Advanced Webforms Part 1 - Demo

HELLO!

Hi, my name is Jacob Rockowitz.

- I am known as jrockowitz on the web.
- I am a Drupal developer and software architect.
- I built and maintain the Webform module for Drupal 8.

GOALS FOR THIS PRESENTATION

- Walk-thru Webform features and functionality
- Show you how to create rich applications
- Think about how to improve your webforms

- Inspire you to learn more about the Webform module and Drupal

OUR ITERATIVE PROCESS

Plan:	What are we trying to build
Build:	Walk-thru how to build it
Test:	Review and test what we built
Improve:	Enhance what we built
Review:	Recap what we explored

WEBFORM

INTRODUCTION

THE WEBFORM MODULE IS A POWERFUL AND FLEXIBLE OPEN SOURCE FORM BUILDER & SUBMISSION MANAGER FOR DRUPAL 8

IT PROVIDES ALL THE FEATURES EXPECTED FROM AN ENTERPRISE PROPRIETARY FORM BUILDER...

...COMBINED WITH THE FLEXIBILITY AND OPENNESS OF DRUPAL

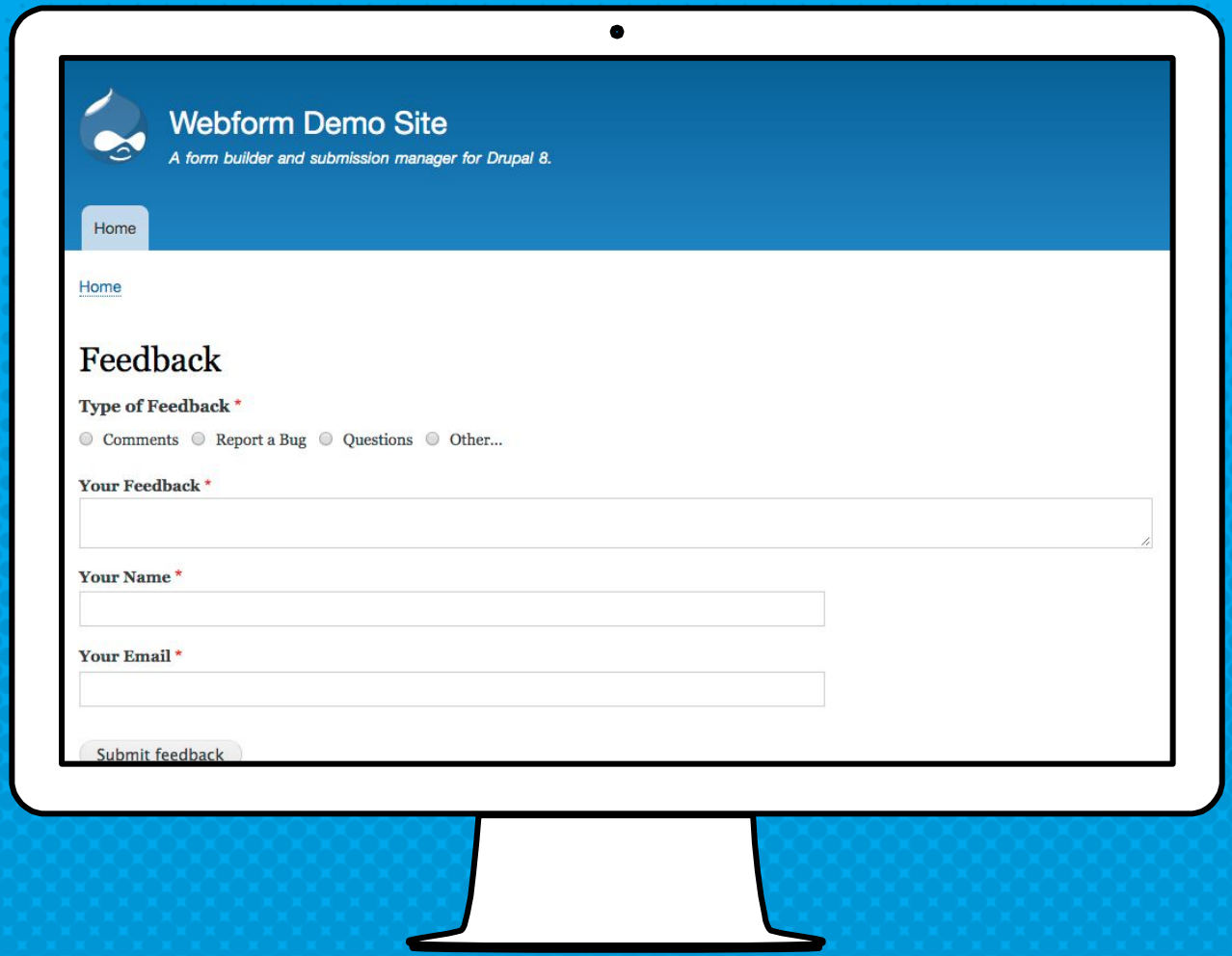
WEBFORM: THE USE CASE...

- **BUILD** a form or copy a template
- **PUBLISH** the form as a page, node, or block
- **COLLECT** form submissions
- **SEND** confirmations and notifications
- **REVIEW** results online
- **DISTRIBUTE** results as CSV or remote post

WEBFORM: ONCE AGAIN, THE USE CASE...

FEEDBACK FORM

THIS IS A FEEDBACK FORM



Feedback Form: Plan

- Collect feedback about a website from users
- Submissions will generate emails
- Display a confirmation page

Feedback Form: Build

- Elements (Type, Message, Name, and Email)
- Confirmation email
- Notification email
- Confirmation page

Feedback Form: Test

- × Generate a submission

- × Confirm that emails are sent
- × Confirm that results are saved
- × Confirm that results can be exported


<demo>
Build & Test

Feedback form: Improve

Elements: Default values; Layout

Settings: Ajax; Preview; Modal confirmation

Handler: Route notifications to different emails

Block: Place feedback form on every page.

Advanced: Subscribe users to a mailing list


<demo>
Improve

Feedback Form: Review

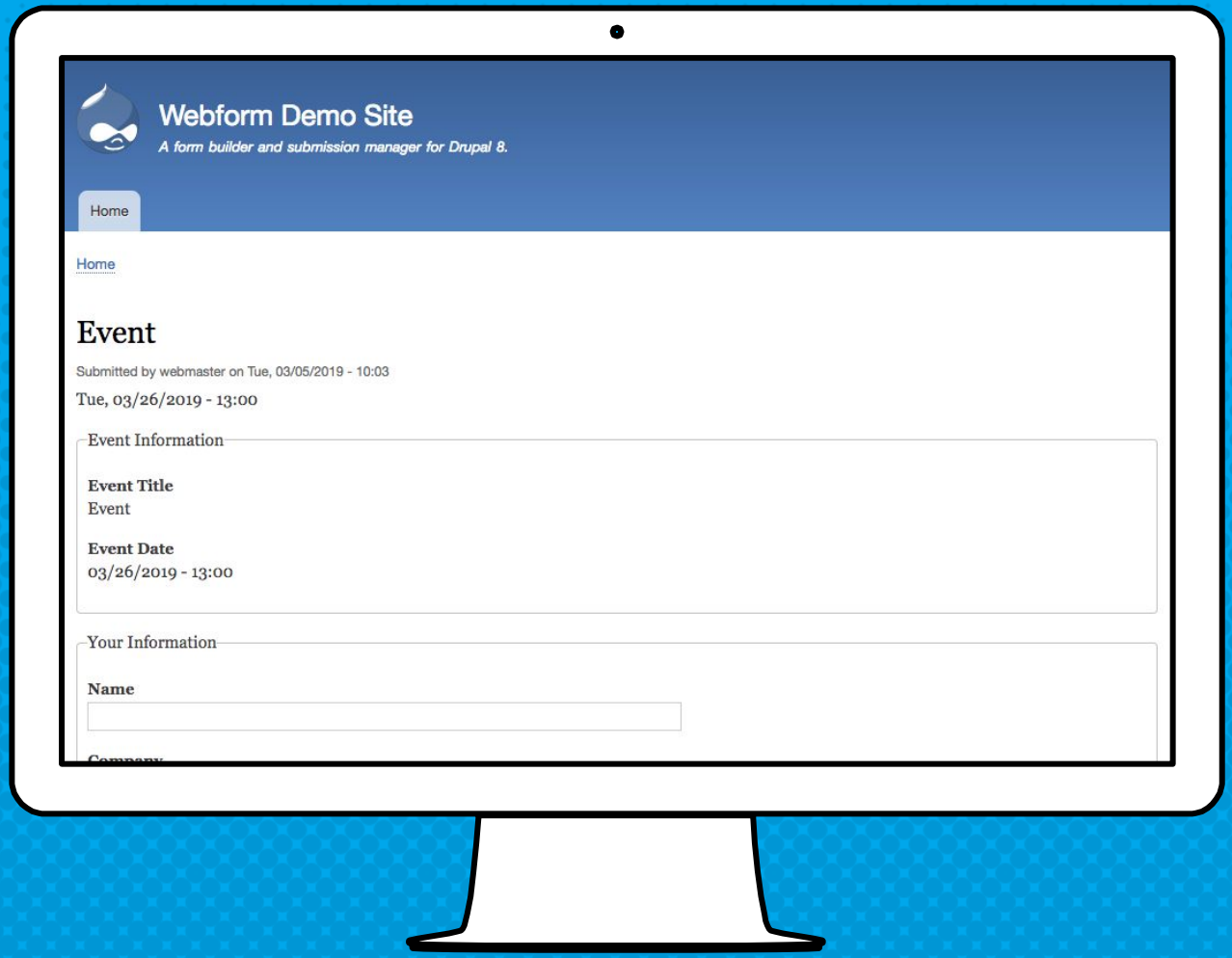
- × Form builder
- × YAML source editor
- × Form settings
- × Confirmations
- × Email handling
- × Blocks
- × Add-ons

BEN SAYS...

ANY QUESTIONS?

EVENT REGISTRATION SYSTEM

THIS IS AN EVENT REGISTRATION FORM



Event registration: plan

- Allow users to register for an event (node)
- Registration form (webform)
- Admins can review registrants (submissions)

EVENT REGISTRATION: GLOSSARY

Webform: Collects a submission via a form

Submission: Data collected via a webform

Content type: Defines a node

Node: A piece of individual content

Reference: A relationship between entities

Event Registration: Build Webform

- Registration form (Event and Contact)
- Confirmation page

Event Registration: Build Event

- Event content type (node)
- Includes date/time field
- Webform field (entity reference/relationship)

EVENT REGISTRATION: TEST

- × Create an event
- × Generate a registration

- × Check confirmation page
- × Confirm event specific registration
- × Customize results


<demo>
Build & Test

Event Registration: Improve

Elements:	Additional guests
Settings:	Ajax; View previous submissions
Handler:	Scheduled email reminders
Advanced:	Open in a modal dialog


<demo>
Improve

Event Registration: Review

- Composite elements
- Element with multiple values
- Webform nodes
- Settings
- Scheduled email handler
- Modals

LILI SAYS...

ANY NEW QUESTIONS?

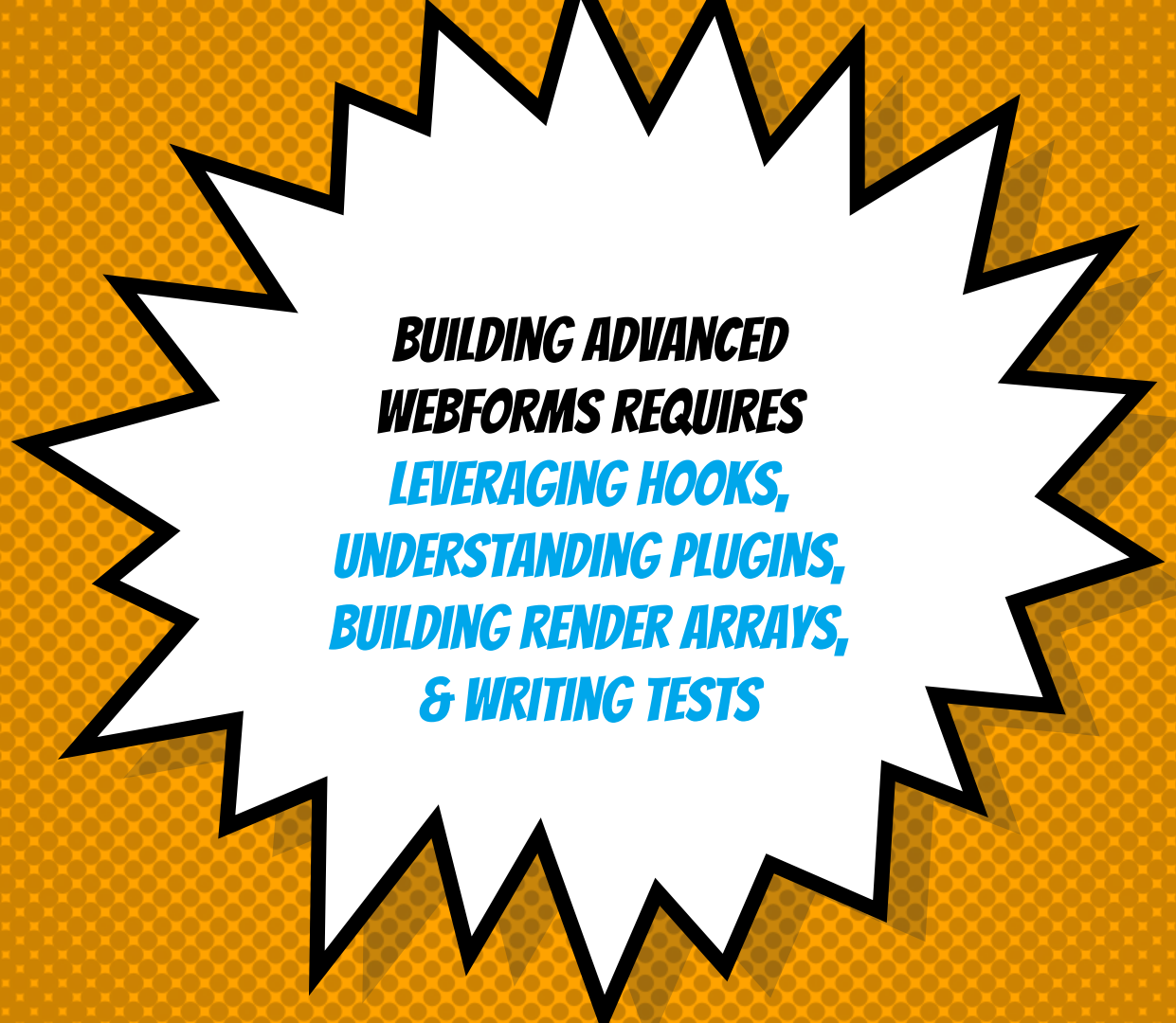

TO BE
CONTINUED...

Advanced Webforms Part II - APIs

Goals for this presentation

- Explore the code behind Webforms
- Understand how a Webform is built
- Learn how to extend the Webform module

- Inspire you to become Webform expert


BUILDING ADVANCED
WEBFORMS REQUIRES
LEVERAGING HOOKS,
UNDERSTANDING PLUGINS,
BUILDING RENDER ARRAYS,
& WRITING TESTS

TESTING WEBFORMS

TESTS CONFIRM

EXPECTATIONS

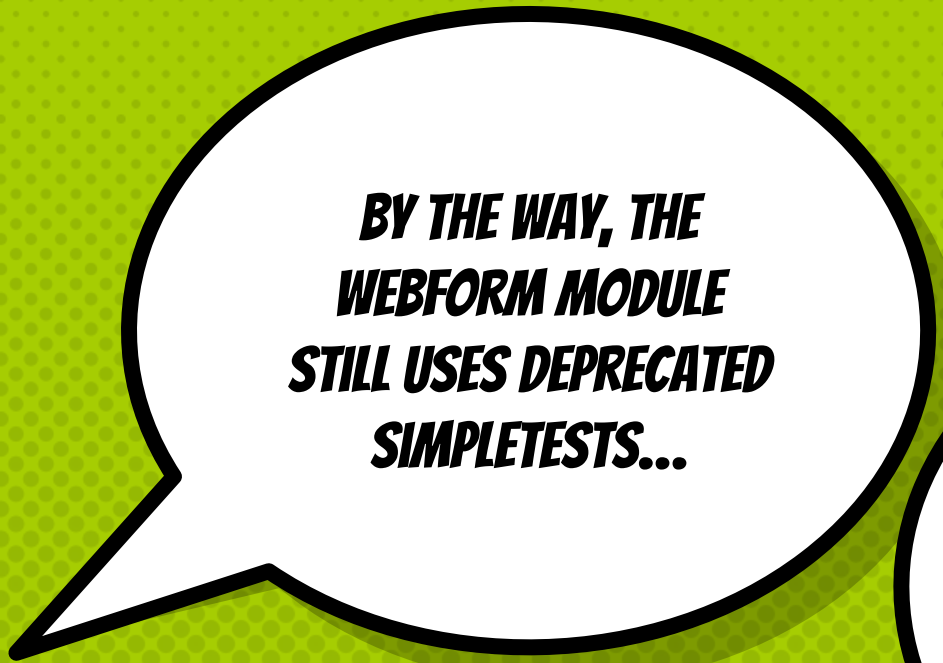
WebForm tests confirm

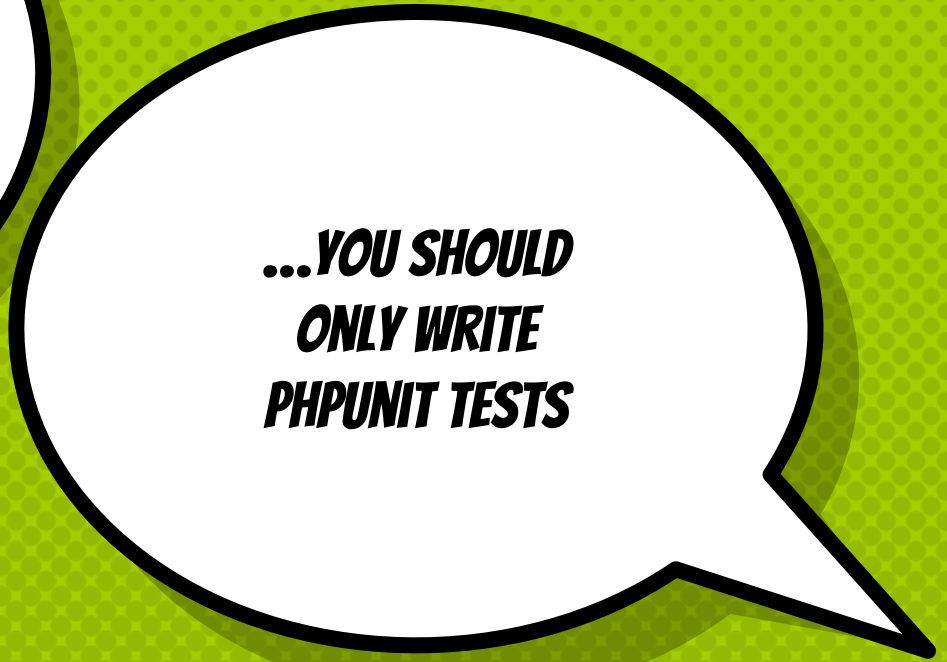
- Rendering: Check an element's markup
- Processing: Check an input's default value
- Validation: Check required error messages
- Settings: Check labels and layout
- Access: Check user access controls

Testing Best Practices

- Create a test module with exported webforms
- Write tests for every element and setting
- Establish repeatable testing patterns
- Organize tests into groups/subdirectories

- Having easy-to-repeat manual tests is okay
- Some tests are better than no tests


BY THE WAY, THE WEBFORM MODULE STILL USES DEPRECATED SIMPLETESTS...


...YOU SHOULD ONLY WRITE PHPUNIT TESTS


<demo>
Testing Webform
Overview: /admin/structure/webform
Settings: ../webform?search=confirmation
Elements: ../webform?search=email
Modules: /admin/modules(webform_test)

```
@see \Drupal\webform\Tests\Element\WebformElementEmailTest

// Check basic email multiple rendering.
$this->drupalGet('webform/test_element_email');
$this->assertRaw('<label for="edit-email-multiple-basic">Multiple email
  addresses (basic)</label>');
$this->assertRaw('<input type="text"
  id="edit-email-multiple-basic" name="email_multiple_basic" value=""
  size="60" class="form-text webform-email-multiple" />');
$this->assertRaw('Multiple email addresses may be separated by
  commas.');

// Check that second email address is not valid.
$edit = [
 'email_multiple_basic' => 'example@example.com, xxx',
];
$this->drupalPostForm('webform/test_element_email', $edit, t('Submit'));
$this->assertRaw('The email address <em class="placeholder">xxx</em> is
  not valid.');
```

```
@see \Drupal\webform\Tests\Settings\WebformSettingsConfirmationTest

// Check confirmation modal.
$webform = Webform::load('test_confirmation_modal');
$this->postSubmission($webform, ['test' => 'value']);
$this->assertRaw('This is a <b>custom</b> confirmation modal.');
$this->assertRaw('<div class="js-hide webform-confirmation-modal...>');

// Check custom confirmation page.
$webform = Webform::load('test_confirmation_page_custom');
$this->postSubmission($webform);
$this->assertRaw('<div style="border: 10px solid red; padding: 1em;"
  class="webform-confirmation">');

// Check confirmation URL.
$webform = Webform::load('test_confirmation_url_message');
$this->postSubmission($webform);
$this->assertRaw('This is a custom confirmation message.');
$this->assertUrl('<front>');
```

```
@see \Drupal\webform_ui\Tests\WebformUiElementTest

// Check access allowed (200) to textfield element.
$this->drupalGet('admin/structure/webform/manage/contact/element/add/tex
tfield');
$this->assertResponse(200);

// Check access denied (403) to password element,
// which is disabled by default.
$this->drupalGet('admin/structure/webform/manage/contact/element/add/pas
sword');
$this->assertResponse(403);
```

Additional Resources

- Testing | Drupal.org
  http://dgo.to/2819027
- An Overview of Testing in Drupal 8 | Lullabot
  https://www.lullabot.com/articles/an-overview-of-testing-in-drupal-8
- Automated Testing in Drupal 8 | Drupalize.me
  https://drupalize.me/series/automated-testing-drupal-8

WEBFORM

ENTITIES

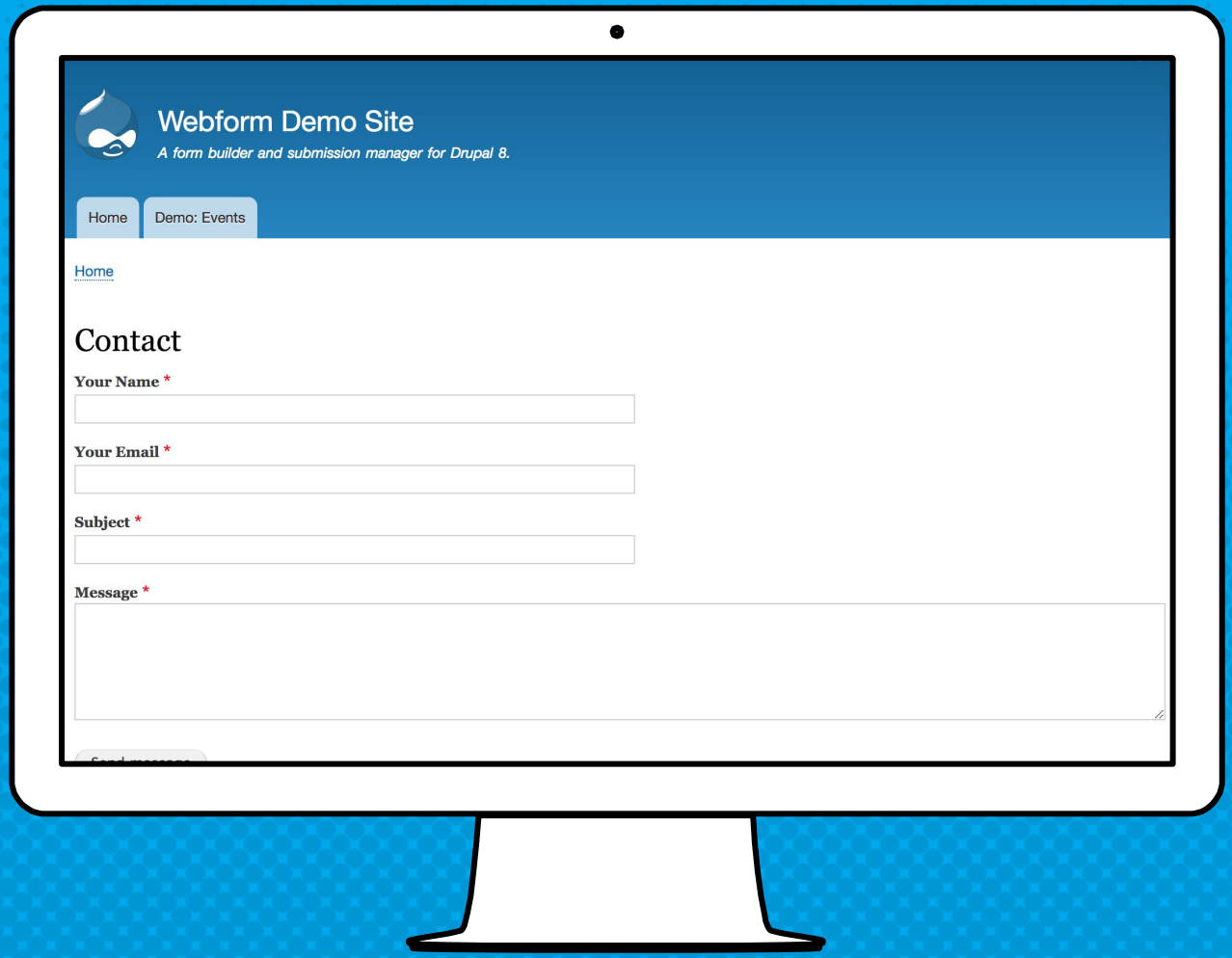
EVERYTHING IN
DRUPAL 8 IS
AN ENTITY
(OR A PLUGIN)

What is an entity?

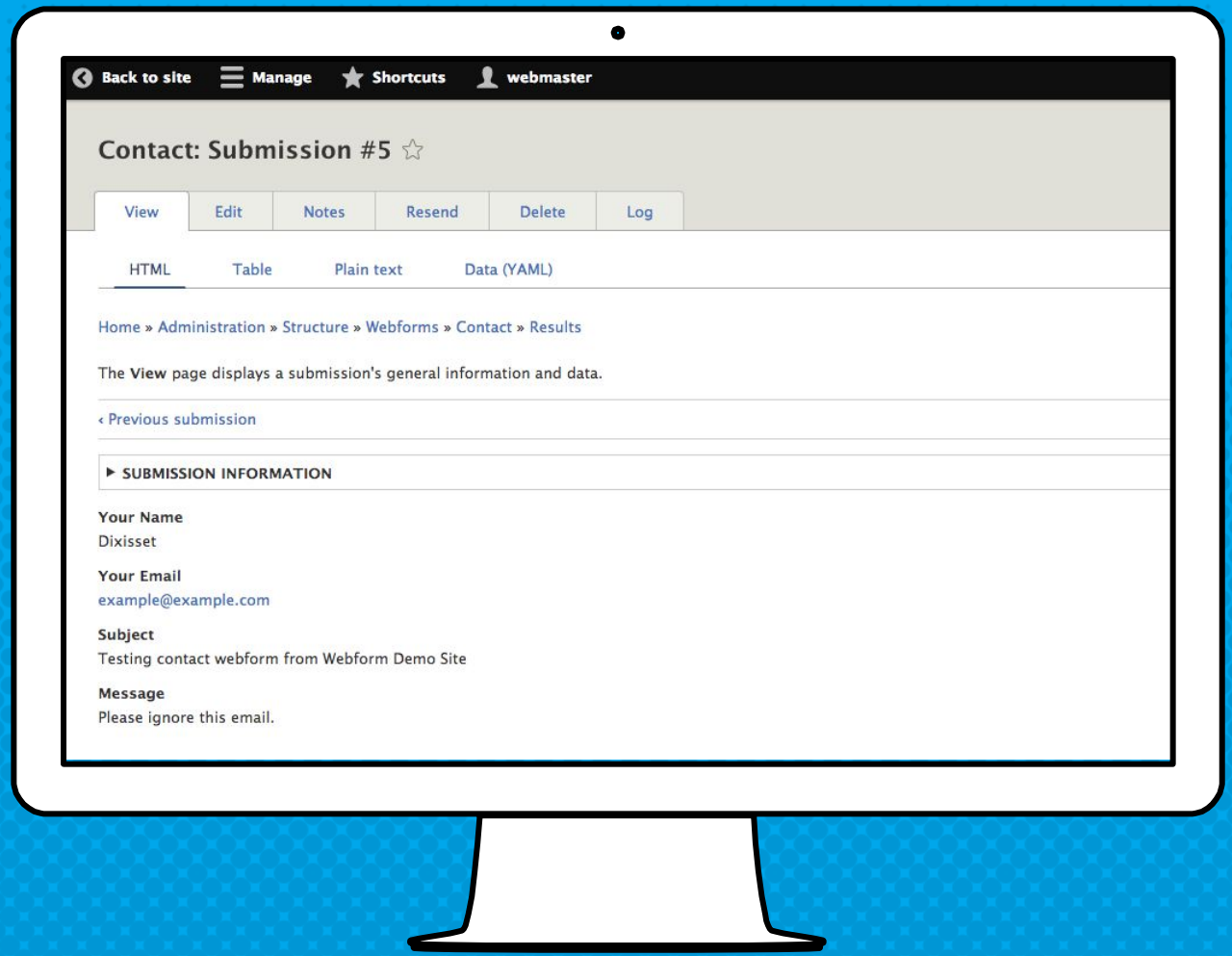
Any defined chunk of data in Drupal. This includes things like nodes, users, taxonomy terms, files, etc. Contributed modules can define custom entities. Each entity type can have multiple bundles.

-- https://dgo.to/937

THIS IS A WEBFORM ENTITY



THIS IS A WEBFORM SUBMISSION ENTITY



About Webform Entities

- Webforms are config entities (exportable)
- Submissions are content entities (database)

- Webforms do not use Field API
- Submissions use an Entity-attribute-value model

Entity-attribute-value model

Entity–attribute–value model (EAV) is a data model to encode, in a space-efficient manner, entities where the number of attributes (properties, parameters) that can be used to describe them is potentially vast, but the number that will actually apply to a given entity is relatively modest.

-- https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model

A SIMPLE WAY
TO STORE
A LOT OF DATA


<demo>
Webform Entities
Overview: /devel/entity/info
Export: …/webform/manage/contact/export
API: /webform/contact/api

Additional Resources

- Introduction to Entity API in Drupal 8 | Drupal
  http://dgo.to/2078191
- What Are Drupal Entities? | Drupalize.me
  https://drupalize.me/videos/what-are-drupal-entities
- Entities 101: Understanding Data Structures in Drupal
  https://youtu.be/LT83PfumjPU

DISCOVERING THE SOURCE ENTITY

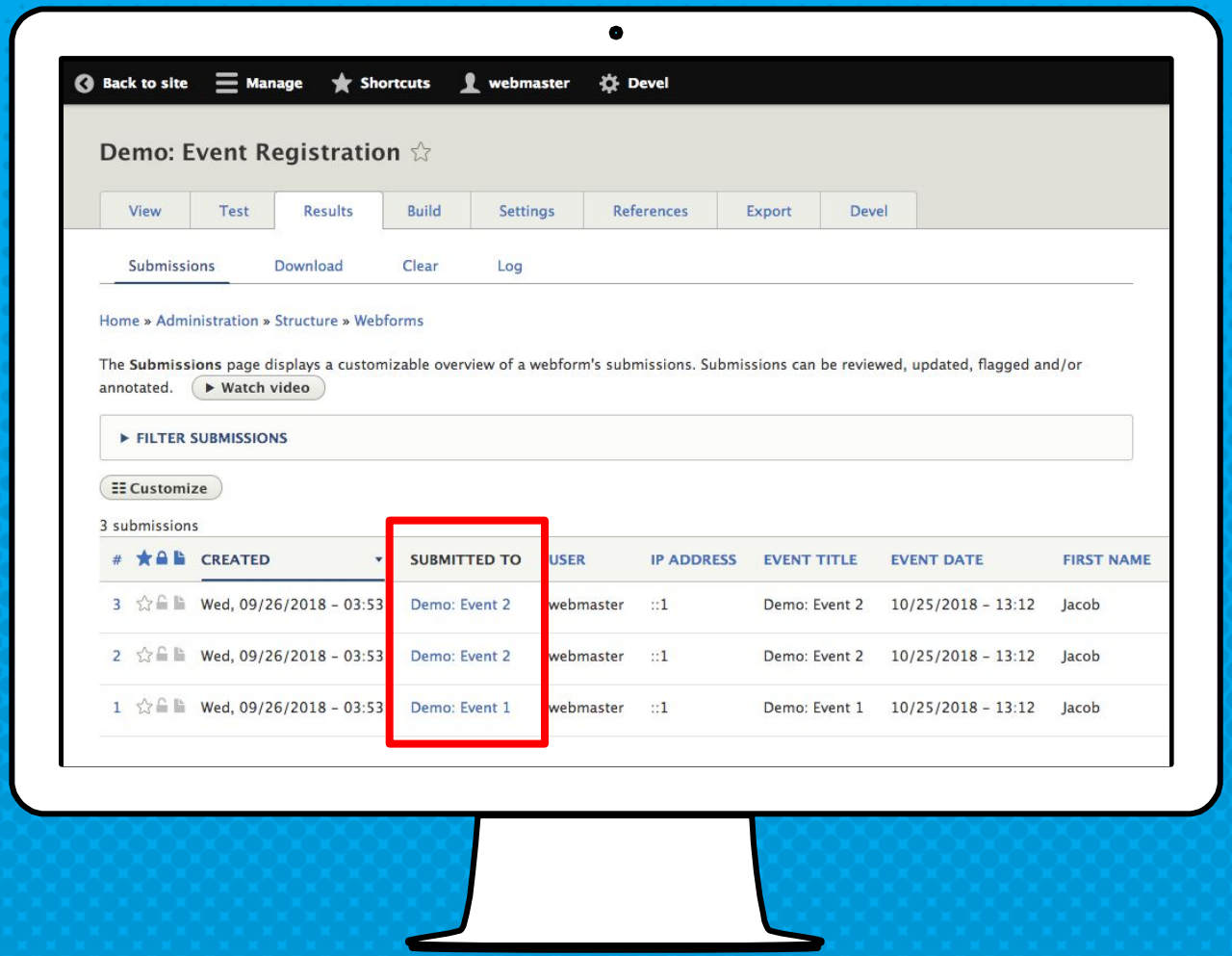
A SOURCE ENTITY
TRACKS & CREATES
A RELATIONSHIP TO
THE DRUPAL ENTITY
FROM WHICH A WEBFORM
WAS SUBMITTED

Source entity overview

- Allows a webform to be reused multiple times
- Determined via the current route or query string

- Webform nodes use source entities
- Webform blocks use source entities
- Even paragraphs use source entities

THESE ARE SOURCE ENTITIES



The source entity can be used to track...

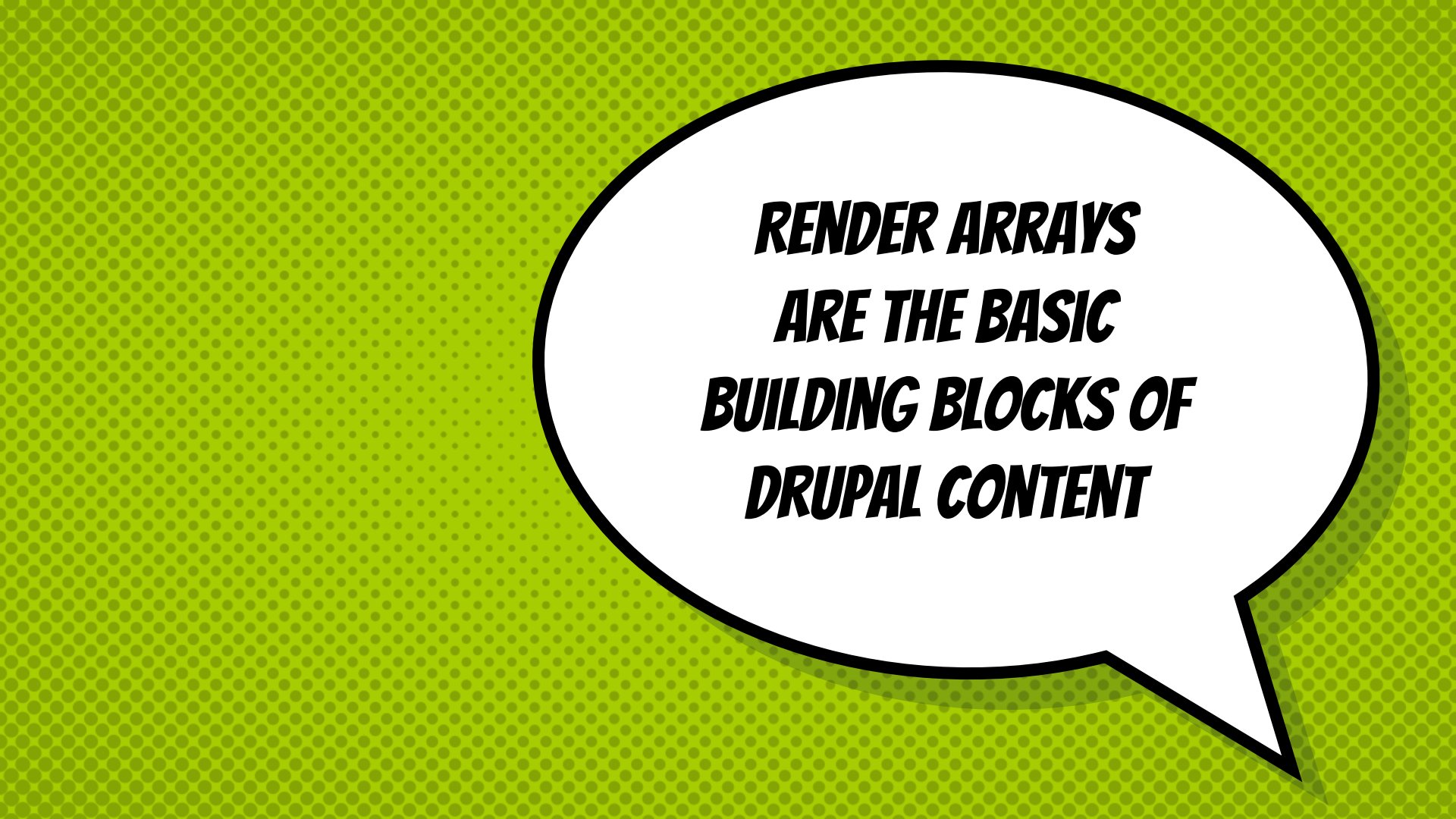
- **Site Feedback**
  A form that tracks which page the comments are related to.
- **Event Registration**
  A registration form that tracks which event a user has registered for.
- **Application Evaluation**
  An evaluation form attached to applications.

```
@see \Drupal\webform\Plugin\WebformSourceEntity\QueryStringWebformSourceEntity

public function getSourceEntity(array $ignored_types) {
  // Get and check source entity type.
  $source_entity_type = $this->request->query->get('source_entity_type');
  if (!$source_entity_type ||
    !$this->entityTypeManager->hasDefinition($source_entity_type)) {
    return NULL;
  }

  // Get and check source entity id.
  $source_entity_id = $this->request->query->get('source_entity_id');
  if (!$source_entity_id) {
    return NULL;
  }

  // Get and check source entity.
  $source_entity = $this->entityTypeManager
    ->getStorage($source_entity_type)->load($source_entity_id);

  return ($source_entity) ? $source_entity : NULL;
}
```

```
@see \Drupal\webform\Plugin\WebformSourceEntityInterface

/**
 * Interface of a webform source entity plugin.
 */
interface WebformSourceEntityInterface
  extends PluginInspectionInterface {

  /**
   * Detect and return a source entity from current context.
   *
   * @param string[] $ignored_types
   *   Entity types that may not be used as a source entity.
   *
   * @return \Drupal\Core\Entity\EntityInterface|null
   *   Source entity or NULL when no source entity is found.
   */
  public function getSourceEntity(array $ignored_types);

}
```

UNDERSTANDING FORM API (FAPI)

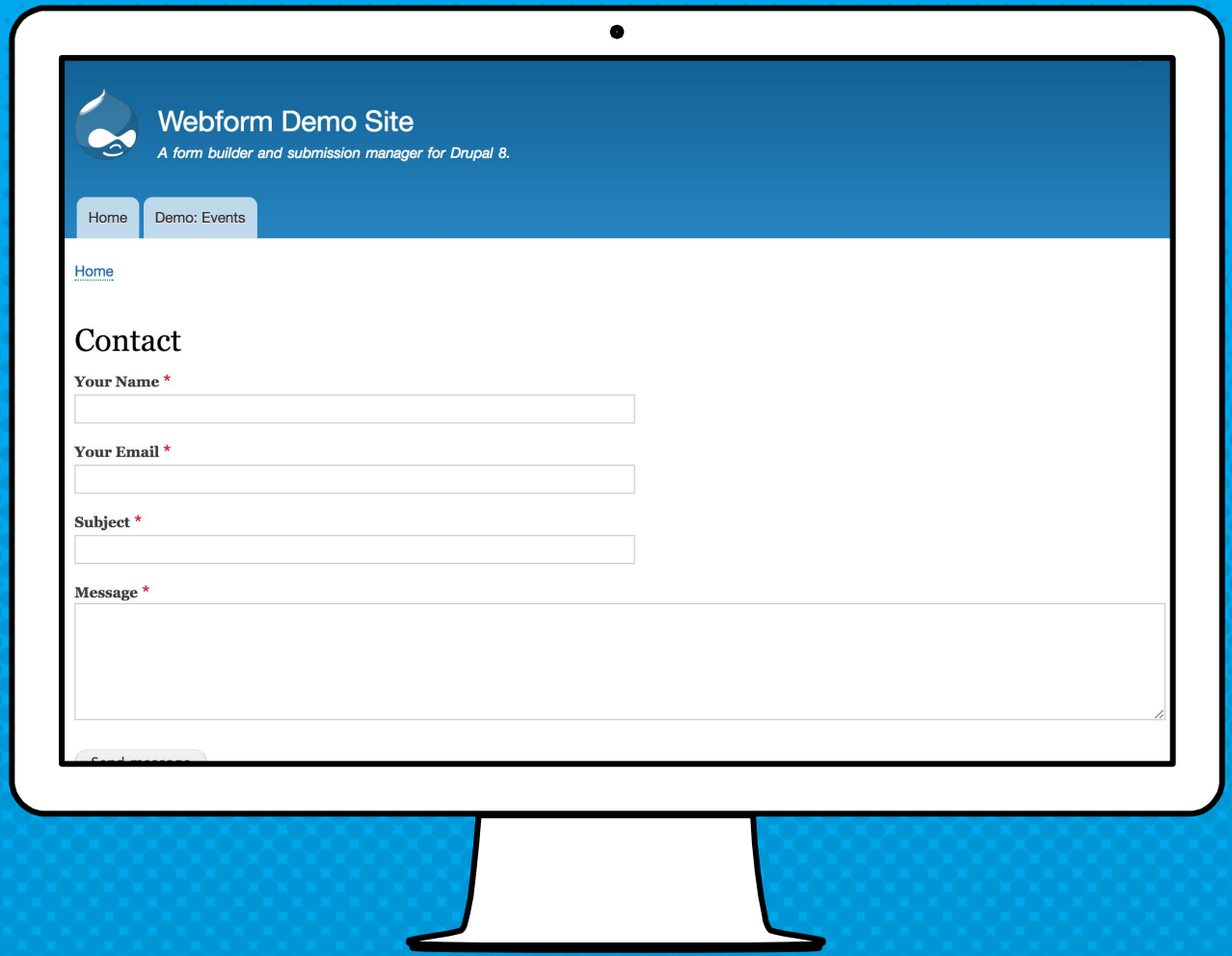
WEBFORMS ARE
RENDER ARRAYS
WHICH CONTAIN
ELEMENTS THAT
BUILD, VALIDATE, & SUBMIT
FORM VALUES
USING DRUPAL'S
FORM API (FAPI)


RENDER ARRAYS
ARE THE BASIC
BUILDING BLOCKS OF
DRUPAL CONTENT

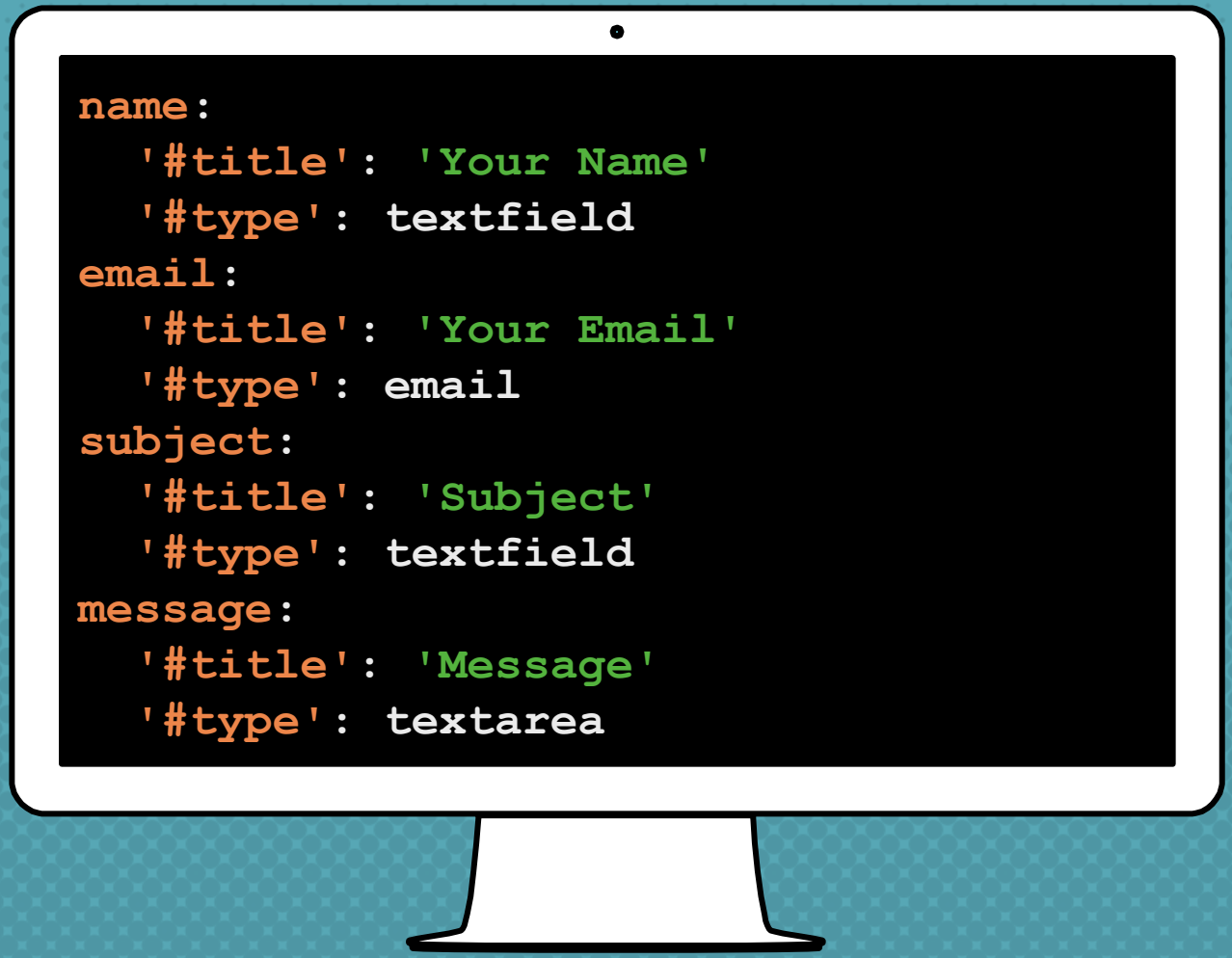
Drupal's Form API Overview

- An element is anything displayed on a page
- An input is an element that collects data
- A composite is a group of elements
- A form is collections of elements and inputs

THIS IS A FORM

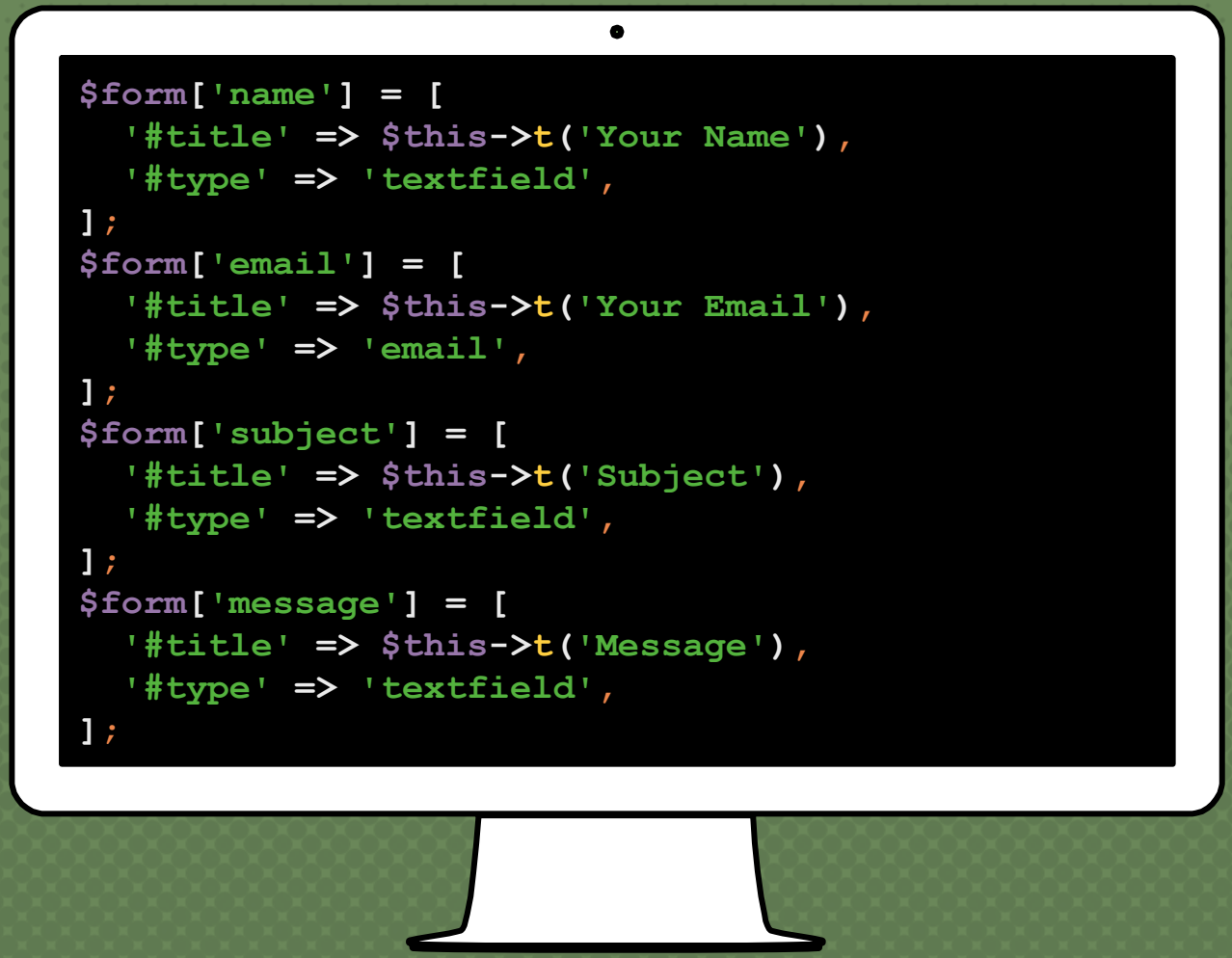


THIS IS A RENDER ARRAY AS YAML



```yaml
name:
  '#title': 'Your Name'
  '#type': textfield
email:
  '#title': 'Your Email'
  '#type': email
subject:
  '#title': 'Subject'
  '#type': textfield
message:
  '#title': 'Message'
  '#type': textarea
```

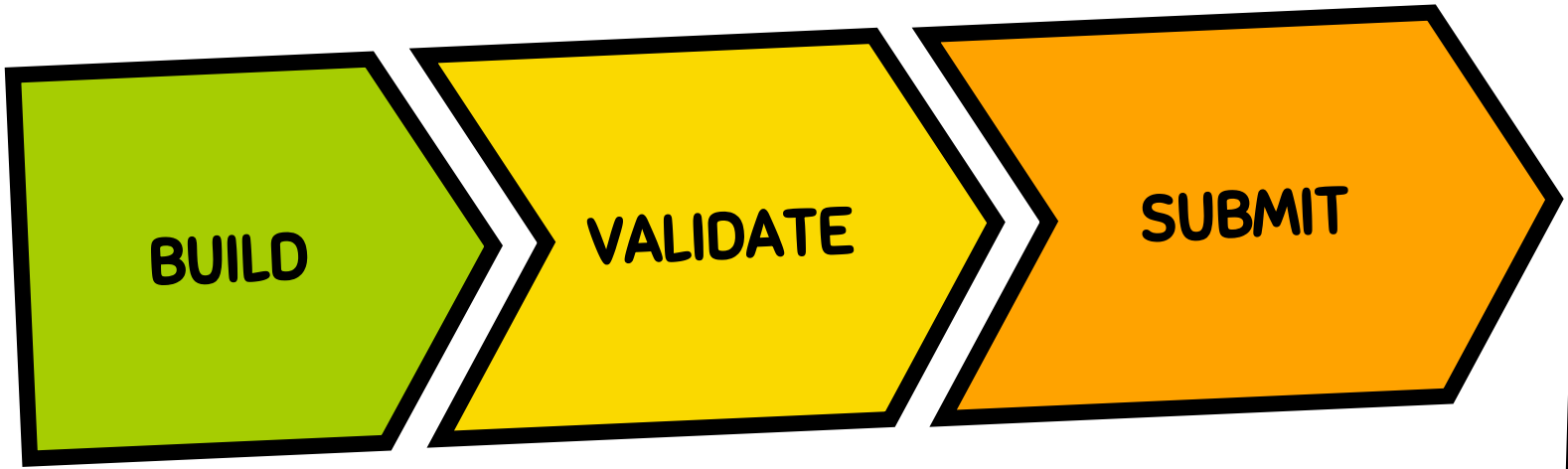
THIS IS A RENDER ARRAY AS PHP



```php
$form['name'] = [
  '#title' => $this->t('Your Name'),
  '#type' => 'textfield',
];
$form['email'] = [
  '#title' => $this->t('Your Email'),
  '#type' => 'email',
];
$form['subject'] = [
  '#title' => $this->t('Subject'),
  '#type' => 'textfield',
];
$form['message'] = [
  '#title' => $this->t('Message'),
  '#type' => 'textfield',
];
```

```
@see \Drupal\system\Form\SiteInformationForm

class SiteInformationForm extends ConfigFormBase {
 public function buildForm(array $form, FormStateInterface $form_state) {
   $site_config = $this->config('system.site');
   $form['site_information'] = [
     '#type' => 'details',
     '#title' => t('Site details'),
   ];
   $form['site_information']['site_name'] = [
     '#type' => 'textfield',
     '#title' => t('Site name'),
     '#default_value' => $site_config->get('name'),
     '#required' => TRUE,
   ];
   $form['site_information']['site_slogan'] = [
     '#type' => 'textfield',
     '#title' => t('Slogan'),
     '#default_value' => $site_config->get('slogan'),
   ];
   return parent::buildForm($form, $form_state);
 }
}
```

How Drupal handles a form



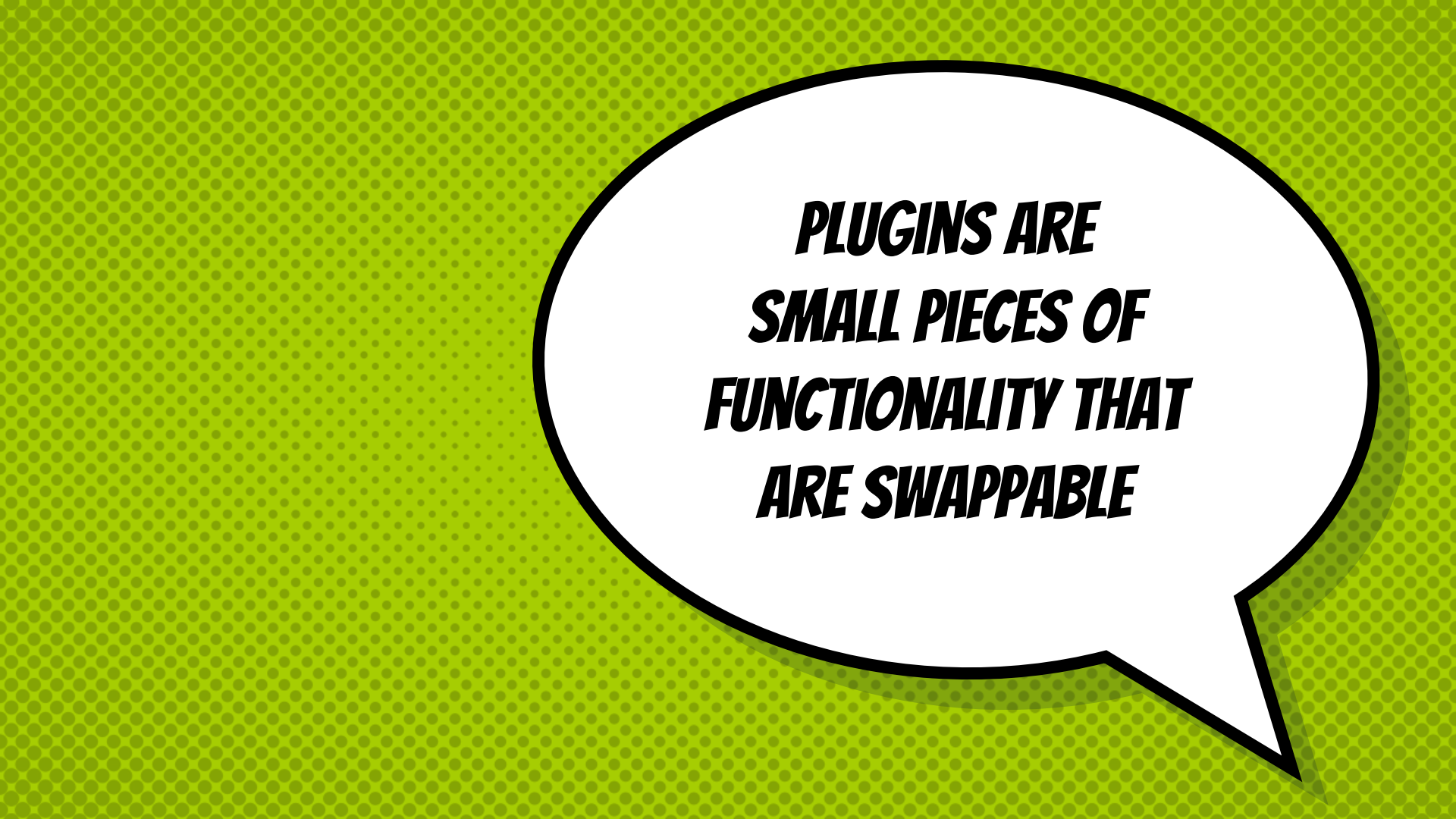
```
@see \Drupal\Core\Form\FormInterface

interface FormInterface {
  /**
   * Returns a unique string identifying the form.
   */
  public function getFormId();

  /**
   * Form constructor.
   */
  public function buildForm(array $form, FormStateInterface $form_state);

  /**
   * Form validation handler.
   */
  public function validateForm(array &$form, FormStateInterface $form_state);

  /**
   * Form submission handler.
   */
  public function submitForm(array &$form, FormStateInterface $form_state);
}
```

Additional Resources

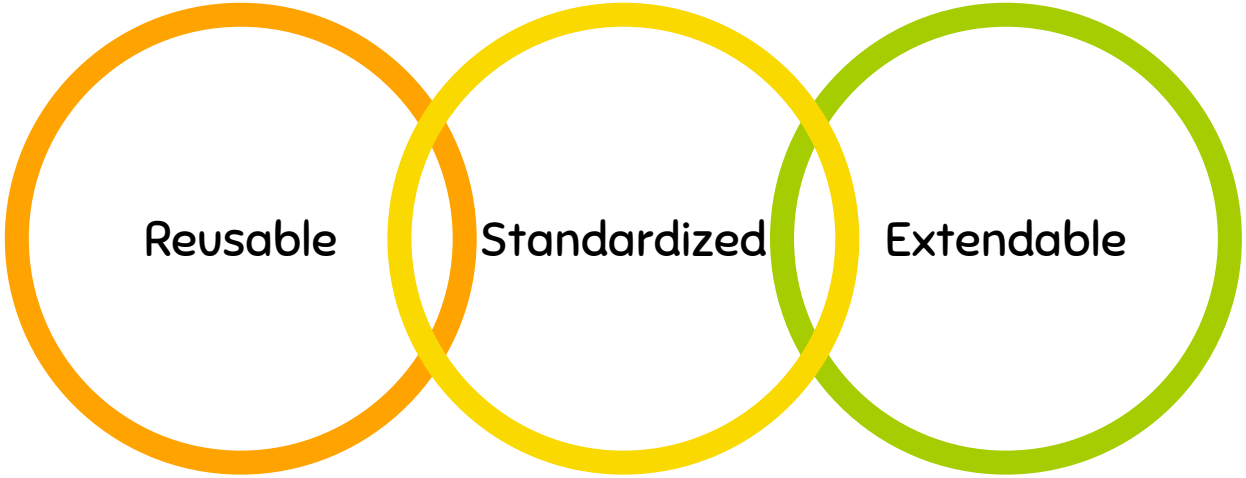
- Form API | Drupal.org
  http://dgo.to/2817929
- Forms (Form API) | Drupalize.me
  https://drupalize.me/topic/forms-form-api
- Examples for Developers
  https://www.drupal.org/project/examples

CREATING FORM ELEMENTS

A FORM ELEMENT IS DEFINED USING A RENDER ARRAY, WHICH IS PROCESSED BY A RENDER ELEMENT PLUGIN, WHICH CREATES AN INPUT ON A FORM


PLUGINS ARE
SMALL PIECES OF
FUNCTIONALITY THAT
ARE SWAPPABLE

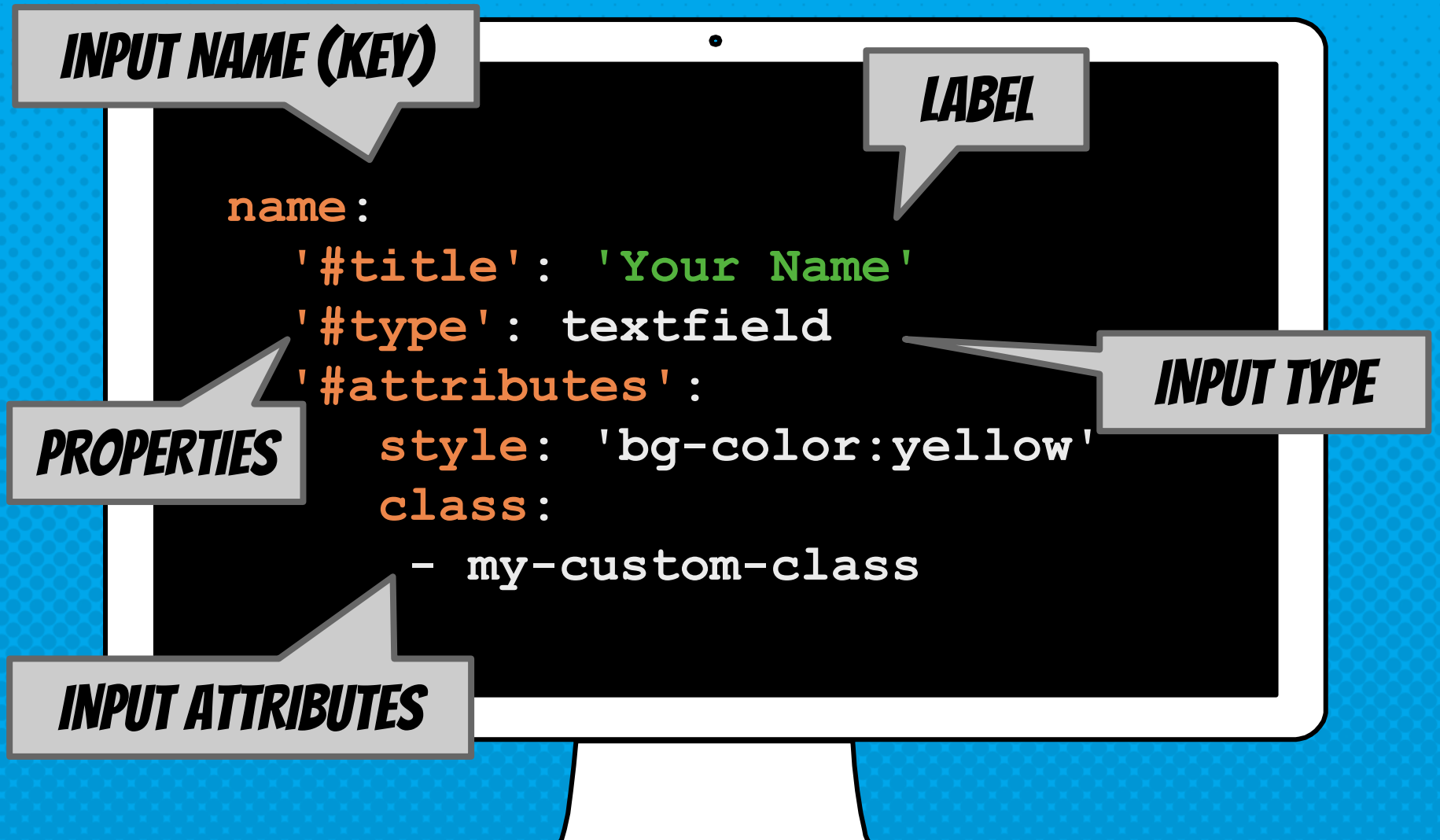
PLUGINS ARE...



Form elements overview

- Form elements are plugins
- Form elements extend render elements

- Element properties begin with a hash (#)
- Elements must have keys


INPUT NAME (KEY)
LABEL
name:
'#title': 'Your Name'
'#type': textfield
'#attributes':
style: 'bg-color:yellow'
class:
- my-custom-class
PROPERTIES
INPUT TYPE
INPUT ATTRIBUTES


FORM ELEMENT
Your Name
XHTML MARKUP
<div class="js-form-item form-item js-form-type-textfield
form-type-textfield js-form-item-name form-item-name">
<label for="edit-name">Your Name</label>
<input style="background-color:yellow"
class="my-custom-class form-text"
data-drupal-selector="edit-name"
type="text"
id="edit-name"
name="name"
value=""
size="60"
maxlength="255">

LABEL
INPUT ATTRIBUTES
INPUT TYPE
INPUT NAME (KEY)

Form element plugin methods

× Define properties	`FormElement::getInfo`
× Set input value	`FormElement::valueCallback`
× Build input	`FormElement::buildElementName`
× Process input	`FormElement::processElementName`
× Render input	`FormElement::preRenderElementName`
× Validate value	`FormElement::validateElementName`

Form element tips

- Copy and extend existing elements
- Use #element_validate to alter form values
- For composite elements you must use #tree

```php
// @see \Drupal\Core\Render\Element\Textfield

class Textfield extends FormElement {

  public function getInfo() {}

  public static function valueCallback(&$element, $input,
    FormStateInterface $form_state) {
    if ($input !== FALSE && $input !== NULL) {
      return str_replace(["\r", "\n"], '', $input);
    }
    return NULL;
  }

  public static function preRenderTextfield($element) {
    $element['#attributes']['type'] = 'text';
    Element::setAttributes($element, ['id', 'name', 'value', 'size',
      'maxlength', 'placeholder']);
    static::setAttributes($element, ['form-text']);
    return $element;
  }
}
```

```
@see \Drupal\Core\Render\Element\FormElementInterface

/**
 * Provides an interface for form element plugins.
 */
interface FormElementInterface extends ElementInterface {

  /**
   * Determines how user input is mapped to an
   * element's #value property.
   */
  public static function valueCallback(&$element, $input,
    FormStateInterface $form_state);

}
```

Additional resources

- Render arrays | Drupal.org
  http://dgo.to/2456267
- Form Element Reference | Drupalize.me
  https://drupalize.me/tutorial/form-element-reference
- Form and render elements | Drupal API
  https://api.drupal.org/api/drupal/elements

CREATING WEBFORM ELEMENTS

WEBFORM ELEMENTS ARE
WRAPPERS THAT ENHANCE
DRUPAL FORM ELEMENTS

WebForm Element Plugins Overview

- Requires a corresponding FormElement
- Both plugins must use the same plugin ID (#type)
- Handles everything related to an element
- Base classes help organize related elements
- Traits also help organize related behaviors

WebformElement plugin methods

- Defines default properties — `WebformElement::getDefaultProperties`
- Prepares an element — `WebformElement::prepare`
- Determine behaviors — `WebformElement::hasMultipleValues`
- Display submission value — `WebformElement::buildHtml`
- Exports values — `WebformElement::getTableColumn`
- Builds configuration form — `WebformElement::form`


<demo>
Webform Elements
Overview: …/reports/webform-plugins/elements
Modules: webform_example_element.module
webform_example_composite.module

```
@see \Drupal\webform\Plugin\WebformElementInterface

interface WebformElementInterface {
  function getDefaultProperties();
  // Inspection.
  function isInput(array $element);
  function isContainer(array $element);
  function isComposite();
  function supportsMultipleValues();
  // Element processing.
  function initialize(array &$element);
  function prepare(array &$element, $webform_submission);
  function finalize(array &$element, $webform_submission);
  // Submission rendering.
  function buildHtml(array $element, $webform_submission);
  function buildText(array $element, $webform_submission);
  // Entity operations.
  function preCreate(array &$element, array $values);
  function postLoad(array &$element, $webform_submission);
  function postSave(array &$element, $webform_submission);
  function postDelete(array &$element, $webform_submission);
  // Configuration form.
  function form(array $form, $form_state);
}
```

```
@see \Drupal\webform_example_composite\Element\WebformExampleComposite

class WebformExampleComposite extends WebformCompositeBase {

  public function getInfo() {
    return parent::getInfo() + ['#theme' => 'webform_example_composite'];
  }

  public static function getCompositeElements(array $element) {
    $elements = [];
    $elements['first_name'] = [
      '#type' => 'textfield',
      '#title' => t('First name'),
    ];
    $elements['last_name'] = [
      '#type' => 'textfield',
      '#title' => t('Last name'),
    ];
    return $elements;
  }
}
```

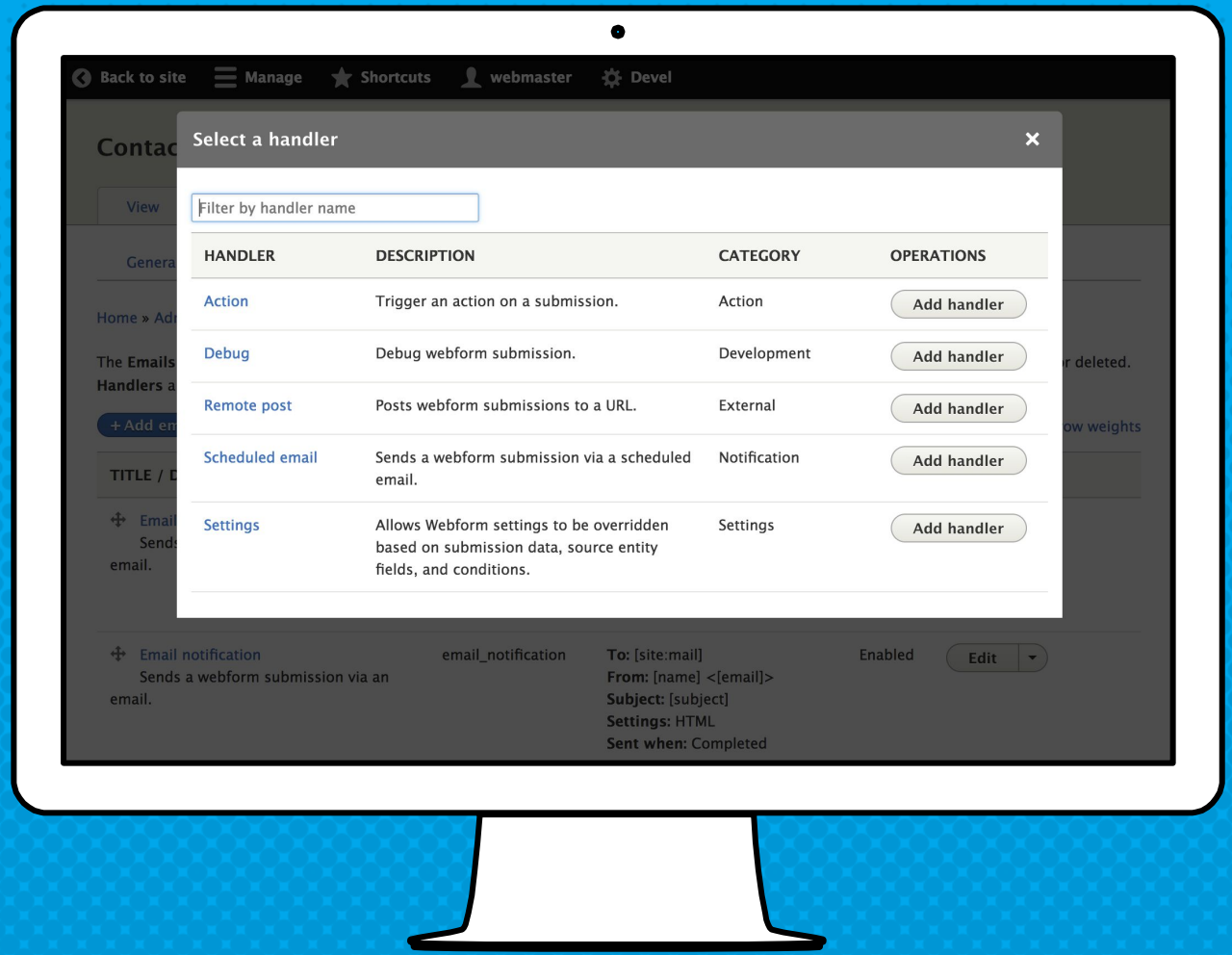
Creating Custom Webform Elements

- Create a custom module
- Extend or copy existing form element plugin
- Build a test webform
- Define webform element plugin
- Test webform integration
- Write tests

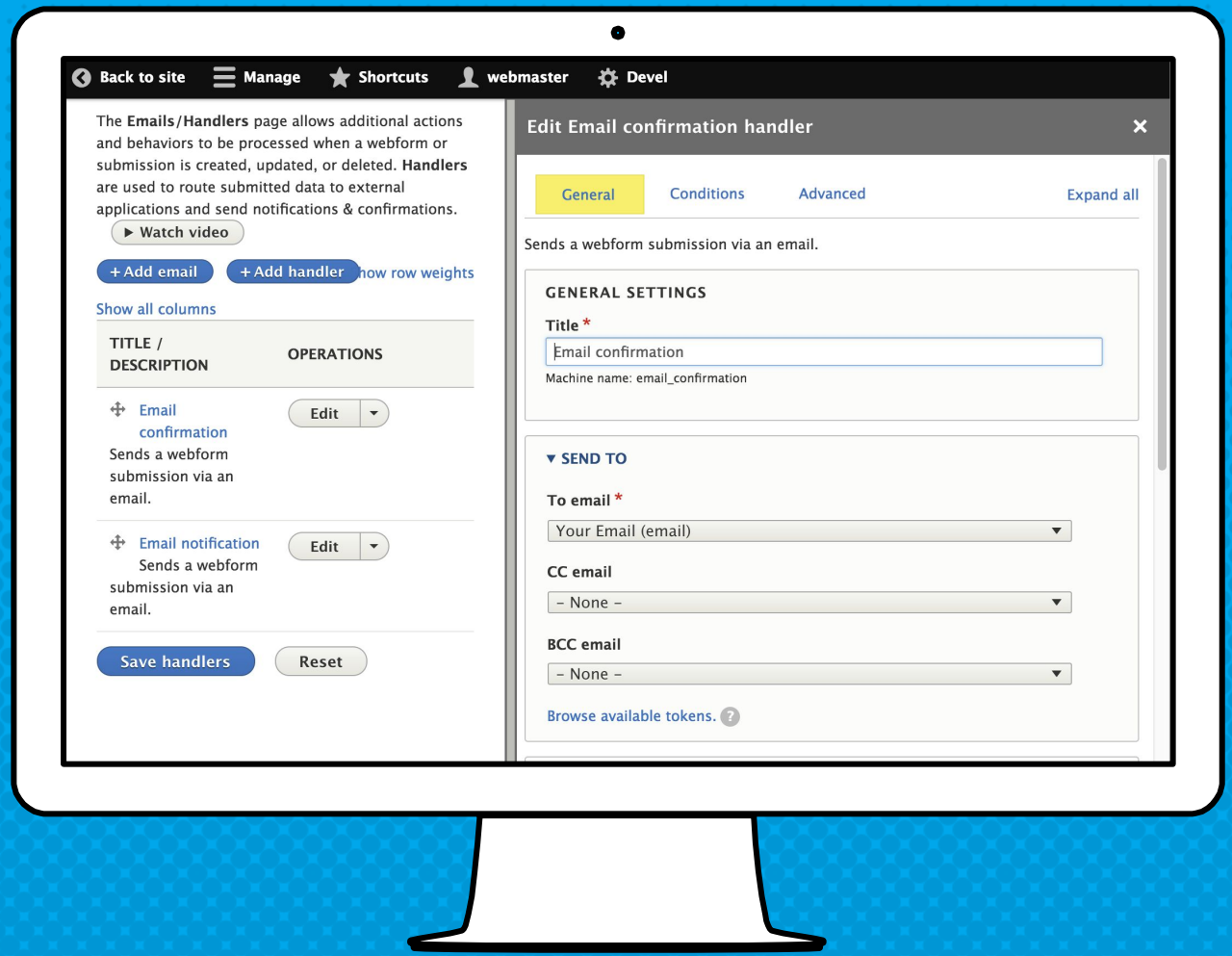
Implementing Webform Handlers

WEBFORM HANDLER PLUGINS ARE USED TO ROUTE SUBMITTED DATA TO APPLICATIONS & SEND EMAIL NOTIFICATIONS

THESE ARE WEBFORM HANDLERS



THIS IS AN EMAIL HANDLER



WEBFORM HANDLER PLUGIN OVERVIEW

- Contains methods that act like hooks
- Reacts to a submission's state
- Supports conditional logic

WEBFORM HANDLER PLUGIN METHODS

- Configuration settings — `WebformHandler::getConfiguration`
- Override settings — `WebformHandler::overrideSettings`
- Alter forms & elements — `WebformHandler::alterElements`
- Entity operations — `WebformHandler::postSave`
- Element operations — `WebformHandler::createElement`
- Handler operations — `WebformHandler::createHandler`


<demo>
Webform Handlers
Overview: …/reports/webform-plugins/handlers
Test: …/contact/settings/handlers
Modules: webform_example_handler.module

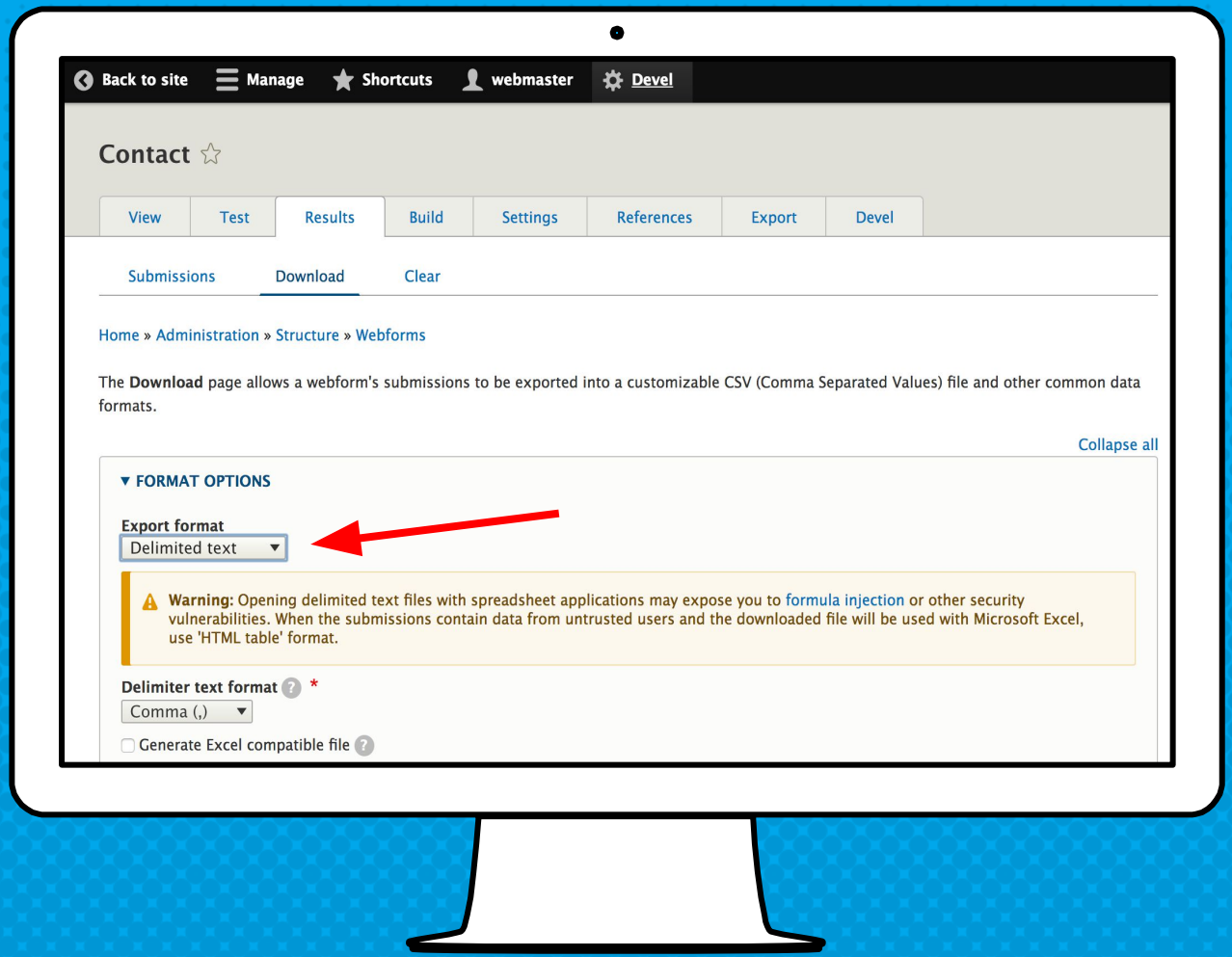
```
@see \Drupal\webform\Plugin\WebformHandlerInterface

interface WebformHandlerInterface {
  // Configuration form.
  function defaultConfiguration()
  function buildConfigurationForm(array $form, $form_state);
  function validateConfigurationForm(array &$form, $form_state);
  function submitConfigurationForm(array &$form, $form_state);
  // Elements.
  function alterElements(array &$elements, $webform);
  // Settings.
  function overrideSettings(array &$settings, $webform_submission);
  // Submission form.
  function alterForm(array &$form, $form_state, $webform_submission);
  function validateForm(array &$form, $form_state, $webform_submission);
  function submitForm(array &$form, $form_state, $webform_submission);
  function confirmForm(array &$form, $form_state, $webform_submission);
  // Operations.
  function postLoad($webform_submission);
  function postSave($webform_submission);
  function postDelete($webform_submission);
}
```

```
@see \Drupal\webform_example_handler\Plugin\WebformHandler\ExampleWebformHandler

class ExampleWebformHandler extends WebformHandlerBase {

  function defaultConfiguration() {
    return [
     'message' => 'This is a custom message.',
     'debug' => FALSE,
    ];
  }

  function confirmForm(array &$form, $form_state, $webform_submission){
    $message = $this->configuration['message'];
    $message = $this->tokenManager->replace($message,
      $webform_submission);

    $this->messenger()
      ->addStatus(Markup::create($message), FALSE);
  }

}
```

Extending Webform Exporters

WEBFORM EXPORTER PLUGINS ARE USED TO DOWNLOAD SUBMISSIONS INTO SPREADSHEETS & OTHER APPLICATIONS

THESE ARE EXPORTERS



Webform Exporter Plugin Overview

- Always extend an existing Webform Exporter
- Use DelimitedWebformExporter for CSV files
- Drush command is available for automation

WebForm Exporter Plugin Methods

- Configuration settings — `WebformExporter::getConfiguration`
- Writing data — `WebformExporter::writeHeader`
- File naming — `WebformExporter::getBaseFileName`


<demo>
Webform Exporters
Overview: …/reports/webform-plugins/exports
Test: …/contact/results/download

```
@see \Drupal\webform\Plugin\WebformExporterInterface

interface WebformExporterInterface {

  /**
   * Write header to export.
   */
  public function writeHeader();

  /**
   * Write submission to export.
   */
  public function writeSubmission($webform_submission);

  /**
   * Write footer to export.
   */
  public function writeFooter();
}
```

```php
@see \Drupal\webform\Plugin\WebformExporter\TableWebformExporter

class TableWebformExporter extends TabularBaseWebformExporter {

  public function writeSubmission($webform_submission) {
    // Get the submission's data as a simple associative array.
    $record = $this->buildRecord($webform_submission);
    // Build the table data.
    $row = [];
    foreach ($record as $item) {
      $row[] = '<td>' . nl2br(htmlentities($item)) . '</td>';
    }
    // Write the table row.
    $file_handle = $this->fileHandle;
    fwrite($file_handle, '<tr valign="top">');
    fwrite($file_handle, implode(PHP_EOL, $row));
    fwrite($file_handle, '</tr>');
  }

}
```

ARE WE DONE?

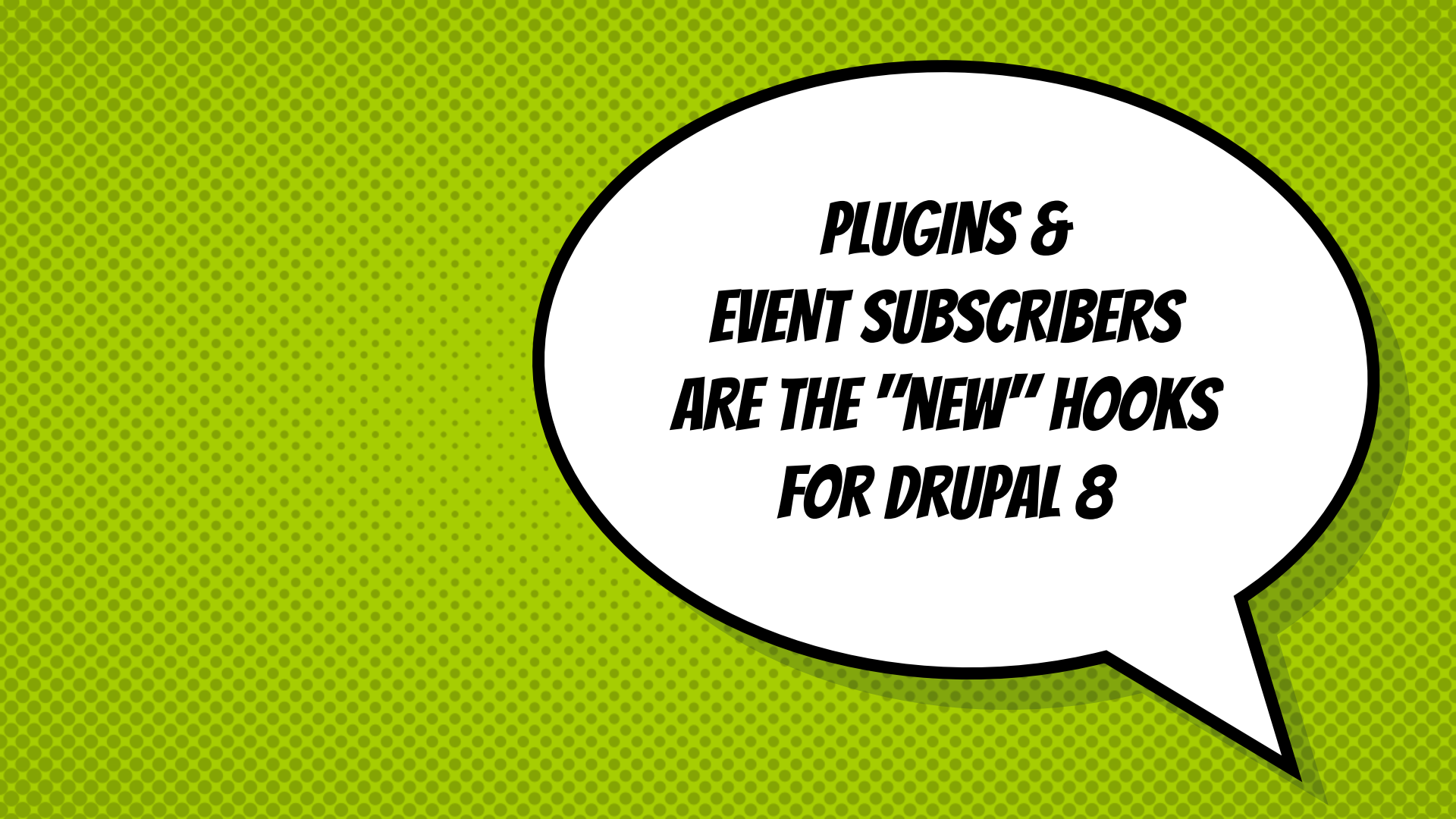
Implementing Webform Hooks

HOOKS ARE

FUNCTIONS THAT DEFINE OR ALTER BEHAVIORS

Webform & Drupal hooks overview

- Handler plugins and hooks are very similar
- Handlers are applied to a single form
- Hooks can be applied to all forms
- All entity hooks are applicable to webforms


PLUGINS &
EVENT SUBSCRIBERS
ARE THE "NEW" HOOKS
FOR DRUPAL 8

Form hooks

hook_webform_submission_form_alter

Element hooks

hook_webform_element_alter

Option hooks

hook_webform_options_alter

Handler hooks

hook_webform_handler_invoke_alter

Entity hooks

hook_webform_submission_insert
hook_webform_submission_load
hook_webform_submission_save
hook_webform_submission_delete
etc…

More hooks...

hook_webform_libraries_info_alter
hook_webform_access_rules_alter

@see webform.api.php

```
@see webform.api.php

/**
 * Act on a webform handler when a method is invoked.
 *
 * @param \Drupal\webform\Plugin\WebformHandlerInterface $handler
 *   A webform handler attached to a webform.
 * @param string $method_name
 *   The invoked method name converted to snake case.
 * @param array $args
 *   Argument being passed to the handler's method.
 */
function hook_webform_handler_invoke_alter($handler, $method_name, array &$args) {
  $webform_id = $handler->getWebform()->id();
  $handler_id = $handler->getHandlerId();
  // If contact webform email confirmation has been saved.
  if ($webform_id === 'contact'
    && $handler_id === 'email_confirmation'
    && $method_name === 'post_save') {
    $webform_submission = $handler->getWebformSubmission();
    // Do something with the webform submission.
  }
}
```

ARE WE
THERE YET?

ADDITIONAL WEBFORM RESOURCES

Connecting with me

- Jacob Rockowitz (Blog)
  http://jrockowitz.com

- jrockowitz on Drupal.org
  https://www.drupal.org/u/jrockowitz

- jrockowitz on Twitter
  https://twitter.com/jrockowitz

Getting Help & Support

- Drupal Slack #Webform
  https://drupal.slack.com/messages/C78MFLN9K

- Drupal Answers
  https://drupal.stackexchange.com/questions/tagged/webforms

- Webform Issue Queue
  https://www.drupal.org/project/issues/webform?version=8.x

WEBFORM

OPEN COLLECTIVE

What is Open Collective?

Open Collective is a service which allows Open Source projects to transparently collect and distribute funds. Organizations who back an Open Collective will get a receipt for their financial contributions and be able to see exactly how the collected money is being spent.

Why should you support the webform module?

- Marketing – Dedicated logo
- Security – Providing $50 USD thanks
- Training – Offset travel costs
- Accessibility – Professional a11y review
- Support – Priority support (TBD)
- More to come...

JOIN THE WEBFORM MODULE'S OPEN COLLECTIVE

OPENCOLLECTIVE.COM/WEBFORM

GETTING INVOLVED

JOIN US FOR CONTRIBUTION OPPORTUNITIES

Friday, April 12, 2019

Mentored Contribution	First Time Contributor Workshop	General Contribution
9:00–18:00 Room: 602	9:00–12:00 Room: 606	9:00–18:00 Room: 6A

#DRUPALCONTRIBUTIONS

WHAT DID YOU THINK?

Locate this session at the DrupalCon Seattle website:
http://seattle2019.drupal.org/schedule

Take the Survey!
https://www.surveymonkey.com/r/DrupalConSeattle

RALPH SAYS...

THANK YOU!!!

ANY QUESTIONS?