

Introduction to Dojo Toolkit



Topics

- What is and Why Dojo Toolkit?
- Dojo Architecture
- Loading Dojo toolkit
- Remoting (Ajax operation)
- Dojo Helloworld example

Topics Covered in Advanced Dojo Presentation

- Creation of Dojo Widgets
- Dojo Drag and Drop
- Dojo Animation
- Dojo Storage
- Performance tuning

What is and Why Dojo Toolkit?

What is Dojo Toolkit?

- Open Source DHTML toolkit written in JavaScript
 - > It is a set of JavaScript libraries
- Aims to solve some long-standing historical problems with DHTML
 - > Browser incompatibility
- Allows you to easily build dynamic capabilities into web pages
 - > Widgets
 - > Animations
- Server technology agnostic

Why Dojo Toolkit?

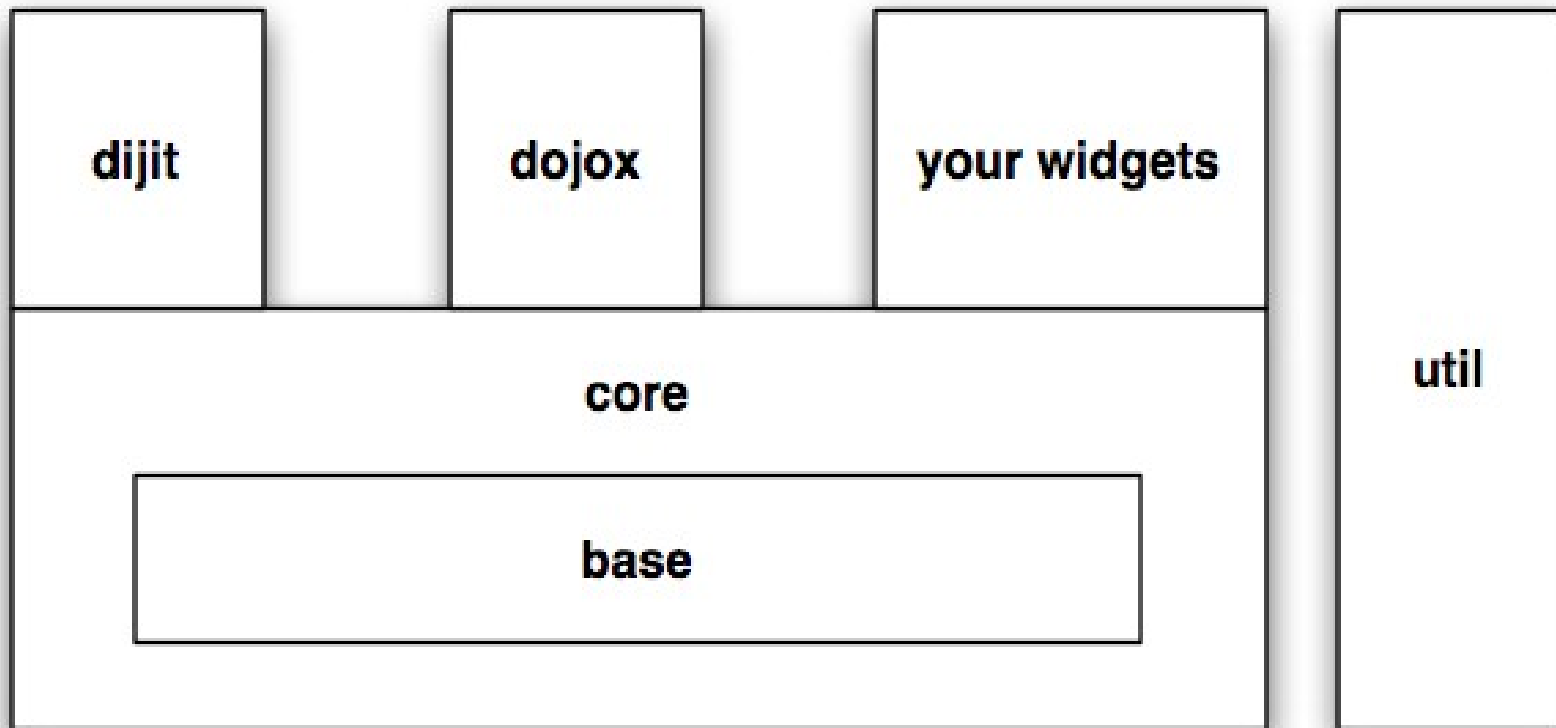
- You can use Dojo to make your web applications more useable, responsive, and functional
 - > Supports AJAX
- Dojo provides lots of plumbing facilities
 - > Hides XMLHttpRequest processing
 - > Handles browser incompatibilities
- Strong developer community

Features of Dojo Toolkit

- Powerful AJAX-based I/O abstraction (remoting)
- Graceful degradation
- Backward, forward, bookmarking support
- Aspect-oriented event system
- Markup-based UI construction through widgets
- Widget prototyping
- Animation
- Lots of useful libraries

Dojo Architecture

Dojo Toolkit Libraries



Dojo Toolkit: 3 main parts:

Dojo

- > Browser normalization, package loading, DOM access and manipulation, Firebug Lite debugging, events, data access, Drag and drop, Asynchronous remote calls, JSON encoding/decoding

dijit

- > Interface **widgets**, Advanced UI controls, Template driven

dojoX

- > **Inventive** innovative: graphics, offline, widgets like grid spreadsheet,

Loading Dojo Toolkit

Two Options

- Load it from the network
- Load it from locally installed version

(Option 1) Load it from network

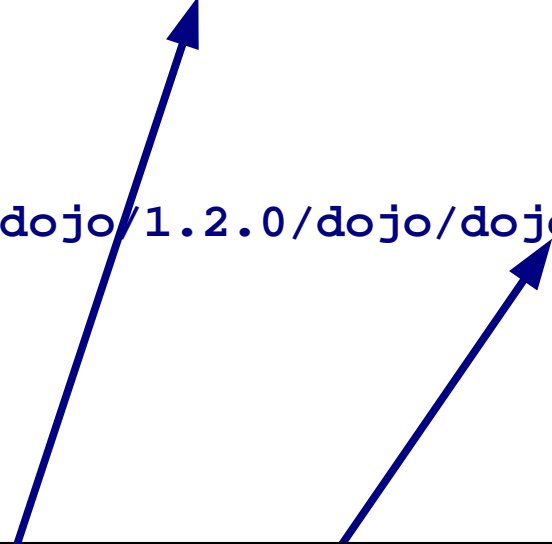
- Download nothing and simply pull Dojo from the network

AOL:

```
<SCRIPT TYPE="text/javascript"  
SRC="http://o.aolcdn.com/dojo/1.2.0/dojo/dojo.xd.js">  
</SCRIPT>
```

Google:

```
<SCRIPT TYPE="text/javascript" SRC="  
http://ajax.googleapis.com/ajax/libs/dojo/1.2.0/dojo/dojo.xd.js  
">  
</SCRIPT>
```



script element is responsible for loading the base Dojo script

(Option 2) Load it from locally installed version

- Download from <http://dojotoolkit.org/downloads>

```
<script type="text/javascript"  
    djConfig="parseOnLoad: true"  
    src="js/dojo/dojo.js" >  
</script>
```


script element is responsible for loading the base Dojo script

The package system handles the loading of all other dependencies and modules once dojo.js has been loaded into the page

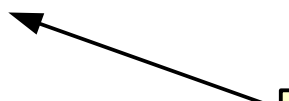
Setting up dojo javascript in the page

```
<head>
  <style type="text/css">
    @import "js/dojo/resources/dojo.css";
  </style>
  <script type="text/javascript"
    src="js/dojo/dojo.js"
    djConfig="parseOnLoad: true"
    isDebug: true >
  </script>
</head>
```

Load the dojo style sheet

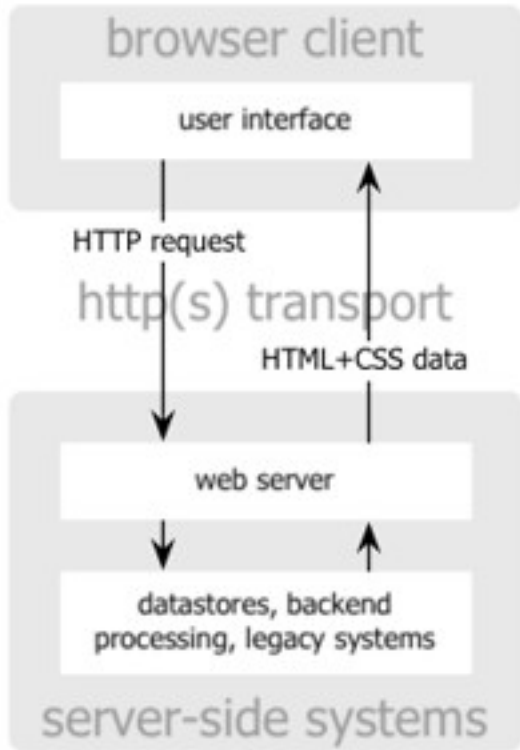


Use firebug lite for debugging



Remoting (Ajax operations)

Traditional Web



classic web application model

AJAX



within a browser, there is AJAX engine

Ajax web application model

Performing XMLHttpRequest (XHR)

- `dojo.xhrGet()`
- `dojo.xhrPost()`
- `dojo.xhrDelete()`
- `dojo.xhrPut()`

dojo.xhrGet()

- The dojo.io.* namespace contains generic APIs for doing network I/O
 - > `dojo.xhrXXX()` hides low-level `XMLHttpRequest` operations
- Also handles
 - > back-button interception
 - > transparent form submission
 - > advanced error handling

Getting Data from the Server

```
<script type="text/javascript">
```

```
  dojo.xhrGet({  
    url: 'sayHello',  
    load: helloCallback,  
    error: helloError,  
    content: {name: dojo.byId('name').value }  
  });
```

call url

Callback function

On error function

Content to send

```
</script>
```

Example: Dojo HelloWorld Example

Connecting an Event to the Widget

```
<head>
```

```
...
```

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

```
...
```

```
</head>
```

```
<body class="tundra">
```

```
Name: <input name="Name" id="name" type="text" />
```

```
<button dojoType="dijit.form.Button" id="helloButton">
```

```
    Hello World!
```

```
    <script type="dojo/method" event="onClick">  
        makeAjaxCall();
```

```
    </script>
```

```
</button>
```

```
</body>
```

attach an event to button through
a script type of dojo/method

Getting Data from the Server

```

<head>
  <script type="text/javascript">
    function makeAjaxCall(){
      dojo.xhrGet({
        url: 'sayHello.jsp',
        load: helloCallback,
        error: helloError,
        content: {name: dojo.byId('name').value }
      });
    }
    function helloCallback(data,ioArgs) {
      dojo.byId("returnMsg").innerHTML = data;
    }
  </script>
</head>
<body>
  Name: <input name="Name" id="name" type="text" />
  <button dojoType="dijit.form.Button"
    <script type="dojo/method" event="onClick">
      makeAjaxCall();
    </script>
  ...
  <p id=returnMsg></p>

```

The sayHello JSP

```
<%  
    String returnString = request.getParameter("name");  
  
    if (returnString == null || returnString.isEmpty()) {  
        // Return error message  
        returnString = "Name is required.";  
        out.print("Error: " + returnString);  
    } else {  
        // Return the name  
        out.print("Hello: " + returnString);  
    }  
>%
```

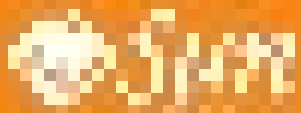
dojo.xhrPost a Form

```

<head>
  <script type="text/javascript">
    function makeAjaxCall(){
      dojo.xhrPost({
        url: 'sayHello',
        load: helloCallback,
        error: helloError,
        form: 'myForm'
      });
    }
    function helloCallback(data,ioArgs) {
      dojo.byId("returnMsg").innerHTML = data;
    }
  </script>
</head>
<body>
  <form id="myForm" method="POST">
    Name: <input type="text" name="name">
  </form>
  <button dojoType="dijit.form.Button"
    <script type="dojo/method" event="onClick">
      makeAjaxCall();
    </script>
  </button>

  <p id=returnMsg></p>
</body>

```



Introduction to Dojo Toolkit

