@DARCAR31

DARIA CARAWAY

HOW TO HAVE AN AMICABLE BREAKUP WITH A JAVASCRIPT LIBRARY

@DARCAR31

Software Engineer at workday.

Living in

Boulder, CO

My cat

Odin

DARIA CARAWAY

ABOUT ME

@DARCAR31

DATING AND JAVASCRIPT LIBRARIES

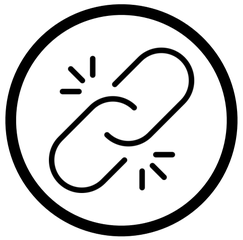
CHOOSING A JAVASCRIPT LIBRARY IS A LOT LIKE DATING: THE LONGER YOU COMMIT YOURSELF, THE MESSIER IT IS WHEN YOU DECIDE TO BREAK UP.

COMMON RELATIONSHIP

Honeymoon Phase

Everything is great, they make your life so easy.

Attached at the Hip



You start using the library everywhere.

Opened a Joint Banking Account

You are dependent on The library.

They broke your heart:
Library is deprecated

There is someone else:
Found a more feature rich library

THE BREAKUP

You come with too much baggage:
Library bundle is too large

It's not you it's me:
Product needs change

POST-BREAKUP WHAT NOW?

Completely Move On: Remove all references to the old library and start clean.

Ghost Them: Ignore the usages in the code exist and build up tech debt.

Remain friends: Remove references piece by piece while implementing the new solution.

@DARCAR31

HOW TO PREPARE FOR THE INEVITABLE FUTURE BREAKUP?

Speed dating

Prototype multiple libraries early

Easily bail on bad relationships

Quickly switch out libraries without sacrificing code clarity or maintainability.

Prevent future mistakes

Write useful tests that won't break when you change the underlying implementation

PROTOTYPE OUT MULTIPLE POTENTIAL LIBRARIES,
IN AN INTEGRATED ENVIRONMENT.

SPEED DATING

GETTING TO KNOW THE DATING POOL AND EACH LIBRARY

- How do you interface with the library?
- How does each library compare and differ from others?
- Learn about design and coding styles in the community.

GETTING TO KNOW YOURSELF

- What are you expectations for customization, performance, etc...
- What feature set does your app need now? And in the future?
- Share prototypes with team members and stakeholders.

@DARCAR31

YOU THINK YOU'VE FOUND "THE ONE"

SPEED DATING ALLOWS YOU TO FEEL MORE CONFIDENT IN YOUR DECISION, AND MAKES THE LIBRARY USEFUL FOR LONGER.

QUICKLY SWITCH OUT LIBRARIES WITHOUT SACRIFICING CODE CLARITY OR MAINTAINABILITY.

PREPARE TO BAIL ON BAD RELATIONSHIPS

. . . with wrapper components

THE PRENUPTIAL AGREEMENT & WRAPPER COMPONENTS

- Create a contract to fit our application's use cases
- Normalize that interface with a wrapper component

THE PRENUPTIAL AGREEMENT & WRAPPER COMPONENTS

- Create a contract to fit our application's use cases
- Normalize that interface with a wrapper component

WHAT IS THE CONTRACT?

TypeScript/Flow

React Prop Types

JSDoc

Animals and photos from: www.seattlehumane.org

DEMO APP INTRO

Discover Adoptable Pets In Seattle!

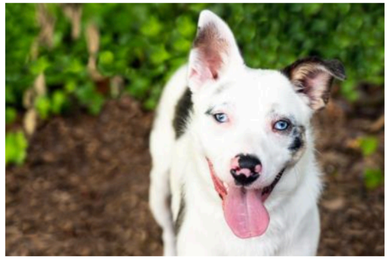
Fetch Random Pet

Favorite

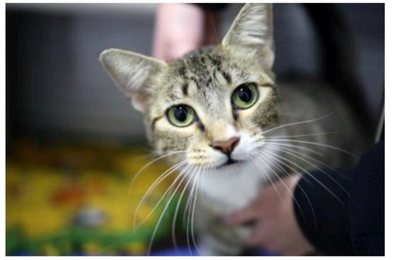
Clear Favorites

Meet Puma!

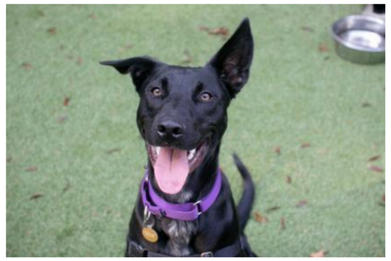
Meet Kitty Toldeo!



Meet Robin!

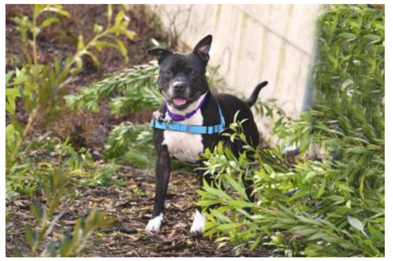


Meet Linsay!



Meet Ele Ele!

Meet Your Highness Gracie!



Meet Bruno!

WRAPPER EXAMPLE - BUTTON

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}
```

```tsx
export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

WRAPPER EXAMPLE - BUTTON

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}
```

```tsx
export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

WRAPPER EXAMPLE – BUTTON

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}
```

```tsx
export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

WRAPPER EXAMPLE - BUTTON

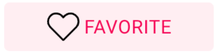
```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}
```

```tsx
export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

WRAPPER COMPARISON – GROMMET VS MATERIAL UI

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

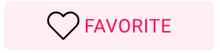

```tsx
import MaterialUIButton  from '@material-ui/core/Button'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

     return (
      <MaterialUIButton
        onClick={onClick}
        color={primary ? 'primary' : 'secondary'}
        data-testid={testId}
      >
        {icon}
        {label}
      </MaterialUIButton>
    )
  }
}
```



WRAPPER COMPARISON – GROMMET VS MATERIAL UI

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

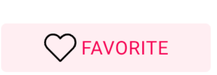

```tsx
import MaterialUIButton  from '@material-ui/core/Button'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

     return (
      <MaterialUIButton
        onClick={onClick}
        color={primary ? 'primary' : 'secondary'}
        data-testid={testId}
      >
        {icon}
        {label}
      </MaterialUIButton>
    )
  }
}
```



WRAPPER COMPARISON – GROMMET VS MATERIAL UI

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```

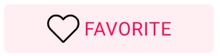

```tsx
import MaterialUIButton  from '@material-ui/core/Button'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <MaterialUIButton
        onClick={onClick}
        color={primary ? 'primary' : 'secondary'}
        data-testid={testId}
      >
        {icon}
        {label}
      </MaterialUIButton>
    )
  }
}
```



WRAPPER COMPARISON – GROMMET VS MATERIAL UI

```tsx
import { Button as GrommetButton } from 'grommet'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <GrommetButton
        onClick={onClick}
        icon={icon}
        label={label}
        primary={primary}
        data-testid={testId}
      />
    )
  }
}
```


```tsx
import MaterialUIButton  from '@material-ui/core/Button'

interface ButtonProps {
  icon?: React.ReactElement
  label?: string
  primary?: boolean
  testId?: string
  onClick: ((evt: React.MouseEvent) => void)
}

export class ButtonComponent extends React.Component<ButtonProps> {
  render() {
    const { icon, label, onClick, primary, testId } = this.props

    return (
      <MaterialUIButton
        onClick={onClick}
        color={primary ? 'primary' : 'secondary'}
        data-testid={testId}
      >
        {icon}
        {label}
      </MaterialUIButton>
    )
  }
}
```



WRAPPER EXAMPLE – API UTILITY

```
import * as superagent from 'superagent'

export const ApiUtility = {
  get<ResolveType>(url: string): Promise<ResolveType> {
    return superagent
      .get(url)
      .then(response => response.body)
  },
  post<PayloadType, ResolveType>(url: string, payload: PayloadType) : Promise<ResolveType> {
    return superagent
      .post(url)
      .send(JSON.stringify(payload))
      .then(response => response.body)
  }
}
```

WRAPPER EXAMPLE - CUSTOM COMPONENT

Odin's closeup #1

```
interface CardProps {
  imgUrl: string
  text: string
  width: number
}

export class CardComponent extends React.Component<CardProps> {
  render() {
    const { imgUrl, text, width } = this.props

    return (
      <div className="card" style={{width}}>
        <img alt="Card" src={imgUrl}/>
        <div className="card-title">{text}</div>
      </div>
    )
  }
}
```

WRAPPER EXAMPLE - CUSTOM COMPONENT

```
interface CardProps {
  imgUrl: string
  text: string
  width: number
}
```

Odin's closeup #1

```
export class CardComponent extends React.Component<CardProps> {
  render() {
    const { imgUrl, text, width } = this.props

    return (
      <div className="card" style={{width}}>
        <img alt="Card" src={imgUrl}/>
        <div className="card-title">{text}</div>
      </div>
    )
  }
}
```

WRITE TESTS THAT DO NOT RELY ON IMPLEMENTATION DETAILS.

PREVENT FUTURE MISTAKES

WRAPPER USAGE - BUTTON

```
export class ButtonSidebar extends React.Component<ButtonSidebarProps> {
  render() {
    const { getRandomPet, favoritePet, clearFavorites } = this.props

    return (
      <div className='button-sidebar-container'>
        <ButtonComponent testId='button-get-pet' primary={true} label='Fetch Random Pet' onClick={getRandomPet} />
        <div className='button-sidebar-favorite-container'>
          <ButtonComponent testId='button-favorite-pet' icon={<HeartIcon/>} label='Favorite' onClick={favoritePet} />
          <ButtonComponent testId='button-clear' label='Clear Favorites' onClick={clearFavorites} />
        </div>
      </div>
    )
  }
}
```

WRAPPER USAGE - BUTTON

```
export class ButtonSidebar extends React.Component<ButtonSidebarProps> {
  render() {
    const { getRandomPet, favoritePet, clearFavorites } = this.props

    return (
      <div className='button-sidebar-container'>
        <ButtonComponent testId='button-get-pet' primary={true} label='Fetch Random Pet' onClick={getRandomPet} />
        <div className='button-sidebar-favorite-container'>
          <ButtonComponent testId='button-favorite-pet' icon={<HeartIcon/>} label='Favorite' onClick={favoritePet} />
          <ButtonComponent testId='button-clear' label='Clear Favorites' onClick={clearFavorites} />
        </div>
      </div>
    )
  }
}
```

TEST EXAMPLE - BUTTON

```
it('renders the button sidebar', () => {
  const { getByTestId } = render(
    <ButtonSidebar
      getRandomPet={getRandomPetMock}
      favoritePet={favoritePetMock}
      clearFavorites={clearFavoritesMock}
    />)

  expect(getByTestId('button-get-pet')).toBeDefined()
  expect(getByTestId('button-favorite-pet')).toBeDefined()
  expect(getByTestId('button-clear')).toBeDefined()
})
```

```
it('calls the getCat click function', () => {
  const { getByTestId } = render(
    <ButtonSidebar
      getRandomPet={getRandomPetMock}
      favoritePet={favoritePetMock}
      clearFavorites={clearFavoritesMock}
    />)

  fireEvent.click(getByTestId('button-get-pet'))

  expect(getRandomPetMock).toHaveBeenCalled()
})
```

DEMO – BREAKING UP WITH GROMMET AND SUPERAGENT

WHAT DOES THIS MEAN?

@DARCAR31

BREAKUPS DON'T HAVE TO BE MESSY

Remain nimble in this ever changing environment

Don’t be tied down to your past decisions

Contain library interactions into wrapper components and control the wrapper interface

LIBRARY OPINIONS DON'T HAVE TO AFFECT YOUR CODE DECISIONS

TRY THIS TECHNIQUE AND SEE IF IT CAN HELP SAVE YOUR CODE FROM SOME MESSY BREAKUPS.

CAN THIS WORK FOR YOU?

@DARCAR31

THANK YOU

https://v2.grommet.io/

https://material-ui.com/

https://github.com/visionmedia/superagent

https://www.seattlehumane.org