SIMPLIFIED JS JAR•GON

TOGGLE PARTICLES ANIMATION

What do you want to learn…

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Prototype

The Prototype JavaScript Framework is a JavaScript framework created by Sam Stephenson in February 2005 as part of the foundation for AJAX support in Ruby on Rails. It is implemented as a single file of JavaScript code, usually named prototype.js. Prototype is distributed standalone, but also as part of larger projects, such as Ruby on Rails, script.aculo.us and Rico.

Features

Prototype provides various functions for developing JavaScript applications. The features range from programming shortcuts to major functions for dealing with XHR.

Prototype also provides library functions to support classes and class-based objects, something the JavaScript language lacks. In JavaScript, object creation is prototype-based instead: an object creating function can have a prototype property, and any object assigned to that property will be used as a prototype for the objects created with that function. The Prototype framework is not to be confused with this language feature.

Sample utility functions

The `$()` function

The **dollar function**, `$()`, can be used as shorthand for the `getElementById` function. To refer to an element in the DOM of an HTML page, the usual function identifying an element is:

```
document.getElementById('id_of_element').style.color =
```

The `$()` function reduces the code to:

```
$('id_of_element').setStyle({ color: '#ffffff' });
```

The `$()` function can also receive an element as parameter and will return, as in the previous example, a prototype extended object.

```
var domElement = document.getElementById('id_of_elemen
var prototypeEnhancedDomElement = $(domElement);
```

The `$F()` function

Building on the `$()` function: the `$F()` function returns the value of the requested form element. For a 'text' input, the function will return the data contained in the element. For a 'select' input element, the function will return the currently selected value.

```
$F('id_of_input_element')
```

The `$$()` function

The dollar dollar function is Prototype's *CSS Selector Engine*. It returns all matching elements, following the same rules as a selector in a CSS stylesheet. For

example, if you want to get all `<a>` elements with the class “pulsate”, you would use the following:

```
$$('a.pulsate')
```

This returns a collection of elements. If you are using the script.aculo.us extension of the core Prototype library, you can apply the “pulsate” (blink) effect as follows:

```
$$('a.pulsate').each(Effect.Pulsate);
```

The Ajax object

In an effort to reduce the amount of code needed to run a cross-browser `XMLHttpRequest` function, Prototype provides the `Ajax` object to abstract the different browsers. It has two main methods: `Ajax.Request()` and `Ajax.Updater()`. There are two forms of the `Ajax` object. `Ajax.Request` returns the raw XML output from an AJAX call, while the `Ajax.Updater` will inject the return inside a specified DOM object. The `Ajax.Request` below finds the current values of two HTML form input elements, issues an HTTP POST request to the server with those element name/value pairs, and runs a custom function (called `showResponse` below) when the HTTP response is received from the server:

```
new Ajax.Request('http://localhost/server_script', {
  parameters: {
    value1: $F('form_element_id_1'),
    value2: $F('form_element_id_2')
  },
  onSuccess: showResponse,
  onFailure: showError
});
```

Object-oriented programming

Prototype also adds support for more traditional object-oriented programming. The `Class.create()` method is used to create a new class. A class is then assigned a `prototype` which acts as a blueprint for instances of the class.

```
var FirstClass = Class.create( {
  // The initialize method serves as a constructor
  initialize: function () {
    this.data = 'Hello World';
  }
});
```

Extending another class:

```
Ajax.Request = Class.create( Ajax.Base, {
  // Override the initialize method
```

```
  initialize: function(url, options) {
    this.transport = Ajax.getTransport();
    this.setOptions(options);
    this.request(url);
  },
  // ...more methods add ...
});
```

The framework function `Object.extend(dest, src)` takes two objects as parameters and copies the properties of the second object to the first one simulating inheritance. The combined object is also returned as a result from the function. As in the example above, the first parameter usually creates the base object, while the second is an anonymous object used solely for defining additional properties. The entire sub-class declaration happens within the parentheses of the function call.

Source:

- Prototype JavaScript Framework↗. *From Wikipedia, the free encyclopedia.*
- Prototype JavaScript framework: a foundation for ambitious web applications↗. *Official site.*
- Prototype JavaScript framework↗. *Github.*

Simplified JavaScript Jargon↗ (short SJSJ) is a community-driven attempt at explaining the loads of buzzwords making the current JavaScript ecosystem in a few simple words. The idea is not to replace individual documentations, but to act as some kind of glossary that can be easily referenced.