



QooXDOO

THE NEW ERA OF WEB DEVELOPMENT

qooxdoo

Andreas Ecker, Derrell Lipman

The Ajax Experience, 25-27 July 2007

- Client-side JavaScript framework
- Professional application development
- Comprehensive GUI toolkit
- Advanced client-server communication
- Open Source (LGPL / EPL)
- Initiated by 1&1 Internet AG in 2005

<http://qooxdoo.org>

## ☺ No browser extensions

(ActiveX, Java, Flash, Silverlight, ...)

## ☺ Common web browser

- Internet Explorer      7                      6
- Firefox                      2.0                      1.5
- Opera                      9
- Safari (WebKit)              3.0

# Requirements (Developers)

QooXDoo

- ☺ No HTML
- ☺ No CSS
- ☺ No DOM
- ☺ No specific server-side software
- ☺ Develop on any platform
- ☺ Object-oriented programming
- ☺ JavaScript

- Many standard widgets
- Simple creation of custom widgets
- Powerful layout managers
- Full keyboard support
- Drag & drop
- Event-based programming
- Real-world theming

## Events:

- Property value changes
- Abstraction layer directly above DOM
  - “execute”, “beforeDisappear”, etc.
- Unified key event handling
  - “keydown”, “keypress”, “keyinput”, “keyup”
- Unified mouse event handling
  - mouse clicks, mouse wheel, drag & drop
- Custom events

- Decoupled from widget code
- Simple yet powerful configuration files
- No CSS means no CSS hacks
- Easily done by designers, not only developers
- Switching at run-time

- **Skeleton**  
as a perfect base for custom web applications
- **Concise project structure**  
for application classes as well as static resources
- **Pre-configured**  
as an out-of-the-box solution for taylor-made production use

## Application development (`$ make source`)

- Checks, Debugging

## Application deployment (`$ make build`)

- Compressor (for minimum size)
- Optimization (for maximum performance)
- Dependencies (with automatic resolving)
- Modules, Packages (for optimal distribution)
- Target directories (for optimal integration)

# Working Example

QooXDOO

```
qx.Class.define("my.Application",
{
  extend : qx.application.Gui,           // extend the Gui class

  members :
  {
    main : function()                   // override main()
    {
      this.base(arguments);           // call superclass' main()

      var button =
        new qx.ui.form.Button("First Button", // create a button
                               "./resource/image/test.png");

      button.set( { top: 50, left: 50 } ); // position it
      button.addToDocument();           // add it to the document

      button.addEventListener("execute", // action upon click/Enter
                              function(e) { alert("Hello World!"); } );
    }
  }
});
```

- Fully class-based
- Namespaces
- Only noncritical manipulation of native objects
- Static classes, abstract classes, singletons
- Interfaces, mixins
- Public, protected, private members
- Dynamic properties

# Class Declaration



```
qx.Class.define("qx.ui.form.Button", {  
    extend      : qx.ui.basic.Atom,  
    implement  : [qx.IClickable, ...],  
    include    : [qx.MToggle, ...],  
    construct  : function() { ... },  
    properties : { ... },  
    statics   : { ... },  
    members   : { ... },  
    settings  : { ... },  
    variants  : { ... },  
    events    : { ... },  
    destruct  : { ... },  
    defer     : { ... }  
});
```

- **Generic transport layer**
  - Queues, Timeouts
  - synchronous, async, cross-domain
  - **Data exchange**
    - via JSON, XML, JavaScript, ...
- **Not just wrapper around XMLHttpRequest**
  - Transport implementation depends on needs

- **XMLHttpRequest**

- sync
- async

- **Mime Types**

- Text
- JavaScript
- JSON
- XML
- HTML

- **Iframe**

- async
- file upload
- form fields

- **Mime Types**

- Text
- JavaScript
- JSON
- XML
- HTML

- **Script**

- async
- cross domain

- **Mime Types**

- JavaScript
- JSON

- **Use your Existing Backend**
- **Remote Procedure Calls (RPC)**
  - **Easy and secure data exchange**
  - **RPC servers (backends):**
    - Java, PHP, Perl included with qooxdoo
    - Embedded Javascript (ejs) backend is used in Samba4
    - Documentation to write a backend is provided

- Automatic generation (`$ make api`)
- Full-featured JavaScript parser
- Javadoc-like comments
- Also for documenting a *custom* application!
- API viewer application

<http://api.qooxdoo.org>

# Sample API Documentation



```
/**
 * Set whether the open/close button should be displayed
 * on a branch, even if the branch has no children.
 *
 * @type member
 *
 * @param b {Boolean}
 *   <i>>true</i> if the open/close button should be shown;
 *   <i>>false</i> otherwise.
 *
 * @return {void}
 */
setAlwaysShowOpenCloseSymbol : function(b)
{
    ...
}
```

- **Browser independence**
- **High performance**
- **No memory-leaks**
- **OO programming**
- **Back button, bookmarking**
- **Internationalization**
- **Migration support**

...

- **Unified code formatter**
- **Unit testing**
- **Logging**
- **Debugging**
- **Widget inspector**
- **Profiler**
- **IDE support**

- Eclipse Rich Ajax Platform (“RAP”)
- Borland/CodeGear “Delphi for PHP”
- XML-based description of interface
- ASP.Net
- Pure Java-based qooxdoo application (planned)
- GUI designer (planned)

<http://qooxdoo.org>

