Agile Engineering for the Web

Welcome! Help keep things moving by setting up now:

1. Install Git: http://git-scm.com/downloads
2. Install Node.js: http://nodejs.org/download/
3. Download source code: Open command prompt and run

   git clone https://github.com/jamesshore/how_to_tabs

4. When you're done, help your neighbor!

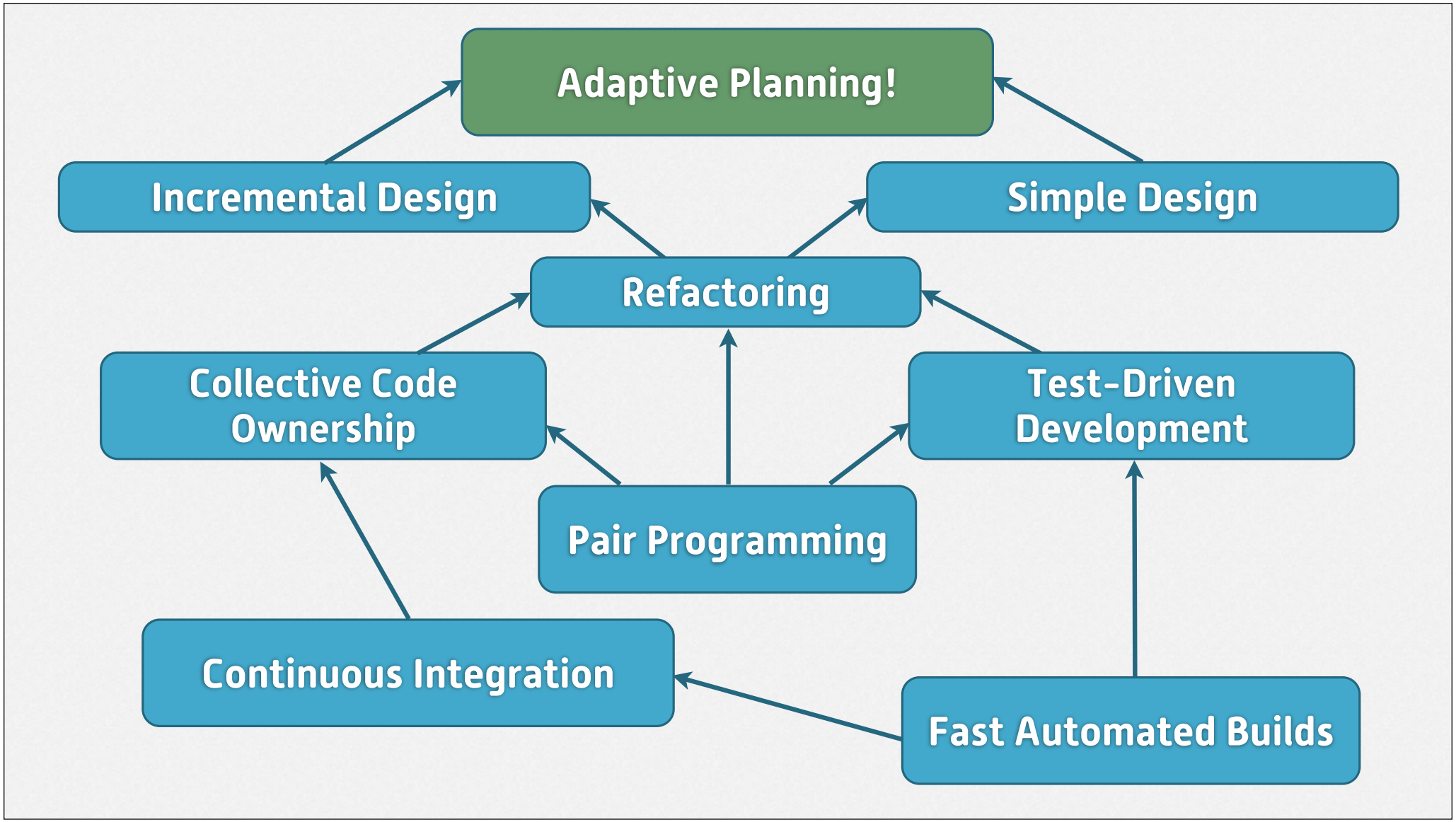
Agile Engineering for the Web

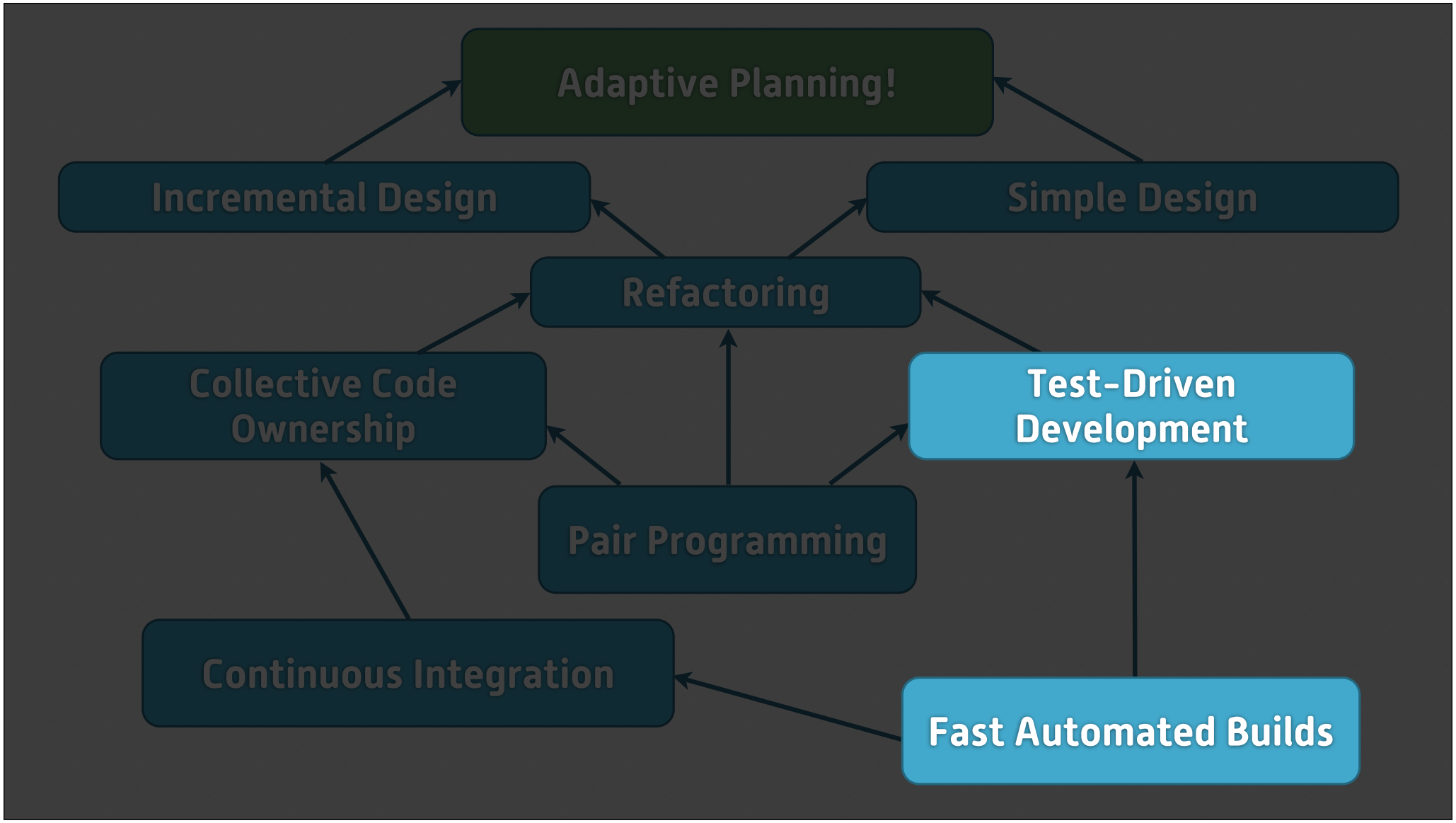
PRESENTED BY

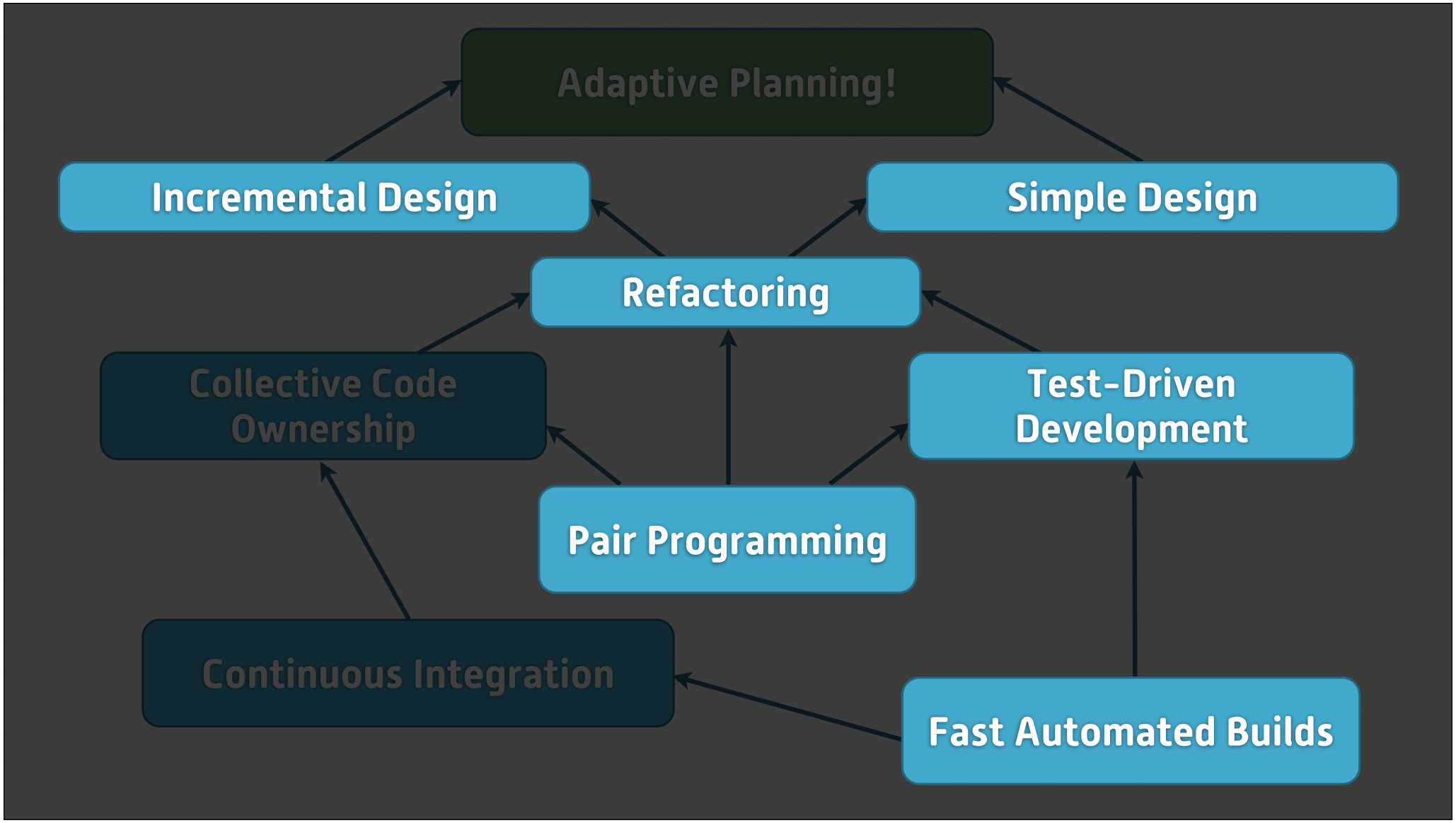
James Shore

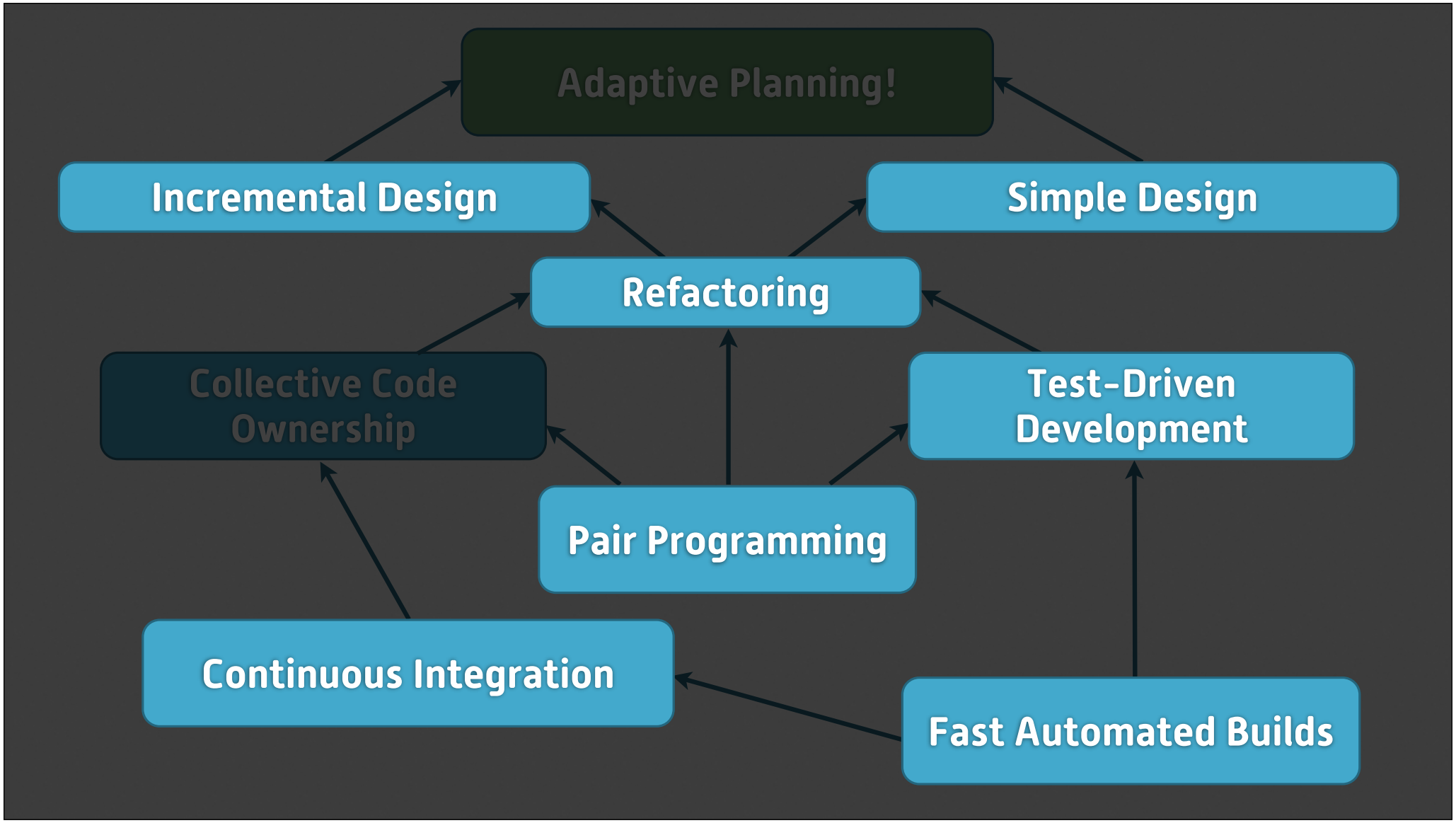
SCREENCAST: letscodejavascript.com
TWITTER: @jamesshore
EMAIL: jshore@jamesshore.com

Agile India 2016
Bangalore, India
14 March 2016


Adaptive Planning!
Incremental Design
Simple Design
Refactoring
Collective Code Ownership
Test-Driven Development
Pair Programming
Continuous Integration
Fast Automated Builds


Adaptive Planning!
Incremental Design
Simple Design
Refactoring
Collective Code Ownership
Test-Driven Development
Pair Programming
Continuous Integration
Fast Automated Builds


Adaptive Planning!
Incremental Design
Simple Design
Refactoring
Collective Code Ownership
Test-Driven Development
Pair Programming
Continuous Integration
Fast Automated Builds


Adaptive Planning!
Incremental Design
Simple Design
Refactoring
Collective Code Ownership
Test-Driven Development
Pair Programming
Continuous Integration
Fast Automated Builds

1 The Reproducible Build

Automated Builds

- Custom task documentation
- Custom command-line parameters
- Dependency resolution
  (preferably incremental)
- Write arbitrary code without plugins
  (preferably in standard JavaScript)
- Easily run command-line tools
  (or a rich plug-in ecosystem)
- Straightforward and simple

Grunt

- ~~Custom~~ task documentation
- Custom command-line parameters
- Dependency resolution
  ~~(preferably incremental)~~
- Write arbitrary code without plugins
  (preferably in ~~standard~~ JavaScript)
- ~~Easily~~ run command-line tools
  (or a rich plug-in ecosystem)
- Straightforward and simple

Gulp

- ~~Custom task documentation~~
- ~~Custom command-line parameters~~
- Dependency resolution
  ~~(preferably incremental)~~
- Write arbitrary code without plugins
  (preferably in standard JavaScript)
- ~~Easily~~ run command-line tools
  (or a rich plug-in ecosystem)
- ~~Straightforward and simple~~

npm

- ~~Custom task documentation~~
- ~~Custom command-line parameters~~
- ~~Dependency resolution~~
  ~~(preferably incremental)~~
- ~~Write arbitrary code without plugins~~
  ~~(preferably in standard JavaScript)~~
- Easily run command-line tools
  ~~(or a rich plug-in ecosystem)~~
- Straightforward and simple

make

- ~~Custom task documentation~~
- Custom command-line parameters
- Dependency resolution
  (preferably incremental)
- Write arbitrary code without plugins
  ~~(preferably in standard JavaScript)~~
- Easily run command-line tools
  (or a rich plug-in ecosystem)
- ~~Straightforward and simple~~

Webpack

- ~~Custom task documentation~~
- ~~Custom command-line parameters~~
- ~~Dependency resolution (preferably incremental)~~
- ~~Write arbitrary code without plugins (preferably in standard JavaScript)~~
- ~~Easily run command-line tools~~ (or a rich plug-in ecosystem)
- Straightforward and simple

Jake

- Custom task documentation
- Custom command-line parameters
- Dependency resolution
  (preferably incremental)
- Write arbitrary code without plugins
  (preferably in standard JavaScript)
- Easily run command-line tools
  ~~(or a rich plug-in ecosystem)~~
- Straightforward and simple

What Should the Build Do?

- **Lint: JSHint**, JSLint, ESLint, etc.
- **Cross-browser test: Karma**, Test'em
- **Integration test: Selenium WebdriverJS**, PhantomJS, CasperJS
- **Create distribution package: Browserify, Webpack**
- **Run localhost server: http-server**

2 Continuous Integration

Continuous Integration

1. **Integrate Frequently**
2. **Integrated Code is Known-Good**

The Integration Process

1. Integrate and build locally
   - if it fails: fix integration conflict
2. Build on separate machine
   - if it fails: fix environmental issue
3. Publish known-good code

Doing It Wrong

1. Publish unknown-quality code
2. “CI” tool builds the code
   if it fails: get an email
3. Blame the last person to check in
4. Go to lunch

CI: My Recommendation

- **Start the build manually** (yes, really)
- **Launch browsers manually**
- **Use a simple shell script to publish** known-good code (e.g., git branches)

https://github.com/jamesshore/automatopia

Now that I've alienated everyone

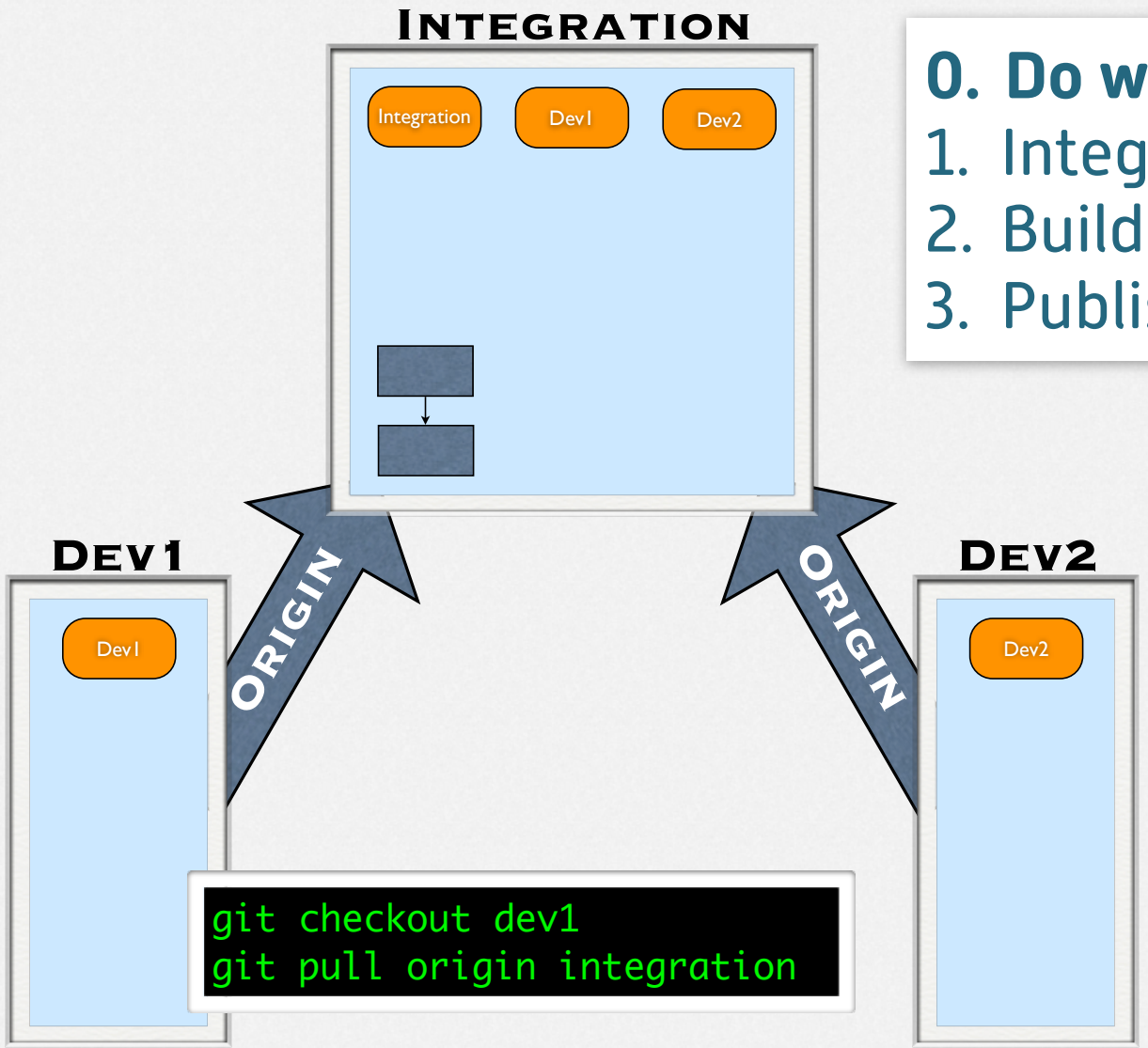
A build server can be useful for...

- Slow builds
- Third-party cross-browser testing

Challenge: Continuous Integration

Design a process for manual continuous integration that results in guaranteed-good builds. Assume a central "CI" repo and machine are available for testing.

1. What branches do you set up?
2. Which shell commands do you type?



0. **Do work**
1. Integrate and build locally
2. Build on separate machine
3. Publish known-good code

3 Lint

4 Cross-Browser Testing

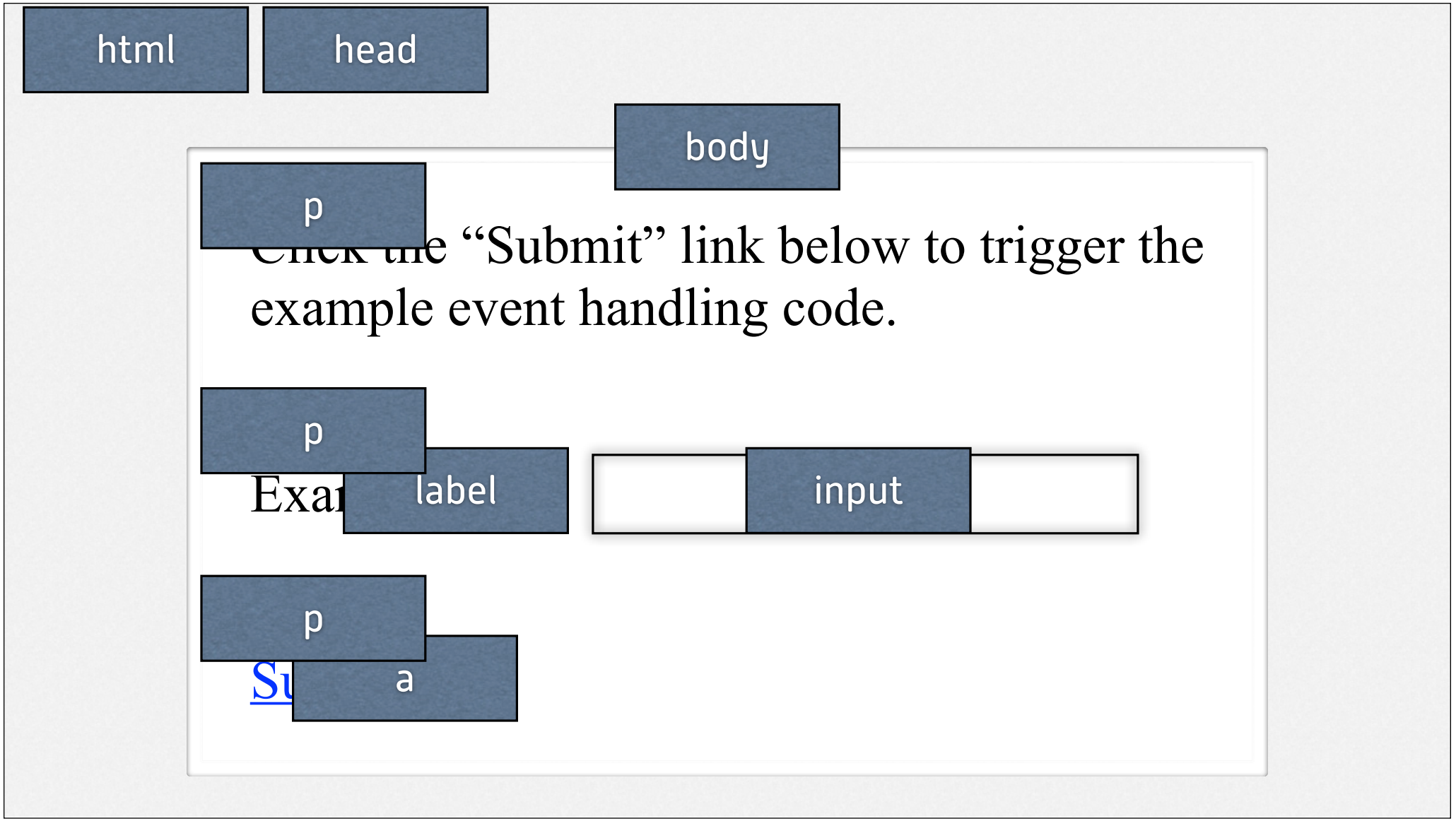
5 Front-End Modules

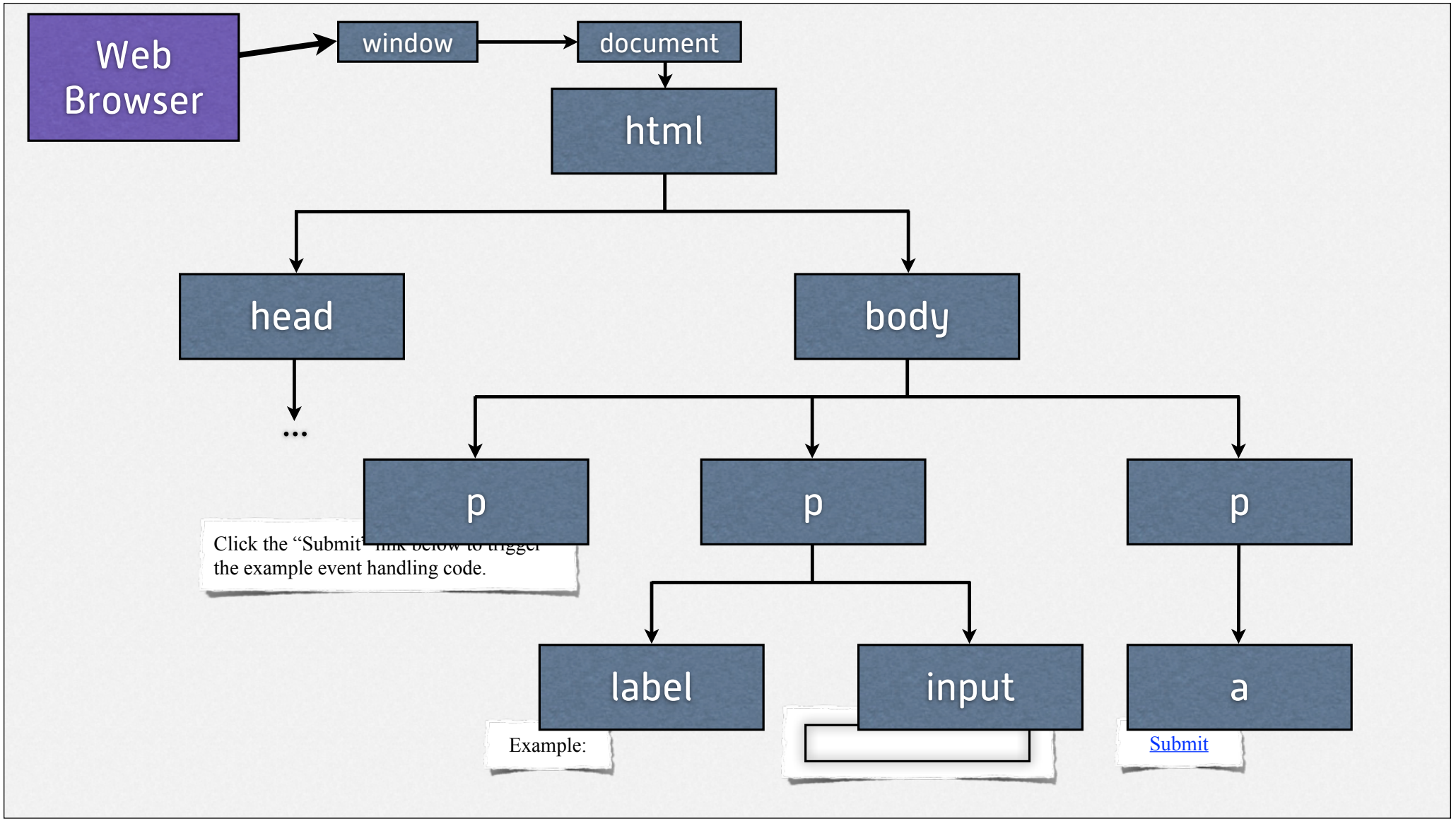
6 The DOM

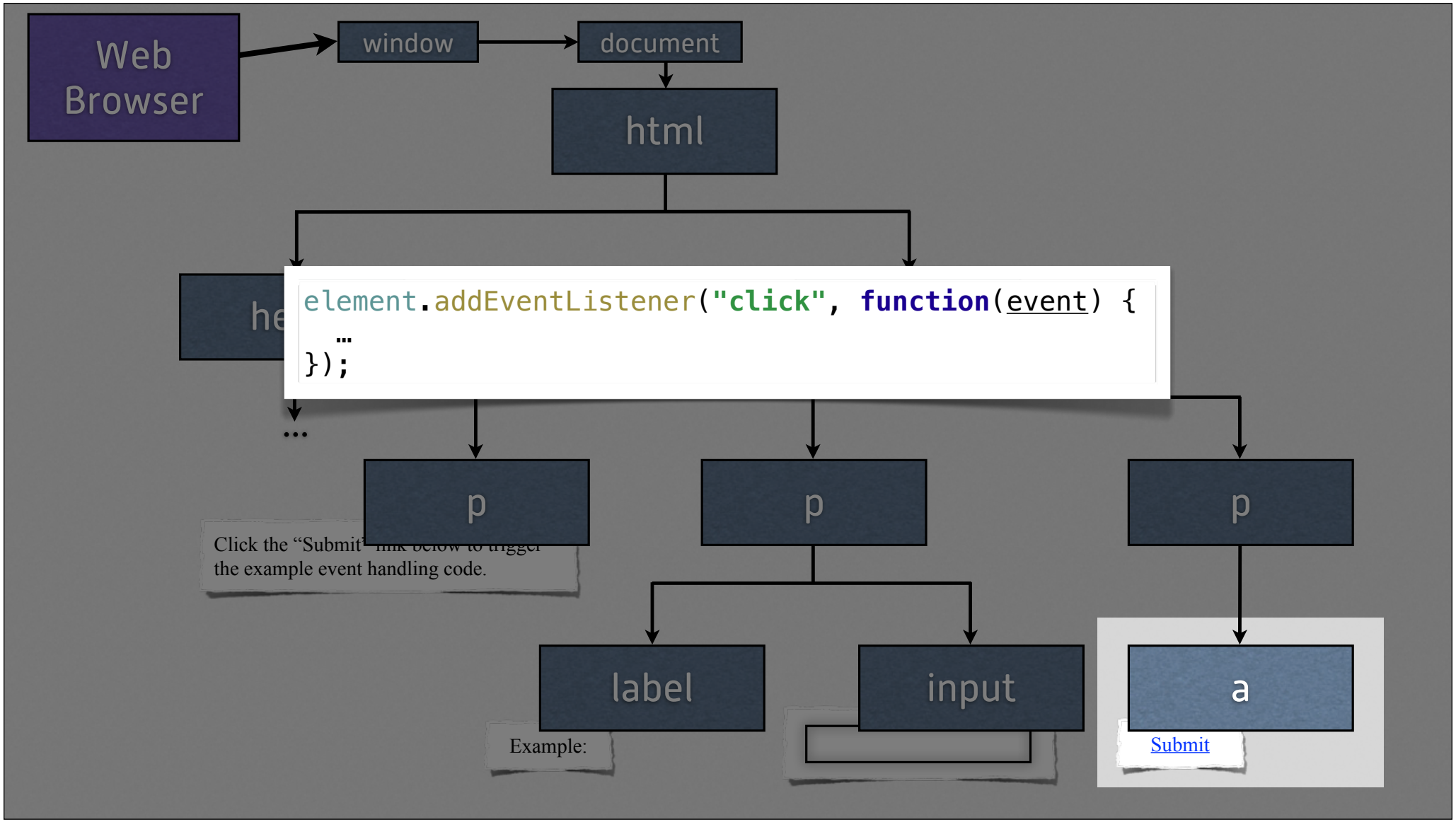
Click the “Submit” link below to trigger the example event handling code.

Example field:

Submit


html
head
body
p
Click the “Submit” link below to trigger the example event handling code.
p
label
input
p
a


Web Browser
window
document
html
head
body
...
p
p
p
Click the "Submit" link below to trigger the example event handling code.
label
input
a
Example:
Submit


Web Browser
window
document
html
he
element.addEventListener("click", function(event) {
…
});
…
p
p
p
Click the "Submit" link below to trigger the example event handling code.
label
input
a
Example:
Submit

DOM Events Summary

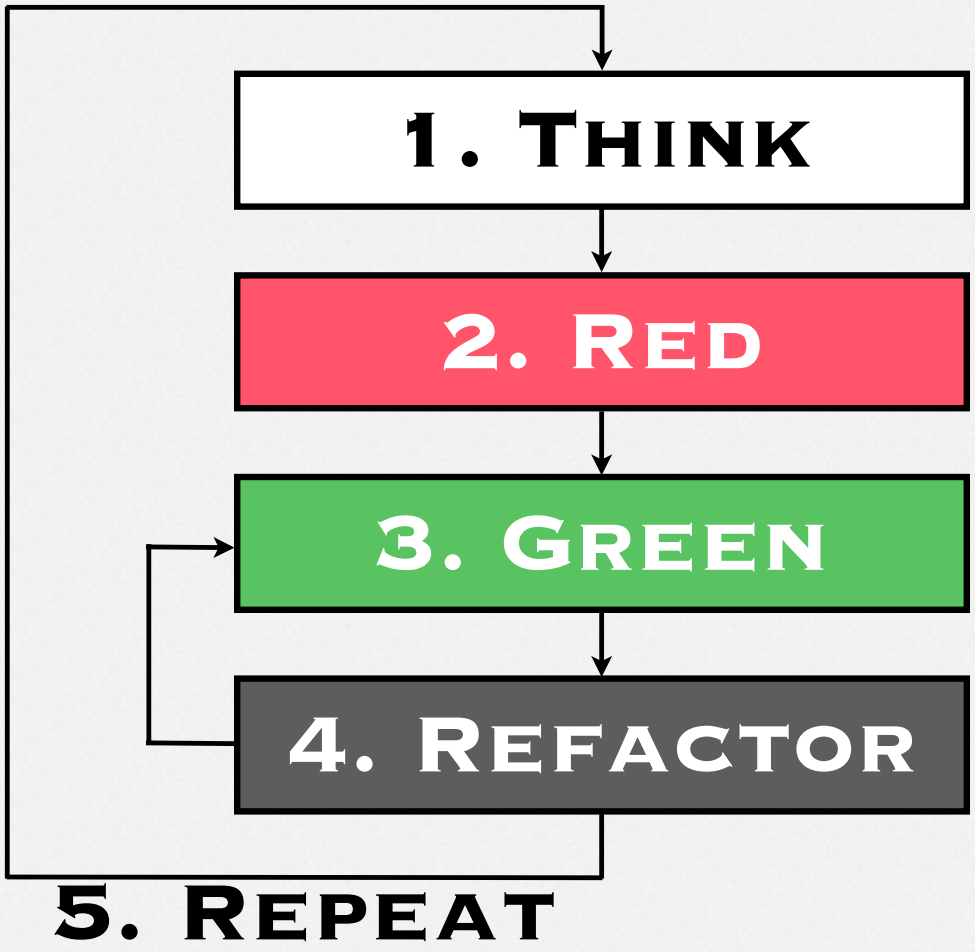
When an event is triggered...

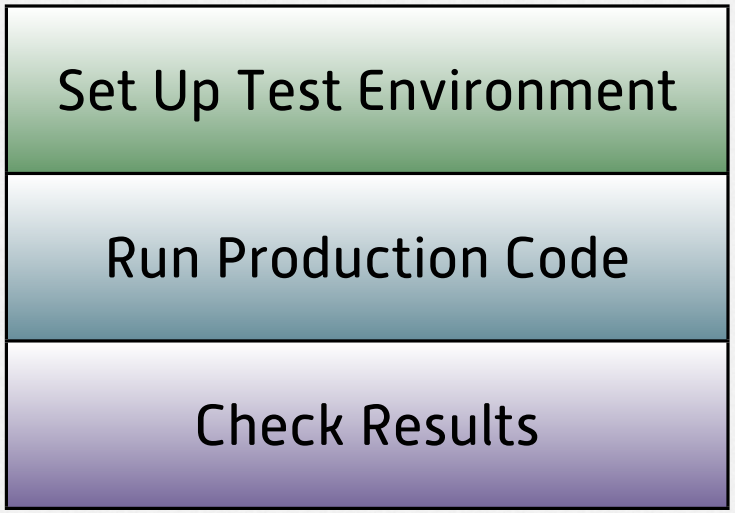
A. Event Handlers execute: CANCEL WITH event.stopPropagation()

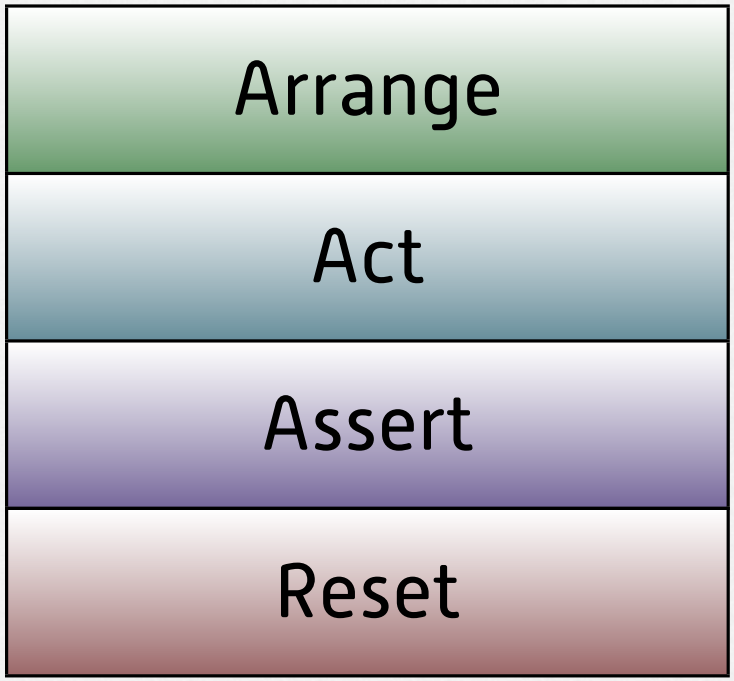
1. Capturing Phase — EVENT PROPAGATES DOWNWARD THROUGH DOM
2. Target Phase — EVENT IS AT TARGET ELEMENT
3. Bubbling Phase — EVENT PROPAGATES BACK UP THROUGH DOM

B. Default Action occurs. CANCEL WITH event.preventDefault()

7 Test-Driven Development


1. Think
2. Red
3. Green
4. Refactor
5. Repeat


Set Up Test Environment
Run Production Code
Check Results



// CREATE DOM ELEMENTS

// RUN PRODUCTION CODE

// CHECK DOM ELEMENTS

// ERASE DOM ELEMENTS

Agile Engineering for the Web

PRESENTED BY

James Shore

SCREENCAST: letscodejavascript.com
TWITTER: @jamesshore
EMAIL: jshore@jamesshore.com

Agile India 2016
Bangalore, India
14 March 2016