**Introduction**

*This is the introduction and uses the first 4 slides. There isn’t any scripted content for this section, as you just expand on the slides.*

[Slides 1,2,3,4 -> Intro material ]

**Comparison of SPA Vs. Traditional Web Applications**

*This is the framework introduction and uses slides 5 through 17. You should be familiar with the text below which is the information to talk about over these slides. Slide position is mentioned where possible.*

[Slides 5,6,7 -> Variance in composing pages / page layout ]

There have been various methods over the years for composing pages. There are very traditional includes, such as in PHP, ASP, etc.

In the J2EE world there has been JSTL for includes, and libraries like Tiles for composing pages.

Even before all of this, designers and developers were using frames to try and avoid rebuilding the entire page.

With an SPA, even when a template engine isn’t used, the entire page is rendered once, and from there only the portions of the page concerned with what is changing are updated. Users spend an increasing amount of time interacting with an application before it has to go back to the server. The blink of the page render is gone and replaced by progress bars or animations.

Responsibility also changes. The server can now focus on areas like services and security. Server side developers can focus on data storage and service interfaces without knowing anything about the look, feel or even design of the front end. Services can be designed to be a mix of public and private APIs.

On the front end, the responsibility for page composition is taken on, as well as routing logic and some business logic. It should be noted here that what doesn’t change, is the need to validate all data arriving at the server. Do not calculate the shopping cart total on the front end and then trust the value ☺.

For developers, there is much more clarity of role. Work can be more easily divided into roles; with developers work on the server, look & feel, and front end application logic.

**Framework Tour and Comparison**

*Compare the frameworks based on their usage and features, then give a deeper overview of each framework.*

[Slides 8,9,10,11 -> Basics framework comparison]

There are a growing number of frameworks for developing client based applications, whether “SPA” style or not. We’re going to look at a couple of them and talk about their popularity, differences and finish with some warnings about relying too heavily on these frameworks.

All of these frameworks work to provide a structured MVC/MV\* pattern for building applications. Each usually provides more or less features with the trade being control of implementation and architecture or ease of use in the basic case.

Market Share **@TODO**

Trends and other Info **@TODO**

**Angular.JS** [Slide 12]

Angular is a library currently supported by Google. It’s widely popular and based on adding additional markup to your html. By placing markup you give Angular instructions and information about your page.

Angular is one of the larger frameworks available, because of all the features it provides. It’s Very declarative and inspired by ideas like Silverlight, which is exemplified by the two way data binding and html markup control. Angular has no dependencies, but wants to stand on its own; so it doesn’t work well with other technologies like JQuery.

= Still more to add here =

**Backbone.JS** [Slide 13]

@TODO

**Ember.JS** [Slide 14]

@TODO

**Knockout.JS** [Slide 15]

@TODO

**Data Binding and Conclusion** [Slide 16, 17]

@TODO

Session 1.

Closing (30 minutes)

We’ll take 30 minutes to try and answer form frequently asked questions about SPAs. Hopefully this brief talk will prime attendees for asking questions and talking about the exercises and how they solved them.

FAQ entries:

1. Different front ends for mobile vs. desktop.
2. Deploying SPA’s as part of a class deployment process.
3. Development environment.

Depending on questions and talking by attendees, there will be a summary. We’ll show a list of additional resources on the board and probably as part of the *handout?*

**Narrative:**

Your boss wants to quickly deploy a restaurant reservation system to take advantage of a new public rest api created by the chamber of commerce. It allows anyone to get information about restaurants and make reservations. The chamber of commerce plans to expand this service in the future, so he wants to get in on the ground floor.

Since your boss wants this application quickly, so he has already had the web designer draw up a mockup for you to use in building the application.

Since you’ve been hearing about building SPAs and learning about Backbone; you’ve decided to use Backbone to build the application.

Session 1:

Explain that the fetch is asynchronous.

Handlebars: bring up the value of client side templating. And not having to dom manipulate via jquery

Session 2:

Session 3:

When creating the router, you will also have to modify the handlebars template to use your route. This change isn’t shown, but we can help you with it if you are stuck.

Session 4:

Session 5:

Session 6:

Session 7:

Talk about testing views.