

How to Develop Large Scalable JavaScript Applications

By – Narendra Sisodiya



Split your application into multiple
Modules

setup

remove selected

|

new contact

Search...

A



Afiyb Turlet

Product Web Assistant
(415) 711-1234



Agair Rukge

District Research Consultant
(415) 711-1234



Alusjo Epusiesavr

Internal Markets Administrator
(415) 711-1234



Amoop Vurlu

Product Web Assistant
(415) 711-1234

Toolbar Module

Setting Module

AddNewContacts Module

Search Module

Contact Area Module

ContactList Module

Rules

Each Module developed separately.

Each Module can be
loaded/unloaded from Page

Rules

You can load any module and test it without loading others.

If (at run time) one module fails to load. Other modules should work.
There should be minimum side effect in such case

What does a module do ?

It does its work perfectly. Think like a two IFRAMEs, we can load any type of web page, both work independently.

Each module will be given one DOM element where it will load/play its functionality.

Ofcourse, Module can talk to each other easily.

Rules

Module **MUST** not modify any DOM element outside its DOM container.

Lifecycle of Module

Module - start

Create HTML

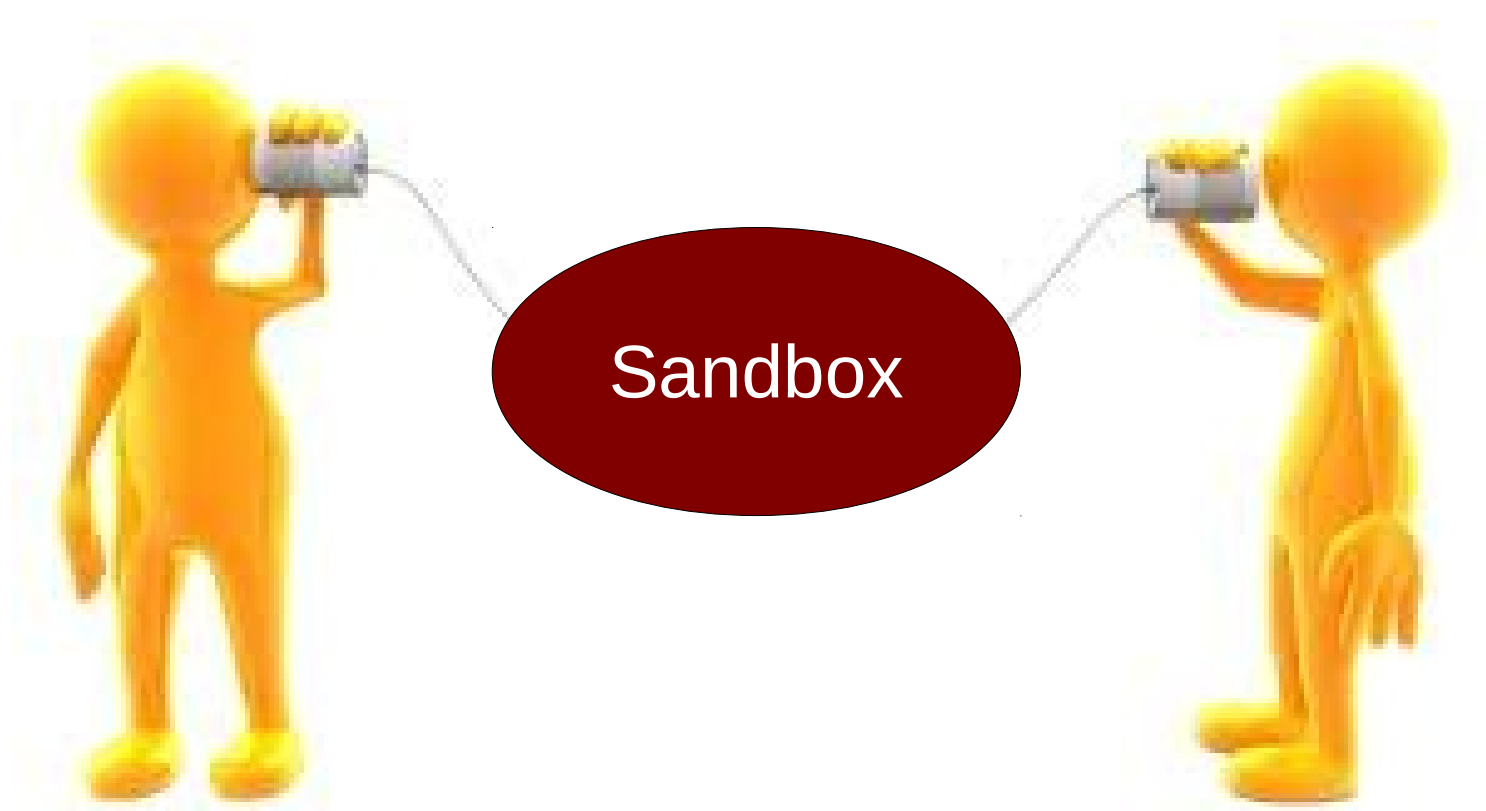
Load Data from Server

Assign Callback

Pub/Sub Callback

Load other Module if any

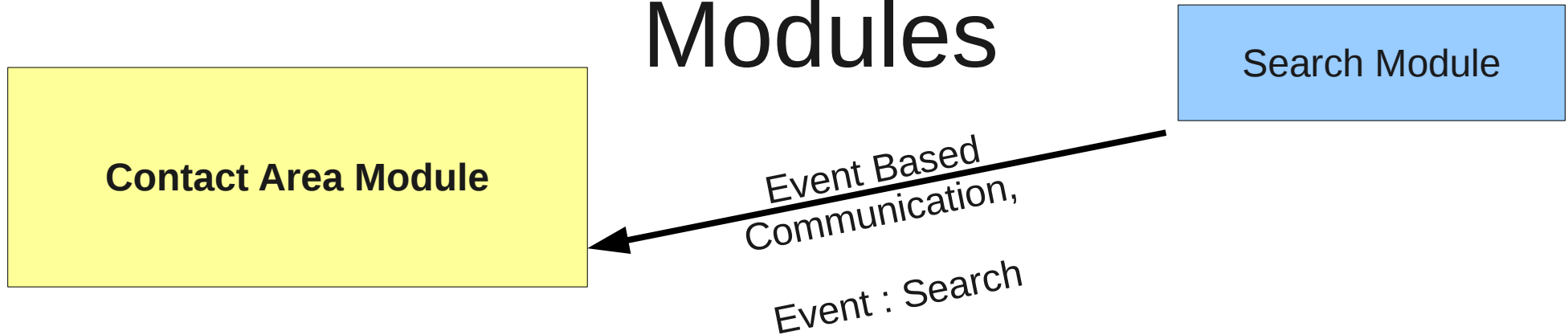
Module - end



Module talk to each other using
Pub/Sub Architecture,

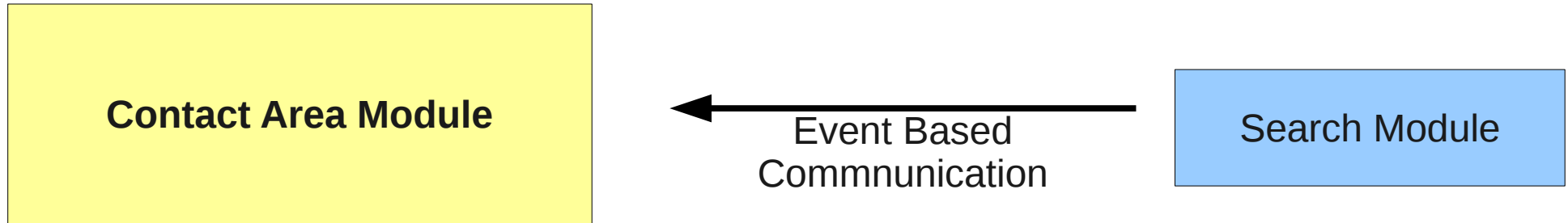
Event based communication.
Via - sandbox

How a Module Talks to Other Modules



If user types something in search module, it will send 'search string' to Contact Area Module.

Example



If Search module fails to load, user will not be able to search.

But Contact Area module must be working perfectly. Both modules are loosely coupled.

How a Module Talks to Other Modules



If you add a “new Contact”

AddNewContastModule sends new contact information to server (in Ajax) and gets response.

If it gets proper response then it sends “Reload Event” to Contact Area Module

How a Module Talks to Other Modules



If Search module fails then user will not be able to search.

But Contact Area module must be working perfectly. Both modules are loosely coupled.

Application Start here

```
<html>
<head>
  <script type="text/javascript" src="/external-libs/jquery-1.7.1.js"></script>
  <script type="text/javascript" src="/external-libs/lb-min-2012-08-04-0752.js"></script>
  <script type="text/javascript" src="/elpaJsRepo/index.js"></script>
  <link rel="stylesheet" href="/elpaJsRepo/blogReaderApp/index.css" type="text/css">
  <script type="text/javascript" src="/elpaJsRepo/blogReaderApp/index.js"></script>
  <script type="text/javascript"
src="/elpaJsRepo/blogReaderApp/blogDisplayPanel/index.js"></script>
  <script type="text/javascript" src="/elpaJsRepo/blogReaderApp/navigator/index.js"></script>
  <script type="text/javascript" src="/elpaJsRepo/blogReaderApp/footer/index.js"></script>
  <script type="text/javascript" src="/elpaJsRepo/blogReaderApp/header/index.js"></script>

  <script type="text/javascript">
    $(function(){
      var application = new lb.core.Module("applicationContainer",
elpaJsRepo.blogReaderApp);
      application.start();

    });
  </script>

</head> <body> <div id="applicationContainer"></div> </body> </html>
```

Who will initialize this module?

```
elpaJsRepo.blogReaderApp = function(sandbox){  
  var $_ = commonLbConstructor;  
  
  $_.prototype = {  
    start: function(){  
      this.initHTML();  
      this.loadModules();  
    },  
    initHTML: function(){  
      this.$.append('\n  
        <div class="footer" id="'+ this.id +'_footer"></div>\n  
      ');  
    },  
    loadModules: function(){  
      this.footerModule = new lb.core.Module(this.id + "_footer",  
elpaJsRepo.blogReaderApp.footer);  
      this.footerModule.start();  
    }  
  };  
  return new $_(sandbox);  
}
```


Coding – How to create a module ?

```
elpaJsRepo.blogReaderApp.footer = function(sandbox){  
  
    var $_ = commonLbConstructor ;  
  
    $_.prototype = {  
        start: function(){  
            this.initHTML();  
  
        },  
        end: function(){  
  
        },  
        initHTML: function(){  
            this.$.append("<p>THIS IS FOOTER PANEL</p>");  
        }  
    };  
    return new $_(sandbox);  
};
```

Summary

A Module loads in an EMPTY div element.

Ex `<div id="applicationContainer"></div>`

Now you can load a module in this container using:

```
var application = new lb.core.Module( "applicationContainer",  
    elpaJsRepo.blogReaderApp);  
application.start();
```

Or simpler – `(new lb.core.Module(<id>, <function>)).start();`

Using this, start function will be executed

Inside the module -

```
this.$ 'is equal to' $("#applicationContainer");
```

So that you can perform DOM operations on the container element

Sending Events

Subscribe – Module 1

```
this.sandbox.events.subscribe(  
    {event: 'onBlogLinkSelected'},  
    function(args){  
        self.loadBlog(args.value);  
    }  
);
```

Publish – Module 2

```
this.sandbox.events.publish({event: 'onBlogLinkSelected', value: $(this).data("blogid") });
```

Links

Video - (Nicholas Zakas: Scalable JavaScript Application Architecture)

<http://www.youtube.com/watch?v=vXjVFPosQHw>

Slides - <http://www.slideshare.net/nzakas/scalable-javascript-application-architecture>

Blog - <http://www.ubelly.com/2011/11/scalablejs/>

3 Implementations

<https://github.com/flosse/scaleApp>

<https://github.com/aranm/scalable-javascript-architecture>

https://github.com/eric-brechemier/lb_js_scalableApp

I have used lb_js_scalableApp for this Demo

<https://github.com/nsisodiya/Demo-Scalable-App>