Szeged 2008 Drupalcon

JavaScript components

Konstantin Käfer

Contents

1. **Functions and scope**
2. Patterns
3. Drupal’s JavaScript facilities
4. Debugging and Analyzing

Functions

- Functions are **first class entities**
  - Store in variables, pass as parameters, return from functions
  - Can be defined at any place
- Functions can contain properties
- Anonymous functions
- Closures

Functions *(II)*

```
var foo = function(callback) {
  callback();
  return function() {
    print("Returned function called");
  };
};

foo(function() {
  print("Passed function called");
})();

foo.bar = "baz";
```

Prototypal OOP

- JavaScript doesn’t have classes
- Prototype of a function used as base class

```
var Foo = function() { /* ... */ };

Foo.prototype = {
  'bar': function() { /* ... */ },
  'baz': function() { /* ... */ }
};

var instance = new Foo();
instance.bar();
instance.baz();
```

Prototypal OOP *(II)*

- Function is constructor
- “Instances” have an implicit link to the base class

```
var Foo = function() { /* ... */ };
Foo.prototype = {
  'bar': function() { /* ... */ }
};

var instance = new Foo();
instance.bar();

Foo.prototype.baz = function() { /* ... */ };
instance.baz();
```

Prototypal OOP *(III)*

- Any objects can be extended/changed at any time

```
Number.prototype.celsiusToFahrenheit = function() {
  return (this * 9 / 5) + 32;
};

js> (34).celsiusToFahrenheit();
93.2
js> (0).celsiusToFahrenheit();
32
```

Scope

- JavaScript has a lexical (static) scope
- Scope contains everything that is visible when the function is **defined**
- `this` is the context the function is executed in
- Functions can be executed in other contexts

Scope *(II)*

```
var bar = function() {
  var foo = "foo";

  return function() {
    console.log(foo);
  };
}();

js> bar();
foo
```

- Scope lives on even after the outer function returned

Scope *(III)*

```
var formulas = {
  'Celsius': {
    'Fahrenheit': 'this * 1.8 + 32',
    'Reaumur': 'this * 0.8'
  },
  'Fahrenheit': {
    'Celsius': '(this - 32) / 1.8',
    'Reaumur': '(this - 32) / 2.25'
  },
  'Reaumur': {
    'Celsius': 'this * 1.25',
    'Fahrenheit': 'this * 2.25 + 32'
  }
};
```

Scope *(IV)*

```
(function() {
  var formulas = ...;

  for (var from in formulas) {
    for (var to in formulas[from]) {
      Number.prototype[from + 'To' + to] =
      (function(formula) {
        return function() {
          return eval(formula);
        };
      })(formulas[from][to]);
    }
  }
})();
```

Contents

1. Functions and scope
2. **Patterns**
3. Drupal's JavaScript facilities
4. Debugging and Analyzing

Decorator

- Creating an instance that is *slightly* different
- JavaScript allows overwriting methods

```
Function.prototype.decorate = function(pre, post) {
  var old = this;

  return function() {
    if (pre) pre.apply(this, arguments);
    old.apply(this, arguments);
    if (post) post.apply(this, arguments);
  };
};
```

Decorator *(II)*

```
var foo = function() { };
foo.prototype.bar = function(message) {
  print(message);
};

var baz = new foo();
baz.bar = baz.bar.decorate(function() {
  print("pre");
});

baz.bar("message");
```

Delegates

- Delegate tasks to another object
- One object can use many different delegate objects

```
var simpleDataSource = {
  'length': function() { return storage.length; },
  'item': function(i) { return storage[i]; }
};
```

Delegates *(II)*

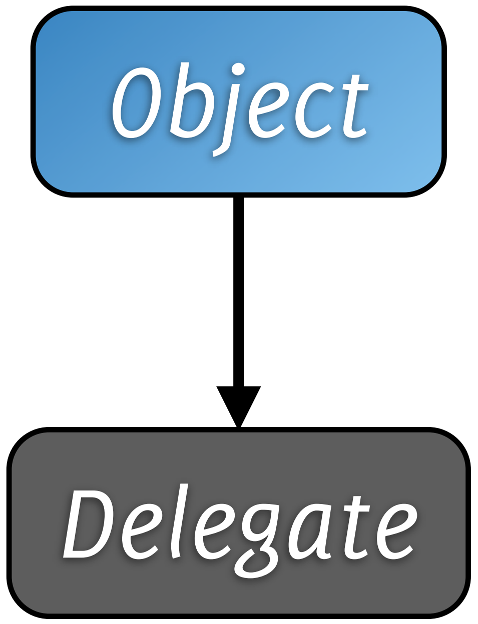
```
var urlDataSource = function(src) {
  this.source = src;
};

urlDataSource.prototype = {
  'length': function() {
    return loadData(this.source,
      { 'command': 'length' });
  },
  'item': function(i) {
    return loadData(this.source,
      { 'command': 'fetch', 'id': i });
  }
};
```

Delegates *(III)*

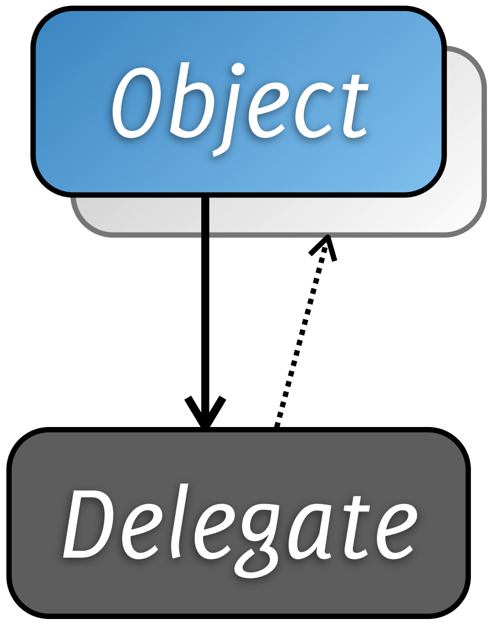
```
var obj = function(datasource) {
  this.datasource = datasource;
};

obj.prototype.foo = function() {
  var length = this.datasource.length();
  for (var i = 0; i < length; i++) {
    var item = this.datasource.item(i);
    // do something with item
  }
};

var foo = new obj(simpleDataSource);
var bar = new obj(new urlDataSource
                    ("http://example.com"));
```

Delegates *(IV)*

- Pull vs. Push



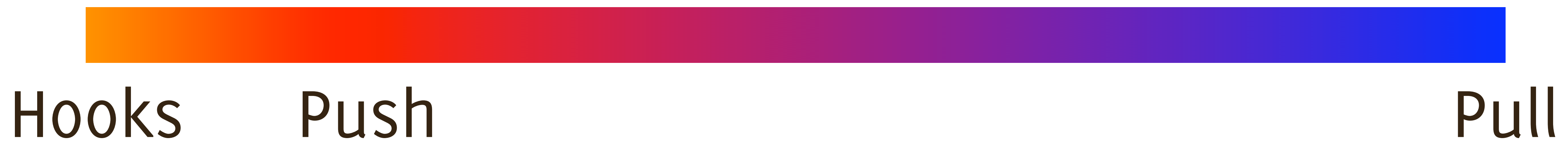
Synchronous Call



Asynchronous Call

(Calls supplied method when the delegated action is finished)

Observer



- Get notifications when something happens
- “Subscribe” to events
- Like module_invoke_all (Drupal’s hooks)

Observer *(II)*

```
var foo = function() { };
foo.prototype.foo = function() {
  for (var func in this.callbacks) {
    this.callbacks[func]();
  }
  // Do something
};

var bar = new foo();

bar.callbacks.push(function() {
  // Hook implementation
});
```

Contents

1. Functions and scope
2. Patterns
3. **Drupal’s JavaScript facilities**
4. Debugging and Analyzing

Adding JavaScript

- `drupal_add_js(...);`
  - `$data`: Path, Array or Code
  - `$type`: `core/module/theme` `setting` `inline`
  - `$scope`: header/footer
  - `$defer`: Delay execution
  - `$cache`: Prevent caching

Adding JS files

```
drupal_add_js(
  drupal_get_path('module', 'mymodule') .
  '/mymodule.js'
);

drupal_add_js(array('mymodule' =>
  'mysetting'), 'setting');

drupal_add_js('$(document).ready(function() {
  alert('Hello World!');
});', 'inline');
```

AJAX Callbacks

- In the menu system:

```
$items['mymodule/js'] = array(
  'title' => 'JavaScript callback',
  'page callback' => 'mymodule_js',
  'access callback' => TRUE,
  'type' => MENU_CALLBACK,
);
```

- Menu callback:

```
function mymodule_js() {
  // Generate $data...
  drupal_json($data);
}
```

Translation

- Similar to PHP
- `Drupal.t('This is a string.');`

-
  ```
  Drupal.t('Do you really want to delete %object?',
    { '%object': object.name });
  ```

-
  ```
  Drupal.formatPlural(count,
    '1 comment', '@count comments');
  ```
- POT extractor and Drupal parse JavaScript files

Behaviors

- All functions in `Drupal.behaviors` are executed onready (when the DOM is ready)
- Functions have to be non-destructive
- Functions get a `context` parameter to act on
- Advantage: Can be called later on as well

-
  ```
  Drupal.behaviors.foo = function(context) {
    $('.foo:not(.foo-processed)', context).each(
      function() { … }
    );
  };
  ```

Theming

- Theme functions for HTML code
- Advantage: Themes can change markup
- In the module’s JavaScript:

```
var elem = Drupal.theme('mymodule');
$('body').append(elem);

Drupal.theme.prototype.mymodule = function() { /*...*/ }
```

- In the theme’s JavaScript:

```
Drupal.theme.mymodule = function() {
  return $('<div class="mymodule"></div>');
}
```

ahah.js

- Similar to AJAX
- Uses iframe to load new HTML
- Integrated into Forms API: `#ahah_path`, `#ahah_effect`, `#ahah_method`, `#ahah_wrapper`
- Only available for types `'button'`, `'submit'` and `'image_button'`

ahah.js (II)

- `#ahah_path`: The Drupal path of the callback to load
- `#ahah_wrapper`: ID of a DOM node where returned HTML will be inserted
- `#ahah_method`: How the HTML will be inserted `(prepend | append | replace | before | after)`
- `#ahah_effect`: Animation effect for the new HTML `(none | slide | fade | …)`

ahah.js (III)

- Return value: JSON

```
{
  'status': true,
  'data': '…'
}
```

- Content of `data` is inserted into the wrapper

Case study: Autocomplete

- Two objects:
  - `jsAC`: Handles the UI
  - `ACDB`: Handles data retrieval
- Attached with `Drupal.behaviors`

Contents

1. Functions and scope
2. Patterns
3. Drupal’s JavaScript facilities
4. **Debugging and Analyzing**

Firebug web development evolved

- Advanced JavaScript console
- Logging to the console (console.log())
- DOM inspector
- JavaScript debugger (with backtrace!)
- Profile JavaScript activity

- http://getfirebug.com

Firebug Lite

- Console for other browsers
- No profiling
- Doesn’t do your laundry

- http://getfirebug.com/lite.html
- http://drupal.org/project/firebug

- JavaScript debugger for IE
- Free of charge
- Some configuration work needed

- http://msdn.microsoft.com/vstudio/express/vwd/

WebDevHelper

- JavaScript console for IE
- HTTP Logger
- JavaScript backtracer

- http://projects.nikhilk.net/WebDevHelper/Default.aspx

JavaScript Lint

- Lint = tool for analyzing code
- Discovers sloppy coding
- Command line interface
- Use as pre-commit hook for `<insert RCS>`

- http://javascriptlint.com/
- TextMate bundle: http://andrewdupont.net/2006/10/01/javascript-tools-textmate-bundle/