Application Monitoring with Cross-Platform SignalR

Steve Kollmansberger

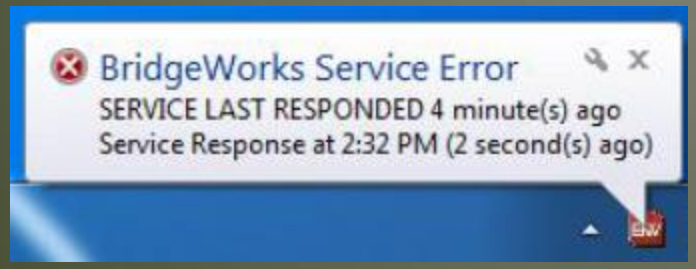
WSDOT Developer Conference

June 9, 2015

PROBLEM?

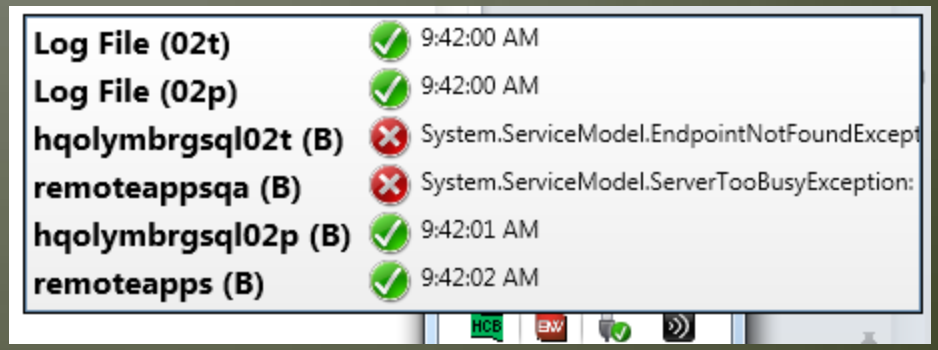
Feb 2012: BW Service Monitor

- *"Let's never be surprised to find out that the service is down again."*

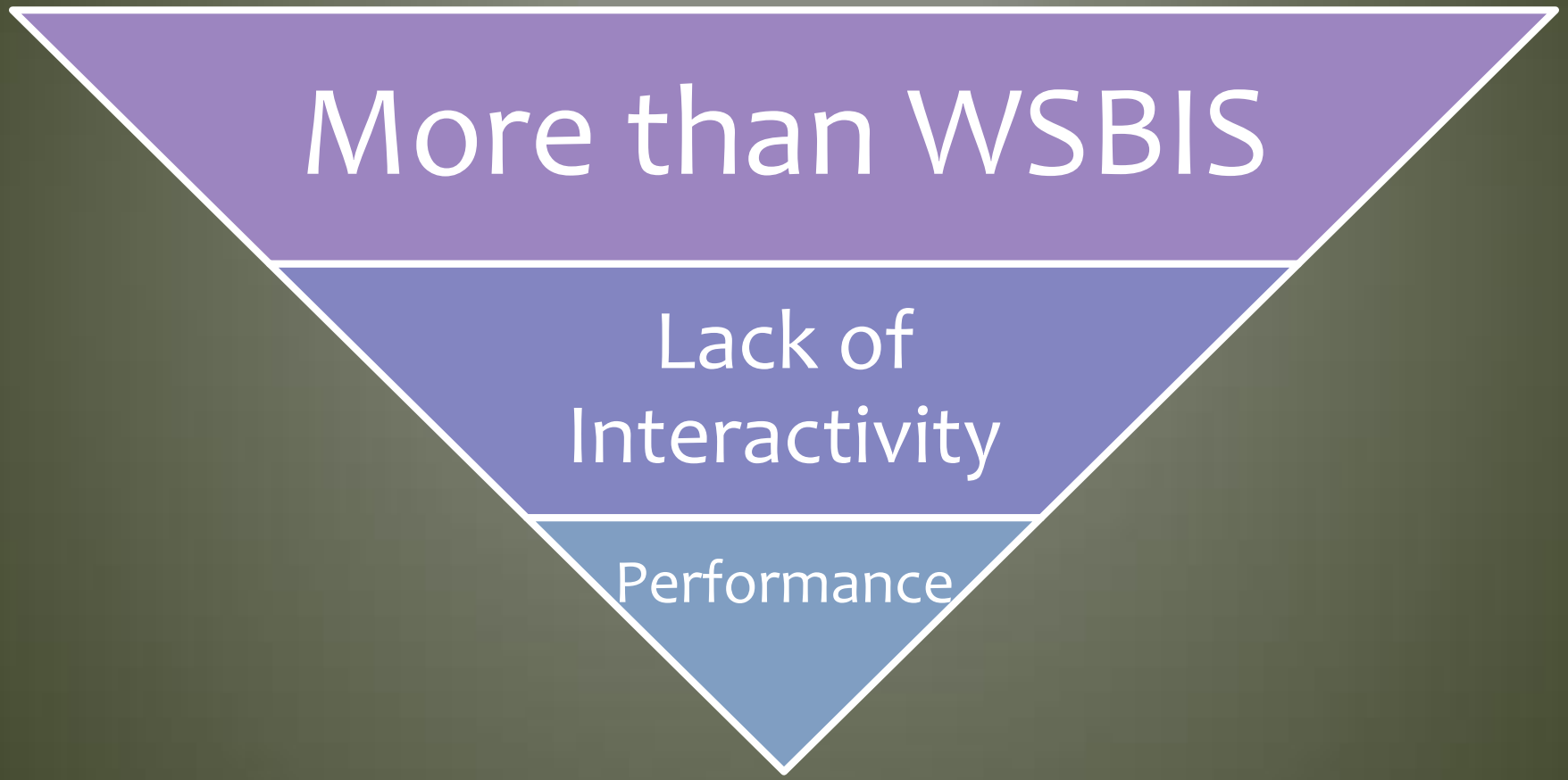


Feb 2015: WSBIS Service Monitor

- *The old monitoring system wasn't compatible with the new application*
- *Wanted to monitor more ways*



ISSUES REMAIN



MONITORING: THE NEW HOTNESS

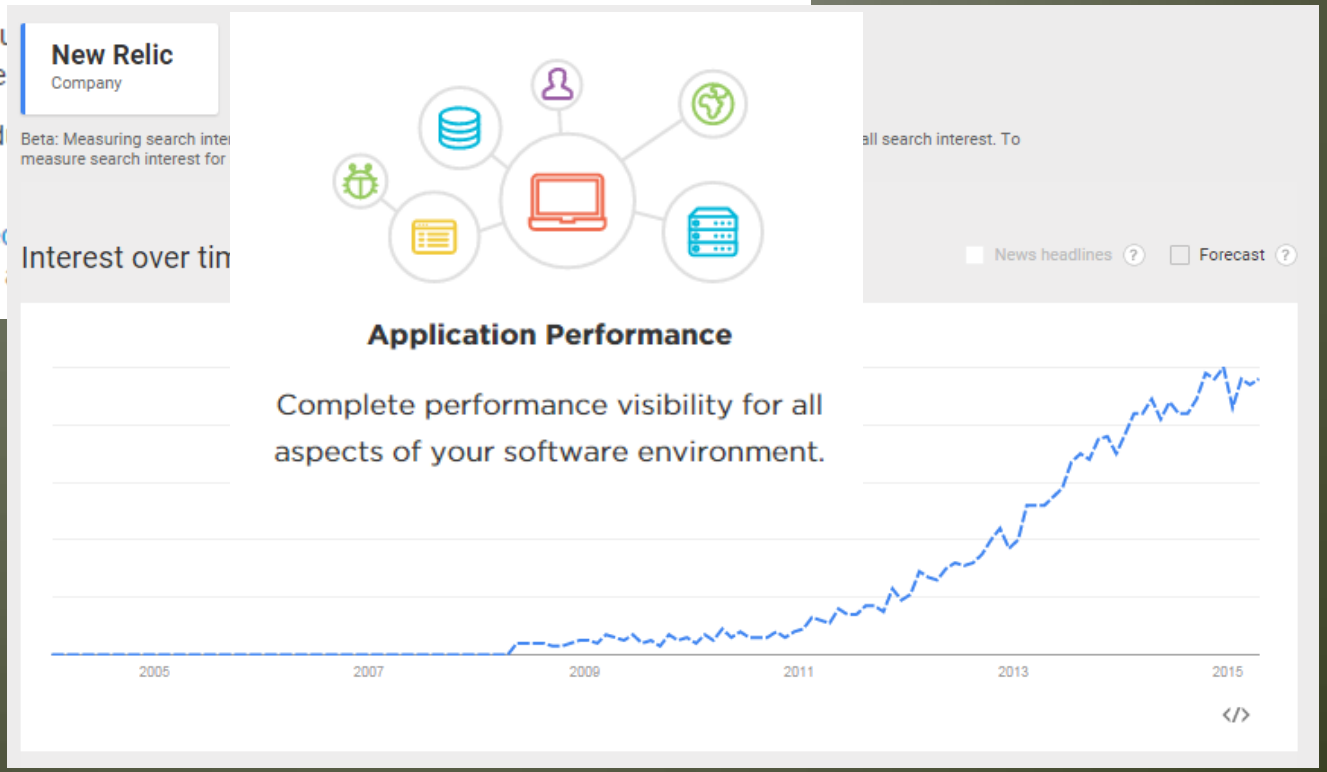
Test-First Programming becomes Monitoring-First Programming

Or TDD becomes MDD. When we collectively own the whole product lifecycle we can write our automated production monitoring checks first, and see them fail before we start implementing our features.

This forces us to make sure that featu
just like TDD, while ensuring we have

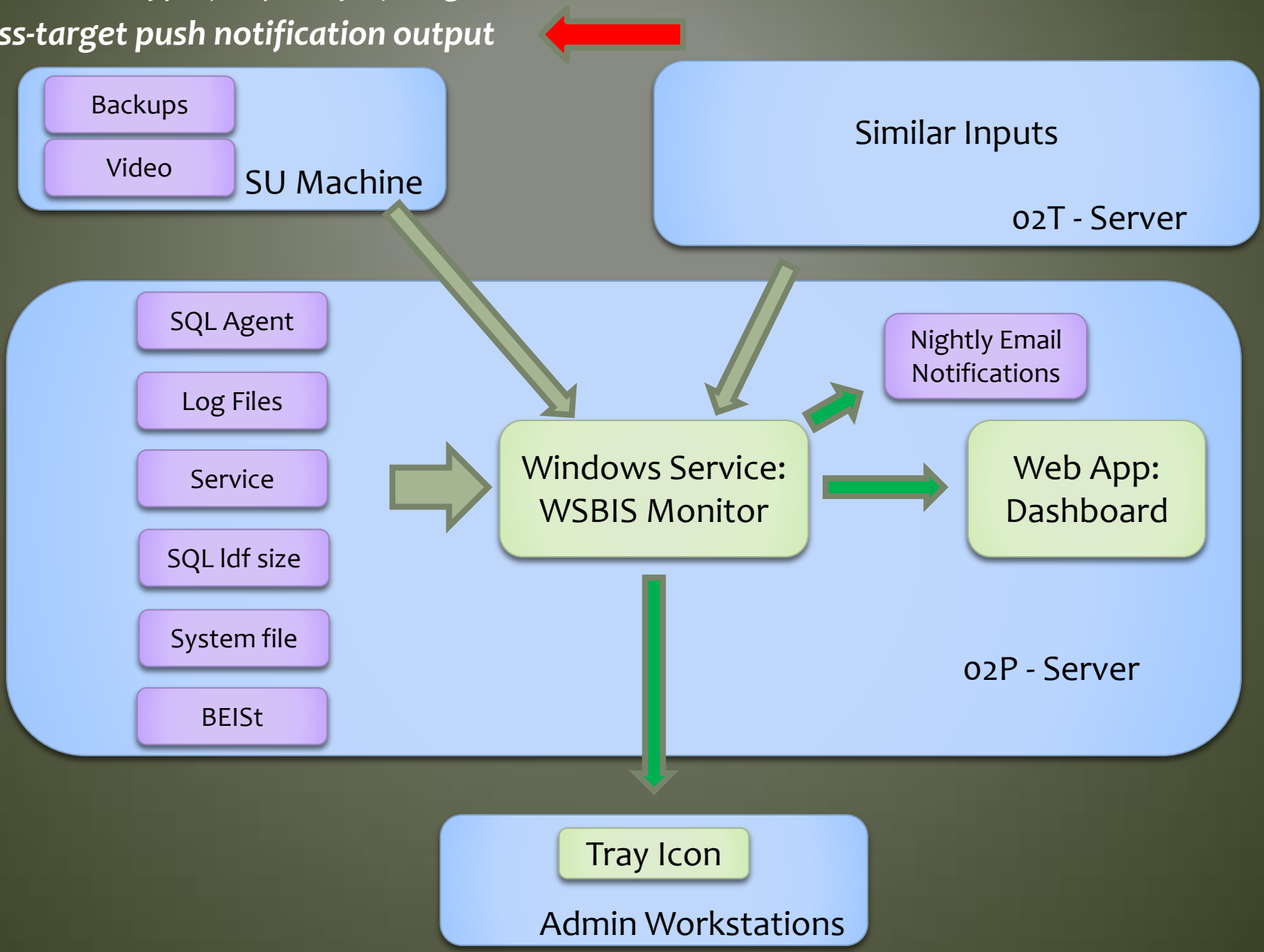
It also helps us have trust in our prod
our production system.s

Just like with TDD, MDD gives us feed
improve our designs. It also gives us

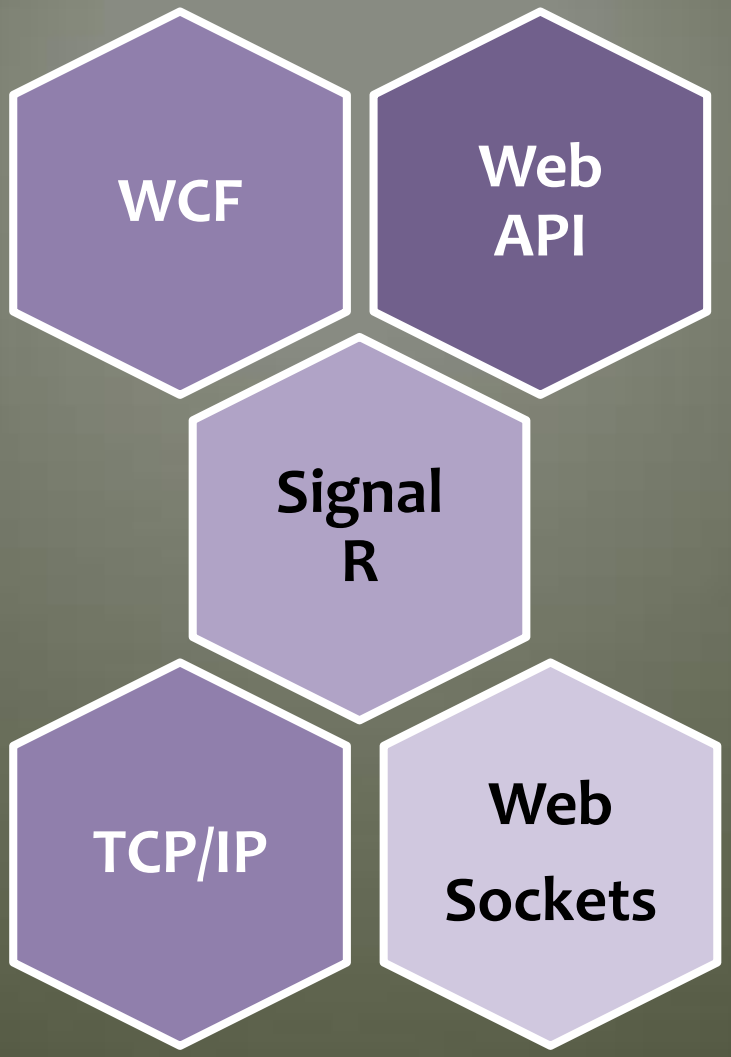


Central Monitoring Service

- *Any number/type/frequency of targets*
- ***Cross-target push notification output***



HOW TO PROVIDE PUSH NOTIFICATIONS TO CLIENTS?



SERVER IMPLEMENTATION

Subtype Hub

```
public class MonitorHub : Hub<SignalRInterfaces.IMonitorServer>, SignalRInterfaces.IMonitorClient
{
    // any client can call these methods below (from IMonitorClient)
    public void MuteCheckable(Guid id, int? minutes)
    {
        DateTime? until = null;
        if (minutes.HasValue)
            until = DateTime.Now.AddMinutes(minutes.Value);
        string name = Context.User.Identity.Name;
        Host.Mute(id, name, until);
    }

    public IEnumerable<SignalRInterfaces.CheckStatus> GetAllCheckables()
    {
        return Host.GetAllCheckables();
    }

}
```

Free* Authentication

Free Serialization

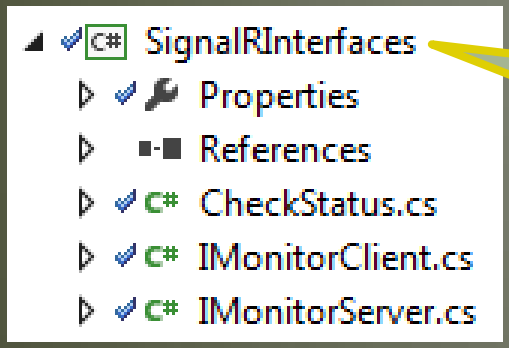
* Authentication must be configured and is hard to find in the documentation ☹

SERVER IMPLEMENTATION

```
// These are messages that the server can issue to the clients
public interface IMonitorServer
{
    void UpdateCheck(SignalRInterfaces.CheckStatus status);

}
```

Interface for server -> client calls



Project for DTOs and Client/Server RPC Interfaces

SERVER SELF HOSTING

Port to listen

```
string url = "http://localhost:8811";
SignalR = WebApp.Start(url);

// Undocumented (lovely)
// GetHubContext<T>(T2) where T can be an interface with server-called methods,
// and T2 is the string name of the actual concrete type with methods clients can call
Context = GlobalHost.ConnectionManager.GetHubContext<SignalRInterfaces.IMonitorServer>("MonitorHub");
```

Self Host Start

Defaults to dynamically typed, but static typing possible. *

```
Context.Clients.All.UpdateCheck(status);
```

Can send message to all or selected clients. Don't need to track clients manually.

* SignalR was originally "untyped" and has been adding type-safe capabilities still below par and poorly documented ☹

CLIENT (JS)

Setup and Connect

```
// Set the hubs URL for the connection
// The URL comes from App/config.js
// If you are using a SignalR hub inside your ASP.NET web application
// you don't need to provide the URL!  In this case, the SignalR hub is
// hosted outside the web app so we must provide the URL
$.connection.hub.url = config.hub;

// Declare a proxy to reference the hub.
// Note that the word "monitorHub" here
// means the class "MonitorHub" in the SignalR host C#
var monitor = $.connection.monitorHub;
```

```
// Start the connection. This establishes a "two-way"
// link between the client and the server
$.connection.hub.start().done(function () {
```

```
// The SignalR server can "push" notifications to us with various methods
// These methods are dynamic and match "by name" on the server C# and here on the client side
// This will be called by SignalR whenever the server provides (pushes) a new status notification
monitor.client.updateCheck = function (status) {
    // Note that push notifications can send arbitrary serializ
    // The serialization happens "for free"
    // In this case, the C# call Context.Clients.All.UpdateCheck(status);
    // passes a status of type CheckStatus (C#), which is a DTO
    // It is serialized by SignalR into JSON (since this client is JavaScript)
    // and the "status" we get here is a JSON object with the same fields as the C# class.
```

Handle calls from server

CLIENT (JS)

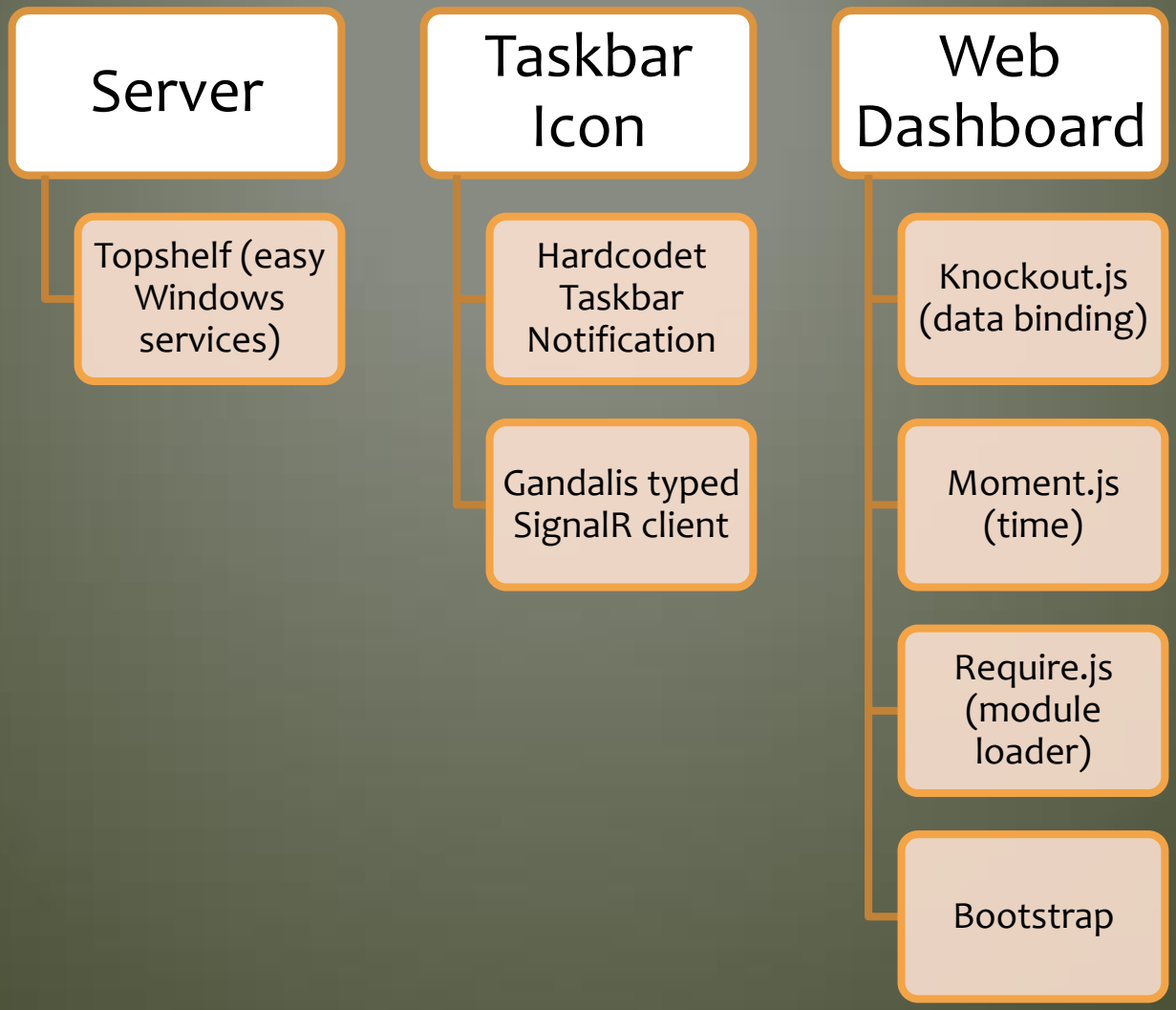
```
// Make an initial request for the current status of all systems
// We can call methods declared on the server
// This method is spec'd in IMonitorClient.cs and implemented in MonitorHub.cs
// All calls are async, so expect a JS Promise to handle the result (if any)
monitor.server.getAllCheckables() // NOTE: SignalR automatically re-cases first letter between C# methods and JS methods
    .done(function (data) {
        // As above, serializatio    f parameters and return value is automatic
        // This return is of C# type    merable<CheckStatus>
```

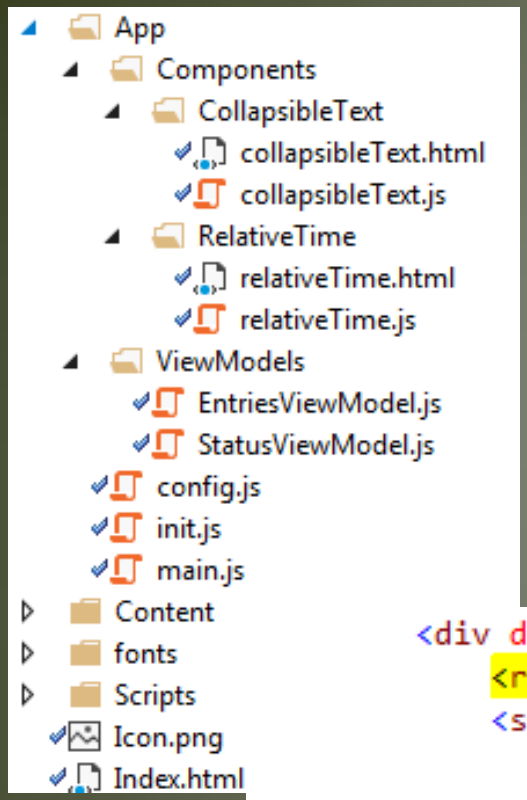
Call methods on server

```
// SignalR will automatically attempt to restore a physical connection using
// whatever transport is appropriate.  If the connection cannot be restored,
// a disconnected event will be raised.
// By default, a 30-second timeout exists (during which SignalR will attempt to restore
// connection) before the event is raised
$.connection.hub.disconnected(function () {
    viewModel.connectionWarning("Connection lost");
    viewModel.entries([]);
});
```

Handle Disconnections

LIBRARIES


Server
Topshelf (easy Windows services)
Taskbar Icon
Hardcodet Taskbar Notification
Gandalis typed SignalR client
Web Dashboard
Knockout.js (data binding)
Moment.js (time)
Require.js (module loader)
Bootstrap



Similar to…

Custom elements are a W3C Working Draft

```
<div data-bind="if: Muted">
    <relative-time params="{value: Muted}"></relative-time> by <span data-bind
    <span data-bind="if: MutedUntil">
        until
        <relative-time params="{value: MutedUntil}"></relative-time>
    </span>
</div>

    </td>
</tr>
<tr class="detail" data-bind="if: Message">
    <td></td>
    <td colspan="6">
        <collapsible-text params="{length: 75, content: Message}"></collapsible-text>
    </td>
</tr>
```

QUESTIONS?

It sees you when you're sleeping
It knows when you're awake
It knows if you've been down or up
So be up for "upness" sake

You better watch out
You better not crash
Better not error
I'm telling you why

Monitoring system is coming to town