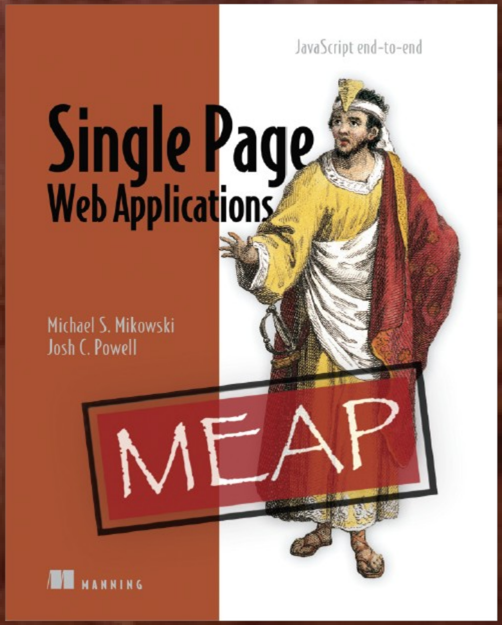
Make your SPA rock (solid)

Designing and testing SPAs for performance and quality

Single Page Web Applications
http://manning.com/mikowski

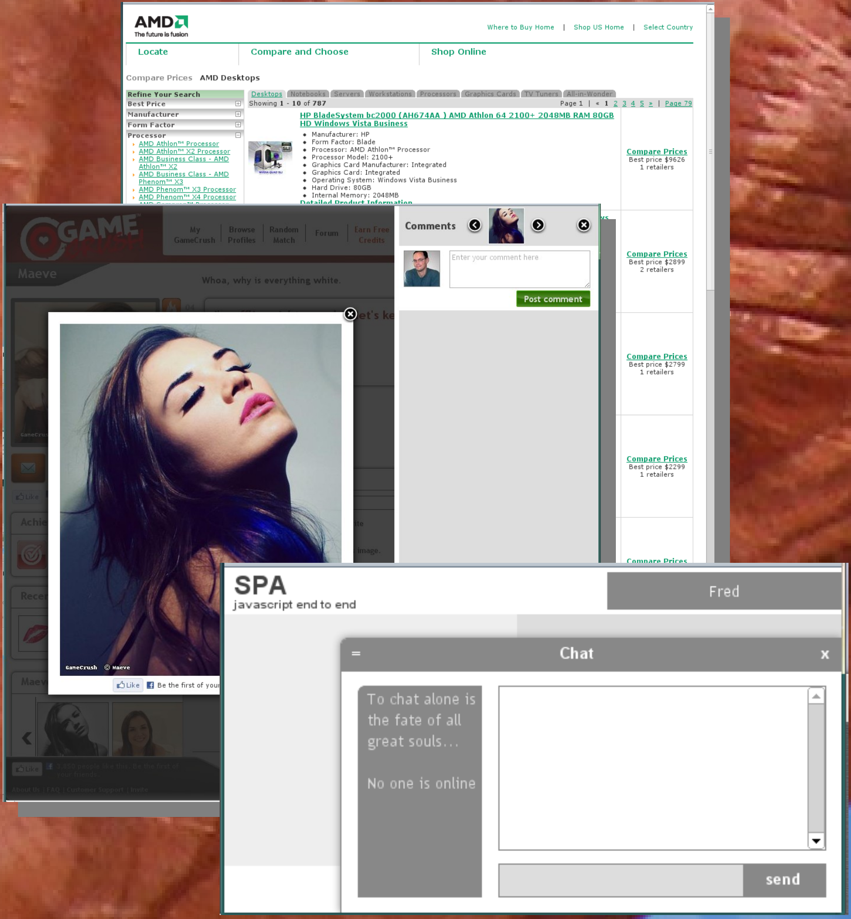
About Michael S. Mikowski

- **Single Page Web Applications** – JavaScript end-to-end
- **Senior Director, UX Engineering** at Qualaroo
- Developer on **Six production SPAs** since 2006, **Architect** on all but one
- Previous dev manager on **HP/HA mod_perl clusters** (~2B web transactions per week)
- First SPA: European and US AMD "**wheretobuy.amd.com**," rel. 2007



About SPAs...

- SPAs are web applications that **don't reload** during a user session
- Users are now expecting **native application-like** performance
- **SPAs have been around a long time:** Flash games, Java office suites, Javascript calculators
- We are are talking about **Navtive JavaScript SPAs**

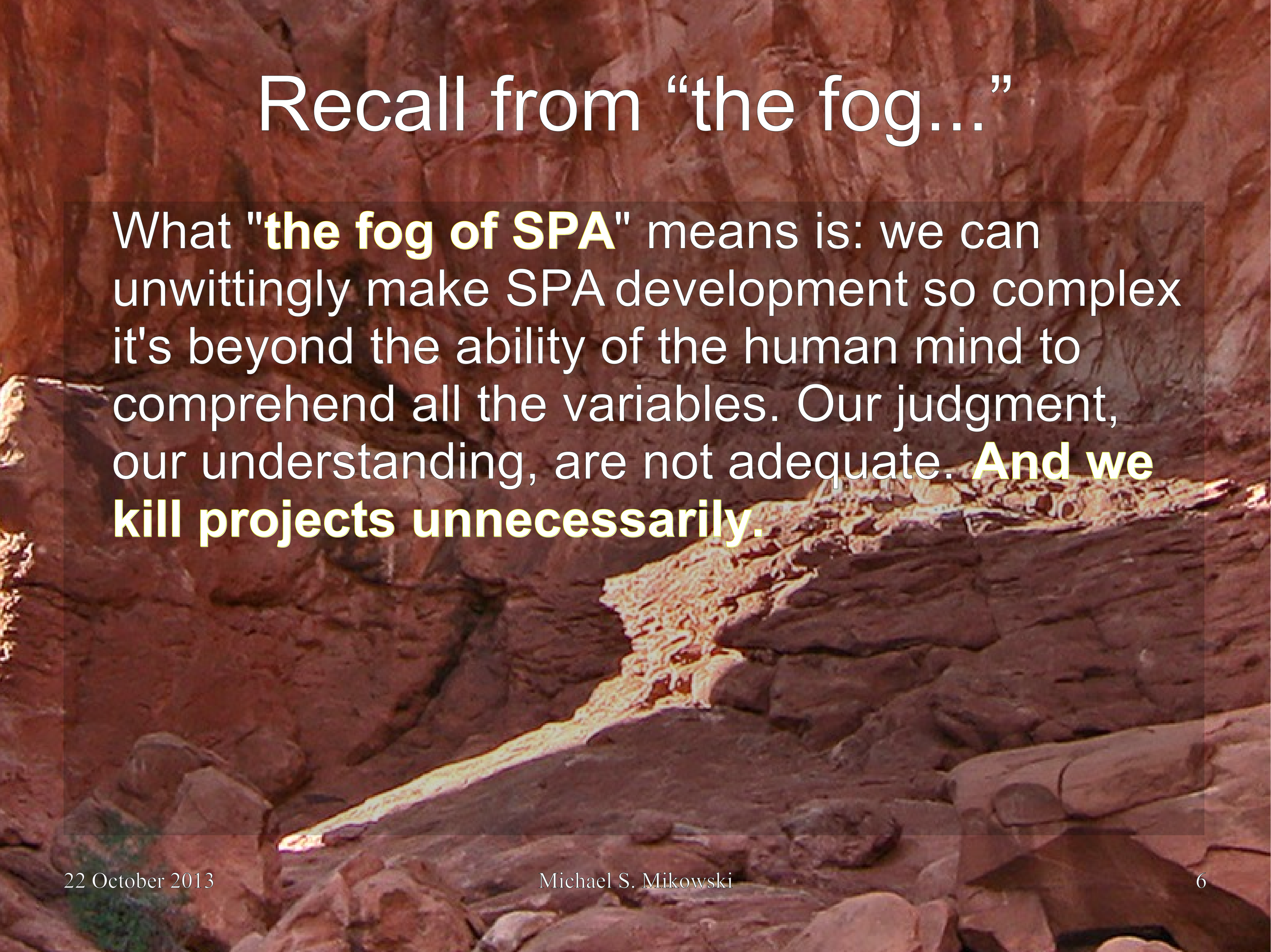


A bit more about SPAs

- Business logic: **Server** $\Rightarrow$ **Browser**
- JS coding at a scale an order of magnitude greater than a traditional websites (100k lines)
- One SPA may require **many developers**
- **Conventions and discipline previously reserved for server-side development becomes a must for working at this scale**

About this presentation

- This is a sequel to April's "**The Fog of SPA**"
- You need not have attended the prior presentation to win
- This will not be slam poetry presentation
- Be grateful this is not a sequel to "Monsters Inc."
- We will be working with latest, "live" code
- "Comfortably Numb"



Recall from “the fog...”

What "**the fog of SPA**" means is: we can unwittingly make SPA development so complex it's beyond the ability of the human mind to comprehend all the variables. Our judgment, our understanding, are not adequate. **And we kill projects unnecessarily.**

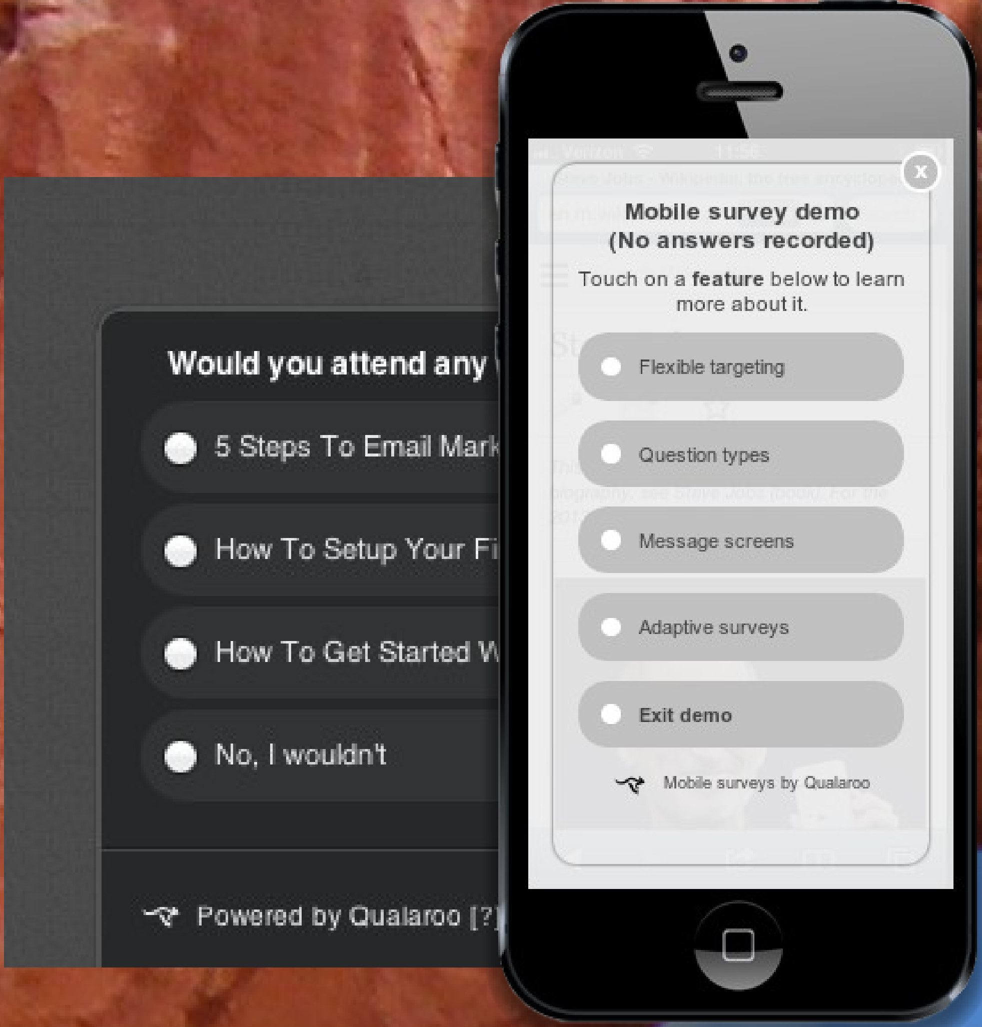
Seven lessons of SPA dev

1) Architect for workflow and testing
2) Design third-party style modules
3) Start at the front
4) Plan for many SPAs
5) Use a common language
6) Test the client back-end
7) Avoid shiny objects

Conventions and discipline previously reserved for server-side development become a must when working at this scale!

What does Qualaroo Do?

- Qualaroo provides online surveys and conversion products
- We *are* third party JS
- Interesting req's
  - No jQuery
  - No external images
  - No external CSS
  - In-line data


Would you attend any

5 Steps To Email Mark

How To Setup Your Fi

How To Get Started W

No, I wouldn't

Powered by Qualaroo [?]

Mobile survey demo
(No answers recorded)
Touch on a **feature** below to learn more about it.

Flexible targeting

Question types

Message screens

Adaptive surveys

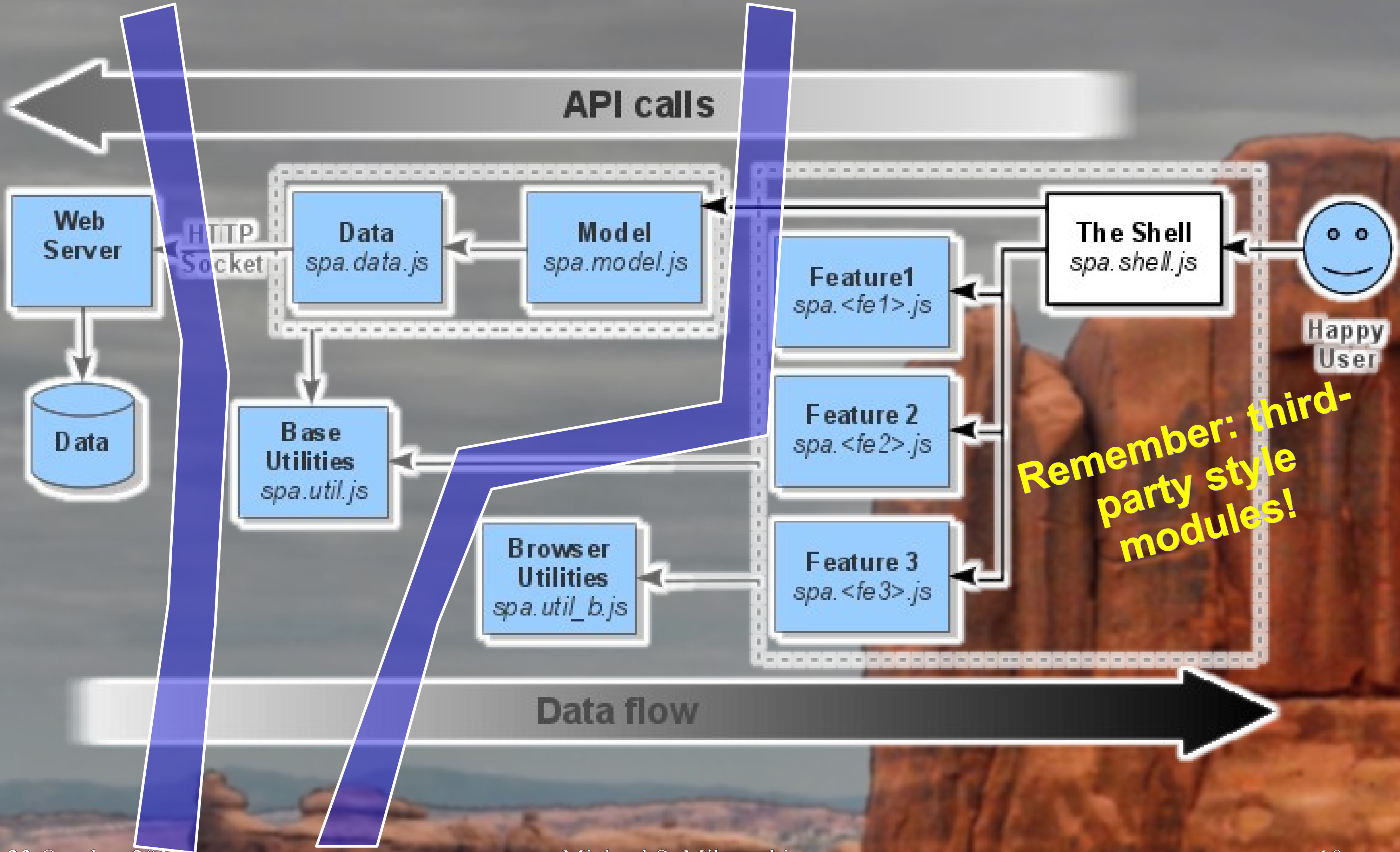
Exit demo

Mobile surveys by Qualaroo

Demo 1: Emulator

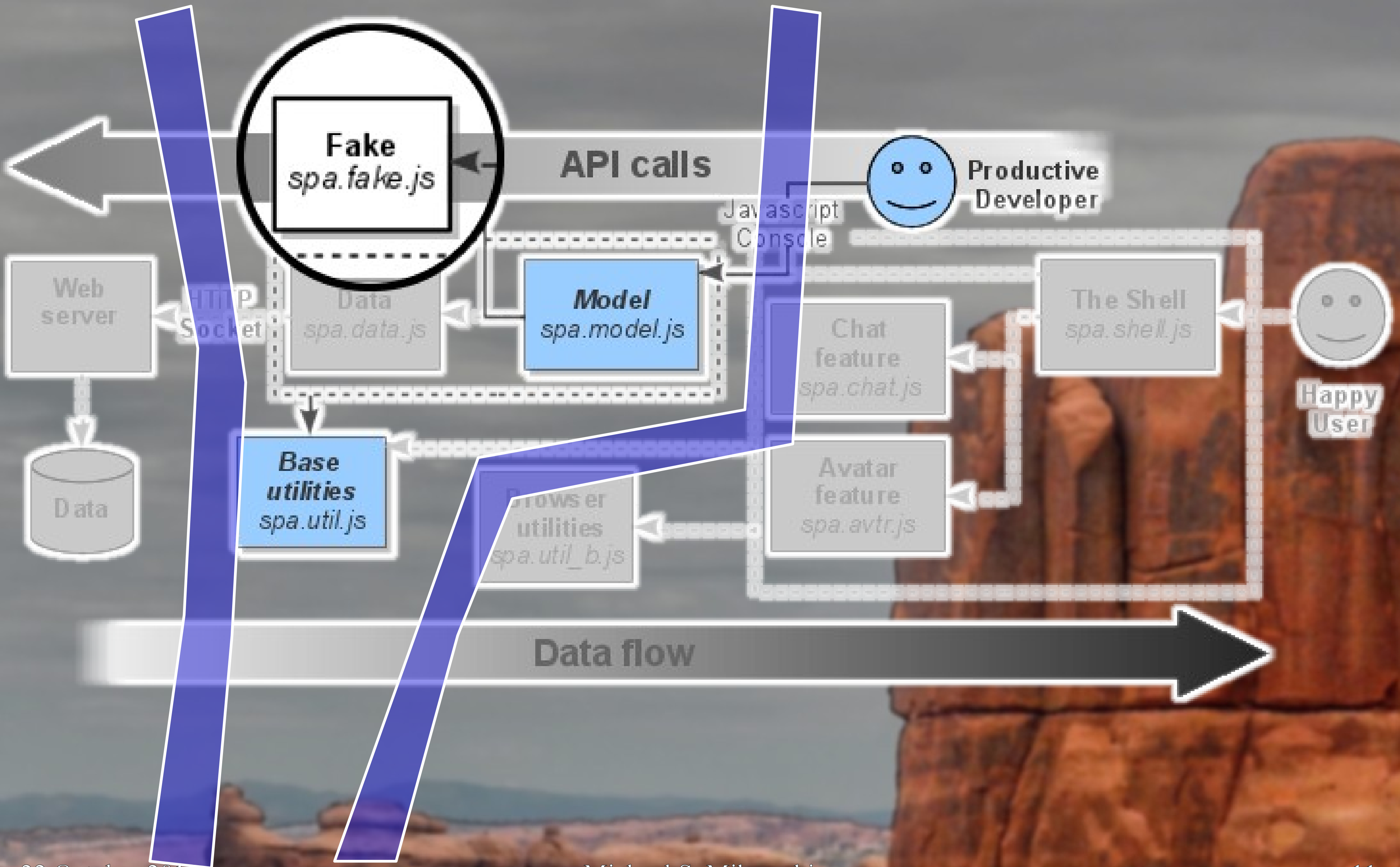
- SDRC command language
- A good test of a model – can you run it from a command line?
- Start backend, get model (model = KI._nudgeModel_)
- Respond to screener (postResponse)
- Stop the Survey (stopNudge)
- Start the survey (selectNudge)
- Walk through responses – show NPS (postResponse)
- Select node ('_3', '_a', etc)

The architecture



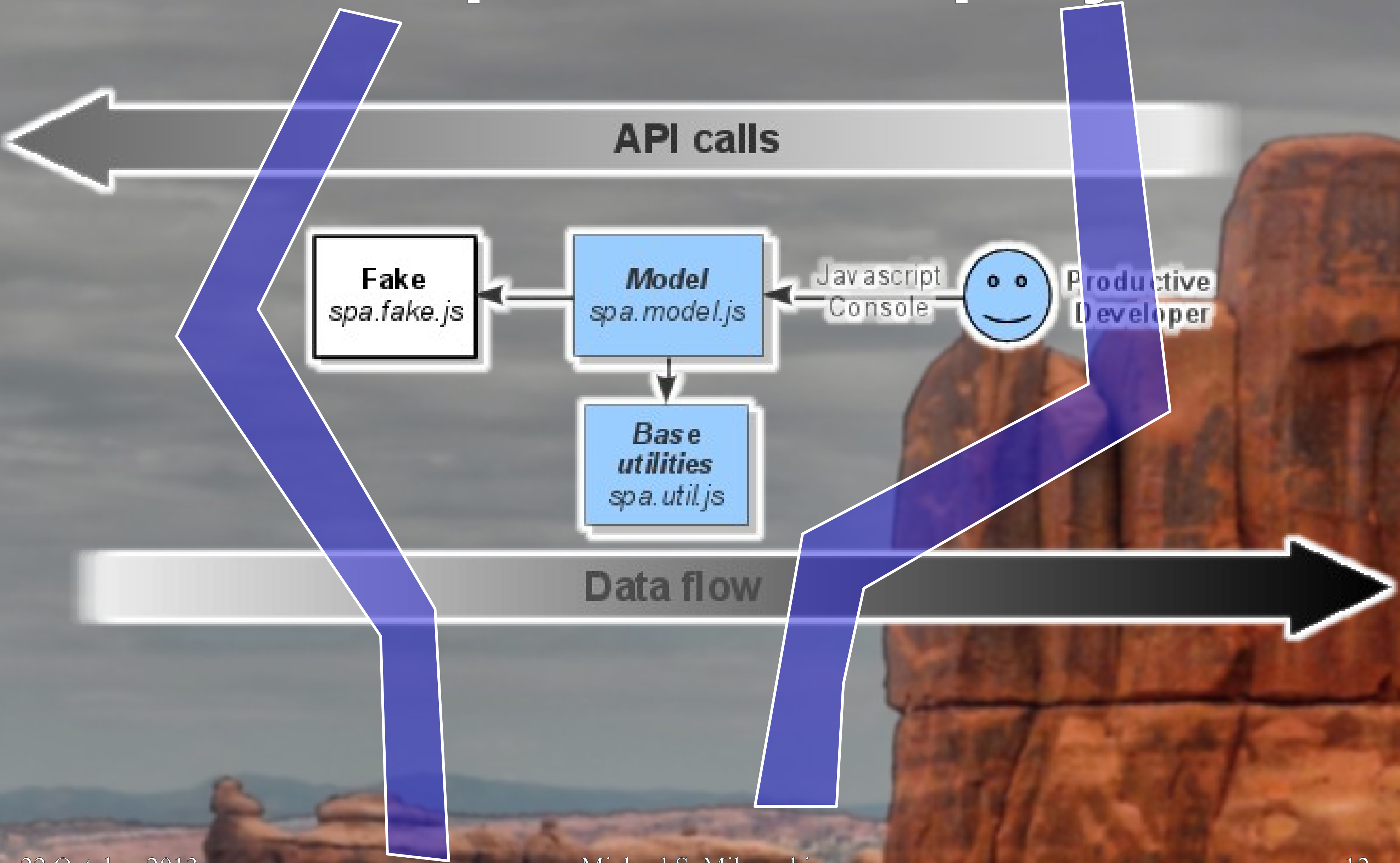
Remember: third-party style modules!

Fake data



Fake
spa.fake.js
API calls
Productive
Developer
Javascript
Console
Web
server
HTTP
Socket
Data
spa.data.js
Model
spa.model.js
Chat
feature
spa.chat.js
The Shell
spa.shell.js
Happy
User
Data
Base
utilities
spa.util.js
Browser
utilities
spa.util_b.js
Avatar
feature
spa.avtr.js
Data flow

Components in-play



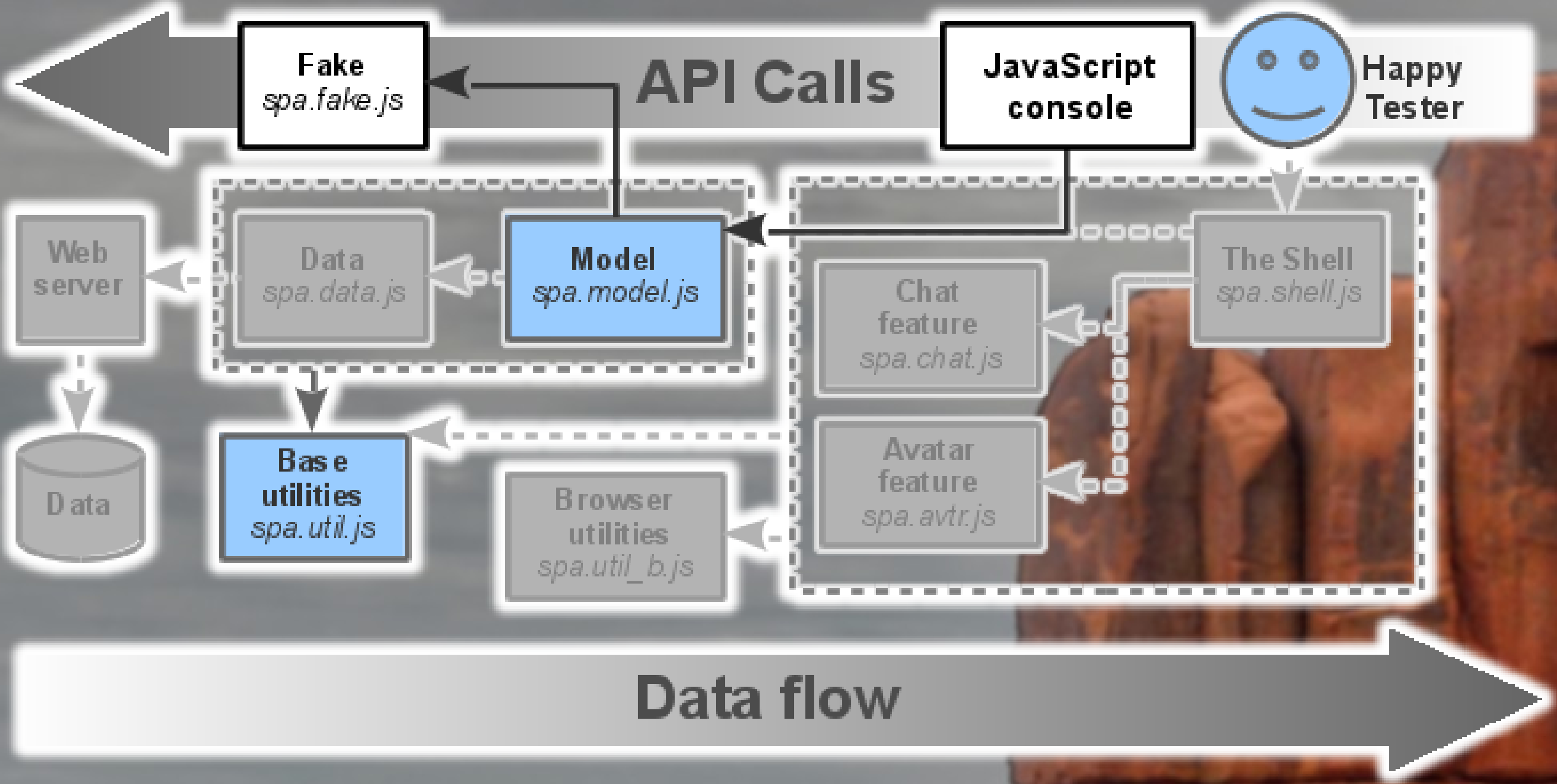
Six testing modes

1) Fake data, Model, JS Console
2) Fake data, Model, Test Suite
3) Fake data, Model, Browser
4) Live data, Model, JS Console
5) Live data, Model, Test Suite
6) Live data, Model, Browser

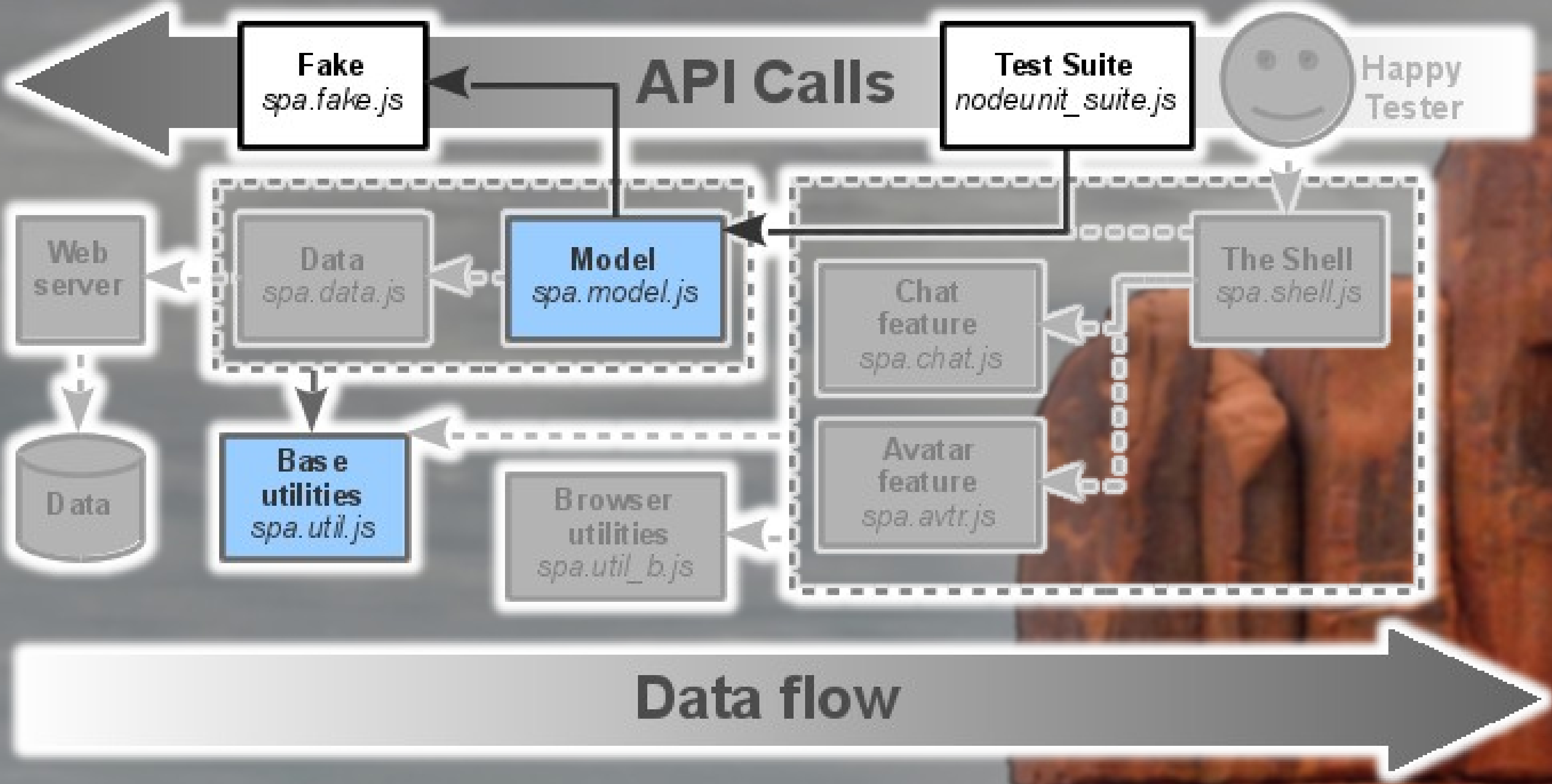
best

worst

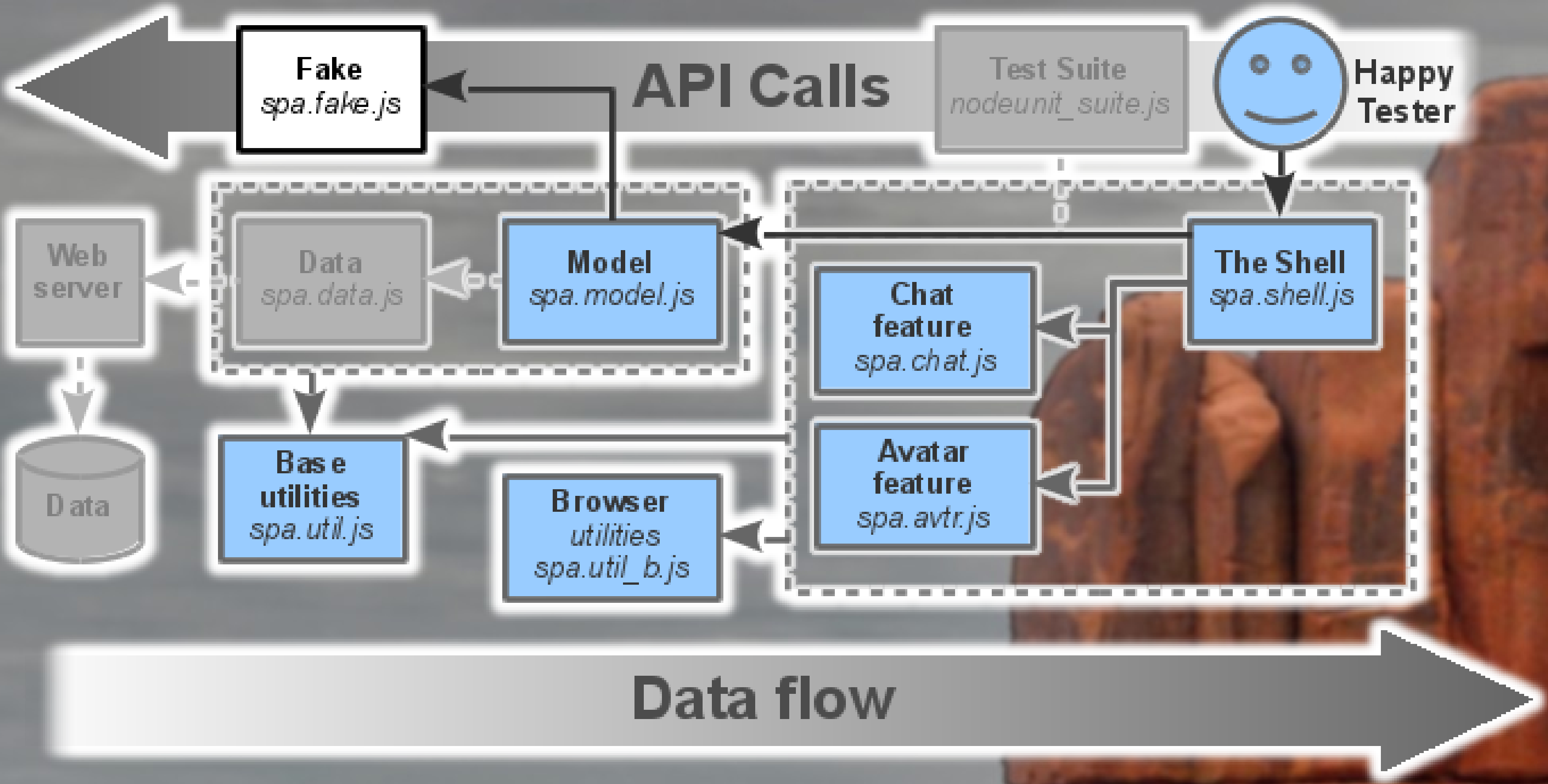
1) Fake data, model, console



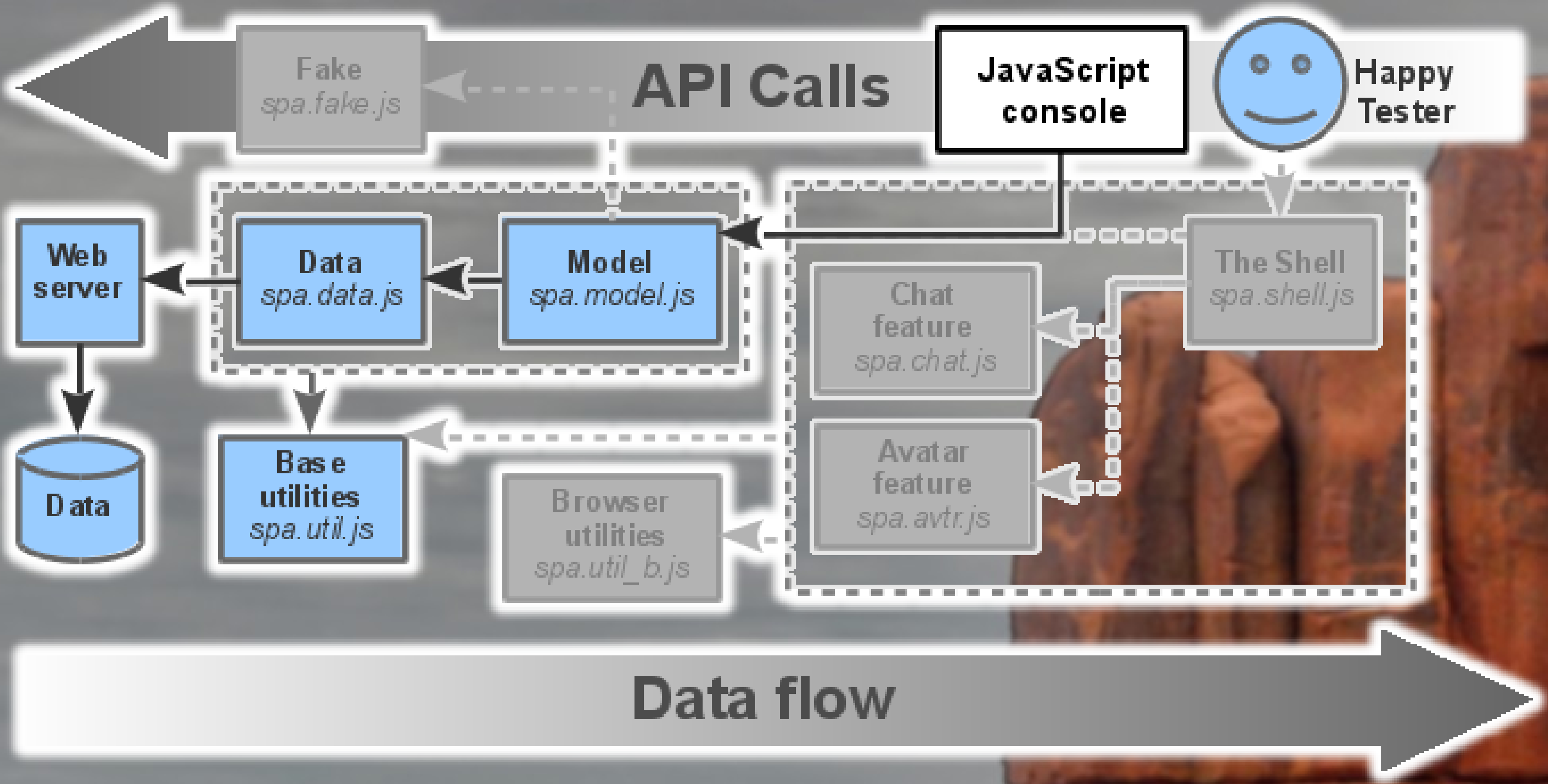
2) Fake data, model, test suite



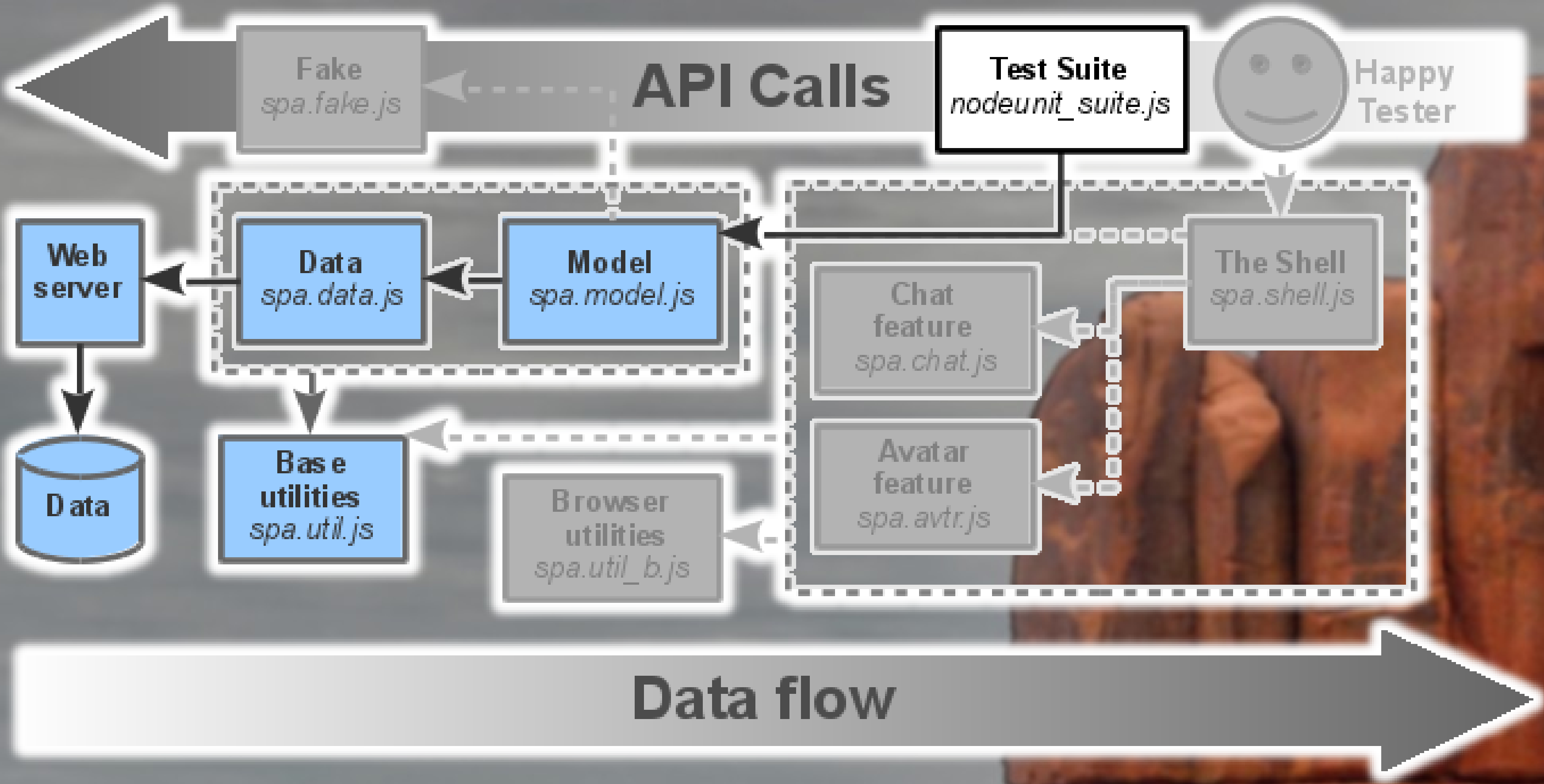
3) Fake data, model, browser



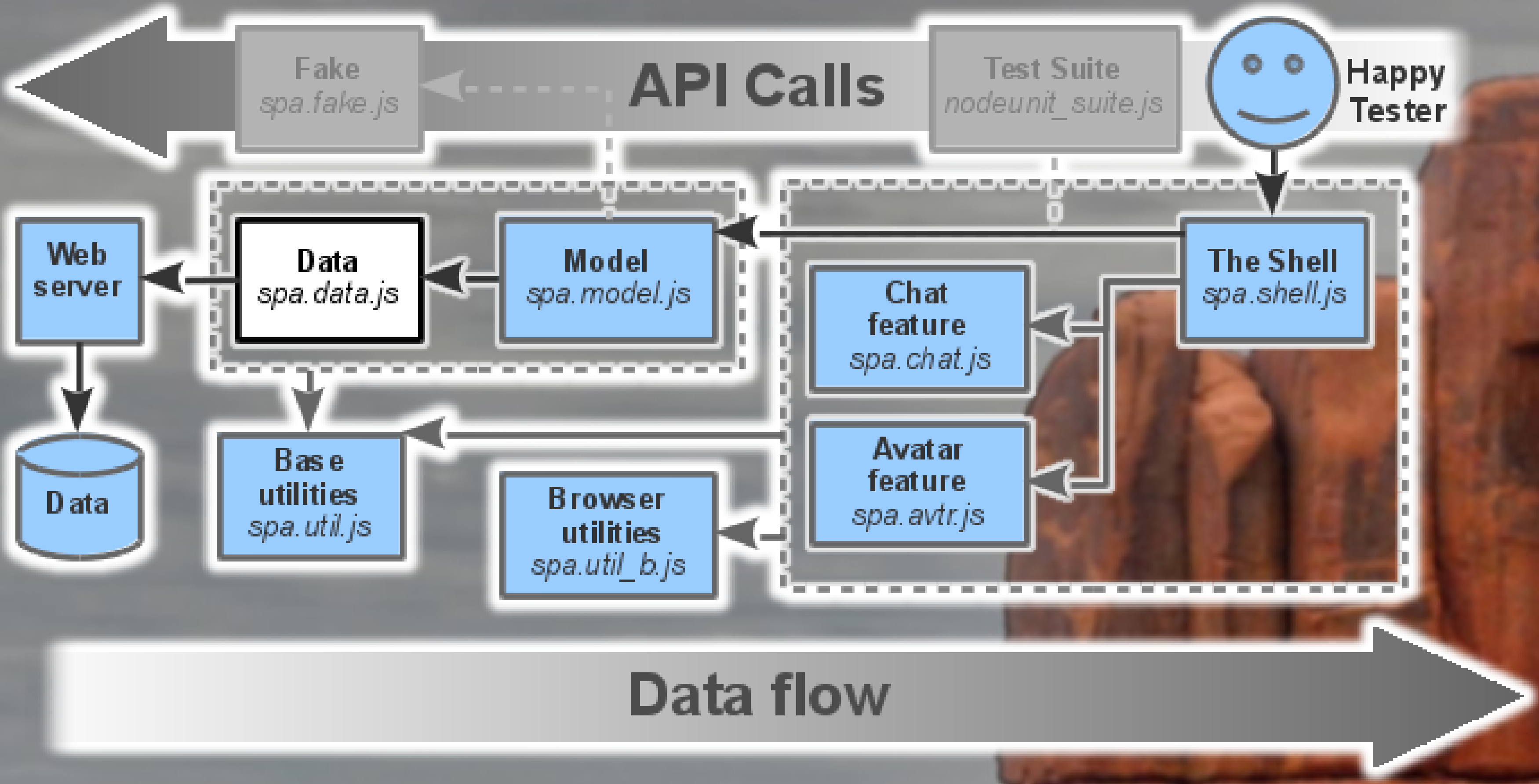
4) Live data, model, console

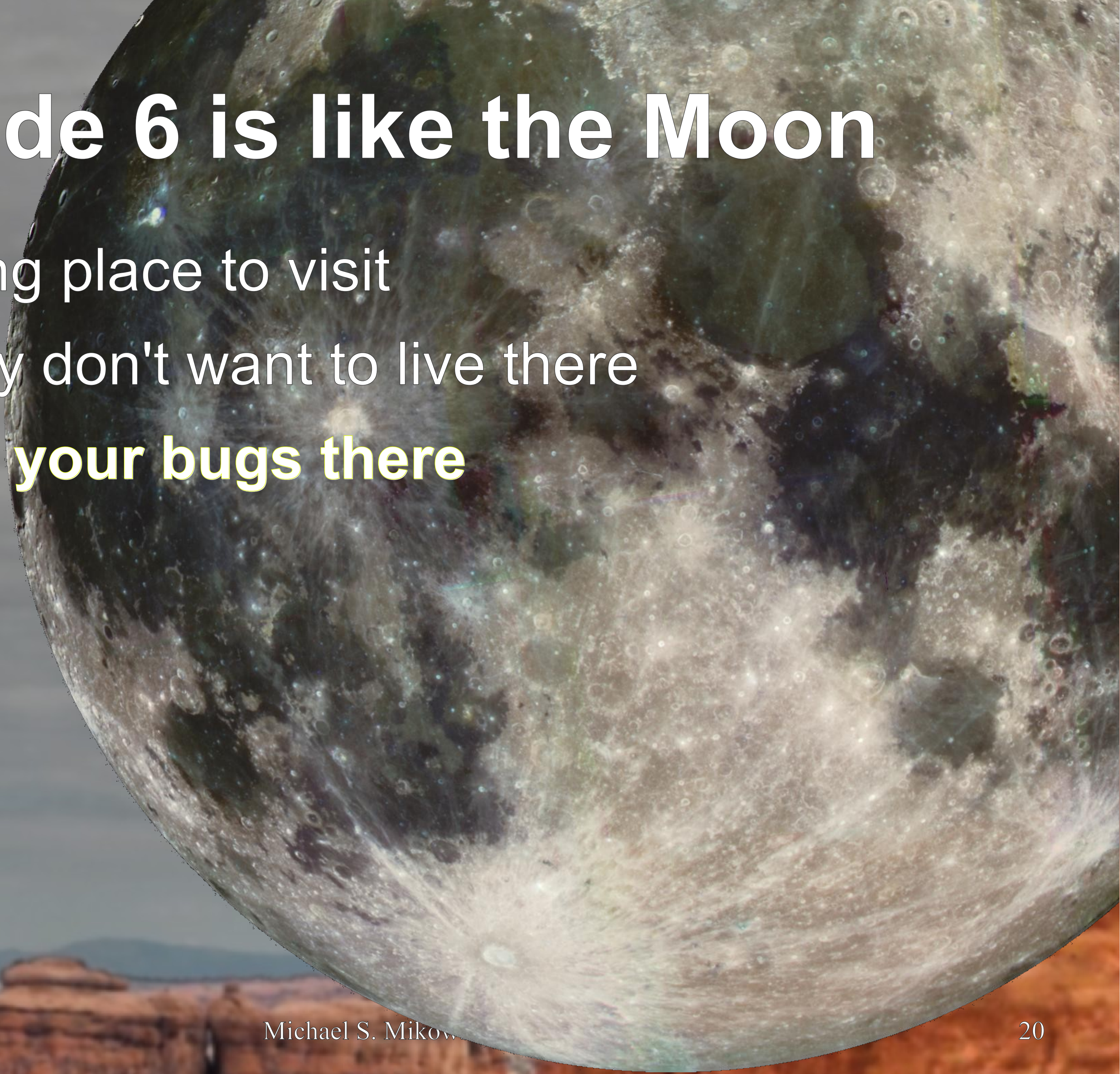


5) Live data, model, test suite



6) Live data, model, browser





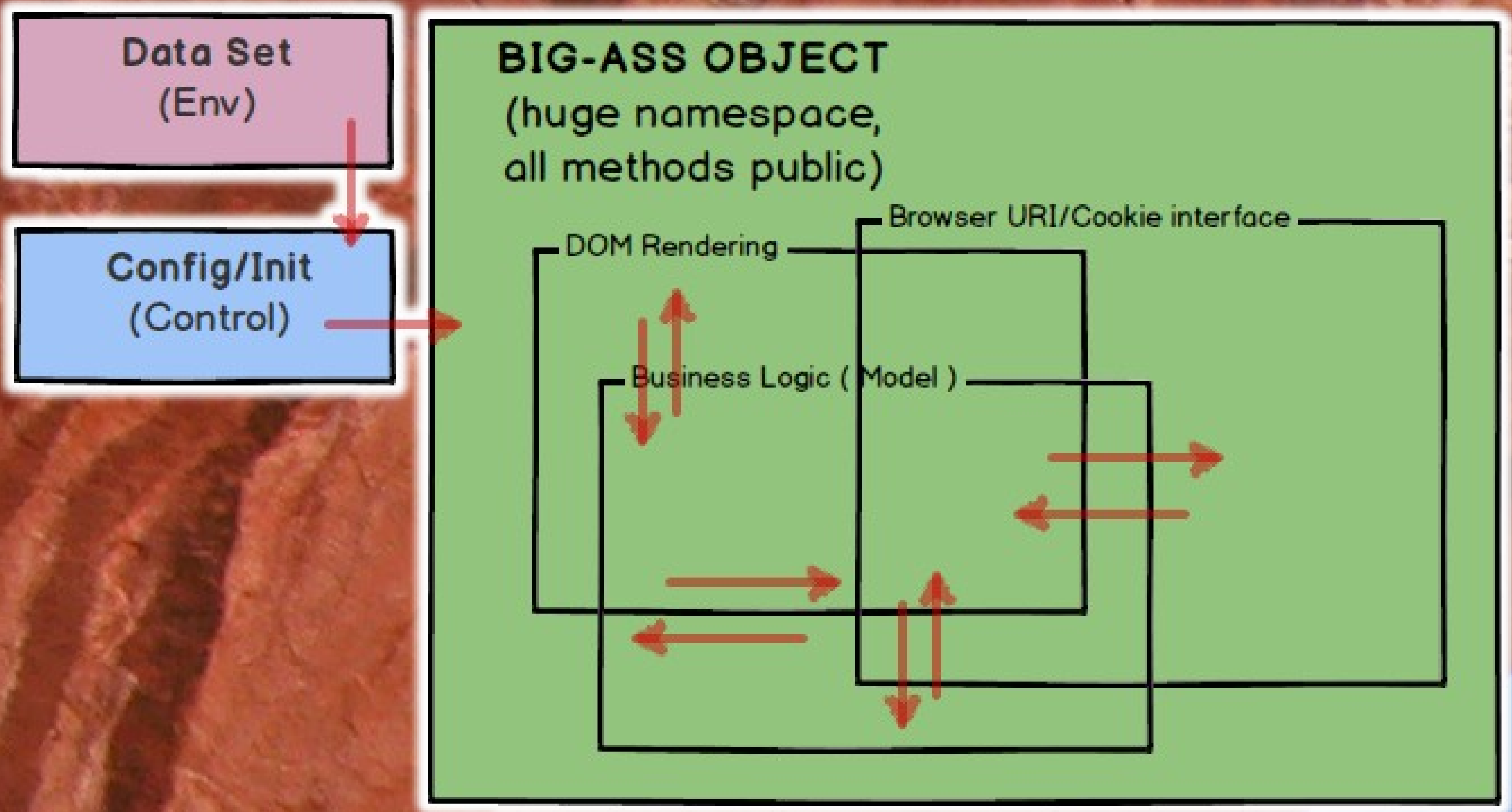
Mode 6 is like the Moon

- Interesting place to visit
- You really don't want to live there
- **… Or fix your bugs there**

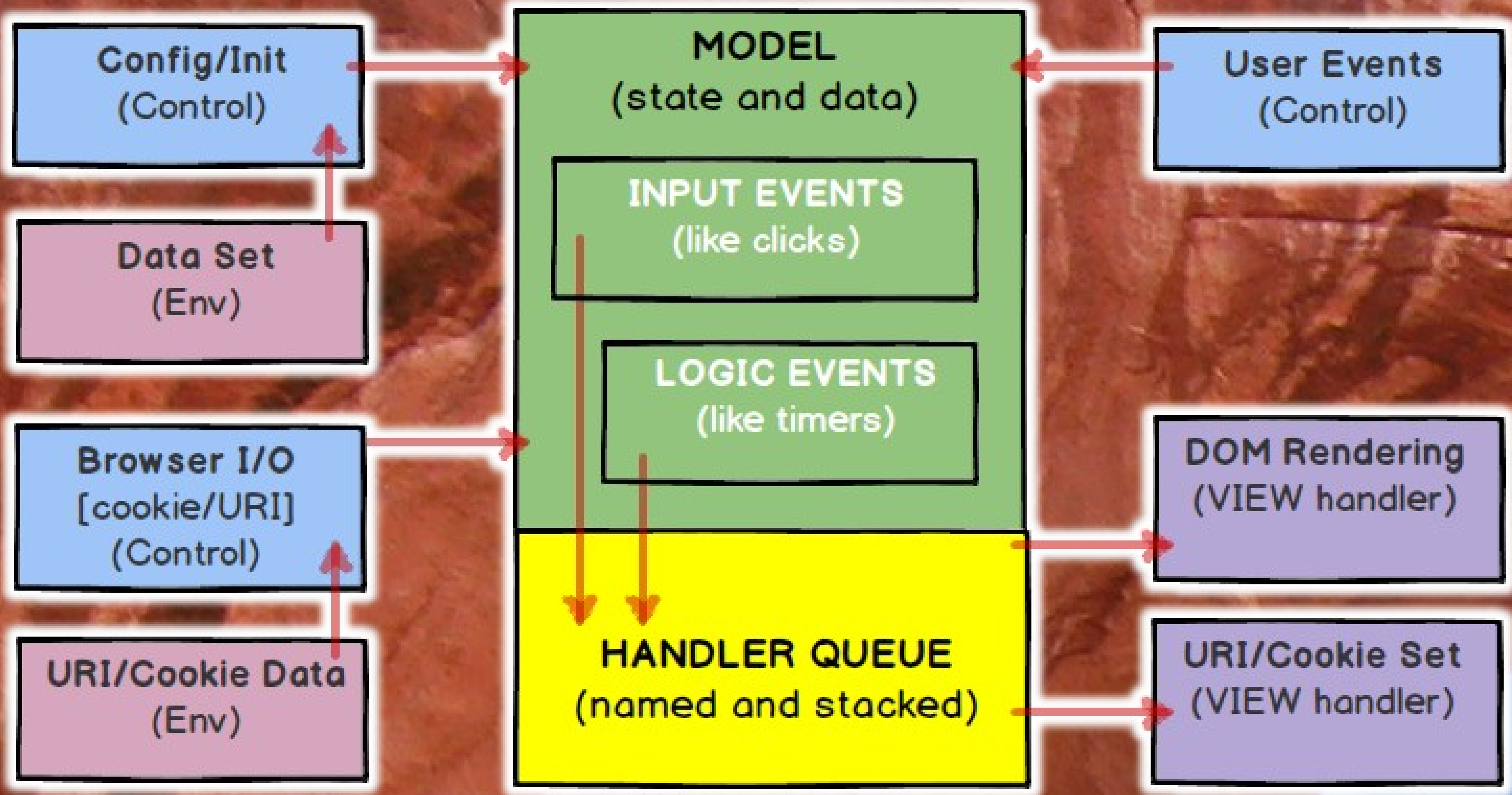
Testing using handlers

- Similar to **Apache** API - state model sends data through **handlers**
- Usually handlers do I/O with real-world entities
  - Present information to users
  - Write cookies to store state
  - Send data back to browser
- Like Apache, we can “stack” handlers
- Run application like production, but *also* test
- “Remora code”

We went from this...



To this


Config/Init
(Control)
Data Set
(Env)
Browser I/O
[cookie/URI]
(Control)
URI/Cookie Data
(Env)
MODEL
(state and data)
INPUT EVENTS
(like clicks)
LOGIC EVENTS
(like timers)
HANDLER QUEUE
(named and stacked)
User Events
(Control)
DOM Rendering
(VIEW handler)
URI/Cookie Set
(VIEW handler)

Get started with nodeunit

- Install NodeJS

```
sudo install -g nodeunit
```

- A nodeunit suite is nodejs executable JS file
- Manifest outlines tests to run in-order

```
  module.exports = {
      testOne   : testOne,
      testTwo   : testTwo,
      testThree : testThree
    };
```

Get started with nodeunit (2)

- Each test runs in order and receives a test object
- **`test_obj.expect(<number>)`** starts test
- Followed by any number of assertions using test_obj, like **`test_obj.ok( true === true, 'truth prevails!' );`**
- Test is concluded by **`test_obj.done();`**
- Promises or other closures can be used to ensure processes are complete.

Demo 2: Build a test

- Build a regression test
- Use emulator
- Use local data collector
- Use fake data
- Use handlers
- Test NPS value of “nothing”
- Place results into unit tests

Related tools

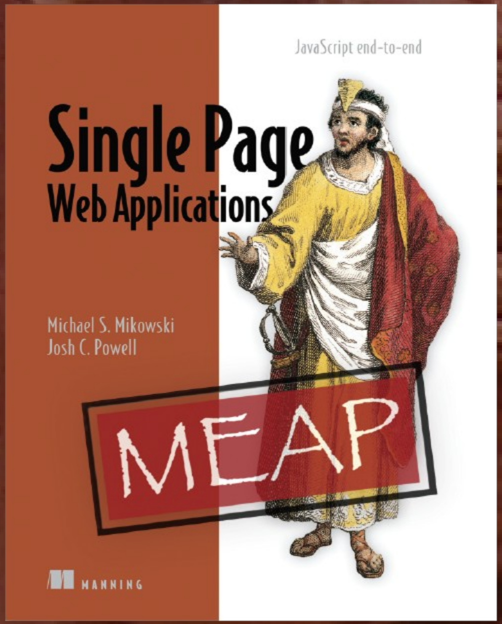
- Use a good code standard
- Use a code-review tool
  - No review, no merge to master (release-intent)
  - "Fresh eyes" prevent silly or obvious mistakes
  - Peer review encourages developer quality
  - All developers become familiar with all code
  - Developers learn from each other
  - Easiest to start “fresh”
- Use a commit hook to ensure all JavaScript passes JSLint (note JSON comment deal)

Parting thoughts

- Put as much into the model as possible
- Constantly refactor to encompass state
- Reduce problems to simplest test mode
- Solving problems in integration testing mode is like using “notepad”
- Consider using the stacked handler technique for testing and reporting
- Use code reviews and JSLint commit hook

Thank You

- 50% off book - Use code dotd1022au at www.manning.com/mikowski/
- Questions?



Make your SPA rock (solid)

Designing and testing SPAs for performance and quality

Single Page Web Applications
http://manning.com/mikowski