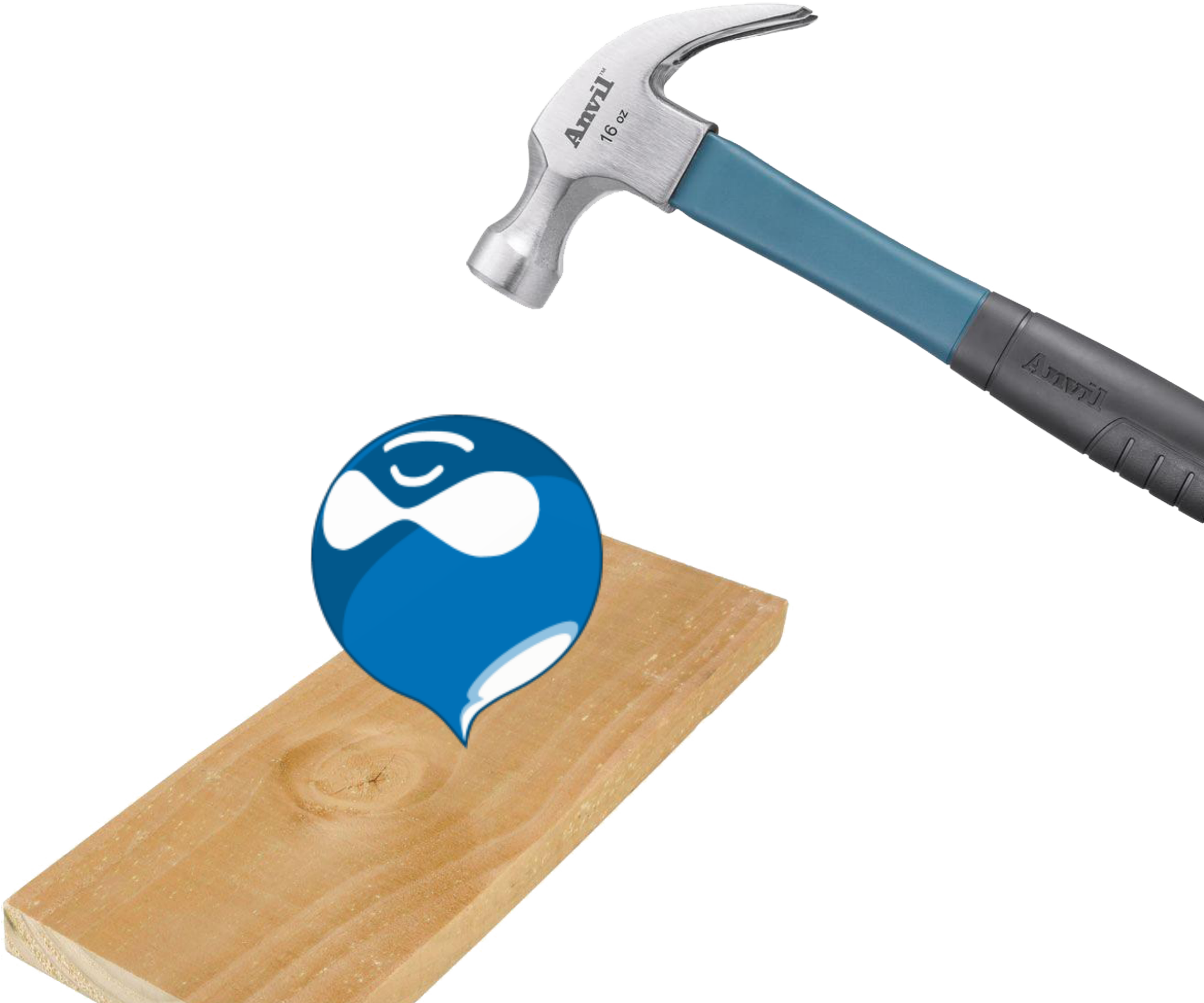
I Used Drupal Core to Build a Mind Reading Machine

Christian Knoebel
Princeton University

drupal.org: cknoebel
cknoebel@princeton.edu

Learning Objectives

- Drupal as an application platform
- Maybe there isn’t a module for that
- De-magicize some parts of D8 core API


Anvil
16 oz
Anvil

The Algorithm

- Who really wins March Madness?
- What’s the best way to boost jam sales?
- How much do you love me?
- What don’t I know?

pairLab demo:

Which issue is more important to you
in the 2020 Presidential Election?

http://bit.ly/njdrupalcamp

```
// OMG this is crappy
```

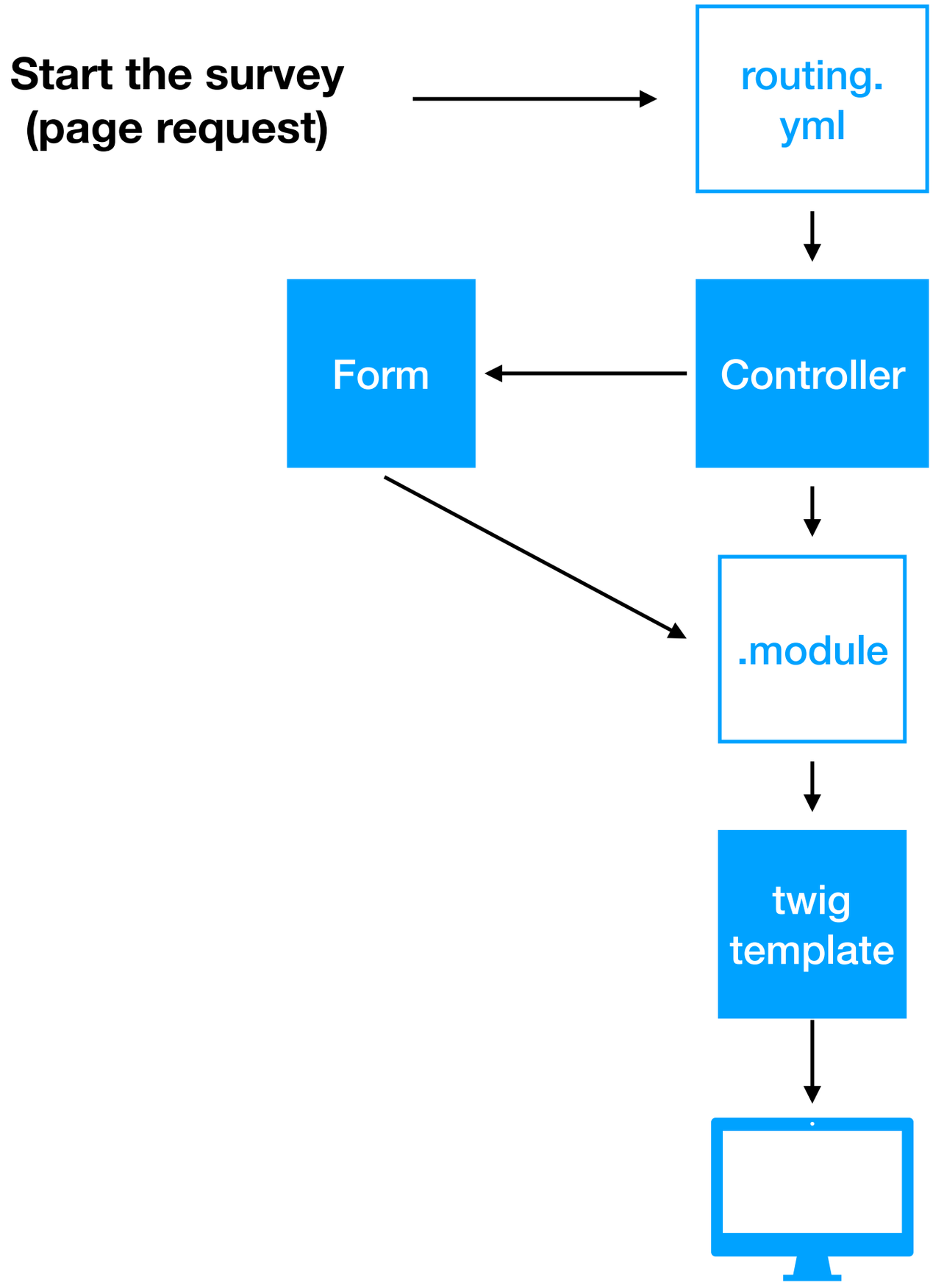
– Me

StackExchange

What is an API?

- Authentication API
- Block API
- Cache API
- CKEditor API
- Configuration API
- Database API
- Entity API
- Entity Validation API
- Filter API
- Form API
- JavaScript API
- Layout API
- Logging API
- Menu API
- Middleware API
- Migrate API
- Plugin API
- Quick Edit API
- RESTful Web Services API
- Render API
- Responses in Drupal 8
- Routing system
- Runtime Assertions
- Serialization API
- Services
- State API
- Text Editor API
- Tour API
- Translation API
- Typed Data API
- Update API

- **Entity API**
- **Render API**
- **Form API**
- **Services**


Start the survey
(page request)
routing.
yml
Form
Controller
.module
twig
template

Controller Code Pattern

- Get what we need to build the page → *Entity API & Services*
- Prep for output → *Render API*
- Send it off

Survey Start page delivers content

Let's Start!

Which issue is more important to you in the 2020 Presidential Election?

Hello and thank you for participating in this survey about finding the most important political issues in the upcoming Presidential election.

Begin the Survey

Survey node: field_survey_question

Survey node: field_survey_hello_message

URL from the route name and query string parameters

Template: survey-start.html.twig

```php
use Drupal\Core\Controller\ControllerBase;
use Drupal\Core\Url;

class SurveyStart extends ControllerBase {
  public function content() {
```

```php
use Drupal\Core\Controller\ControllerBase;
use Drupal\Core\Url;

class SurveyStart extends ControllerBase {
  public function content() {

  // Get the query string values
  $request = \Drupal::request()->query->get('suid');
  $suid = (isset($request)) ? $request : '';
  $request = \Drupal::request()->query->get('embed');
  $embed = (isset($request) && $request == 'true') ? 'true' : 'false';

  // Create a unique participant token
  $participant_token = \Drupal::service('uuid')->generate();
```

```php
// Get the Survey's node ID
$node_storage = $this->entityTypeManager()->getStorage('node');
$node_storage_ids = $node_storage->getQuery()
  ->condition('status', 1)
  ->condition('type', 'survey')
  ->condition('uuid', $suid)
  ->execute();

$survey_nid = reset($entity_ids);

// Load the Survey node
$node = $node_storage->load($survey_nid);
```

```php
// Get the survey metadata we need to pass to the page
$survey_question = $node->get('field_survey_question')->value;
$node_hello_message = $node->get('field_survey_hello_message')->value;
```

```php
// Get the Survey's node ID
$node_storage = $this->entityTypeManager()->getStorage('node');
$node_storage_ids = $node_storage->getQuery()
  ->condition('status', 1)
  ->condition('type', 'survey')
  ->condition('uuid', $suid)
  ->execute();

$survey_nid = reset($entity_ids);

// Load the Survey node
$node = $node_storage->load($survey_nid);
```

```php
// Get the survey metadata we need to pass to the page
$survey_question = $node->get('field_survey_question')->value;
$node_hello_message = $node->get('field_survey_hello_message')->value;

// Build the URL for the button
$url = Url::fromRoute('pairlab.segmentqs', [
  'suid' => $suid,
  'ptoken' => $participant_token,
]);
```

```php
// Build render array
$build['#theme'] = 'survey_start';

$build['survey_question'] = [
  '#type' => 'markup',
  '#markup' => $survey_question,
];

$build['hello_message'] = [
  '#type' => 'markup',
  '#markup' => $node_hello_message,
];

$build['begin_survey'] = [
  '#type' => 'link',
  '#title' => 'Begin the Survey',
  '#url' => $url,
  '#attributes' => [
    'class' => ['btn btn-primary']
  ]
];
```

```php
// Build render array
$build['#theme'] = 'survey_start';

$build['survey_question'] = [
  '#type' => 'markup',
  '#markup' => $survey_question,
];

$build['hello_message'] = [
  '#type' => 'markup',
  '#markup' => $node_hello_message,
];

$build['begin_survey'] = [
  '#type' => 'link',
  '#title' => 'Begin the Survey',
  '#url' => $url,
  '#attributes' => [
    'class' => ['btn btn-primary']
  ]
];

return $build;
```

```
<div id="survey-start-wrapper">
  <h2>{{ element.survey_question }}</h2>
  <div>{{ element.hello_message }}</div>
  <div class="flex-container">
    <!-- some other html tags -->
  </div>
  <div class="start-button">{{ element.begin_survey }}</div>
</div>
```

Challenge Page collects data

Survey node:
field_survey_question

Which issue is more important to you in the 2020 Presidential Election?

Count ideas to calculate progress bar fill-in

Gun rights/Second Amendment

Foreign affairs

Choose two ideas randomly & score the winner

I Don't Know / They're the Same

Suggest an idea instead ...

Form elements for submitting data

I'm Done Voting

URL from the route name and query string parameters

Capture submit values in hidden fields

Render as a form

Submit/process data

```
// Get the query string values
$suid =    // Same approach as before
$ptoken = //
$embed =  //

// Load the survey node
$node_storage = // Same approach as before using Entity API

// Ready the question
$question = // Grab field_survey_question value via Entity API

// Get the challening ideas pair
$challengers =  // Use Enity API to pick two random Idea nodes.
                // Return node IDs.
```

```php
// Build the progress bar
// Initialize the database connection
$connection = \Drupal::database();

// Get the participants for this segment
$participants = $connection->select('participant', 'p')
  ->fields('p', ['user_points'])
  ->condition('p.participant_token', $participant_token, '=')
  ->execute();

foreach ($participants as $participant) {
  $participant_response_count = $participant->user_points;
}

// Get the possible number of pairs
$node_storage = $this->entityTypeManager()->getStorage('node');
$entity_ids = $node_storage->getQuery()
  ->condition('type', 'idea')
  ->condition('field_idea_survey_nid', $survey_nid)
  ->condition('field_idea_status', 1)
  ->accessCheck(FALSE);
$entity_ids = $entity_ids->execute();
$idea_count = count($entity_ids);
```

```php
$number_combinations = // PHP math to calculate the number of combination

//Calculate progress
$participant_progress =
  round(($participant_response_count / $number_combinations) * 100);
```

```php
if ($participant_progress < 101) {
  // Get the form
  $form = $this->formBuilder()->getForm(
    '\Drupal\pairlab\Form\SurveyChallengeForm',
    $challengers,
    $question,
    $survey_nid,
    $suid,
    $ptoken,
    $embed,
    $participant_progress
  );

  $form['#theme'] = 'survey_challenge';

  return $form;
} else {
  return $this->redirect('pairlab.end_survey', array(), ['query' => [
      'suid' => $suid,
      'ptoken' => $ptoken,
      'embed' => $embed,
    ]]);
}
```

```php
public function buildForm(array $form, FormStateInterface $form_state,
  $challengers = NULL, $question = NULL, $survey_nid = NULL, $suid = NULL,
  $ptoken = NULL, $embed = NULL, $participant_progress = NULL) {

  // Pass key/values to other callbacks
  $form_state->setFormState(array(
    'survey_nid' => $survey_nid,
    'suid' => $suid,
    'participant_token' => $ptoken,
    'embed_flag' => $embed,
  ));

  // Build the "I'm Done Voting" URL
$url = Url::fromRoute( 'pairlab.end_survey'    , [
  'suid' => $suid,
  'ptoken' => $ptoken,
  'embed' => $embed
]);
```

```php
// Prepare the challengers for use in the idea pair fields
$idea = array();

foreach($challengers as $nid => $label) {
  $idea[$nid] = $label;
}

$idea_first = reset($idea);
$idea_first_key = key($idea);
$idea_last = end($idea);
$idea_last_key = key($idea);
```

```php
// Build the form
$form['question'] = [
  '#type' => 'item',
  '#markup' => $this->t($question),
  '#prefix' => '<h2>',
  '#suffix' => '</h2>',
];

// Why an inline template? I want to provide a style attribue, but Drupal
// strips it out of the attributes
// It won't strip style out of a twig template, even if it's inline
$form['progress_bar'] = [
  '#type' => 'inline_template',
  '#template' => '<div class="progress"><div class="progress-bar"
    role="progressbar" aria-valuenow="{{ progress_value }}" aria-valuemin="0"
    aria-valuemax="100" style="width: {{ progress_value }}%;">
    <span class="sr-only">{{ progress_value }}%</span></div></div>',
  '#context' => [
    'progress_value' => $participant_progress
  ],
];
```

```php
$form['idea-pair-1'] = [
  '#type' => 'item',
  '#markup' => $this->t($idea_first),
  '#prefix' => '
    <div id="pair-score-wrapper" data-toggle="tooltip" data-placement="top"
      title="Choose the best of the two" value="1">
      <div id="pair-wrapper" class="btn-group pair-wrapper">
        <div id="idea-pair-1" class="btn btn-primary pair left" type="button"
        value="' . $idea_first_key . '">',
  '#suffix' => '
    </div>',
];
```

```php
$form['idea-pair-2'] = [
  '#type' => 'item',
  '#markup' => $this->t($idea_last),
  '#prefix' => '
      <div id="idea-pair-2" class="btn btn-primary pair right" type="button"
        value="' . $idea_last_key . '">',
  '#suffix' => '
      </div>
    </div>',
];
```

```
// Form is submitted by javascript when any of these are clicked
$form['idea-score-1'] = [
  '#type' => 'item',
  '#markup' => $this->t('A little'),
  '#prefix' => '
    <div id="idea-score-wrapper" class="btn-group idea-score-wrapper">
      <div id="idea-score-1" class="btn btn-info idea-score left"
        type="button" value="3">',
  '#suffix' => '
      </div>',
];

$form['idea-score-2'] = [
  '#type' => 'item',
  '#markup' => $this->t('Moderately'),
  '#prefix' => '
      <div id="idea-score-2" class="btn btn-info idea-score center"
        type="button" value="5">',
  '#suffix' => '
      </div>',
];
```

```
$form['idea-score-3'] = [
  '#type' => 'item',
  '#markup' => $this->t('Very'),
  '#prefix' => '
      <div id="idea-score-3" class="btn btn-info idea-score right"
        type="button" value="7">',
  '#suffix' => '
      </div>
    </div>',
];
```

```php
$form['skip'] = [
  '#type' => 'item',
  '#markup' => $this->t('I Don\'t Know / They\'re the Same'),
  '#prefix' => '
      <div id="skip-wrapper">
        <div id="skip" class="btn btn-default fade-in" type="button"
          data-toggle="tooltip" data-placement="left"
          title="Skip if you\'re not sure" value="1">',
  '#suffix' => '
        </div>
      </div>
    </div>',
];
```

```php
$form['suggestion'] = [
  '#type' => 'textfield',
  '#title' => '',
  '#description' => $this->t(''),
  '#id' => 'suggestion',
  '#placeholder' => $this->t('Suggest an idea instead ...'),
  '#attributes' => array(
    'data-toggle' => array('tooltip'),
    'data-placement' => array('bottom'),
    'title' => array('Add your great idea here'),
    'class' => array('fade-in'),
  )
];
```

```php
$form['submit'] = [
  '#type' => 'submit',
  '#value' => $this->t('Submit Your Suggestion'),
];

$form['done'] = [
  '#type' => 'link',
  '#title' => $this->t('I\'m Done Voting'),
  '#url' => $url,
  '#attributes' => [
    'class' => ['btn btn-default fade-in'],
    'role' => 'button',
    'data-toggle' => 'tooltip',
    'data-placement' => 'left',
    'title' => 'End the survey anytime'
  ]
];
```

```php
// Hidden fields, used to store values for submit.
// Why? The random idea generator regenerated two new ideas on form rebuild.
// It rebuilds the form before we can grab the chosen values and screws up
// the results. To compensate, we're using jQuery, which operates
// indipendantly of form rebuild to grab and store the values.

// Stores idea 1.
$form['idea-1'] = [
  '#type' => 'textfield',
];

// Stores idea 2.
$form['idea-2'] = [
  '#type' => 'textfield',
];

// Stores the choice winner.
$form['winner'] = [
  '#type' => 'textfield',
];

// Stores the idea score.
$form['idea-score'] = [
  '#type' => 'textfield',
];
```

```
// Hidden fields, used to store values for submit.
// Why? The random idea generator regenerated two new ideas on form rebuild.
// It rebuilds the form before we can grab the chosen values and screws up
// the results. To compensate, we're using jQuery, which operates
// indipendantly of form rebuild to grab and store the values.

// S
$for
  '#
];

// S
$for
  '#
];

// Stores the choice winner.
$form['winner'] = [
  '#type' => 'textfield',
];

// Stores the idea score.
$form['idea-score'] = [
  '#type' => 'textfield',
];
```

```
// Idea Score buttons
// If an idea score button is clicked:
// - set the hidden idea score field with the right value
// - submit the form
$('.kd-crawford-survey-challenge-form .idea-score').click(function() {
  var buttonValue = $(this).attr( "value" );
  $('#edit-idea-score').val(buttonValue);
  $('.kd-crawford-survey-challenge-form #idea-score-wrapper').
    addClass('slide-out-to-left');
  $(this).parents('form').submit();
});
```

```php
// Honeypot.
$form['pooh-bear'] = [
  '#type' => 'checkbox',
  '#title' => 'Contact me by carrier pigeon only',
  '#default_value' => 0,
];

// Kill the cache for this form or else it will hang on to the first
// pair it picks
$form['#cache'] = ['max-age' => 0];

return $form;
```

```php
public function submitForm(array &$form, FormStateInterface $form_state) {

    // Set variables based on $form_state
    $idea_1 = $form_state->getValue('idea-1');
    $idea_2 = $form_state->getValue('idea-2');
    $winner = $form_state->getValue('winner');
    $idea_score = $form_state->getValue('idea-score');
    $suggestion = $form_state->getValue('suggestion');
    $survey_nid = $form_state->get('survey_nid');
    $ptoken = $form_state->get('participant_token');
    $suid = $form_state->get('suid');
    $embed = $form_state->get('embed_flag');
```

```php
// Write results to the database
$connection = \Drupal::database();

if (empty($suggestion) && $winner) {
  // Save the vote response
  $vote = $connection->insert('responses')
  ->fields([
    'sid' => $survey_nid,
    'idea_1' => $idea_1,
    'idea_2' => $idea_2,
    'winner' => $winner,
    'idea_score' => $idea_score,
    'participant_token' => $ptoken,
    'created' => REQUEST_TIME,
  ])
  ->execute();
```

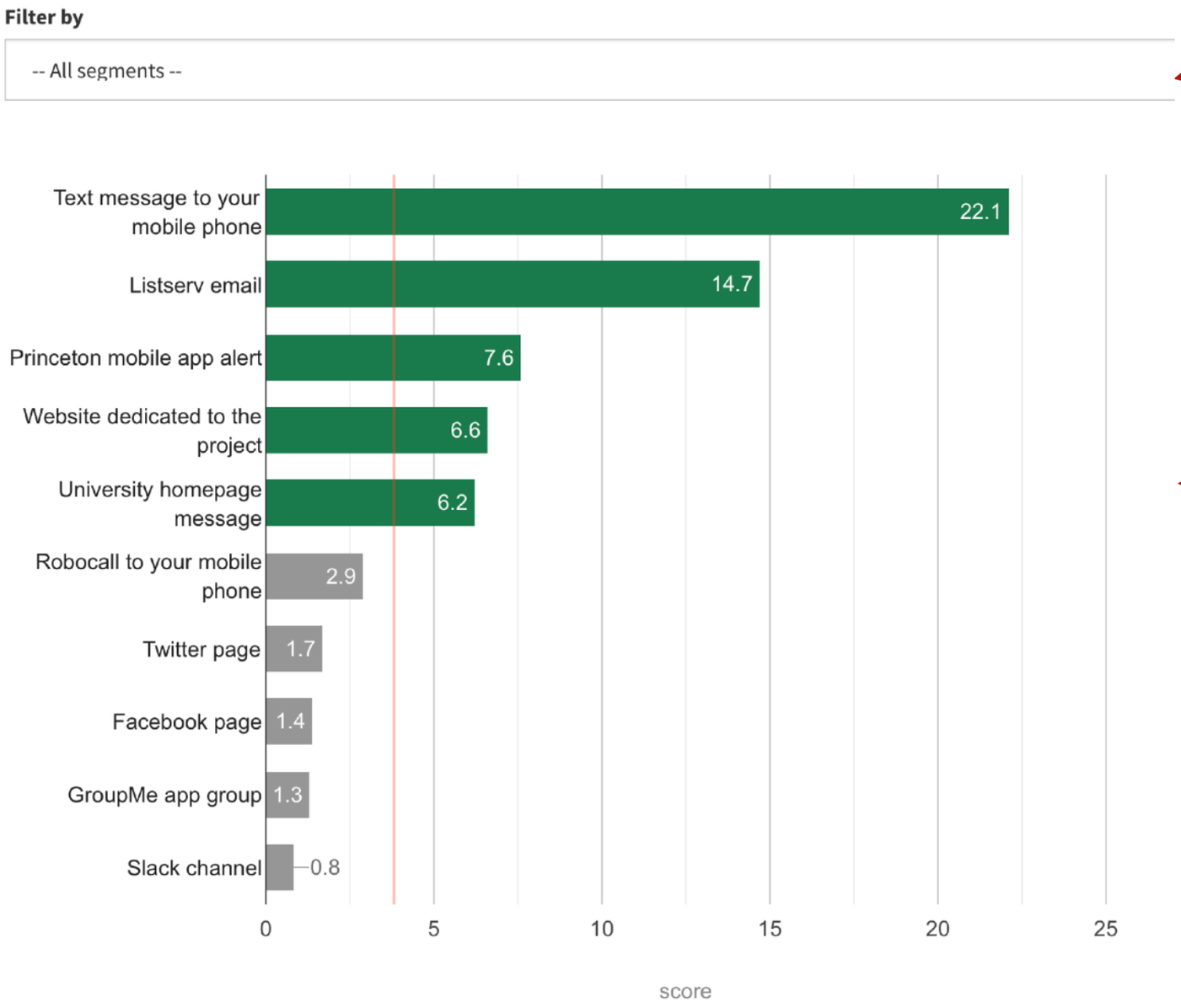
```php
$node = Node::create([
  'type' => 'idea',
  'title' => $title . ' (' . $survey_nid . ')',
  'field_idea_suggestion' => 1,
  'field_idea_idea' => $suggestion,
  'field_idea_status' => 2,
  'field_idea_survey_nid' => $survey_nid,
  'uid' => $survey_uid,
  'status' => 1,
]);
$node->enforceIsNew();
$node->save();

// Thank the user
\Drupal::messenger()->addStatus(t('
  <div class="suggestion-message">
  Thanks! We\'ll review your idea and it will appear soon.
  Keep the great ideas coming.</div>
'));
```

```
$form_state->setRedirect('pairlab.challenge', array(), ['query' => [
  'suid' => $suid,
  'ptoken' => $ptoken,
  'embed' => $embed,
]]);

return;
```

Results Ranking page shows results



Survey node:
field_survey_segment_x

Results array & mean score packaged into something Google Charts can read

Template: survey_results_ranking.html.twig

```php
$results = // Function that returns an array of Idea nids and their score

// Get the mean of the results scores
$results_mean = // Function that returns the mean of the results scores

// Build the rows for Google Charts
$color_above_average = '#147D4C';
$color_below_average = '#999';
$chart_rows = '';

foreach ($results as $idea_nid => $score) {
  $idea_title = $this->entityTypeManager()->getStorage('node')->load($idea_nid)
    ->get('field_idea_idea')->value;
  $color = ($score >= $results_mean) ? $color_above_average :
    $color_below_average;

  $chart_rows = $chart_rows . '["' . $idea_title . '",' . round($score,1) .
    ',"' . $color . '","' . round($score,1) . '"],';

  /* Example:

  ['My First Idea', 20, '#147D4C', '20'],
  ['My Second Idea', 16, '#147D4C', '16'],
  ['My Third Idea', 10, '#999', '10'],
  ['My Fourth Idea', 8, '#999', '8'],

  */
}
```

```php
// Build render array
$build['#theme'] = 'survey_results_ranking';

$build['segment_filter'] = [
  '#type' => 'select',
  '#title' => 'Filter by',
  '#options' => $segments,
  '#id' => 'segment-filter',
];

$build['chart_mean'] = [
  '#type' => 'markup',
  '#markup' => $results_mean,
];

$build['chart_rows'] = [
  '#type' => 'markup',
  '#markup' => $chart_rows,
];

return $build;
```

```
<script>

google.charts.load('current', {packages: ['corechart', 'bar']});
google.charts.setOnLoadCallback(draw_chart);

function draw_chart() {

  // Initialize data
  var data = new google.visualization.DataTable();
  var chartwidth = document.getElementById('ranking-chart').offsetWidth;

  // Create columns
  data.addColumn('string', 'Idea');
  data.addColumn('number', 'Pair score');
  data.addColumn({type: 'string', role: 'style'});
  data.addColumn({type: 'string', role: 'annotation'});

  // Add data
  data.addRows([{{ element.chart_rows|raw }}]);

  var options = // set options for display
```

```
  var chart = new google.visualization.BarChart(
    document.getElementById('ranking-chart')
  );

  chart.draw(data, options);

  // Function that draws the mean line
  drawAxisLine(chart, {{ element.chart_mean|raw }}, 'vertical', '#ff0000');

}

</script>
```

```
<div id="survey-results-wrapper">
  <div id="ranking-segment-filter">{{ element.segment_filter }}</div>
  <div id="ranking-chart"></div>
</div>
```

Results

pairlab.io