Advanced JavaScript

Juhan Aasaru
08 March 2010

About me - Juhan Aasaru

- Java Software Architect and Developer in Webmedia
  - Started in Webmedia summer 2005
  - Participated in development of around 10 web-based information systems
- 2005 - 2008 Tartu University - M.S. Informatics
- 2001 - 2005 Tallinn University of Technology Network Software

Presentation overview

- Why we need heavy JavaScript
- Set of frameworks and libraries
    - jQuery
        - and Prototype
    - Google Web Toolkit
        - and SmartGWT
    - Direct Web Remoting (DWR)
- Development tools
    - Firebug

Why we need heavy JavaScript?

- For development of enhanced user interfaces and dynamic websites
- To make application feel more responsive because JavaScript can respond to user actions quickly
- User does not need to install any new software or browser plugins
- No pure JavaScript in this presentation but libraries and tools inststead

jQuery v1.4

jQuery

- Powerful JavaScript library
- Simplify common JavaScript tasks
- Access parts of a page
  - using CSS or XPath-like expressions
- Modify the appearance of a page
- Alter the content of a page
- Change the user’s interaction with a page
- Add animation to a page
- Provide AJAX support
- Abstract away browser quirks
- Methods can be chained (methods return "this")

jQuery - apply css dynamically

```
<html>
  <body>
    <h2 id="myTitle">Presentations</h2>
    <ul>
      <li>JavaScript</li>
      <li>Advanced JavaScript</li>
      <li>AJAX</li>
   </ul>
  </body>
</html>
```

```
.header {
  font-size: 15px;
}
.bigText {
  font-style: bold;
  color:blue;
  font-size: 1.5em;
}
```

jQuery - using selectors

- Selecting part of document is fundamental operation
- A JQuery object is a wrapper for a selected group of DOM nodes
- $() function is a factory method that creates JQuery objects
- $(“li”) is a JQuery object containing all the “li” elements in the document
- $(".bigText") is a jQuery object containing all elements that have class="bigText"
- $("#myTitle") = document.getElementById("myTitle")
- $("h2#myTitle") - checks also parent
- $("div:hidden") - select all hidden elements
- $(":parent") - select all elements that have children
- ... many more ...

jQuery - manipulation

- .addClass() method changes the DOM nodes by adding a ‘class’ attribute
    - The ‘class’ attribute is a special CSS construct that provides a visual architecture independent of the element structures
- $(“li”).addClass(“bigText”) will change all occurrences of <li> to <li class=“bigText”>
- $('.bigText').append('<i>Look at here!</i>');
- $('#myTitle').hasClass('bigText') - test
- $('.bigText').remove(); - remove from DOM

jQuery - events

- To make this change, put it in a function and call it when the document has been loaded and the DOM is created

```
function makeBig(){$("p").addClass("bigText")}
<body onLoad="makeBig()">
```

- We had to alter the HTML (bad)
- Structure and appearance should be separated!
- Also, onLoad waits until all images *etc* are loaded. Tedious.

jQuery - events

- JQuery provides an independent scheduling point after DOM is created and before images are loaded
  - $(document).ready(makeBig);
- No HTML mods required. All done in script.
- Better solution:

  -
    ```
    $(document).ready(function(){
            $(“p”).addClass(“bigText”)
    });
    ```

```
<html><head>
<script src="jquery.js" type="text/javascript"></script>
<script src="test.js" type="text/javascript"></script>
…
```

jQuery - more events

- bind(eventname, function) method
  - eventname = blur, focus, focusin, focusout, load, resize, scroll, unload, click, dblclick, mousedown, mouseup, mousemove, mouseover, mouseout, mouseenter, mouseleave, change, select, submit, keydown, keypress, keyup, error
- $(“a[@href]”).bind(‘click’,
      function(){ $(this).addClass(‘bigText’);   }
  );
- $("a[@href]").click(fnc) - shortcut

jQuery AJAX request

- Example request

```
$.ajax({
     type: 'GET'
     url: 'ajax/test.html',
     data: 'name=John&location=Boston',
     success: function(data) {
         $('.result').html(data);
         alert('Load was performed.');
     }
});
```

jQuery plugins

- jQuery offers a mechanism for adding in methods and functionality, bundled as plugins
- Most of the methods and functions included in the default download are written using the jQuery plugin construct.
- Plugin filename pattern: **jquery.[insert name of plugin].js**, eg. jquery.debug.js

Simple jQuery plugin example

- define

```
jQuery.fn.debug = function() {
    return this.each(function(){
      alert(this);
    });
};
```

- call

  $("div p").debug();

jQuery alternative

- Prototype
  - built into Ruby on Rails
  - many Rails-like constructs
  - a bit better API design
  - last version from September 2009
  - No method chaining as in jQuery:

```
$('a:contains("sign")').
parent().
addClass("bigText")
```

Google Web Toolkit

ver 2.0

Google Web Toolkit (GWT)

- Free open source framework for creating browser-based rich AJAX applications with Java
- Code written in Java is compiled to JavaScript
- Pure JavaScript / DHTML at the client side
- Supports all major browsers
- Pure Java at the server side
- Development tools
    - Eclipse plugin
    - Plugins for browsers
- Used by many products at Google, including Google Wave

GWT Architecture

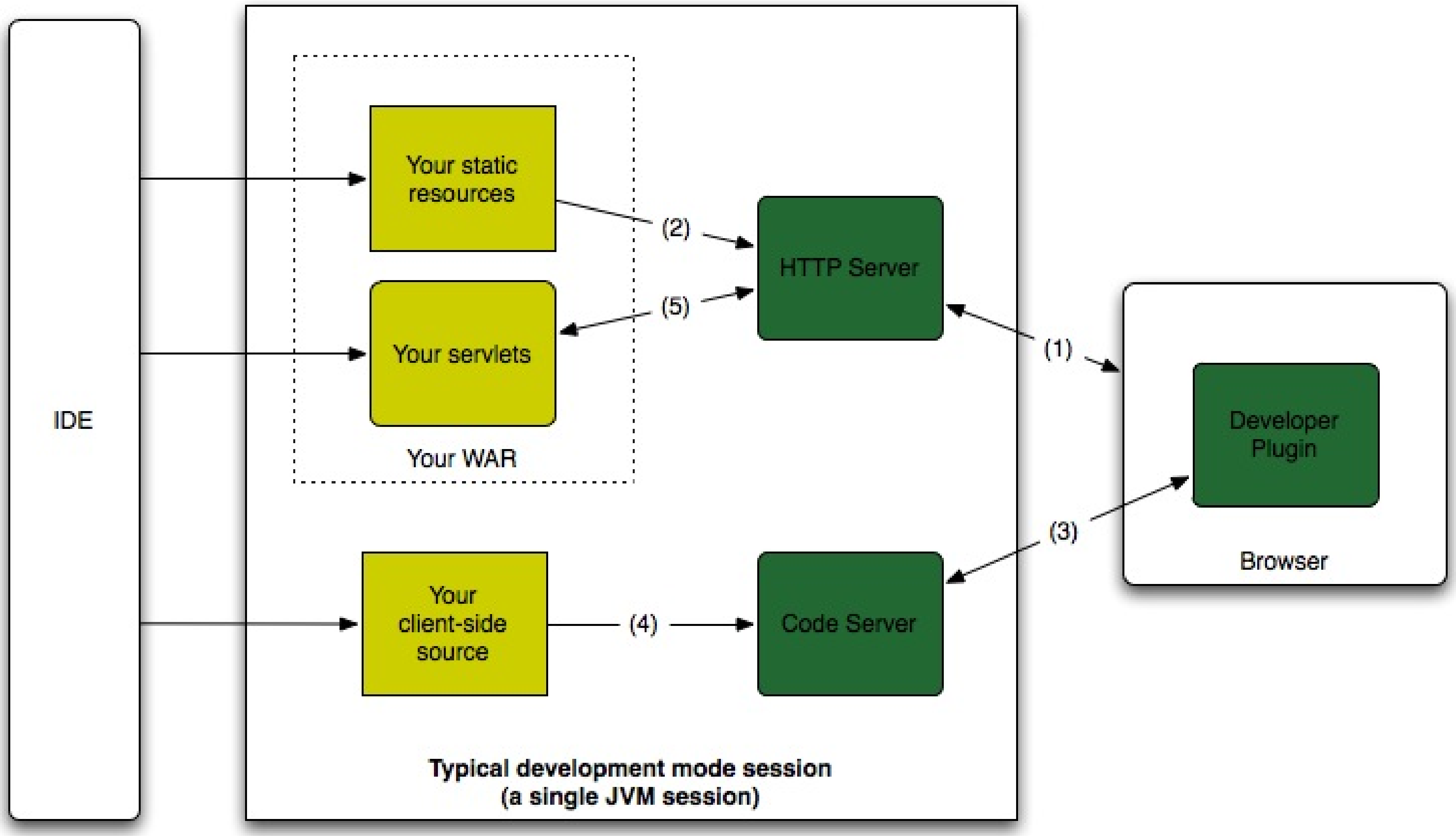
- Model-View-Presenter paradigm
  - Model - business objects
  - View - UI components (can be separate for mobile)
  - Presenter - business logic and events
- Client-Server communication
  - Always asychronous
  - Remote Procedure Calls (GWT RPC) - mechanism for passing Java objects to and from a server over standard HTTP
  - JSON or XML over HTTP for data-retreival

GWT Features

- API for creating GUI applications, similar to Swing and SWT
- API for manipulating the Web browser's DOM (Document Object Model)
- Java-to-JavaScript compiler
    - Developer uses pure Java APIs
    - JavaScript skills not required
    - No need to handle browser incompatibilities
- Widgets can be reuse
- UI layer is separate from business logic
- Back button works and URL-s are bookmarkable
    - Programmatically

GWT Development Mode

- Java bytecode runs in the JVM
  - Allows debugging
  - Browser plugin needed



SmartGWT
based on SmartClient library

ver 2.1

SmartGWT

- Smart GWT is a GWT based framework
- LGPL license (server-side functions propertiary)
- Comprehensive widget library for application UI
- Ties widgets to server-side data management

SmartGWT demo

- Showcase

Firebug
Web Development Evolved.

Firebug v1.5

Firebug features

- Inspect HTML and modify style and layout in real-time
    - get xpath of each element
- Advanced JavaScript debugge
- Analyze network usage and performance
- Extendendable
-

Firebug demo...

- Inspect element, edit it
- Copy element xPath address
- Element information sub panels
    - style - which css instructions apply
    - computed - browser computed values
    - layout - detailed positioning info of element
    - DOM - JavaScript properties of element
- Detach Firebug window from browser
- Console - debugger
    - pause on error
    - insert JavaScript commands
    - profile JavaScript

Firebug demo

- script panel
  - set breakpoint
  - set conditional breakpoint
  - watch variable values
  - step into, step over
  - pause button pauses on next fireing function call
- net panel
  - shows all requests and time taken
  - see detailed request and response headers
  - can disable browser cache
  - pause button breaks on next xml/html request to see where it came from

Thank you! Questions?

Slides and references used

- *JQuery http://www.edshare.soton.ac.uk/1178/*
- *Prototype http://blog.thinkrelevance.com/2009/1/12/why-i-still-prefer-prototype-to-jquery*
- *Firebug demo http://getfirebug.com*