NOTES & DOMINO ENTWICKLERCAMP

assono

dojo und Notes

EntwicklerCamp
09. März 2010

Innovative Software-Lösungen.

www.assono.de

NOTES & DOMINO ENTWICKLERCAMP

assono

Bernd Hort

Diplom-Informatiker, Universität Hamburg

seit 1995 entwickle ich Lotus Notes Anwendungen

IBM Certified Application Developer, System Administrator & Instructor

Sprecher auf diversen Konferenzen & Lotusphere 2008

bhort@assono.de
http://www.assono.de/blog
040/73 44 28-315

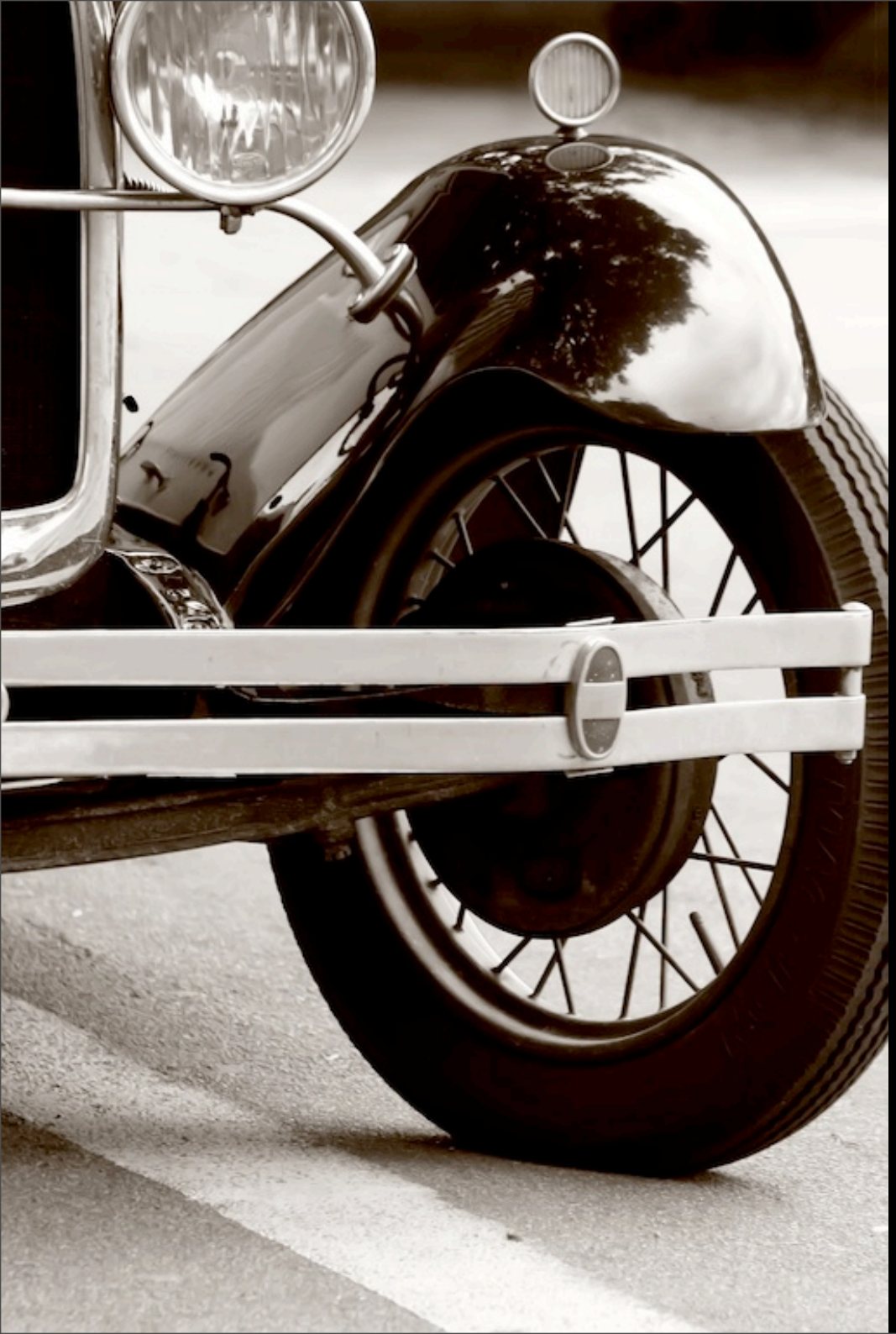
assono
IT-Consulting & Solutions

NOTES & DOMINO ENTWICKLERCAMP

assono

Agenda

- Motivation
- Einführung in dojo
- Core
- Dijit
- DojoX
- Dojo und Notes
- Dojo und XPages
- Fragen & Antworten



Das Rad nicht neu erfinden!

Motivation – Warum dojo?

Open Source

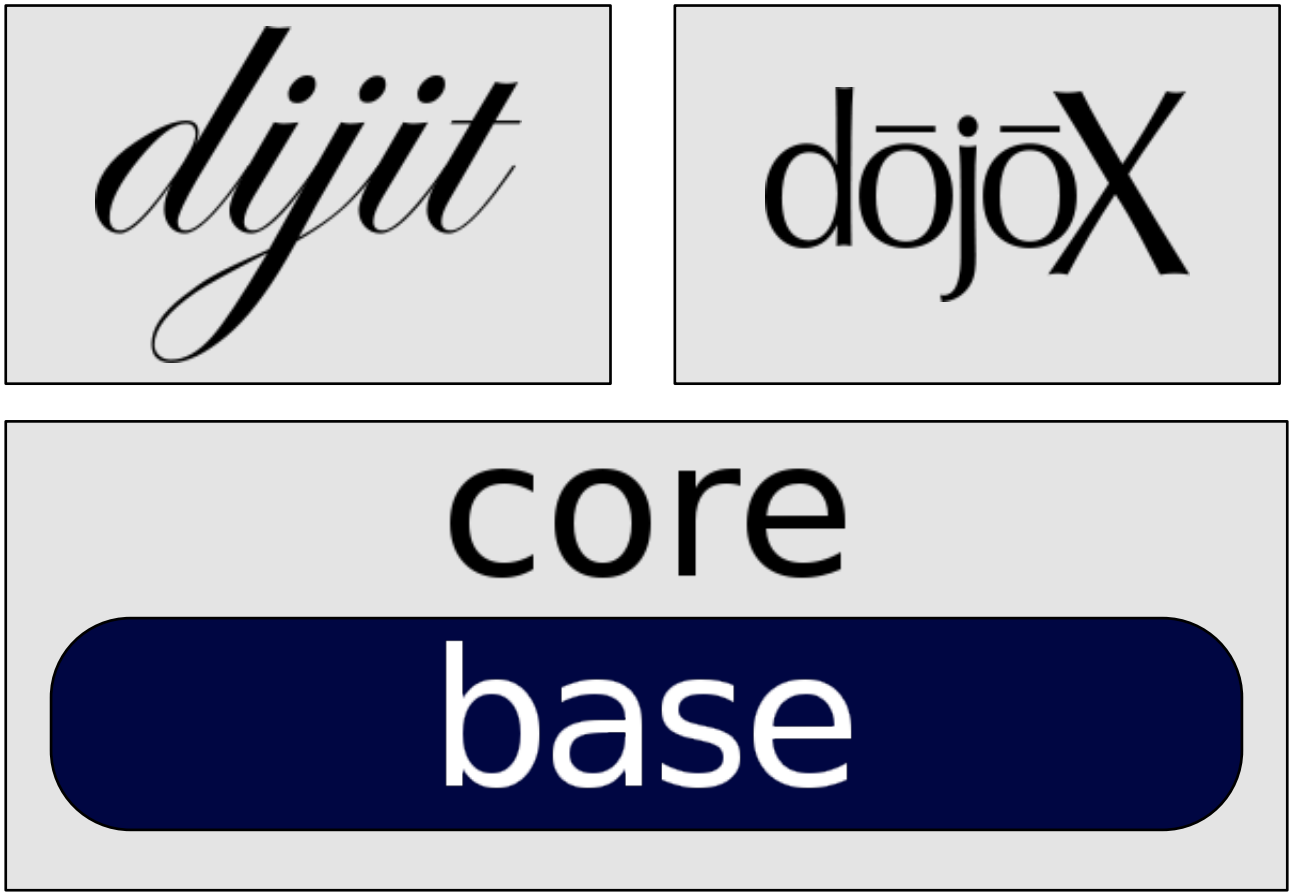
Browserunterstützung SUPPORTING

Vielfältige UI Elemente

Internationalisierung

Verwendung in XPages

Dojo besteht aus drei Teilen



core - base

- Browser-Erkennung
- JSON encoding / decoding
- Laden von Packages
- Eventhandling
- Animation effects
- AJAX
- CSS Utilities
- OOP Unterstützung
- Firebug Integration

base

core – Ergänzung

- Drag & Drop
- I18N Support
- Datumsformatierung
- Zahlenformatierung
- String Utils
- Cookie Handling
- Extended Animations
- Back Button Handling

core

dijit

- Oberflächen Elemente (interface widgets)
- „Theme“ Unterstützung
- Internationalisierung
- Keyboard Unterstützung

dojoX

- „quasi“ experimentell -
  „The future, today“
- Vieles stabil und produktiv einsetzbar
- Charts
- SVG Unterstützung
- DojoX Offline – Integration mit 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>
```

Basis CSS

Theme CSS

Basis Konfiguration muss vor dem Laden von dojo.js erfolgen!

dojo.js (Base)

Nachladen von 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"> </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 - Erklärung

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

- Simples Basis CSS – nur Abstände und Font-Definitionen

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

- **Tundra ist eines der mit gelieferten Themes**

Hello World - Erklärung

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

- Basis Konfiguration von dojo
  - Erzeugt ein djConfig Objekt
  - Das Objekt wird beim Laden von dojo.js ausgewertet
  - Eigenschaft baseScriptUri: Verzeichnis von dojo
  - Eigenschaft parseOnLoad: Analyse des HTML Codes
  - Eigenschaft extraLocale: zusätzliche Sprachen laden
  - Muss vor dem Laden von dojo.js definiert werden

Hello World - Erklärung

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

- Laden von „base“

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

- Steuern welche Packages nachgeladen werden
  - dojo.parser: Parst den HTML Code auf der Suche nach dojo relevanten Elementen
  - dijit.form.Button: Lädt das „Button“ Widget
  - dijit._Calendar: Lädt das „Calender“ Widget

Kritikpunkt – Nachladen von Packages

- Nachladen von Packages (dojo.require) bedeutet viele kleine Serveraufrufe
- Aus Performancegründen eigentlich nicht so gut
- Dojo bietet die Möglichkeit sich eine Datei mit allen benötigten Dateien zusammenzustellen
  "Custom Dojo Build"
- Blog-Eintrag von Tim Tripcony „I still have one concern about Dojo"
  http://www.timtripcony.com/blog.nsf/d6plinks/TTRY-7LLHQP
- Tim Tripcony hat ein Tool „JSFactory"
  http://www.timtripcony.com/blog.nsf/d6plinks/TTRY-7FUSZF

Hello World - Erklärung

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

- Erzeugt einen Button
  - Parser erkennt dojoType
  - „dijit.form.Button“ lädt das Button Widget
- Registriert das Event „onClick“
  - Parser erkennt 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"> </td>
  <td valign="top"
    <input id="calChinese" dojoType="dijit._Calendar" lang="zh-cn" />
  </td>
</table>
```

- Erzeugt zwei Kalenderobjekte
  - Parser erkennt dojoType
  - „dijit._Calendar“ lädt das Kalender Widget
  - über die Sprachangabe „lang“ wird die Vorgabesprache übersteuert

base – dojo.addOnLoad()

base

- Registriert eine Funktion, die aufgerufen wird, sobald das DOM geladen ist und alle Widgets intialisiert sind
- `dojo.addOnLoad(functionPointer);`
  Einfacher Funktionsaufruf
- `dojo.addOnLoad(object, "functionName");`
  Aufruf einer Methode eines Objektes
- `dojo.addOnLoad(object, function() { /* ... */});`
  Definition einer neuen Methode in dem Objekt

base – CSS Klassen

base

- `dojo.addClass(DomNode|NodeId, ClassName)`
  fügt dem DOM-Knoten eine CSS-Klasse hinzu
- `dojo.removeClass(DomNode|NodeId, ClassName)`
  entfernt eine CSS-Klasse von einem DOM-Knoten
- `dojo.hasClass(DomNode|NodeId, ClassName)`
  prüft einem DOM-Knoten eine CSS-Klasse zugewiesen wurde

base – DOM Knoten

base

- `dojo.byId(NodeId)`
  liefert den DOM-Knoten mit der gegebenen Id zurück
- `dojo.clone(Object)`
  klont ein Objekt, auch DOM-Knoten, inklusive aller Kinder
- `dojo.isDescendant(ChildDomNode, ParentDomNode)`
  prüft einem DOM-Knoten Kind eines anderen DOM-Knoten ist

base – dojo.query() (1)

base

- `dojo.query(CSS3-Selector)`
  liefert alle DOM-Knoten, die dem CSS3-Selektor entsprechen
  - Klassenname, z.B. „.foo“
  - HTML-Elemente, z.B. „span“
  - CSS Hierarchien, z.B. „table div“
  - Direkte Kindelemente (>), z.B. „#tabular_data > div“
  - Universal Selektor (*)
  - Unmittelbar vorangestellte Geschwisterknoten (~)
  - vorangestellte Geschwisterknoten (+)

base – dojo.query() (2)

base

- `dojo.query(CSS3-Selector)`
  Abfrage nach Attributen
  - [foo] Attribut ist vorhanden
  - [foo='bar'] Attributwert stimmt genau überein
  - [foo~='bar'] Attributwert stimmt mit einem Eintrag in der Liste der Werte überein
  - [foo^='bar'] Attributwert beginnt mit
  - [foo$='bar'] Attributwert endet mit
  - [foo*='bar'] Attributwert enthält

base – dojo.query() (3)

base

- `dojo.query(CSS3-Selector)`
  Pseudoklassen
  - :first-child – erstes Kind
  - :last-child – letztes Kind
  - :only-child – nur ein Kind
  - :empty – kein Kind
  - :checked – aktivierte Radiobuttons & Checkboxes
  - :nth-child(n) – n-te Kind
  - :nth-child(even) – alle „geraden“ Kinder
  - :nth-child(odd) – alle „ungeraden“ Kinder
  - :not(...) - Umkehrungen

base – OOP – declare

- `dojo.declare(className: String, superclass: Function|Function[], props: Object)`

  Erzeugt eine neue Klasse, die von einer oder mehreren Klassen erben kann und die Eigenschaftes eines Objektes übernehmen kann

base

base – OOP – declare - Beispiel

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

base – OOP – extend

- `dojo.extend(constructor: Object, props: Object...);`

  Fügt die Eigenschaften und Methoden einer oder mehrerer Klassen zu einer Klasse hinzu (Prototypische Vererbung)

base

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 – Aspektorientierte Programmierung – connect

base

- Ruft eine Methode / Funktion auf, sobald ein Event ausgelöst wurde oder eine Methode eines anderen Objektes aufgerufen wurde.

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

- Verknüpfung mit DOM-Knoten

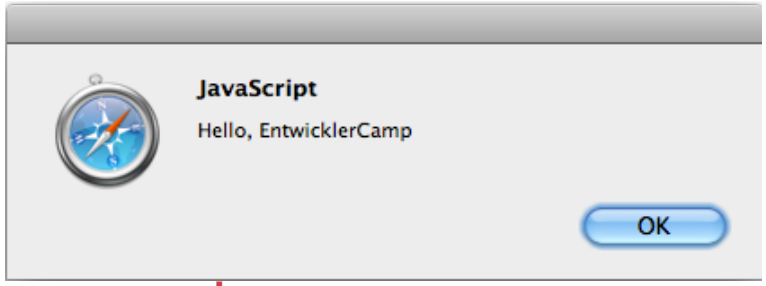
```
dojo.connect(dojo.byId(„foo“),
„onmouseover“, function(evt)
{console.log(evt)});
```

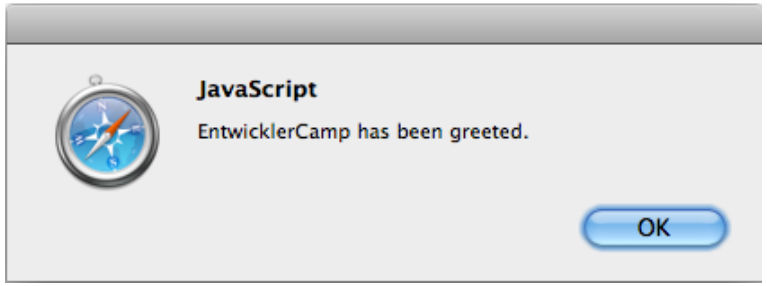
base – Aspektorientierte Programmierung – connect

- Das Event kann auch einfach ein Methodenaufruf eines anderen Objektes sein.

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 ("EntwicklerCamp");
```





base – AJAX

base

- `dojo.xhr(method: String, args: dojo.__XhrArgs, hasBody: Boolean?);`
  Startet eine AJAX Anfrage
- Unterstützte Methoden:
  "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`
  Die URL für den Aufruf
- `handleAs:String`
  Format der Serverantwort
  - text (default)
  - json
  - json-comment-optional
  - json-comment-filtered
  - javascript
  - xml

base – XhrArgs (2)

base

- `form:DOMNode`
  Die Werte des Formulars werden in der Form feld1=wert1&feld2=wert2&.... übertragen
- `content:Object`
  Die Eigenschaften des Objektes werden in der Form eigenschaft1=wert1&eigenschaft2=wert2&.... übertragen
- `headers:Object`
  Zusätzlicher HTTP Header
  Die Eigenschaften des Objektes werden als Name-Wert-Paare übertragen

base – XhrArgs (3)

base

- `load:Function`
  Funktionsreferenz, die bei erfolgreicher Serverantwort aufgerufen wird
- `error:Function`
  Funktionsreferenz, die bei fehlerhafter Serverantwort aufgerufen wird
- `handle:Function`
  Funktionsreferenz, die bei jeder Serverantwort aufgerufen wird

base – XhrArgs (4)

base

- `sync:Boolean`
  Synchronisierter Aufruf
  (Browser blockiert bis zur Antwort)
  - Vorgabe ist `false`
- `preventCache:Boolean`
  Bei `true` wird der Browser daran gehindert die Anfrage zu cachen
- `timeout:Integer`
  Millisekunden, die auf die Antwort gewartet wird vor Aufrufen der Fehlerfunktion

base – Basis Animation

base

- `dojo.animateProperty(args: dojo.__AnimArgs);`
  Basis für alle Animationseffekte

-
```
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

- Einblenden und Ausblenden

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 – Weitere Animationseffekte

core

- `dojo.require("dojo.fx");`
  Nachladen des fx-Packages
- `dojo.fx.slideTo({ node: node, left:"40", top:"50", unit:"px" }).play();`
  Verschiebt den DOM-Knoten von seiner aktuellen Position auf die angegebene Position
- `dojo.fx.wipeIn({ node: node, duration: 200).play();`
  DOM-Knoten in einem Wischeffekt erscheinen lassen
- `dojo.fx.wipeOut({ node: node, duration: 200).play();`
  DOM-Knoten in einem Wischeffekt verschwinden lassen

dojo.back – Handling des Back.Buttons

core

- Für die meisten Anwender ist der "Zurück-Knopf" der wichtigste Knopf
- dojo.back erlaubt das Setzen von Referenzpunkten
- `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

- Registrieren von Eventhandlern an beliebigen DOM-Knoten

core

- DOM-Knoten werden anhand von CSS3-Selektoren identifiziert (siehe auch 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

- Zwei Ansätze
  - Einfaches Verschieben von HTML-Elementen auf dem Bildschirm
  - Container-Definitionen zum Bestimmen von Quelle und Ziel
- Beide Ansätze sind sehr einfach umgesetzt und bis ins kleinste steuerbar

dojo.dnd.movable

core

- `dojo.require("dojo.dnd.movable");`
  Nachladen des Drag&Drop-Packages

- `<div dojoType="dojo.dnd.Moveable">`
  `Some Content</div>`
  Die Angabe des dojoTypes führt automatisch dazu, dass das HTML-Element beliebig auf dem Bildschirm verschoben werden kann.

dojo.dnd.movable – mit „Handle“

core

- Ein Kindelement innerhalb des zu bewegenden HTML-Elementes kann als “Griffpunkt“ definiert werden.

- ```
  <div dojoType="dojo.dnd.Moveable"
  handle="dragHandle">

     <textarea>Dieser Text kann editiert werden
      </textarea>

  ```
  ```

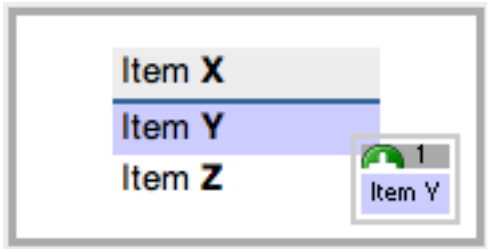
dojo.dnd.Source – Einfache Containerdefinition

core

- `dojo.require("dojo.dnd.Source");`
  Nachladen des Drag&Drop-Packages
- Definition der Liste

```
<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>
```

- Mit dem richtigen Stylesheet verbunden ergibt sich auch eine visuelle Unterstützung beim Verschieben



Weitere dojo core Packages

core

- dojo.cldr – Berücksichtigung von Länderunterschieden
- dojo.colors – Farben
- dojo.currency – Behandlung von Währungen
- dojo.date – Datumsfunktionen
- dojo.i18n – Übersetzungshilfen
- dojo.io – AJAX-ähnliche Funktionen ohne XmlHttpRequest
- dojo.number – Formartieren und Parsen von Zahlen
- dojo.regexp – Reguläre Ausdrücke
- dojo.string – Stringfunktionen

dijit - Einführung

- Dijit bezeichnet zwei Dinge
  - Das Widget Framework basierend auf den base und core Elementen
  - Ein einzelnes Widget wird auch als dijit bezeichnet
- Dijits sind voll „theme-able“ - können im Aussehen angepasst werden
  - Gewünschtes „Theme“ als CSS einbinden
  - `<body class="{Theme}">`
  - Beliebtestes „Theme“ ist „Tundra“

Zwei Arten dijits einzubinden

- Dijits können auf zwei Arten eingebunden werden
  - Deklarativ
  - Programmatisch
- Beim deklarativen Ansatz wird der HTML-Code nach Attributen „dojoType“ abgesucht
- Gefundene Elemente werden automatisch um die notwendigen JavaScript-Aufrufe ergänzt
- Vorsicht! HTML Validierungs-Services melden den HTML-Code als nicht valide!

Dijits deklarativ einbinden


- ```
  var djConfig = {
      baseScriptUri : "js/dojo/",
      parseOnLoad : true
  };
  ```
  Konfiguration zum Parsen des HTML-Codes beim Laden

- `dojo.require("dojo.parser");`
  Notwendiges Package zum Parsen

- `dojo.require("dijit.form.Button");`
  Das zum dijit gehörige Package laden

- `<button dojoType="dijit.form.Button">ok</button>`
  Dojo Attribute hinzufügen
  ```

Dijit programmatisch einbinden

- `dojo.require("dijit.form.Button");`
  Das zum dijit gehörige Package laden

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

- Der programmatische Ansatz ist der performantere Ansatz.
  - Nur bei großen HTML-Dokumenten wichtig
  - Oder bei hoch dynamischen Webseiten
  ```

Zugriff auf dijits

- Jedes dijit wird vom dijit Manager verwaltet
- Jedes dijit hat eine eindeutige Id
  - Entweder manuell definiert
  - Oder automatisch generiert
- Die Methode `dijit.byId(<someID>);` liefert das dijit Objekt zurück
- Nicht zu verwechseln mit `dojo.byId(<someID>);`
  - Die Methode liefert den DOM-Knoten zurück

dijit Methoden

Methode	Beschreibung
`buildRendering`	Konstruiert die visuelle Repräsentation
`connect`	Verbindet ein Object / Event mit entsprechenden Methoden des dijit und registriert für `disconnect()`
`create`	Startet den Lebenszyklus des dijits
`destroy`	Zerstört das dijit, lässt aber die Kindelemente intakt
`destroyDescendants`	Zerstört alle Kindelemente rekursiv
`destroyRecursive`	Zerstört rekursiv das dijit und alle Kindelemente

dijit Methoden

Methode	Beschreibung
`destroyRendering`	Entfernt alle DOM-Knoten des dijit
`disconnect`	Löst die Verbindungen zu registrierten Objekten / Events
`getDescandants`	Liefert alle Kindelemente
`isFocusable`	Ermittelt ob das dijit auswählbar ist
`isLeftToRight`	Ermittelt die Leserichtung
`onBlur`	Wird aufgerufen, wenn das dijit den Fokus verliert
`onClose`	Wird aufgerufen, wenn das dijit zerstört wird

dijit Methoden

Methode	Beschreibung
`onFocus`	Wird aufgerufen, wenn das dijit den Fokus erhält
`postCreate`	Wird aufgerufen, wenn die DOM-Knoten des dijits erzeugt wurden
`postMixInProperties`	Wird aufgerufen, nachdem alle Parameter verarbeitet wurde, aber bevor die DOM-Knoten erzeugt werden
`setAttrbute`	Setzt HTML Attribute direkt
`startUp`	Wird aufgerufen, nachdem alle dijit Kinder erzeugt wurden, aber bevor sie dargestellt werden

dijit Attribute

Attribut	Beschreibung
`attributeMap`	Liste aller Attribute, vornehmlich HTML Attribute
`class`	HTML Klassen Attribute
`id`	Die ID des dijit
`lang`	Die Sprache – erlaubt das Übersteuern der automatisch bestimmten Sprache
`srcNodeRef`	Referenz auf den DOM-Knoten mit dem dojoType Attribute Der DOM-Knoten wird häufig ersetzt.
`style`	Die HTML Style Attribute

dijit.form.NumberTextBox

- Widget für die Formatierung von Zahlenangaben

- Einbindung

```
dojo.require
("dijit.form.NumberTextBox")
<input dojoType="dijit.form.NumberTextBox" >
```

- Speichert Zahlenangabe mit „.“ als Dezimaltrennzeichen

- In der Standardausführung können die eingegebenen Werte nicht von Domino verarbeitet werden.
  => dijitDominoPatch.js

dijit.form.CurrencyTextBox

- Widget für die Formatierung von Währungsangaben
- Einbindung

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

- Speichert Zahlenangabe mit „.“ als Dezimaltrennzeichen
- In der Standardausführung können die eingegebenen Werte nicht von Domino verarbeitet werden.
  => dijitDominoPatch.js

dijit.form.DateTextBox

- Widget für die Datumsauswahl
- Einbindung

```
dojo.require
("dijit.form.DateTextBox")
<input dojoType="dijit.form.DateTextBox" >
```

- Speichert Datumsangabe im Format
  [Jahr]-[Monat]-[Tag]
- In der Standardausführung können die eingegebenen Werte nicht von Domino verarbeitet werden.
  => dijitDominoPatch.js

dijit.form.TimeTextBox

- Widget für die Zeitauswahl
- Einbindung

```
dojo.require
("dijit.form.TimeTextBox")
<input dojoType="dijit.form.TimeTextBox" >
```

- Speichert Zeitangabe im Format
  T[Stunden(24h-Basis)]:[Minuten]:[Sekunden]
- In der Standardausführung können die eingegebenen Werte nicht von Domino verarbeitet werden.
  => dijitDominoPatch.js

dijitDominoPatch.js

- Patch im Entwicklungsstadium
- Wandelt Inputfelder vor der Initialisierung von dojo in das richtige Format um.
- Formatiert die Eingabe vor der Übertragung auf den Domino Server wieder in das von Domino erwartete Format um.
- Benutzung ohne Gewähr!

dijitDominoPatch.js - Initailisierung

- Anpassung djConfig

```
var djConfig = {
    afterOnLoad : true,
    parseOnLoad : false
};
```

- Aufruf von

```
onload="textBoxDominoLoadPatch();"
```

- Die Funktion untersucht alle Input-Elemente auf das Attribut „dojoType“
- Am Ende der Funktion wird mit

```
dojo.parser.parse();
```

der Parser manuell aufgerufen.

dijitDominoPatch.js – Speichern (1)

- Mit dojo.connect mit dem onSubmit-Event verbinden

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

- Alle auf der Webseite registrierten Widgets untersuchen

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

- Bei entsprechenden Widgets mit

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

die Input-Felder suchen.

dijitDominoPatch.js – Speichern (2)

- Mit dojo.style() nach dem versteckten Input-Element suchen

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

- Den Inhalt neu formatieren

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

dijit.form.ComboBox

- Widget für die Zeitauswahl
- Einbindung

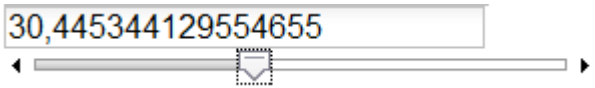
```
dojo.require
("dijit.form.ComboBox")
<input dojoType="dijit.form.ComboBox" >
```

- Einfügen auch neuer Werte!
- Notesfeld „Dialogliste“
  Option „Neue Werte zulassen“ deaktivieren

- In der Auswahlliste „Formel verwenden“ und gespeicherten Wert hinzufügen

```
@Trim(@Unique("Category A" : "Category B" : "Category C" :
comboboxfield))
```

dijit.form.Slider

- Widget für die Eingabe einer Zahl über einen Slider 

- Beispiel, wo es sinnvoller ist das Widget über JavaScript einzubinden
- Platzhalter-Div mit ID an gewünschter Stelle positionieren
- Das Input-Element zur einfacheren Ansprache auch mit einer ID versehen

dijit.form.Slider - Code(1)

- Zunächst den Vorgabewert bestimmen

```
var defaultValue = 10;
```

- Dann den aktuellen Wert aus dem Input-Element bestimmen

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

- Das value.replace ist notwendig, weil in dem Input-Element das Dezimaltrennzeichen ein Komma ist, aber ein Punkt erwartet wird.

dijit.form.Slider - Code(2)

- Den Platzhalter-Knoten einer Variablen zuweisen

  var sliderNode = dojo.byId("slider");

- Neues Sliderobjekt erzeugen und an Platzhalter-Div anbinden

```
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);
```

Funktion zum Aktualisieren des Input-Elementes

dijit.form.Slider - Code(3)

- Neues Div erzeugen und an das Slider-Div als Kind anhängen

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

- Neues Sliderruler erzeugen und an neuen Dom-Knoten anbinden

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

DojoX - Charting

- Definition des Chart nur über JavaScript möglich
- Erwartet als Daten ein Array von Objekten mit den Werten
- Bestimmen der Werte über ein WebQueryOpen-Agent, der entsprechenden JavaScript-Code generiert und ein Feld „ComputedForDisplay“ speichert
- Im „HTML Head Content“ den Feldinhalt in JavaScript-Variable schreiben

Ressourcen

Offizielle Webseite
http://www.dojotoolkit.org

API Dokumentation
http://api.dojotoolkit.org

Demos
http://demos.dojotoolkit.org

DojoCampus
http://dojocampus.org/

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

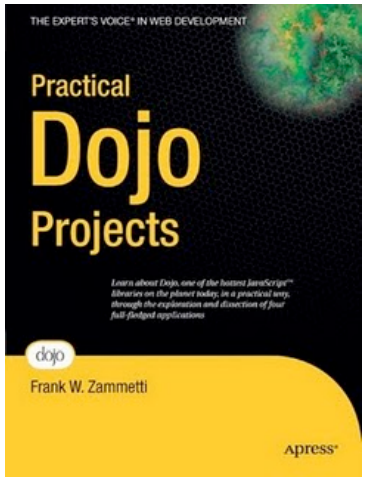
Alte Webseite
http://o.dojotoolkit.org

Ressourcen

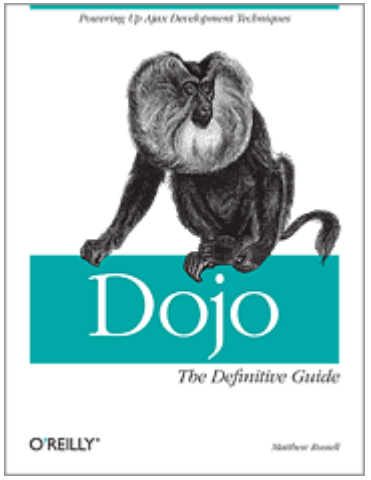
- OpenNTF CodeBin - Dojo - Easy As 123
  http://www.openntf.org/Projects/codebin/codebin.nsf/0/A2A3F3DB69E21F2A8625740E005B8EC9
- Dojomino – Dojo Domino Framework
  http://dojomino.com/
- Sitepen Labs Dojo
  http://o.sitepen.com/labs/dojo.php
- Dojo Toolbox – Adobe AIR Anwendung mit Offline API
  http://o.sitepen.com/labs/toolbox/

Bücher

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



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



Fragen?

jetzt stellen – oder später:

bhort@assono.de
http://www.assono.de/blog
040/73 44 28-315

assono
IT-Consulting & Solutions

Folien & Beispieldatenbank unter
http://www.assono.de/blog/d6plinks/EntwicklerCamp-2010-dojo