Advanced WebForms

http://bit.ly/advanced-webforms

This work is licensed under a Creative Commons Attribution 4.0 International License.

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.

There are a lot of resources available for learning about the basics of the Webform module

Advanced presentations should teach you a few new things & inspire you to learn more

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

WEBFORM

BASICS

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

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

ONCE AGAIN, THE USE CASE...

<demo>
Overview: /admin/structure/webform
Build: …/webform/manage/contact
Source: …/webform/manage/contact/source
Configure …/webform/manage/contact/settings
Test: /webform/contact/test
Results: …/contact/results/submissions
Download: …/contact/results/download
Submission:…/contact/submission/1

Additional Resources

- Webform Articles
  http://dgo.to/2932764
- Webform Videos
  http://dgo.to/2834424
- Webform Features (Outdated)
  http://dgo.to/2837024

TESTING
WEBFORMS

TESTS CONFIRM

EXPECTATIONS

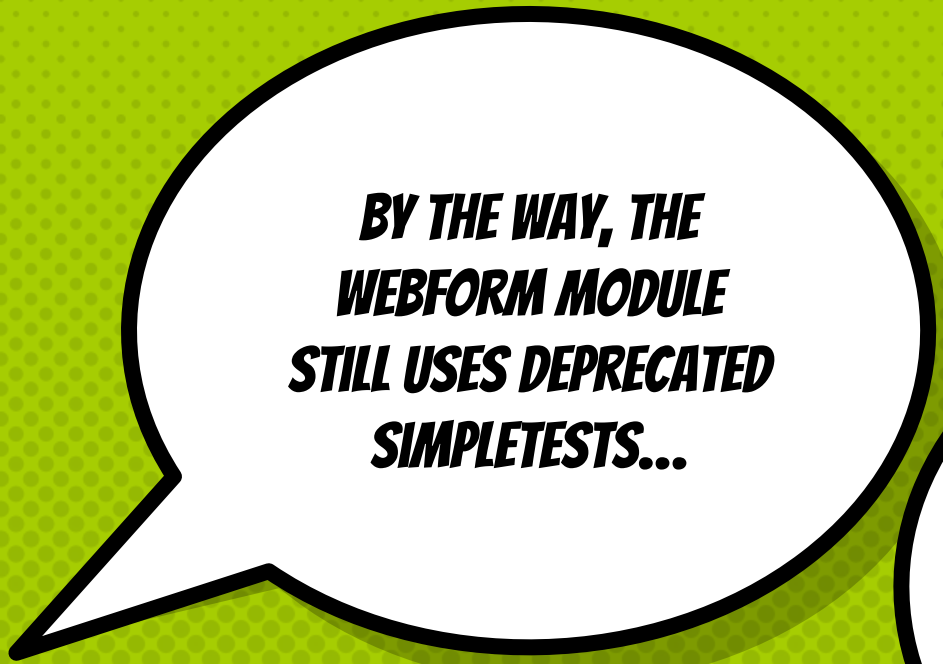
WEBFORM TESTS CONFIRM

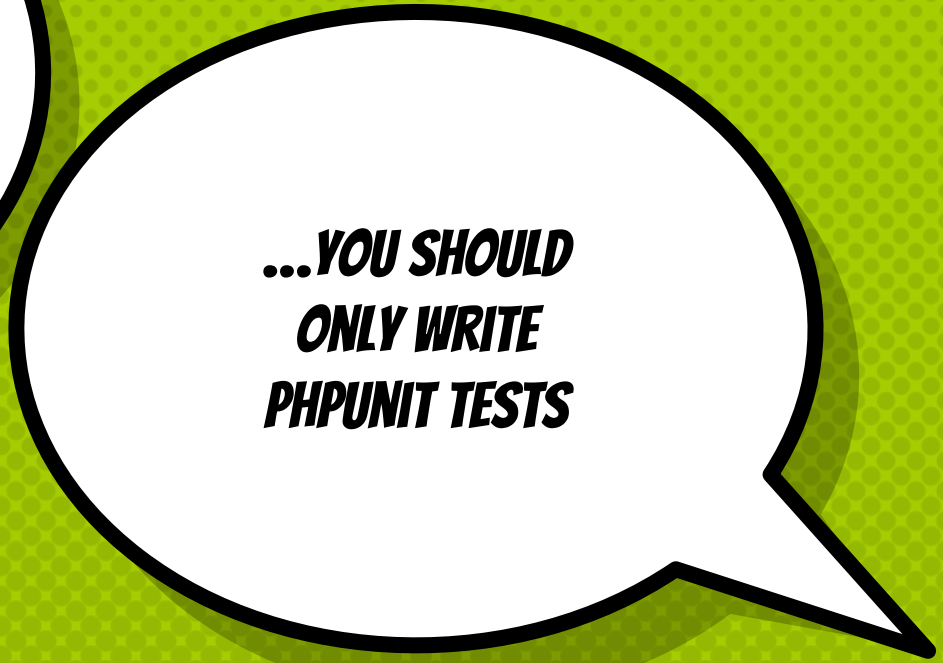
- Rendering: Check an element's markup
- Processing: Check an input's default value
- Validation: Check required error messages
- Settings: Check labels and layout
- Access: Check user access controls

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>
Overview: /admin/structure/webform
Settings:…/src/Tests/Settings
Setting: WebformSettingsConfirmationTest
Elements:…/src/Tests/Element
Element: WebformElementEmailTest
Modules: /tests/modules

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

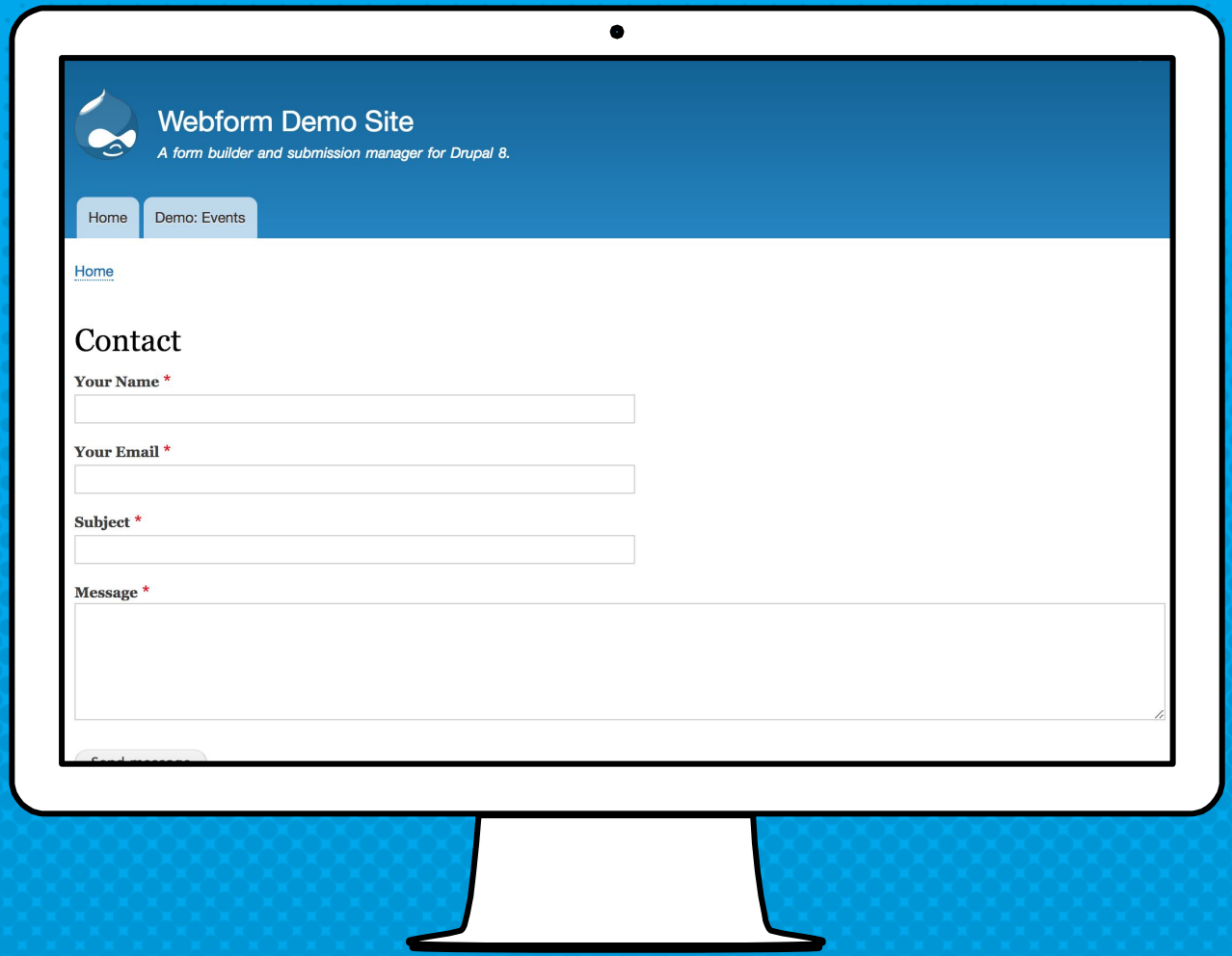
EVERYTHING IN
DRUPAL 8 IS
AN ENTITY
(OR A PLUGIN)

What is an entity?

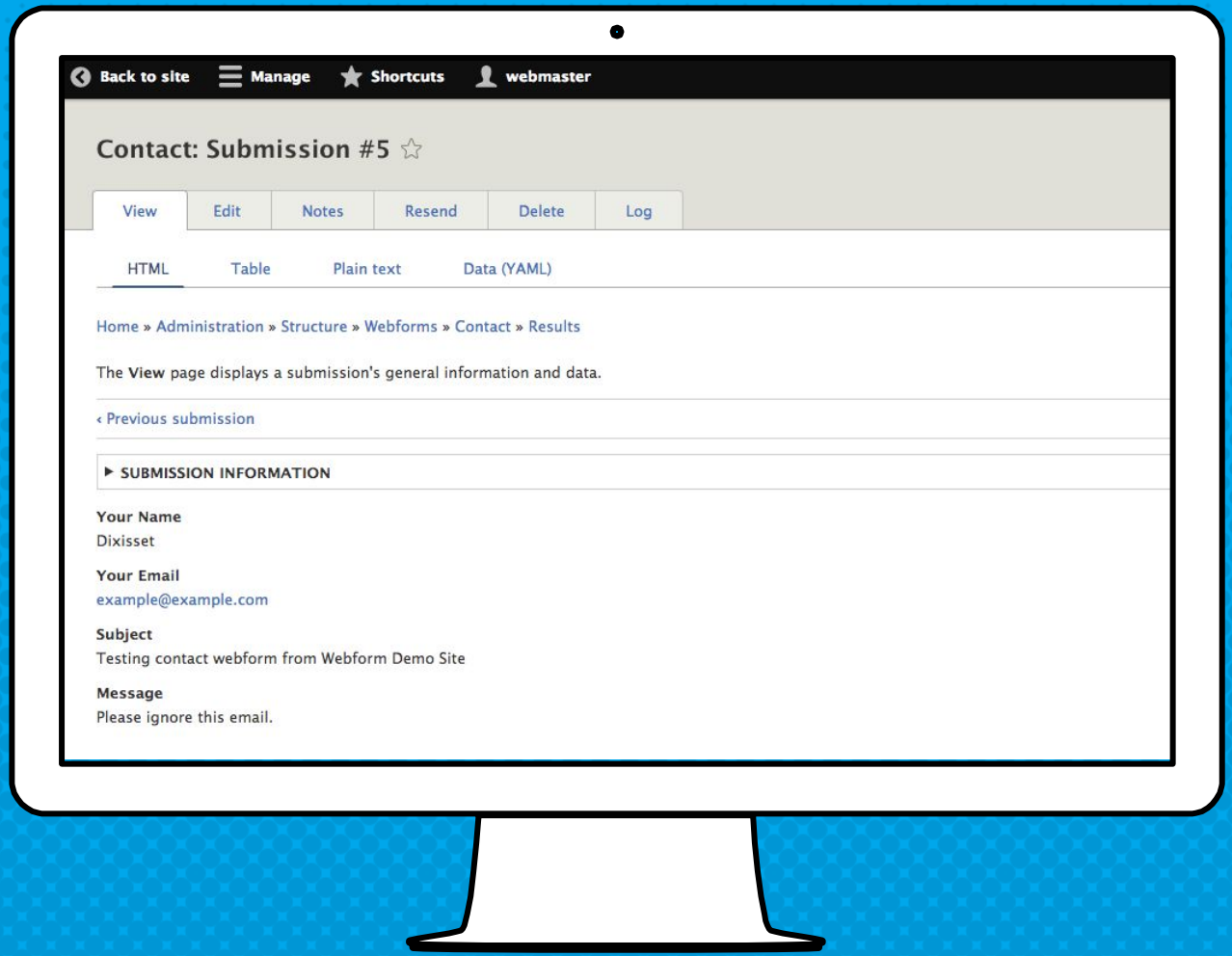
Any defined chunk of data in Drupal. This includes things like nodes, users, taxonomy terms, files, etc. Contributed modules can define custom entities. Each entity type can have multiple bundles.

-- 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.

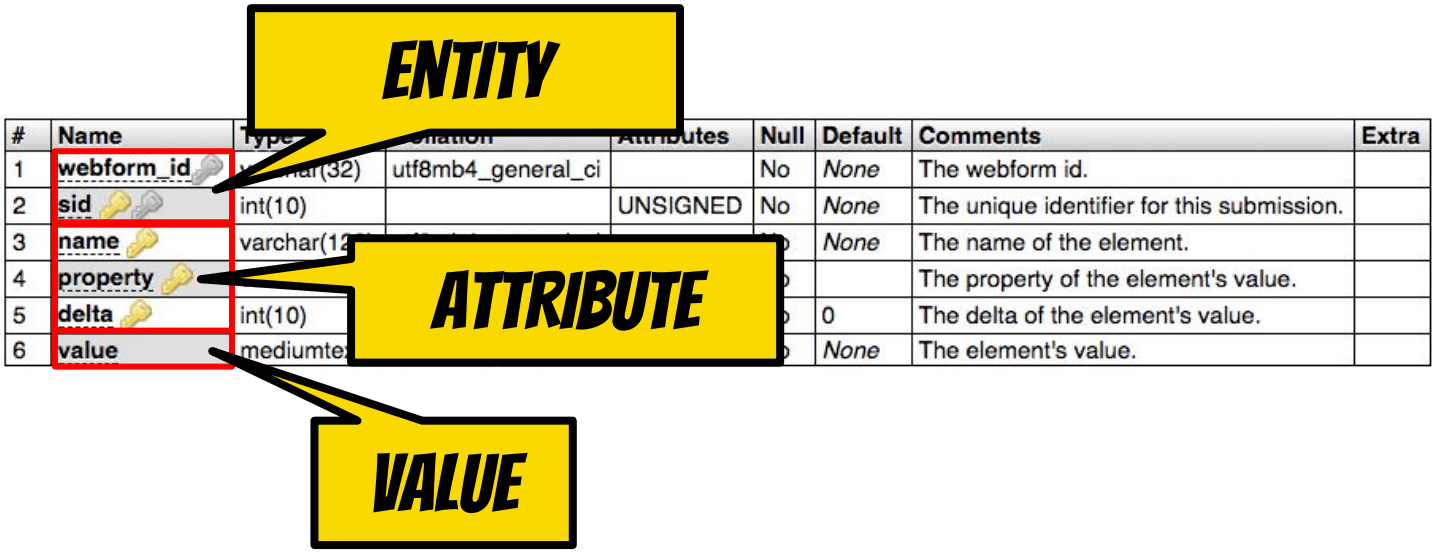
-- https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model

A SIMPLE WAY
TO STORE
A LOT OF DATA

WEBFORM_SUBMISSION

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**sid** 🔑	int(11)			No	*None*		AUTO_INCREMENT
2	**webform_id** 🔗	varchar(32)	ascii_general_ci		No	*None*	The ID of the target entity.	
3	**uuid** 🔗	varchar(128)	ascii_general_ci		No	*None*		
4	**serial**	int(11)			Yes	*NULL*		
5	**token**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
6	**uri**	varchar(2000)	utf8mb4_general_ci		Yes	*NULL*		
7	**created**	int(11)			Yes	*NULL*		
8	**completed**	int(11)			Yes	*NULL*		
9	**changed**	int(11)			Yes	*NULL*		
10	**in_draft**	tinyint(4)			Yes	*NULL*		
11	**current_page**	varchar(128)	utf8mb4_general_ci		Yes	*NULL*		
12	**remote_addr**	varchar(128)	utf8mb4_general_ci		Yes	*NULL*		
13	**uid** 🔗	int(10)		UNSIGNED	Yes	*NULL*	The ID of the target entity.	
14	**langcode**	varchar(12)	ascii_general_ci		Yes	*NULL*		
15	**entity_type**	varchar(32)	ascii_general_ci		Yes	*NULL*		
16	**entity_id**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
17	**sticky**	tinyint(4)			Yes	*NULL*		
18	**notes**	longtext	utf8mb4_general_ci		Yes	*NULL*		

webform_submission_data



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	webform_id	varchar(32)	utf8mb4_general_ci		No	*None*	The webform id.	
2	sid	int(10)		UNSIGNED	No	*None*	The unique identifier for this submission.	
3	name	varchar(12				*None*	The name of the element.	
4	property						The property of the element's value.	
5	delta	int(10)				0	The delta of the element's value.	
6	value	mediumte				*None*	The element's value.	

<demo>
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

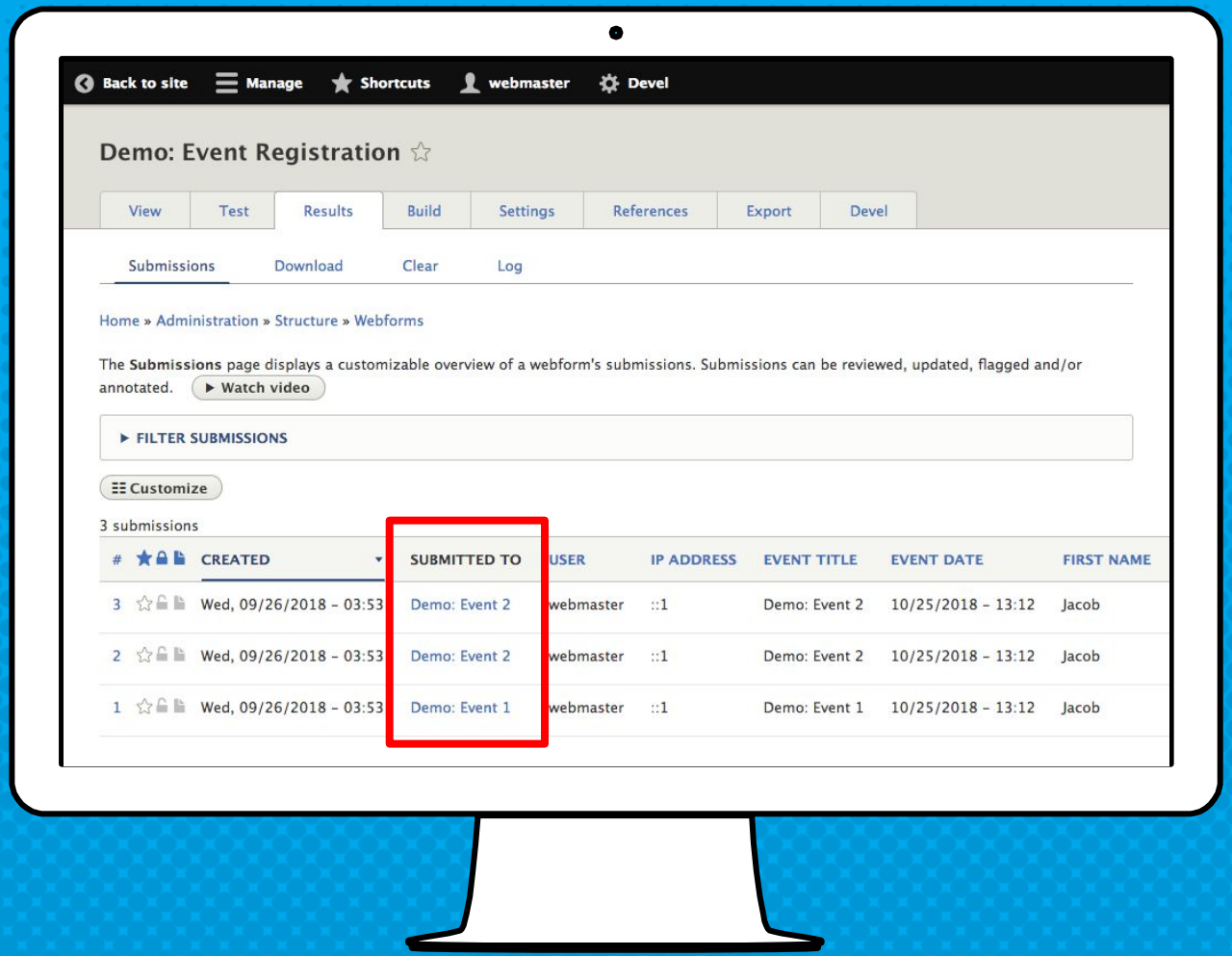
- Webform nodes use source entities
- Webform blocks use source entities
- Even paragraphs use source entities

WEBFORM_SUBMISSION

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra
1	**sid**	int(11)			No	*None*		AUTO_INCREMENT
2	**webform_id**	varchar(32)	ascii_general_ci		No	*None*	The ID of the target entity.	
3	**uuid**	varchar(128)	ascii_general_ci		No	*None*		
4	**serial**	int(11)			Yes	*NULL*		
5	**token**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
6	**uri**	varchar(2000)	utf8mb4_general_ci		Yes	*NULL*		
7	**created**	int(11)			Yes	*NULL*		
8	**completed**	int(11)			Yes	*NULL*		
9	**changed**	in			Yes	*NULL*		
10	**in_draft**	ti			Yes	*NULL*		
11	**current_page**	v			Yes	*NULL*		
12	**remote_addr**	v			Yes	*NULL*		
13	**uid**	in		UNSIGNED	Yes	*NULL*	The ID of the target entity.	
14	**langcode**	var	ascii_general_ci		Yes	*NULL*		
15	**entity_type**	rchar(32)	ascii_general_ci		Yes	*NULL*		
16	**entity_id**	varchar(255)	utf8mb4_general_ci		Yes	*NULL*		
17	**sticky**	tinyint(4)			Yes	*NULL*		
18	**notes**	longtext	utf8mb4_general_ci		Yes	*NULL*		

SOURCE ENTITY

THESE ARE SOURCE ENTITIES



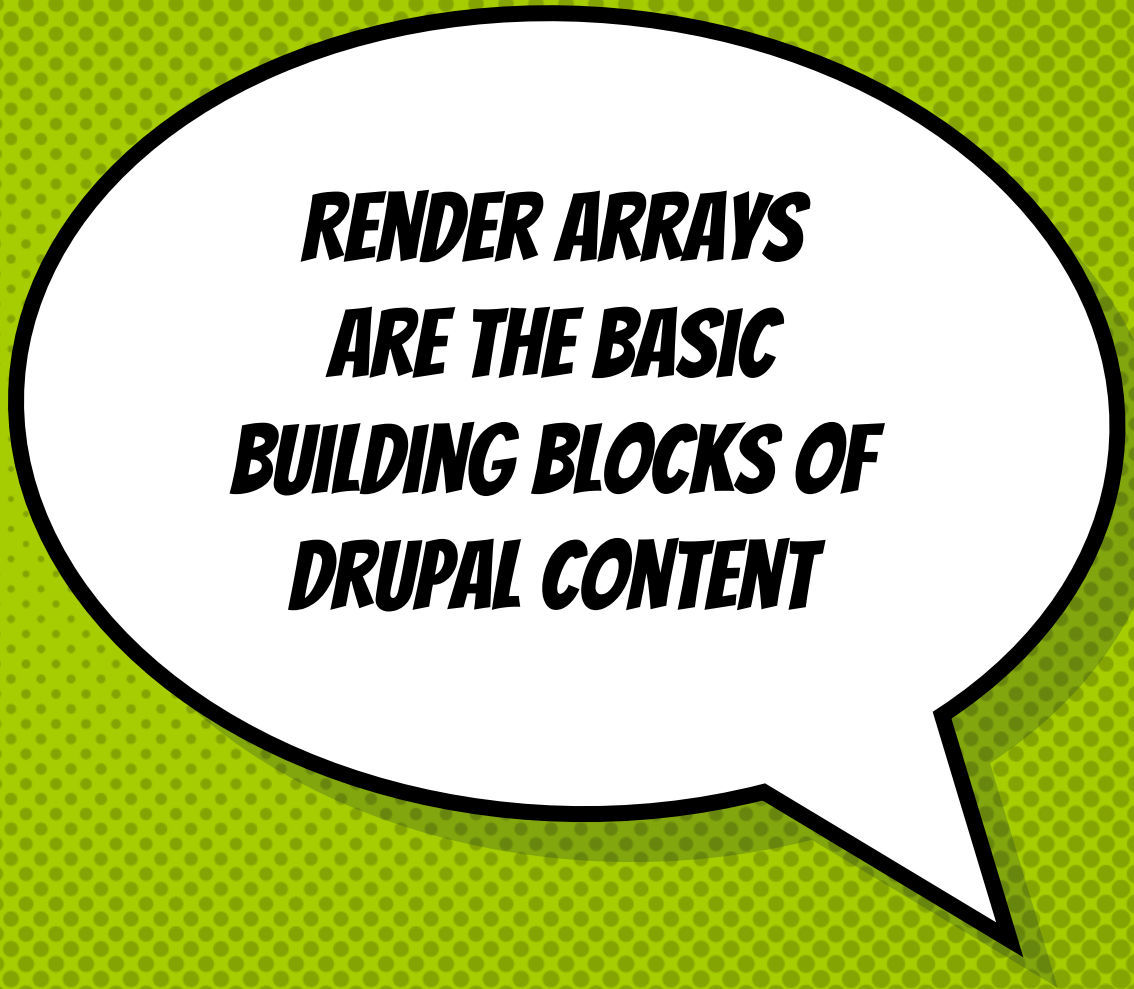
The source entity can be used to track...

- **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.

<demo>
Overview: …/webform/submissions/manage
Node: …/manage/contact/references
Module: webform_node.module
Plugin: RouteParametersWebformSourceEntity
QueryStringWebformSourceEntity
Interface: WebformSourceEntityInterface

Understanding Form API (FAPI)

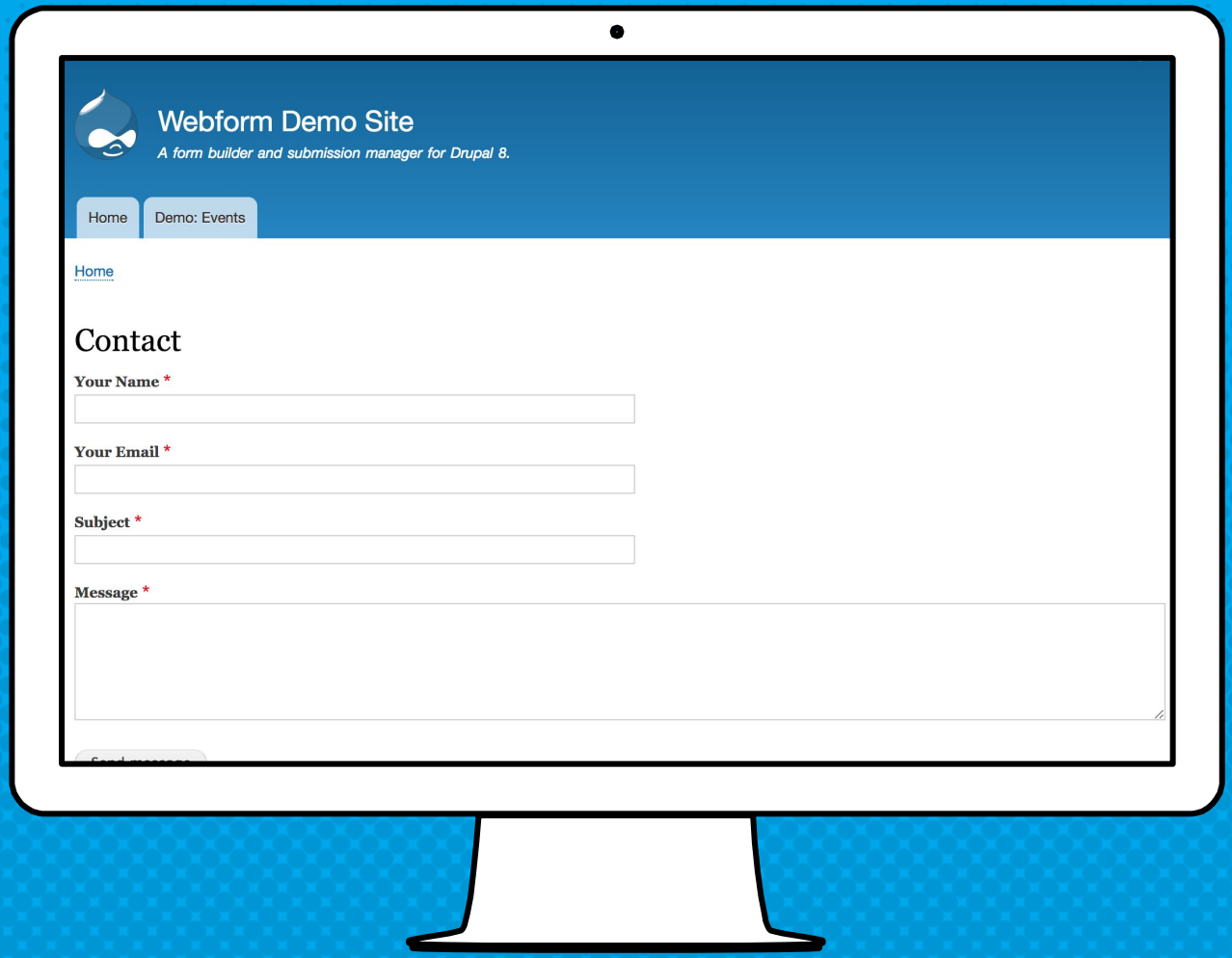
WEBFORMS ARE
RENDER ARRAYS
WHICH CONTAIN
ELEMENTS THAT
BUILD, VALIDATE, & SUBMIT
FORM VALUES
USING DRUPAL'S
FORM API (FAPI)


RENDER ARRAYS
ARE THE BASIC
BUILDING BLOCKS OF
DRUPAL CONTENT

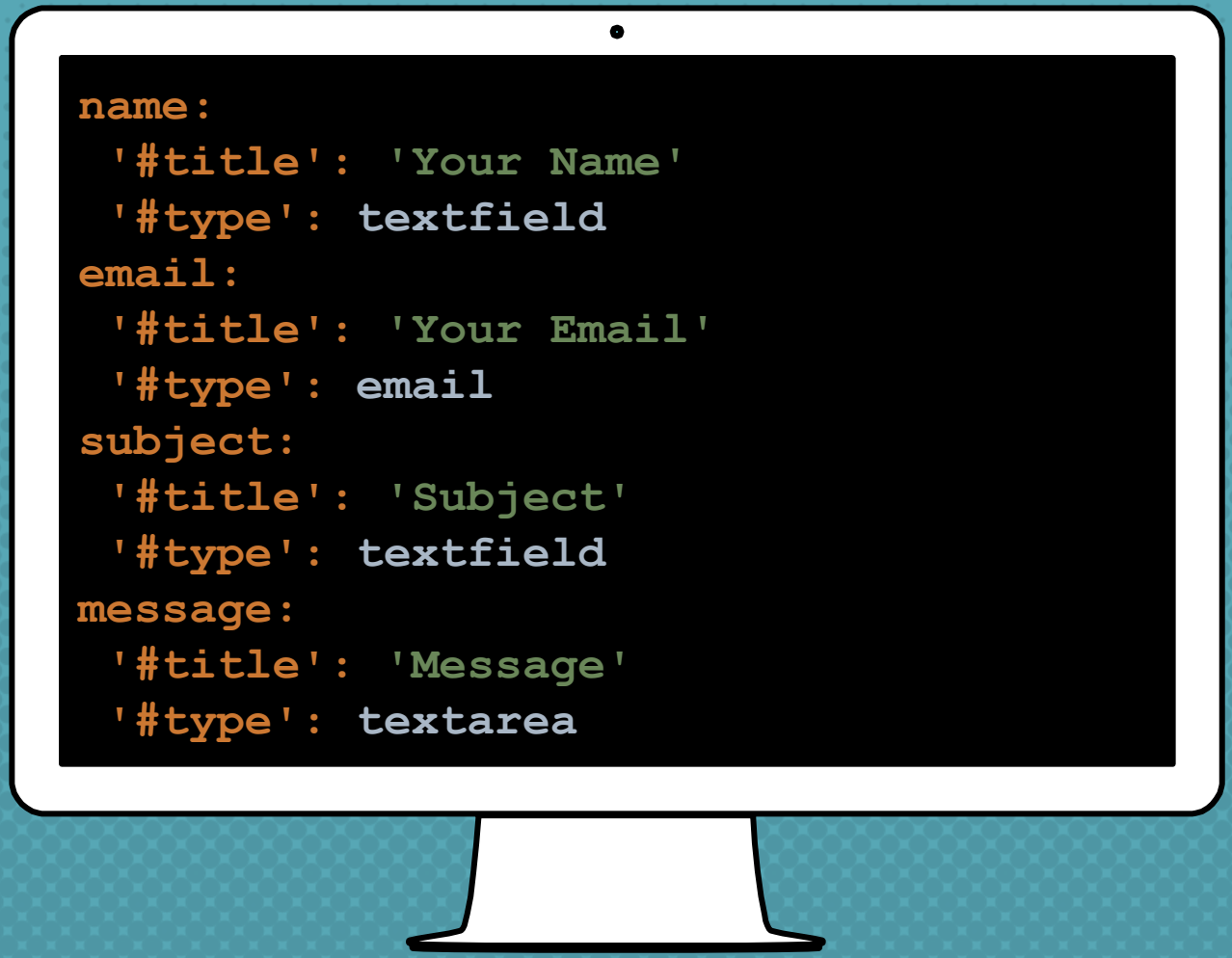
Drupal Form API Overview

- 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

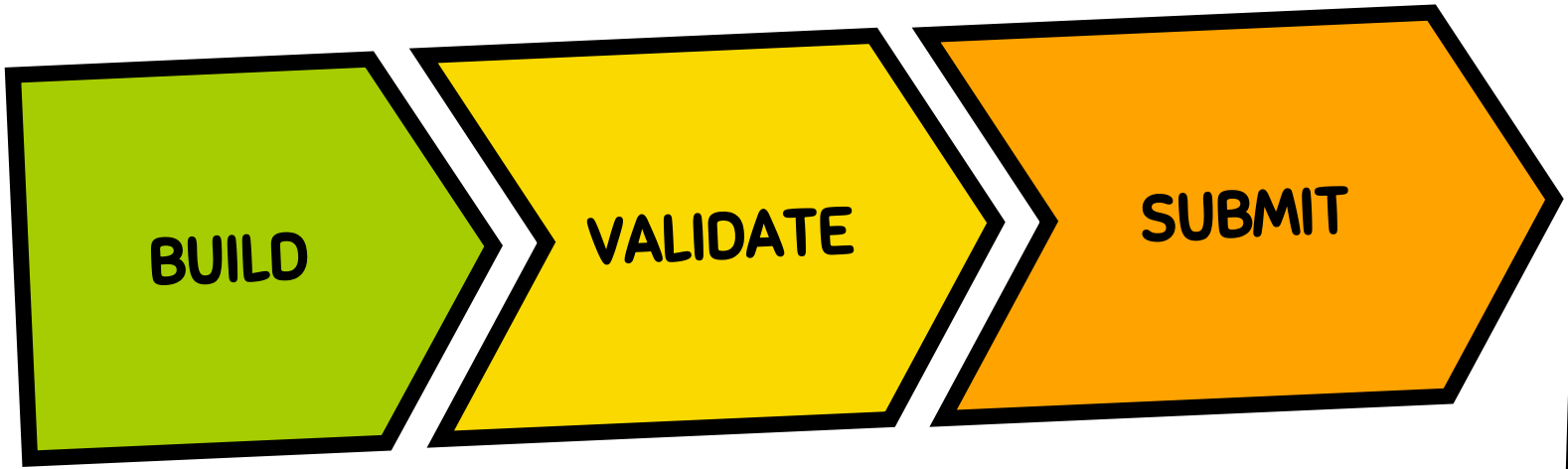


```yaml
name:
  '#title': 'Your Name'
  '#type': textfield
email:
  '#title': 'Your Email'
  '#type': email
subject:
  '#title': 'Subject'
  '#type': textfield
message:
  '#title': 'Message'
  '#type': textarea
```

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',
];
```

How Drupal handles a form



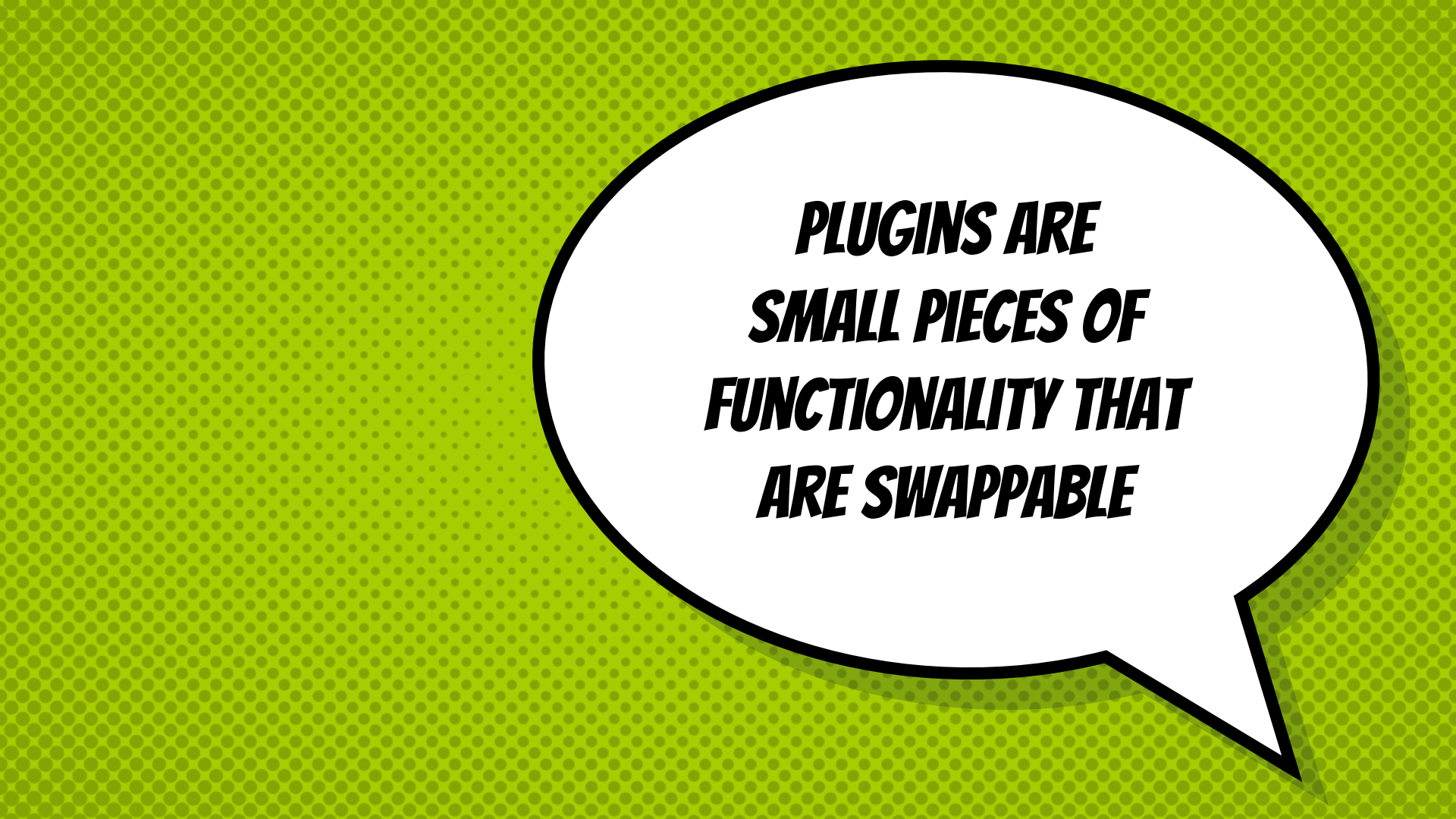
<demo>
Forms: /admin/config
Interface: FormInterface
Examples: http://dgo.to/examples

Additional Resources

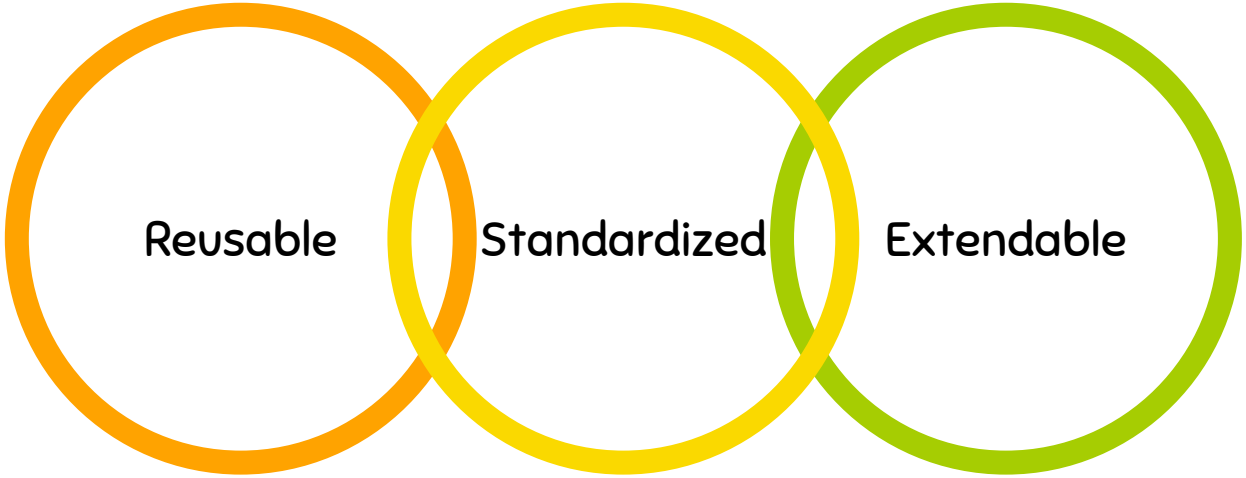
- Form API | Drupal.org
  http://dgo.to/2817929
- Forms (Form API) | Drupalize.me
  https://drupalize.me/topic/forms-form-api
- Form API Life Cycle | Drupalize.me
  https://drupalize.me/tutorial/form-api-life-cycle

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

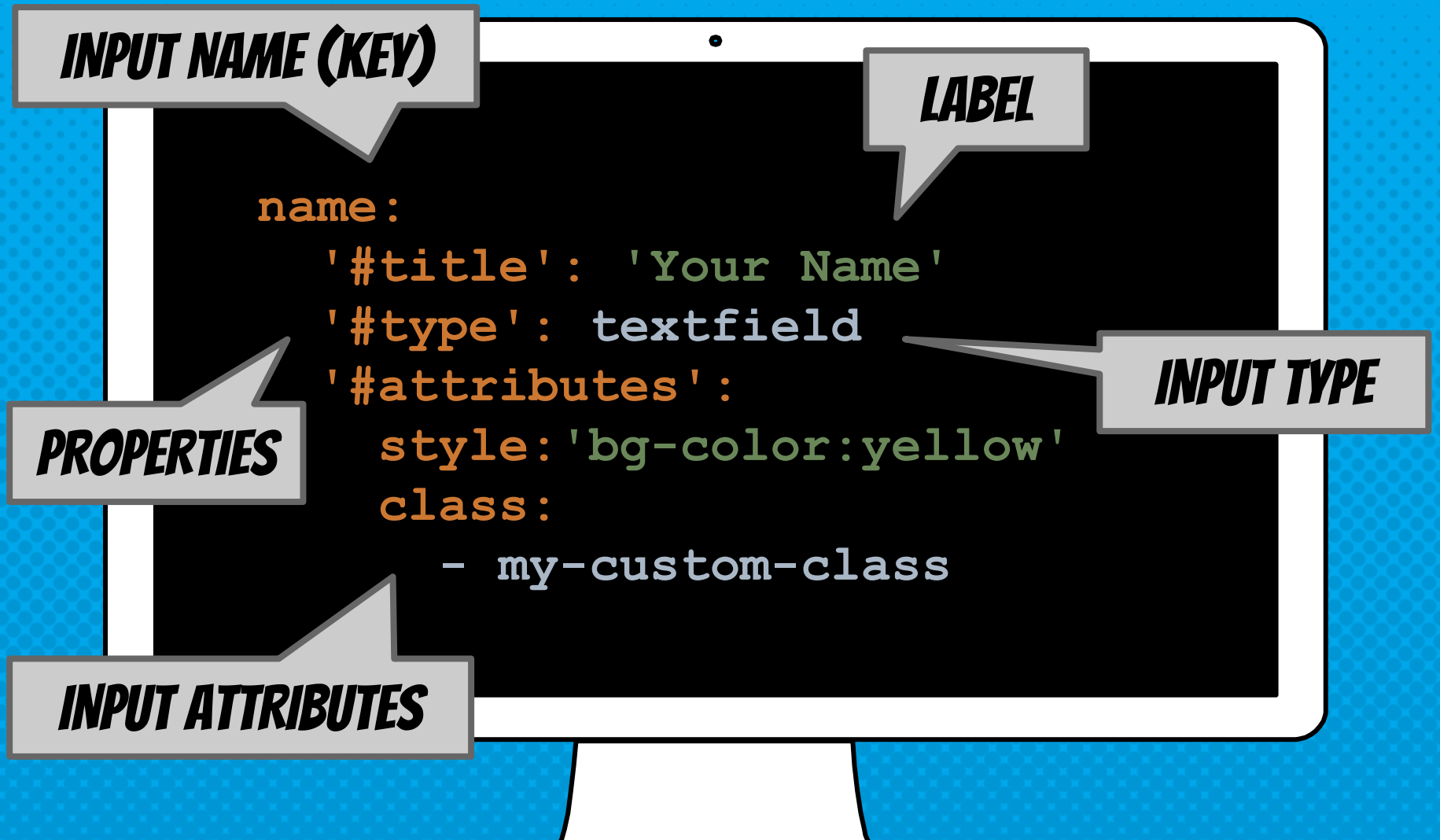
PLUGINS ARE...

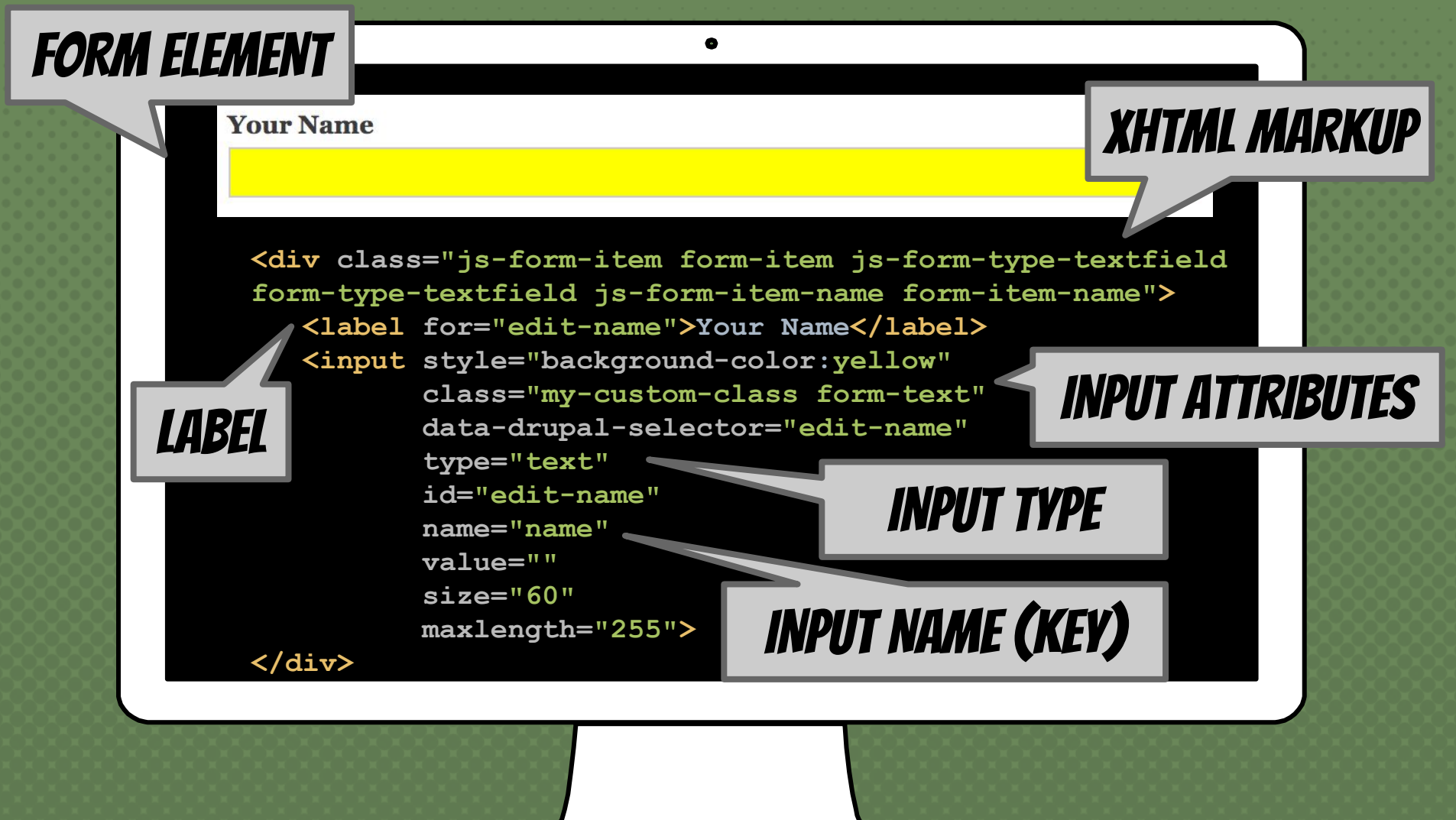


Form elements overview

- 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

<demo>
Overview: /devel/elements
Interface: …/lib/Drupal/Core/Render/Element
Example: …/Render/Element/Textfield

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 help group related behaviors

Form Element 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>
Overview: …/reports/webform-plugins/elements
Test: …/elements/textfield
Plugins: …/src/Plugin/WebformElement/
Interface: WebformElementInterface
Modules: webform_example_element.module
webform_example_composite.module

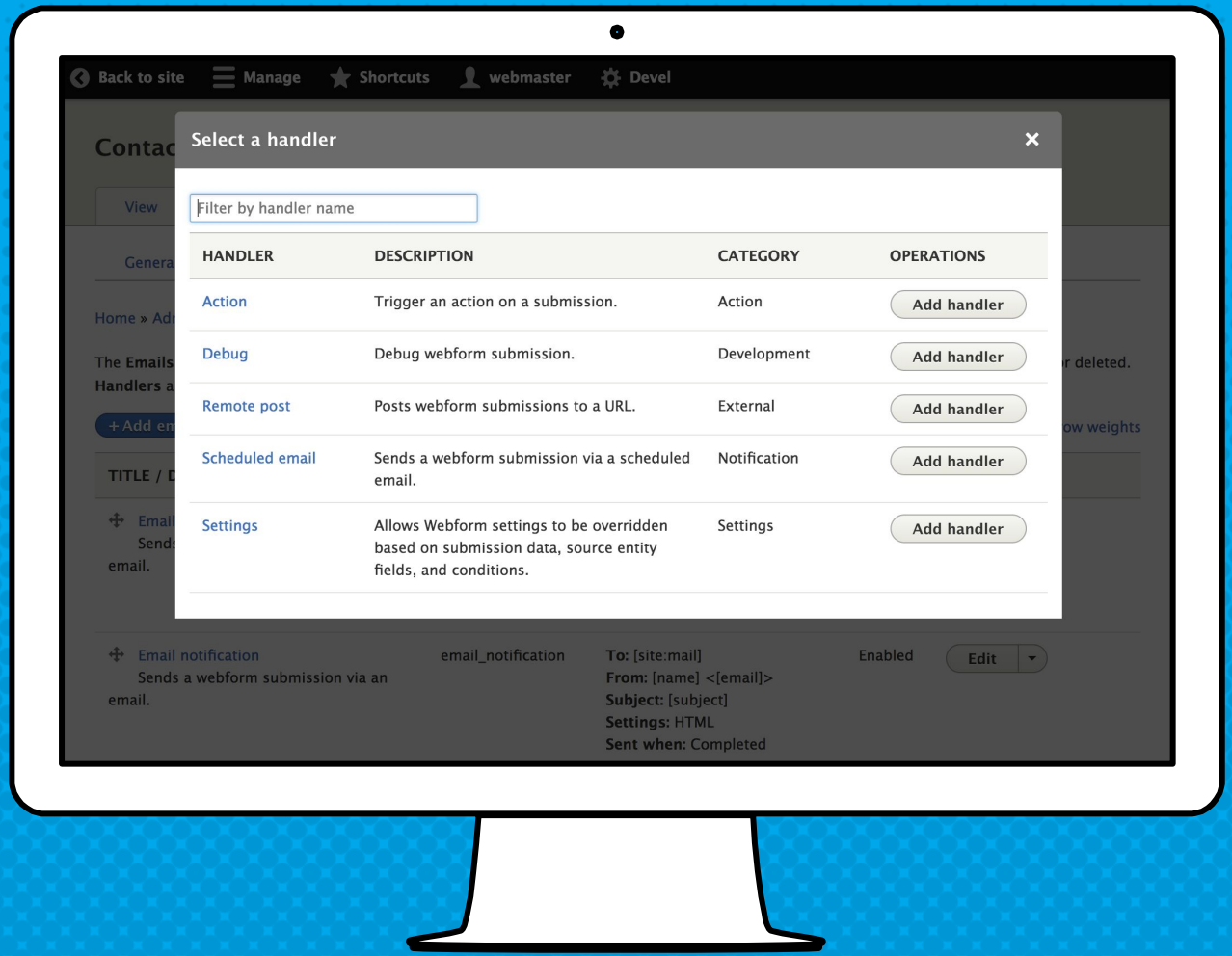
Creating Custom Webform Elements

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

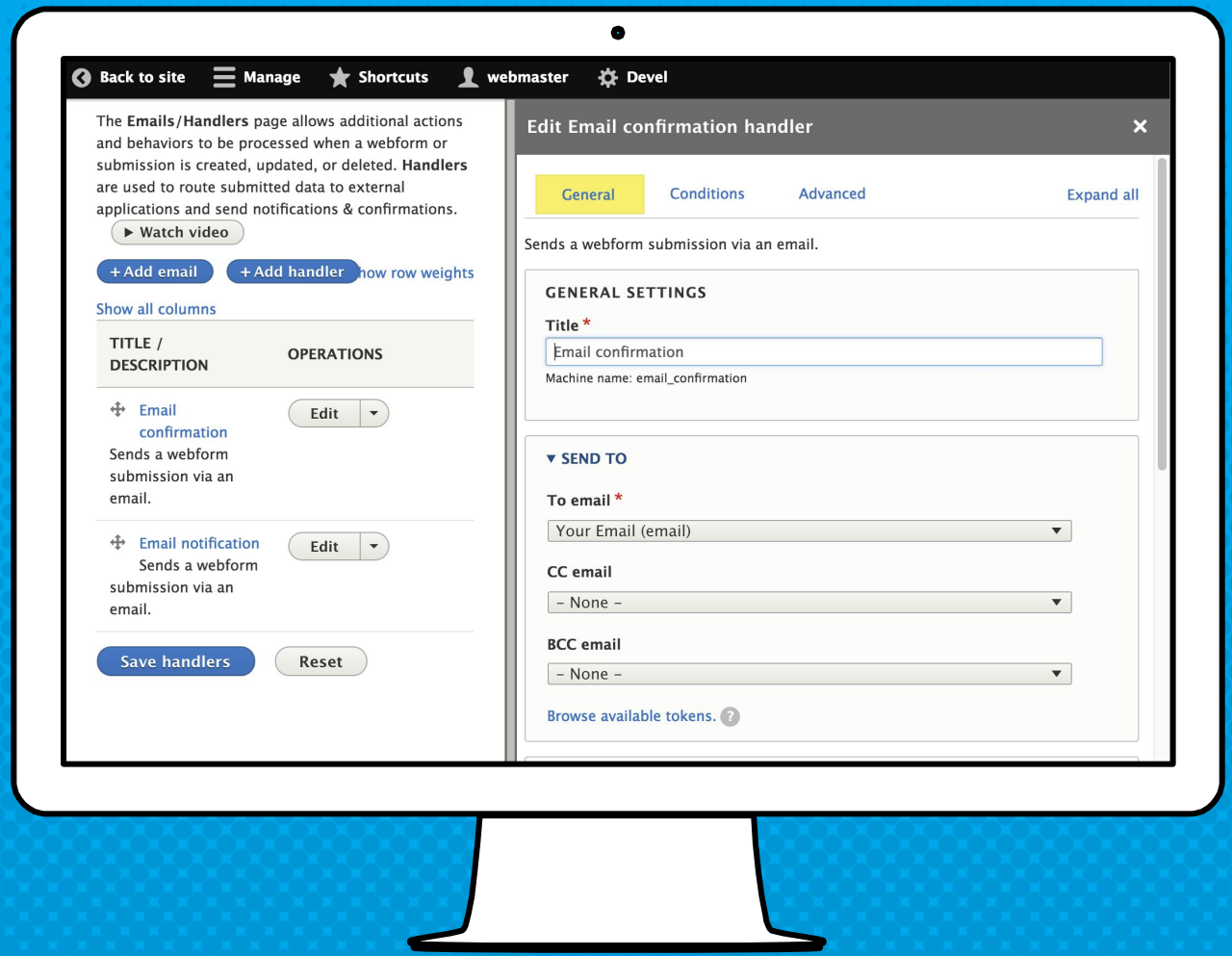
Implementing Webform Handlers

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

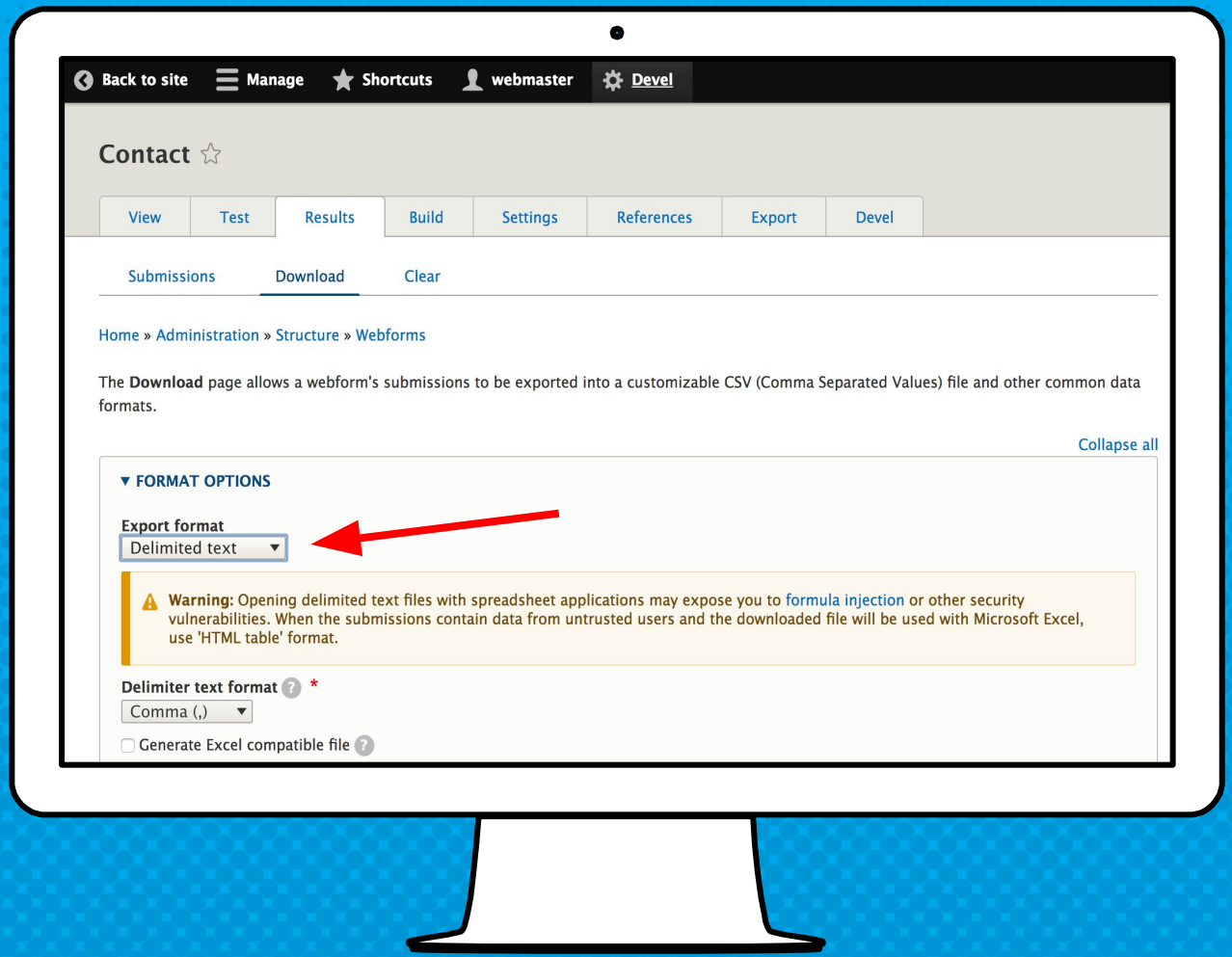
- Configuration settings — `WebformHandler::getConfiguration`
- Override settings — `WebformHandler::overrideSettings`
- Alter forms & elements — `WebformHandler::alterElement`
- Entity operations — `WebformHandler::postSave`
- Element operations — `WebformHandler::createElement`
- Handler operations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

<demo>
Overview: …/reports/webform-plugins/handlers
Test: …/contact/settings/handlers
Plugins: …/src/Plugin/WebformHandler/
Interface: WebformHandlerInterface
Module: webform_example_handler.module

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>
Overview: …/reports/webform-plugins/exports
Test: …/contact/results/download
Plugins: …/src/Plugin/WebformExporter/
Interface: WebformExporterInterface

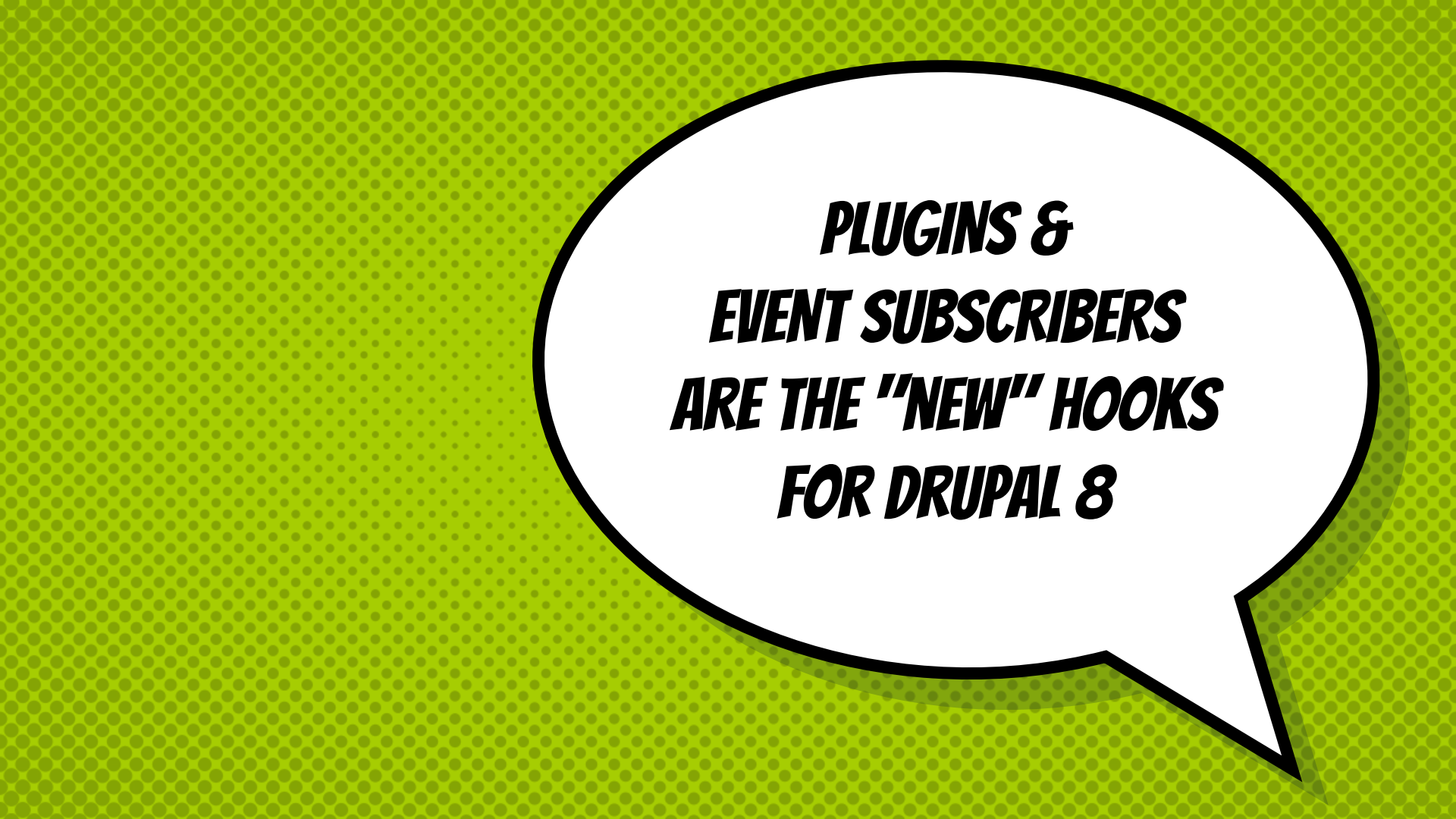
Implementing Webform Hooks

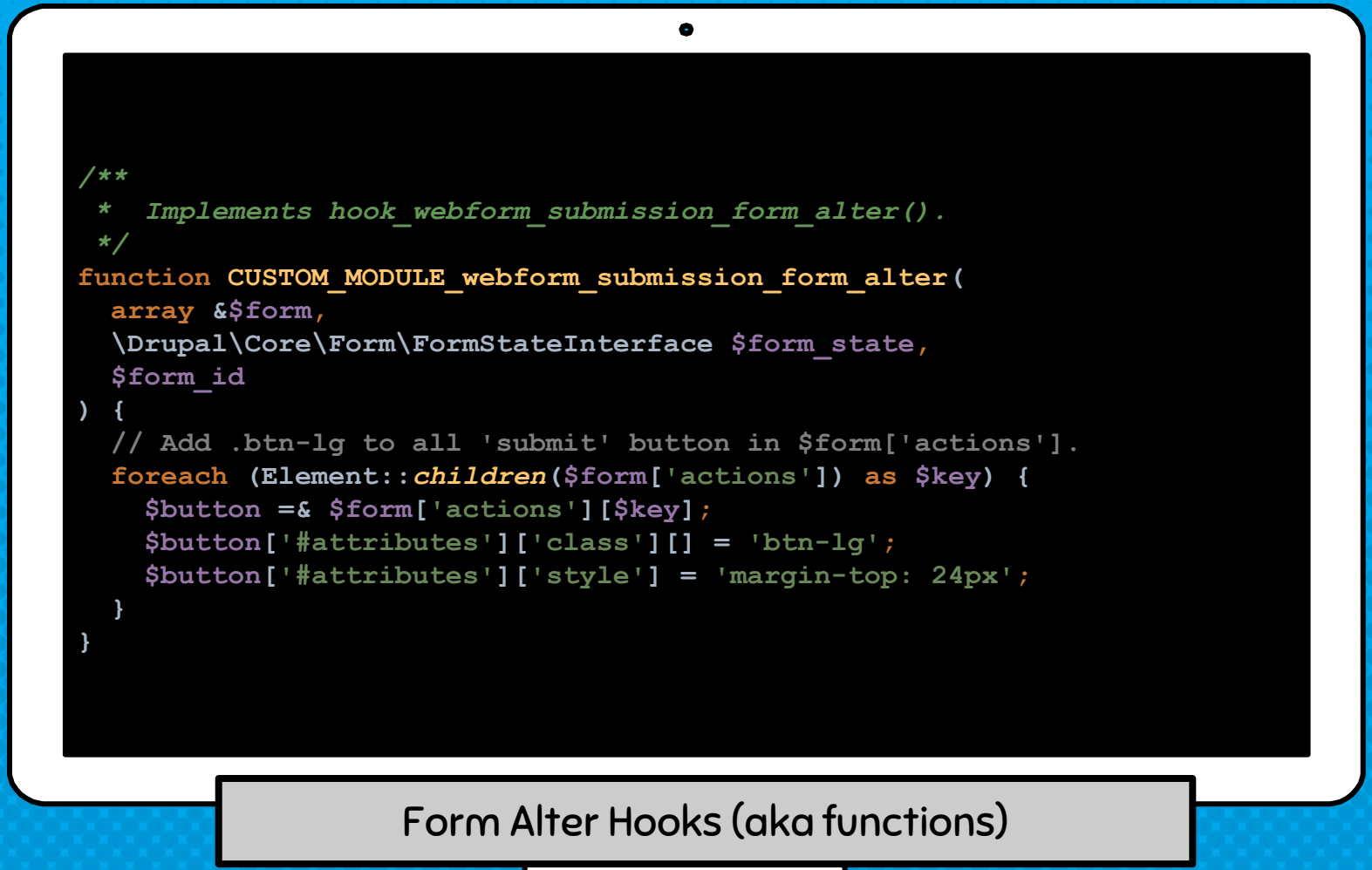
HOOKS ARE

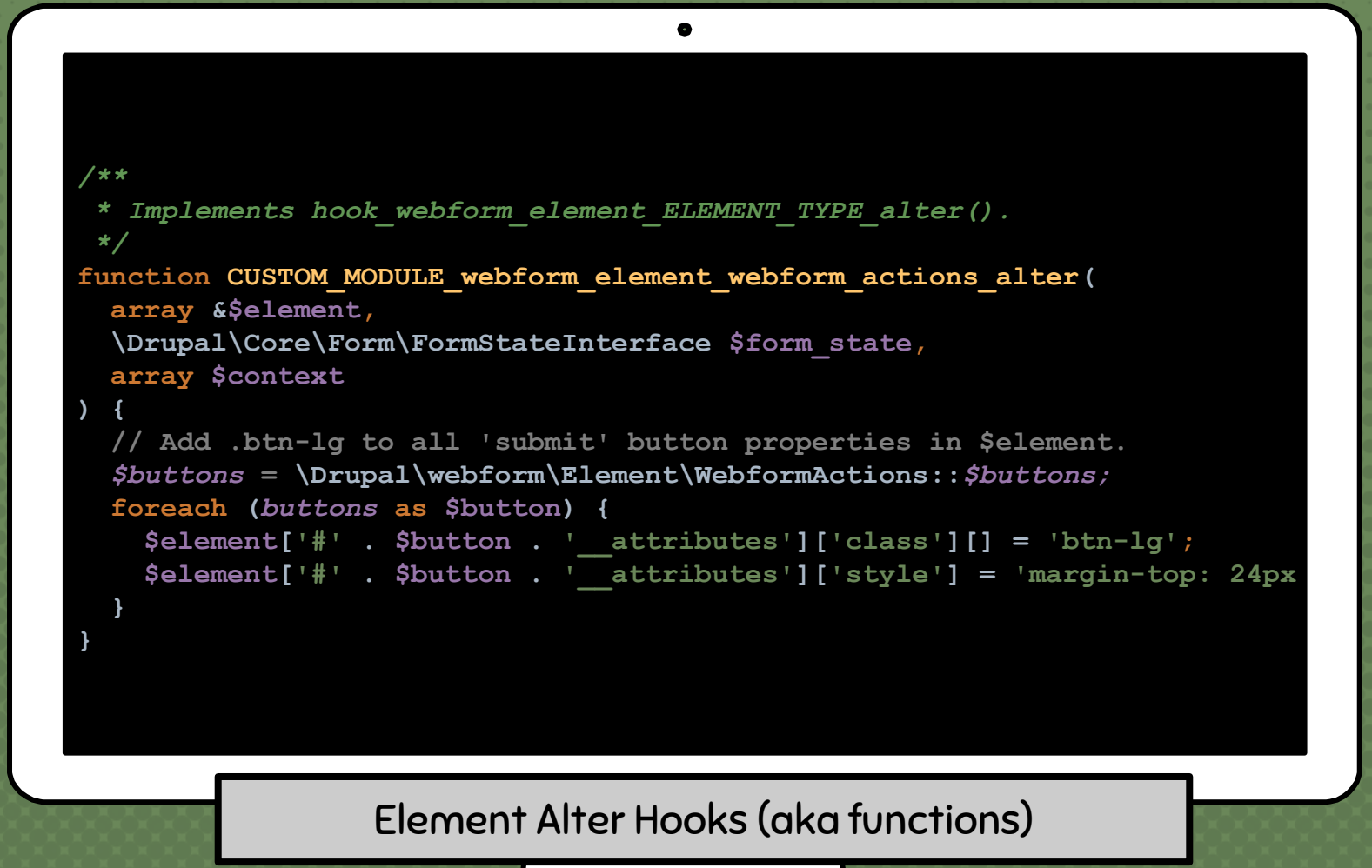
FUNCTIONS THAT DEFINE OR ALTER BEHAVIORS

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


/**
* Implements hook_webform_submission_form_alter().
*/
function CUSTOM_MODULE_webform_submission_form_alter(
array &$form,
\Drupal\Core\Form\FormStateInterface $form_state,
$form_id
) {
// Add .btn-lg to all 'submit' button in $form['actions'].
foreach (Element::children($form['actions']) as $key) {
$button =& $form['actions'][$key];
$button['#attributes']['class'][] = 'btn-lg';
$button['#attributes']['style'] = 'margin-top: 24px';
}
}
Form Alter Hooks (aka functions)


/**
* Implements hook_webform_element_ELEMENT_TYPE_alter().
*/
function CUSTOM_MODULE_webform_element_webform_actions_alter(
array &$element,
\Drupal\Core\Form\FormStateInterface $form_state,
array $context
) {
// Add .btn-lg to all 'submit' button properties in $element.
$buttons = \Drupal\webform\Element\WebformActions::$buttons;
foreach ($buttons as $button) {
$element['#' . $button . '__attributes']['class'][] = 'btn-lg';
$element['#' . $button . '__attributes']['style'] = 'margin-top: 24px
}
}
Element Alter Hooks (aka functions)

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

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

RALPH SAYS...

THANK YOU!!!

ANY QUESTIONS?