from the panel Parameters: id Returns: none.
setTitle	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

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	Initial value of progress (0 - 1.0)
roundCorners	Render the widget with round corners
radius	Radius of round corners (default 4)
label	Text used when rendered to a form layout

events

click	Fired when progressbar clicked. Parameters: progressbar widget
dblclick	Fired when progressbar double clicked. Parameters: progressbar widget

methods

getValue	Returns current value of progress bar
setValue	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: {

	}
}
```

options

id	User defined id for widget, if not supplied one will be assigned.
x	Left position on page.
Y	Top position on page.
text	Label
value	Radio button value.
group	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

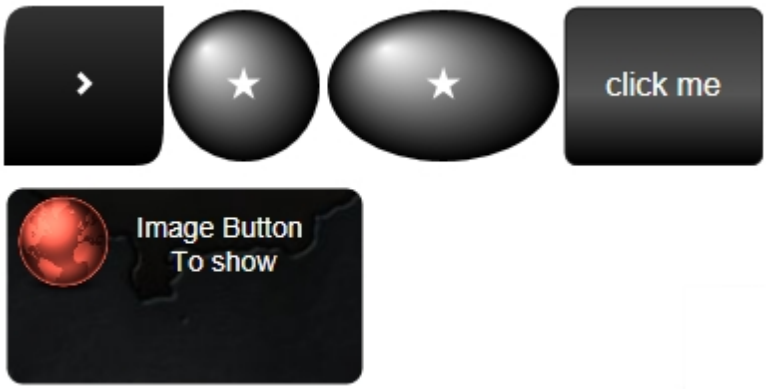
SVG Button

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

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 button.
height	Height of button.
icon	Name of button icon to use (see built-in icons)
iconAlign	Position of icon on button ("left" or "right").
text	Button text.
textXOffset	Offset from button centre for text.
textYOffset	Offset from button centre for text.
textAlign	Button text alignment ("left","center","right").
textVAlign	Button text vertical alignment ("top","center","bottom").
color	Button colour ("blue", "red","green","orange","purple" or "black").
transparent	Button with no fill color but can have a border.
startColor	User defined colour range (ie #000). Works with endColor.
endColor	User defined colour range.
border	Whether to display the border (true/false).
borderColor	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).
	
	image: {
	src:'pathToImage',
	x:10,
	y:10,
	width:32,
	height:32
	}

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"c gradient centre (default is centre of button)
ry	"radial"c 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 textt. Return: button text
textColor	Sets the color of the button text. Parameters: CSS color "ie #fff" Return: none

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

events

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

Created with the Personal Edition of HelpNDoc: Free EBook and documentation generator

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

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

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

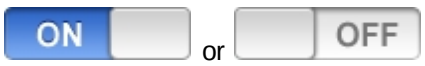
Switch

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

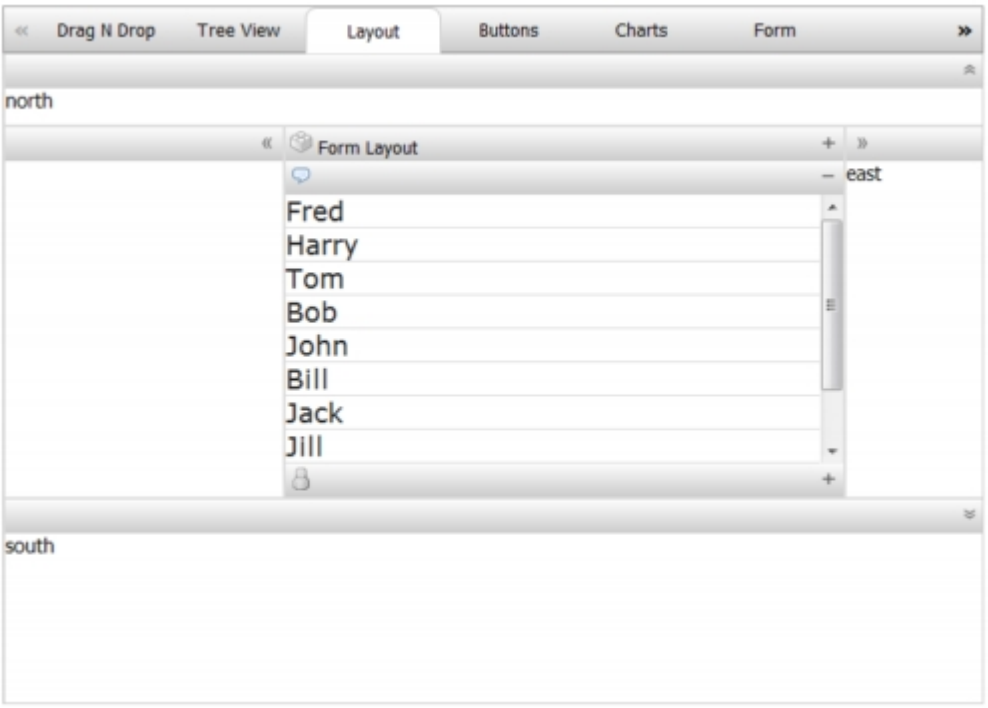
Tab Panel

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

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.
style	CSS styling applied to core widget.
tabWidth	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.
tabHeight	Height of tabs.
tabScrollWidth	Width of scroll control (Default 25).
tabStyle	CSS styling applied to each tab.
activeTab	First tab to display (default 0).

events

showTab	Fired when a tab is shown. Parameters: tab panel widget, active tab

	beforeCloseTab	Fired before a tab is closed. Parameters: tab panel widget, tab index Return false if you want to abort closure.
	afterCloseTab	Fired after a tab is closed Parameters: tab panel widget, tab index Return none
	beforeShowTab	Fired before a tab is about to be shown. Parameters tab panel widget, tab index Return false to cancel show of tab
methods		
	showTab	Show a particular tab. Parameters: tab index Return none
	closeTab	Close a particular tab Parameters: tab index Return none
	addTab	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

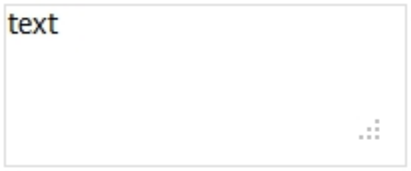
Text Area

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	Fired when the input is clicked Parameters: input Return: none
dblclick	Fired when the input is double clicked Parameters: input 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

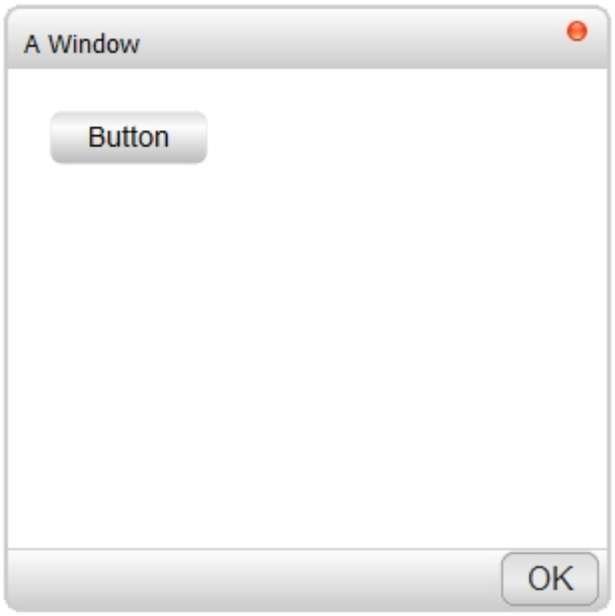
Created with the Personal Edition of HelpNDoc: Easily create EBooks

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();
```



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

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 button.
height	Height of button.
modal	The background widgets are unavailable while the window is showing
roundCorners	Always true
closable	Always true
draggable	Always true
panelCls	By default is "ui-window"
bodyStyle	By default #fff
autoFocus	Sets whether to automatically focus 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).
autoClose	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)

Created with the Personal Edition of HelpNDoc: Free HTML Help documentation generator

Window Form

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.

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

Additional Widgets

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

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.

Created with the Personal Edition of HelpNDoc: Easy EPub and documentation editor

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	A user defined object to extend the widget with additional methods.
	
	Add to config widget prior to creation .
	
	fn has parent property referring back to the widget
	
	{
	...
	fn: {
	myfunc:function()
	{
	this.parent //reference back to the widget
	// Code
	}
	}
	...
	}

anchor	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. Values should be 'left' or 'top' or 'left top' This is applied to all widgets
anchorMargin	A JSON object with the right and bottom offsets { right:10 bottom:20

	} These offsets are taken off the calculated width or height.
disabledCls	specifies a css class to use is enable is set to false

Methods

show	Show a hidden widget.
hide	Hide a visible widget.
fadeOut	Hide the widget by slowly changing its transparency Parameters: animSpeed,easing,callback *animSpeed*: Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast". *easing:* The way it is faded (see easing list) *callback:* Function to call when animation is complete
fadeIn	Show the widget by slowly changing its transparency Parameters: animSpeed,easing,callback *animSpeed*: Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast". *easing:* The way it is faded (see easing list) *callback:* Function to call when animation is complete
fadeToggle	Toggle the transparency of a widget Parameters: animSpeed,easing,callback *animSpeed*: Fade out time in milliseconds (1000 = 1 sec). Can also be the word "slow" or "fast". *easing:* The way it is faded (see easing list) *callback:*

	Function to call when animation is complete
setDraggable	Set where the widget can be dragged around the page Parameters: true/false
isDraggable	Returns true/false as to whether the widget is draggable
isVisible	Returns true/false as to where the widget is visible
isEnabled	Returns true/false if a widget is enabled. Disabling a button prevents the click events.
destroy	Destroys the widget object and removes from page.

getPosition	Return the current top-left of the widget in a JSON object
	
	{
	x:150,
	y:67
	}

getWidth	Returns the width of the widget.
getHeight	Returns the height of the widget.
moveTo	Moves the widget to a specific location on the page. Parameters: x,y,animSpeed,easing,callback) *animSpeed: in milliseconds* *easing:* The way it is faded (see easing list) *callback:* Function to call when animation is complete
moveBy	Same as moveTo but the x & y parameters are relative to the widget current position
animate	Animates a specific CSS properties. Parameters: properties,animSpeed,easing,callback) properties: a JSON object of CSS properties with their target value *animSpeed: in milliseconds* *easing:* The way it is faded (see easing list) *callback:* Function to call when animation is complete

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

fireEvent	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 Parameters event, params event = an event name ('click') params = a parameters object

Events

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

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

Created with the Personal Edition of HelpNDoc: iPhone web sites made easy

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	A JSON object with a collection of parameters to be posted to the above URL.
	
	params:{
	comand:'mycommand'
	}
	
	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:
	
	{"success":true,"results":[{"columnName1":"Row1 - Column1","columnName2":"Row1 - Column2" etc..
	
	In PHP, you can get fields from the database and return the output using your command as a function in your API like so:
	
	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");
	}

	// 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
	$this->pushResult("results",$data);
	}
	
	Note: This function should be put into ajax.php on the server side. (You do not have to use PHP).

autoDestroy	Set to true if you want dataStore to be deleted when the widget bond to it is destroyed (default false) 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.

events

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

methods

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

getAt	Returns a give record by index number (starts at 0). Parameters: index Return: record

add	Adds a new record to data store
	
	Parameters: array of data for record fields or JSON record
	
	`[ 'Value 1','Value 2' ]`
	
	or
	
	`{`
	`  'field1','value 1',`
	`  'field2','value 2'`
	`}`
	
	Parameters: data,nobind,first (true/false)
	
	Return: none

removeAll	Removes all the records in a data store. Parameters: none Return: none
removeAt	Removes a record by index Parameters: index Returns true/false
removeLast	Remove last record in collection.
getCount	Get the record count of a data store. Parameters: none Return: count
fetch	Executes an AJAX call to the URL property and if successful loads the new data. Parameters: callback callback is an optional parameter which defines a function to be called upon a successful load. The callback function is passed the dataStore object. Also if the dataStore has been used by other widgets (like dataview) this will be refreshed.
setParam	Sets a URL parameter ready for the data fetch. Parameters: sParam,value

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	Returns the dataStores as a string in JSON form or CSV Parameters: type,separator type: "csv" or "json" (default) separator: character used as a field data separator (default comma)
sort	Sorts the data array records in ascending or descending order Parameters: fieldname, ascending (true/false) This will also refresh any bindings.
find	Find a record that matches a value Parameters: sField,sValue,start,matchAny,caseSensitive sField: field name sValue: value to find start: start position on record collection (default 0) matchAny:(true/false)Look for text at any position in field value (default true) caseSensitive:(true/false) for case sensitivity (default true)
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 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	Load the data record with data. `[{ 'field2':'Value 1','field2':'Value 2' }]` 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. Return: none
loadText	Loads the record from a string Parameters: record_data,separator record_data: string of field data or order of field declaration separator: separating character between field data (default comma)
set	Set a field by name. Parameters: name, value ,nobind Return: true / false if false name not found. 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.
get	Get the value of a field in the data record. Parameters: name Return: value
isModified	Returns true/false if record has changed
commit	Clears the modified status
asString	Returns the record as a string. Parameters: type,separator

	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 "" + value + ""; 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

Created with the Personal Edition of HelpNDoc: Easily create EBooks

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

eval	Return evaluation string as a result. Parameters: string, onerror function
cssOpacity	Returns the cross browser CSS for changing the opacity of a HTML element Parameters: opacity (0 - 1.0) Return CSS styling
showBusy	Shows an animated busy graphic
hideBusy	Hides the animated busy graphic
delay	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
replaceAll	Replaces a sub string within a string Parameters string,substr,replacestr Return new string
round	Returns a number rounded to a specific number of decimal places Parameters: number, decimals
roundUp	
ucwords	Returns a string the first letter of each word has been capitalised Parameters string
dateAdd	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
fromUDateTime	Converts a string in the format YYYYMMDDHHMMSS to a date object
trim	Returns a string trimmed of leading and trailing spaces.
ltrim	Returns a string trimmed of leading spaces.

rtrim	Returns a string trimmed of trailing spaces.
urldecode	return a string url decoded. ie %20 is replaces by a space
urlencode	return a string url encoded. ie a space is replaces by %20
JSON2string	Returns a JSON object as a string Parameters objJSON Return string
mousePosition	Returns the mouse position from a HTML event object Parameters event Return { x: 100, y: 100 }
clone	Create new JSON object as a copy of one you pass. Parameters objJSON Returns objJSON copy
generatePassword	Returns unique password based on the parameters passed. Parameters: length, strength Where: length is the number of characters in password. strength is a bitswise 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
parseJSON	Returns a string in JSON format as a JSON object.
guid	Returns a GUID unique string.
extractIndex	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.
loadCSS	Loads a style sheet dynamically

	Parameters: css src, callback
downloadURL	Downloads the file at the specified url without creating a leave window alert Parameters: url
scrypt	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
switchTheme	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
updateJSONObject	Updates one JSON object with another (or array) where the names match Parameters: json, data json = JSON object to be updated

Created with the Personal Edition of HelpNDoc: Produce Kindle eBooks easily

Macros

Quick function calls to help development time.

Methods

$debug	Echo's a message to the JavaScript debug console. Parameter: variable or text
$msg	Shows a simple message box. Parameters: string message
$popup	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
$error	Shows an error message box. Parameters: string message
$warning	Shows a warning message box. Parameters: string message
$yesno	Shows a message box with yes/no buttons. Parameters: string message,callback The callback is passed a string parameter with "YES" or "NO"
$delay	Performs the same function as the idev.utils.delay method Parameters timeout,callback function, scope
$get	Returns a give widget object by id

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

Created with the Personal Edition of HelpNDoc: Write EPub books for the iPad

Build-In Icons

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

Use one of the following names with the icon or button widgets.

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

globeAlt2, globeAlt, globe, warning, arrowleftalt, arrowalt, code, arrowleft, arrow, pensil, pen, plus,minus

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

star, star2,chat,quote,slideshare,gear2, gear, wrench, magic, download, view, noview, cloud, cloud2, cloudDown

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

cross, check, settings, settingsalt, feed, bug, link, calendar, picker, no, photo, printer, export, import, run ,magnet

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

apps, locked, lab, umbrella, tv, layout, present, size, help, arrowfirst, arrowlast, back, forward

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

Created with the Personal Edition of HelpNDoc: Easily create Help documents

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 your need to include the phonegap.js (provided) in your preferences.js

Created with the Personal Edition of HelpNDoc: Easy EPub and documentation editor

UX Widgets

User e**X**tension 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

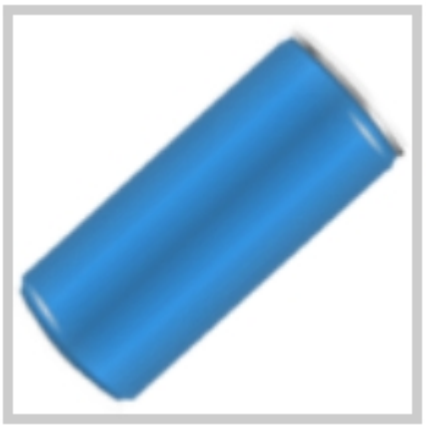
A UX pictures widget creates a rolling sequence of images for the presenting of products and services etc. These images fade in and out as they swap over.

```
{
	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

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.
animSpeed	Speed in milliseconds for fading
delaySpeed	Speed in milliseconds for switch images
pictures	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.
ds	Data store object (see Data Store section)
imageField	Field in the dataStore that contains the url of the image

events

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

method

pause	Pause animation
run	Start animation

Created with the Personal Edition of HelpNDoc: Produce electronic books easily

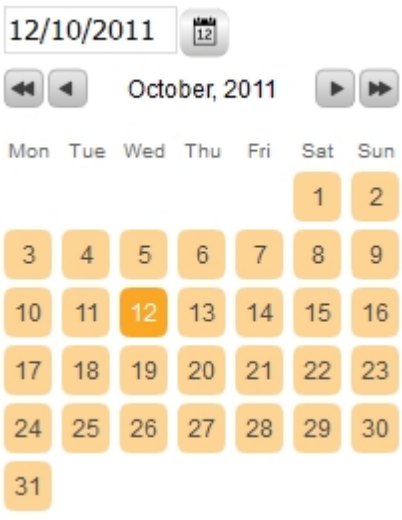
Date Picker

Date Picker

A date picker widget.

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

	}
}
```



options

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.
inputHeight	Height of date input element
value	Initial data value
format	Format to use when displaying data in input (mode 2)
mode	The date picker widget has 2 modes (default 1) 1. Open calendar, show the month in full 2. An input box with trigger button to show calendar
iconColor	Colour of icon on trigger button (default #000)
colorScheme	Calendar colour scheme aqua armygreen bananasplit beige deepblue greenish lightgreen ocean_blue orange (default) peppermint pink purple torqoise

events

click	Fired when date changes. Parameters: date picker widget

method

getValue	Returns the selected date in JSON format.
	
	Returns...
	
	{
	year:2012,
	month:1,
	day:1
	}

getDate	Returns the selected date as a standard date object

getValue2String	Returns the selected date in string format String is format is "yyyymmdd"

setValue	Sets the widget to a particular date.
	
	Parameters: date
	
	Where date = {
	year:2012,
	month:1
	day:1
	}

setDate	Set the widget date. Parameters: date Where date is a standard data object
setDateFromString	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

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.
startcolor	Colour of first arc section of gauge (default #33FF33)
centercolor	Colour of centre arc section of gauge (default #FFCC00)
endcolor	Colour of end arc section of gauge (default #FF0000)
value	Initial value (between 1 - 100)
title	Title to display on gauge

events

click	Fired when gauge is clicked. Parameters: gauge widget

methods

getValue	Returns the current gauge value
setValue	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

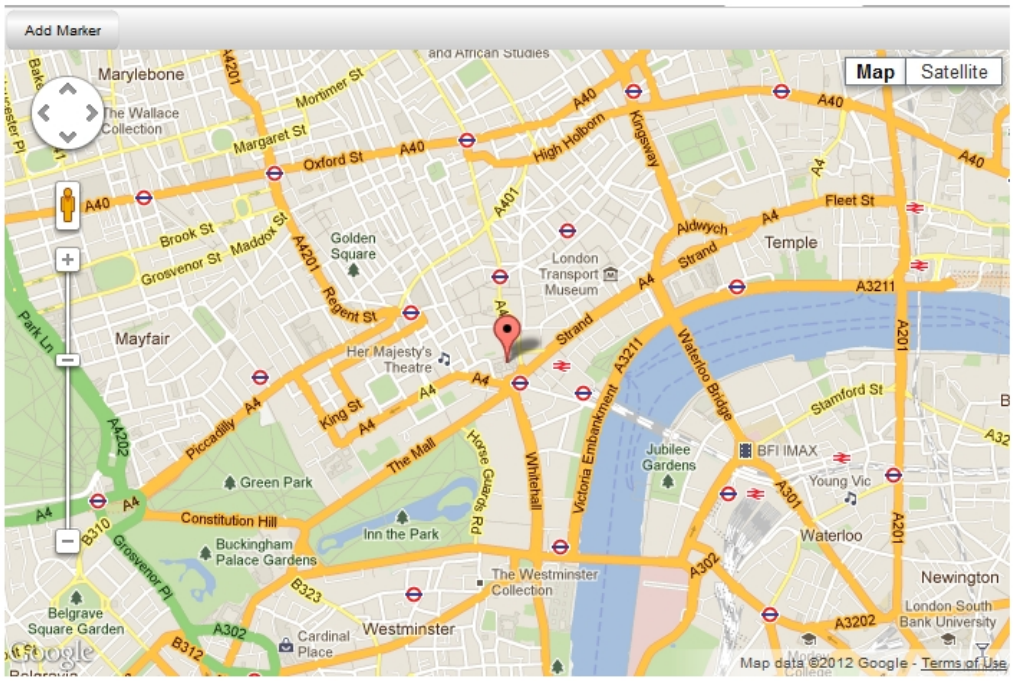
Map

Map

UX widget to add a google map widget.

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

	}
}
```



options

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.
location	GEO location, place name, postal code or longitude and latitude (comma separated).

events

afterRender	Fired after the map widget has been rendered. Parameters: map widget

methods

setCenter	Set the centre of the map to a given place name, postal code or longitude and latitude.

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

Created with the Personal Edition of HelpNDoc: Free help authoring environment

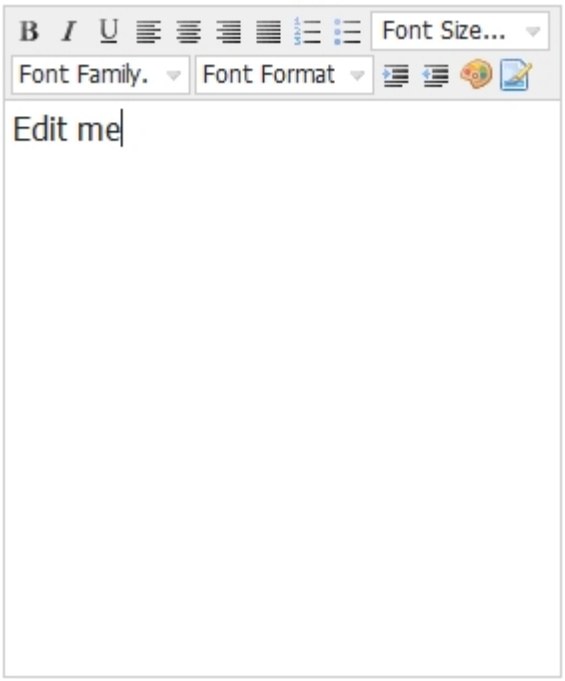
Rich Text

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

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	Initial text value

events

afterRender	Fired after the widget has been created. Parameters: widget

methods

getValue	Returns the current text
setValue	Sets the widget text

Created with the Personal Edition of HelpNDoc: Easily create CHM Help documents

Signature

Signature

A signature widget.

```
{
	wtype:'signature',
	x:220,
	y:5,
	width:220,
	height:280,
	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	Initial signature value
displayOnly	Make the widget display only (default false)
bgColor	Set background colour (default #FFFFFF)
penColor	Set colour of drawing pen (default #145394)
lineColor	Set colour of signature guide line (default #CCCCCC)
lineWidth	Set width of guide line (default 2)
lineMargin	Set the margin at each end of the guide line (default 2)

	lineTop	Position of guide line (default 35)
	style	CSS styling for core widget
events		
	afterRender	Fired after the widget has been created. Parameters: widget
	click	Fired when the widget is clicked Parameters: widget Return: none
methods		
	clear	Clear the signature Parameters: widget
	getValue	Returns the signature as a string
	getImage	Returns the signature as a string that can be the src of a HTML image element
	setValue	Sets a new signature Parameters: signature (must be string)

Created with the Personal Edition of HelpNDoc: Free help authoring environment

Treeview

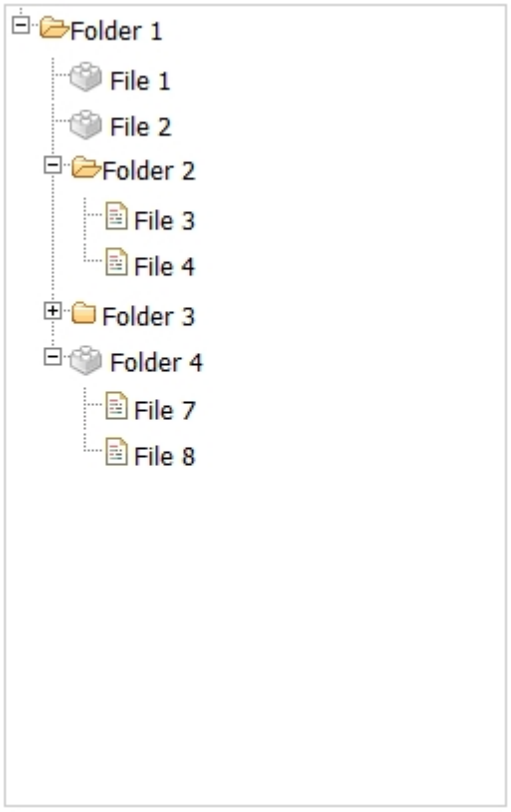
Treeview

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.

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

events

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

method

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

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

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:'tws0001',
            number:'ACC001'
        },
        events: {

        }
}
```

Displays a button that will allow uploading of files

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.
text	Text to be shown on the upload button (default 'Upload')
url	URL the file will be posted to
params	Parameters to be posted with file
image	URL of image to be used on the upload button.

events

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

methods

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

Created with the Personal Edition of HelpNDoc: Easily create iPhone documentation

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 ***idev.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 *dbExecute*

dbExecute	Executes an SQL statement on a database.
	
	Parameters: db,sSQL,args,success,failed
	
	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
	
	No return this is a asynchronous operation
	
	example:
	
	`idev.local.dbExecute(db,"select * from tbNames",function(tx,rx)`
	`{`
	`  // Success function`
	`  for(var i=0; i<rs.rows.length; i++)`
	`  {`
	`  var row = rs.rows.item(i)`
	
	`  $debug(row['fdName']);`
	`  }`
	`},`
	`function(tx,error)`
	`{`
	`  // Failed function`
	`  $debug(error.message);`
	
	`});`
	
	The args parameter is good for insert statements
	
	`INSERT INTO todo(todo, added_on) VALUES (?,?)`
	
	`where the args[0] replaces the first ? and so on.`

Created with the Personal Edition of HelpNDoc: Generate EPub eBooks with ease

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

currentLocation

This obtains the devices current location. The first time this is used the user will be asked if they want their location determined

Parameters: success,failed

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.*geoposition* property.

```
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;
     }
}
```

No return

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

localSupport	This checks that local storage is supported by the browser. Return true or false
storeLocal	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
removeLocal	This removes previously store data Parameters: key key = name of the stored data to remove Return true or false
countLocal	Returns the number of local data values that have been stored
clearLocal	Clears all stored data
getLocal	Gets the previously store data. Parameters: key,default Returns the value stores or the default value

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

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 {
        o
        o
        o
}
```

or

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

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 chance 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"
}
```

Your 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);

Created with the Personal Edition of HelpNDoc: Free iPhone documentation generator

Themes

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.

Created with the Personal Edition of HelpNDoc: Easy CHM and documentation editor

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.

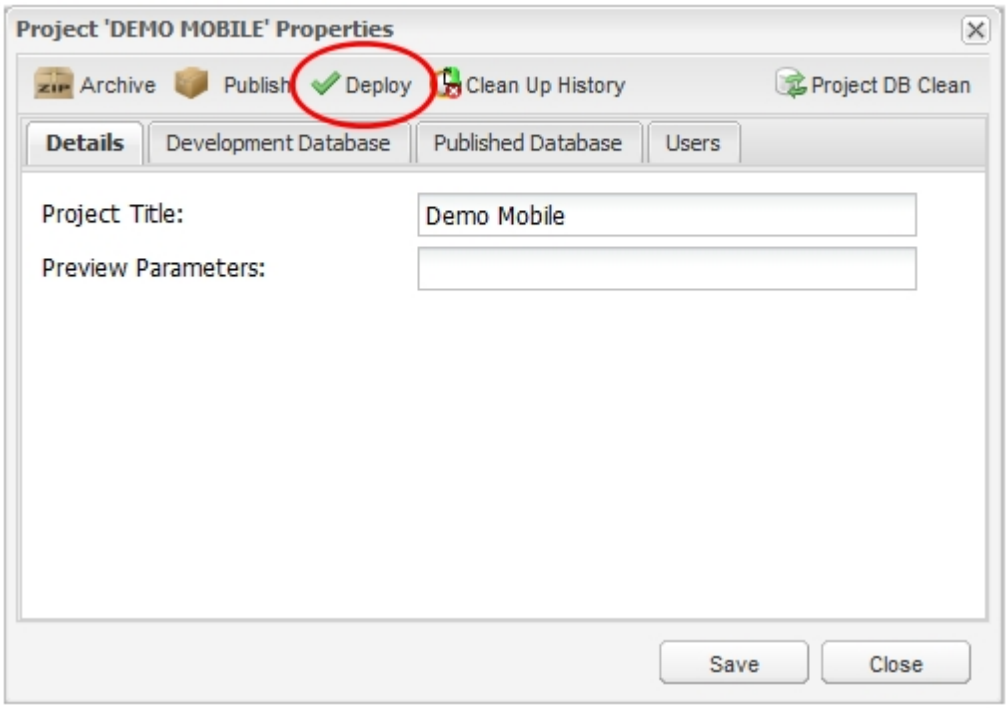
Created with the Personal Edition of HelpNDoc: Full-featured multi-format Help generator

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.

Created with the Personal Edition of HelpNDoc: Full-featured Documentation generator

Acknowledgement

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

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

Please respect their licensing

Created with the Personal Edition of HelpNDoc: Free Web Help generator