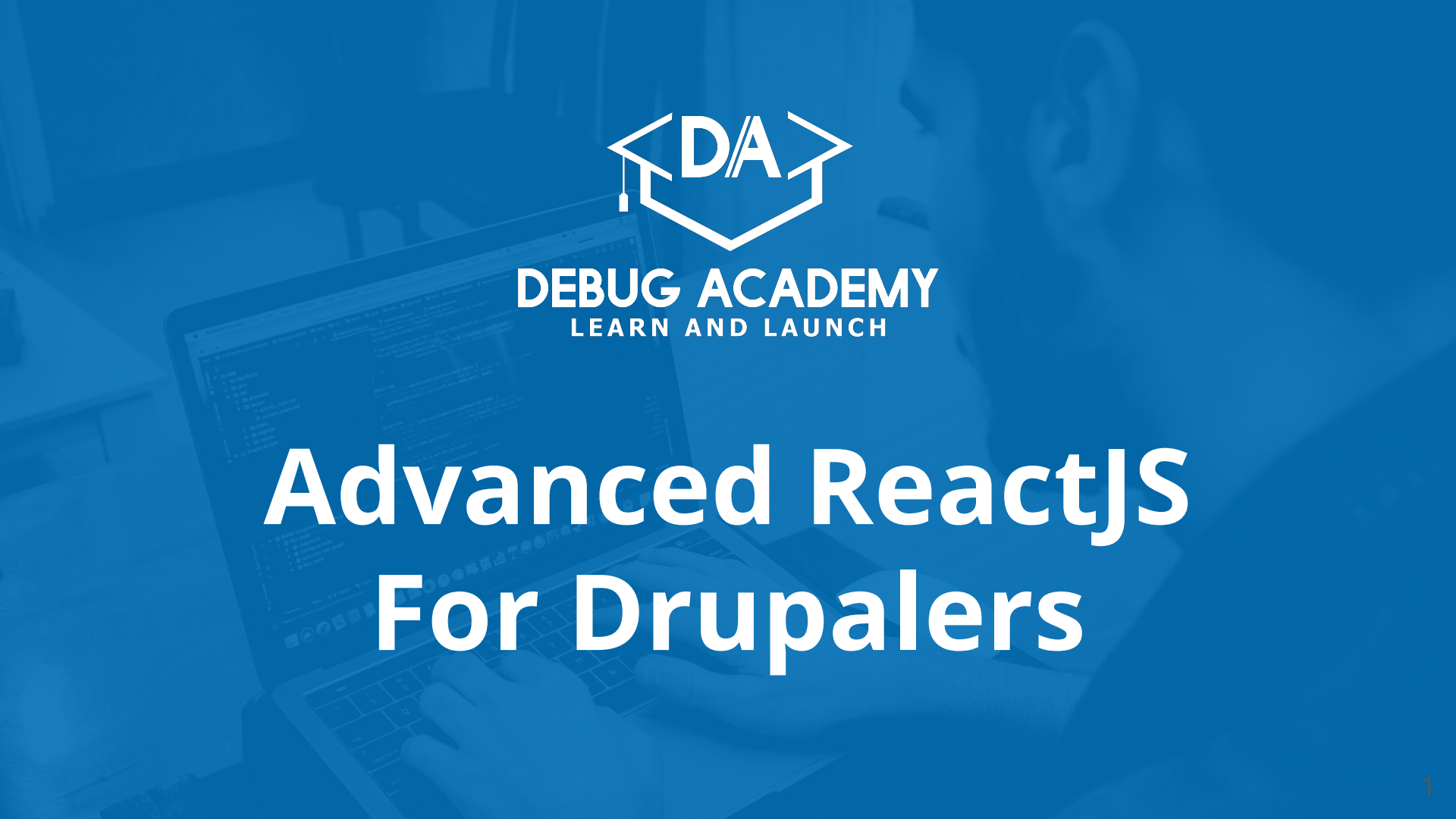


Advanced ReactJS For Drupalers

Class or Functional components?

Class components

- Contain lifecycle methods
  - Similar to Drupal hooks
- Natively support state
- Include `this`
- Greater set of functionality
- Encourages explicit code

Functional

- AKA “Stateless” components
  - Now “functional” components, thanks to hooks
- Allows for very concise syntax
- Able to avoid `this`
- Can do *almost* everything class components can do
- It’s what the cool kids are doing

Fully decoupled React Apps

- **React-router-dom**
  - Declarative, relatively straightforward routing
  - Used by the JS Initiative's **drupal-admin-ui**
- Next.js
  - Focus on server side rendering
  - More opinionated than react-router-dom
- Gatsby
  - Allows "building" website and deploying only static files
  - Saves API data in static files

react-router-dom Fundamentals

- `<BrowserRouter>`
  - Wraps the app which has routes
- `<Link>`
  - Links to a path
- `<Route>`
  - Wraps route-specific content

```
<BrowserRouter>
   <Link
     to={`/node/${node.id}`}
     >{node.title}</Link>
   <Route
     path="/"
     component={LoginForm}
     exact={true}>
   <Route
     path="/node/:nid"
     render={({match}) => (
      <DNode nid={match.params.nid}>
     )}>
</BrowserRouter>
```

Styling options

CSS resides in CSS file(s)

- Use CSS file(s) in component(s)
  - `Css-loader` package with webpack enables this
  - `Import ‘./css/style.css;`
- Then add normal CSS class names to components
  - className = “my-css-class”

Inline styles

- Create JS object with CSS properties, like: `myStyle={marginTop: `5px`}`
- Pass the style object to a component’s style prop: `<Title style={myStyle} />`
- Worse performance than CSS classes

cont…

Styling options (cont'd)

Styled-components

- Write standard CSS using template literals:

```
const Title = styled.h1` font-size: 1.6em;
                         color: blue; `;
```

- Allows dynamic CSS rules using JS
- Unused CSS is not included in rendered app
- CSS is written in specific components
- No more conflicting CSS - same benefits of inline CSS

React Portals

Allows rendering React components anywhere on the DOM!

- Pass the component and the regular DOM element:

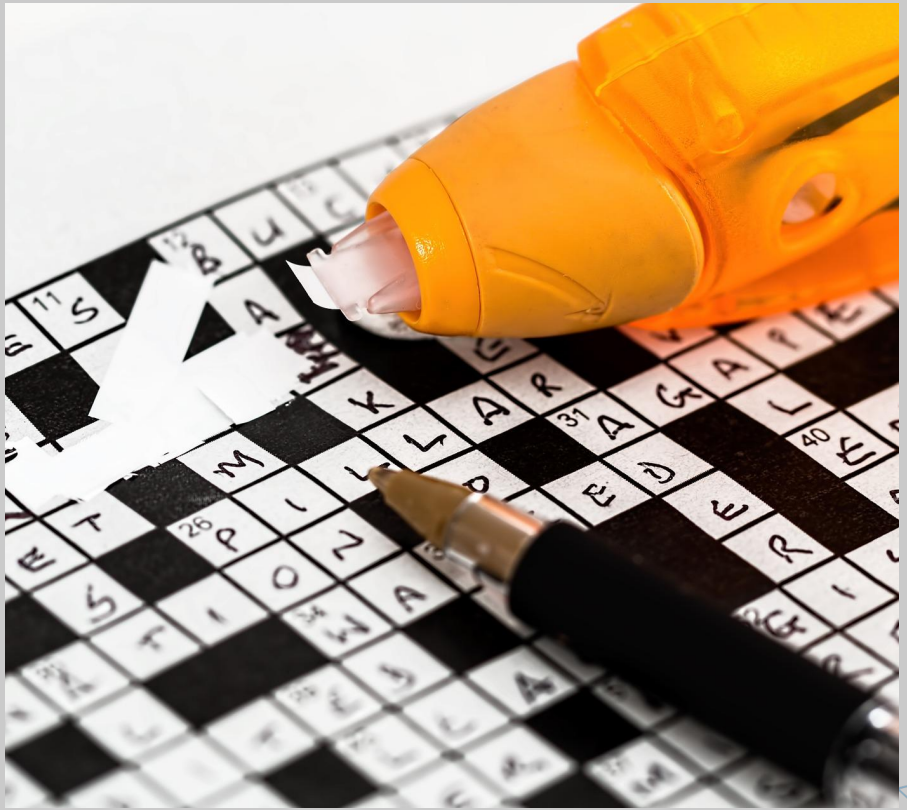
```
ReactDOM.createPortal(Component, domElement);
```

- Context, event bubbling, error handling, etc - continues to function!
- Standard Component, except it can appear outside of the ‘app’

Error Handling

1. Rendering Errors
    - When react does not know what to Render
    - Create Error Boundary Component
    - Wrap component in Error Boundary

2. Other JS errors
    - E.g. Error in event handler
    - Simply use try {} catch {}



Creating Error Boundaries

- An Error Boundary is a special type of custom component
- It must be created as a class component
- To create an error boundary, implement one or both of these lifecycle methods:

- `static getDerivedStateFromError() { }`
  - Return new state values, like: `return { caughtError: true };`
- `componentDidCatch()`
  - For logging information after an error
  - Can also update state here

Implicit vs Explicit state

State is passed to child components Implicitly or Explicitly

- **Implicit state**
  - Developer may not care to know about the passed state
  - Ex: React Context, `React.cloneElement`
- **Explicit state**
  - App developer deliberately passes state
  - Immediately apparent that state is being passed
  - Ex: Higher Order Components (HOC), Render props

Implicit state w/React.cloneElement

Implicitly populates a components children and/or props.

- Generates a copy of component & passes props
- Overwrites conflicting props as needed

```
<List type="ordered">
  <ListItem text="An item" />
  <ListItem text="Another item" />
</List>
```

New Cloned element:

Behind the scenes, React.cloneElement() implicitly passes the list type (& more) to the ListItem(s):

```
render() {
  return React.cloneElement(ListItem, [props], [...children]);
}
```

Making state available downstream

Two primary approaches for implicitly passing state:

React Context API

- Comes OOTB with React
- Straightforward set up
- Shares state - no bells and whistles
- New & shiny, and covers most use cases

Redux

- Not part of React, but plays nicely with it
- Comes with Debug tool
- Can “time travel” through history when debugging
- Supports middleware
- Very robust

React Context

Easily provide data to nested components

Providing Context

- Initialize new context:

```
export const UserContext = React.createContext({ user });
```

- Override initial context for child components:

```
<UserContext.Provider value={masqueradeAsUser}>
</UserContext>
```

React Context (cont'd)

Receiving Context

- Pass context as prop to a component

```
<MyContext.Consumer>
  {activeUser => <WelcomeBlock user={activeUser}/>}
</MyContext.Consumer>
```

- Or get context within a functional component:

```
let activeUser = useContext(activeUser);
```

Redux

React's context API is good for most applications.

In advanced use cases, utilizing Redux for the sake of its toolset may be worthwhile.

16

Redux: Initialization

- Create a reducer function which contains logic for updating redux-provided state
- Create store for storing state
- Wrap components in store's provider

```
import { createStore } from "redux"
import { Provider } from 'react-redux';
const myReducer = function reducer(state = initialState, action) {
  switch(action.type) {
    case "USER_VIEW": return { totalViews: state.totalViews + 1 }}
  return state; }
const store = createStore(myReducer);
<Provider store={store}><App /></Provider>
```

Redux: Retrieving State

- Redux's HOC, `connect()()`, provides component's props
- Integrate appropriate actions to update state
- Update `myNestedComponent`'s export

```
Import { connect } from react-redux;
// myNestedComponent code..
const mapStateToProps = state => ({ totalViews: state.totalViews});
export default connect(mapStateToProps)(myNestedComponent);
const viewAction = function userViewedPage(user) {
  return {
    type: 'VIEW',
    totalViews: user.totalViews,};}
export default connect(mapStateToProps,
viewAction)(myNestedComponent);
```

HOC

- HOC's are a replacement for inheritance in react
- Explicitly pass behavior or props to a component
- Even lifecycle hooks or props can be "passed down"

```
function logProps(WrappedComponent) {
  return class extends React.Component {
    componentWillReceiveProps(nextProps) {
      console.log('Current props: ', this.props);
      console.log('Next props: ', nextProps);
    }
    render() {
    // Wraps the input component in a container,
    // without mutating it. Good!
      return <WrappedComponent {...this.props} />;
    }
  }
```

HOC’s: Keep in mind..

When using an HOC, props can come from 3 sources:

1. Provided from within the HOC
2. Provided from the component itself
3. Passed in as params by the developer

And components cannot have multiple props with the same prop name

Render prop

A technique for explicitly passing value(s) from a component to its children

- Dev creates function which returns their component
- It’s passed to component as a prop, which the component then renders
- Enables passing values to render prop without coding custom logic in parent
- Can also avoid bubbling state up & necessitating parent re-renders

Persisting state across sessions

- Save up to 10mb of data to user's browser using localStorage
- Save data, such as form data, on state change:

```
localStorage.setItem(key, JSON.stringify(myDataObject));
```

- Then retrieve data on component load:

```
const dataString = localStorage.getItem(key);
```

With this, we can finally implement **autosave for Drupal**!

Optimizing “Pure” components

- Some components always return the same output when given the same props
- “Pure” or “Memoized” components do not re-render when props are the same
- To memoize a component class:

```
Class MyComponent extends React.PureComponent
```

- To memoize a function component:

```
Const MyPureFuncComponent = React.memo(function MyComponent...);
```

Asynchronous component loading

- Loading a component normally:

```
import { add } from './math';
// ...
console.log(add(16, 26));
```

- Loading a component asynchronously:

```
import("./math").then(math => {
  console.log(math.add(16, 26));
});
```

Lazy-loading with <Suspense>

Wrap dynamic import in React.lazy(func());

Ensures slow components are non-blocking

Wrap with:
<Suspense fallback={..}>
for ‘loading..’ component

```
const OtherComponent = React.lazy(() =>
import('./OtherComponent'));
function MyComponent() {
  return (
    <div>
      <Suspense fallback={<div>Loading...</div>}>
        <OtherComponent />
      </Suspense>
    </div>
  );
}
```

Accessibility Conerns

- Drupal’s meta info is very good
- Drupal’s HTML is very accessible OOTB
- Consider accessibility implications when fully decoupling

Server side rendering w/Node can improve accessibility

Decoupling Drupal

Methodologies

There are various approaches for decoupling with Drupal.

- Fully decoupled (“headless”)
- Pseudo-decoupled
- Progressively decoupled
- Supplemental

Fully decoupled (headless)

- Drupal acts as a content repository
- Drupal provides no public front end
- A front end is built independently
- The front end consumes data from Drupal via API

Pseudo-decoupled

The same model as fully decoupled, *plus:*

- Drupal provides layout configuration information via API
- The front end uses the layout information to assemble pages
- Enables ‘site builders’ to update site layout via Drupal

Progressively decoupled

- Drupal provides a front end as standard Drupal theme
- Decoupled apps are embedded within certain areas of website, such as
    - Within a Drupal block
    - As a specific region’s content
    - Even replacing the entire <body>

Supplemental

- Drupal provides entire front end
- Drupal developer inserts data as DOM attributes
- JS extracts data from DOM elements
- Supplemental App(s) inserted alongside Drupal elements
    - E.g. supplement form integer widget with React slider widget

Starting a new decoupled app

Starting a Decoupled Drupal project

- Contenta
- Reservoir
- Drupal Boina (Gatsby)
  Github.com/weknowinc/Drupal-boina (drupal starter)
    - For Gatsby
    - Github.com/weknowinc/Gatsby-starter-drupal-boina (gatsby starter)
- Vanilla Drupal

Exposing Drupal data

Drupal aims to be API-first, and it's headed in that direction.

- JSONAPI
  - Endorsed by Dries, de facto standard
- GraphQL
  - Has user interface for navigating data structures: ***GraphiQL***
  - Can be set up server side ***or client side***
  - Retrieves only the fields you request
  - Can be configured to simplify complex queries
- Via DOM element attributes
  - Note: Impractical for 'heavily' decoupled implementations

GraphQL

A *spec* -- Can be implemented in any language

- Server Side: Drupal module (drupal/graphql)
  - Can be configured to run custom db queries
- Client side: graphql-compose
  - Can map JSON API endpoints to graphql

GET'ing related Drupal data

Drupal fields regularly reference other entities.

Options for fetching referenced content:

- Sequential JSONAPI requests
- JSONAPI's "include=field_name" query param
  - + Jsonapi Defaults (Submodule of JSON:API Extras)
- Subrequests module
  - Enables parallel requests w/single Drupal bootstrap
- Server-side GraphQL (Server-side or Client-side)
  - ***Highly*** customizable endpoints via backend development

Looking ahead: GatsbyJS

Nothing to do with the great gatsby..

“Blazing fast site generator for react”

- Exports public API “get” calls into JSON files
- For POSTs:
  - Integrate with external services when possible
- Only deploy static files, no longer deploy Drupal

Fully decoupled static site w/Gatsby

- Use GraphQL for API calls
    - Gatsby plugin for client side GraphQL available!
- Create new app using GatsbyJS
- Deployment: Export “build artifacts” from Gatsby app
    - Downloads public API endpoints into static files!
    - Enables deploying a truly static website
- Gatsby starter: Github.com/weknowinc/Gatsby-starter-drupal-boina

Gatsby-specific techniques

Drupal site setup

- Drupal: provide content in markdown (tui_editor)
- Preprocess inline-images in markdown (gatsby-remark-drupal)
- Deploy your site directly from Drupal (build_hooks module, allows auto-builds to netlify)

Gatsby Packages

- gatsby-image
- Gatsby-source-drupal
- Gatsby-remark-drupal
- Gatsby-transformer-remark
- Gatsby-remark-images
- Gatsby-remark-external-links
- Gatsby-plugin-sharp
- Gatsby-plugin-react-helmet

Career-changing Drupal 8 PT Course

Part-time 3 month Drupal 8 Web Dev course

We **build**, **launch**, & **contribute** Drupal projects in class!

Begins **June 2nd, 2019**

Applications close as soon as class is full.

Other courses

1-2 day courses offered periodically
(Including ReactJS for Drupal)

We come to you: Tailored on-site team training available.

www.debugacademy.com

@debugacademy

Mock API’s for distributed + external + local dev

(copied, do research / rewrite this:)

- GraphQL - serve mock data with graphql-tools from Apollo GraphQL
- Node.js - generate server-side w/JSON Schema faker
- React - integrate API testing w/Jest
- Node.js user JSON Server to generate w/local sample.json files

React Context

Next.JS Fundamentals

- `<BrowserRouter>`
  - Wraps the app which has routes
- `<Link>`
  - Links to a path
- `<Route>`
  - Wraps route-specific content

```
<BrowserRouter>
   <Link
     to={`/node/${node.id}`}
     >{node.title}</Link>
   <Route
     path="/"
     component={LoginForm}
     exact={true}>
   <Route
     path="/node/:nid"
     render={({match}) => (
      <DNode nid={match.params.nid}>
     )}>
</BrowserRouter>
```

Sample Section Divider

Sample Section Divider

I’m a Section Divider

Break up sections with breather slides

Here Is An Example For You

Large callout copy can be larger than body copy, as shown here.

```
<style>
  .debug-class::after {
     Content:“gettin’ a job”;
     visibility: visible;
  }
</style>
```

Here Is An Example For You

Large callout copy can be larger than body copy, as shown here.

```
<style>
  .debug-class::after {
     content:”gettin’ a job”;
     visibility: visible;
  }
</style>
```

Heading One

Heading Two

This is the standard size of copy. Probably the smallest it will go.

Heading Three

- Lorem ipsum dolor
- Vesti ante primus luctus
- Posure ultrices ligula in turpis

Heading Two

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Vesti ante primus luctus
- Posure ultrices ligula in turpis
- Lorem ipsum dolor

Heading One

Heading Two

This is the standard size of copy. Probably the smallest it will go.

Heading Three

- Lorem ipsum dolor
- Vesti ante primus luctus
- Posure ultrices ligula in turpis

Heading Two

Lorem ipsum dolor sit amet, consectetur adipiscing elit.

- Vesti ante primus luctus
- Posure ultrices ligula in turpis
- Lorem ipsum dolor

Passing Variables by Value ($variable)

When a variable is passed by value, it means that the function called actually receives a copy of the variable (this is the default PHP behavior).

If the function makes any changes to the variable it receives, it only makes changes to the copy of the variable - the original variable remains unchanged.

```
function myfunction($variable) {
  // Any work done on $variable in here will only
  // happen on a copy of the $variable received.
  // When the function completes, the copy of $variable
  // within the function will simply cease to exist
}
```

Passing Variables by Value ($variable)

When a variable is passed by value, it means that the function called actually receives a copy of the variable (this is the default PHP behavior).

If the function makes any changes to the variable it receives, it only makes changes to the copy of the variable - the original variable remains unchanged.

```
function myfunction($variable) {
  // Any work done on $variable in here will only
  // happen on a copy of the $variable received.
  // When the function completes, the copy of $variable
  // within the function will simply cease to exist
}
```

Sample Heading

I am sample text, that goes with the above sample heading. Type anything you want in this box, it'll explain all the good stuff you want

- Item one
- Item two
- Item three
- Item four
- Item five

I am a quote or impactful statement!

55

Try It With Me

```
function myfunction($variable) {
  // Any work done on $variable in here will only
  // happen on a copy of (highlight text color), to indicate
change.
  // When the function completes, the copy of $variable
  // within the function will simply cease to exist
}
```


FINANCE REPORT
ACCOUNT
REPORT
DASHBOARD > INCOME
DAILY
WEEKLY
MONTHLY
MON
TUE
WED
THU
FRI
SAT
SUN
TOTAL INCOME
37.91%
31.86%
30.23%

Sample Heading Goes Here

I am sample copy.

Be mindful of how `.copy-code` much copy goes on each slide.

Short digestible content is the best.

I am sample copy.

Be mindful of how much copy goes on each slide.

Short digestible content is the best.

Sample Heading Goes Here

I am sample copy.

Be mindful of how `.copy-code` much copy goes on each slide.

Short digestible content is the best.

I am sample copy.

Be mindful of how much copy goes on each slide.

Short digestible content is the best.

Career-changing Drupal 8 PT Course

Part-time 3 month Drupal 8 Web Dev course

We **build**, **launch**, & **contribute** Drupal projects in class!

Begins June 3rd 2018 **(Next weekend!)**

Applications close as soon as class is full.

Other courses

1-2 day courses offered periodically **(Including ReactJS for Drupal)**

We come to you: Tailored on-site team training available.

www.debugacademy.com

@debugacademy

Thank You!

www.debugacademy.com

@debugacademy

Thank You!

www.debugacademy.com

@debugacademy