The GovData Project
MIT-Harvard Winter Course 2011

Module 4:
Introduction to Javascript Visualizations
How the Web Works Revisited

The real contents of the response.

It’s HTML.
How the Web Works Revisited

The real contents of the response.

It’s HTML.

But it can also be Javascript.
Server computes the response.
How the Web Works (sort of)

Client computes the response.
How the Web Works (sort of)

So the modes are:

1) HTML (server-side computation)

2) Javascript (client-side computation)

3) AJAX (constant communication)

Ajax (programming)

From Wikipedia, the free encyclopedia

For other uses, see Ajax (disambiguation).

Ajax (pronounced /ædʒɪk/; shorthand for Asynchronous JavaScript and XML)[1] is a group of interrelated web development methods used on the client-side to create interactive web applications. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. Data is usually retrieved using the XMLHttpRequest object. Despite the name, the use of XML is not needed, and the requests need not be asynchronous.[2]

Like DHTML and LAMP, Ajax is not one technology, but a group of technologies. Ajax uses a combination of HTML and CSS to mark up and style information. The DOM is accessed with JavaScript to dynamically display, and to allow the user to interact with the information presented. JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources

2. Use good JS libraries to make JS not suck

3. Use cutting-edge visualization properties of HTML5 for good visualizations

4. Modularize your widgets so that they can talk in generic ways.
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources

```html
<html>
<head>
<script>
// define some functions
// a rarely as possibly, call general-purpose APIs
// to obtain data as JSON or XML
</script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
```
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources.

Q: What would an alternative look like:

A: You’d rely on server-side computation for:
- data transformations
- layout and templating (django / rails)
- responding to little AJAX requests

Instead, we recommend:
- do data transformations & event handling in JS
- do layout in static HTML + templating
- do AJAX just for “real” data sources
Javascript: Principles

1. Use pure javascript; not too much reliance on AJAX; just call general APIs for data sources.

Advantages:

1) offloads computations

2) one single, VERY portable file

Disadvantages:

1) makes everything public (not a disadvantage for us)

2) JS sucks, wouldn’t you rather write your business logic in python?

which leads to our second principle ...
2. Use good JS libraries to make JS not suck
Javascript: Principles

2. Use good JS libraries to make JS not suck

   i. jQuery (practically part of stdlib)

      object selecting & event handling

   ii. underscore.js

      better basic data structures (like arrays)

   iii. requireJS

      modularization, dependency tracking, minification

   iv. jQuery.address

      deep linking
i. jQuery (practically part of stdlib)

http://jquery.com/
Javascrip: jQuery

The basic thing that jQuery does is it makes JS a great language in which to build simple GUIs.

Four basic ideas:

i. selecting objects in the page.

ii. handling interactive events in the page.

iii. manipulating (changing) the page

iv. doing AJAX to communicate off the page

The combination of these four is powerful, and jQuery makes them easy in JS.
CDN HOSTED JQUERY

A number of large enterprises provide hosted copies of jQuery on existing CDN networks that are available for public use. Below are links to the CDN-hosted copies of jQuery that you may hotlink to.

```html
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<!-- static structure here -->
</body>
</html>
Javascipt: jQuery

http://docs.jquery.com/Main_Page

JQUERY API REFERENCE

- jQuery Core
- Selectors
- Attributes
- Traversing
- Manipulation
- CSS
- Events
- Effects
- Ajax
- Utilities
- Internals

There are a number of alternative resources for browsing the API.
Javascript: jQuery

```html
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></script>
</head>
<body>
<div class="box" id="box1">Object 1</div>
<div class="box" id="box2">Object 2</div>
</body>
</html>
```
The Basic Syntax:

\[
\$(\text{selector string}).\text{eventName}(\text{function}(e)\{
    //\text{event response code}
\})
\]

This syntax says:

i. find all the objects in the page that match the selector string

ii. and then, bind the event response code to those objects, to be called whenever “eventName” event actually occurs
Javascript: jQuery

The Basic Syntax:

$(nearly any JS object)

= that something, “wrapped” as a jQuery object, with all the good methods jQuery provides

$(selector string)

= a list of all the things that match the selector string on the page, returned as jQuery objects
Javascript: jQuery

The Basic Syntax:

\$ (selector string)

\$ ("tag")

Get me all objects whose tag is “tag”

\$ (".cname")

Get me all objects whose class attribute is “cname”

\$ ("#idval")

Get me all objects whose id attribute is “idval”
Javascript: jQuery

The Basic Syntax:

\$\text{(selector string)}

\$\text{\text{"tag\"\)}}
Get me all objects whose tag is “tag”  \$\text{\text{"div\"\)}}

\$\text{\text{\".cname\"\)}}
Get me all objects whose class attribute is “cname”  \$\text{\text{\".box\"\)}}

\$\text{\text{\"#idval\"\)}}
Get me all objects whose id attribute is “idval”  \$\text{\text{\"#box1\"\)}}
Javascript: jQuery

```html
<body>
  <div class="box" id="box1">Object 1</div>
  <div class="box" id="box2">Object 2</div>
</body>
```

```javascript
$("div")
  [Object 1, Object 2]
```
Javascript: jQuery

```
$(".box")

<body>

<div class="box" id="box1">Object 1</div>
<div class="box" id="box2">Object 2</div>

</body>
```

```
> $(".box")
[<div class="box" id="box1">Object 1</div>, <div class="box" id="box2">Object 2</div>]
> |
```
Javascript: jQuery

```html
<body>
  <div class="box" id="box1">Object 1</div>
  <div class="box" id="box2">Object 2</div>
</body>
```

```javascript
> $('".box")
  [<div class="box" id="box1">Object 1</div>, <div class="box" id="box2">Object 2</div>]

> $('"#box1")
  [ <div class="box" id="box1">Object 1</div> ]
```
### Javascript: jQuery

Lots of other kinds of selector methods:

<table>
<thead>
<tr>
<th>Selector Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Selector (**)</td>
<td>Selects all elements.</td>
</tr>
<tr>
<td>:animated Selector</td>
<td>Selects all elements that are in the progress of an animation at the time the selector is run.</td>
</tr>
<tr>
<td>Attribute Contains Prefix Selector [name</td>
<td>Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-).</td>
</tr>
<tr>
<td>Attribute Contains Selector [name*=&quot;value&quot;]</td>
<td>Selects elements that have the specified attribute with a value containing the a given substring.</td>
</tr>
<tr>
<td>Attribute Contains Word Selector [name-=&quot;value&quot;]</td>
<td>Selects elements that have the specified attribute with a value containing a given word, delimited by spaces.</td>
</tr>
<tr>
<td>Attribute Ends With Selector [name$=&quot;value&quot;]</td>
<td>Selects elements that have the specified attribute with a value ending exactly with a given string. The comparison is case sensitive.</td>
</tr>
<tr>
<td>Attribute Equals Selector [name=&quot;value&quot;]</td>
<td>Selects elements that have the specified attribute with a value exactly equal to a certain value.</td>
</tr>
<tr>
<td>Attribute Not Equal Selector [name!=&quot;value&quot;]</td>
<td>Selects elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.</td>
</tr>
<tr>
<td>Attribute Starts With Selector [name^=&quot;value&quot;]</td>
<td>Selects elements that have the specified attribute with a value beginning exactly with a given string.</td>
</tr>
<tr>
<td>:button Selector</td>
<td>Selects all button elements and elements of type button.</td>
</tr>
<tr>
<td>:checkbox Selector</td>
<td>Selects all elements of type checkbox.</td>
</tr>
<tr>
<td>:checked Selector</td>
<td>Matches all elements that are checked.</td>
</tr>
<tr>
<td>:eq() Selector</td>
<td>Select the element at index n within the matched set.</td>
</tr>
<tr>
<td>:even Selector</td>
<td>Selects even elements, zero-indexed. See also :odd.</td>
</tr>
<tr>
<td>:file Selector</td>
<td>Selects all elements of type file.</td>
</tr>
<tr>
<td>:first-child Selector</td>
<td>Selects all elements that are the first child of their parent.</td>
</tr>
<tr>
<td>:first Selector</td>
<td>Selects the first matched element.</td>
</tr>
<tr>
<td>:gt() Selector</td>
<td>Select all elements at an index greater than n within the matched set.</td>
</tr>
<tr>
<td>Has Attribute Selector [name]</td>
<td>Selects elements that have the specified attribute, with any value.</td>
</tr>
<tr>
<td>:has() Selector</td>
<td>Selects elements which contain at least one element that matches the specified selector.</td>
</tr>
<tr>
<td>:header Selector</td>
<td>Selects all elements that are headers, like h1, h2, h3 and so on.</td>
</tr>
<tr>
<td>:hidden Selector</td>
<td>Selects all elements that are hidden.</td>
</tr>
<tr>
<td>ID Selector (&quot;#id&quot;)</td>
<td>Selects a single element with the given id attribute.</td>
</tr>
<tr>
<td>:image Selector</td>
<td>Selects all elements of type Image.</td>
</tr>
</tbody>
</table>
The Basic Syntax: doing stuff with selected results

\[
\$(\text{selector string}).\text{method} \ (\ \text{args})
\]

... means, “apply the method to ALL the objects that match the selector string.”

\[
\$(\text{selector string}).\text{eventName} \ (\ \text{args})
\]

... means, “bind the event handler specified by ‘args’ to ALL objects matching the selector string.”
Javascript: jQuery

The Basic Syntax: where “doing stuff” == “binding event”

```javascript

$("div").eventName(function(e){
    //event response code

});
```

This syntax says:

i. find all the “div”s in the page

ii. bind the event response to be called whenever “eventName” event actually occurs on a “div”
<script>
$(document).ready(function(){
    $('div').click(function(){
        alert("HERE")
    });
});
</script>
Javascript: jQuery

Lots of event handlers:

.event.bind()
Attach a handler to an event for the elements.

.event.blur()
Bind an event handler to the "blur" JavaScript event, or trigger that event.

.event.change()
Bind an event handler to the "change" JavaScript event, or trigger that event.

.event.click()
Bind an event handler to the "click" JavaScript event, or trigger that event.

.event.delegate()
Attaches a handler to one or more events for all elements that match on a specific set of root elements.

.event.destroy()
Remove all event handlers previously attached using .live() from the element.

.event.error()
Bind an event handler to the "error" JavaScript event.

.event.currentTarget
The current DOM element within the event bubbling phase.

.event.data
The optional data passed to jQuery.fn.bind when the current event executes.

.event.isDefaultPrevented()
Returns whether .preventDefault() was ever called on this event.

.event.isImmediatePropagationStopped()
Returns whether .stopImmediatePropagation() was ever called.

.event.pageX
The mouse position relative to the left edge of the document.

.event.pageY
The mouse position relative to the top edge of the document.

.event.preventDefault()
If this method is called, the default action of the event will not be triggered.

.event.relatedTarget
The other DOM element involved in the event, if any.

.event.result
The last value returned by an event handler that was triggered by this event, as undefined.

.event.stopImmediatePropagation()
Prevents other event handlers from being called.

.event.stopPropagation()
Prevents the event from bubbling up the DOM tree, preventing any parent hand event.

.event.target
The DOM element that initiated the event.

.event.timeStamp
The difference in milliseconds between the time the event is triggered and January 1, 1970

.event.type
Describes the nature of the event.

.event.which
For key or button events, this attribute indicates the specific button or key that was pressed.

.event.focus()
Bind an event handler to the "focus" JavaScript event, or trigger that event on an element.

.event.focusin()
Bind an event handler to the "focusin" JavaScript event.

.event.mousedown()
Bind an event handler to the "mousedown" JavaScript event, or trigger that event.

.event.mouseenter()
Bind an event handler to be fired when the mouse enters an element.

.event.mouseleave()
Bind an event handler to be fired when the mouse leaves an element.

.eventmousemove()
Bind an event handler to the "mousemove" JavaScript event, or trigger that event.

.event.mousemove()
Bind an event handler to the "mousemove" JavaScript event, or trigger that event.

.event.mouseout()
Bind an event handler to the "mouseout" JavaScript event, or trigger that event.

.event.mouseover()
Bind an event handler to the "mouseover" JavaScript event, or trigger that event.

.event.mouseup()
Bind an event handler to the "mouseup" JavaScript event, or trigger that event.

.event.one()
Attach a handler to an event for the elements. The handler is executed only once.

.event.proxy()
Takes a function and returns a new one that will always have a particular context.

.event.ready()
Specify a function to execute when the DOM is fully loaded.

.event.resize()
Bind an event handler to the "resize" JavaScript event, or trigger that event.

.event.scroll()
jQuery does page manipulations:

```javascript
$(selector string).eventName(function(e){
    $(selector).manipulateSomehow()
});
```
Javascript: jQuery

jQuery does page manipulations:

```
<script>
  $(document).ready(function(){
    $('#box1').click(function(){
      $('#box2').append('<div class="hbox">hello</div>');
    });
  });
</script>

“When you click on box1, add “hello” text to box 2.”
```
“When you click on box1, add “hello” text to box 2. But if you click on box 2, remove all the hello text objects.”
Javascript: jQuery

jQuery does page manipulations:

“When you click on box1, add “hello” text to box 2.

But if you click on box 2, remove all the hello text objects ... and count them.”
# Javascript: jQuery

Lots of manipulations.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>.addClass()</code></td>
<td>Adds the specified class(es) to each of the set of matched elements.</td>
</tr>
<tr>
<td><code>.after()</code></td>
<td>Insert content, specified by the parameter, after each element in the set.</td>
</tr>
<tr>
<td><code>.append()</code></td>
<td>Insert content, specified by the parameter, to the end of each element in the set.</td>
</tr>
<tr>
<td><code>.appendTo()</code></td>
<td>Insert every element in the set of matched elements to the end of the target.</td>
</tr>
<tr>
<td><code>.attr()</code></td>
<td>Get the value of an attribute for the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.before()</code></td>
<td>Insert content, specified by the parameter, before each element in the set.</td>
</tr>
<tr>
<td><code>.clone()</code></td>
<td>Create a deep copy of the set of matched elements.</td>
</tr>
<tr>
<td><code>.css()</code></td>
<td>Get the value of a style property for the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.detach()</code></td>
<td>Remove the set of matched elements from the DOM.</td>
</tr>
<tr>
<td><code>.empty()</code></td>
<td>Remove all child nodes of the set of matched elements from the DOM.</td>
</tr>
<tr>
<td><code>.hasClass()</code></td>
<td>Determine whether any of the matched elements are assigned the given class.</td>
</tr>
<tr>
<td><code>.height()</code></td>
<td>Get the current computed height for the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.html()</code></td>
<td>Get the HTML contents of the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.innerHeight()</code></td>
<td>Get the current computed height for the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.outerWidth()</code></td>
<td>Get the current computed width for the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.position()</code></td>
<td>Get the current coordinates of the first element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.prepend()</code></td>
<td>Insert content, specified by the parameter, to the beginning of each element in the set.</td>
</tr>
<tr>
<td><code>.prependTo()</code></td>
<td>Insert every element in the set of matched elements to the beginning of the target.</td>
</tr>
<tr>
<td><code>.remove()</code></td>
<td>Remove the set of matched elements from the DOM.</td>
</tr>
<tr>
<td><code>.removeAttr()</code></td>
<td>Remove an attribute from each element in the set of matched elements.</td>
</tr>
<tr>
<td><code>.removeClass()</code></td>
<td>Remove a single class, multiple classes, or all classes from each element in the set.</td>
</tr>
<tr>
<td><code>.replaceAll()</code></td>
<td>Replace each target element with the set of matched elements.</td>
</tr>
<tr>
<td><code>.replaceWith()</code></td>
<td>Replace each element in the set of matched elements with the provided content.</td>
</tr>
<tr>
<td><code>.scrollLeft()</code></td>
<td>Get the current horizontal position of the scroll bar for the first element in the set.</td>
</tr>
<tr>
<td><code>.scrollTop()</code></td>
<td>Get the current vertical position of the scroll bar for the first element in the set.</td>
</tr>
<tr>
<td><code>.text()</code></td>
<td>Get the combined text contents of each element in the set of matched elements.</td>
</tr>
</tbody>
</table>
Javascript: jQuery

jQuery does AJAX

```javascript
$(selector string).eventName(function(e){

    var send_data = get_data(e);

    $.ajax({
        url: "http://my_api_url",
        data : send_data
        success : doSomethingWithResults
    });
});
```

where `doSomethingWithResults` is a function that does jQuery manipulations and `get_data` computes arguments from data about the event.
"When you click on box1, ask the API how many records there are in BEA_NIPA collection and show the result in box2."
Javascript: jQuery

so that the search only happens when you press “Enter”

Here’s where the input box is put in. It’s a standard HTML thing.
Javascript: Principles

2. Use good JS libraries to make JS not suck

i. jQuery (practically part of stdlib)
   object selecting & event handling

ii. underscore.js
   better basic data structures (like arrays)

iii. requireJS
   modularization, dependency tracking, minification

iv. jQuery.address
   deep linking
Well, the basic idea is that:

a. Real programming languages have good data structures.

b. So Javascript should have them too.
Javascript: Underscore

ii. underscore.js  

Better basic data structures (like arrays)

Object-Oriented and Functional Styles

Collections
- each, map, reduce, reduceRight, detect, select, reject, all, any, include, invoke, pluck, max, min, sortBy, sortedIndex, toArray, size

Arrays
- first, rest, last, compact, flatten, without, uniq, intersect, zip, indexOf, lastIndexOf, range

Functions
- bind, bindAll, memoize, delay, defer, throttle, debounce, wrap, compose

Objects
- keys, values, functions, extend, clone, tap, isEqual, isEmpty, isElement, isArray, isArguments, isFunction, isString, isNumber, isBoolean, isDate, isRegExp, isNaN, isNull, isUndefined

Utility
- noConflict, identity, times, mixin, uniq, and, template

Chaining
- chain, value
ii. underscore.js  

better basic data structures (like arrays)

```javascript
_.each([1, 2, 3], function(num){ alert(num); });
=> alerts each number in turn...
_.each({one : 1, two : 2, three : 3}, function(num, key){ alert(num); });
=> alerts each number in turn...

_.all([true, 1, null, 'yes']);
=> false

_.intersect([1, 2, 3], [101, 2, 1, 10], [2, 1]);
=> [1, 2]

_.zip(['moe', 'larry', 'curly'], [30, 40, 50], [true, false, false]);
=> [['moe', 30, true], ['larry', 40, false], ['curly', 50, false]]

_.keys({one : 1, two : 2, three : 3});
=> ['one', 'two', 'three']
```
Javascrip: Underscore

ii. underscore.js  better basic data structures (like arrays)

And the “right” idea of equality for regular situations.

```javascript
var moe = {name: 'moe', luckyNumbers: [13, 27, 34]};
var clone = {name: 'moe', luckyNumbers: [13, 27, 34]};
moe == clone;
=> false
_.isEqual(moe, clone);
=> true
```

Underscore.strings, an Underscore extension that adds functions for string-manipulation: trim, startsWith, contains, capitalize, reverse, sprintf, and more.
2. Use good JS libraries to make JS not suck

i. jQuery (practically part of stdlib)
   object selecting & event handling

ii. underscore.js
   better basic data structures (like arrays)

iii. requireJS
   modularization, dependency tracking, minification

iv. jQuery.address
   deep linking
The basic idea is that:

a. Real programming languages modularize code.

b. So Javascript should too.

requireJS is a little like Javascript’s answer to the idea of “import” in python.
Javascript: requireJS

iii. requireJS modularization, dependency tracking, minification

http://requirejs.org/

/* ---

RequireJS is a JavaScript file and module loader. It is optimized for in-browser use, but it can be used in other JavaScript environments, like Rhino and Node. Using a modular script loader like RequireJS will improve the speed and quality of your code.

IE 6+ .......... compatible ✓
Firefox 2+ ...... compatible ✓
Safari 3.2+ ..... compatible ✓
Chrome 3+ ...... compatible ✓
Opera 10+ ...... compatible ✓

Get started then check out the API.

--- */
iii. requireJS

**modularization, dependency tracking, minification**

**project.html**

```html
<html>
<head>
  <script data-main="main" src="scripts/require.js"></script>
</head>
<body>
<!-- static contents here -->
</body>
</html>
```

**main.js**

```javascript
require(['"helper/util"'],function() {
    require.read(function(){

        //When this is called, stuff defined in
        //helper.utils is available
        //But it doesn't enter a global namespace.

    });
});
```
Javascript: requireJS

iii. requireJS  modularization, dependency tracking, minification

```javascript
require(['http://gov.thedata.org/data?collectionName=BEA_NIPA&action=count&callback=define'],
  function(count){
    console.log(count)
  });

define(['/path/to/mod1', '/path/to/mod2'], function(mod1, mod2){
  var newFunc(a,b){
    return mod1.func1(a,b) + mod2.constant
  }
});
```
When the page has Javascript, the message contains the JS source code. So you want it to be short.
 iii. requireJS  modularization, dependency tracking, minification

OPTIMIZE

§ 3

Once you are finished doing development and want to deploy your code for your end users, you can use the optimization to combine the JavaScript files together and minify it. In the example above, it can combine main.js and helper/util.js into one file and minify it using Google’s Closure Compiler.

../../requirejs/build/build.sh app.build.js
Javascript: requireJS

`../requirejs/build/build.sh app.build.js`
Javascript: requireJS

When the page has Javascript, the message contains the JS source code. So you want it to be short.

And readability doesn’t count.
2. Use good JS libraries to make JS not suck

i. jQuery (practically part of stdlib)
   object selecting & event handling

ii. underscore.js
    better basic data structures (like arrays)

iii. requireJS
    modularization, dependency tracking, minification

iv. jQuery.address
    deep linking
Deep linking
From Wikipedia, the free encyclopedia

On the World Wide Web, deep linking is making a hyperlink that points to a specific page or image on a website, instead of that website's main or home page. Such links are called deep links.

Deep linking and web technologies

Websites which are built on web technologies such as Adobe Flash and AJAX often do not support deep linking. This can result in usability problems for people visiting such websites. For example, visitors to these websites may be unable to save bookmarks to individual pages or states of the site, web browser forward and back buttons may not work as expected, and use of the browser's refresh button may return the user to the initial page.
Javascript: Deep Linking

iv. jQuery.address  deep linking

Deep linking and web technologies

Websites which are built on web technologies such as Adobe Flash and AJAX often do not support deep linking. This can result in usability problems for people visiting such websites. For example, visitors to these websites may be unable to save bookmarks to individual pages or states of the site, web browser forward and back buttons may not work as expected, and use of the browser’s refresh button may return the user to the initial page.

Usually, DB-backed pages have URLs that look like this:

http://base_address/api_path/?queryString

where queryString is like “key1=val1&key2=val2”

The Javascript actions don’t get registered in this scheme.
Javascript: Deep Linking

iv. jQuery.address  

http://www.asual.com/jquery/address/

**jQuery Address - Deep linking for the masses**

The jQuery Address plugin provides powerful deep linking capabilities and allows the creation of unique virtual addresses that can point to a website section or an application state. It enables a number of important capabilities including:

- Bookmarking in a browser or social website
- Sending links via email or instant messenger
- Finding specific content using the major search engines
- Utilizing browser history and reload buttons
Javascript: Deep Linking

Basic idea:

- make sure every type of "state change" in the JS page is encoded in the URL, via jQuery events (make sure to separate the way that the api variables and the JS states are encoded)

- load up the page based on parsing the encoded URL, where the API sees the variables and the JS gets the state encoding

- use a tool like jQuery.address to do the encoding & decoding
Javascript: Deep Linking

http://base_address/#/api_path/?queryString#encoded_state

```javascript
require(['find', 'jquery', 'jquery.address'],
  function(find) {
    $.address.init(function(e) {
      
      }).externalChange(function(e) {
        var state = $.address.jsonhash();
        state = state || {};
        var params = $.address.parameters();
        if (e.path === 'show') { // show
          console.log("SHOW");
          show.load(params, state);
        } else { // find
          $.address.path('find');
          find.load(params, state);
        }
      });
    
  });
```
### Javascript: Deep Linking

#### The GovData Project

<table>
<thead>
<tr>
<th>Total Slices</th>
<th>Filter date by:</th>
<th>Cluster by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>2681</td>
<td>Year</td>
<td>Agency</td>
</tr>
<tr>
<td></td>
<td>Quarter</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Month</td>
<td></td>
</tr>
</tbody>
</table>

- Department of Commerce > Bureau of Economic Analysis > National Economic Accounts > NIPA Tables (2681 Slices)
HTML5

<nav>    <footer>

<audio>  <video>

<canvas>

inline svg

emphasis on easing JS integration
Scalable Vector Graphics

From Wikipedia, the free encyclopedia

"SVG" redirects here. For other uses, see SVG (disambiguation).

Scalable Vector Graphics (SVG) is a family of specifications of an XML-based file format for describing two-dimensional vector graphics, both static and dynamic (i.e. interactive or animated).
## Javascript + SVG

Variety of Elements to combine.

### SVG Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Defines a hyperlink</td>
</tr>
<tr>
<td>altGlyph</td>
<td>Allows control over glyphs used to render particular character data (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>altGlyphDef</td>
<td>Defines a set of glyph substitutions (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>altGlyphItem</td>
<td>Defines a candidate set of glyph substitutions (e.g., for music symbols or Asian text)</td>
</tr>
<tr>
<td>animate</td>
<td>Animates an attribute or property over time</td>
</tr>
<tr>
<td>animateColor</td>
<td>Specifies a color transformation over time</td>
</tr>
<tr>
<td>animateMotion</td>
<td>Causes an element to move along a motion path</td>
</tr>
<tr>
<td>animateTransform</td>
<td>Animates a transformation attribute on an element</td>
</tr>
<tr>
<td>circle</td>
<td>Defines a circle</td>
</tr>
<tr>
<td>clipPath</td>
<td>Specifies a color profile description</td>
</tr>
<tr>
<td>color-profile</td>
<td>Specifies a color profile description</td>
</tr>
<tr>
<td>cursor</td>
<td>Defines a platform-independent cursor</td>
</tr>
<tr>
<td>definition-src</td>
<td>Defines a separate font definition resource</td>
</tr>
<tr>
<td>defs</td>
<td>A container for referenced elements</td>
</tr>
<tr>
<td>desc</td>
<td>A text-only description for elements in SVG - not displayed as part of the graphics. User agents may display the text as a tooltip</td>
</tr>
<tr>
<td>ellipse</td>
<td>Defines an ellipse</td>
</tr>
<tr>
<td>feBlend</td>
<td>SVG filter. Composites two objects together using different blending modes</td>
</tr>
<tr>
<td>feColorMatrix</td>
<td>SVG filter. Applies a matrix transformation</td>
</tr>
<tr>
<td>feComponentTransfer</td>
<td>SVG filter. Performs component-wise remapping of data</td>
</tr>
<tr>
<td>feComposite</td>
<td>SVG filter.</td>
</tr>
<tr>
<td>feConvolveMatrix</td>
<td>SVG filter.</td>
</tr>
<tr>
<td>feDiffuseLighting</td>
<td>SVG filter.</td>
</tr>
<tr>
<td>font-face-url</td>
<td></td>
</tr>
<tr>
<td>foreignObject</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>A container element for grouping together related elements</td>
</tr>
<tr>
<td>glyph</td>
<td>Defines the graphics for a given glyph</td>
</tr>
<tr>
<td>glyphRef</td>
<td>Defines a possible glyph to use</td>
</tr>
<tr>
<td>hidern</td>
<td></td>
</tr>
<tr>
<td>image</td>
<td></td>
</tr>
<tr>
<td>line</td>
<td>Defines a line</td>
</tr>
<tr>
<td>linearGradient</td>
<td>Defines a linear gradient</td>
</tr>
<tr>
<td>marker</td>
<td></td>
</tr>
<tr>
<td>mask</td>
<td></td>
</tr>
<tr>
<td>metadata</td>
<td>Specifies metadata</td>
</tr>
<tr>
<td>missing-glyph</td>
<td></td>
</tr>
<tr>
<td>mpath</td>
<td></td>
</tr>
<tr>
<td>path</td>
<td>Defines a path</td>
</tr>
<tr>
<td>pattern</td>
<td></td>
</tr>
<tr>
<td>polygon</td>
<td>Defines a closed shape that consists of a set of connected straight lines</td>
</tr>
<tr>
<td>polyline</td>
<td>Defines a set of connected straight lines</td>
</tr>
<tr>
<td>radialGradient</td>
<td>Defines a radial gradient</td>
</tr>
<tr>
<td>rect</td>
<td>Defines a rectangle</td>
</tr>
<tr>
<td>script</td>
<td>Container for scripts (e.g., ECMAScript)</td>
</tr>
<tr>
<td>set</td>
<td>Sets the value of an attribute for a specified duration</td>
</tr>
<tr>
<td>stop</td>
<td></td>
</tr>
<tr>
<td>style</td>
<td>Allows style sheets to be embedded directly within SVG content</td>
</tr>
<tr>
<td>svg</td>
<td>Defines an SVG document fragment</td>
</tr>
<tr>
<td>switch</td>
<td></td>
</tr>
<tr>
<td>symbol</td>
<td></td>
</tr>
<tr>
<td>text</td>
<td></td>
</tr>
<tr>
<td>textPath</td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>A text-only description for elements in SVG - not displayed as part of the graphics. User agents may display the text as a tooltip</td>
</tr>
<tr>
<td>tref</td>
<td></td>
</tr>
<tr>
<td>tspan</td>
<td></td>
</tr>
<tr>
<td>use</td>
<td></td>
</tr>
<tr>
<td>view</td>
<td></td>
</tr>
<tr>
<td>v kern</td>
<td></td>
</tr>
</tbody>
</table>
Javascript + SVG

http://www.w3schools.com/svg/svg_examples.asp

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">

<rect width="300" height="100"
style="fill:rgb(0,0,255);stroke-width:1;
stroke:rgb(0,0,0)"/>

</svg>
```
Javascript + SVG

http://www.w3schools.com/svg/svg_examples.asp

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
  <path d="M153 334
C153 334 151 334 151 334
C151 339 153 344 156 344
C164 344 171 339 171 334
C171 322 164 314 156 314
C142 314 131 322 131 334
C131 350 142 364 156 364
C175 364 191 350 191 334
C191 311 175 294 156 294
C131 294 111 311 111 334
C111 361 131 384 156 384
C186 384 211 361 211 334
C211 300 186 274 156 274"
style="fill:white;stroke:red;stroke-width:2"/>
</svg>
```
<svg width="100%" height="100%" version="1.1" xmlns="http://www.w3.org/2000/svg">
  <path d="M153 334
  C153 334 151 334 151 334
  C151 334 151 334 151 334
  C164 344 171 339 171 334
  C171 322 164 314 156 314
  C142 314 131 322 131 334
  C131 350 142 364 156 364
  C175 364 191 350 191 334
  C191 311 175 294 156 294
  C131 294 111 311 111 334
  C111 361 131 384 156 384
  C186 384 211 361 211 334
  C211 300 186 274 156 274"
  style="fill:white;stroke:red;stroke-width:2"/>
</svg>
```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">
<defs>
<radialGradient id="grey_blue" cx="50%" cy="50%" r="50%" fx="50%" fy="50%">
<stop offset="0%" style="stop-color:rgb(200,200,200);stop-opacity:0"/>
<stop offset="100%" style="stop-color:rgb(0,0,255);stop-opacity:1"/>
</radialGradient>
</defs>

<ellipse cx="230" cy="200" rx="110" ry="100"
style="fill:url(#grey_blue)"/>
</svg>
```
Javascript + SVG

PITA to write SVG directly. Use, e.g. RaphaelJS

http://raphaeljs.com/

Raphaël—JavaScript Library

What is it?
Raphaël is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Also SVG is not supported in IE until 9. But Raphael renders to IE’s equivalent ..
// Creates canvas 320 x 200 at 10, 50
var paper = Raphael(10, 50, 320, 200);

// Creates circle at x = 50, y = 40, with radius 10
var circle = paper.circle(50, 40, 10);
// Sets the fill attribute of the circle
circle.attr("fill", "+f00");
// Sets the stroke attribute of the circle
circle.attr("stroke", "+f00");
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

Protovis

A GRAPHICAL APPROACH TO VISUALIZATION

Protovis composes custom views of data with simple marks such as bars and dots. Unlike low-level graphics libraries that quickly become tedious for visualization, Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.

Protovis is free and open-source, provided under the BSD License. It uses JavaScript and SVG for web-native visualizations; no plugin required (though you will need a modern web browser)! Although programming experience is helpful, Protovis is mostly declarative and designed to be learned by example.

This project is led by Mike Bostock and Jeff Heer of the Stanford Visualization Group, with significant help from Vadim Ogievetsky. We welcome your contributions and suggestions.
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

**Protovis**

**A GRAPHICAL APPROACH TO VISUALIZATION**

Protovis composes custom views of data with simple marks such as bars and dots. Unlike low-level graphics libraries that quickly become tedious for visualization, Protovis defines marks through dynamic properties that encode data, allowing inheritance, scales and layouts to simplify construction.

Protovis is free and open-source, provided under the BSD License. It uses JavaScript and SVG for web-native visualizations; no plugin required (though you will need a modern web browser)! Although programming experience is helpful, Protovis is mostly declarative and designed to be learned by example.

This project is led by Mike Bostock and Jeff Heer of the Stanford Visualization Group, with significant help from Vadim Ogievetsky. We welcome your contributions and suggestions.
Javascript + SVG: protoviz

http://vis.stanford.edu/protovis/

Gallery

Enjoy these sample visualizations built with Protovis. For any example, use your browser to view the source or the backing dataset. If you’ve built something cool with Protovis, let us know or send a merge request, and we’ll add it to the gallery!

Conventional

While Protovis is designed for custom visualization, it is still easy to create many standard chart types. These simpler examples serve as an introduction to the language, demonstrating key abstractions such as quantitative and ordinal scales, while hinting at more advanced features, including stack layout.

Area Charts

Bar & Column Charts

Scatterplots

Pie & Donut Charts

Line & Step Charts

Stacked Charts

Grouped Charts
Javascript + SVG: protoviz

- Anderson's Flowers
- Becker's Barley
- Bertin's Hotel
- Streamgraphs

- Sparklines
- Bullet Charts
- Bubble Charts
- Sizing the Horizon

- Candlestick Charts
- Burin's Antibiotics
- Nightingale's Rose
- Playfair's Wheat

- Gas & Driving
- Seattle Weather
- Marey's Trains
- Stemplots
Javascript + SVG: protoviz

Maps

Protovis offers two avenues of visualizing geospatial data: build on top of existing browser-based map tools (such as Google Maps or OpenLayers), or use our own geo scales for custom visualization design.

Minard’s Napoleon

Oakland Crimespotting

Choropleth Maps

Symbol Maps

Dorling Cartograms

Map Projections

Heatmaps

Dymaxion Maps
Javascript + Canvas

Canvas element
From Wikipedia, the free encyclopedia

The canvas element is part of HTML5 and allows for dynamic, scriptable rendering of 2D shapes and bitmap images. It is a low level, procedural model that updates a bit map and does not have a built in scene graph.

Bitmap instead of vector

<canvas> is a new HTML element which can be used to draw graphics using scripting (especially JavaScript). It can perform some advanced tasks and also interact with

Rectangles    MoveTo
Paths         Transformations
Lines         Animations (sort of)
Arcs          fast for many objects, but you don’t have DOM control.
Ellipses
<html>
<head>
<script type="application/javascript">
    function draw() {
        var canvas = document.getElementById("canvas");
        if (canvas.getContext) {
            var ctx = canvas.getContext("2d");

            ctx.fillStyle = "rgb(200,0,0)"
            ctx.fillRect(10, 10, 55, 50);

            ctx.fillStyle = "rgba(0, 0, 200, 0.5)"
            ctx.fillRect(30, 30, 55, 50);
        }
    }
</script>
</head>
<body onload="draw();">
    <canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
```html
<html>
<head>
    <script type="application/javascript">
        function draw() {
            var canvas = document.getElementById("canvas");
            if (canvas.getContext) {
                var ctx = canvas.getContext("2d");
                ctx.beginPath();
                ctx.arc(75,75,50,0,Math.PI*2,true); // Outer circle
                ctx.moveTo(110,75);
                ctx.arc(75,75,35,0,Math.PI,false); // Mouth (clockwise)
                ctx.moveTo(65,65);
                ctx.arc(60,65,5,0,Math.PI*2,true); // Left eye
                ctx.moveTo(95,65);
                ctx.arc(90,65,5,0,Math.PI*2,true); // Right eye
                ctx.stroke();
            }
        }
    </script>
</head>
<body onload="draw();">
    <canvas id="canvas" width="150" height="150"></canvas>
</body>
</html>
```
```html
<script type="application/javascript">
  function draw() {
    var canvas = document.getElementById("canvas");
    if (canvas.getContext) {
        var ctx = canvas.getContext("2d");
        for(var i=0;i<4;i++){
            for(var j=0;j<3;j++){
                ctx.beginPath();
                var x = 25+j*50; // x coordinate
                var y = 25+i*50; // y coordinate
                var radius = 20; // Arc radius
                var startAngle = 0; // Starting point
                var endAngle = Math.PI+(Math.PI*j)/2; // End point on arc
                var anticlockwise = i%2==0 ? false : true; // clockwise or anticlockwise
                ctx.arc(x,y,radius,startAngle,endAngle, anticlockwise);

                if (i>1){
                    ctx.fill();
                } else {
                    ctx.stroke();
                }
            }
        }
    }
  }
</script>
```
```javascript
<html>
<head>
<script type="application/javascript">
    function draw() {
        var ctx = document.getElementById('canvas').getContext('2d');
        ctx.clearRect(0, 0, 300, 300);
        for (var i=0; i<3; i++) {
            for (var j=0; j<3; j++) {
                ctx.save();
                ctx.strokeStyle = "#99F0F0";
                ctx.translate(50+i*100, 50+j*100);
                drawSpirograph(ctx, 20*(j+2)/(j+1), -8*(i+3)/(i+1), 10);
                ctx.restore();
            }
        }
    }

    function drawSpirograph(ctx, R, r, O) {
        var x1 = R-O;
        var y1 = 0;
        var i = 1;
        ctx.beginPath();
        ctx.moveTo(x1, y1);
        do {
            if (i>20000) break;
            var x2 = (R+r)*Math.cos((i*Math.PI/72)) - (r+O)*Math.cos(((R+r)/r)*(i*Math.PI/72))
            var y2 = (R+r)*Math.sin((i*Math.PI/72)) - (r+O)*Math.sin(((R+r)/r)*(i*Math.PI/72))
            ctx.lineTo(x2, y2);
            x1 = x2;
            y1 = y2;
            i++;
        } while (x2 != R-0 && y2 != 0);
        ctx.stroke();
    }
</script>
</head>
<body onload="draw();">
<canvas id="canvas" width="350" height="350"></canvas>
</body>
</html>
```
globalCompositeOperation

We can not only draw new shapes behind existing shapes but we can also use it to mask off certain areas, clear sections from the canvas (not limited to rectangles like the `clearRect` method does) and more.

type is a string representing any one of twelve compositing operations. Each of the available types is described below.

Note: In all of the examples below the blue square is drawn first and referred to as 'existing canvas content'. The red circle is drawn second and referred to as 'new shape'.

source-over (default)
This is the default setting and draws new shapes on top of the existing canvas content.

destination-over
New shapes are drawn behind the existing canvas content.

source-in
The new shape is drawn only where both the new shape and the destination canvas overlap. Everything else is made transparent.

destination-in
The existing canvas content is kept where both the new shape and existing canvas content overlap. Everything else is made transparent.

source-out
The new shape is drawn where it doesn't overlap the existing canvas content.

destination-out
The existing content is kept where it doesn't overlap the new shape.

source-atop
The new shape is only drawn where it overlaps the existing

destination-atop
The existing canvas is only kept where it overlaps the new
Javascript + Canvas: processingJS

http://processingjs.org/
Javascript + Canvas: processingJS

Exhibition

Galactic Inbox
by Paul at Monocubed
This is a keyboard-driven game written by the GMall team.
Links: @monocubed, monocubed.com

facebook-privacy
by Matt McKeon
Visualization based on the evolution of privacy on Facebook.
Links: @mattmckeon, thovcl.com

ABSTRACT01js
by Marius Watz
abstract01js is a reworking of an old 2D Processing sketch from 2003.
Links: Marius Watz

JS NINJA
by Alistair MacDonald
This is a mouse-driven game that allows your ninja to build up their chi-tokes.
Links: Hyper-Metrix.com

askken.heroku.com
by Michael Aufreiter at Quasipartikel Labs
Visualization based on search engine.
Links: @_mqi, quasipartikel.at

BitTorrent Visualizations
by Chris Lee
A simplified BitTorrent visualization originally written by Abram Stern, updated by Jeff Awwood. Read Jeff Awwood's blog.

TWITCH
by Casey Reas
Twitch is a series of minimal drawings, animation.

Ball Droppings
by Josh Nimoy
Turn your sound on. Draw lines.

Wiki Visualizations
by Matt Ryall
A small collection of animated.
Javascript + random: google vis API
http://code.google.com/apis/charttools
Welcome to the SlickGrid!

Update: the trunk has been switched to 2.0 alpha. The latest stable release is 1.4.3 (tag).

What it is

Do you use SlickGrid?
Add your site to the Used By!

Quite simply, SlickGrid is a JavaScript grid/spreadsheet component.
It is an advanced component and is going to be a bit more difficult to learn and configure, but once you realize its full potential, it will blow your mind!

https://github.com/mleibman/SlickGrid/wiki/Examples
Javascript: Examples
Javascript: Examples

Everyone
Sleeping, eating, working and watching television take up about two-thirds of the average day.
Javascript: Examples

http://www.babynamewizard.com/voyager
Javascript: Examples

http://oakland.crimespotting.org/
Visualization: Blogs & Examples

http://www.visualcomplexity.com
Visualization: Blogs & Examples

http://infosthetics.com/