TEMPLATES
AND THE
SINGLE-PAGE APP
of the future!

@GARANNM
- HTTP://GARANN.COM
a short history
TIMELINE

- STATIC CONTENT
- “CLASSIC” ASP, PHP, ET AL
- DECOUPLED SERVER-SIDE TEMPLATES
- AJAX & DOM MANIPULATION
- SINGLE-PAGE APPS AND CLIENT-SIDE TEMPLATES
ASP Procedures

In ASP you can call a JavaScript procedure from a VBScript and vice versa.

Procedures

The ASP source code can contain procedures and functions:

Example

```html
<html>
<head>
  <%
  sub vbproc(num1,num2)
  response.write(num1*num2)
  end sub
  %>
</head>
<body>
<p>Result: <%= vbproc(3,4) %></p>
</body>
</html>
```

Show example »
**REMEMBER THIS:**

```html
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%>

<body>
<p>Result: <%=call vbproc(3,4)%></p>

</body>
</html>
```
TEMPLATES CIRCA AJAX

SERVER

PAGE

AJAX

CLIENT
TEMPLATES POST-AJAX

☞ SENDING JSON, NOT HTML
☞ RENDERING VIA DOM MANIPULATION
☞ DECOUPLED SERVER-SIDE
☞ Fallback for non-JS clients
☞ TIED TO REQUEST-RESPONSE
Yeah, but..

- Too much DOM manipulation makes a mess
- It was really slow
- Lots of duplicate code
- Rendering coupled to user interaction
TEMPLATES POST-POST-AJAX
the contenders
VERBOSE LOGIC

☞ CAN USE PURE DATA
☞ MIMICS CLASSIC SERVER-SIDE TEMPLATES
☞ LESS PARSING REQUIRED
☞ INITIAL IMPLEMENTATIONS PRETTY UGLY
☞ MODERN IMPLEMENTATIONS AMONG THE FASTEST
NEEDS PRESENTATION-READY DATA

DECOUPLES PRESENTATION AND CODE

EASIER FOR DESIGNERS?

TEMPLATE IS A DUMB RENDERER

WHICH IS SAFER
<html>
<head>
<%
sub vbproc(num1,num2)
response.write(num1*num2)
end sub
%
</head>
<body>
<p>Result: <%call vbproc(3,4)%></p>
</body>
</html>
WHAT EVERYONE’S SO UPSET ABOUT

<html>
<head>
</head>
<body>
<p>Result: <%= num1*num2 %></p>
</body>
</html>
STRING CONCATENATION

☞ HOW IT’S (MOSTLY) DONE
☞ FAST
☞ FLEXIBLE
☞ OUTPUT NOT REALLY REUSABLE
☞ HAVE TO SEARCH FOR INDIVIDUAL ELEMENTS
DOM ELEMENTS

☞ NOT COMMON

☞ ENGINES USING HTML ATTRIBUTES MAY NOT RETURN A DOM

☞ ALLOWS “DATA VIEW” TYPE CONTROL

☞ REFERENCES TO ELEMENTS AND THEIR RELATIONSHIPS
NON-TEMPLATE-TAG FORMAT

☞ MOST TEMPLATE ENGINES DON’T CARE ABOUT FORMAT
☞ CAN BE USED FOR THINGS BESIDES HTML
☞ SOME RELY ON HTML
☞ SOME ASSUME HAML (OR SIMILAR)
A LITTLE HELP?

Template-Engine-Chooser!

- Is this for use on the client or the server?
- How much logic should it have?
- Does it need to be one of the very fastest?
- Do you need to pre-compile templates?
- Do you need partials?
- Do you want a DOM structure, or just a string?
- Aside from template tags, should it be the same language before and after rendering?

DOM.js  github  project (2.742k)
DO.T.js  project (9.8k)
EJS      project
Handlebars.js  project
Hogan.js  project (2.5k)
ICanHaz.js project (5.448k)
Jade Templates  github (30.687k)
JSRender project (30.708k)
Microtemplating  blog post (1k)
Mustache.js  github (14.513k)
Plates.js  github (10.811k)
Transparency  project (5.491k)
Underscore Templates  project (4k)
TYPICAL SITCH

☞ MUSTACHES {{...}}
☞ SOME LOGIC (CONDITIONS, LOOPS, PARTIALS)
☞ PRE-COMPIILATION FOR REUSE
☞ SERVER- OR CLIENT-SIDE
☞ STRING CONCATENATION FOR SPEED
☞ FORMAT AGNOSTIC
rendering vs. manipulation
<h1>Welcome back, {{username}}!</h1>

<h2>Your friends:</h2>
<p>
{{#friends}}
<a href="/user/{{name}}">{{name}}</a>
{{#if online}}
<span class="online">online</span>
{{/if}}
{{/friends}}
</p>
var userObj = {
    username: "tmplM4st3r",
    friends: [
        { name: "1337tmpls", online: true },
        { name: "hbars4lyfe", online: true },
        { name: "belieber42", online: false }
    ]
};
<h1>Welcome back, tmplM4st3r!</h1>

<h2>Your friends:</h2>
<p>
<a href="/user/1337tmpls">1337tmpls</a> <span class="online">online</span>
<a href="/user/hbars4lyfe">hbars4lyfe</a> <span class="online">online</span>
<a href="/user/belieber42">belieber42</a>
</p>
Most template engines will accept any string

Script tag with non-rendered type

External file loaded via AJAX or a loader

String concatenated into JS during build

More fragile
```
var $container, myTmpl, userObj;

$.get("templates/user.tmpl", function(tmpl) {
    myTmpl = Handlebars.compile(tmpl);
    $container.html(myTmpl(userObj));
}, "text");
```
function renderUser( cb ) {
  if ( myTmpl ) {
    cb();
    return;
  }
  $.get( "templates/user.tmpl", function( tmpl ) {
    myTmpl = Handlebars.compile( tmpl );
    cb();
  }, "text" );
}
renderUser( function() {
  $container.html( myTmpl( userObj ) );
});
socket.on( "friendOffline", function( friend ) {

    var friends = userObj.friends;
    $.each( friends, function( i, f ) {
        if ( friend.name === f.name ) {
            f.online = friend.online;
        }
    });

    renderUser( function() {
        $container.html( myTmpl( userObj ) );
    });
});
socket.on("friendOffline", function(friend) {
  $("a[data-name=\"" + friend.name + \\"\"]\")
    .next("span")
    .remove();
});
<h1>Welcome back, {{username}}!</h1>
<h2>Your friends:</h2>
<p>
{{#friends}}
  <div data-name="{{name}}">
    {{> friend }}
  </div>
{{/friends}}
</p>
DEFINING A PARTIAL

Handlebars.registerPartial( 'friend',
  '<a href="/user/{{name}}">{{name}}</a>' +
  '{{#if online}}' +
  '<span class="online">online</span>' +
  '{{/if}}'
 );

var friendTmpl = "{{> friend }}";
AND SO!

socket.on( "friendOffline", function( friend ) {

  $( "div[data-name=\"" + friend.name + "]" )
    .html( friendTmpl( friend ) );

});
COMPOSITION CHOICES

☞ HOW MUCH DOM MANIPULATION IS NEEDED?

☞ HOW LIKELY IS RE-RENDERING?

☞ HOW DIFFICULT IS IT TO FIND THE CHILD ELEMENT?
client-side architectures
MVC

- View and template often synonymous
- In practice, need a view-model
- Controller determines when to render
- Need non-MVC concepts
- Rendering container
- Event handlers
A “VIEW”

☞ THE TEMPLATE

☞ ITS CONTAINER/RENDERING TARGET

☞ VIEW-MODEL/TRANSFORMATION LOGIC

☞ EVENT HANDLING?

☞ ACTUALLY A BUNCH OF STUFF
ABSTRACTED RENDERING

☞ A COMPLETE VIEW SHOULD ONLY NEED TO BE TOLD WHEN TO RENDER

☞ EVERYTHING MAY NOT BE A COMPLETE VIEW

☞ E.G. PARTIALS

☞ EVERYTHING MAY NOT MAP PERFECTLY TO A MODEL
TEMPLATES FILLING IN GAPS

☞ NON-APPLICATION PARTS OF THE PAGE
☞ PIECES OF PROPER MODELS
☞ NON-DATA INPUT STRUCTURES (E.G. CONFIRMATION)
☞ SUB-VIEWS WITHIN PROPER VIEWS
TEMPLATES WITHOUT MVC

☞ Map to states, not data
☞ Generic renderer or tight coupling
☞ Triggered by event, Object.observe()
☞ May need more partials
☞ More data potentially hard-coded
function Renderer() {

    this.render = function( data ) {
        data.container.html(
            Handlebars.compile( data.tmpl )( data.obj )
        );
    }

    return this;

});

Renderer.subscribe( 'formInit', this.render );
Renderer.subscribe( 'formInvalid', this.render );
Renderer.subscribe( 'formSubmitted', this.render );
WITH OBSERVERS

UserData.prototype = {
    get value() {
        return this._value;
    },
    set value( val ) {
        this._value = val;
        this._container.html(
            Handlebars.compile( this._tmpl )( val ) );
    }
};
IN ANY ARCHITECTURE

☞ DECOUPLE MARKUP FROM CODE

☞ REDUCE NEED FOR DOM MANIPULATION

☞ MOVE RENDERING TO AN ABSTRACTION
node.js
not dissimilar from client

Find packages...  or browse packages.

Template
bn-template  JavaScript templating for browsers and node.js
app-template  template application
Temp18  JavaScript Client/Server Template Engine
aejs  Asynchronous Embedded JavaScript Templates
ajs-xgettext  Extract localised text from AJS templates
atpl  A simple template engine based on twig and django.
blue  JSP-like and streamed template engine
bogart-handlebars  Handlebars template engine plugin for Bogart
ajs  Experimental asynchronous templating in Node
amulet  As-soon-as-possible streaming async Mustache templating
bake  static file bakery
batik  Super Simple CoffeeScript Templating
bind  A simple templating engine that smiles back
ClearSilver  ClearSilver template engine bindings for node.js
TwigJS  A port of PHP template engine (www.twig-project.org) to Javascript
bluemold  Template engine based on jQuery template syntax
beard  More than a mustache.
SERVER-SIDE USES

- INITIAL LOAD
- FULL-PAGE RENDERING
- SERVER-SIDE COMPILATION (HOGAN.JS)
- RENDERED HTML SNIPPETS
DUMB VIEWS

☞ SERVER-SIDE MVC IS DIFFERENT
☞ MORE MODELS
☞ MORE controllers
☞ LESS VIEWS
☞ VIEW IS SINGLE-USE
☞ USER INTERACTION NOT RELEVANT
PRESENTATION LOGIC

☞ STILL NEEDED FOR RENDERING

☞ DOES THIS BELONG ON THE SERVER?

☞ IS IT NECESSARY?

☞ CAN IT BE SHARED?

☞ ISOMORPHIC VIEW-MODELS AND VALIDATION
TYPE OF TEMPLATE MATTERS

☞ MAY BE BETTER FOR HAML ET AL

☞ SERVER-SIDE DOM POINTLESS

☞ EXCEPT FOR SCRAPING/CRAWLING

☞ PERFORMANCE MATTERS LESS

☞ BUT ARE YOU ONLY USING TEMPLATES ON THE SERVER?
full-stack templates
THE GOOD STUFF

☞ USE THE TEMPLATE FOR INITIAL LOAD

☞ REUSE IT TO RENDER NEW DATA

☞ SAME TEMPLATE FOR:

☞ SERVER-SIDE CONTROLLER (URL)

☞ CLIENT-SIDE CONTROLLER (LOCATION.HASH)
SHARED ACCESS

☞ EASIEST TO USE SAME LOADER
☞ E.G. REQUIRE WITH TEXT PLUGIN
☞ NO NEED TO CREATE TWO VERSIONS
MANAGING PARTIALS

☞ CAN BE DIFFICULT DEPENDING ON TEMPLATE ENGINE
☞ ARGUES FOR LARGER TEMPLATES
☞ NAMESPACES WORK DIFFERENTLY
☞ SCOPE UNRELIABLE
SOLUTIONS APPEARING
FULL-STACK FRAMEWORKS

☞ REUSE THE FRAMEWORK, REUSE THE TEMPLATES
☞ NOT THERE YET
☞ BUT PEOPLE ARE WORKING ON IT
☞ EXPRESS, GEDDY USE TEMPLATES AVAILABLE ON CLIENT
☞ METEOR, DERBY USE THEIR OWN
WHERE THAT LEAVES US

- Client-side templates: check
- Complex client-side apps: check
- Reuse on the server: check
- One unified do-it-all solution: .. to be continued!
thanks!

@GARANMM - HTTP://GARANN.COM