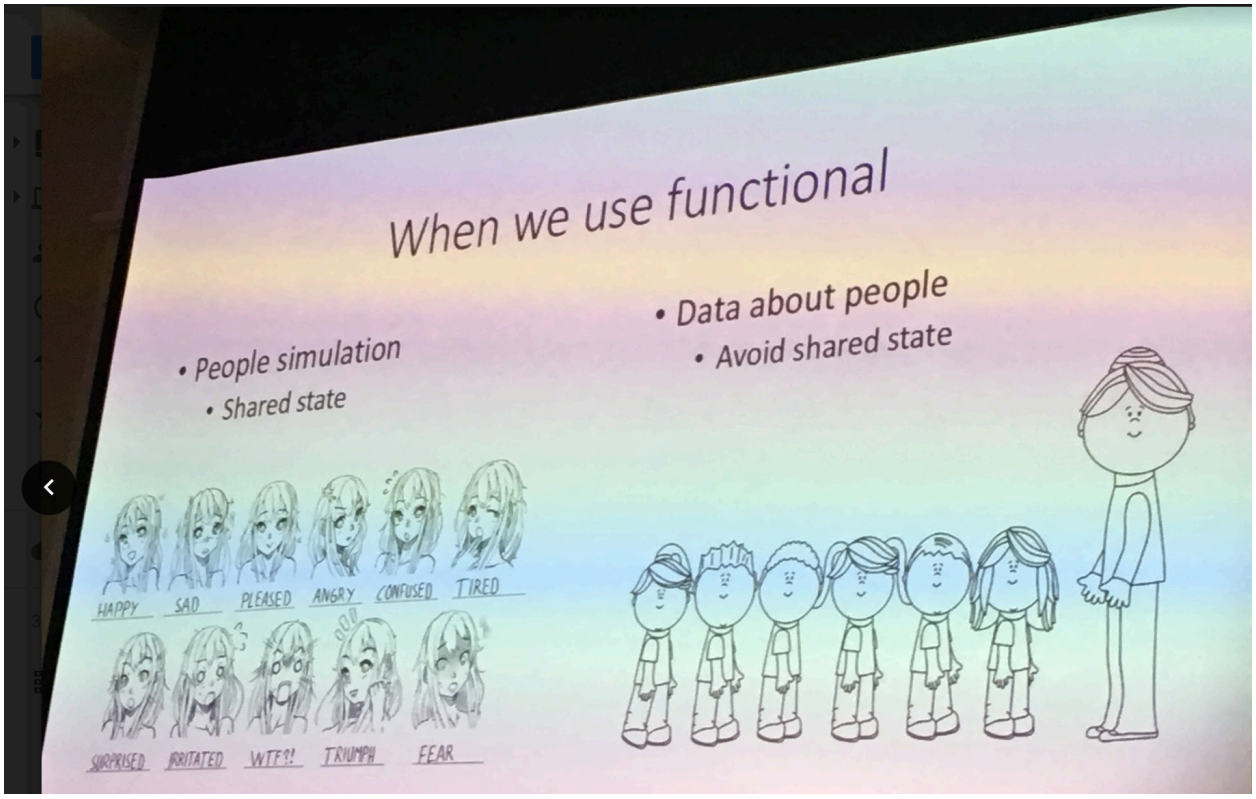
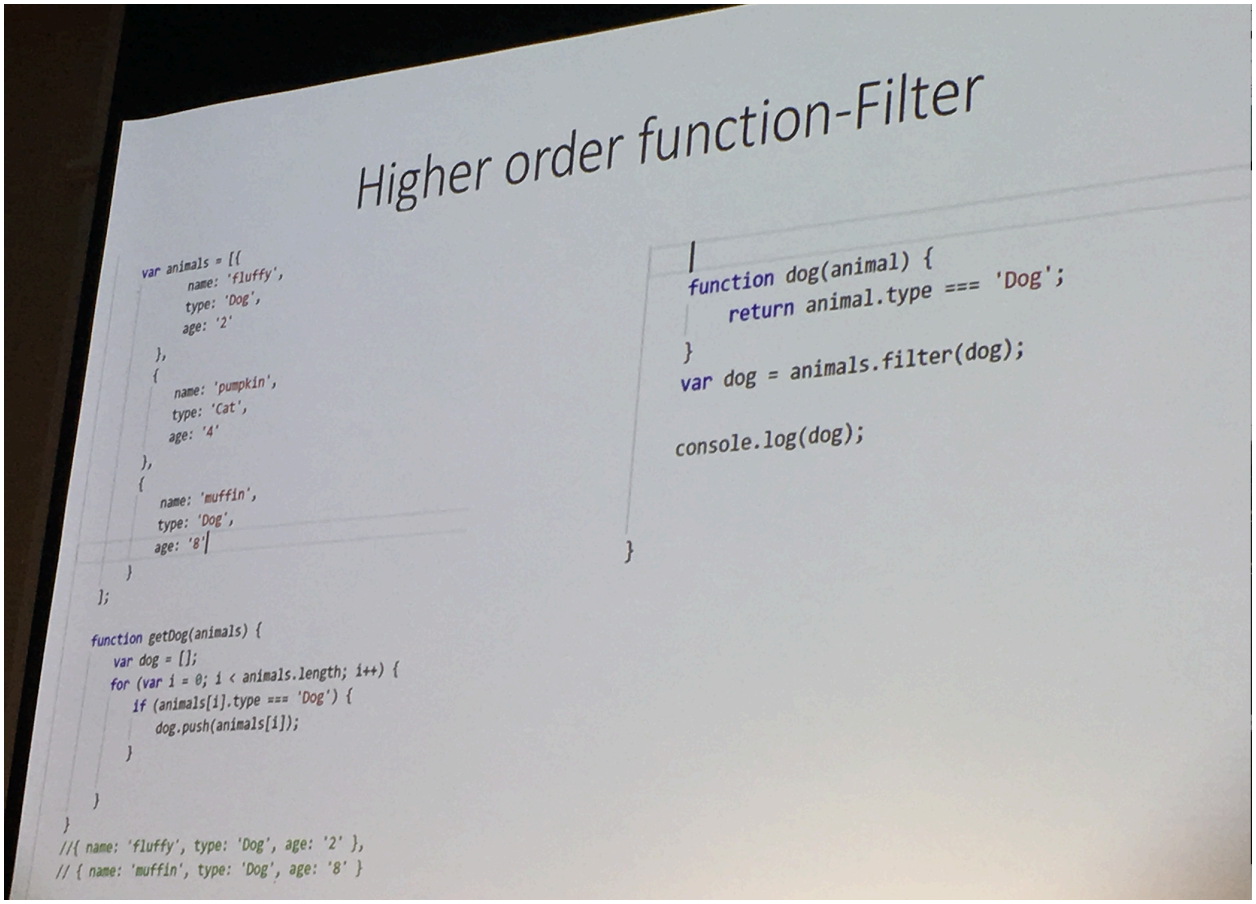
Summary of Sessions at USC SoCal Bootcamp

Last month, I attended this bootcamp, and dabbled in subject matters such as drones, AI, blockchain, AR, chatbot apps using Microsoft, and practical applications like JavasScript. I was pretty impressed with all the presentations. The links provided are from the speaker's github or slideshare/twitter accounts. They are presented in chron order...

**Introduction to Functional Programming** by Farzaneh Orak
She went over some key functional programming principles and showed side-by-side comparisons between OOP and FP. I really like this slide she presents:




So the main message is you can write cleaner and shorter code, variables are immutable therefore data mutation is avoidable. She showed some higher order function examples, such as map, filter, and reduce. Here is one example of filter. To the left is a function written in OOP, and to the right, the same function is written in FP.




--

**A Developer's Survey** to **AI Methodologies** by Barry Stahl
I thought I was going to be in way over my head in this but Mr. Stahl explains the common models of AI by applying this to the game *Chutes and Ladders* and another example of a Slack thread. There are 3 models he covers, which he explained so simply:

• Logic
• Probabilistic/Learning (this can use graphs like Bayseian/Neural networks or Genetic Algorithms)
• Search/Optimization

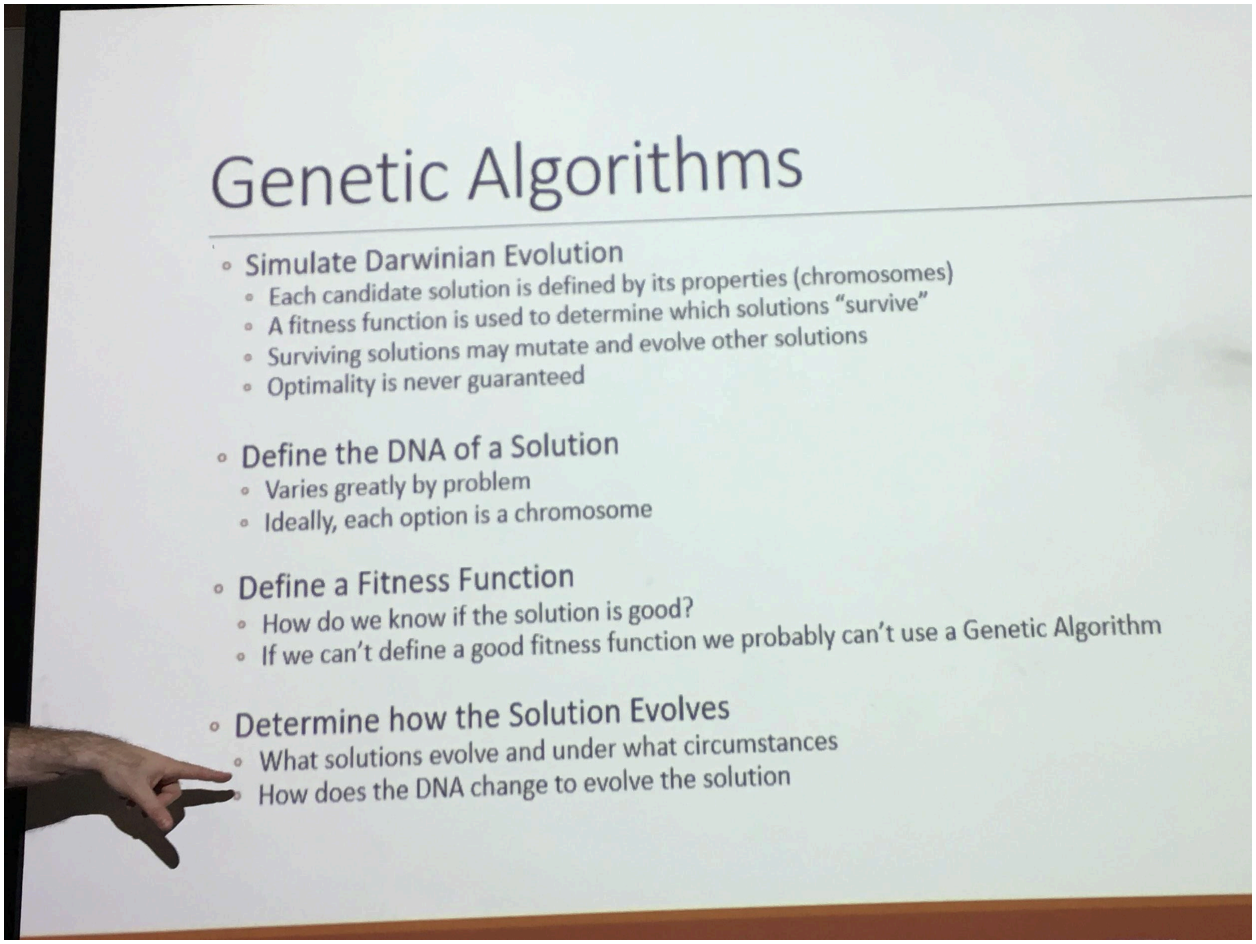
The logic model usually uses conditionals and is rule-based and uses a rules engine. The object-oriented logic models are greedy, linear and aggressively bad strategy. The rules engine asks after each spin, does the rule apply? Where do you want to end up (apply the greedy or linear algorithms). The greedy engine algorithm tires to get the best outcome or "points (if applicable)". The linear strategy gives us a tool to test, so think of these as "link queries." The answer is a boolean: Y/N or T/F. If you look at the board game, it is like a

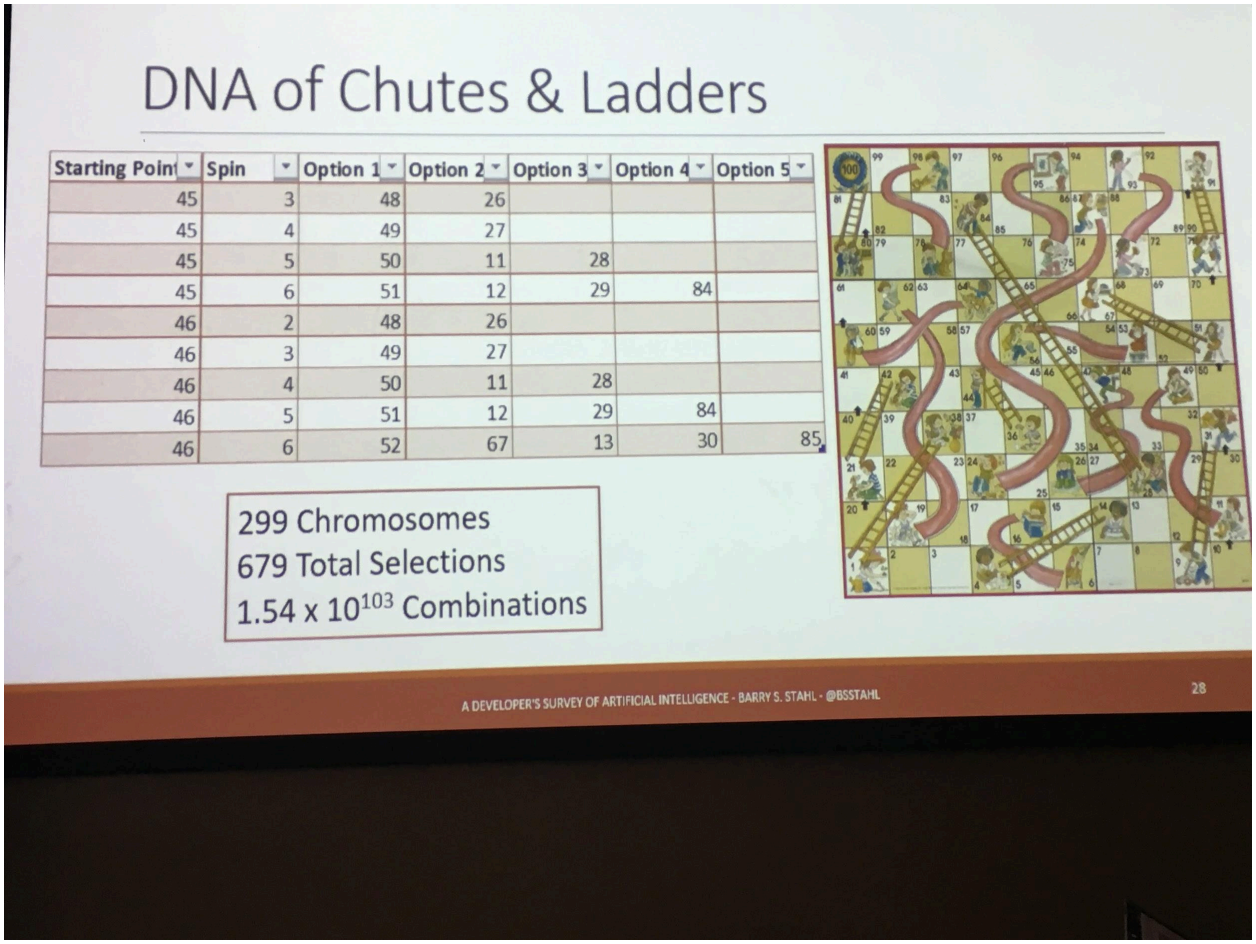
linear array. Wash, rinse and repeat until the game has ended. Track the winners and apply the models to the players.

The probabilistic model is usually unpredictable and analyzes whether an action is directed/causation or undirected/correlation.

The genetic model defines a feature as a chromosome as 1 action and 1 spin, and the nucleotide as a result. The model tracks the surviving chromosome and feature. Here are the rules he created:

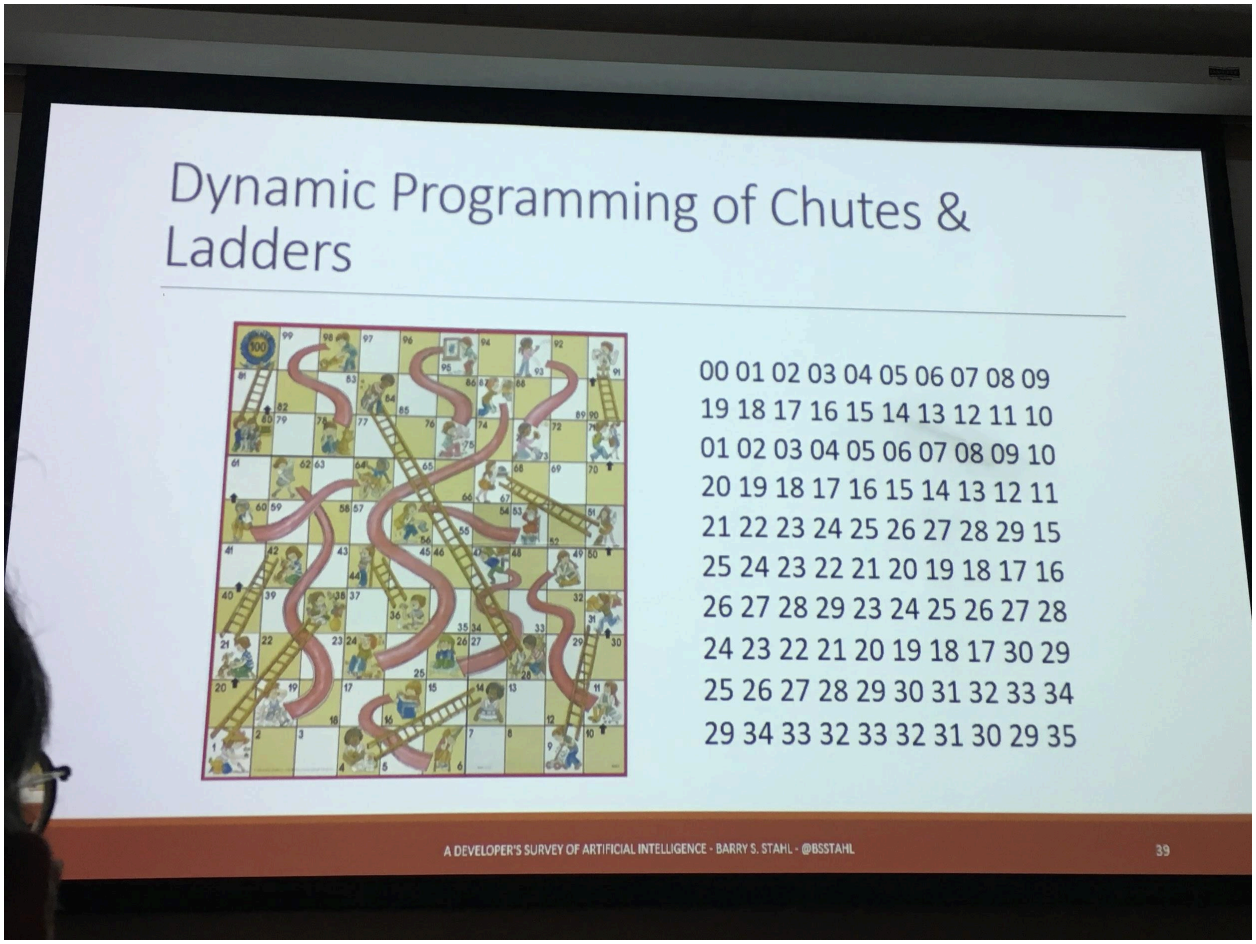
- A candidate is defined by its chromosome
- Each feature is a chromosome
- Each chromosome is equal to one action (ie 1 space or square and 1 spin)
- The rules engine says: "when I'm here and I spin this..." I get a result, which is a nucleotide or option
- Track surviving chromosome/feature -- the percentage of solutions or chromosomes surviving and what condition
- Calculate the maximum number of options (which he said was 5 since he tried them)
- The rows in the figure below represent a chromosome
- The nucleotide represent options or columns
- Select starting DNA to all players using a linear strategy


Genetic Algorithms
Simulate Darwinian Evolution
Each candidate solution is defined by its properties (chromosomes)
A fitness function is used to determine which solutions "survive"
Surviving solutions may mutate and evolve other solutions
Optimality is never guaranteed
Define the DNA of a Solution
Varies greatly by problem
Ideally, each option is a chromosome
Define a Fitness Function
How do we know if the solution is good?
If we can't define a good fitness function we probably can't use a Genetic Algorithm
Determine how the Solution Evolves
What solutions evolve and under what circumstances
How does the DNA change to evolve the solution




He takes the same game and applies "depth-first-search" [https://en.wikipedia.org/wiki/Depth-first_search] where you start from the root and calculate the possible paths in a tree structure.

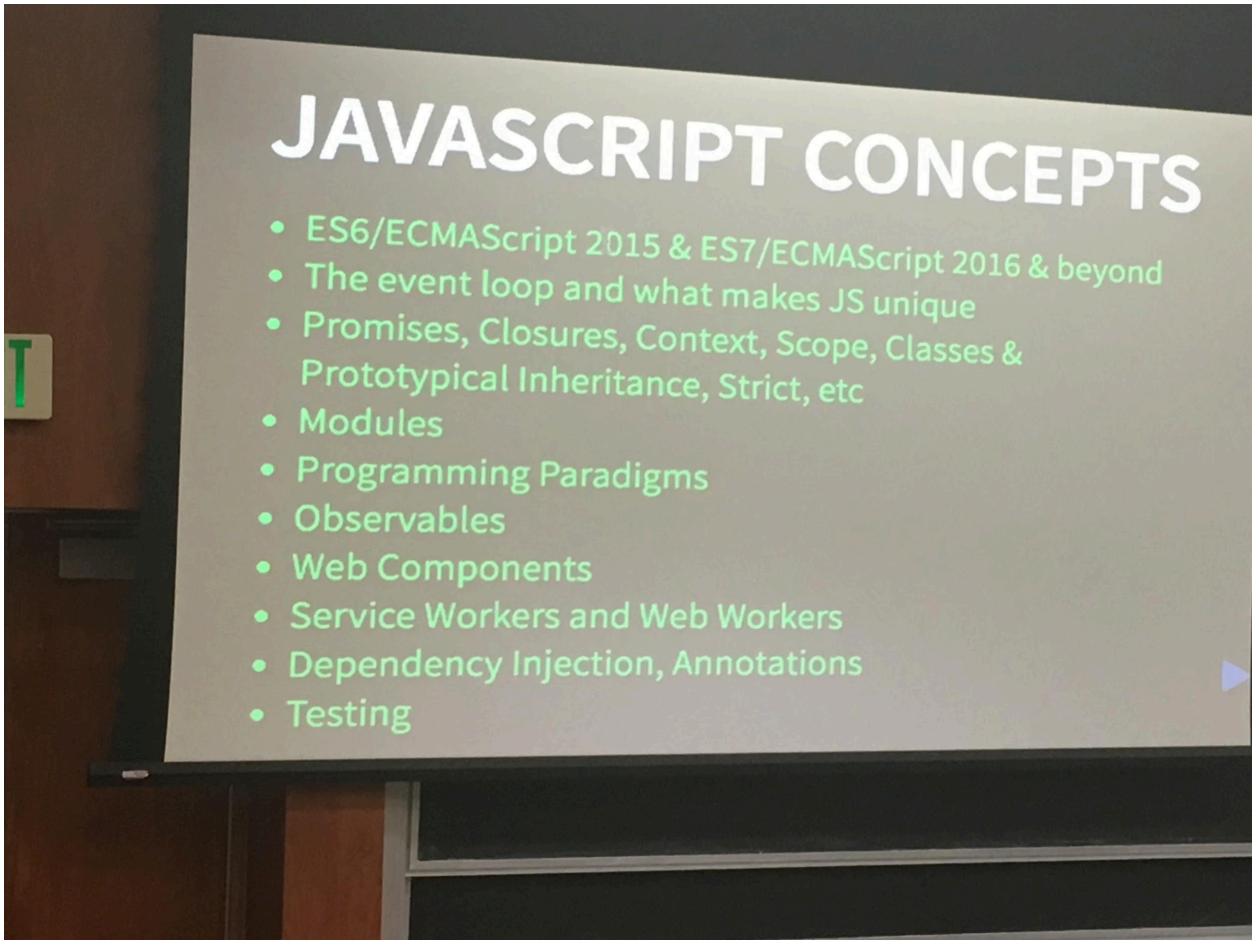
Starting from the top left at square 100 -> you don't need any moves, hence 00. If you look at space 99, this is the shortest path to winning. Dynamic programming is similar to calculating "Fibonacci" sequence.




There was also another demo he showed us using Slack and Microsoft's LUIS engine.

For his demos and Github:
https://github.com/bsstahl/AIDemos

--
**React Native Quickly** by Troy Miles
For more information, here is his deck:
https://github.com/Rockncoder/td/blob/master/react-native-quickly.pdf
And these are his demo files: https://github.com/Rockncoder/td
--
**Survey of JavaScript Concepts 2017** by Todd Zebert




If you are interested in any of these concepts, please view his slides here: https://speakerrate.com/talks/74151-survey-of-javascript-concepts-2017 orhttp://spkr8.com/t/74151

Just click on the slides link:

**DECEMBER 3, 2017 1:45 PM**
**LOS ANGELES, CA**
ES variants, transpiling, and polyfills/shims ES Engines Package management: npm, Bower, etc Task runners: Grunt and Gulp Web APIs Linting Web Components: Polymer, and related libs Functional Programming libs: Underscore.js, lodash.js, Ramda.js, Functional.js, etc

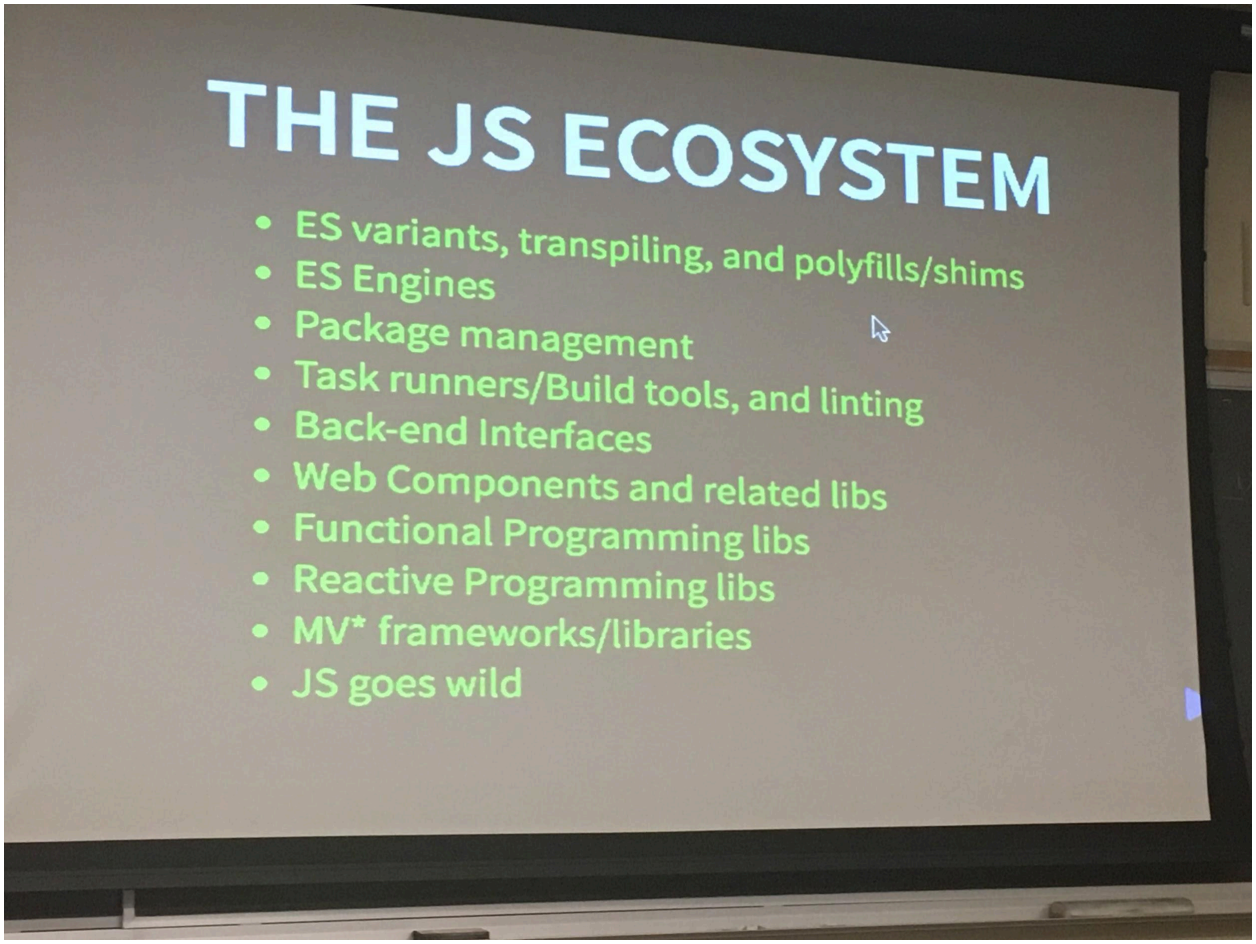
Show more information

http://spkr8.com/t/74161 Copy URL

**Links:** Website Slides

--

**The JS Ecosystem** by Todd Zebert

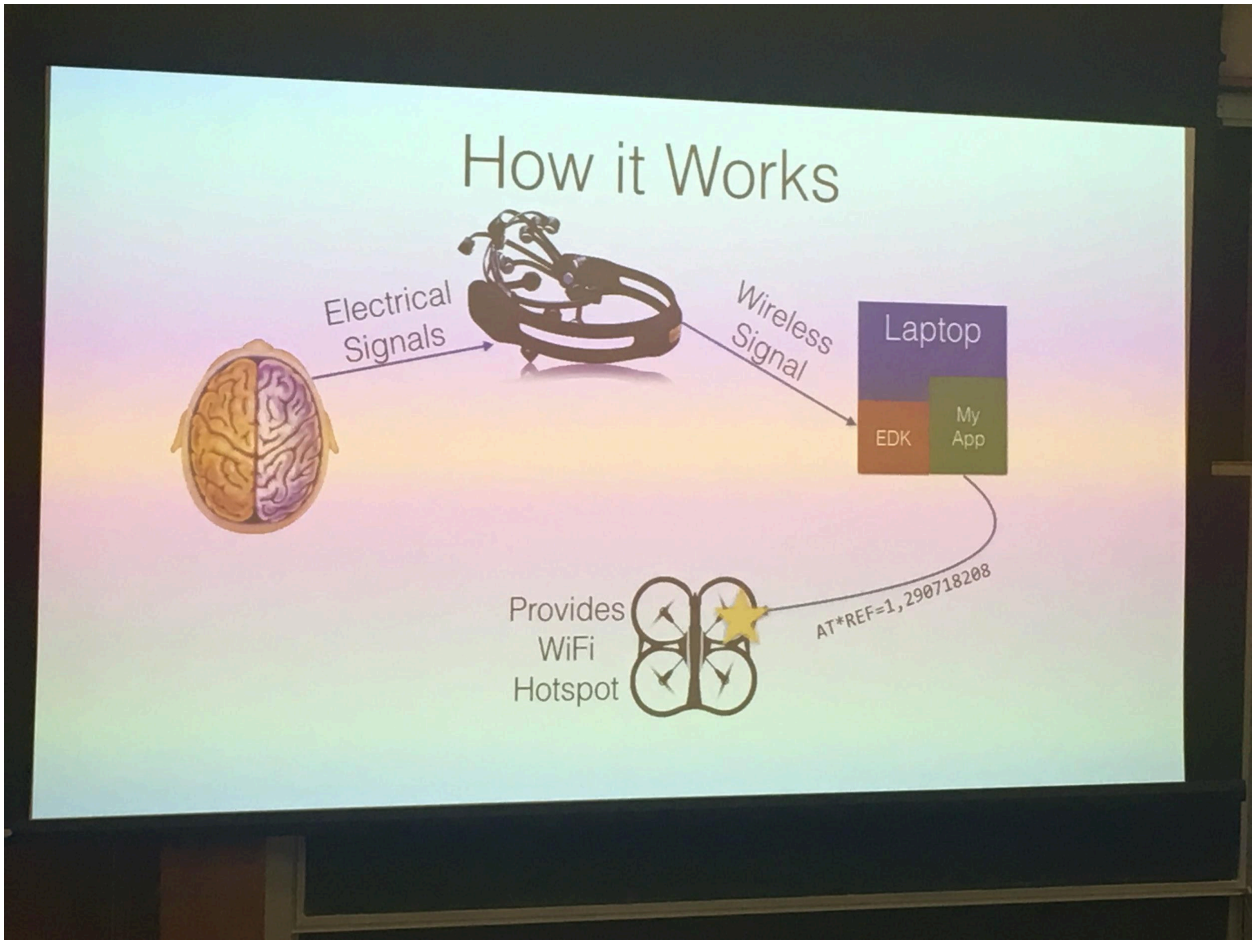


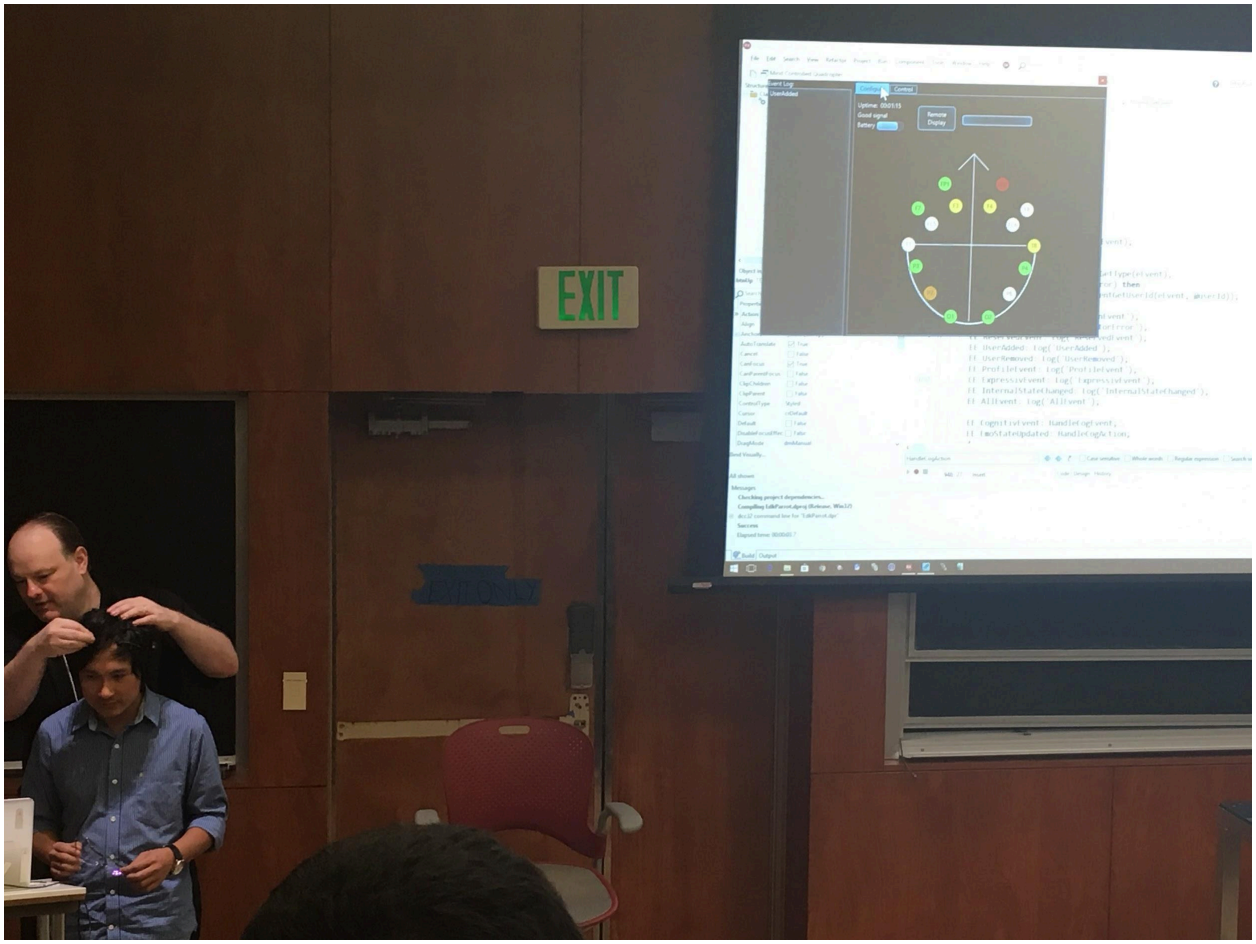

If you are interested in any of these concepts, please view his slides here: https://speakerrate.com/talks/74161-survey-of-the-javascript-ecosystem-2017 orhttp://spkr8.com/t/74161

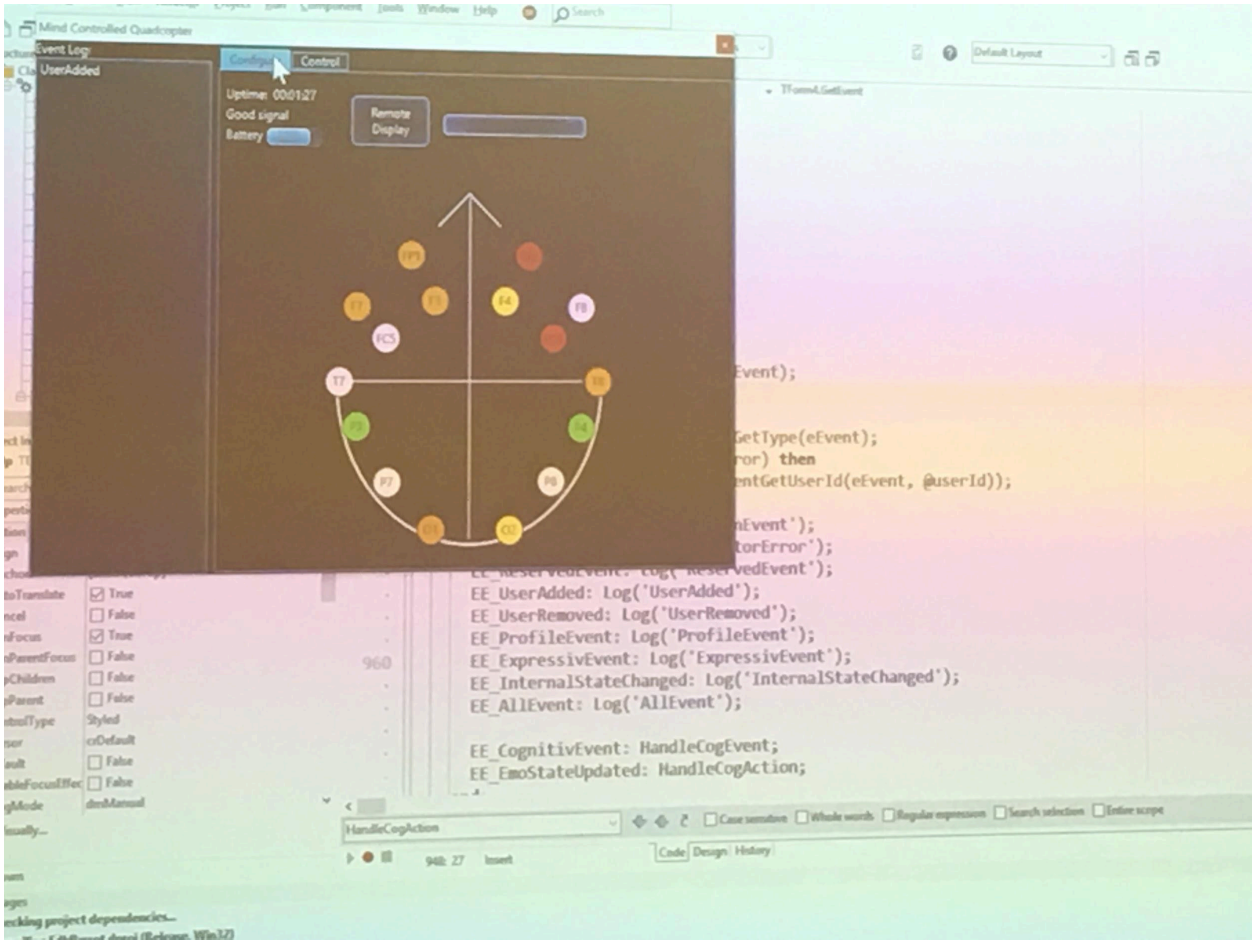
--

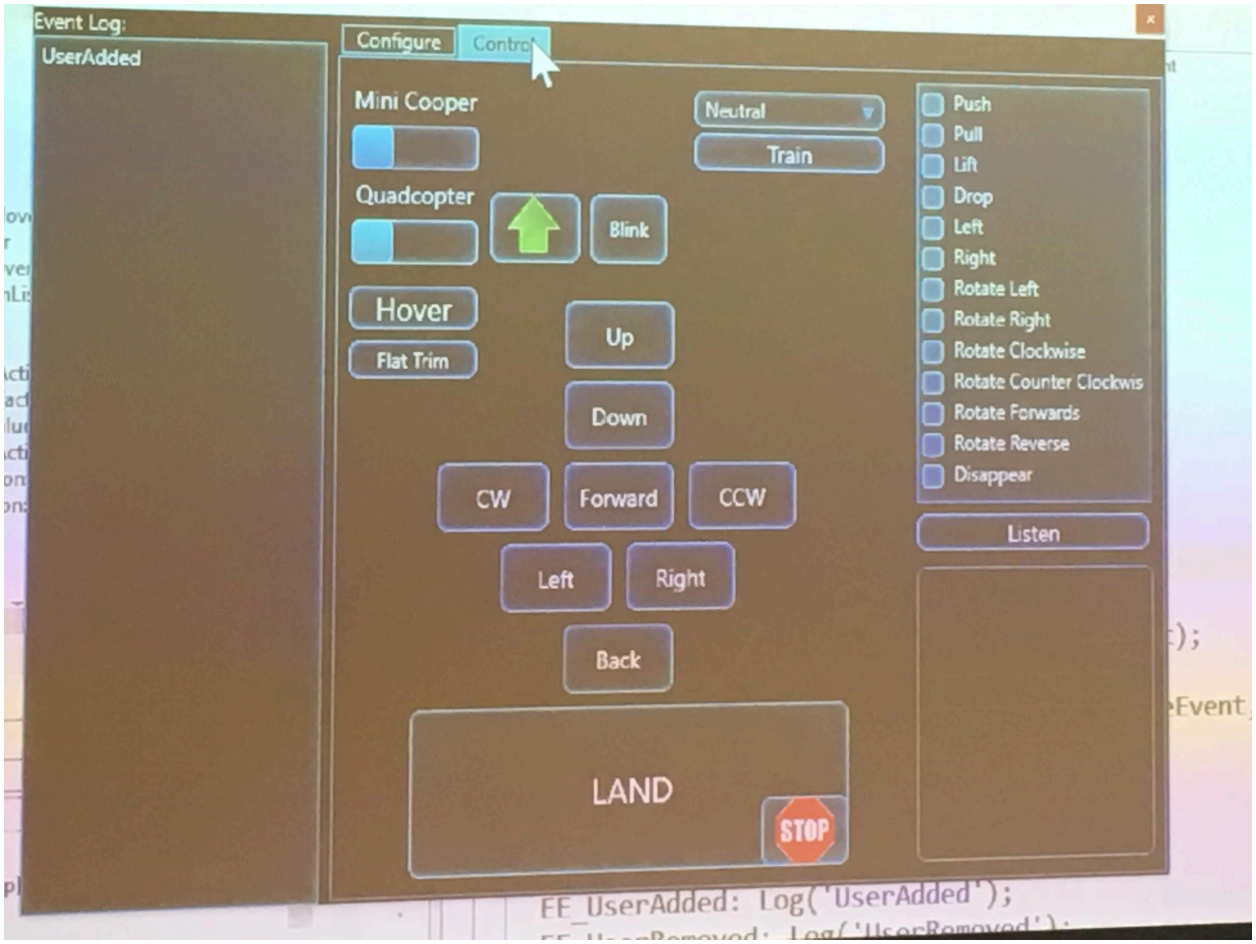
**Mind-Controlling Drone** by Jim McKeeth

I was pretty skeptical when I attended this demo, but his presentation of how the brain communicates using electrical signals, and his experimentation with combining an Emotiv headset [https://www.emotiv.com/ ] with a Parrot Drone [http://developer.parrot.com/ ] was impressive.


How it Works
Electrical Signals
Wireless Signal
Laptop
EDK
My App
Provides WiFi Hotspot
AT*REF=1,290718208


EXIT


Mind Controlled Quadcopter
Event Log:
UserAdded
Configure
Control
Uptime: 00:01:27
Good signal
Battery
Remote Display
EE_UserAdded: Log('UserAdded');
EE_UserRemoved: Log('UserRemoved');
EE_ProfileEvent: Log('ProfileEvent');
EE_ExpressivEvent: Log('ExpressivEvent');
EE_InternalStateChanged: Log('InternalStateChanged');
EE_AllEvent: Log('AllEvent');
EE_CognitivEvent: HandleCogEvent;
EE_EmoStateUpdated: HandleCogAction;
HandleCogAction




Here is the link to my video of this demo:
https://www.linkedin.com/feed/update/urn:li:activity:6343178127509979136
This is the link to his demo in Github: https://github.com/jimmckeeth/Delphi-Emotiv-EPOC
Here is his deck:
https://www.slideshare.net/jimmckeeth/jim-mc-keeth-wearable-thought-input

--

I also attended a blockchain presentation and one in Augmented Reality (AR) using Apple' ARKit Scene, which I will later post separately because I want to try the demo out first.