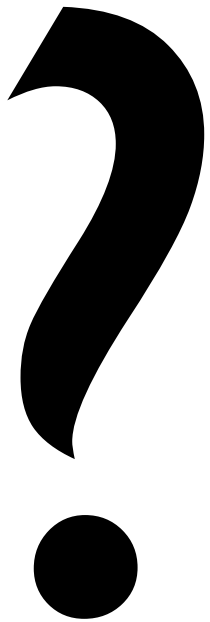
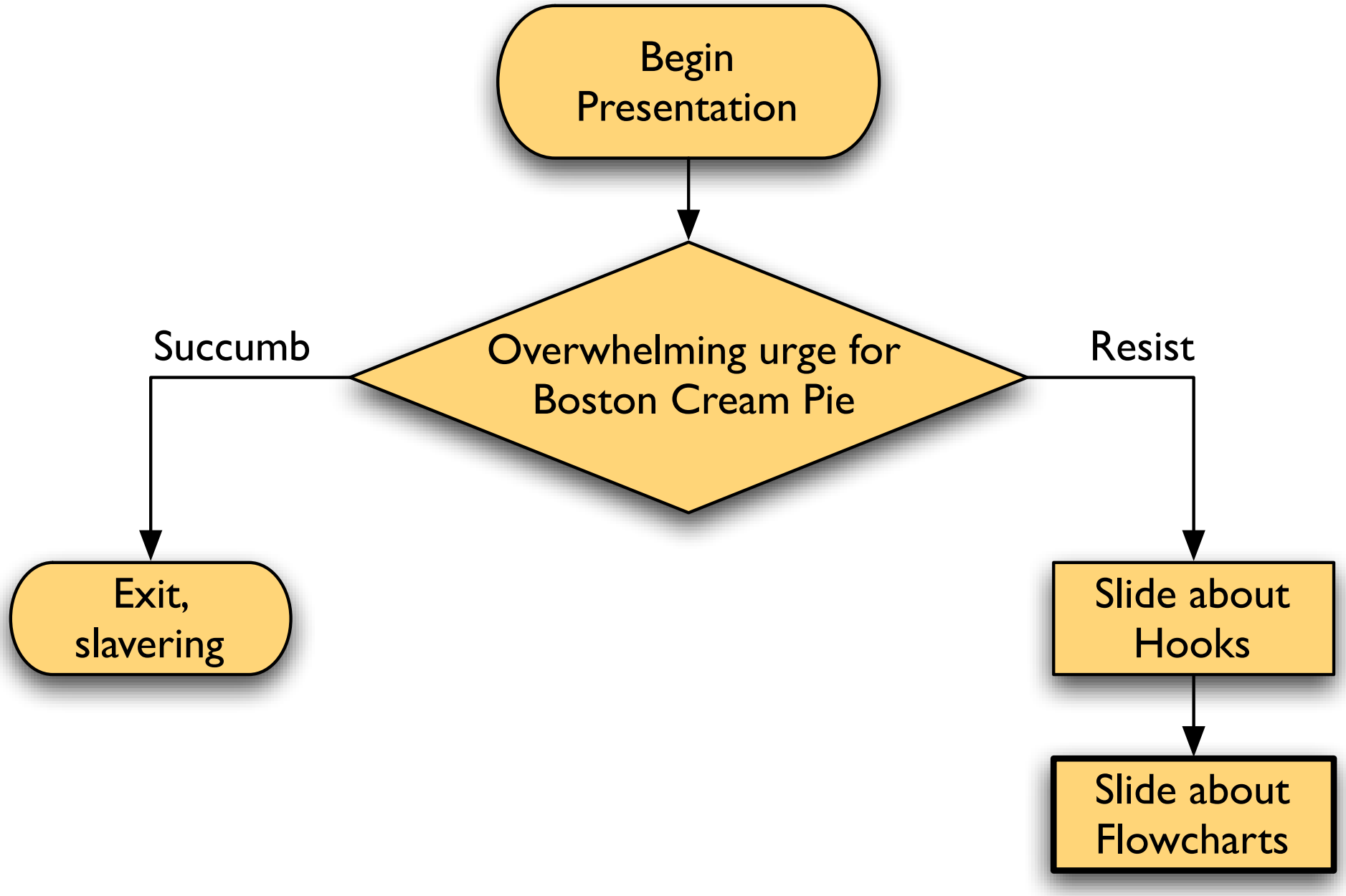
Triggers and Actions and Hooks, Oh, My!

John VanDyk

Drupal's secret language

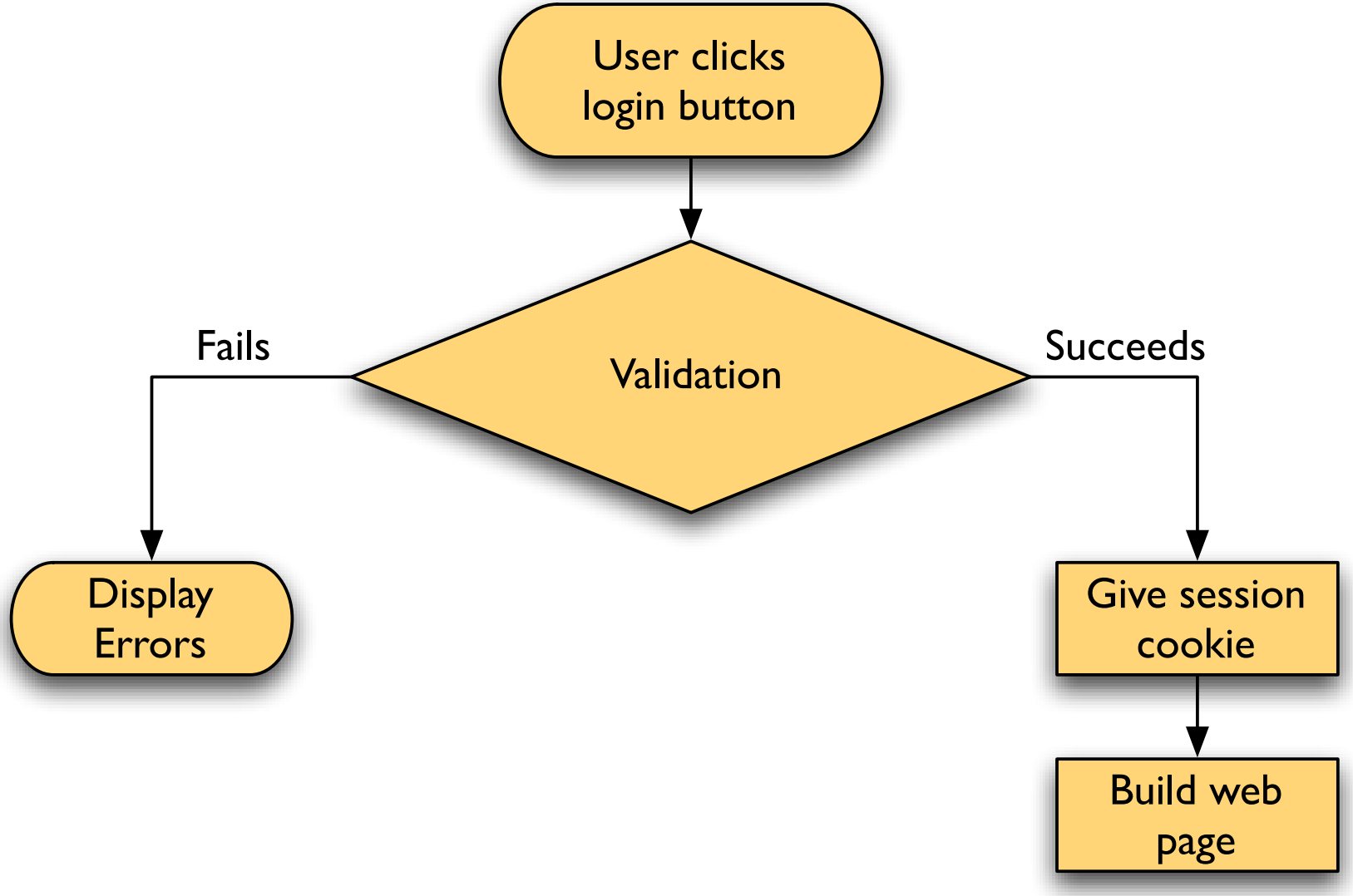
- Hooks
- Triggers
- Actions

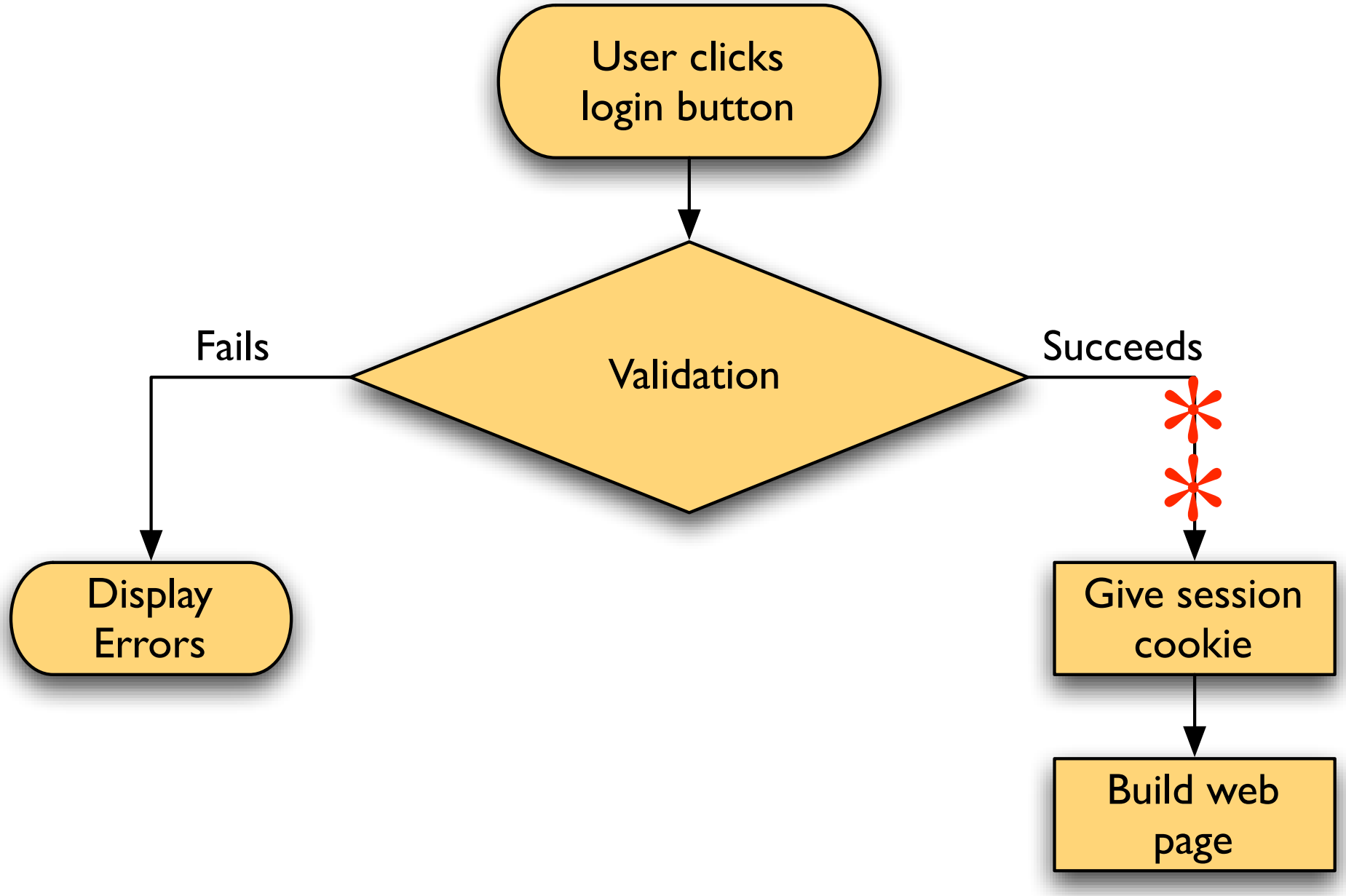



Begin Presentation
Overwhelming urge for Boston Cream Pie
Succumb
Resist
Exit, slavering
Slide about Hooks
Slide about Flowcharts

What is a hook?

What is a hook?




User clicks login button
Validation
Fails
Succeeds
Display Errors
Give session cookie
Build web page

Drupal has Events!

- User logs in
- User logs out
- User account loaded
- User profile updated
- Node created
- Node saved
- Node deleted
- Node viewed in full view
- Node viewed in teaser view

Drupal has Events!

User events!

- User logs in
- User logs out
- User account loaded
- User profile updated

Node events!

- Node created
- Node saved
- Node deleted
- Node viewed in full view
- Node viewed in teaser view

Definitions

- Event: used in the generic programming sense. A message sent from one component of a system to other components.

Definitions

- Hook: a programming technique, used in Drupal, that allows modules to hook into the flow of execution.

Definitions

- Operation: the specific operation that is being performed within a hook. For example, the login operation is an operation of the user hook.

Definitions

- Trigger: a specific combination of a hook and an operation, to which one or more actions can be associated.

Definitions

- Action: “something Drupal does”

```
function beep_beep() {
  watchdog('beep', 'Beep!');
}
```

```
/**
 * Implementation of hook_user().
 */
function beep_user($op, &$edit, &$account) {
  if ($op == 'login') {
    beep_beep();
  }
}
```

```
/**
 * Implementation of hook_nodeapi().
 */
function hook_nodeapi(&$node, $op) {
  if ($op == 'insert') {
    beep_beep();
  }
}
```

Look, Ma, no code!

User Interface

Hook	Operation	Trigger Name
comment	insert	“After saving a comment”
comment	update	“After saving an updated comment”
comment	delete	“After deleting a comment”
comment	view	“When comment is viewed by an authenticated user”

Creating an Action

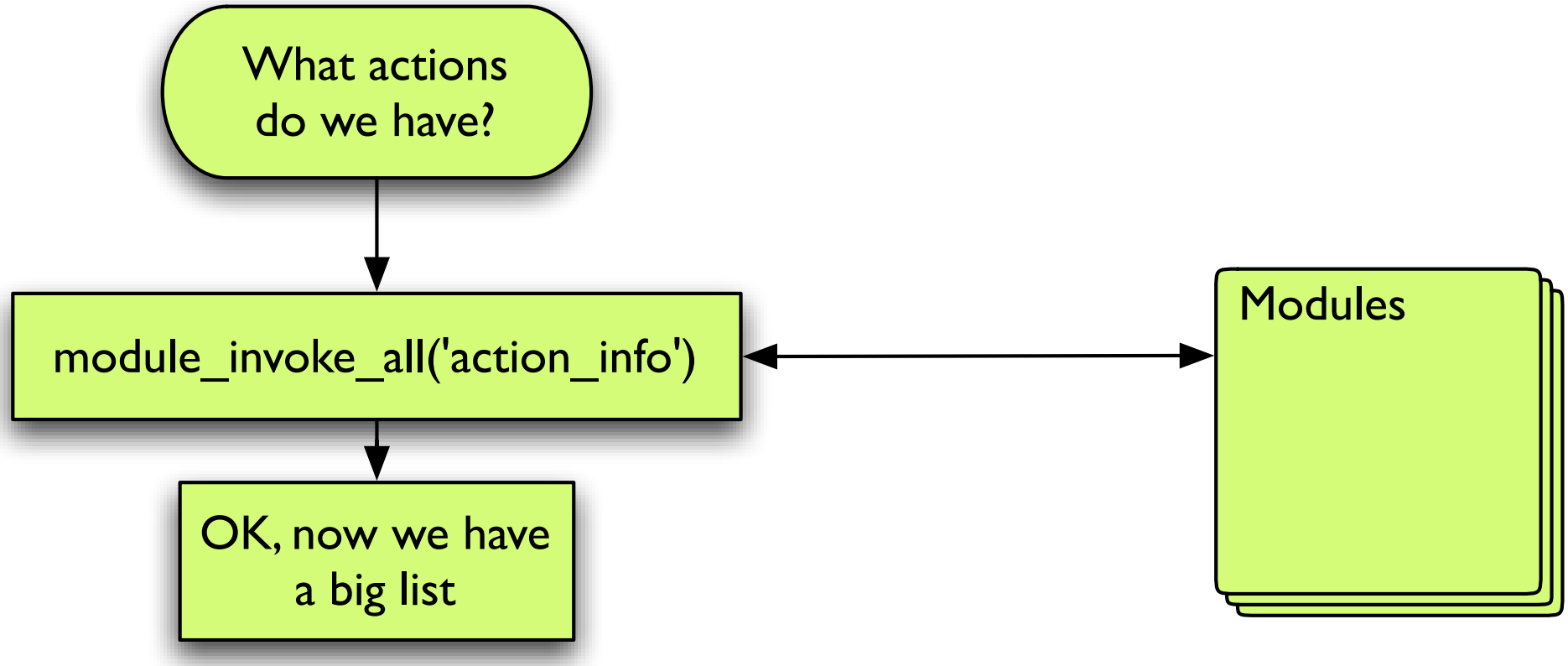
- Tell Drupal about your action
- Write your action

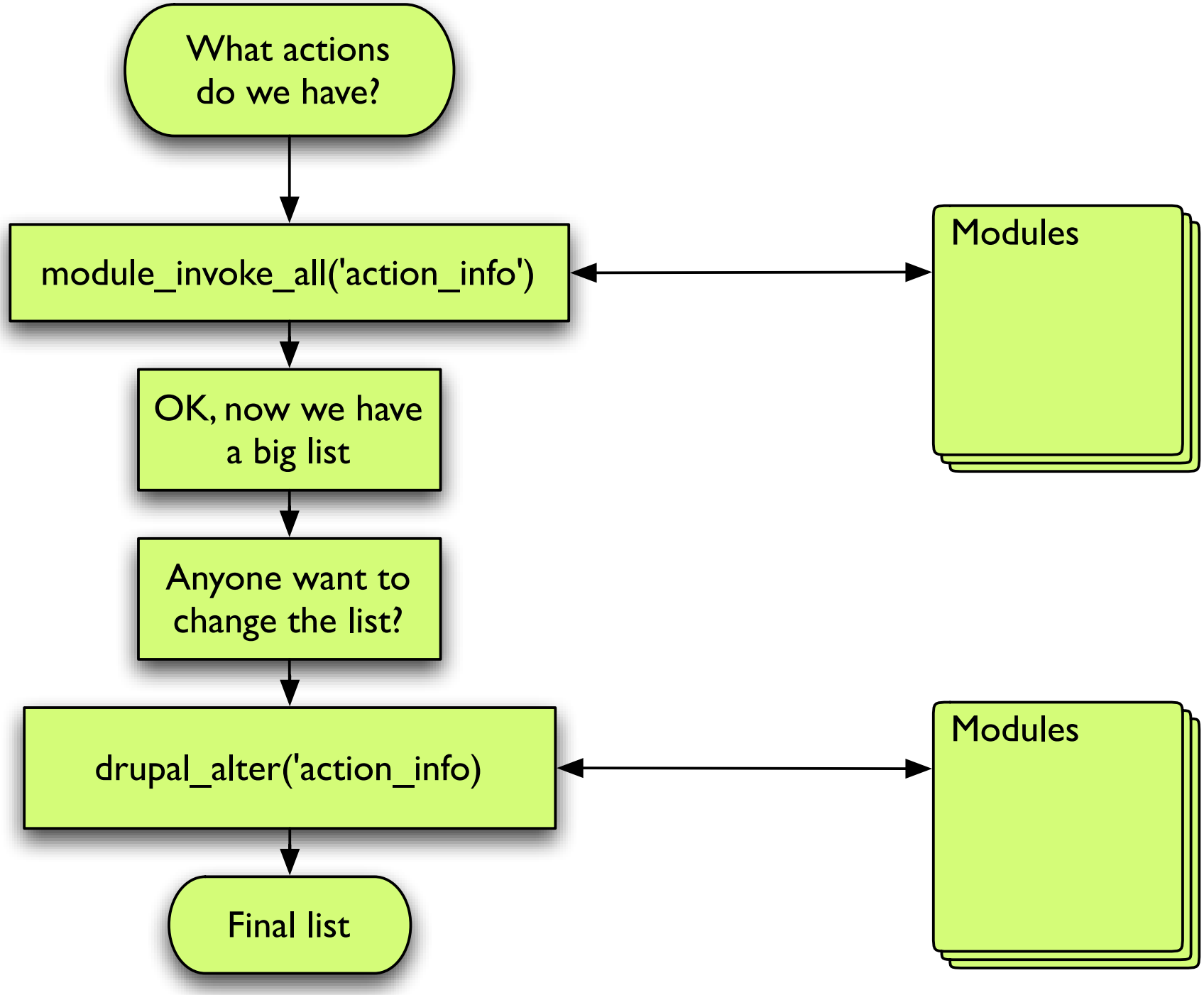
```
/**
 * Implementation of hook_action_info().
 */
function beep_action_info() {
  $info['beep_beep_action'] = array(
    'type' => 'system',
    'description' => t('Beep annoyingly'),
    'configurable' => FALSE,
    'hooks' => array(
      'user' => array('login'),
    ),
  );

  return $info;
}
```

```
/**
 * Simulate a beep. A Drupal action.
 */
function beep_beep_action() {
  beep_beep();
}
```

```
/**
 * Implementation of hook_action_info().
 */
function beep_action_info() {
  $info['beep_beep_action'] = array(
    'type' => 'system',
    'description' => t('Beep annoyingly'),
    'configurable' => FALSE,
    'hooks' => array(
      'nodeapi' => array('view', 'insert', 'update', 'delete'),
      'comment' => array('view', 'insert', 'update', 'delete'),
      'user' => array('view', 'insert', 'update', 'delete', 'login'),
      'taxonomy' => array('insert', 'update', 'delete'),
    ),
  );

  return $info;
}
```

```
/**
 * Implementation of hook_action_info().
 */
function beep_action_info() {
  $info['beep_beep_action'] = array(
    'type' => 'system',
    'description' => t('Beep annoyingly'),
    'configurable' => FALSE,
    'hooks' => array(
      'any' => TRUE,
    ),
  );

  return $info;
}
```


What actions
do we have?
module_invoke_all('action_info')
OK, now we have
a big list
Modules


What actions do we have?
module_invoke_all('action_info')
Modules
OK, now we have a big list
Anyone want to change the list?
drupal_alter('action_info)
Modules
Final list

```
/**
 * @file
 * Makes all actions available to all triggers.
 */

/**
 * Implementation of action_info_alter().
 */
function triggerunlock_action_info_alter(&$info) {
  foreach (array_keys($info) as $key) {
    $info[$key]['hooks'] = 'any';
  }
}
```

Actions Library

Advanced Actions

```
/**
 * Implementation of hook_action_info().
 */
function node_action_info() {
  return array(
    'node_assign_owner_action' => array(
      'type' => 'node',
      'description' => t('Change the author of a post'),
      'configurable' => TRUE,
      'behavior' => array('changes_node_property'),
      'hooks' => array(
        'nodeapi' => array('presave'),
        'comment' => array('delete', 'insert', 'update'),
      ),
```

Simple Actions

- Describe the action with hook_action_info()
- Write the function that does the action

Simple Actions

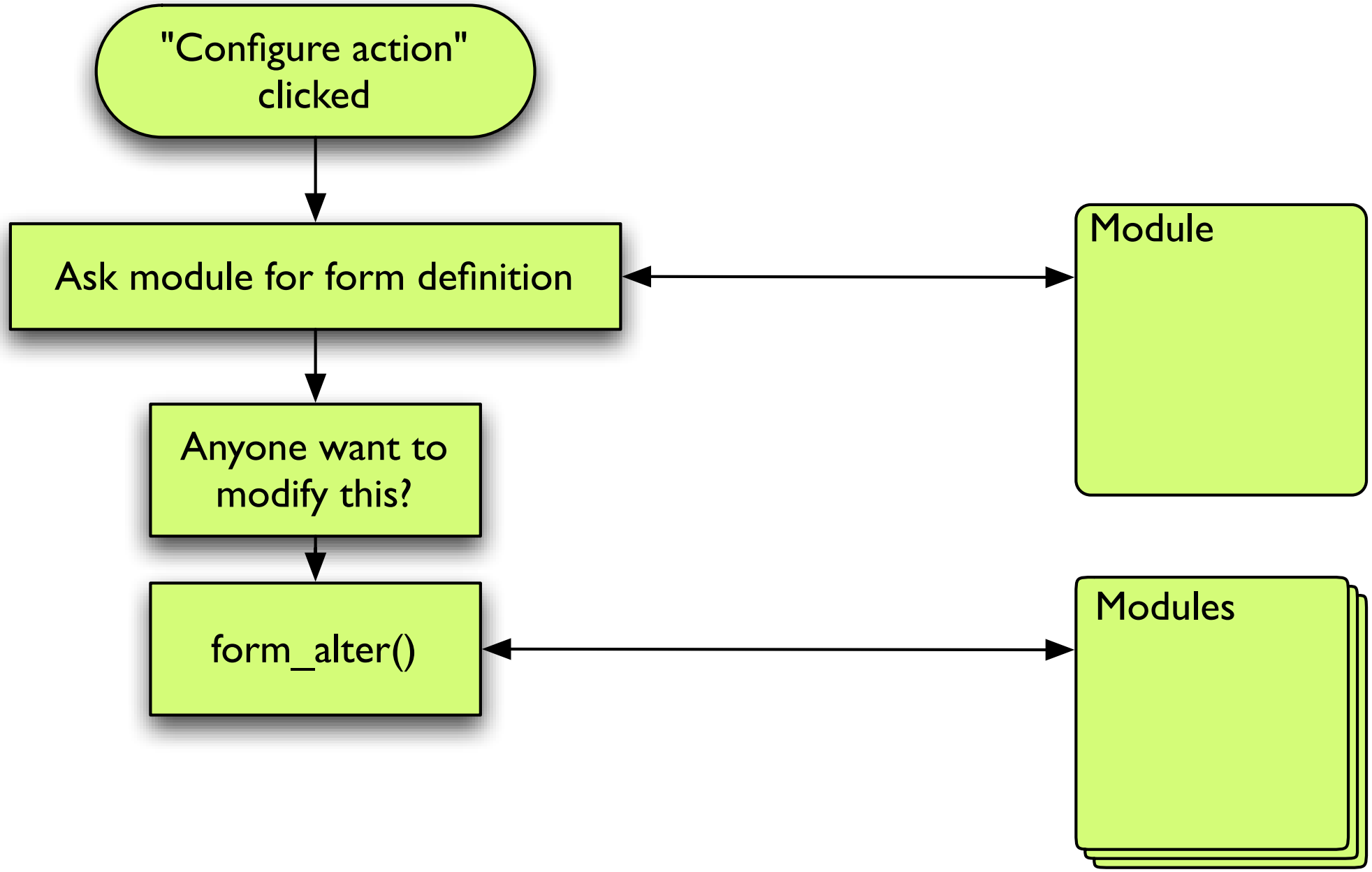
```
/**
 * Implementation of hook_action_info().
 */
function beep_action_info() {
  $info['beep_beep_action'] = array(...)
}

function beep_beep_action() {
  // Do the action
}
```

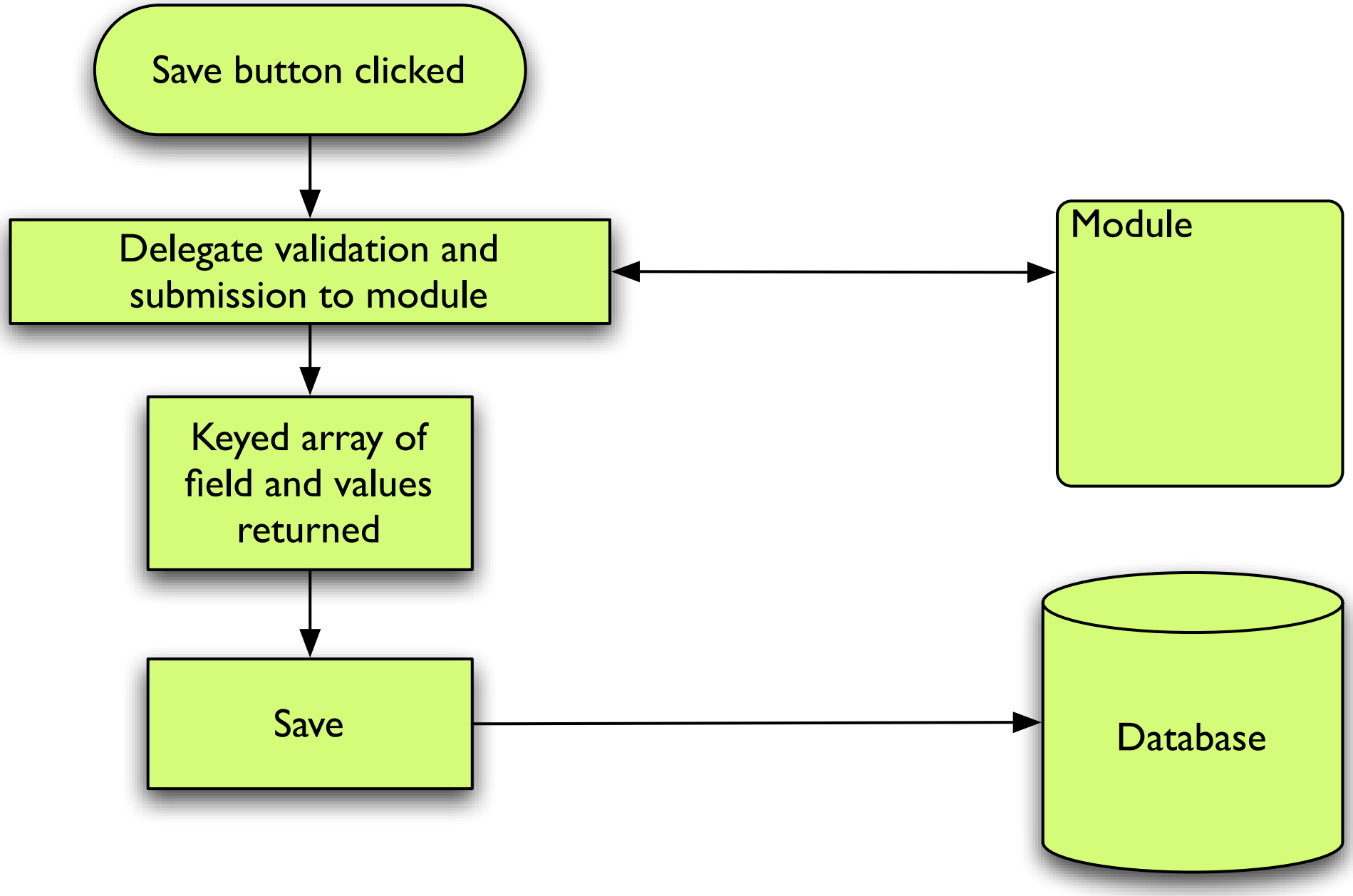
Configurable Actions

- Describe the action with hook_action_info()
- Write the function that does the action
- Provide a form definition for the configuration form
- Optionally provide form validator
- Provide form submit function

Configurable Actions

```
/**
 * Implementation of hook_action_info().
 */
function node_action_info() {
  $info['node_assign_owner_action'] = array(...)
}

function node_assign_owner_action(&$node, $context) {
  // Do the action
}

function node_assign_owner_action_form($context) {
  return $form;
}

function node_assign_owner_action_validate($form, $form_state) {...}

function node_assign_owner_action_submit($form, $form_state) {...}
```


"Configure action" clicked
Ask module for form definition
Module
Anyone want to modify this?
form_alter()
Modules


Save button clicked
Delegate validation and submission to module
Module
Keyed array of field and values returned
Save
Database

```
function node_assign_owner_action_form($context) {
   ...
   $form['owner_name'] = array(
    '#type' => 'select',
    '#title' => t('Username'),
    '#default_value' => $owner_name,
    '#options' => $options,
    '#description' => t('The username of the user to which you would
      like to assign ownership.');,
  );

  return $form;
}
```

```
function node_assign_owner_action_validate($form, $form_state) {
  // Is this a legitimate user?

  $count = db_result(db_query("SELECT COUNT(*) FROM {users} WHERE name =
'%s'", $form_state['values']['owner_name']));

  if (intval($count) != 1) {
    form_set_error('owner_name', t('Please enter a valid username.'));
  }
}
```

```
function node_assign_owner_action_submit($form, $form_state) {
  // Username can change, so we need to store the ID,
  // not the username.

  $uid = db_result(db_query("SELECT uid from {users} WHERE name = '%s'",
$form_state['values']['owner_name']));

  return array('owner_uid' => $uid);
}
```

```
/**
 * Implementation of a configurable Drupal action.
 * Assigns ownership of a node to a user.
 */
function node_assign_owner_action(&$node, $context) {
  $node->uid = $context['owner_uid'];

  $owner_name = db_result(db_query("SELECT name FROM {users} WHERE uid =
%d", $context['owner_uid']));

  watchdog('action', 'Changed owner of @type %title to uid %name.',
array('@type' => node_get_types('type', $node), '%title' => $node-
>title, '%name' => $owner_name));
}
```

I lied to you.

```
/**
 * Simulate a beep. A Drupal action.
 */
function beep_beep_action(&$object, $context) {
  beep_beep();
}
```

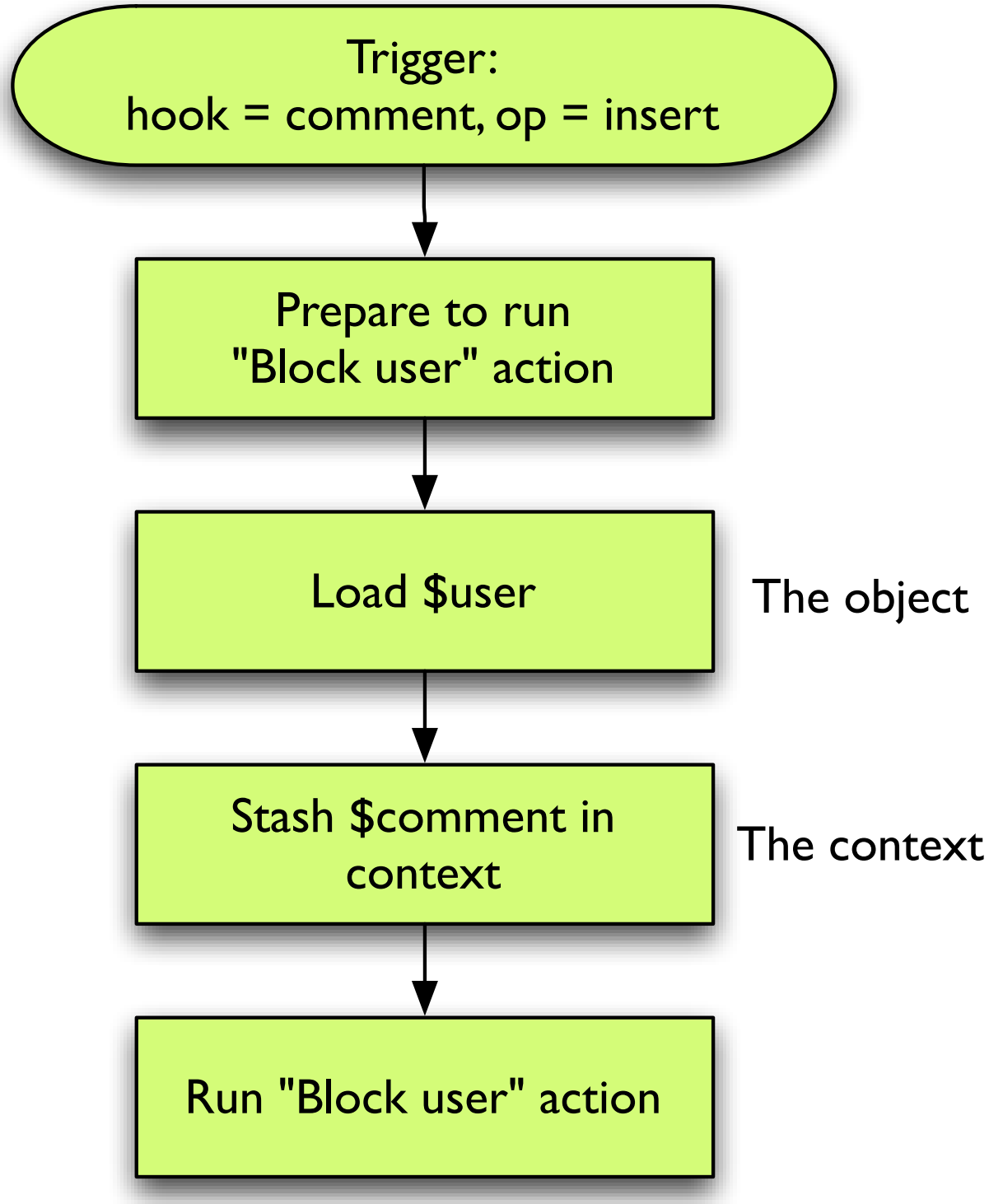
```
/**
 * Implementation of a Drupal action.
 * Sets the status of a node to 1, meaning published.
 */
function node_publish_action(&$node, $context = array()) {
  $node->status = 1;

  watchdog('action', 'Set @type %title to published.',
    array(
      '@type' => node_get_types('name', $node),
      '%title' => $node->title,
    )
  );
}
```

What is context?

- As much information as the Drupal can provide

What is context?

- Which hook is being fired
- Information Drupal already knows (think of hook parameters)
- An email action in user vs nodeapi hook?


Trigger:
hook = comment, op = insert
Prepare to run
"Block user" action
Load $user
The object
Stash $comment in
context
The context
Run "Block user" action

```
/**
 * Implementation of a configurable Drupal action. Sends an email.
 */
function system_send_email_action($object, $context) {

  switch ($context['hook']) {
    case 'nodeapi':
      // Because this is not an action of type 'node' the node
      // will not be passed as $object, but it will still be available
      // in $context.
      $node = $context['node'];
      break;

    // The comment hook provides nid, in $context.
    case 'comment':
      $comment = $context['comment'];
      $node = node_load($comment->nid);
      break;
```

Creating Triggers

hook_hook_info()

```
/**
 * Implementation of hook_hook_info().
 */
function comment_hook_info() {
  return array(
    'comment' => array(
      'comment' => array(
        'insert' => array(
          'runs when' => t('After saving a new comment'),
        ),
        'update' => array(
          'runs when' => t('After saving an updated comment'),
        ),
        'delete' => array(
          'runs when' => t('After deleting a comment')
        ),
        'view' => array(
          'runs when' => t('When a comment is being viewed by an
                            authenticated user')
        ),
      ),
    ),
  );
}
```

```
/**
 * Implementation of hook_hook_info().
 */
function monitoring_hook_info() {
  return array(
    'monitoring' => array(
      'monitoring' => array(
        'overheating' => array(
          'runs when' => t('When hardware is about to melt down'),
        ),
        'freezing' => array(
          'runs when' => t('When hardware is about to freeze up'),
        ),
      ),
    ),
  );
}
```

Triggers

Trigger: When hardware is about to melt down

Choose an action | Assign

Trigger: When hardware is about to freeze up

Choose an action | Assign

The Future

- More modules exposing actions!
- Conditional actions
- Refactoring Drupal
- Actions as wrappers
- Expose actions externally via web services

Actions/triggers
backport
to Drupal 5