Drupal and Security

Nathan Rambeck
Crown Partners
@nrambeck
drupal.org/user/92967

crown
xcel faster™

Security Concerns

- Hackers
  - Cross site scripting (XSS) 80% of vulnerabilities web-wide
  - SQL Injection
  - Cross Site Request Forgeries
- Spammers
- Brute Force Attacks
- Exposing Unintended Data
- Drupal Security is Just One Layer
- Over-securing alienates users

Drupal's Security Methodology

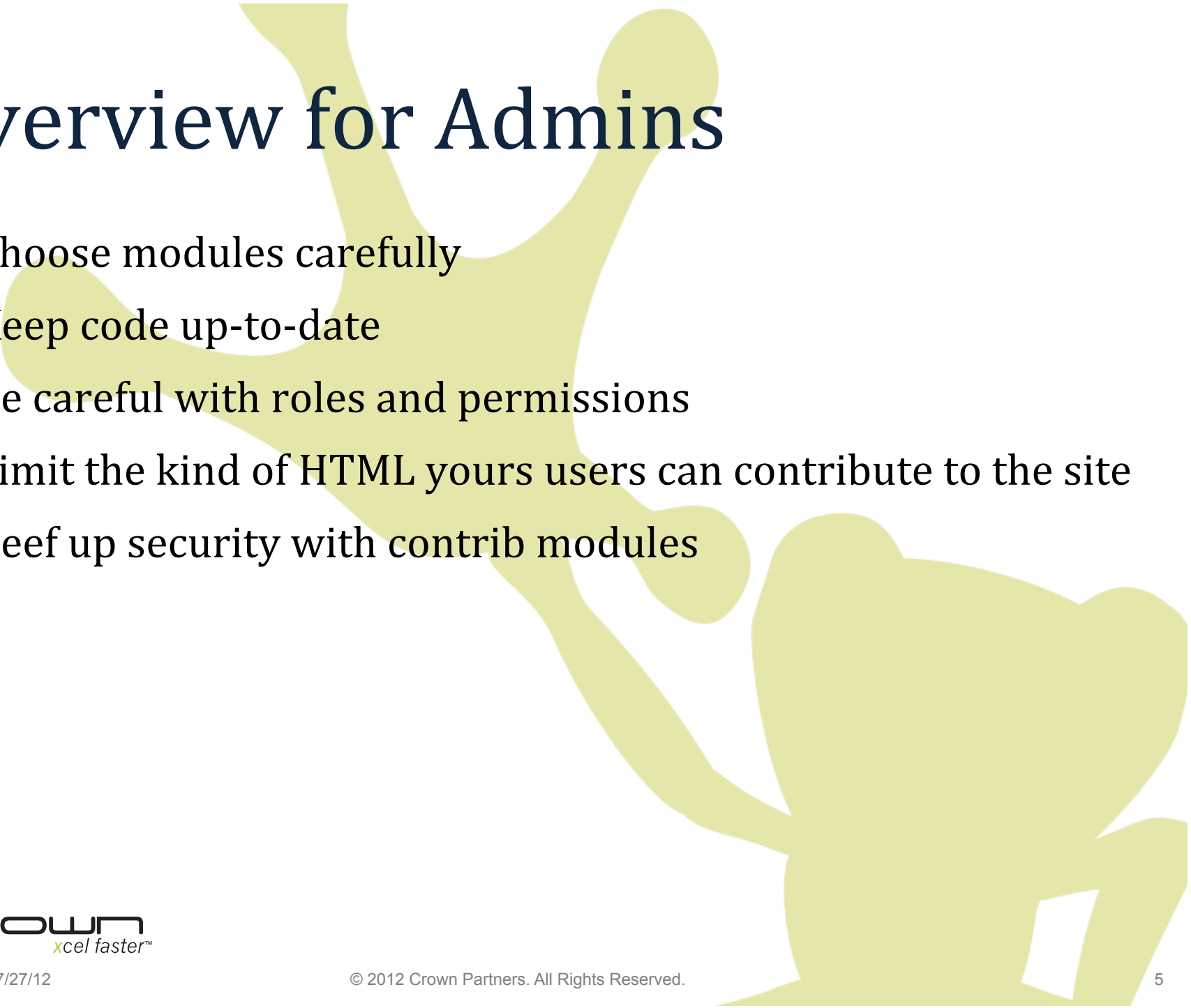
- Security through transparency vs obscurity
- Dedicated security team - http://drupal.org/security-team
- Reporting - security@drupal.org
- Security releases and announcements - http://drupal.org/security
- Security advisory feed
- Security email list
- Update Status module

Security for Site Admins

xcel faster™

Overview for Admins

- Choose modules carefully
- Keep code up-to-date
- Be careful with roles and permissions
- Limit the kind of HTML yours users can contribute to the site
- Beef up security with contrib modules



Choose modules carefully

- Some modules are more secure than others
- Is the module actively maintained?
- Is it popular? http://drupal.org/project/usage
- Are issues in the issue queue addressed in a timely manner?

Keep code up-to-date

- Turn on the Update Status module
- Configure Update Status to email you
- Security updates are most important
- Use drush to update modules
- Use Git or a shell script to update core

Roles and Permissions

- Should self-registration be allowed?
- Carefully create roles to divide trusted users from untrusted
- Review the permissions page carefully
- Edit permissions one role at a time

Input formats and filters

- Don't turn on PHP filter ~~unless it's required~~ ever
- Be careful with the Full HTML filter and don't ever make it the default
- Untrusted user should never be allowed to use the following tags: <img> <script> <iframe> <embed> <object> <input> <link> <style> <meta> <frameset> <base>

Images and media

- Allowing users the flexibility to post any kind of media also opens your site up to hackers/spammers
- Contrib modules provide a safe way to allow users to contribute images and other media
- Image Field – Users upload their own images
- Media module – Users may embed media from a pre-selected group of 3rd parties (ie. Youtube, Vimeo, etc.)

Security related modules

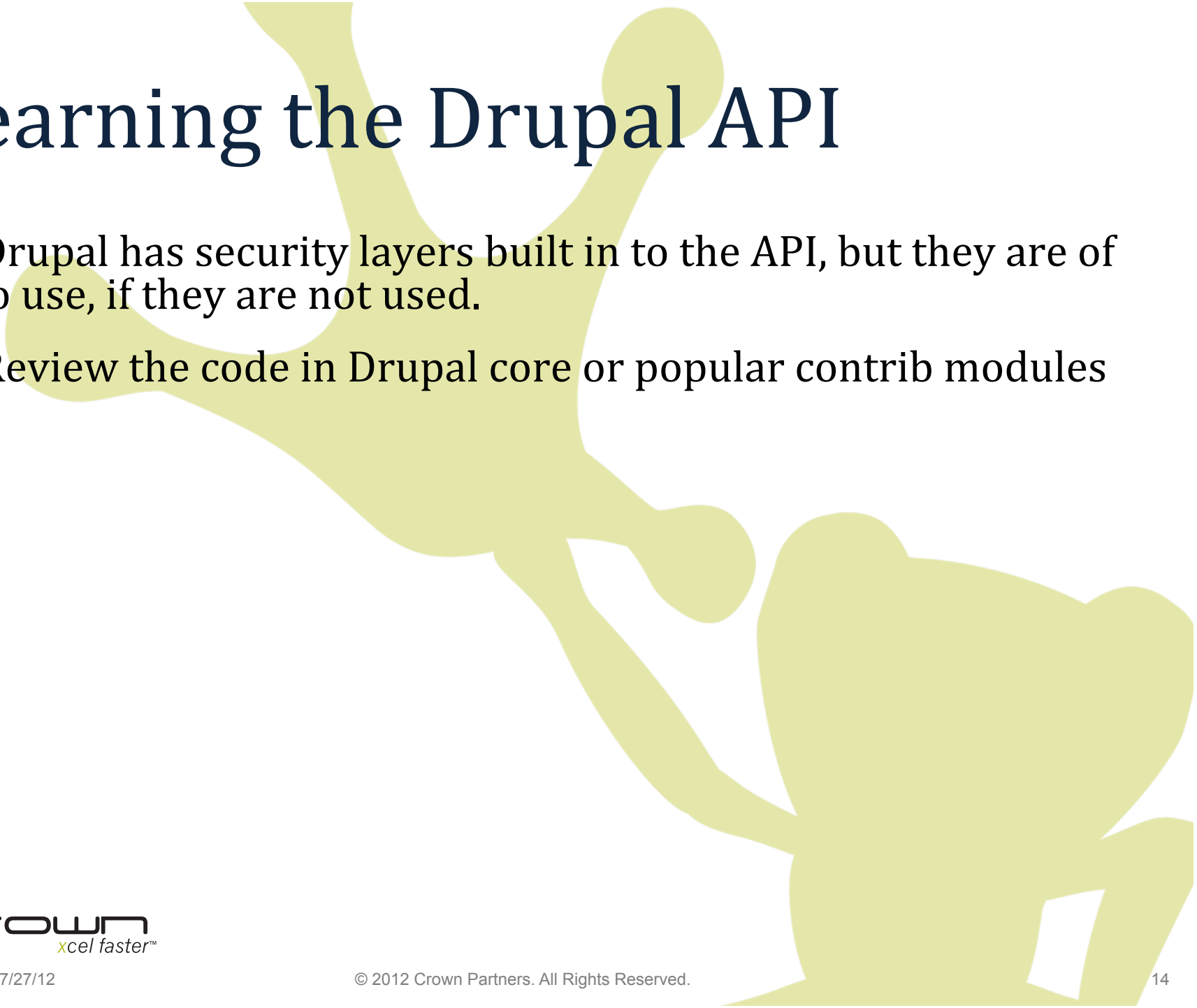
- Security Review (http://drupal.org/project/security_review)
- Flood Control (http://drupal.org/project/flood_control)
- Secure Login (http://drupal.org/project/securelogin)
- Persistent Login (http://drupal.org/project/persistent_login)
- Password Policy (http://drupal.org/project/password_policy)

Overview for Developers

- **Learn the Drupal API**
- Input filter functions
- Query data filtering
- User access
- Forms
- Filter Access
- URLs

Learning the Drupal API

- Drupal has security layers built in to the API, but they are of no use, if they are not used.
- Review the code in Drupal core or popular contrib modules



Filtering Input

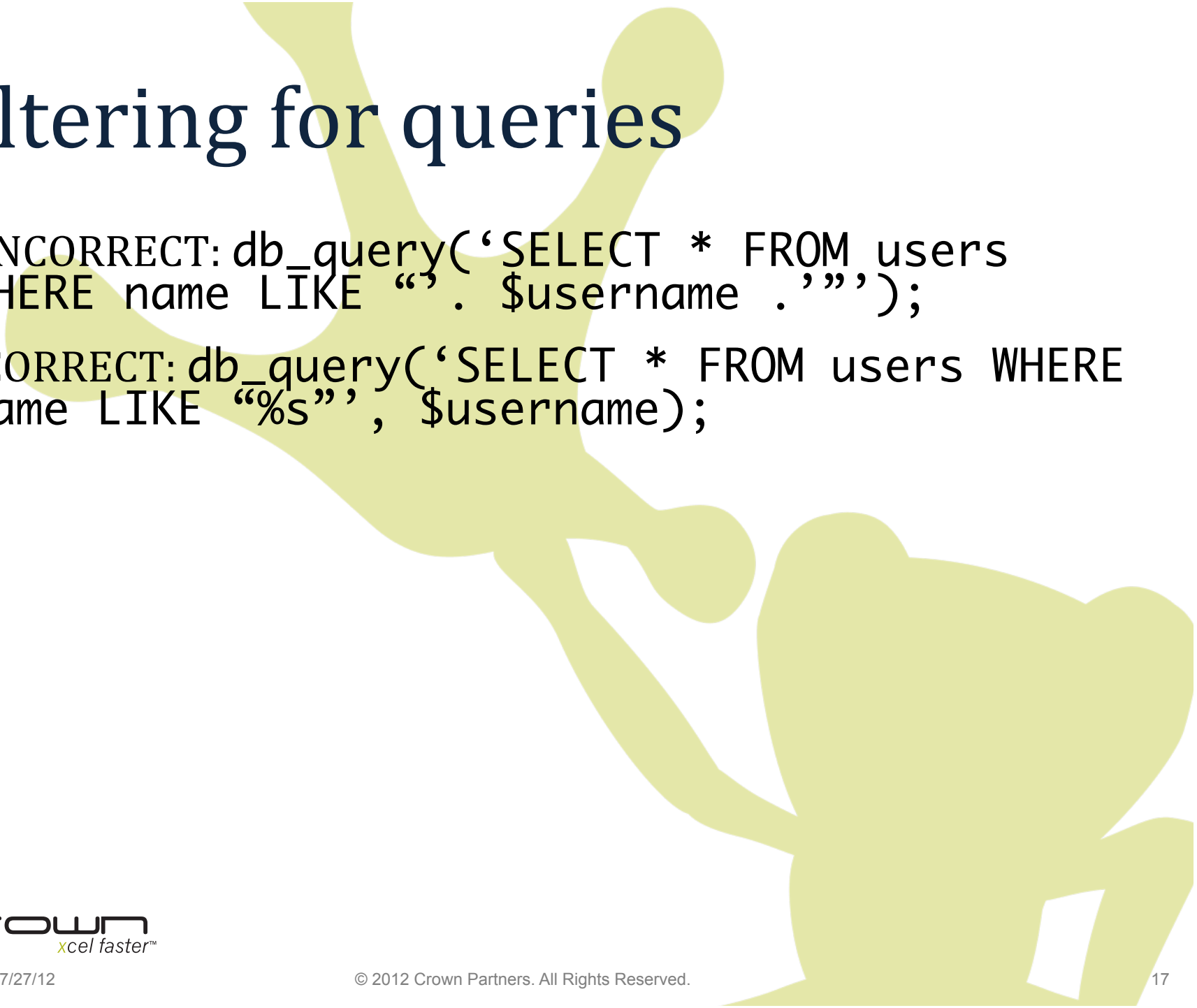
- Input from users should always be treated as potentially malicious
- **check_plain() –** filters text as plain text; used for fields like the title of a post which doesn't allow HTML
- **check_markup() –** filter text as HTML according to the rules of a specific filter defined in Drupal
- **filter_xss() –** allows most HTML, but filters for Cross Site Scripting attacks.

Filtering Input

- **t() –** Translate function uses tokens to allow easy filtering of mixed static/variable text.
  t("You just entered @title as the title", array("@title" => $title);

Filtering for queries

- INCORRECT: `db_query('SELECT * FROM users WHERE name LIKE "'. $username .'"');`
- CORRECT: `db_query('SELECT * FROM users WHERE name LIKE "%s"', $username);`



User Access

- **user_access()** – Determine if a user has a specific permission

```
if (user_access('access content')) {
  // Do this
} else {
  drupal_access_denied();
}
```

- **hook_perm()** to define your own permissions
- User access in menu items

```
116  /**
117   * Implementation of hook_menu().
118   */
119  function block_menu() {
120    $items['admin/build/block'] = array(
121      'title' => 'Blocks',
122      'description' => 'Configure what block content appears in your site
123      'page callback' => 'block_admin_display',
124      'access arguments' => array('administer blocks'),
125      'file' => 'block.admin.inc',
126    );
```

User Access

- Global $user object
- Because it is global, it is easy to access, but also easy to modify

```
global $user;
if ($user->uid = 1) {
  print_r($variable);
}
```

- Oops. Now, all visitors to your site will become user 1 with unlimited access

URLs

- **check_url()** – strips user input urls of dangerous protocols and encodes for output to HTML
- Use **confirm_form()** in callbacks for urls that take permanent action. This prevents malicious urls like below from eating your lunch.
- (CSRF vulnerabilities)

Forms

- Use the Form API
- It may seem difficult to use, but is quite flexible and provides a robust layer of security
- To prevent Cross Site Forgery Requests (CSFR) Drupal generates tokens for each form
- Select field data cannot be manipulated
- Validate variable form input

Flood Control

- Prevent users from executing actions too frequently
- Already built-in to public forms (like contact form)
- Prevents form to email spam
- Prevents dictionary attacks to guess passwords
- http://api.drupal.org/api/search/7/flood

Security Resources

- http://drupal.org/security
- http://drupal.org/writing-secure-code
- Twitter: @drupalsecurity
- http://ha.ckers.org/xss.html
- http://crackingdrupal.com


Questions?
Nathan Rambeck
@nrambeck
crownpartners.com
crown
xcel faster™