The Prototype Framework Part I: Ajax Support

(Prototype 1.6 Version)

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom mix of topics
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

- **Overview of Prototype**
- **Installation**
- **Ajax.Request**
  - Basics
  - Options
- **HTML lookup and insertion**
- **Ajax.Updater**
- **Ajax.PeriodicalUpdater**
- **Handling JSON Data**

Introduction

Overview of Prototype

- **Ajax utilities**
  - Ajax.Request, Ajax.Updater, Ajax.PeriodicalUpdater
  - Wraps response in Ajax.Response
    - Several new properties, but especially responseJSON
- **General DOM utilities**
  - $() to find element
  - $F() to get form value
  - *element*.update() to put into innerHTML
  - Many helpers in Element class
- **General utilites**
  - Extensions for Class, Function, Array, String

Ajax Utilities

- **Ajax.Request**
  - Takes URL and options object that designates "onSuccess" and "parameters".
- **Ajax.Updater**
  - Takes id of result region and URL.
  - Invokes URL once and puts responseText in result region
- **Ajax.PeriodicalUpdater**
  - Takes id of result region, URL, options object with "frequency" property. Can call "stop" on updater later.
- **Ajax.Response**
  - Passed to response handler functions
  - properties: responseText, responseXML, responseJSON

Downloading and Installation

- **Download**
  - http://www.prototypejs.org/download
    - Download a *single* .js file (e.g., prototype-1.6.02.js)
      - Usually renamed to prototype.js
    - This tutorial corresponds to Prototype 1.6
- **Online API**
  - http://www.prototypejs.org/api
- **Tips and Tutorials**
  - http://www.prototypejs.org/learn
- **Browser Compatibility**
  - Firefox: 1.5 or later
  - Internet Explorer: 6.0 or later (does not work in IE 5!)
  - Safari: 2.0 or later
  - Opera: 9.25 or later

Ajax.Request

Ajax.Request

- **new AjaxRequest(relative-url, options)**
  - Calls relative-url, wraps response in Ajax.Reponse, passes response to function specified in options
- **Options**
  - An anonymous object
  - Most important property: onSuccess
- **Basic example**
  - ```
    new Ajax.Request(
      "some-file.jsp",
      {onSuccess: someHandlerFunction});
    ```
    - someHandlerFunction should take one argument of type Ajax.Reponse.
    - It is automatically wrapped in anonymous function with local copy of request object, so it is threadsafe.

Ajax.Request Example Code: JavaScript

```
function showTime1() {
  new Ajax.Request(
    "show-time.jsp",
    { onSuccess: showAlert });
}

function showAlert(response) {
  alert(response.responseText);
}
```

This is a Prototype Ajax.Response object, not the raw XmlHttpRequest.
```

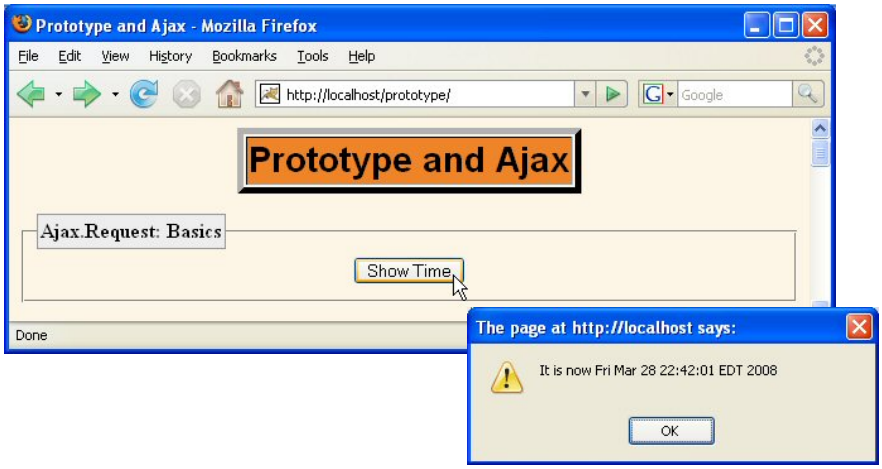
Ajax.Request Example Code: HTML

```
...
<head><title>Prototype and Ajax</title>...
<script src="./scripts/prototype.js"
        type="text/javascript"></script>
<script src="./scripts/prototype-examples.js"
        type="text/javascript"></script>
</head>
<body>...
<fieldset>
  <legend>Ajax.Request: Basics</legend>
  <form action="#">
    <input type="button" value="Show Time"
           onclick='showTime1()'/>
  </form>
</fieldset>
```


Ajax.Request Example Code: JSP

```
It is now <%= new java.util.Date() %>
```

Ajax.Request: Results



Passing Parameters

Ajax.Request Options { property1: v1, property2: v2...}

- **The second arg is an anonymous object with these as the most important properties**
  - onSuccess (default: none)
    - Response handler function (takes Ajax.Response as arg)
    - There are also many similar related options: onComplete, onFailure, onException, onXYX (for HTTP status codes)
  - parameters (default: empty string)
    - Can be explicit parameter string: "p1=val1&p2=val2"
    - Can also be parameter object: {p1: val1, p2: val2}
      - Values will be escaped automatically
  - asynchronous (default: true)
  - method (default: post)
  - evalJS (default: true)
    - Response text passed to "eval" if response type is application/javascript or a similar type
  - evalJSON (default: true)
    - Response text passed to eval (with parens added) and sent to responseJSON if response type is application/json

Ajax.Request Parameters Example Code: JavaScript

```
function showParams1() {
  new Ajax.Request(
    "show-params.jsp",
    { onSuccess: showAlert,
      parameters: "param1=foo&param2=bar" });
}

function showAlert(response) {
  alert(response.responseText);
}
```

Ajax.Request Parameters Example Code: HTML and JSP

```
<fieldset>
  <legend>Ajax.Request:
          The 'parameters' Option</legend>
  <form action="#">
    <input type="button" value="Show Params"
           onclick='showParams1()'/>
  </form>
</fieldset>
```


Ajax.Request Parameters Example Code: HTML and JSP

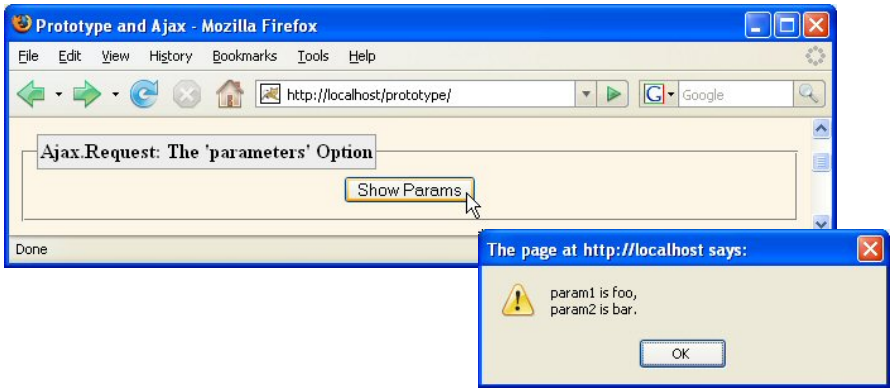
- **HTML**

```
<fieldset>
  <legend>Ajax.Request:
          The 'parameters' Option</legend>
  <form action="#">
    <input type="button" value="Show Params"
           onclick='showParams1()'/>
  </form>
</fieldset>
```

- **JSP (show-params.jsp)**

```
param1 is ${param.param1},
param2 is ${param.param2}.
```

Ajax.Request Parameters: Results



Utilities for Reading and Writing HTML Elements

- **$("id")**
  - Returns element with given id [getElementById("id")]
    - Can also take an Element instead of an element id
    - Can also take multiple arguments, in which case it returns an array of the Element results
    - Yes, "$" is really the function name
- **$F("id")**
  - Returns value of form element with given ID
    - Single value for most elements, array for multi-select lists
    - For textfields, equivalent to $("id").value
- **update("html-string")**
  - Inserts into innerHTML property
  - E.g., $("result-region").update("<h1>Test</h1>")

Building Parameter Strings

- **The $F function does not escape values**
  - So, this could yield illegal results
    - { onSuccess: someHandlerFunction, parameters: "param1=" + $F("field1") }
      - If field 1 contains spaces, ampersands, etc., this causes problems
      - You could do escape($F("field1")), but this gets a bit verbose
- **Instead of a parameter string, you can supply parameter object**
  - { param1: "val1", param2: "val2" ... }
  - Values (usually from $F(...)) are automatically escaped, then whole thing converted to parameter string
  - You can also do this explicitly anywhere with $H function that creates Hash, and toQueryString method
    - $H({p1: "val1", p2: "val2").toQueryString() returns "p1=val1&p2=val2"

Ajax.Request: Reading/Writing Utils Example Code: JavaScript

```
function showParams2() {
  var params =
    { param1: $F("param1"),
      param2: $F("param2") };
  new Ajax.Request(
    "show-params.jsp",
    { onSuccess: updateResult,
      parameters: params });
}

function updateResult(response) {
  $("result1").update(response.responseText);
}
```

Original (not escaped) value of textfield whose id (not name) is "param1".

Parameter object is converted to parameter string with escaped values.

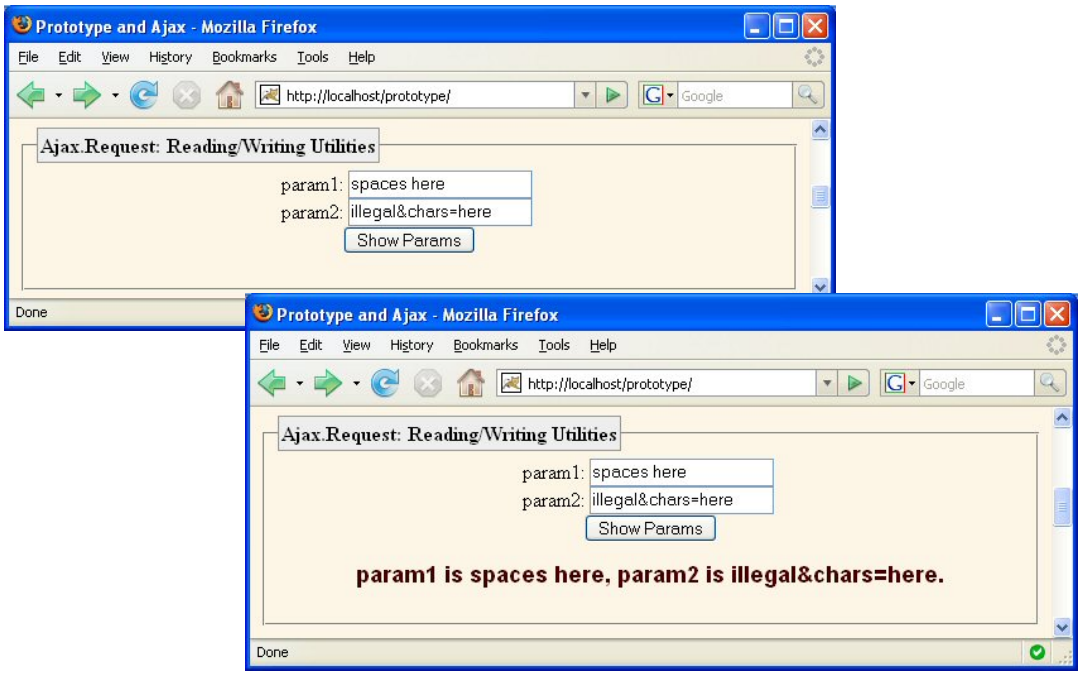
Finds element whose id is "result1".

Inserts into innerHTML property.

Ajax.Request: Reading/Writing Utils Example Code: HTML

```
<fieldset>
  <legend>Ajax.Request:
          Reading/Writing Utilities</legend>
  <form action="#">
    param1:
    <input type="text" id="param1"/>
    <br/>
    param2:
    <input type="text" id="param2"/>
    <br/>
    <input type="button" value="Show Params"
           onclick='showParams2()'/>
    <h2 id="result1"></h2>
  </form>
</fieldset>
```


Ajax.Request: Reading/Writing Utils: Results



Ajax.Updater

Ajax.Updater Example Code: JavaScript

```
function showParams3() {
  var params =
    { param1: $F("param3"),
      param2: $F("param4") };
  new Ajax.Updater(
    "result2",
    "show-params.jsp",
    { parameters: params });
}
```

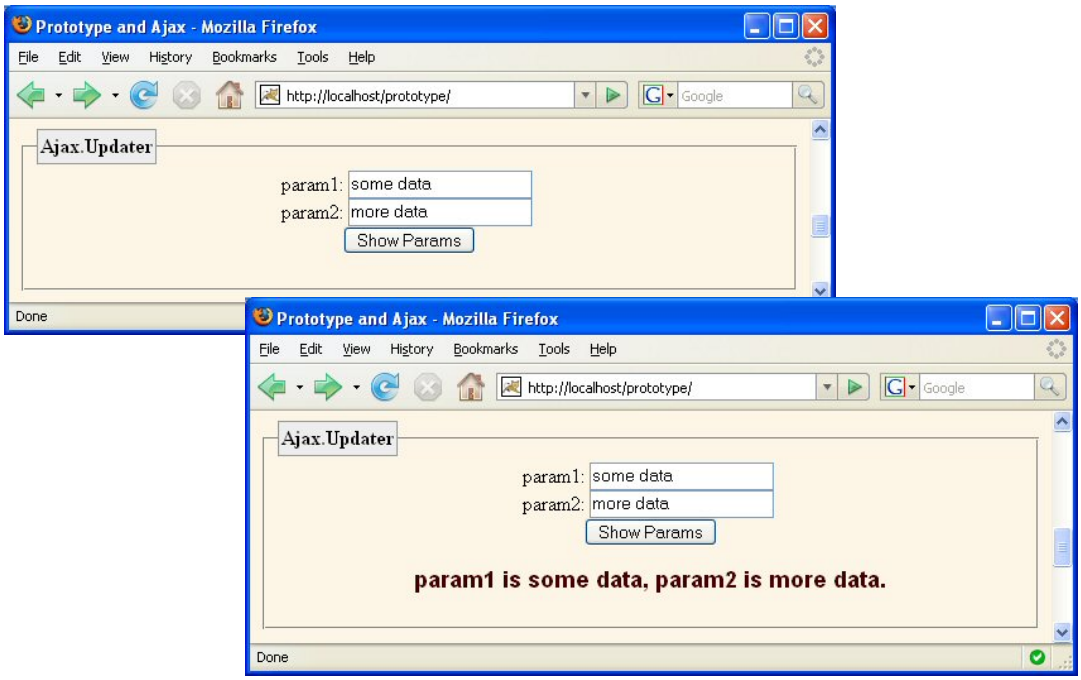
id of element into whose innerHTML the responseText is inserted

- **Notes**
  - No onSuccess needed
  - Can update a single element only. If you need to update more, use Ajax.Request with onSuccess
    - You could also return script from server, but then server needs to know name of DOM elements

Ajax.Updater Example Code: HTML

```
<fieldset>
  <legend>Ajax.Updater</legend>
  <form action="#">
    param1:
    <input type="text" id="param3"/>
    <br/>
    param2:
    <input type="text" id="param4"/>
    <br/>
    <input type="button" value="Show Params"
           onclick='showParams3()'/>
    <h2 id="result2"></h2>
  </form>
</fieldset>
```


Ajax.Updater: Results



Ajax.Updater Options

- **evalScripts**
  - Should <script> tags in the response be evaluated?
  - Default is false, so this option is very important if you return <script> tags that set event handlers (e.g. for scriptaculous in-place editors) for elements that are inserted
- **insertion**
  - Where text should be inserted relative to what is already there.
  - Default is to replace any existing content.
  - Options are 'top', 'bottom', 'before', 'after'
- **Standard options still supported**
  - parameters, onSuccess, etc.
- **Example**
  - var params = { param1: "foo", param2: "bar" };
  - new Ajax.Updater("some-id", "some-address",
    { evalScripts: true, insertion: 'top', parameters: params });

Ajax.PeriodicalUpdater

Ajax.PeriodicalUpdater Example Code: JavaScript

```
var updater; // Save ref so can "stop" later

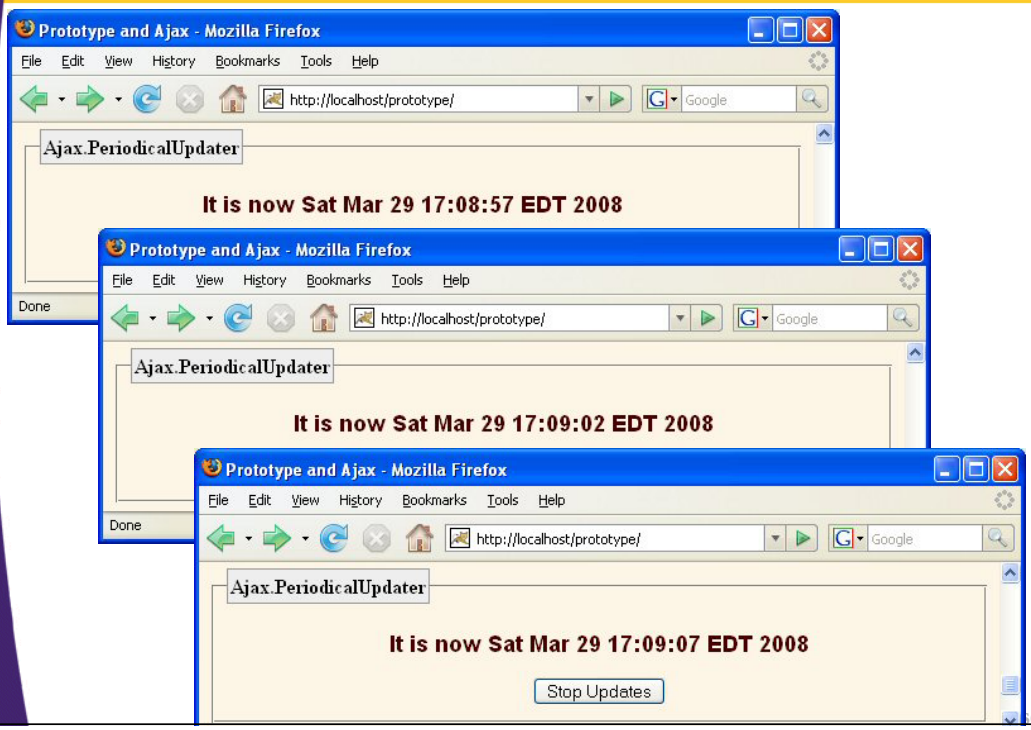
function showTime2() {
  updater = new Ajax.PeriodicalUpdater(
    "result3",
    "show-time.jsp",
    { frequency: 5 });
}
```


Ajax.PeriodicalUpdater Example Code: HTML

```
<fieldset>
  <legend>Ajax.PeriodicalUpdater</legend>
  <form action="#">
    <h2 id="result3"></h2>
    <script type="text/javascript">
      showTime2();
    </script>
    <input type="button" value="Stop Updates"
           onclick='updater.stop()'/>
  </form>
</fieldset>
```

- **Note**
  - Form needed only if you want to let user stop the action

Ajax.PeriodicalUpdater: Results



Handling JSON Data

Ajax.Response Properties

- **The arg passed to response handler has these as most important properties**
  - status, statusText
    - HTTP status code and corresponding text
  - responseText
    - Same as normal XmlHttpRequest.responseText
  - responseXML
    - Same as normal XmlHttpRequest.responseXML
  - responseJSON
    - Response text wrapped in parens and passed to "eval"
    - Only available if response type is application/json
  - headerJSON
    - Evaluated content of X-JSON response header
    - Alternative for responseJSON for small amounts of data

Ajax.Response Methods

- **Can also call these methods on response**
  - getHeader(responseHeaderName)
    - Gets header value.
    - Does not throw exception if header missing (unlike native XmlHttpResponse method)
  - getAllHeaders()
    - Returns a string with all headers, delimited by newlines
    - Does not throw exception if there are no headers
  - getResponseHeader, getAllResponseHeaders
    - Native version of above methods
    - Throws exceptions if headers missing
    - Used only with preexisting code that handled exeptions. Use getHeader and getAllHeaders otherwise

Using JSON Data with responseJSON property

- **Response object properties**
  - responseText, responseXML, and responseJSON
  - Response object passed to handler function (designated with onSuccess, etc.)
  - For more details see Ajax.Response in online API
- **Behavior**
  - Response text wrapped in parens and passed to "eval"
    - Server returns { prop1: val1, prop2: val2 } without parens
- **Requirements**
  - responseJSON populated only if response type is application/json

responseJSON Example Code: Core JavaScript

```
function showNums() {
  new Ajax.Request(
    "show-nums",
    { onSuccess: showNumberList,
      method: "get" });
}

function showNumberList(response) {
  var obj = response.responseJSON;
  var list = makeList(obj.fg, obj.bg,
                      obj.fontSize, obj.numbers);
  $("result4").update(list);
}
```

Strings

Integer

Array of doubles

responseJSON Example Code: Auxiliary JavaScript

```
function makeList(fg, bg, fontSize, nums) {
  return(
    listStartTags(fg, bg, fontSize) +
    listItems(nums) +
    listEndTags());
}

function listStartTags(fg, bg, fontSize) {
  return(
    "<div style='color:" + fg + "; " +
                "background-color:" + bg + "; " +
                "font-size:" + fontSize + "px'>\n" +
    "<ul>\n");
}
```


responseJSON Example Code: Auxiliary JavaScript (Continued)

```
function listItems(items) {
  var result = "";
  for(var i=0; i<items.length; i++) {
    result = result + "<li>" + items[i] + "</li>\n";
  }
  return(result);
}

function listEndTags() {
  return("</ul></div>");
}
```

responseJSON Example Code: HTML

```
<fieldset>
  <legend>Ajax.Request: responseJSON</legend>
  <form action="#">
    <input type="button" value="Show Nums"
           onclick='showNums()'/>
    <div id="result4"></div>
  </form>
</fieldset>
```


responseJSON Example Code: Servlet

```
public class ShowNumbers extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    String fg = ColorUtils.randomColor();
    request.setAttribute("fg", fg);
    String bg = ColorUtils.randomColor();
    request.setAttribute("bg", bg);
    String fontSize = "" + (10 + ColorUtils.randomInt(30));
    request.setAttribute("fontSize", fontSize);
    double[] nums =
      { Math.random(), Math.random(), Math.random() };
    request.setAttribute("nums", nums);
    response.setContentType("application/json");
    String outputPage = "/WEB-INF/results/show-nums.jsp";
    RequestDispatcher dispatcher =
      request.getRequestDispatcher(outputPage);
    dispatcher.include(request, response);
}}
```

responseJSON Example Code: Servlet

```
{ fg: "${fg}",
  bg: "${bg}",
  fontSize: ${fontSize},
  numbers: [ ${nums[0]}, ${nums[1]}, ${nums[2]}]
}
```

- **Notes**
  - No enclosing parens. Prototype will wrap in parens and then pass to "eval".
  - Types
    - fg and bg: Strings
    - fontSize: Integer
    - numbers: Array of doubles

responseJSON Example Code: Auxiliary Java Code

```
public class ColorUtils {
  private static String[] colors = {
    "aqua", "black", "blue", "fuchsia", "gray",
    "green", "lime", "maroon", "navy", "olive",
    "purple", "red", "silver", "teal", "white", "yellow"
  };

  /** A random number between 0 and range-1, inclusive. */

  public static int randomInt(int range) {
    return((int)(Math.random() * range));
  }

  /** One of the official HTML color names, at random. */

  public static String randomColor() {
    return(colors[randomInt(colors.length)]);
  }
}
```

Wrapup

Recommended Books

- ***Prototype and script.aculo.us: You Never Knew JavaScript Could Do This!***
  - By Christophe Porteneuve
- ***Prototype and Scriptaculous in Action***
  - By Dave Crane, Bear Bebeault, Tom Locke

Summary

- **Ajax.Request**
  - new Ajax.Request("url", { onSuccess: handler, ... });
    - Also has parameters option (string or object)
- **Ajax.Updater**
  - new Ajax.Updater("id", "url", {options});
- **Ajax.PeriodicalUpdater**
  - new Ajax.PeriodicalUpdater("id", "url", { frequency: ...});
- **Ajax.Response**
  - Has responseJSON property
- **Utility function**
  - $("some-id") → Element with that id
  - $F("some-id") → Value of Element with that id
  - someElement.update("html") – inserts in innerHTML

Questions?