CascadiaJS

Merrick Christensen

**require.js**

Project bringing an object oriented model to js, script loading and dependency management. You can also load other files besides js.

Inheritance, private and public, require.

A module system from this is going into the next version of js.

**Notes from his actual speech:**

Javascript Modules:

-Private variables

AMD & require.js

-Asyncronous Module Definitions

-Plugin System using AMD

Functions:

require()

define()

-Let the build name your modules.

-You can greatly customize your module loader.

Require errbacks

-Error checking callbacks

-Graceful Module Shutdown: If JS throws an error, it can be configured to restart the app, or even just specific modules, without page refresh.

Require Plugins

-Not just for plugin, you can include things like json files and it will be pre-parsed for you. You can even require things like coffee script without having to compile it to JS! This can be called “Smart Resource Loading.” Require intelligently knows what you are loading and loads the resource correctly for you.

Angelina Fabbro

**shadow dom – inspector web – web components**

How do we currently build websites?

<script stuff></script>

<link styles />

<div class=”content”>Some Content</div>

We very rarely have one widget and one style sheet, and, we usually get very specific with our css selectors. It can get complicated, and, we can have namespace collisions. We can do better!

What we need is encapsulation. Up until now the iframe was the only way to do it. The shadow dom/inspector web accomplishes this.

We see shadow dom already on things like video elements, we had the ability to edit things like navigation selectors, but, we just didn’t know it because it uses the shadow dom.

Shadow dom is a small document that lives inside of its host element. The parent element essentially doesn’t know anything about the child shadow element.

Spooky = pageElementId()

Shadow = new WebkitShadowRoot(‘Spooky’);

In the shadow root, if you create a content tag, <content></content>, your browser will take the parent element and place it in the content section.

HTML Templating and custom dom elements. Client Side templating built into the browser.

<element name=”news” extends=”ul”>

<template>

<h2>Breaking News</h2>

<ul>

<content select=”breaking”></content>

</ul>

<ul>

<content></content>

</ul>

</template>

</element>

<ul is=”news”>

<li></li>

<li class=”breaking”></li>

<li></li>

<li class=”breaking”></li>

<li class=”breaking”></li>

<li></li>

<li></li>

</ul>

UL will be reordered based on your html template. The key here is the markup is untouched. Styles will not cross the boundry of the shadow dom element. There is a way to do it, but, there is still some work that needs to be done on this front.

This project seems awesome and seems to be getting some real support in browsers, but, it is still in development. It seems that require.js does a lot of the same things but much more. Either way, it is a move in the right direction for js.

Kit Camridge

**browser.js**

-You can either use 3rd party frameworks or make your own. Whatever the case, you will always have the following.

-Spoke on base elements or a project base

-Spoke of differences between elements and attributes. (Spoke way too fast and was too ambiguous to understand. At this point I have no idea where he is going with this…It seems he wants to make us aware that every browser implements every element differently, and that every element has bugs.)

General-purpose solutions like jquery are better to use than specialized frameworks. The more general you can get and the least amount of frameworks you can use, the better you app will run and the better supported your app will be.

Stop writing library plug-ins! He suggest writing them lower level as it will increase your processing speeds.

Suggests you accept constraints.

I disagree with this developers view on app development. While I agree on increasing processing speed, skipping jquery plugins is not the way to do it.

After rolling through Kit’s presentation again, I think I got a better understanding of why he has the views he has. Kits focus in developing an application is on complete cross browser compatibility, including working around every bug in the specific browsers html implementation that can be found. So, Kit says that if you want to use a general purpose solution like jquery, you need to accept that there may be inherent bugs that jquery didn’t work around. Given this, Kit says that if you want to be sure every potential bug between all browsers is weeded out, “roll your own solution.”

I completely understand the mindset here as I have similar ideas in my game engine development, but, I think we are running into a “don’t re-invent the wheel” scenario.

Itay Neeman

**bigdata.js splunk**

Data engine for “evented” data

“Big Data” What makes up big data?

-You need to get the data into your data stream

-Connect to a stream or something?

-Ask your data a question?

-Show me the last posts in the last 24 hours.

-Answer from data.

This is just a standard model, but, Splunk really focuses on live data. Real time is real time, data is pushed almost instantly.

Here is the idea this speaker wants to get across:

You attach to a DB system which accepts data computation. That is, the idea is you pass the data computation to the server rather than asking for the data and calculating your modifications locally. You pass the server json, and the server takes that json and computes exactly what you are asking for. He suggests using node.js as your server. Using this model, you wont even have to reload your page to get live data. Neeman suggests using SocketIO. He also suggests making a framework where instead of polling the server, you set up a stream where the server just pushes to the user based on what the user asked for.

Neat idea. It seems like this system is better for feed based apps like twitter which deal with large amounts of data or apps where you need live data down to the second.

Russel Branca

**D3.js**

“Visualizing Volcanos”

Interactive Visualizations: Human pattern recognition.

Core Components:

-Ability to ask new questions and quickly get answers.

-Ability to filter data

-Ability to distribute and manipulate data.

-Need Sexy data

Help your users figure out what data they want to ask and give them the ability to quickly get an answer.

Branca advocates for D3 saying that it is a great framework for setting up dynamic data systems. D3 really helps you “discover” data.

It sounds like D3 is a great library for apps that need to really analyze data. It doesn’t sound like a solution for apps like we create. The example he used in his speech was an app that visualized volcanic activity/earthquakes.

<http://d3js.org/>

CouchDB: Distribute your applications with your data. It makes the app static, but, it’s a cool way to distribute demo’s without having to worry about database connections.

Daniel Worthington

**Excel in the Browser**

Report generating. Can view data in useful ways.

Can download and view in excel.

**Why JS?**

It’s faster to download the JS library than to do it on the server. Cross platform support.

3 main visual tools: Charts, Shapes, Tables

**Main Point to take from this speech:**

Simplify your data. Json and XML are great.

Tom Rudick

**Windows Azure**

-Easy service creation using node.js

-Easy javascript app creation

Windows Azure has a sleek web interface where you can create services without touching the back end at all. If you don’t mind touching the back end, you can write scripts that run on the Azure node.js server.

Windows Azure is an app and service creation framework which allows for easy management and creation of apps.

It has support for push notifications, text notifications, and easy integration with things like openID and social networking API’s. Azure has objects which handle this integration for you.

Rudick only mentioned Windows phone devices, but, I would bargain he is paid to do so. Since this is JS and node.js it obviously will work on other devices.

As far as use in professional development, I cannot speak for the usefulness of Azure as not enough was spoken on it other that a few small things you can do with it. I will say that it looks like a highly specialized framework and probably has quirks in development that come with that. Even so, it looks like Microsoft maintains all of the back end integration with all services such as openID, so the upkeep of that is on them, which will speed up development and lower maintenance cost on companies using Azure.

He also did a cool demo where anybody could text his number and any text on that number updated a feed on his app. The app totaled 18 lines of code which is truly impressive given all of the frameworks it would take to run such an application.

Anton Molleda

**HTML5**

Basically, from what I can gather, Anton is showing us how you can achieve 3d interfaces using HTML5 and his 3D Engine. I am not sure how efficient the engine is, but, it is impressive. Their engine is featured on the site <http://www.projectprometheus.com/> and it does look really good.

Molleda suggests not writing your own 3D library and instead participating in webGL as “it’s not worth it.” I tend to agree as writing your own 3d library is truly akin to re-inventing the wheel.

Domenic Denicola

**Real World Windows 8 apps**

Developer Experience: Denicola says that Windows 8 is a 1.0 version, and what that means is that it will be about 2 more versions before it is a truly polished experience. Not a great way to market windows 8 to a room full of devs, but, Windows 8 native JS app development is very interesting to me. What was very interesting about his speech is that you can actually compile your JS project to run on mac as well as windows 8. I don’t know what the specifications on that are, but, it was mentioned.

-When deving JS apps in windows 8, use the pre-built controls(ui-elements) as these controls react to OS specific things like color scheme an opacity of windows, among other things.

Denicola says that you need the following to start making a WinJS app:

-A module System

-A package system

-A way to stop writing raw html/css

-A build process

WinJS currently has no support for unit tests/headless testing.

Matthew Bergman

**Tyranny of Choice**

Tyranny of Choice: How many choices a day do you make when programming?

He say’s 1 real choice.

When we deal with things we don’t really know, we usually go to modules. Unfortunately, when trying to decide what libraries, modules, or dependencies you are going to use for your project, there are TOO MANY CHOICES!

Bergman suggests making your modules smaller. Instead of having large frameworks, you should split our functionality into a smaller, easier to integrate, module.

There is a reason people use rails. They use rails because it decreases the choices you have to make, and in business every choice has a cost associated with it. Because of this, rails overpowers other fantastic frameworks like Sinatra.

Bergman insists that we need to create great learning and search resources for JS to get new devs up to speed faster, as well as weed out garbage from greatness in JS modules. At this point, we don’t have and resources like this. With this, we also need a better way to judge the health of modules. Currently on github there are just star ratings, he says this is not a good way to judge the health of modules.

Brock Whitten

**Writing Scalable Systems**

Controlling complexity is the essence of computer programming. This is something we often forget. As software engineers our job is not just to “get stuff to work,” we need to control our systems complexity as well.

Minimum Complexity is the key to writing scalable systems.

What is complexity?

-What is the mental overhead for a developer to look at an existing code base?

-What does it take for a user to use a project?

-Lines or code. The more you condense the easier your code will be to understand.

200-400 lines is a sweet spot. This is only a general guideline.

A scalable system is something where you can replace a part without breaking the whole. Agility is the key!

**Harp platform**: Developing within dropbox. Whitten feels we overcomplicate our projects. (Something I tend to agree with to a degree). He is making an architecture which he feels will simplify our dev tools.

**Axon**: Push-Pull message system. “Message Broker”

Jonathan Turner

**TypeScript**

A superset of JS made by Microsoft. The idea is to take JS and add things like classes, modules, and types. TypeScript can integrate completely with existing JS.

Jason Denizak

**TypeScript: The Good Parts**

-How can typescript help developers make better application compilation?

-Denizak wants us to “work on shit that matters.” He wants us to work on projects we care about.

-Typescript can help with code reuse.

-In using typescript, you do not need to change the way you write code.

-“JS is the bytecode of the web.”

-Denizak feels that having many different languages that compile to JS is a good thing because it shows increased innovation.

-Denizak says that in the JS community we need to focus on writing fantastic interfaces with our systems. Interfaces will make programming in our systems easier and truly increase your API’s us within the community.

Fun quote: “Doesn’t polymorphism suck?” –Jason Denizak

---

**Note:**

I think what can be taken from Denizak’s speech, along with Jonathan turners and many others here, is that there are many layers above JS being created, all pushing that their way is the best way. I personally think over the next few years that we will see these frameworks boil down to their best few, and that will become the “go do” frameworks for web development.

---

**Note:**

After 2 days of this conference, I am starting to see 2 viewpoints for the future of javascript. On one team, we have those who believe simplified API’s will be the future of JS, where you work with libraries that do massively complex things with a couple of calls. It seems like this group wants to move above the OOP model and kind of abstract it out of development as much as possible.

Then you have the group that wants to make JS a “big boy language.” Which this group is also for creating API’s to work with, they also believe in creating an extremely powerful language that operates much like java or C++. My views personally sit with this group as a solid foundation is crucial to not only creating a language which is easy to pick up and create complex OO systems in, but also creating an environment which can support the use of many different amazingly complex API’s in a single application.

In the current JS model, supporting many different API’s in a single application can be a challenge. I feel through the combinations of projects like require.js and (shadow dom, inspector web, web components) we will see an OO programming environment that can support the views that have been presented at this conference. However, before we get to that point, I think we will see many of the projects here get weeded out for better or more reusable systems.

---

John-David Dalton

**Lo-Dash**

Lo-Dash is a drop-in replacement for Underscore.js that delivers performance, bug fixes, and additional features.

**Performance**:

-Avoid native methods. This not only gives performance, but compatibility.

-Reduce the number of function calls.

-Use method compilation. (Dynamically creating a method at run time that is optimized for your current environment.)

**Bug fixes**:

-Provides bug fixes for issues we see in certain browsers, making a dev environment that is the same across all browsers.

**Features**:

-AMD support out of the box.

-Deep cloning capability.

-And more…

**Custom Builds**:

-Allows you to make builds which have only the methods you need, keeping your file size small.

-Can make builds for mobile, legacy, strict mode, backbone, etc.

**Backbone Boilerplate**: The next version of boilerplate will include lo-dash.

**Geek Out**: Custom Minification. Custom Compiling. Unit Tests.

Emily Rose

**Hardware and JS**

Hardware controlled with JS. Freaking Awesome! Helicopters, light machines, fog machines, etc. The setup even goes as far as allowing you to hold your hand above certain lights to create effects on the web page. Really cool stuff!

Chris Castle

**Keg + node.js = beer | Keg.io**

A keg controlled by a node.js/js interface. Basically this guy did this just because he could. Lots of win! Beyond just controlling the thing, they captured any and all data about beer consumption from the machine. The graphical output gives you things like the temp of the beer and beer age. Notifications are set up so that if the temp gets too high a screen pops up. The system is also set up with RFID so that you can have a list of people who are allowed to drink from the keg.

Rick Waldron

**robots.js**

Robots controlled with JS. Brilliant! Waldron speaks on programming robots in js with emphasis on the importance of Asyncronization. Robots actually exist physically, so when you tell the robot to do something, you must wait till that action is finished before you can tell it to do something else. Waldron calls robots controlled by a node.js server nodebots.