qooxdoo
The New Era of Web Development
Brian Brooks, Mike Pack, Drew Ford
What is qooxdoo?

- [http://cse1.cs.colorado.edu/~packt/](http://cse1.cs.colorado.edu/~packt/)
- JavaScript
- Ajax framework
- GUI Toolkit
- Client-server communication
- Object Oriented framework
- Platform-independent
- Cross-browser
- Open source (LGPL/EPL)
- No HTML, CSS, or DOM knowledge needed
qooxdoo OO

- JavaScript is not object oriented!
  - Prototype-based language
- qooxdoo OO:
  - Classes (Inheritance, Constr/Destr, access modifiers, static/abstract classes, singleton)
  - Interfaces
  - Mixins
qooxdoo OO

- **Class definitions:**
  - `qx.Class.define("my.spiffy.class");`

- **Class instantiation:**
  - `var myClass = new my.spiffy.class;`

- **Inheritance:**
  ```javascript
  qx.Class.define("my.spiffier.class", {
  extend: my.spiffy.class
  });
  ```
Properties & Constructor / Destructor:

qx.Class.define("my.spiffy.class" {  
members: {  
  __myPrivateVar,
  __myProtectedVar,
  publicVarName: "Ken Anderson",
  getProtected: function () { return \n      this.__myProtectedVar; }
  },
  construct: function () { ... },
  destruct: function () { ... }
});
qooxdoo OO

- Static / Abstract / Singleton Classes:

```javascript
qx.Class.define("qx.test.blah", {
  type : "abstract"
  ...
});
```

- Multiple Inheritance is not supported.
- Interfaces:

```javascript
qx.Interface.define("my.Iface", {
  extend: [SuperInterfaces],
  members: { ... },
  events: { ... }
});
```
Mixins:

```javascript
qx.Mixin.define("myMixin", {
  include: [SuperMixins],

  members : { ... }
});
```
qooxdoo: How does it work??

- Domain-specific language (DSL)
  - qooxdoo OO + Web 2.0 + multi-browser
  - Python scripts: lexer, parser, preprocessor, optimizer, compiler, etc...
  - Compiles to native JavaScript
  - Bootstrapped via generated index.html
- EBNF-like grammar
  - Primitive types, JS constructs, Classes, etc...
BOM (Browser Object Model)

- Models an ideal browser
- Adds flexibility to browser compatibility
- Abstract base level browser functionality
  - Document, Window, History JavaScript objects
  - Client functions
  - HTML DOM information
    - Background, Class, Location, etc.
  - Low level AJAX
- Mostly used in high level classes
BOM

- History
  - Add to browser history
  - Simulate Back/Forward button
- XMLHttpRequest
  - Crude low-level implementation
- Client Information
  - Engine, Locale, Platform
- Native Events
  - Attach handlers to DOM nodes
IO (Input/Output)

- Manage remote communication
  - AJAX
  - IFrame
  - Dynamic Scripts
    - JSON (JavaScript Object Notation)
    - RPC (Remote Procedure Calls)
- Load Images
- Load Scripts
IO

- Two layers of communication abstraction
  - qx.io2.HttpRequest
    - Higher level than BOM XmlHttpRequest
    - Represents HTTP request as qooxdoo class
    - Fires events upon status change
  - qx.io.remote.Request
    - Higher level than qx.io2.HttpRequest
    - Exchange between communication methods
      - XmlHttpRequest
      - IFrame
      - Script
AJAX Demo

- AJAX Demo
GUI

- Input Fields
- Scroll Bars
- Text
- Popups
- Trees
- Tab View
- Tables

- Widgets / Controls
- Menu
- RSS Feeds
  - Feed Reader allows for aggregation of blogs, podcasts, etc...
Widgets & Custom Controls

- Provide interaction between user and application
- Allow for User Input, Validation, etc...
  - File Tree Demo
- Menu Controls
Forms

- Create custom forms using
  - Widgets ([File Tree Demo](#))
  - GUI features
  - Events
    - Handle events such as clicked buttons and keystrokes using listeners
- [Login Demo](#) and [Window Demo](#)