prototype

a JavaScript framework

Presented by:
Joe Chan
Brian Hallesy
Jarret Lavallee

What is it?

- "Prototype is a JavaScript framework that aims to ease the development of dynamic web applications."
- Library of JavaScript functions used for developing JavaScript applications
- One JavaScript file (prototype.js)
- Cross-browser compatibility
- Requires no installation/configuration on the server side
- Lightweight framework - 126k

How did it come to be?

- Prototype was created by Sam Stephenson.
- He wanted to create a browser programming environment which implemented functionality seen in dynamic languages such as Ruby.
- Prototype 1.0 was released in March 2005 as part of the Ruby on Rails framework. Since then, it has become one of the more popular JavaScript frameworks.
- Most recent version: Prototype 1.6.0.3 released on September 29, 2008

Services provided by Prototype

- Support for Ajax.
- Provides numerous utility methods, which can be used by developers to address common scripting needs.
- Provides DOM extensions.
- Includes Object-oriented functionality, such as defining classes and implementing inheritance.

Ajax

- The general idea of using Ajax calls is to communicate between a web server and browser without reloading the page.
- Prototype allows you to safely and easily deal with Ajax calls. This functionality is supported across browsers.
- Prototype provides modules such as Ajax.Request and Ajax.Updater, which are used for this purpose.

Ajax.Request()

Synopsis:

```
new Ajax.Request(url[, options])
```

Example:

```javascript
new Ajax.Request('/some_url',
        {
                method:'get',
                onSuccess: function(transport){
                var response = transport.responseText ||
"no response text";
                alert("Success! \n\n" + response);
        },
    onFailure: function(){ alert('Something went
wrong...') }
});
```

Ajax.Request

- initialize()
- request()
- setRequestHeaders()
- onStateChange()
- header()
- evalJSON()
- evalResponse()
- respondToReadyState()
- dispatchException()

Back in my day...

```
function ajaxRequest()
{
var xmlHttp;
try
{
// Firefox, Opera 8.0+, Safari
xmlHttp=new XMLHttpRequest();
}
catch (e)
{
// Internet Explorer
try
{
xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
}
catch (e)
{
try
{
xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
}
catch (e)
{
alert("Your browser does not support AJAX!");
return false;
}
}
}
```

```
// continued...
xmlHttp.onreadystatechange=function()
{
if(xmlHttp.readyState==4)
{                   var response = xmlHttp.
responseText;            alert("Success! \n\n" +
response);
}
}
xmlHttp.open("GET","time.asp",true);
xmlHttp.send(null);
}
```

Ajax.Updater()
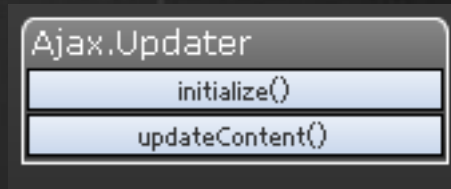
Synopsis:
```
new Ajax.Updater(container, url[, options])
```
Example:
```
new Ajax.Updater('my_placeholder', '/content.php',
      {
            method: 'get';
      });
```

Utility methods

- Prototype provides a number of utility methods.
- These provide functionality such as creating arrays and hashes, or retrieving the value of an element in a form.
- $() == document.getElementById(). It makes use of Prototype's DOM extensions by applying them to elements passed through it.

Utility Methods

$(s|el [,s|el,..])
Takes one or more element ID's or elements and mixes in Element methods

$$(cssSelectors s[,s,s..])
Returns array of elements using CSS Selectors

$A(a)
Returns array with Array and Enumerable methods

$H(o)
Returns array with Hash and Enumerable methods

$w(s)
Splits string on spaces

Try.these(f[,f,f..])
Exits after first successful function

$F = Form.Element.getValue()

$R = new ObjectRange

DOM Extensions

- Biggest part of the Prototype framework.
- Extensions provide methods that aren't native in JavaScript to DOM elements.
- Easy to add your own DOM methods with Element. AddMethods()
- Most browsers (IE excluded) support the adding of methods to native objects (such as HTMLElement), these extensions are available without explicitly calling Element.extend()

Object-oriented support

- Prototype adds Object-oriented support for JavaScript developers.
- A class is created using the Class.create() method. The attributes of the class will all be passed in as the parameters of Class.create().
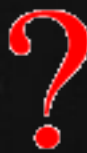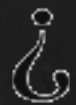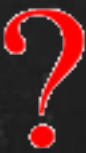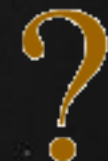
Object-oriented support

- Prototype also allows for inheritance. A class can specify its superclass as the first parameter to Class.create().
- While a subclass does not automatically inherit the methods of its superclass, it can access them via a $super argument as the first parameter of the overriding methods.

prototype

Demo

Questions

The End
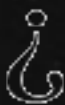
References

- http://www.prototypejs.org/
- http://en.wikipedia.org/wiki/Prototype_Javascript_Framework