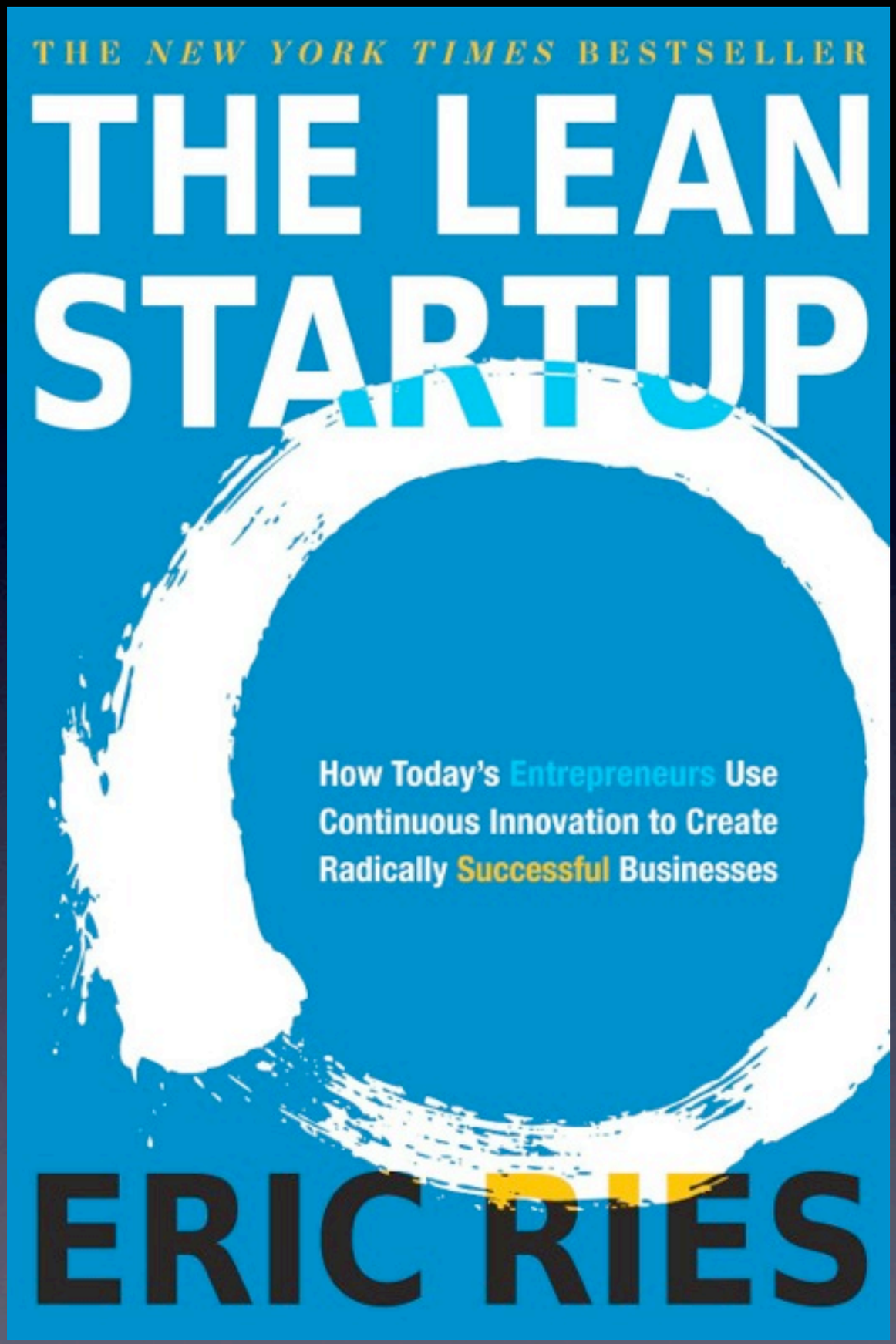
Deep Dish

Chicago-Style Functional Testing

Erlang User Conference
Stockholm, Sweden
June 14, 2013


THE NEW YORK TIMES BESTSELLER
THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Continuous Integration

Continuous Integration

- Test Ideas on Customers Quickly

Continuous Integration

- Test Ideas on Customers Quickly
- Eliminate Feature Inventory

Continuous Integration

- Test Ideas on Customers Quickly
- Eliminate Feature Inventory
- Make Your Engineers Happier

Continuous Integration

- Unit tests for functions and modules
- Functional tests for applications
- Integration tests for multiple applications
- Immune system for online services

Continuous Integration

Theory	Practice

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	Run A Few Tests, Then Commit

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	Run A Few Tests, Then Commit
Deliver Features Several Times Per Day	

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	Run A Few Tests, Then Commit
Deliver Features Several Times Per Day	Break The Build Several Times Per Day

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	Run A Few Tests, Then Commit
Deliver Features Several Times Per Day	Break The Build Several Times Per Day
Eliminate Need For QA	

Continuous Integration

Theory	Practice
Run All Tests, Then Commit	Run A Few Tests, Then Commit
Deliver Features Several Times Per Day	Break The Build Several Times Per Day
Eliminate Need For QA	You Still Need QA

Functional Tests Are Slow

Why Are Functional Tests Slow?

Why Are Functional Tests Slow?

- Lots of Repeated Logic

Why Are Functional Tests Slow?

- Lots of Repeated Logic
- UI Code Is Slow

Why Are Functional Tests Slow?

- Lots of Repeated Logic
- UI Code Is Slow
- Drawing Is Slow

Why Are Functional Tests Slow?

- Lots of Repeated Logic
- UI Code Is Slow
- Drawing Is Slow
- Generating HTML Is Slow

Why Are Functional Tests Slow?

- Lots of Repeated Logic
- UI Code Is Slow
- Drawing Is Slow
- Generating HTML Is Slow
- Event Loops Impose Overhead

Erlang Can Help

Problem	Erlang's Solution

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	I/O Lists and Shared Binaries

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	I/O Lists and Shared Binaries
Event Loops Are Slow	

Erlang Can Help

Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	I/O Lists and Shared Binaries
Event Loops Are Slow	Don't Use JavaScript

Erlang Can Help

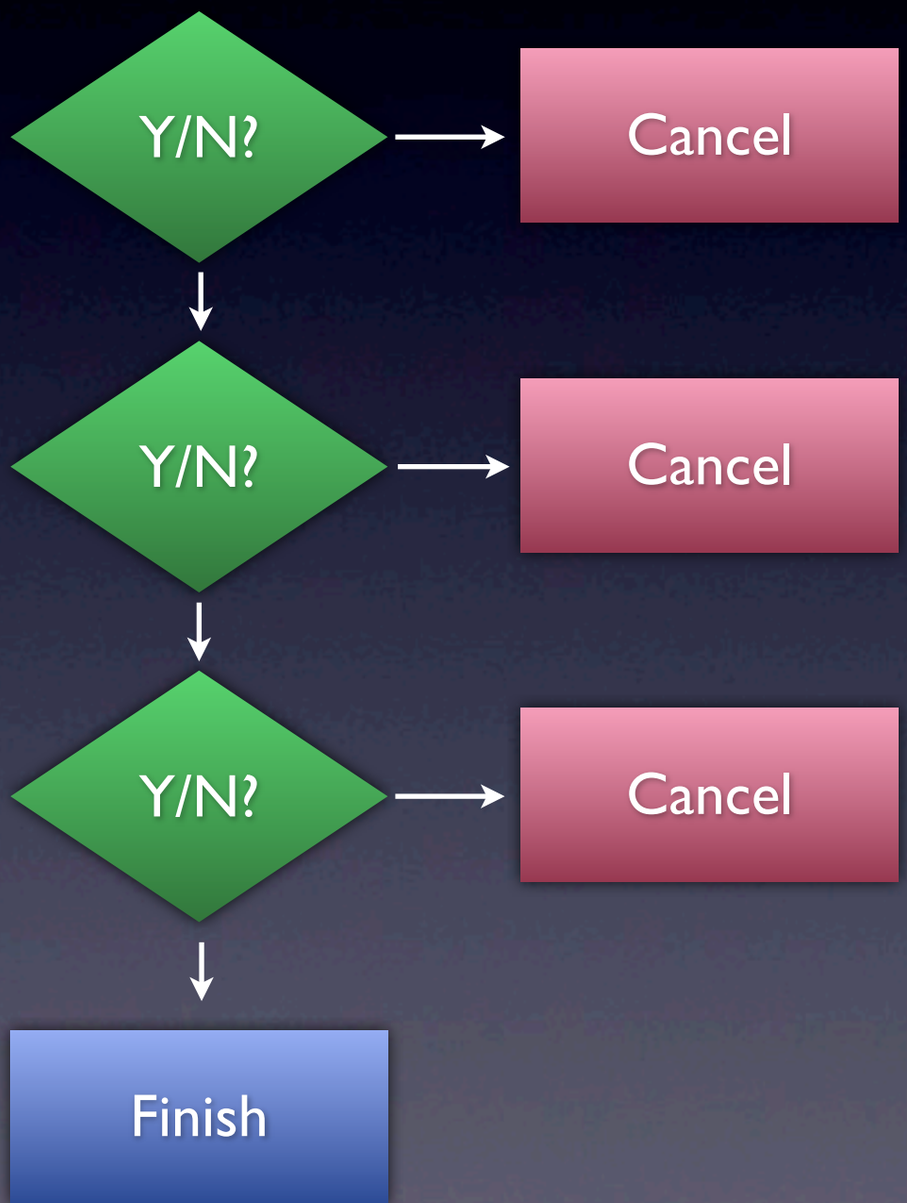
Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	I/O Lists and Shared Binaries
Event Loops Are Slow	Don't Use JavaScript
Repeated Logic	

Erlang Can Help

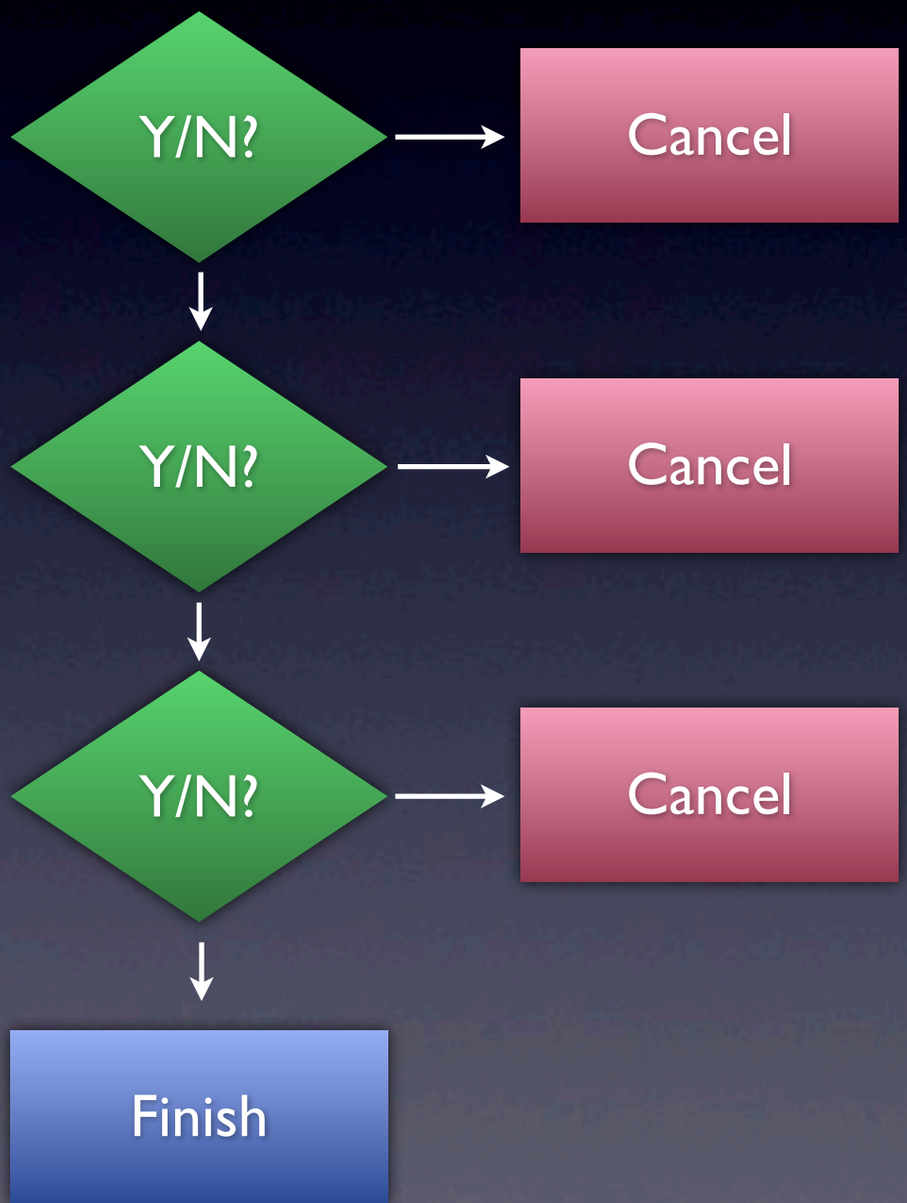
Problem	Erlang's Solution
Drawing Is Slow	No Drawing Libraries
Generating HTML Is Slow	I/O Lists and Shared Binaries
Event Loops Are Slow	Don't Use JavaScript
Repeated Logic	Deep Dish

Repeated Logic

Workflow

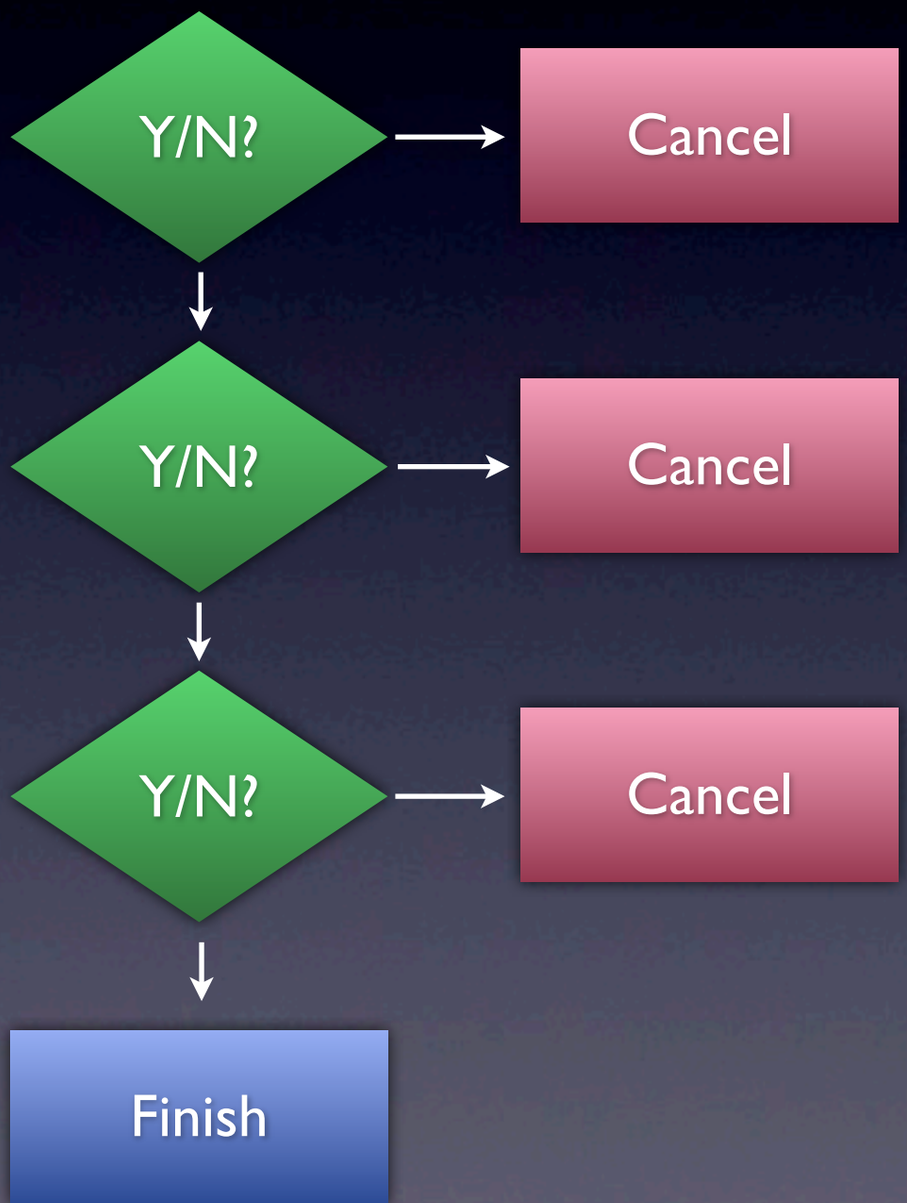


Workflow



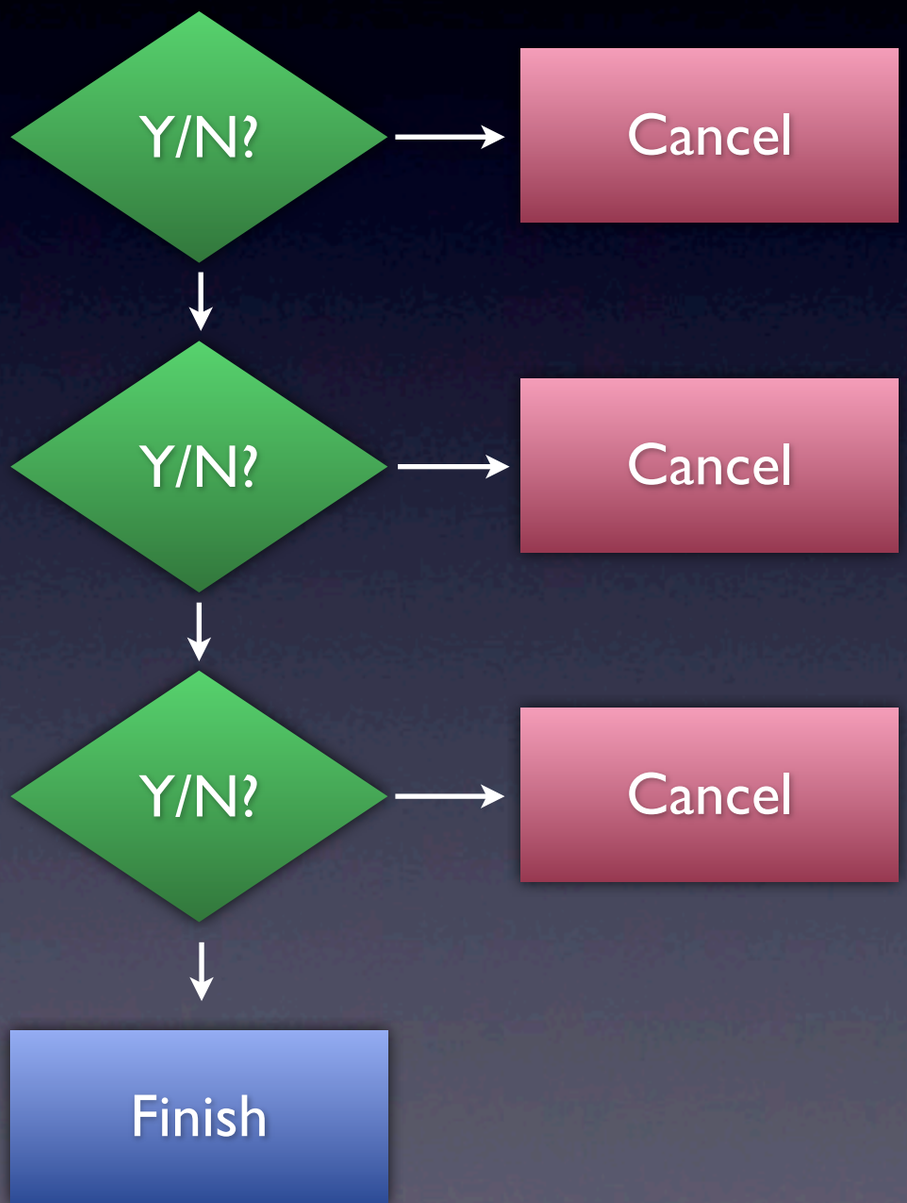
Tests

Workflow

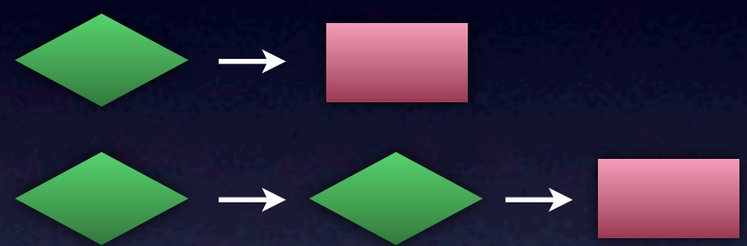


Tests

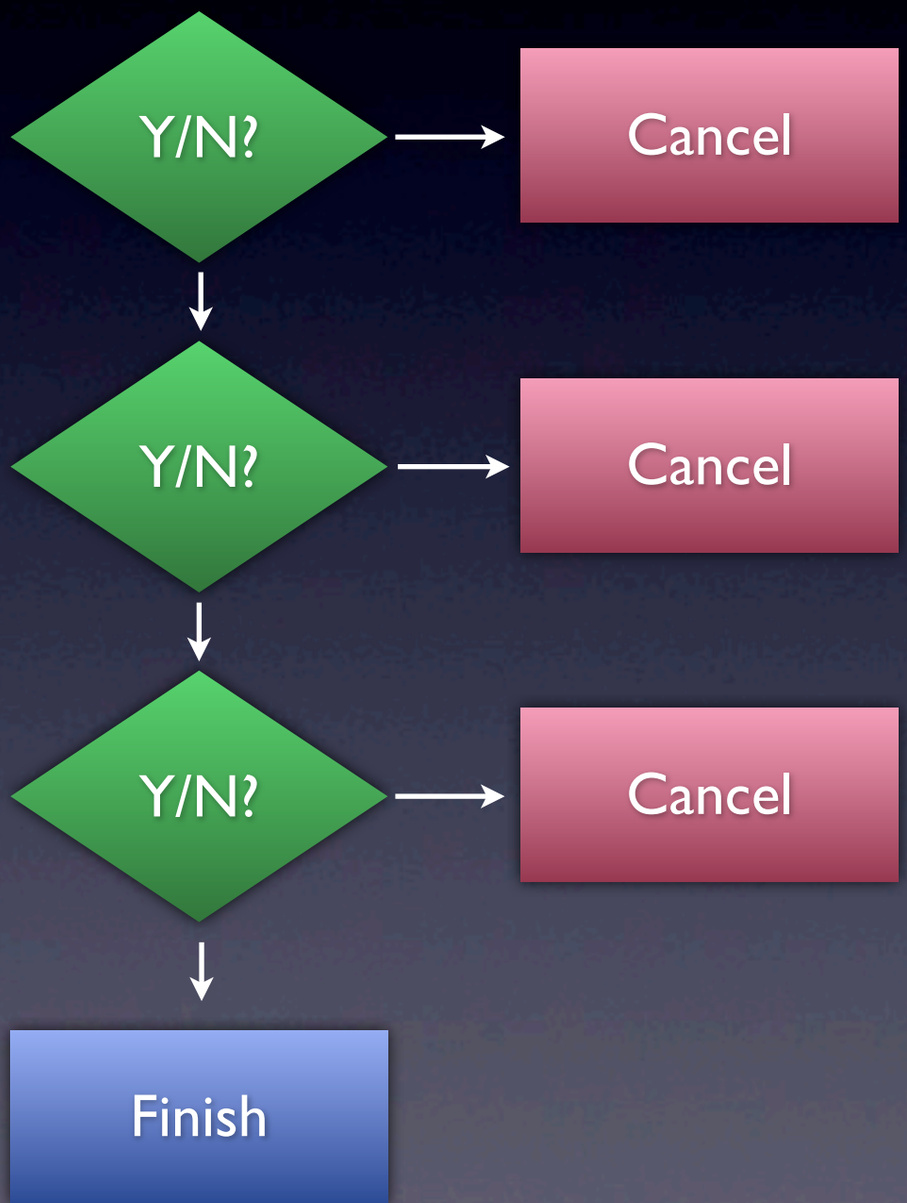
Workflow



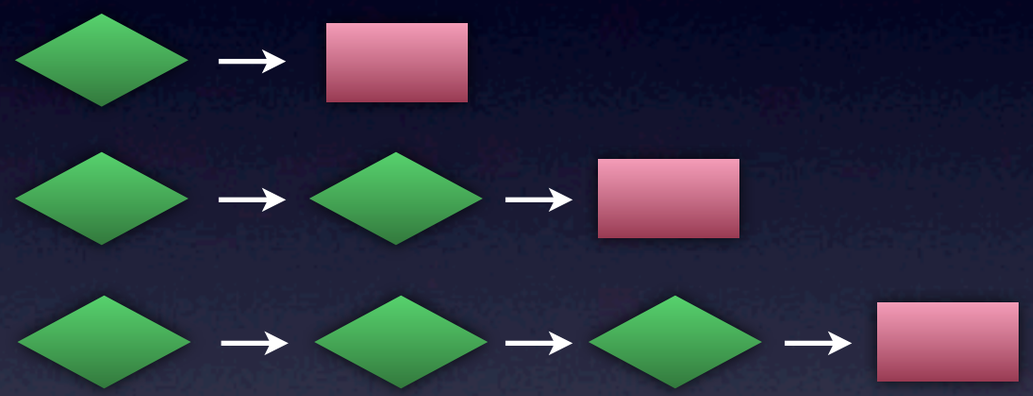
Tests



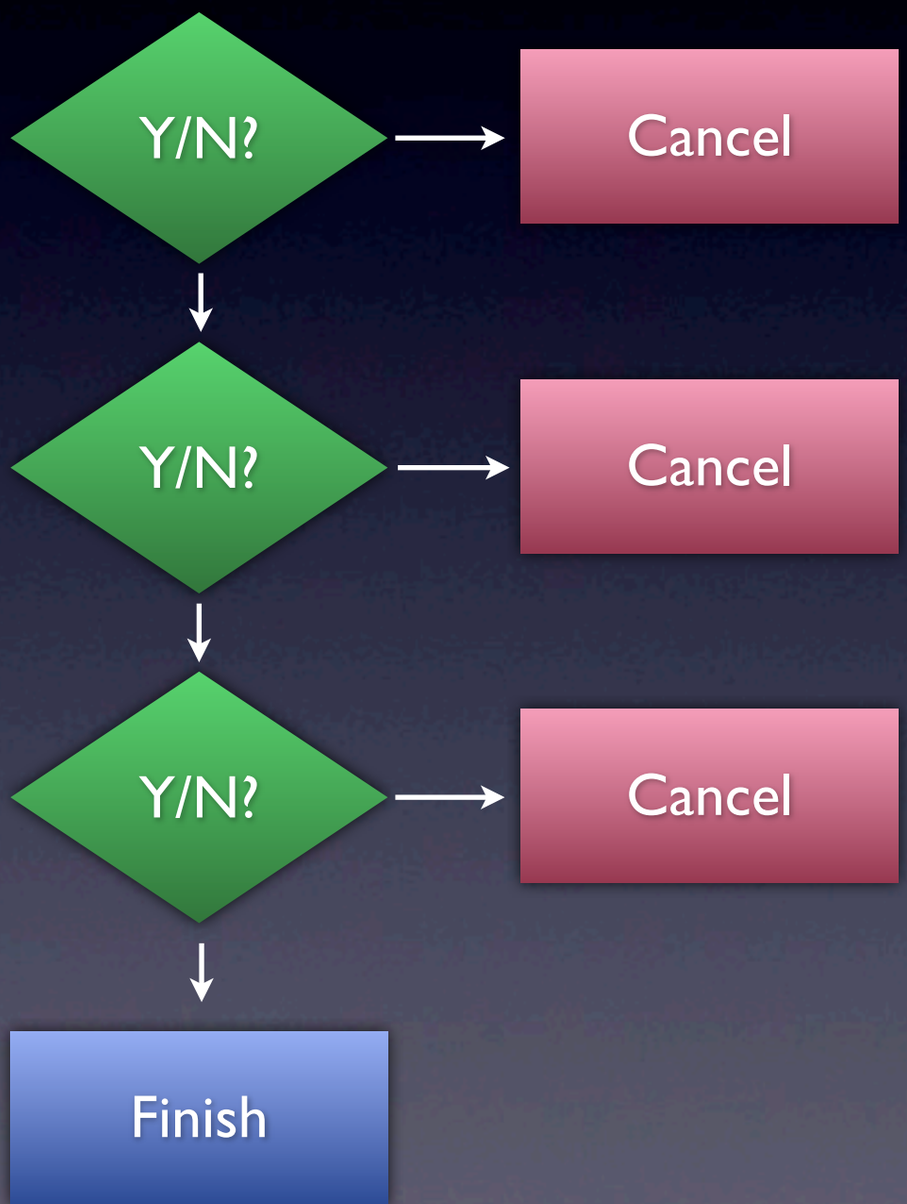
Workflow



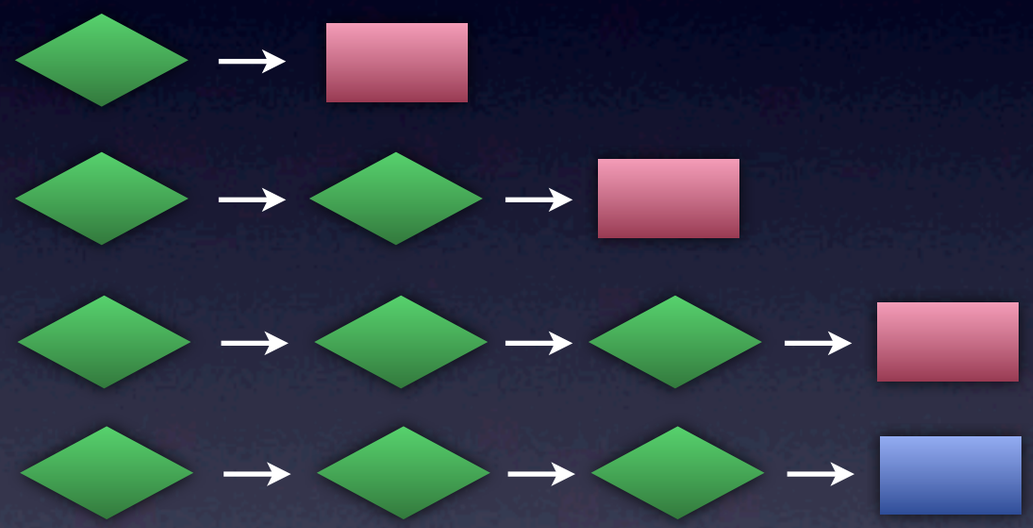
Tests



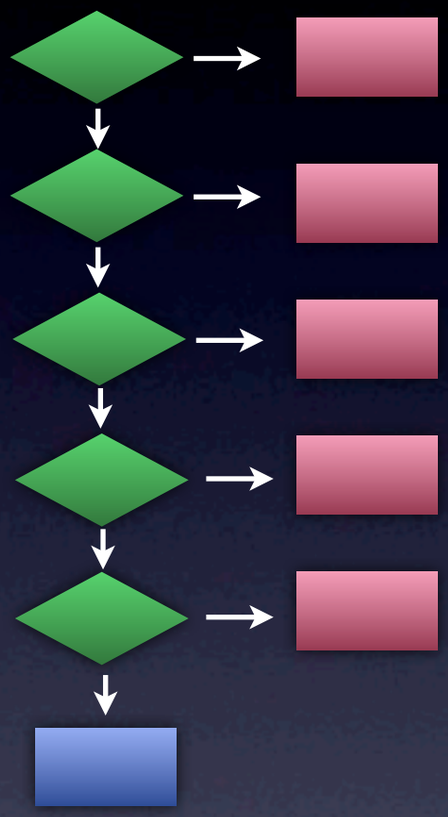
Workflow



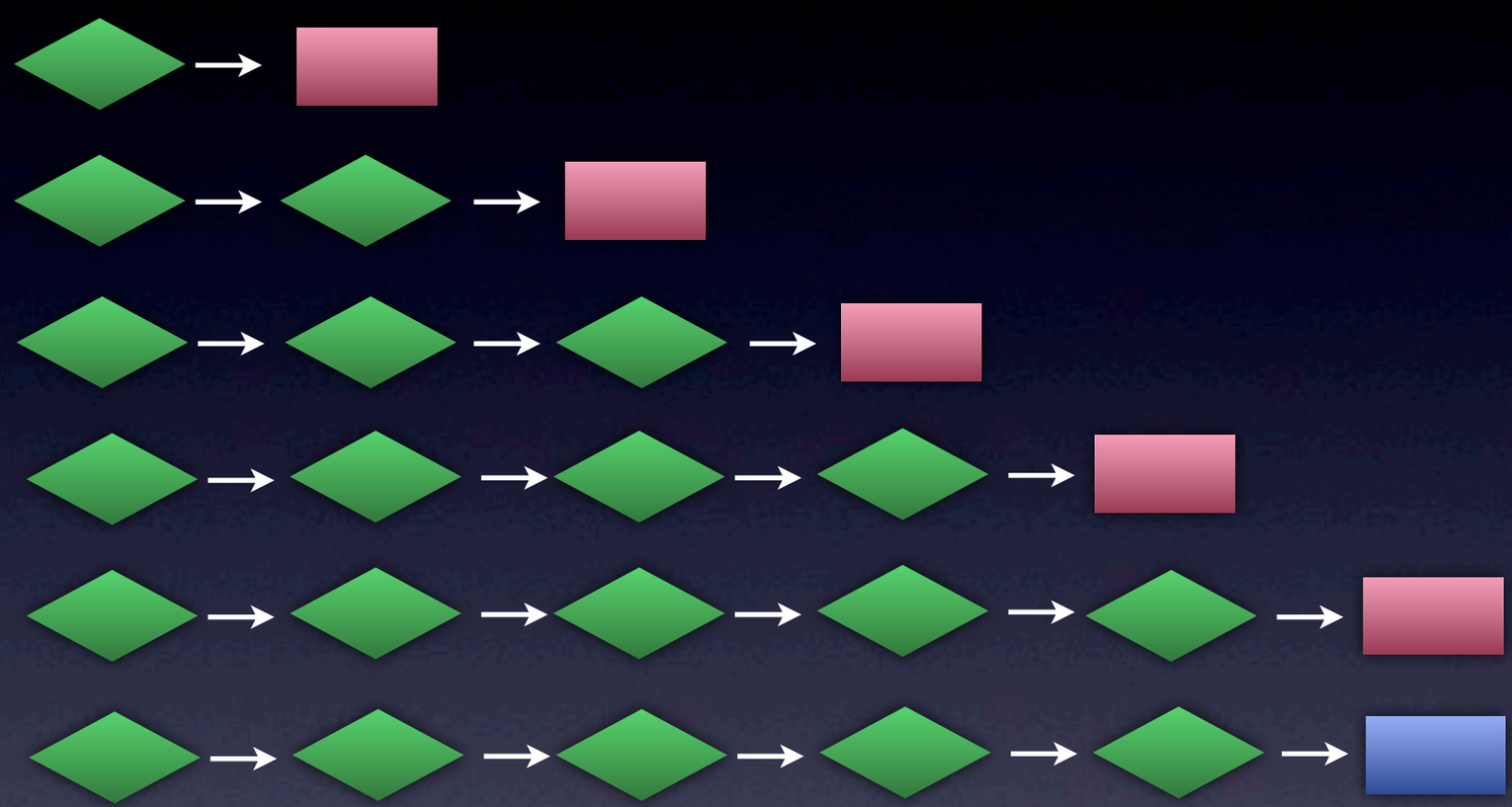
Tests



Workflow



Tests



O(n)

$O(n^2)$

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_two() {
    do_step_one("Continue");
    R = do_step_two("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_three() {
    do_step_one("Continue");
    do_step_two("Continue");
    R = do_step_three("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_four() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    R = do_step_four("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_five() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    R = do_step_five("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_six() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    R = do_step_six("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_seven() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    do_step_six("Continue");
    R = do_step_seven("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_eight() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    do_step_six("Continue");
    do_step_seven("Continue");
    R = do_step_eight("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_nine() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    do_step_six("Continue");
    do_step_seven("Continue");
    do_step_eight("Continue");
    R = do_step_nine("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_ten() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    do_step_six("Continue");
    do_step_seven("Continue");
    do_step_eight("Continue");
    do_step_nine("Continue");
    R = do_step_ten("Cancel");
    assert_something(R);
}
```

“Simon Says” Test Code

```
test_cancel_at_step_one() {
    R = do_step_one("Cancel");
    assert_something(R);
}
```

```
test_cancel_at_step_ten() {
    do_step_one("Continue");
    do_step_two("Continue");
    do_step_three("Continue");
    do_step_four("Continue");
    do_step_five("Continue");
    do_step_six("Continue");
    do_step_seven("Continue");
    do_step_eight("Continue");
    do_step_nine("Continue");
    R = do_step_ten("Cancel");
    assert_something(R);
}
```

Total Code Size = $O(n^2)$

Deep Dish

Maintain Stack of State

- All stateful resources should respond to *push* and *pop* messages
- *Push* copies the current (top-most) state and adds it to the stack
- *Pop* discards the current (top-most) state

Push

Pop

Maintain Stack of State

State

Output

Maintain Stack of State

State

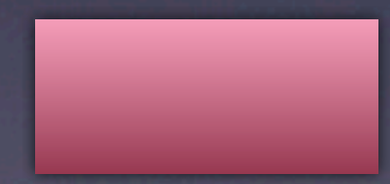
Output

```
Test 1: passed
```

Maintain Stack of State

State

Output

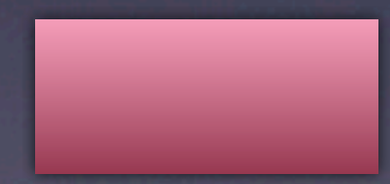


```
Test 1: passed
```

Maintain Stack of State

State

Output



```
Test 2: passed

Test 1: passed
```

Maintain Stack of State

State

Output

pop

```
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output

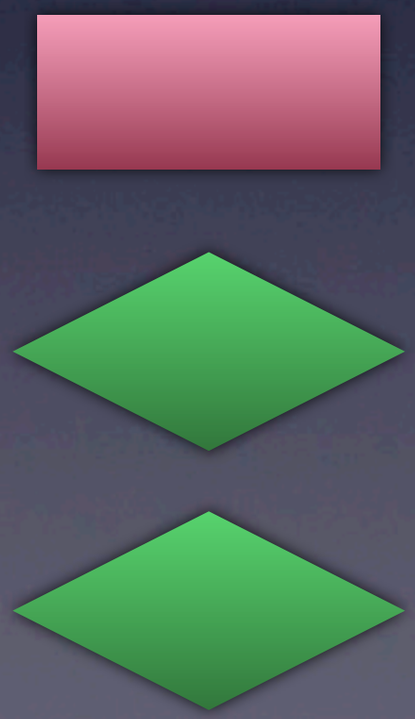
```
Test 2: passed

Test 1: passed
```

Maintain Stack of State

State

Output

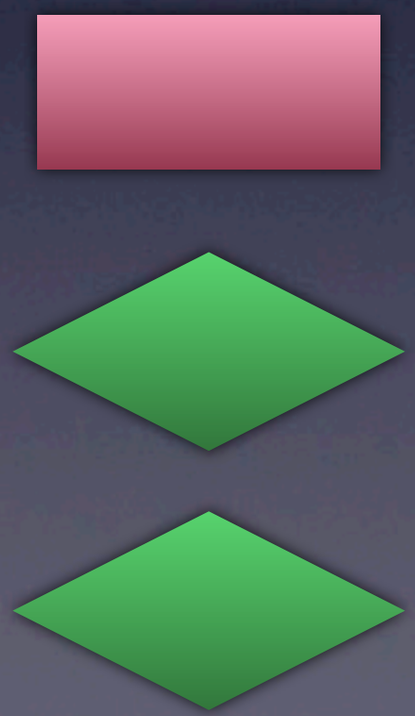


```
Test 2: passed

Test 1: passed
```

Maintain Stack of State

State



Output

```
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output

pop

```
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

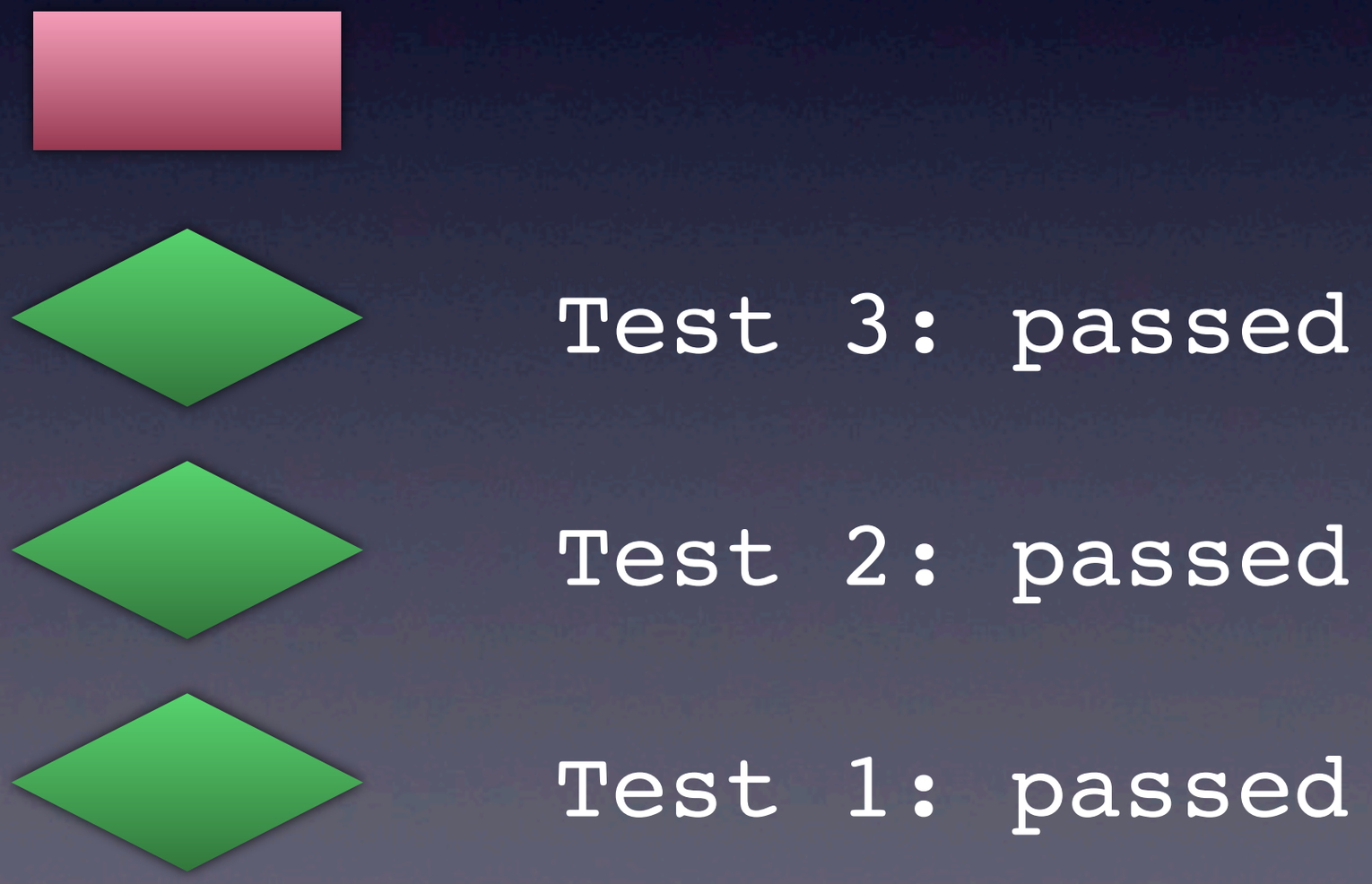
Output

```
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output



Maintain Stack of State

State

Output

```
Test 4: passed
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output

pop

```
Test 4: passed
```

```
Test 3: passed

Test 2: passed

Test 1: passed
```

Maintain Stack of State

State

Output

```
Test 4: passed
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output

```
Test 4: passed
Test 3: passed
Test 2: passed
Test 1: passed
```

Maintain Stack of State

State

Output

```
Test 5: passed

Test 4: passed

Test 3: passed

Test 2: passed

Test 1: passed
```

Maintain Stack of State

State	Output
pop	`Test 5: passed`
	`Test 4: passed`
	`Test 3: passed`
	`Test 2: passed`
	`Test 1: passed`

Maintain Stack of State

State

Output

```
Test 5: passed

Test 4: passed

Test 3: passed

Test 2: passed

Test 1: passed
```

Maintain Stack of State

State

Output

```
Test 5: passed

Test 4: passed

Test 3: passed

Test 2: passed

Test 1: passed
```

pop

Stack

Implementations

gen_server

```
handle_call(push, [State|OldState]) ->
    {noreply, [State, State|OldState]).
```

```
handle_call(pop, [State|OldState]) ->
    {noreply, OldState).
```

gen_server

```
% old and busted
handle_call(do_something, State) ->
```

```
% new hotness
handle_call(do_something, [State|OldState]) ->
```

SQL Database

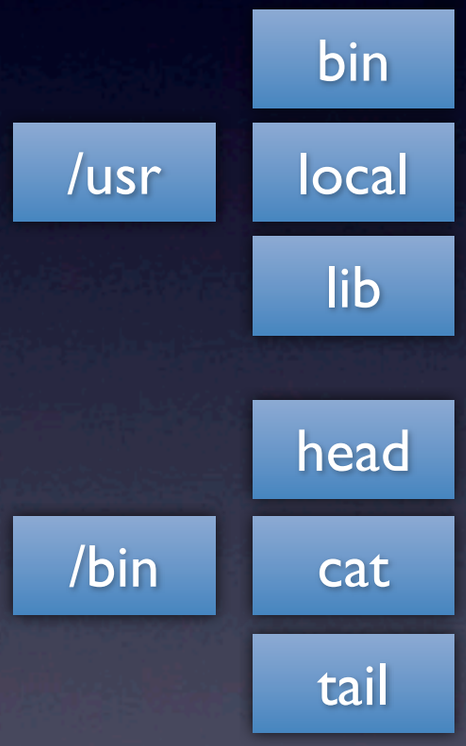
- SAVEPOINT
- ROLLBACK

Desktop Environment

- Undo
- Redo

File System

Original

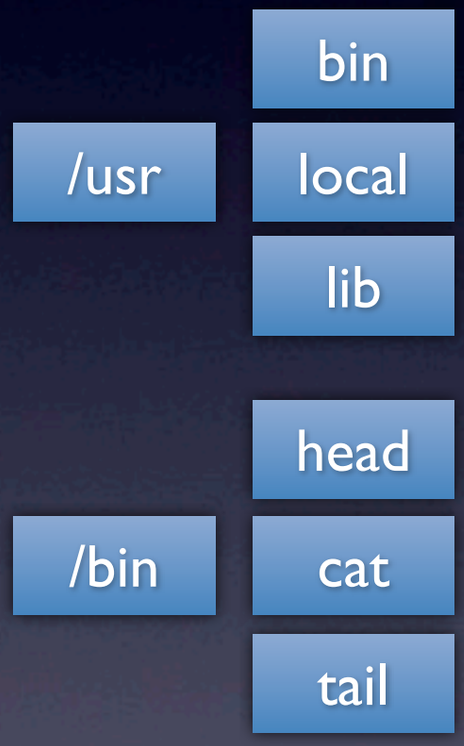


File

File System

Original

Modification

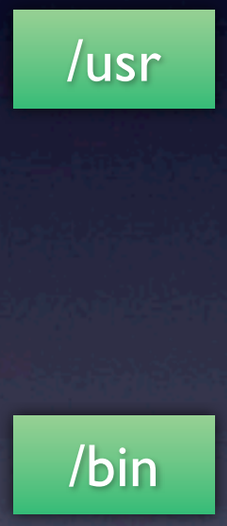


File

File System

Original

Modification

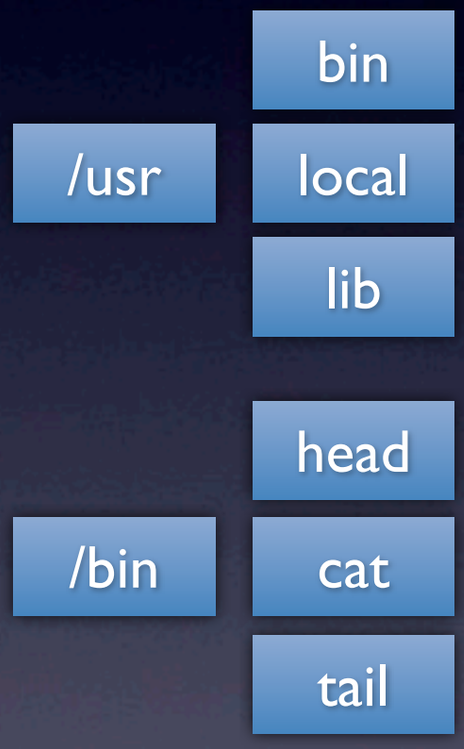


File

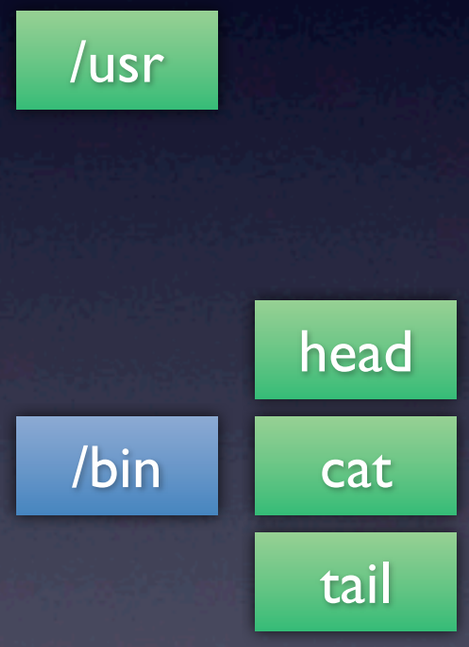
Hard Link

File System

Original



Modification

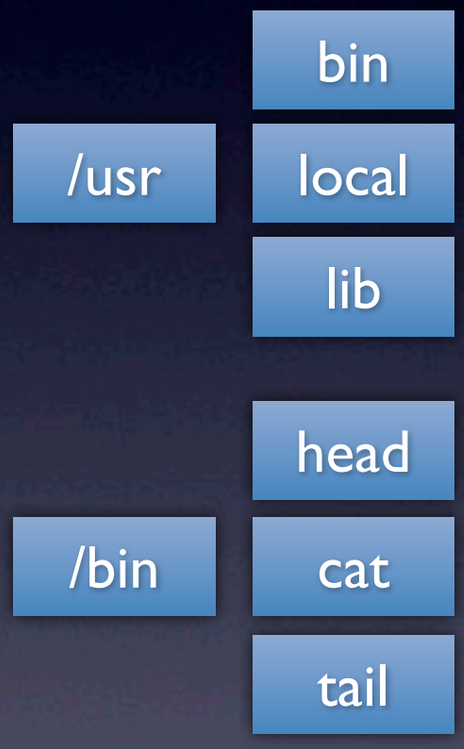


File

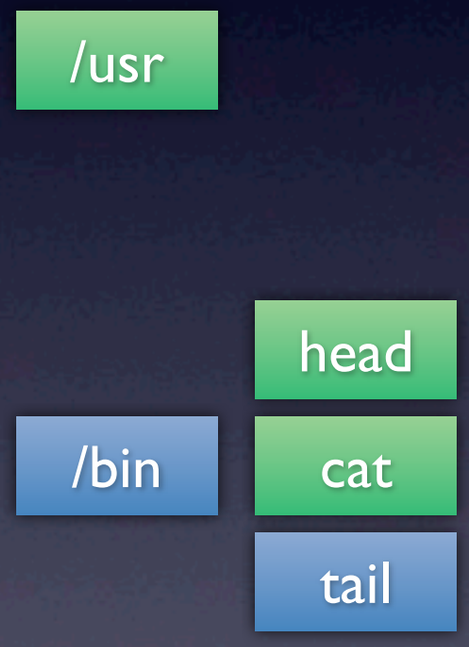
Hard Link

File System

Original



Modification



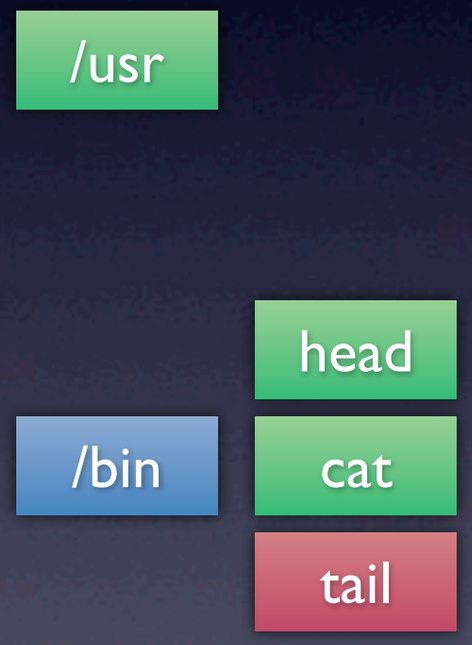
File

Hard Link

File System

Original

Modification



File

Hard Link

Changed File

API Design

“Sequential” API

```
some_test() ->
    push(),
    do_step_one(“Cancel”),
    assert_things(),
    pop(),

    do_step_one(“Continue”),
    assert_things(),

    push(),
    do_step_two(“Cancel”),
    assert_things(),
    pop(),

    ...
```

“Functional” API

```
some_test() ->
    transaction(fun() ->
        do_step_one(“Cancel”),
        assert_things()
    end),

    do_step_one(“Continue”),
    assert_things(),

    transaction(fun() ->
        do_step_two(“Cancel”),
        assert_things()
    end),

    ...
```

“Recursive” API

```
do_something(Assertions, Continuations)
```

- Assertions - Functions that make assertions about the results
- Continuations - Functions that execute inside a push/pop

“Recursive” API

```
dish_do:get_request("/",
     [ fun dish_assert:http_ok/1,
       fun(Res) ->
           dish_assert:content_type(
               “text/html”, Res)
       end ], % Assertions

     [ {“Click Login”,
       fun(Resp1) ->
           dish_do:follow_link(“Login”,
               Resp1, [], [])
       end}
     ]).   % Continuations
```

“Recursive” API

```
get_request(“/”, Assertions, Continuations)->
    Result = get_the_page(“/”),

    lists:map(fun(A) -> A(Result) end,
        Assertions),

    push(),
    lists:map(fun(C) -> C(Result) end,
        Continuations),
    pop().
```

“Recursive” API

```
get_request(URL, Headers, As, Cs)

post_request(URL, Headers, As, Cs)

follow_link(URL, Response, As, Cs)

follow_redirect(Response, As, Cs)

submit_form(FormName, FormValues, Response, As, Cs)

find_link_with_text(LinkName, Response, As, Cs)

read_email(ToAddress, Subject, As, Cs)
```

Benefits &

Drawbacks

Asymptotic Behavior

	Thin Crust	Deep Dish
Run Time	$O(N^2)$	O(N)
Space Required	O(M)	O(M+N*log(M))
Code Size	$O(N^2)$	O(N)
Scope Level	O(I)	O(N)

Demo