

A stylized silhouette of a city skyline with various skyscrapers of different heights and shapes, rendered in shades of gray and black against a dark background.

CUSTOM CONTROLS

Presented by **Henri Reterink**

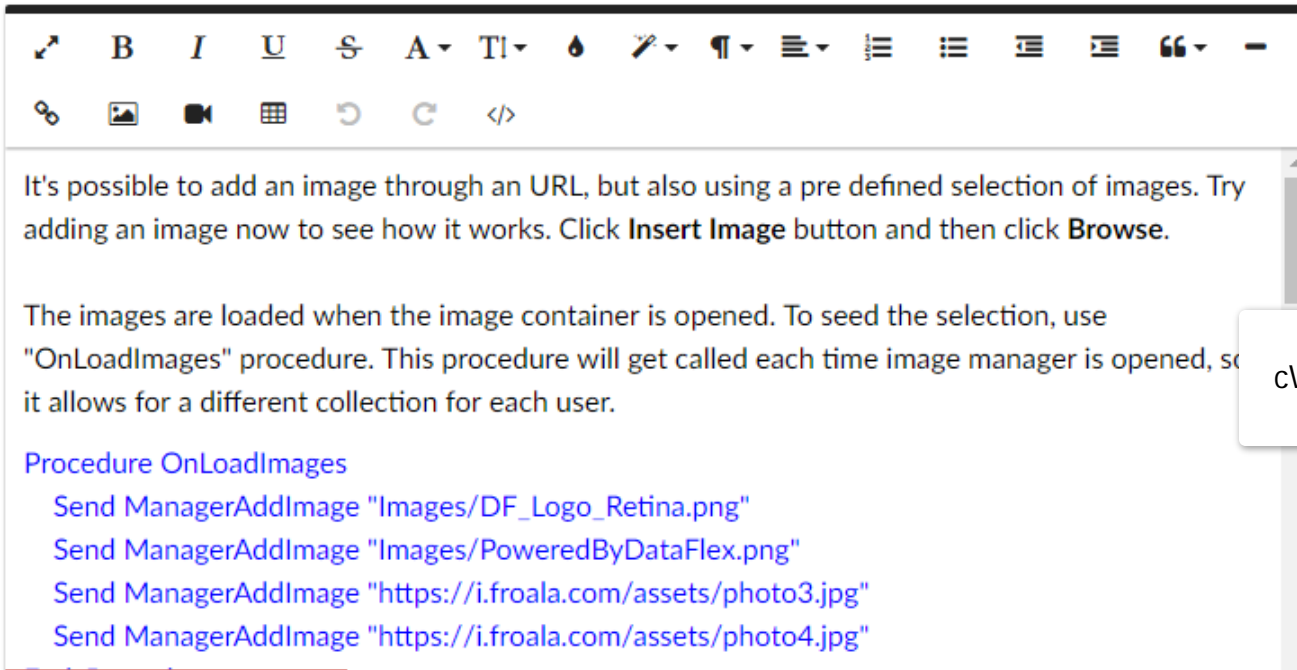
Why a custom control?

- Extra functionalities!
- Generic reusable controls
 - Graphs / Charts
 - Scheduler
 - HTML Editor (Froala)
 - TimePicker
- More feature specific
 - Procedural questionnaire
 - Dynamically generated dashboard

HTML Editor control example

Demo

Image selection



The screenshot shows the Froala HTML editor interface. At the top is a blue header with the word "Demo". Below it is the title "Image selection". The editor's toolbar is visible, containing icons for undo, redo, bold, italic, underline, strikethrough, text color, background color, link, unlink, list, ordered list, table, image, video, table border, link icon, refresh, and code. The main text area contains the following content:

It's possible to add an image through an URL, but also using a pre defined selection of images. Try adding an image now to see how it works. Click **Insert Image** button and then click **Browse**.

The images are loaded when the image container is opened. To seed the selection, use "OnLoadImages" procedure. This procedure will get called each time image manager is opened, so it allows for a different collection for each user.

[Procedure OnLoadImages](#)

- Send ManagerAddImage "Images/DF_Logo_Retina.png"
- Send ManagerAddImage "Images/PoweredByDataFlex.png"
- Send ManagerAddImage "https://i.froala.com/assets/photo3.jpg"
- Send ManagerAddImage "https://i.froala.com/assets/photo4.jpg"

cWebFroalaEditor

Timepicker example

Demo 3: no zero padding, interval of 5 for seconds and minutes

For seconds and minutes (can be set separately), the picker will move up or down in intervals of 5. Editing the field will round up or down to the nearest 5

Label

8:30:15



cWebTimeForm

8 : 30 : 15

Value is loaded from the DB. Editing and saving will

What is a custom control?

- DataFlex component that you can add to a view.
- Usually something visible



DataFlex

- Server side portion
- A Class
- Provides API's for the application to work with



JavaScript

- Client side portion
- Implements the wanted feature using the framework API's
- Usually renders something on screen
- A pseudo "Class" in javascript

Knowledge required

- DataFlex classes and subclassing
- JavaScript
- HTML and DOM Manipulation
- WebApp Framework basics

To some extent...

- CSS
- Ajax

A grayscale silhouette of a city skyline with various skyscrapers of different heights and shapes, set against a dark gray background. The buildings are rendered in shades of black and gray, creating a layered effect.

Case

“Build a scheduler control for the DF Web Framework”

Determining the initial set of requirements

- CRUD: Create, Read, Update and Delete events
- Easy to use, intuitive
- Relatively easy to implement in a variety of applications
- Accept event data in a fixed format, but from multiple sources
- Flexible and customizable

Build or wrap?

Building it ourselves

Pros

- Control over usage, look, etc.
- Possibly more DF “native”

Cons

- Having to build every single piece ourselves

Wrapping an existing library

Pros

- No reinventing the wheel
- Can save a lot of time

Cons

- Limited control over usage, look, etc.
- No API standards
- Possible dependency on other libraries like jQuery

Winner: DHTMLx Scheduler

- Clean, simple but effective look
- User friendly interaction with drag&drop and in-calendar dialogs
- Very flexible
- Very customizable
- Well documented(!)

Cons:

- License for commercial products



addEvent

adds a new event

```
string addEvent(object event);
```

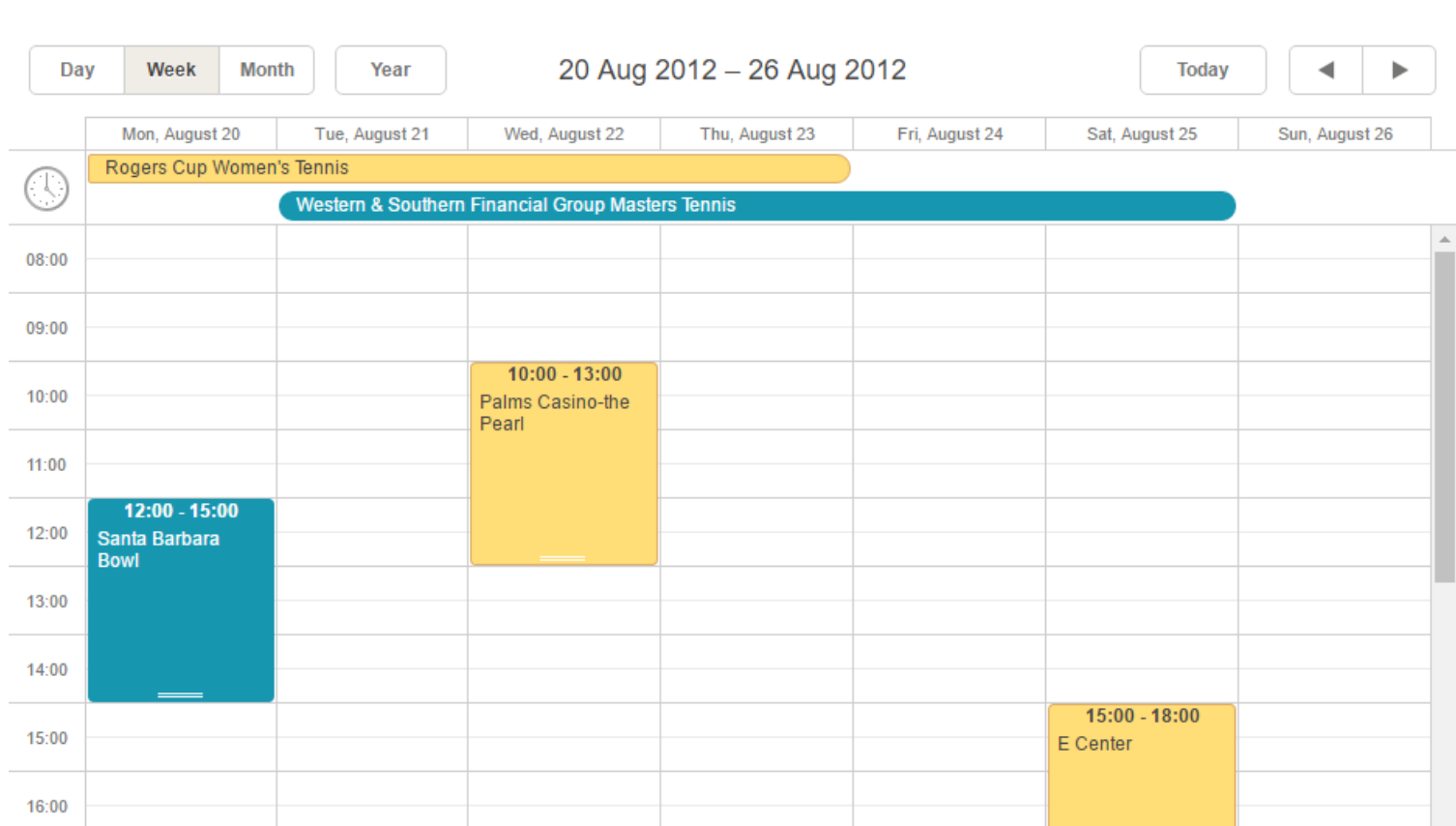
Parameters

<i>event</i>	<i>object</i>	the event object
--------------	---------------	------------------

Returns

<i>string</i>	the event's id
---------------	----------------

Winner: DHTMLx Scheduler





Setting up...

Building for the future. Better, faster, everywhere.

SYNERGY 2017

At the DataFlex side...

- Create a package for our Class
- Extend from a WebApp Framework base class, common ones are
 - cWebObject
 - cWebBaseUIObject
 - cWebBaseControl

```
cWebSchedulerDemo.pkg* x
01 Use cWebBaseControl.pkg
02
03 Class cWebDhxScheduler is a cWebBaseControl|
04
05 End_Class
```

...

- Define our JS class name
 - This will have to match with the name defined in the JS file later

```
Class cWebDhxScheduler is a cWebBaseControl  
  
  Procedure Construct_Object  
    Forward Send Construct_Object  
  
    Set psJSClass to "dfhx.WebDhxScheduler"  
  End_Procedure  
  
End_Class
```



...

- Configure super classes

```
Procedure Construct_Object
  Forward Send Construct_Object

  // Configure super classes
  Set pbFillHeight to True
  Set piColumnSpan to 0
  Set pbShowLabel to False

  Set psJSClass to "dfhx.WebDhxScheduler"
End_Procedure
```



At the JavaScript side...

- Create a .JS file
- Create a “namespace” object (optional)
- Create our constructor function
 - Configure superclasses

```
// Create namespace object (dfhx = dataflex dhtmlx)
if(!dfhx){
  var dfhx = {};
}

dfhx.WebDhxScheduler = function WebDhxScheduler(sName, oPrnt){
  // Configure superclasses
  dfhx.WebDhxScheduler.base.constructor.apply(this, arguments);
};
```

JS

...

- Define our class (matches with class defined in DF)
 - Extend from a base class, common ones:
 - df.WebObject
 - df.WebBaseUIObject
 - df.WebBaseControl

```
// Create namespace object (dfhx = dataflex dhtmlx)
if(!dfhx){
    var dfhx = {};
}

dfhx.WebDhxScheduler = function WebDhxScheduler(sName, oPrnt){
    // Configure superclasses
    dfhx.WebDhxScheduler.base.constructor.apply(this, arguments);
};

df.defineClass("dfhx.WebDhxScheduler", "df.WebBaseControl", {
    // Control functions and logic will go here
});
```

JS

WebProperties

- Can be used to store things like data, status information and settings for the control
- Property value is shared and maintained between Client and Server
- Engine generates default Get & Set functions in JS if no custom ones are defined
 - Called when executing WebGet and WebSet functions in DF, and at several other times
 - Implement a custom setter method as `set_psPropertyName`
 - Implement a custom getter method as `get_psPropertyName`

WebProperties

- Dataflex definition

```
// Defines whether events can be dragged or not
{WebProperty = True}
Property Boolean pbAllowEventDrag
```



- JavaScript definition

```
dfhx.WebDhxScheduler = function WebDhxScheduler(sName, oPrnt){
    dfhx.WebDhxScheduler.base.constructor.apply(this, arguments);

    // Define our property
    this.prop(df.tBool, "pbAllowEventDrag", true);
};
```

- Our custom setter

```
set_pbAllowEventDrag : function(bVal){
    if(this._eControl){
        // DhxNoResize true tells the dHTMLx scheduler to not allow event dragging
        // Calling "Set pbAllowEventDrag false" from DF would set DhxNoResize to true, and block event dragging
        df.dom.toggleClass(this._eControl, "DhxNoResize", !bVal);
    }
},
```

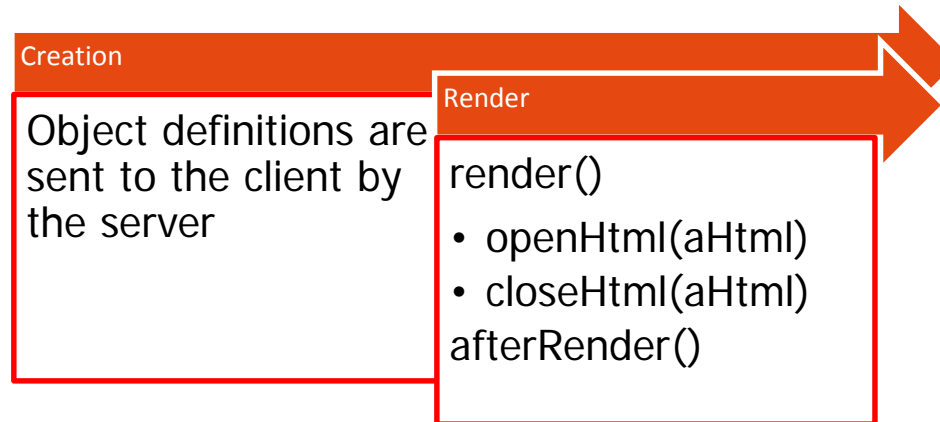


Displaying your control on screen

OpenHtml, CloseHtml and AfterRender

Rendering the control

- UI Objects render themselves on the client by generating HTML
- Recursive functions `openHtml`, `closeHtml` and `afterRender` are triggered
 - `openHtml` and `closeHtml` are used to insert HTML elements into the document
 - `afterRender` is called afterwards to perform DOM Manipulation



openHtml, closeHtml

- Our openHtml and closeHtml functions
 - Create a container for our scheduler to fit in using HTML elements
 - On initialization we'll tell the dHTMLx Scheduler to render itself inside this container

```
// Creates the HTML elements to "draw" for our control
openHtml : function(aHtml){
  // "Forward send"
  dfhx.WebDhxScheduler.base.openHtml.call(this, aHtml);
```

```
  aHtml.push('<div class="dhx_cal_container" style="width:100%; height:100%;">');
  aHtml.push('    <div class="dhx_cal_navline">');
  aHtml.push('      <div class="dhx_cal_date"></div>');
  aHtml.push('    </div>');
  aHtml.push('    <div class="dhx_cal_header"></div>');
  aHtml.push('    <div class="dhx_cal_data"></div>');
  aHtml.push('</div>');
},
```

```
// Creates the HTML elements to "draw" for our control
// Called after all openHtml functions are done
closeHtml : function(aHtml){
  // Just "Forward Send" for now
  dfhx.WebDhxScheduler.base.closeHtml.call(this, aHtml);
},
```

JS

afterRender

- Our openHtml and closeHtml functions
 - Create a container for our scheduler to fit in using HTML elements
 - On initialization we'll tell the dHTMLx Scheduler to render itself inside this container

```
// Called after the control has been rendered on screen
afterRender : function(){
  // Find our container, mark it as our control root
  this._eControl = df.dom.query(this._eElem, ".dhx_cal_container");

  // Forward send
  dfhx.WebDhxScheduler.base.afterRender.apply(this, arguments);

  // This function sets dHTMLx Scheduler properties and initializes it in our container
  this.initScheduler();

  // Can only call this setter after control is rendered
  this.set_pbAllowEventDrag(this.pbAllowEventDrag);
},
```

JS

Simple Scheduler

Day **Week** Month 24 Aug 2015 – 30 Aug 2015 Today < >

	Mon, August 24	Tue, August 25	Wed, August 26	Thu, August 27	Fri, August 28	Sat, August 29	Sun, August 30
0 ⁰⁰							
1 ⁰⁰							
2 ⁰⁰							
3 ⁰⁰							
4 ⁰⁰							
5 ⁰⁰							
6 ⁰⁰							



Making it do stuff

ServerActions and ClientActions

ServerActions

- Basically just DF procedures / functions
- Can be called by the JS Client by using:
 - `this.serverAction("MethodName", aParams);`
- Optionally provide a return value
- Must be made available to the client by using
 - `WebPublishProcedure ProcedureName`
 - `WebPublishFunction FunctionName`

ServerAction to load events

- Our ServerAction, a DataFlex procedure

```
Procedure LoadEvents String sMode String sStartDate String sEndDate
  tWebValueTree tVT
  String[] aParams
  Date dStart dEnd
  // Event Array to be filled and sent to the client
  tWebDhxEvent[] aEvents

  Move (ConvertFromClient(typeDate, sStartDate)) to dStart
  Move (ConvertFromClient(typeDate, sEndDate)) to dEnd

  // Event hook, allows augmentation for application specific logic. Fill event array
  Send OnLoadEvents (&aEvents) sMode dStart dEnd

  // Serialize Event array to a WebValueTree
  ValueTreeSerializeParameter aEvents to tVT

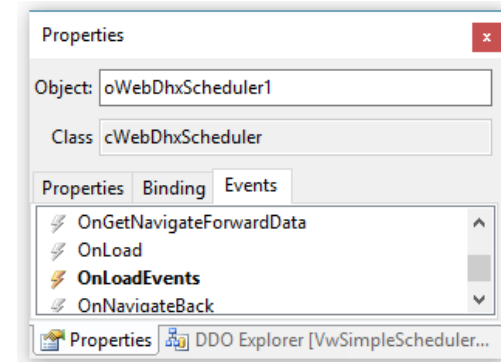
  // Call clientaction to process data
  Move sMode to aParams[0]
  Move sStartDate to aParams[1]
  Move sEndDate to aParams[2]

  Send ClientAction "handleEvents" aParams tVT
End_Procedure
```



Event “hook”

- Event “hook” for our ServerAction
 - Allows you to define the main universal logic yourself
 - Developers can place application specific logic inside the event
 - Shows up in Object Properties panel!



```
// Called by LoadEvents
{ MethodType=Event }
Procedure OnLoadEvents tWebDhxEvent[] ByRef aEvents String sMode Date dStart Date dEnd
    // Augment this procedure with your own business logic to load events
End_Procedure
```



Exposing and calling our procedure

- Exposing our procedure to the client

```
Procedure End_Construct_Object
  Forward Send End_Construct_Object

  // Allows our "ServerAction" to be called from the JS Client
  WebPublishProcedure LoadEvents

End_Procedure
```



- Calling our procedure from the client

```
loadData : function(sMode, dStart, dEnd){
  // Calls the "LoadEvents" procedure on the server, with given parameters
  this.serverAction("LoadEvents", [ sMode, this.toSvrDate(dStart), this.toSvrDate(dEnd) ], null, function(){
    // Logic to be executed if the function / procedure has returned something
  });
},
```

The logo consists of the letters 'J' and 'S' in white, set against a yellow square background.

JS

ClientActions

- Functions in JavaScript
- Can be called by the DF Server by using
 - Send ClientAction "methodName" aParams
 - Send ClientAction "methodName" aParams tWebValueTree
- tWebValueTree is data serialized to a format the client and server can both read and deserialize
- Work asynchronously
- Parameters are provided as Strings

Calling our ClientAction

- Calling our ClientAction “handleEvents” from the server

```
// Serialize data
ValueTreeSerializeParameter aEvents to tVT

// Call clientaction to process data
Move sMode to aParams[0]
Move sStartDate to aParams[1]
Move sEndDate to aParams[2]

Send ClientAction "handleEvents" aParams tVT
End_Procedure
```



Our ClientAction

- Our ClientAction, a function in JavaScript

```
// (De)serializers for data sent by and to the server
deserializeVT : df.sys.vt.generateDeserializer([ dfhx.tWebDhxEvent ]),
serializeVT : df.sys.vt.generateSerializer([ dfhx.tWebDhxEvent ]),

// ClientAction called by the server
handleEvents : function(sMode, sStart, sEnd){
    // Deserialize received data
    var i, aEvents = this.deserializeVT(this._tActionData);

    if(this._oScheduler){
        this._oScheduler.clearAll();

        // Convert dates
        for(i = 0; i < aEvents.length; i++){
            aEvents[i].start_date = df.sys.data.stringToDate(aEvents[i].start_date, "yyyy/mm/ddThh:mm:ss.fff");
            aEvents[i].end_date = df.sys.data.stringToDate(aEvents[i].end_date, "yyyy/mm/ddThh:mm:ss.fff");
        }

        // Pass to scheduler control
        this._oScheduler.parse(aEvents, "json");
    }
},
```

JS



Using your control in multiple projects

Building for the future. Better, faster, everywhere.

SYNERGY 2017

How to import your control

- Develop your control in a separate, standalone workspace
- In your application workspace...
 - add the control workspace as a library
 - copy the control .JS file to an AppHtml subfolder
 - copy any other required files used by the control as well if needed
 - add a reference to the required .JS files in your Index.html
- You can even add it to the class palette for convenience!

```
<!-- DHTMLX Scheduler -->
<script src="dhtmlx/dhtmlxscheduler.js" type="text/javascript"></script>
<script src="dhtmlx/ext/dhtmlxscheduler_limit.js"></script>
<script src="dhtmlx/ext/dhtmlxscheduler_units.js"></script>
<script src="dhtmlx/ext/dhtmlxscheduler_timeline.js"></script>
<script src="dhtmlx/ext/dhtmlxscheduler_multiselect.js"></script>
<script src="dhtmlx/ext/dhtmlxscheduler_tooltip.js"></script>
<!-- <link rel="stylesheet" href="dhtmlx/dhtmlxscheduler.css" type="text/css"> -->
<link rel="stylesheet" href="dhtmlx/dhtmlxscheduler_flat.css" type="text/css">
```

```
<!-- DataFlex Custom Controls (do not remove this line, used for automatic insertion) -->
<script src="Custom/WebDhxScheduler.js"></script>
```



Demo...

Building for the future. Better, faster, everywhere.

SYNERGY 2017

There's much more to learn!

- Some things we haven't talked about include...
 - Events in-depth
 - Private client properties
 - Synchronized WebProperties
 - Serializing and deserializing data
 - Etc.
- Want some examples? The default WebApp classes are right there!
 - Take a look in the AppHtml/DfEngine of your webapp
 - Study controls like the WebButton, WebForm, etc.



Thank you for your attention
Are there any questions?