assono

dōjō and Notes

ILUG

10th November 2010

Innovative Software-Lösungen.

www.assono.de

Bernd Hort

Graduated in Computer Science, University Hamburg

Lotus Notes Application Developer since 1995

IBM Certified Application Developer, System Administrator & Instructor

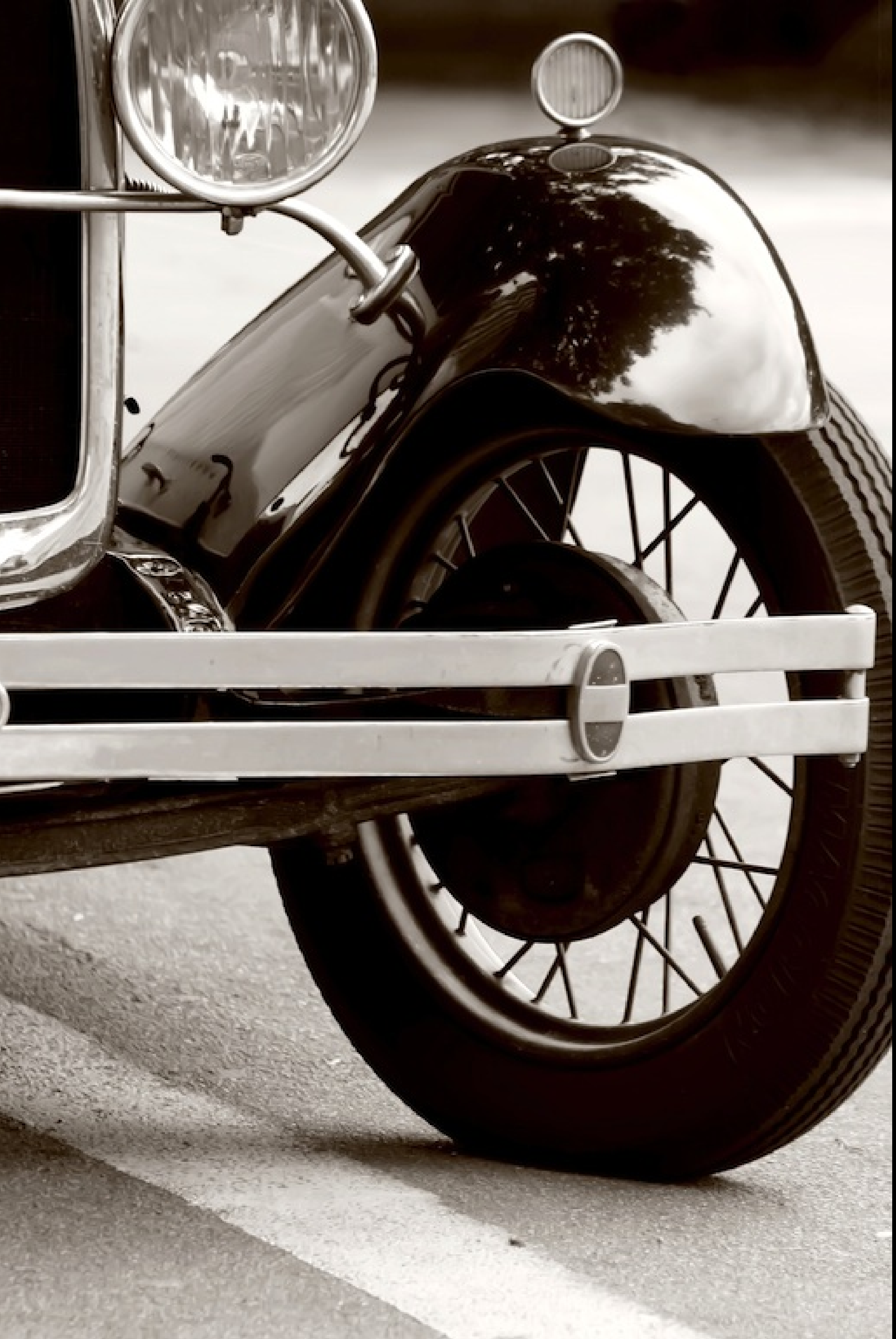
assono

IT-Consulting & Solutions

bhort@assono.de
http://www.assono.de/blog
+49.40.73 44 28-315

Agenda

- Motivation
- Introduction in dojo
- Core
- Dijit
- DojoX
- Dojo and Notes
- Questions & Answers



Don't Reinvent The Wheel!

Motivation – Why dojo?

Open Source

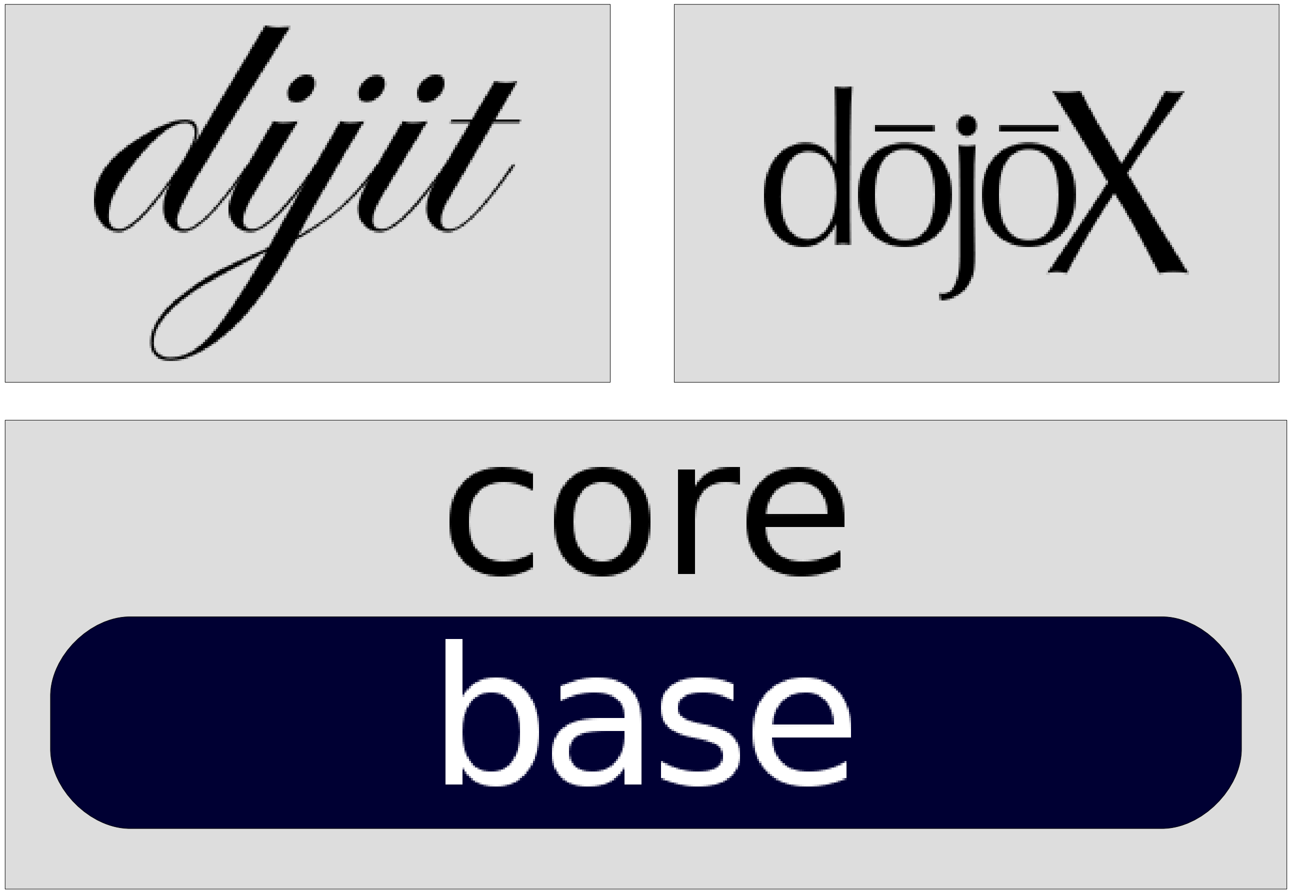
Browser Support SUPPORTING

Rich UI Widgets

I18N - Internationalization

Used in XPages

The three parts of dojo



core - base

- Browser detection
- JSON encoding / decoding
- Package loading
- Event handling
- Animation effects
- AJAX
- CSS utilities
- OOP support
- Firebug integration

base

assono

core – other packages

core

- Drag & drop
- I18N support
- Date formatting
- Number formatting
- String utilities
- Cookie handling
- Extended animations
- Back button handling

dijit

- GUI elements – dojo interface widgets
- „Theme“ support
- Internationalization
- Keyboard support

dojoX

- Experimental - „The future, today“
- Some parts are really stable and production ready
- Charts
- SVG support
- DojoX Offline – Integration with Gears
- DojoX Widgets

«Page»
dojo-Hello World

Hello World

```
<html>
  <head>
    <title>My First Dojo App</title>
    <link rel="StyleSheet" type="text/css"
      href="js/dojo/dojo/resources/dojo.css">
    <link rel="StyleSheet" type="text/css"
      href="js/dojo/dijit/themes/tundra/tundra.css">
    <script type="text/javascript">
      var djConfig = {
        baseScriptUri : "js/dojo/",
        parseOnLoad : true,
        extraLocale: ['en-us', 'zh-cn']
      };
    </script>
    <script type="text/javascript" src="js/dojo/dojo/dojo.js"></script>
    <script language="JavaScript" type="text/javascript">
      dojo.require("dojo.parser");
      dojo.require("dijit.form.Button");
      dojo.require("dijit._Calendar");
    </script>
  </head>
```

Base CSS

Theme CSS

Base Configuration
must be defined before dojo.js is loaded!

dojo.js (Base)

Loading
of packages

Hello World

```
  <body class="tundra">
    <div style="position:relative;top:10px;left:10px;width:80%;">
      <button dojoType="dijit.form.Button" id="myButton">
        Press me, NOW!
        <script type="dojo/method" event="onClick">
          alert('You pressed the button');
        </script>
      </button>
      <br><br>
      <table border="0"><tr>
        <td valign="top">
          <input id="calEnglish" dojoType="dijit._Calendar" lang="en-us" />
        </td>
        <td width="25">&nbsp;</td>
        <td valign="top"
          <input id="calChinese" dojoType="dijit._Calendar" lang="zh-cn" />
        </td>
      </table>
    </div>

  </body>
</html>
```

Button

Calender US

Calender CN

Hello World - Explanation

```
<link rel="StyleSheet" type="text/css"
   href="js/dojo/dojo/resources/dojo.css">
```

- Simple basic CSS – only margins and font definitions

```
<link rel="StyleSheet" type="text/css"
     href="js/dojo/dijit/themes/tundra/tundra.css">
```

- Tundra is one of the build in Themes

Hello World - Explanation

```
<script type="text/javascript">
   var djConfig = {
   baseScriptUri : "js/dojo/",
   parseOnLoad : true,
   extraLocale: ['en-us', 'zh-cn']
   };
</script>
```

- Basic configuration of dojo
  - Creates a djConfig object
  - The object will be read from the dojo.js
  - Property baseScriptUri: The dojo directory
  - Property parseOnLoad: Analyze of the HTML code
  - Property extraLocale: Additional languages
  - Must be defined before dojo.js is loaded

Hello World - Explanation

```
<script type="text/javascript" src="js/dojo/dojo/dojo.js"></script>
```

- Loading of the „base“ part

```
<script language="JavaScript" type="text/javascript">
  dojo.require("dojo.parser");
  dojo.require("dijit.form.Button");
  dojo.require("dijit._Calendar");
</script>
```

- Definition which packages has to be loaded
  - dojo.parser: Parsing of the HTML code for dojo relevant elements
  - dijit.form.Button: „Button“ widget
  - dijit._Calendar: „Calender“ widget

Criticism – Use of packages

- The use of packages (dojo.require) means a lot of small server request
- From a performance point of view a lot of small requests are suboptimal
- Dojo has a tool for building one file with all the needed packages – „Custom Dojo Build"
- Blog Tim Tripcony „I still have one concern about Dojo"
  http://www.timtripcony.com/blog.nsf/d6plinks/TTRY-7LLHQP
- Tim Tripcony also offers a tool „JSFactory"
  http://www.timtripcony.com/blog.nsf/d6plinks/TTRY-7FUSZF

Hello World - Explanation

```
<button dojoType="dijit.form.Button" id="myButton">
  Press me, NOW!
  <script type="dojo/method" event="onClick">
     alert('You pressed the button');
  </script>
</button>
```

- Creation of a button
  - Parser recognizes the dojoType
  - „dijit.form.Button“ loads the button widget
- Registration of the „onClick“ event
  - Parser recognizes type=“dojo/method“

Hello World - Erklärung

```
<table border="0"><tr>
  <td valign="top">
    <input id="calEnglish" dojoType="dijit._Calendar" lang="en-us" />
  </td>
  <td width="25">&nbsp;</td>
  <td valign="top"
    <input id="calChinese" dojoType="dijit._Calendar" lang="zh-cn" />
  </td>
</table>
```

- Creation of the two calendar objects
  - Parser recognizes dojoType
  - „dijit._Calendar“ loads the calendar widget
  - With the „lang“ attribute the default language is overwritten

base – dojo.addOnLoad()

base

- Register a function that will be invoked as soon as the DOM is loaded and all widgets are initialized

- `dojo.addOnLoad(functionPointer);`
  Simple function call

- `dojo.addOnLoad(object, "methodName");`
  Calls a method in the given object

- `dojo.addOnLoad(object, function(){ /* ... */});`
  Defines a new method in the object which will be called

base – CSS Classes

base

- `dojo.addClass(DomNode|NodeId, ClassName)`
  adds a CSS class to the DOM node
- `dojo.removeClass(DomNode|NodeId, ClassName)`
  removes a CSS class from a DOM node
- `dojo.hasClass(DomNode|NodeId, ClassName)`
  checks whether the DOM node has a CSS class

base – DOM Nodes

base

- `dojo.byId(NodeId)`
  returns the DOM node with the given id

- `dojo.clone(Object)`
  clones an object, like a DOM node, inclusive all of its children

- `dojo.isDescendant(ChildDomNode, ParentDomNode)`
  checks whether a DOM node is a child of another DOM node

base – dojo.query() (1)

- `dojo.query(CSS3-Selector)` returns all DOM nodes which meet a given CCS3 selector
  - Class name, e.g.. „.foo“
  - HTML elements, e.g. „span“
  - CSS hierarchies, e.g. „table div“
  - Direct child elements(>), e.g. „#tabular_data > div“
  - Universal selector (*)
  - Immediate predecessor siblings (~)
  - Preceded-by-sibling (+)

base

base – dojo.query() (2)

base

- `dojo.query(CSS3-Selector)`
  Query by attributes
  - [foo] Attribute exists
  - [foo='bar'] Attribute has a given value
  - [foo~='bar'] Attribute value machtes with element from list
  - [foo^='bar'] Attribute value starts with
  - [foo$='bar'] Attribute value ends with
  - [foo*='bar'] Attribute value matches substring

base – dojo.query() (3)

base

- `dojo.query(CSS3-Selector)`
  Pseudo classes
  - :first-child – First child
  - :last-child – Last child
  - :only-child – Only one child
  - :empty – No childs
  - :checked – Activated radio buttons & check boxes
  - :nth-child(n) – Nth child
  - :nth-child(even) – All „even“ childs
  - :nth-child(odd) – All „odd“ childs
  - :not(...) - Negation

base – OOP – declare

- `dojo.declare(className: String, superclass: Function|Function[], props: Object)`
  Creates a new class which will inherit from one or more classes and will inherit the properties of a given object

base

base – OOP – declare - Sample

```
function MyClass1() {
  var firstName = "Mick";
  this.getFirstName = function() {
    return firstName;
  }
}
function MyClass2() {
  var lastName = "Foley";
  this.getLastName = function() {
    return lastName;
  }
}
dojo.declare("org.ilug.AnotherClass", [ MyClass1, MyClass2],
  { middleName : "William",
    getMiddleName : function() {
      return this.middleName;
     }
   }
 );
var o = new org.ilug.AnotherClass();
 alert(o.getFirstName() + " " + o.getMiddleName() + " " +
   o.getLastName());
```

assono

base – OOP – extend

base

- `dojo.extend(constructor: Object, props: Object...);`
  Adds the properties and methods from one or more classes to another class (Prototype inheritance)

base – OOP – extend - Beispiel

```
function MyClass1() {
  var firstName = "Mick";
  this.getFirstName = function() {
    return firstName;
  }
}
function MyClass2() {
  var lastName = "Foley";
  this.getLastName = function() {
    return lastName;
  }
}
function MyClass3() {
  this.sayName = function() {
    alert("I am " + this.getFirstName() + " " + this.getLastName());
  }
}
dojo.extend(MyClass3, new MyClass1(), new MyClass2());
var mc3 = new MyClass3();
mc3.sayName();
```

base – Aspect-Oriented Programming – connect

- Calls a method / function as soon as a event has been triggered or a method of another object has been called

```
dojo.connect(object: Object|null,
event: String, context: Object|
null, method: String|Function);
```

- Connection to a DOM node

```
dojo.connect(dojo.byId(„foo"),
„onmouseover", function(evt)
{console.log(evt)});
```

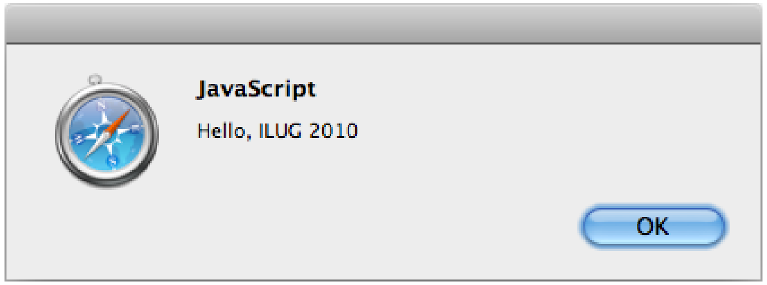
base

base – Aspect-Oriented Programming – connect

- The event can also be a method call of another object.

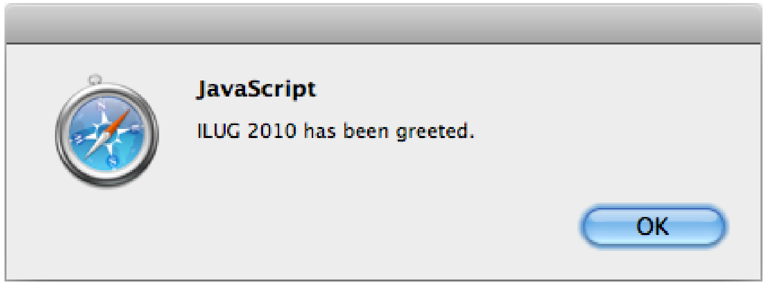
base

```
function MyClass(){
  this.sayHello = function(inName){
    alert("Hello, " + inName)}
}
function AnotherClass(){
  this.echo = function(message){
    alert(message + " has been greeted.")}
}
foo = new MyClass();
bar = new AnotherClass();
var handle = dojo.connect(foo, "sayHello", bar,
"echo");
foo.sayHello ("ILUG 2010");
```



JavaScript

Hello, ILUG 2010



JavaScript

ILUG 2010 has been greeted.

assono

base – AJAX

base

- `dojo.xhr(method: String, args: dojo.__XhrArgs, hasBody: Boolean?);`
  Starts an AJAX call

- Supported methods:
  “DELETE“, „GET“, „POST“ & „PUT“

- Alternative
  - `dojo.xhrDelete(args: dojo.__XhrArgs);`
  - `dojo.xhrGet(args: dojo.__XhrArgs);`
  - `dojo.xhrPost(args: dojo.__XhrArgs);`
  - `dojo.xhrPut(args: dojo.__XhrArgs);`

base – XhrArgs (1)

base

- `url:String`
  The URL for the call
- `handleAs:String`
  Format of the server response
  - text (default)
  - json
  - json-comment-optional
  - json-comment-filtered
  - javascript
  - xml

base – XhrArgs (2)

base

- `form:DOMNode`
  The values from the form will be submitted as field1=value1&field2=value2&....

- `content:Object`
  The properties of the object will be submitted as property1=value1&property2=value2&....

- `headers:Object`
  Additional HTTP Header
  The properties of the object will be submitted as name-value-pairs

assono

base – XhrArgs (3)

base

- `load:Function`
  The Function, which will be called in case of a successful server response
- `error:Function`
  The function, which will be called in case of an error
- `handle:Function`
  The function, which will be called in any case

base – XhrArgs (4)

base

- `sync:Boolean`
  Synchronized call (Browser is blocked until the response arrived)
  - Default is `false`
- `preventCache:Boolean`
  On `true` the browser won't cache the request
- `timeout:Integer`
  The amount of milliseconds to wait until an error is thrown

base – Basic Animation

base

- `dojo.animateProperty(args: dojo.__AnimArgs);`
  Basic for all animation effects

-
  ```
  dojo.animateProperty({ node: node,
  duration:2000,
    properties: {
      width: { start: '200', end: '400',
          unit:"px" },
      height: { start:'200', end: '400',
          unit:"px" },
      paddingTop: { start:'5', end:'50',
          unit:"px" }
      }
  }).play();
  ```

base – fadeIn & fadeOut

base

- `dojo.fadeOut({node : "myDiv", duration : 2000}).play();`
- `dojo.fadeIn({node : "myDiv", duration : 2000}).play();`
-
```
function doFading() {
dojo.fadeOut({node : "myDiv", duration :
2000, onEnd : function() {
    dojo.fadeIn({node : "myDiv", duration :
        2000}).play();
      }
    }).play();
}
```

dojo.fx – Additional Animation Effects

core

- `dojo.require("dojo.fx");`
  Loading of the fx package

- `dojo.fx.slideTo({ node: node, left:"40", top:"50", unit:"px" }).play();`
  Moves the DOM node from its current position to the given position

- `dojo.fx.wipeIn({ node: node, duration:200).play();`
  DOM node is wiped in

- `dojo.fx.wipeOut({ node: node, duration:200).play();`
  DOM node is wiped out

dojo.back – Back Button Handling

core

- The back button is the most used button in a browser
- With dojo.back defined states can be added to the history

-
  ```
  dojo.require("dojo.back");
  dojo.back.setInitialState(state);
  dojo.back.init();
  dojo.back.addToHistory(state);
  var state = {
      back: function() { alert("Back was clicked!"); },
      forward: function() { alert("Forward was clicked!"); }};
  ```

dojo.behavior – Event Handling

- Registration of event handlers on any kind of DOM node

core

- DOM nodes will be identified based on CSS3 selectors (see also dojo.query())

```
dojo.require("dojo.behavior");
dojo.behavior.add({"#myDiv" : {
    found : function (elem) {
      alert("Found the div: " + elem);
    },
    onmouseover: function (evt) {
      alert("onMouseOver fired");
    },
    onmouseout: function (evt) {
      alert("onMouseOut fired");
    }
  }
 });
dojo.behavior.apply();
```

dojo.dnd – Drag & Drop

core

- Two approaches
  - Just moving of html elements on the screen
  - Container definition to define source and target
- Both approaches are simple to use and could be controlled very accurately

dojo.movable

- `dojo.require("dojo.dnd.movable");`
  Loading of the Drag&Drop package

core

- `<div dojoType="dojo.dnd.Moveable">`
  `Some Content</div>`
  Just by specifying the dojoTypes the HTML element can be moved on the screen.

dojo.dnd.movable – with „Handle“

core

- A child node inside the HTML element to be moved could become a handle.
-
```
<div dojoType="dojo.dnd.Moveable"
handle="dragHandle">
  <div id="dragHandle"></div>
  <textarea>Dieser Text kann editiert werden
   </textarea>
</div>
```

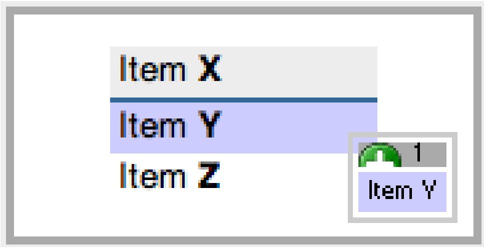
dojo.dnd.Source – Simple Container Definition

core

- `dojo.require("dojo.dnd.Source");`
  Loading of the Drag&Drop package

- Defining of the list

```
<div dojoType="dojo.dnd.Source"
class="container">
 <div class="dojoDndItem">Item X</div>
 <div class="dojoDndItem">Item Y</div>
 <div class="dojoDndItem">Item Z</div>
</div>
```



- Together with the right CSS style sheet there is visual feedback during the move

Other dojo core Packages

core

- dojo.cldr – Support for different country settings
- dojo.colors – Colors
- dojo.currency – Formatting and parsing of currency values
- dojo.date – Date functions
- dojo.i18n – Helper functions for internationalization
- dojo.io – AJAX alike functions without XmlHttpRequest
- dojo.number – Formartting and parsing of numbers
- dojo.regexp – Regular expressions
- dojo.string – String functions

assono

dijit - Introduction

- Dijit stands for
  - The widget framework based on the base and core elements
  - One widget is also called a dijit

- Dijits are fully „theme-able“ - they can be adapted to a „Look & Feel“
  - The „Theme“ has to be added as CSS
  - `<body class="{Theme}">`
  - One of the favorite themes is „Tundra“

Two Ways to Use dijits

- Dijits can be used in two ways
  - Declarative
  - Programmatically
- With the declarative approach the HTML code get parsed in search for attributes „dojoType"
- If the parser finds a dojoType attribute the JavaScript code will be injected automatically
- Be aware that HTML validators won't approve your code as valid with dojoType attributes in it.

Dijits Declarative Use

-
  ```
  var djConfig = {
      baseScriptUri : "js/dojo/",
      parseOnLoad : true
  };
  ```
  In the configuration set the option parseOnLoad to true

-
  ```
  dojo.require("dojo.parser");
  ```
  Load the parser package

-
  ```
  dojo.require("dijit.form.Button");
  ```
  Load the dijit package you want to use

-
  ```
  <button dojoType="dijit.form.Button">ok</button>
  ```
  Add the dojo attribute

Dijit Programmatic Use

- `dojo.require("dijit.form.Button");`
  Load the dijit package

-
```
var myDijit =
new dijit.form.Button(
   {label : "Ok"}
);
dojo.byId("divButton").appendChild(myDijit.do
mNode);
```

- The programmatic approach is performance wise the better approach.
  - But the differences will appear only with big web pages
  - Or within high dynamic web pages

Access to dijits

- Every dijit could be accessed by the dijit Manager
- Every dijit has an unique id
  - Either manually defined
  - Or automatically generated
- The method `dijit.byId(<someID>);` returns the dijit object
- Not to be confused with `dojo.byId(<someID>);`
  - Which will return the DOM node

assono

dijit Methods

Methode	Beschreibung
`buildRendering`	Constructs the visual part representation
`connect`	Connects the object / event to the specified method of the dijit and automatically register it for `disconnect()` on the dijit destroy
`create`	Begins the life cycle of the dijits
`destroy`	Destroys the dijit but keeps the child elements
`destroyDescendants`	Destroys the child elements recursively
`destroyRecursive`	Destroys the dijit and all child elements recursively – This is the recommended method

dijit Methods

Methode	Beschreibung
`destroyRendering`	Destroys all DOM nodes of the dijit
`disconnect`	Disconnects a handle
`getDescandants`	Returns all child elements
`isFocusable`	Checks whether the dijit can be focused
`isLeftToRight`	Checks the DOM for the text direction for bidirectional support
`onBlur`	This method will be called if the dijit loses the focus
`onClose`	This method will be called if the dijit is destroyed

dijit Methods

Methode	Beschreibung
`onFocus`	This method will be called if the dijit gains the focus
`postCreate`	This method will be called after the DOM-Nodes of the dijit has been fully created
`postMixInProperties`	This method will be called after all parameters has been processed but before the DOM nodes are created
`setAttribute`	Sets a HTML attribute directly
`startUp`	This method will be called after all dijits children have been created but before they are displayed.

dijit Attributes

Attribut	Beschreibung
`attributeMap`	List of all attributes, mainly HTML attributes
`class`	HTML class attribute
`id`	The dijit ID
`lang`	The language – a way to override the default language from the browser
`srcNodeRef`	Reference to the DOM node with the dojoType attribute – most of the time this DOM node will be hidden or removed from the page Use this attribute with care!
`style`	The HTML style attribute

dijit.form.NumberTextBox

- Widget for the formatting of number inputs
- Usage
```
dojo.require
("dijit.form.NumberTextBox")
<input dojoType="dijit.form.NumberTextBox" >
```
- Returns the values with a dot as decimal separator
- If the language settings of the browser has a locale which uses a different decimal separator (like in Germany) Domino will throw an error!
  => dijitDominoPatch.js

dijit.form.CurrencyTextBox

- Widget for the formatting of currency inputs
- Usage

```
dojo.require
("dijit.form.CurrencyTextBox")
<input dojoType="dijit.form.CurrencyTextBox"
currency="EUR">
```

- Returns the values with a dot as decimal separator
- If the language settings of the browser has a locale which uses a different decimal separator (like in Germany) Domino will throw an error!
  => dijitDominoPatch.js

dijit.form.DateTextBox

- Widget for date formatting
- Displays a calendar on entering the field
- Usage
```
dojo.require
("dijit.form.DateTextBox")
<input dojoType="dijit.form.DateTextBox" >
```
- Returns the date in the format [Year]-[Month]-[Day]
- If the language settings of the browser has a locale which assumes a different format Domino throws an error! => dijitDominoPatch.js

dijit.form.TimeTextBox

- Widget for time formatting
- Displays a time slider on entering the field
- Usage
  ```
  dojo.require
  ("dijit.form.TimeTextBox")
  <input dojoType="dijit.form.TimeTextBox" >
  ```
- Returns the values in the format
  T[Hours(24h)]:[Minutes]:[Seconds]
- Domino could not handle that format and trows an error.
  => dijitDominoPatch.js

dijitDominoPatch.js

- Patch in development stage
- Appache licence
- Change all input fields to the right format before dojo is initialized
- On submit changes the input values back to the format Domino is expecting
- Use at your own risk!

dijitDominoPatch.js - Initialize

- Change djConfig
```
var djConfig = {
    afterOnLoad : true,
    parseOnLoad : false
};
```
- Call on load
```
onload="textBoxDominoLoadPatch();"
```
- The functions searchs for all input elements with the attribute „dojoType"
- At the end of the function the dojo parser will be called with `dojo.parser.parse();` manually.

dijitDominoPatch.js – Save (1)

- With dojo.connect connect to the onSubmit event

```
dojo.connect(document.forms[0], "onsubmit",
textBoxDominoSubmitPatch);
```

- Search for all widgets that are registered on the web page

```
dijit.registry.forEach(function(widget,
index, hash){...}
```

- Search for the input fields in that widget

```
dojo.query(„input“, widget.domNode).
```

dijitDominoPatch.js – Save (2)

- With dojo.style() search for the hidden input elements

  ```
  dojo.style(node, "display") == "none"
  ```

- Change the value with a regular expression

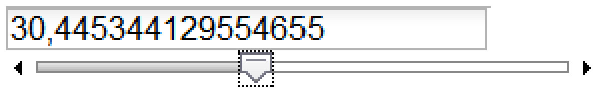
  ```
  node.value = node.value.replace(/^(.*)-(.*)-
  (.*)$/g, "$3.$2.$1");
  ```

dijit.form.ComboBox

- Widget for selecting values
- Usage
  `dojo.require ("dijit.form.ComboBox")`
  `<input dojoType="dijit.form.ComboBox" >`
- Possible to add new values
- Notes field „Dialog list"
  Option „Allow values not in list" deactivate
- In the formula for the choices add the saved value `@Trim(@Unique("Category A" : "Category B" : "Category C" : comboboxfield))`

dijit.form.Slider

- Widget to enter number values with a slider



30,445344129554655

- Better use this dijit programmatically via JavaScript instead of declarative approach
- Place an place holder div with an id on the web page
- Add also an id to the input element to get a handle to it

dijit.form.Slider - Code(1)

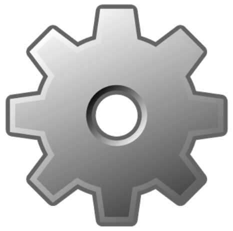
- First set the default value

```
var defaultValue = 10;
```

- Afterwards get the current value from the input element

```
if (dojo.byId("horzsliderfield").value != ""){
  defaultValue =
    dojo.byId("horzsliderfield").
    value.replace(/,/g, ".");}
```

- The value.replace call is needed because the decimal separator from Domino (at least in Germany) is a comma and the dijit is expecting a dot.



dijit.form.Slider - Code(2)

- Store a reference to the place holder div in a variable
  var sliderNode = dojo.byId("slider");

- Create a new slider object and bind it to the place holder div

```
var slider = new dijit.form.HorizontalSlider({
      name: "slider",
      value: defaultValue,
      minimum: 10,
      maximum: 60,
      intermediateChanges: true,
      style: "width:300px;",
      onChange: function(value){
        var currentValue = value+"";
        dojo.byId("horzsliderfield").value =
           currentValue.replace(/\./g, ",");
      }
  }, sliderNode);
```

Function to change the value of the input element

dijit.form.Slider - Code(3)

- Create a new div and add it as a child element to the slider div

```
var rulesNode = document.createElement('div');
sliderNode.appendChild(rulesNode);
```

- Create a new slider ruler and bind it to the new DOM node

```
var sliderRules = new dijit.form.HorizontalRule({
        count:11,
        style:"width:5px;"
    }, rulesNode);
```

DojoX - Charting

- Definition of the chart only via JavaScript possible
- Expects an array of objects as values
- A way to get the values is to write an WebQueryOpen agent which will generates the needed JavaScript code and writes it to a field „ComputedForDisplay"
- In the „HTML Head Content" the field value will be placed in a JavaScript variable

assono

Resources

Official web site
http://www.dojotoolkit.org

API Documentation
http://api.dojotoolkit.org

Demos
http://demos.dojotoolkit.org

DojoCampus
http://dojocampus.org/

Dojo Documentation
http://docs.dojocampus.org/

Old web side
http://o.dojotoolkit.org

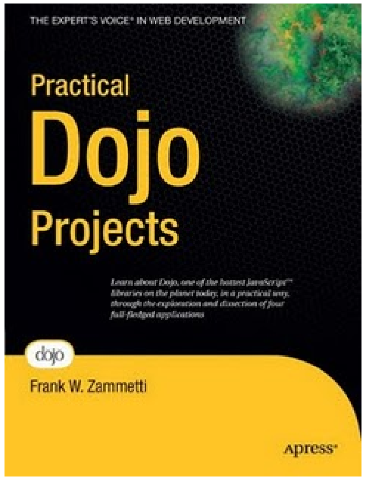
assono

Resources

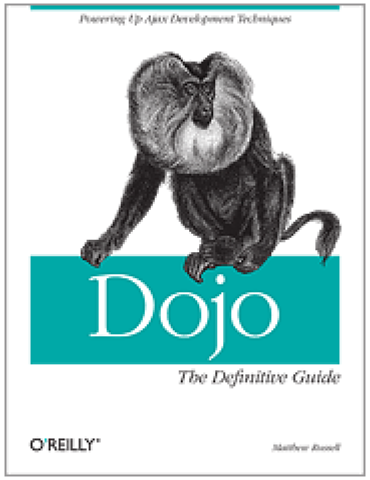
- OpenNTF CodeBin - Dojo - Easy As 123
  http://www.openntf.org/Projects/codebin/codebin.nsf/0/A2A3F3DB69E21F2A8625740E005B8EC9
- Dojomino – Dojo Domino Framework
  http://dojomino.com/
- Sitepen Labs Dojo
  http://o.sitepen.com/labs/dojo.php
- Dojo Toolbox – Adobe AIR Application with offline API
  http://o.sitepen.com/labs/toolbox/
- My Blog serie
  http://www.assono.de/blog/d6plinks/dojo

Books

- Frank W. Zammetti
  "Practical Dojo Projects"
  Apress Verlag



- Matthew A. Russell
  "Dojo – The Definitive Guide"
  O´Reilly Verlag



Questions?

Now or later:

- bhort@assono.de
- http://www.assono.de/blog
- +49.40.73 44 28-315

IT-Consulting & Solutions

Slides & demo application
http://www.assono.de/blog/d6plinks/ILUG-2010-dojo