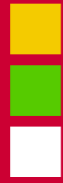


Rich Internet Application development using the Dojo Toolkit

FrOSCon, Sankt Augustin, 20. August 2011
Dr. Alexander Kläser
[klaeser at univention dot de](mailto:klaeser@univention.de)



Goal of this presentation

- ❖ Brief overview of existing JavaScript frameworks
- ❖ General overview of Dojo (structure, features, resources)
 - ❖ Know what you can do with it and whether it could help you
- ❖ Some details w.r.t. Rich Internet Application (RIA) development using Dojo



Presentation outline

- Overview over JavaScript Frameworks
- Overview over the Dojo Toolkit
- Dojo core features
- Dojo and Widgets

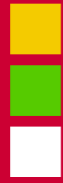




Presentation outline

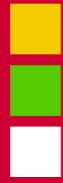
- Overview over JavaScript Frameworks
- Overview over the Dojo Toolkit
- Dojo core features
- Dojo and Widgets





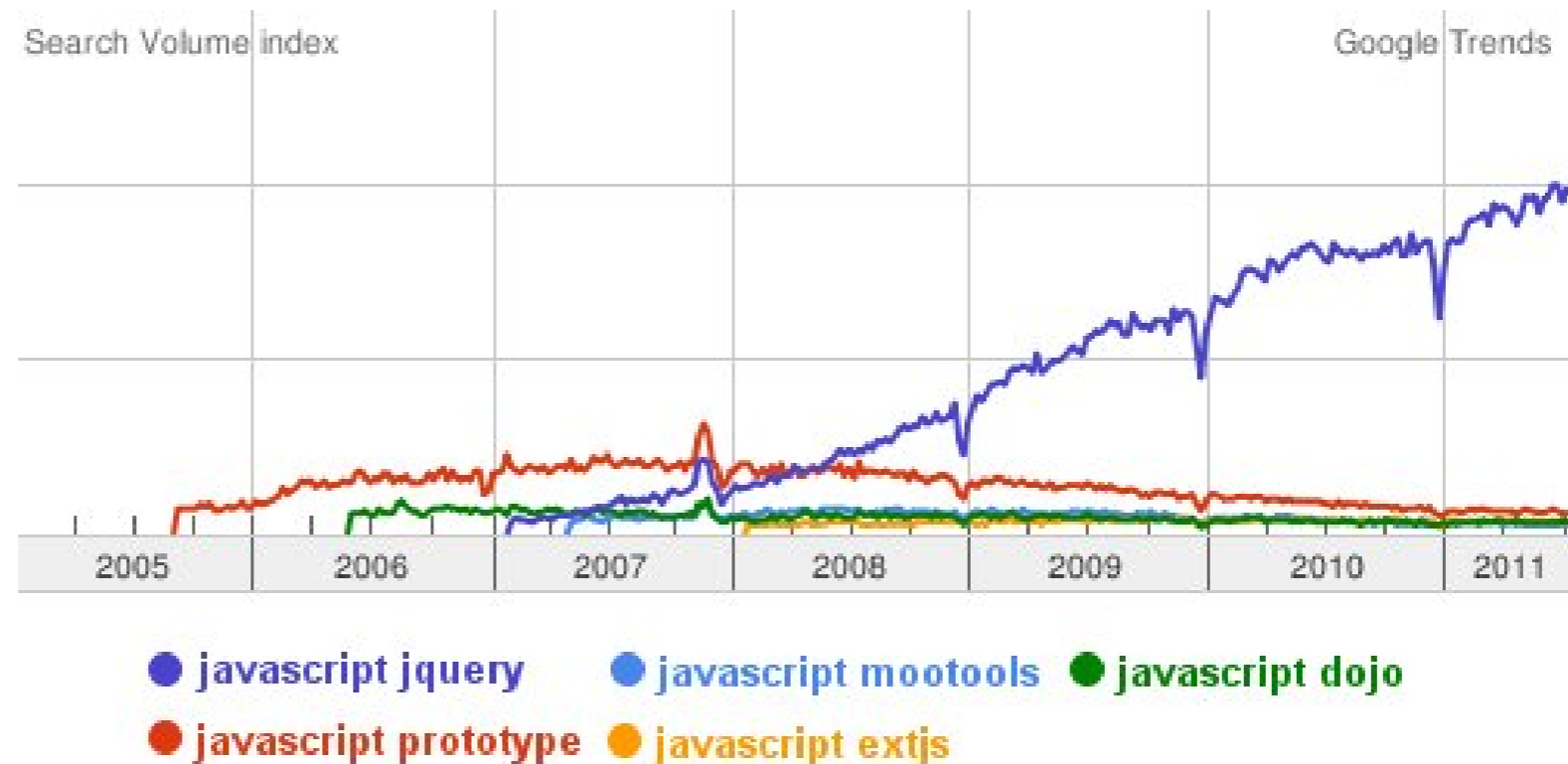
Our situation @Univention

- ❖ Redesign of our administrative web frontend... what it can do:
 - ❖ Domain wide configuration: users, groups, computers, ACLs, DNS, DHCP, mail, shared folders, printers, Nagios, policies, ...
 - ❖ Computer wide configuration: system statistics, software management, process overview, ...
 - ❖ Virtual machine management
 - ❖ Thin client services
 - ❖ Management system for school environments (class rooms, ...)
 - ❖ ...
- ❖ Solution: rewrite the client side part as RIA (w/AJAX technologies)
 - ❖ Communication with existing server part (written in Python)
 - ❖ A JavaScript library simplifies cross-browser compatibility

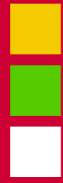


JavaScript libraries overview

- ❖ There are many (OpenSource) JavaScript libraries:
 - ❖ Prototype, JQuery, Yahoo UI, Dojo, Mootools, ExtJS, ...



- ❖ **Note:** "JavaScript Prototype" is not necessarily the library
- ❖ Is the winner JQuery?



Our conclusions

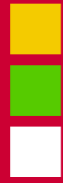
- JQuery is a very popular library
 - Great for manipulating the DOM
 - But, no ready-to-use infrastructure for RIA (see below)
- ExtJS and Dojo seem to be the only libraries that offer a rich feature set:
 - Many widgets, consistent API, module management, i18n, l10n, layout management, DOM manipulation, data abstraction, OOP, theming, data grids, build system etc.
- ExtJS
 - Many features and widgets, a very good documentation with tutorials
 - Development is closed (product of the company Sencha)
 - Dual licensing model (commercial + LGPL)
- Dojo
 - Open development
 - License: Academic Free License v2.1 and modified BSD license
 - Non-profit Dojo Foundation holds intellectual property rights
- ... so the winner is Dojo... (at least in our case)
- See also: <http://dojotoolkit.org/reference-guide/quickstart/introduction/whydojo.html>



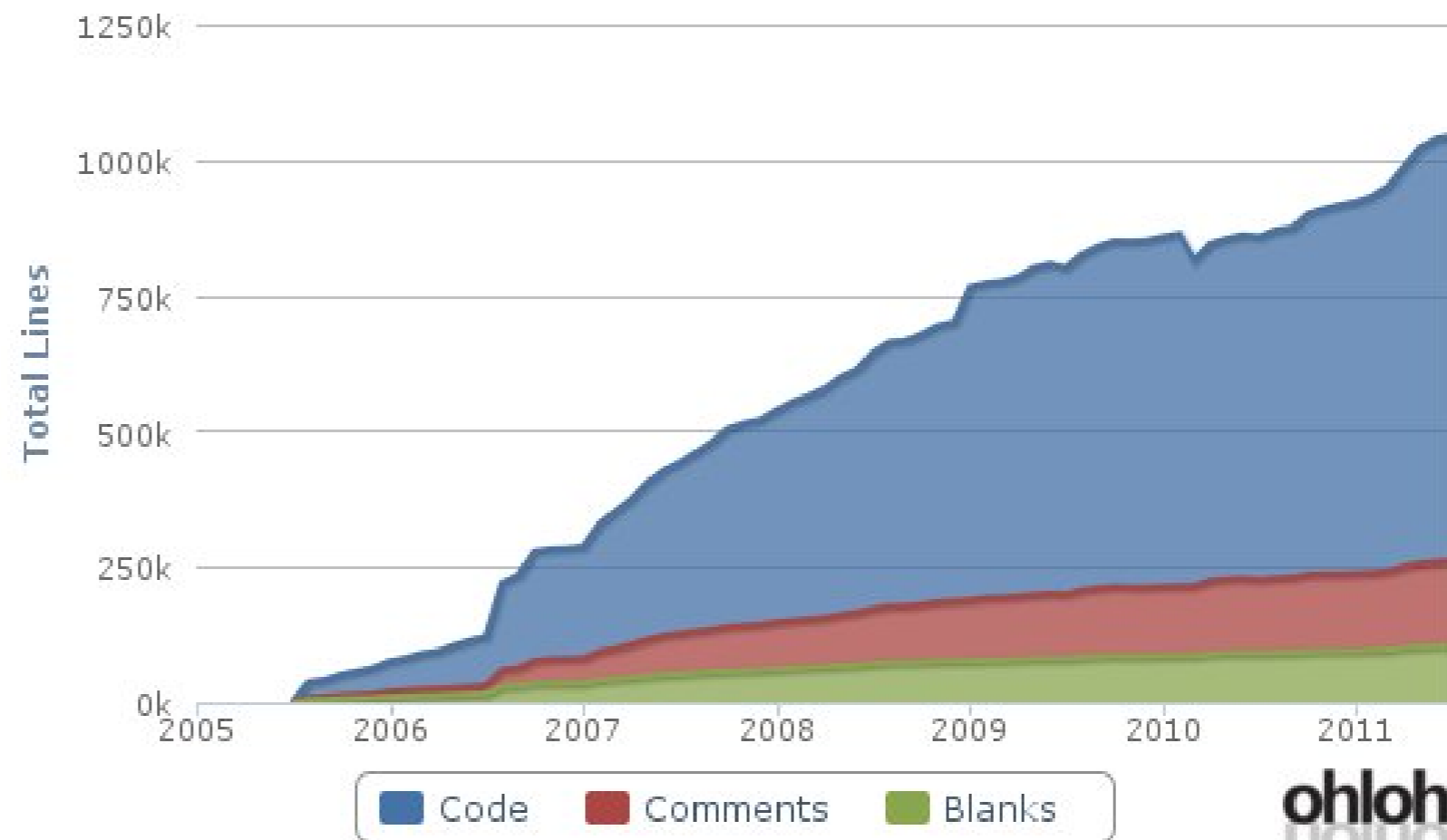
Presentation outline

- Overview over JavaScript Frameworks
- Overview over the Dojo Toolkit
- Dojo core features
- Dojo and Widgets

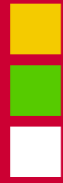




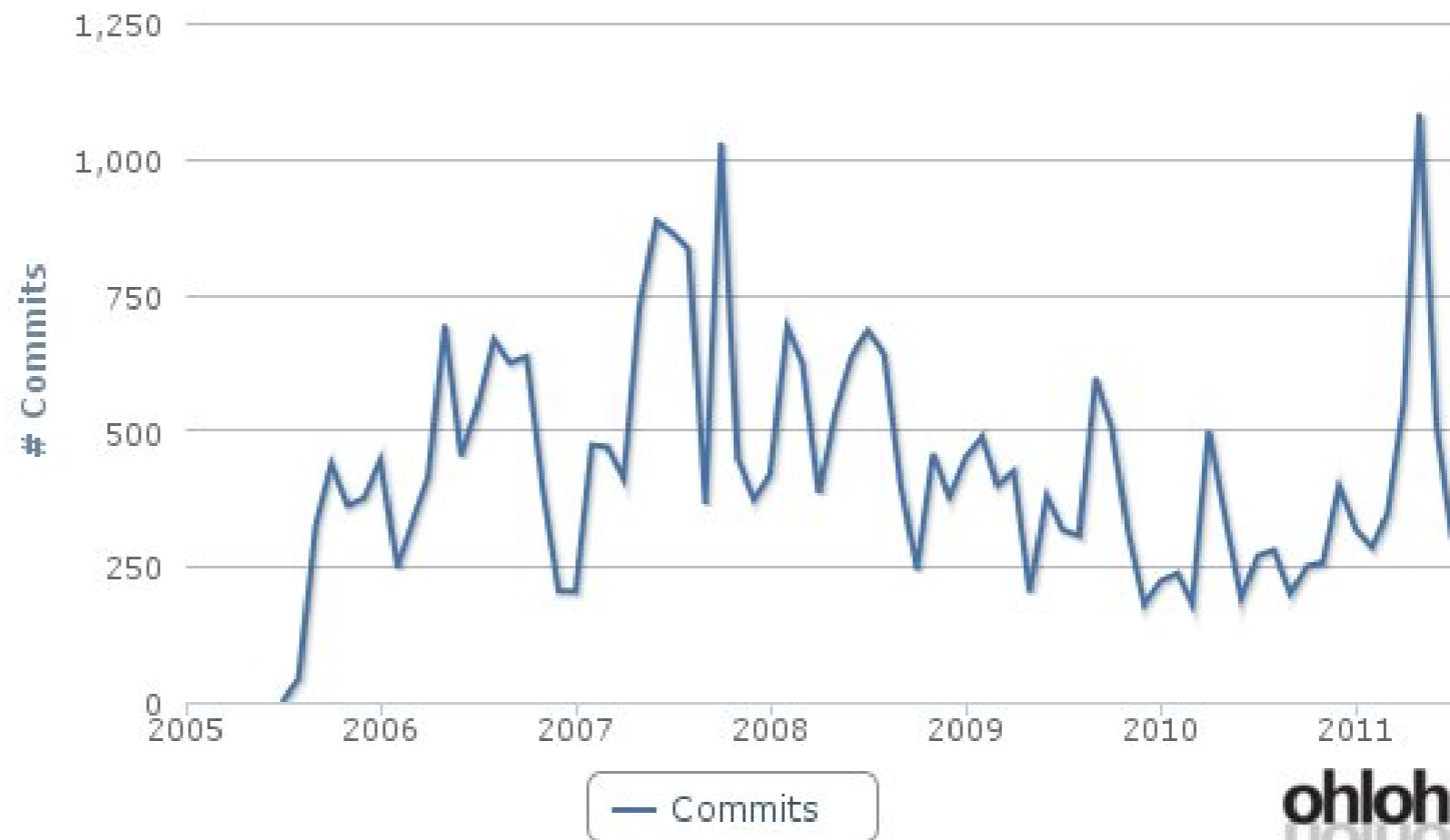
Statistics about Dojo (lines of code)



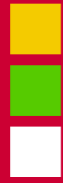
ohloh
40140



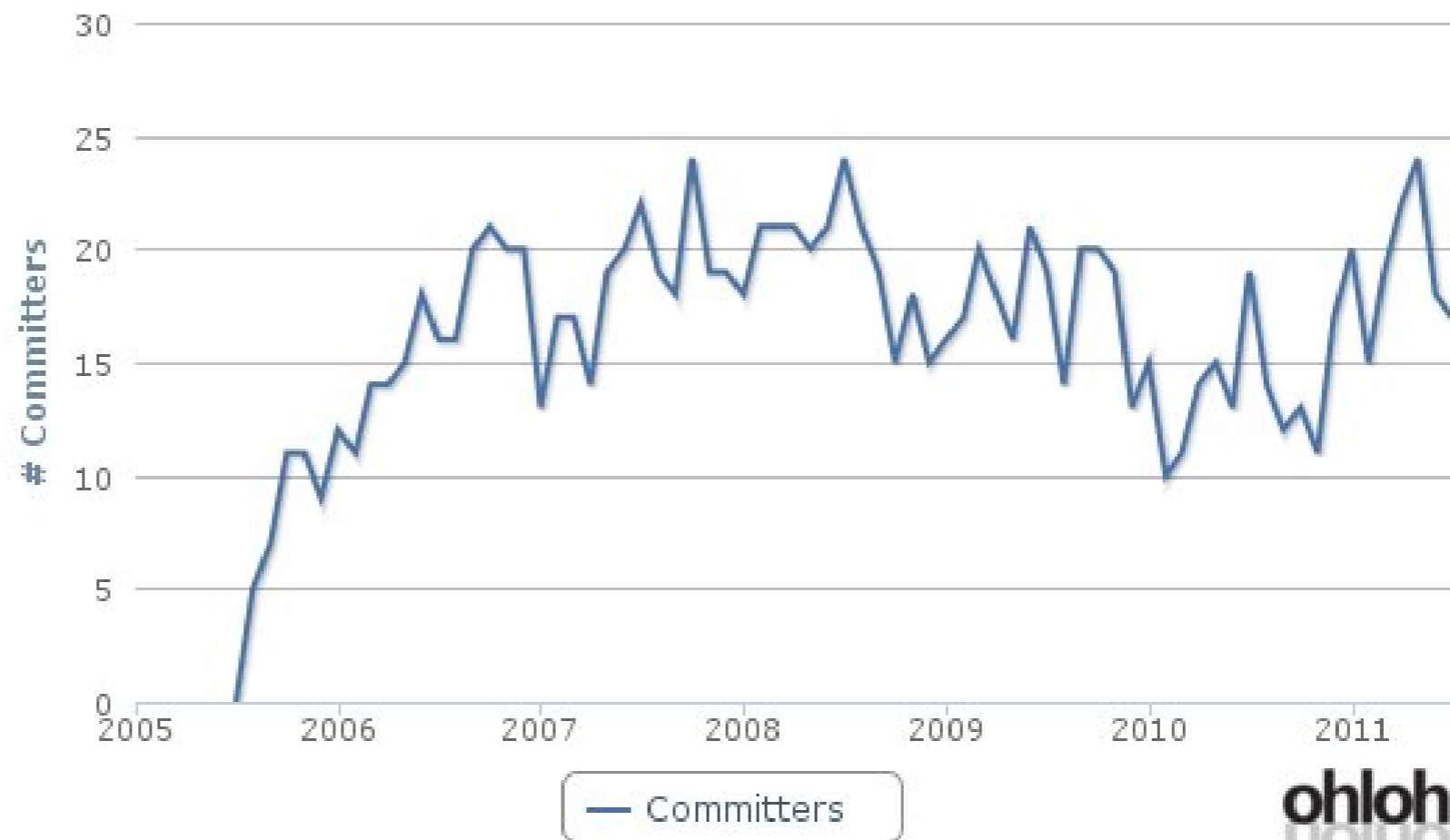
Statistics about Dojo (commits per month)



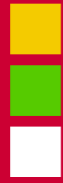
ohloh
40140



Statistics about Dojo (committers per month)

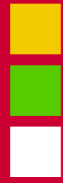


ohloh
40140



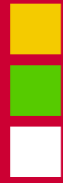
Dojo overview

- ❖ Dojo consists of 3 main parts...
- ❖ **dojo**: the core
 - ❖ Browser normalization, module loading, DOM manipulation, events, string functions, data access, drag 'n' drop, XHR calls, JSON encoding/decoding, effects/animations, OOP etc.
- ❖ **dijit**: interface widgets, advanced UI controls, template driven
 - ❖ Form elements (calendar, dynamic combo boxes, slider etc.), dialogs, tree, layout, menus, tooltips, WYSIWYG editor etc.
- ❖ **dojox**: various extra projects, partly experimental
 - ❖ More widgets, charts, data grid, file uploads, dynamic CSS rules, collections, syntax highlighting, sprintf, more editor plugins, more animation effects, JSON schema validation, JSON query syntax etc



Dojo philosophy

- ❖ Non-intrusive
 - ❖ Native objects (e.g., Array, String) are not extended with additional functionality
 - ❖ Instead extensions are provided in a clean separate namespace
 - ❖ When possible, these extensions (functions) refer to existing, native implementations of the browser
- ❖ Ease of use and performance
- ❖ Code that is as simple as possible
- ❖ High-quality "full-stack" library
- ❖ Dojo is modular, modules are loaded when need



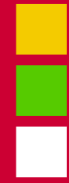
How to use Dojo

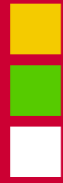
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>Tutorial: Hello Dojo!</title>
  <!-- load Dojo and style sheets for widgets -->
  <script src="...dojo/dojo.js"></script>
  <link rel="stylesheet" type="text/css"
href="...dojo/resources/dojo.css" />
  <link rel="stylesheet" type="text/css"
href="...dijit/themes/claro/claro.css" />
  <!-- ... eventually other widget specific CSS files ... -->
</head>
<body class="claro">
  <h1 id="greeting">Hello</h1>
</body>
</html>
```



Presentation outline

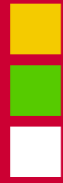
- Overview over JavaScript Frameworks
- Overview over the Dojo Toolkit
- Dojo core features
- Dojo and Widgets





Dojo Base

- ❖ **Base:** no additional module needs to be loaded
- ❖ Execute JS functions when then DOM is ready: `dojo.addOnLoad()` / `dojo.ready()`
- ❖ Packaging mechanism: `dojo.require()`, `dojo.provide()`
 - ❖ Separation of code, usage of namespaces improves reusability :)
 - ❖ Namespaces: `dojo.foo.bar` → `dojo/foo/bar.js`
 - ❖ Browser can load modules on demand → one module, one file
 - ❖ No need to track dependencies (a module can load another module)
 - ❖ A module is just a file, otherwise there is no restriction (may declare any kind of class)
 - ❖ *Modules in dojo/dijit are already compatible with the AMD API (Asynchronous Module Definition), see:*
<http://wiki.commonjs.org/wiki/Modules/AsynchronousDefinition>



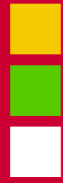
Dojo Base

(packaging mechanism)

```
// some code...
dojo.require ("my.Module"); // → my/Module.js
dojo.addOnLoad(function() {
    var m = new my.Module();
    // ...
});

// in the package my/Module.js
dojo.provide ("my.Module");
dojo.require ("my._ClassBase"); // → my/_ClassBase.js

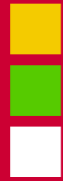
dojo.declare ("my.Module", my._ClassBase, {
    // ... the code ...
});
```



Dojo Base

(object oriented programming)

- ❖ Dojo provides tools to make object oriented programming in JS easier (encapsulating the prototype)
 - ❖ Declaring classes w/inheritance: `dojo.declare()`
 - ❖ Call to parent method: `this.inherited(arguments)`
 - ❖ Extend existing classes: `dojo.extend()`
 - ❖ Mixins: `dojo.mixin()`, `dojo.delegate()`
- ❖ Use of mixins
 - ❖ A mixin provides certain functionality (methods, variables) via inheritance
 - ❖ Mixin cannot be instantiated alone
- ❖ For multiple inheritance
 - ❖ Dojo uses C3 linearization (as Python, Perl) to resolve the order in which methods are inherited



Dojo Base

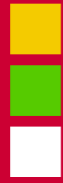
(object oriented programming 2)

```
dojo.declare("some.Child", [ some.Parent, some.OtherParent ],
{
    myVal: "default value",

    constructor: function(args) {
        // parent constructor is automatically called
        dojo.mixin(this, args);
    }

    myMethod: function() {
        // call overloaded parent method via this.inherited()
        this.inherited(arguments);
        // mycode...
    }
});

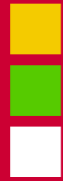
var a = new some.Child();
var b = new some.Child({ myVal:"x" });
// a.myVal == "default value"
// b.myVal == "x"
```



Dojo Base

(signals/events, language tools)

- ❖ Events, signal handling
 - ❖ Normalization of events
 - ❖ Signal mechanism for events/function calls: `dojo.[dis]connect()`
 - ❖ Publish/subscribe mechanism: `dojo.publish()`, `dojo.subscribe()`
- ❖ Language tools
 - ❖ Scoping & function parameters: `dojo.hitch()`, `dojo.partial()`
 - ❖ Type checking: `dojo.is[String,Array,Function,Object,Alien]()`



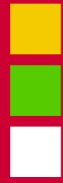
Dojo Base

(signals/events, language tools 2)

```
obj = {
  foo: function() { console.log('obj.foo()'); },
  bar: function() { console.log('obj.bar()'); },
  val: 42
};
dojo.connect(obj, 'foo', obj, 'bar');
dojo.connect(obj, 'foo', dojo.hitch(obj, function() {
  // we are in the scope of 'obj'
  console.log('signal for obj.foo()');
  console.log('this.val=' + this.val); // this.val == 42
}));
obj.foo();

/* output:
obj.foo()
obj.bar()
signal for obj.foo()
this.val=42 */
/* full code: http://jsfiddle.net/rTUfr/1/ */

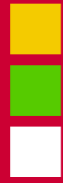
// ... it is also possible to connect to DOM-events
dojo.connect(domNode, "onclick", dojo.hitch(this, "handleClick"));
```



Dojo Base

(utilities for arrays, strings, and others)

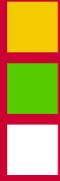
- ❖ Array utility functions:
 - ❖ Iteration: `dojo.forEach()`, `dojo.map()`
 - ❖ Filtering: `dojo.filter()`
 - ❖ Condition check: `dojo.every()`, `dojo.some()`
 - ❖ Index of a given element: `dojo.indexOf()`
- ❖ String processing:
 - ❖ `dojo.trim()`
 - ❖ A template mechanism: `dojo.replace()`
- ❖ JSON conversion: `dojo.toJson()`, `dojo.fromJson()`
- ❖ Browser Sniffing: `dojo.isIE < 7`, `isFF`, `isWebKit` ...



Dojo Base

(Ajax calls, dojo.Deferred)

- ❖ Ajax wrapper functions: `dojo.xhr[Get,Post,Delete,Put]()`
- ❖ Handling of asynchronous events: `dojo.Deferred`
 - ❖ Asynchronous operation abstraction
 - ❖ Return a `dojo.Deferred` if the function is asynchronous
 - ❖ Multiple `dojo.Deferred` can be nested and chained



Dojo Base

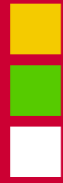
(Ajax calls, dojo.Deferred 2)

```
// xhrPost() returns an object of type dojo.Deferred
var deferred = dojo.xhrPost({
    url: '/echo/json/',
    handleAs: 'json',
    content: { /* ... a dict of name/string pairs ... */ }
});

// register callback... then() returns a new deferred
deferred = deferred.then(function(data) {
    var str = '';
    dojo.forEach(data.messages, function(i) {
        str += '<div>' + i + '</div>';
    });
    dojo.body().innerHTML = str;
    return data.value; // return value is fed to next deferred
});

// deferreds can be chained
deferred.then(function(value) {
    dojo.body().innerHTML += '<div>Value: ' + value + '</div>';
});

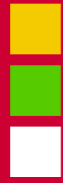
/* full code: http://jsfiddle.net/hMREq/5/ */
```



Dojo Base

(DOM/CSS manipulation, animation)

- ❖ Querying DOM nodes
 - ❖ With standard CSS3 selectors: `dojo.query()`
 - ❖ By their ID: `dojo.byId()`
 - ❖ Chaining of DOM manipulation: `dojo.NodeList`
- ❖ Placement in the DOM: `dojo.place()`
- ❖ Manipulation of DOM attributes: `dojo.attr()`
- ❖ Manipulation of CSS style: `dojo.style()`
- ❖ Manipulation of CSS classes: `dojo.[add,remove,has]Class()`
- ❖ Manipulation/querying of DOM node size/position: `dojo.coords()`, `dojo.position()`, `dojo.[margin,content]Box()`
- ❖ Animation: `dojo.fadeIn/Out()`, `dojo.animateProperty()`



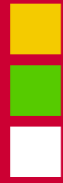
Dojo Base

(DOM/CSS manipulation, animation 2)

```
// dojo.query uses standard CSS3 selectors
// it returns a dojo.NodeList, a subclass of Array
dojo.query("ul > li").forEach(function(n) { ... });
dojo.query("#mycontainer").addClass("mylist");
dojo.query("a").onclick(function() { ... });
dojo.query".styleA.styleB").map(function(n) { ... });

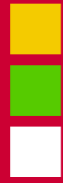
// calls can be chained
dojo.query".container")
    .addClass('newClass')
    .fadeIn().play();

// other kinds of manipulation
dojo.style(domNode, "height", "50px");
dojo.place("<p>Hi</p>", dojo.body());
dojo.addClass(domNode, "myCssClass");
var width = dojo.marginBox(domNode).w;
dojo.marginBox(domNode, {w: 300, h: 400});
dojo.fadeIn({ node: node }).play();
```



Dojo core

- ❖ **Core:** everything else below `dojo.*` that needs to be loaded via `dojo.require()`
- ❖ Dojo's Dom/Widget parsing package: `dojo.parser` (→ see `dijit`)
- ❖ Effects library on top of base animations: `dojo.fx`. [`sizeTo()`, `slideBy()`, `crossFade()`, `wipeTo()`, `smoothScroll()`]
- ❖ Utility classes for internationalization: `dojo.i18n`
- ❖ More string functions: `dojo.string`
- ❖ Caching of inline text: `dojo.cache()`
- ❖ HTTP cookie manipulation: `dojo.cookie()`
- ❖ Data access/manipulation layer: `dojo.store` (obsolete API is `dojo.data`)
- ❖ ...



Build system

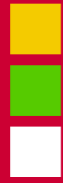
- ❖ Works transparently with Package System
- ❖ Collects all referenced JS module files of an Dojo application and builds a single minified JS file
 - ❖ Minification: Comments, whitespace, newlines are removed, local variables names replaced by short ones
- ❖ ...



Presentation outline

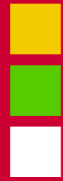
- Overview over JavaScript Frameworks
- Overview over the Dojo Toolkit
- Dojo core features
- Dojo and Widgets





Dijit

- ❖ dijit = Dojo widgets
- ❖ A widget encapsulates the DOM representation and offers a convenient object-oriented interface
 - ❖ Events, attribute settings, GUI logic, ...
- ❖ Widgets are created in a programmatic or a declarative manner
- ❖ Types of widgets:
 - ❖ Layout widgets (border, tabs, accordion, ...)
 - ❖ Form widgets w/validation (drop down, date pickers, popup, ...)
 - ❖ Dialogs, Menus, ...
- ❖ Support for i18n
- ❖ Widgets are themeable



Screenshot

Popups and Alerts

File Edit View Themes Dialogs TextBox Padding Help Disabled Click me!

Tree

- Continent
- Africa
- Asia
 - China
 - India
 - Russia
 - Mongolia
- Oceania
- Europe
 - Germany
 - France
 - Spain

Rootless Tree

Calendar

Color Picker

Log globals

Log widgets

✖ Destroy All

Buttons can do an action, display a menu, or both:

Enabled:

Disabled:

CheckBox

unchecked checked disabled disabled and checked

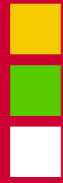
Radio Buttons

news talk weather (disabled)

I am the last Tab

Info Alternate Themes Bottom 3 x

See: <http://download.dojotoolkit.org/release-1.6.1/dojo-release-1.6.1/dijit/themes/themeTester.html>



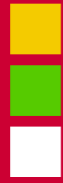
Widgets from markup (declarative)

```
<div dojoType="dijit.layout.TabContainer" style="width:
100%; height: 100%;">
  <div dojoType="dijit.layout.ContentPane" title="Tab 1"
selected="true">
    Content tab 1
  </div>
  <div dojoType="dijit.layout.ContentPane" title="Tab 2">
    Content tab 2
  </div>
</div>

/* specify 'djConfig="parseOnLoad:true"' in script tag... */

/* ... after loading, Dojo will parse the DOM tree */

/* full code: http://jsfiddle.net/H4DXH/ */
```

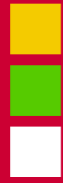


Widgets from code (programmatic)

```
dojo.require("dijit.layout.TabContainer");
dojo.require("dijit.layout.ContentPane");
dojo.addOnLoad(function() {
    // Create outer tab container
    var container = new dijit.layout.TabContainer({
        style: 'height: 100%; width: 100%;'
    });
    container.placeAt(dojo.body());

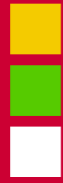
    // Construct tab content panes
    dojo.forEach([1, 2], function(i) {
        container.addChild(new dijit.layout.ContentPane({
            title: 'Tab ' + i,
            // 'content' can be HTML, DOM-node, or Dojo-widget
            content: 'Content tab ' + i
        }));
    });
    container.startup();
});

/* Extensive use in RIA: dynamic creation/manipulation of widgets */
/* full code: http://jsfiddle.net/ghZxW/10/ */
```



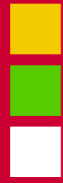
Special widget classes

- ❖ Base class: `dijit._Widget`
 - ❖ Base class for all widgets
 - ❖ Provides many basic features for widgets (attributes, watch, connect, DOM handling, ...)
- ❖ Template: `dijit._Templated`
 - ❖ Mixin `dijit._Templated` to support building dijit UI using templates
 - ❖ Template: custom HTML code with variable/event hooks
 - ❖ Use `dojoAttachPoint` and `dojoAttachEvent` in HTML-tags to reference DOM nodes and add event listeners
- ❖ Container
 - ❖ Mixin `dijit._Container` to support managing children dijits
 - ❖ Container's `startup()` calls children's `startup()`
 - ❖ Removing a child dijit from the container doesn't destroy it



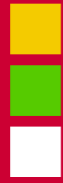
Widget attributes

- ❖ `dijit._Widget.set()` is the standard API to set and get attributes
 - ❖ If necessary, custom setter/getter functions can be provided
 - ❖ Setter functions : `_setXXXAttr()`
 - ❖ Getter functions : `_getXXXAttr()`
 - ❖ For the attribute email : `_setEmailAttr()` and `_getEmailAttr()`
- ❖ `dijit._Widget.watch()` notifies observers upon attribute changes
- ❖ Attributes are mapped to DOM nodes via `attributeMap`
 - ❖ Attribute modification updates the DOM node automatically



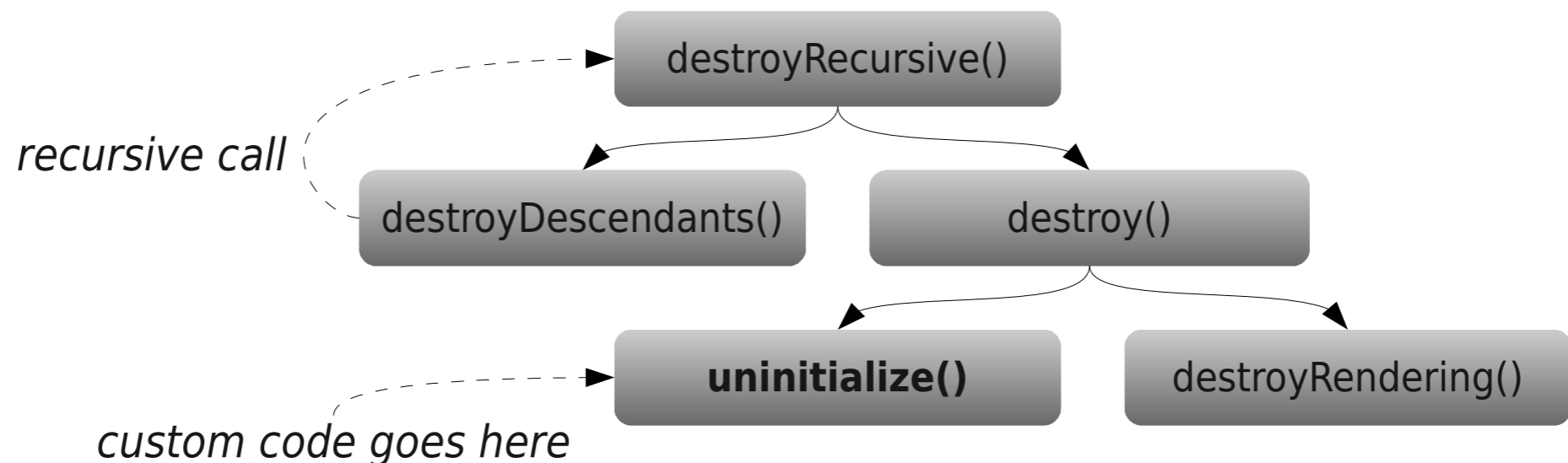
Dijit life cycle - creation

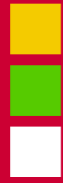
- ❖ **preamble()** - *originates from dojo.declare*
 - ❖ Advanced feature, return value is passed over to constructor
- ❖ **constructor()** - *originates from dojo.declare*
 - ❖ Initialize and add additional properties
- ❖ **postMixInProperties()** - *originates from dijit._Widget*
 - ❖ All member methods/variables have been mixed in from all ancestors
- ❖ **buildRendering()** - *originates from dijit._Widget*
 - ❖ DOM nodes are set up
- ❖ **postCreate()** - *originates from dijit._Widget*
 - ❖ Executed when widget is created and visibly placed in the DOM
 - ❖ Children cannot be safely accessed here
- ❖ **startup()** - *originates from dijit._Widget*
 - ❖ Fires when widget and all children have been created
 - ❖ For programmatic approach: call startup() manually after adding all children



Dijit life cycle - destruction

- ❖ **destroy()** - *originates from dijit._Widget*
 - ❖ Destroys widget itself
- ❖ **destroyRecursive()** - *originates from dijit._Widget*
 - ❖ Destroys children and widget itself
- ❖ **uninitialize()** - *originates from dijit._Widget*
 - ❖ Destructor method for custom clean-up
- ❖ See also: http://dojotoolkit.org/reference-guide/dijit/_Widget.html





Conclusion

- ❖ There are many JavaScript libraries out there
- ❖ Yet only a few provide a good feature set for Rich Internet Application (RIA) development
- ❖ The Dojo Toolkit
 - ❖ OpenSource license & open development
 - ❖ Many features & good quality code
 - ❖ Many widgets that are themeable
 - ❖ Great for RIA development & pure DOM manipulation



The end...

- Thank you very much for your attention!
- Much more could be said...
- Do you have questions?

Resources

- <http://dojotoolkit.org/>
- <http://dojotoolkit.org/api/>
- <http://dojotoolkit.org/reference-guide/>
- <http://dojotoolkit.org/documentation/>
- <http://www.sitepen.com/blog/>

