React In A Design Process

Chris Williams
Senior Software Engineer, Direct Supply
github.com/chriswxyz

Blueprint v3
View on GitHub

Dark theme ⇧D
Search... ⇧S

BLUEPRINT
CORE 3.18.1
- Accessibility
- Classes new
- Colors
- Typography
- Variables

COMPONENTS
- Breadcrumbs
- Button
- Button group
- Callout
- Card
  - Elevation
  - Props
  - CSS
- Collapse
- Collapsible list
- Divider new
- Editable text
- HTML elements new
- HTML table
- Hotkeys
- Icon
- Menu
- Navbar
- Non-ideal state
- Overflow list new

Card

Edit this page

A card is a bounded unit of UI content with a solid background color.

Analytical applications

User interfaces that enable people to interact smoothly with data, ask better questions, and make better decisions.

Explore products

Props
Interactive
Elevation
0 1 2 3 4

View source on GitHub

Elevation

Apply an `elevation` value to a card to apply a drop shadow that simulates height in the UI. Five elevations are supported, from 0 to 4.

The `Classes.ELEVATION_*` constants can be used on any element (not just a `Card`) to apply the drop shadow.

Props

This component is a simple stateless container for its children.

```
import { Button, Card, Elevation } from "@blueprintjs/core";

<Card interactive={true} elevation={Elevation.TWO}>
    <h5><a href="#">Card heading</a></h5>
    <p>Card content</p>
    <Button>Submit</Button>
</Card>
```

`interface ICardProps extends IProps, HTMLAttributes` @blueprintjs/core

Props	Description
`className`	`string` A space-delimited list of class names to pass along to a child element. Inherited from `IProps.className`

Project Clarity | Documentation | Icons | Community | What's New

V3 is in prelease. Install it using the @next tag from NPM.

VERSION V3

- Component Status
- Get Started
- Patterns
  - Application Layout
  - Color Palette
  - Internationalization
  - Navigation
  - Themes
  - Typography
- Components
  - Accordion
  - Alerts
  - Badges
  - Buttons
  - Button Group
  - Cards
  - Checkboxes
  - Datagrid
  - Date Picker
  - Dropdowns
  - Forms
  - Grid
  - Header
  - Inputs
  - Labels
  - Lists
  - Login Page
  - Modals
  - Password
  - Progress Bars
  - Radio Buttons
  - Select
  - Sidenav
  - Signposts
  - Spinners
  - Stack View
  - Stepper
  - Tables
  - Tabs
  - Textareas
  - Toggle Switches

Buttons

Design Guidelines | Code & Examples

Buttons allow an application to communicate action and direct user intent.

Three different types

SOLID BUTTON | OUTLINE BUTTON | FLAT BUTTON

Solid buttons look heavy on purpose. They direct the user's attention to the **primary action** the application is suggesting that the user take.

Outline buttons provide a lighter weight button style. They are used to indicate a **secondary action** that compliments a primary action or to reduce visual noise when there are many action of **equal** importance on the page.

Flat buttons are used in multiple scenarios. They are used as **tertiary buttons**. They can also be used inline because they are different from content in style and recognizable as buttons alongside content.

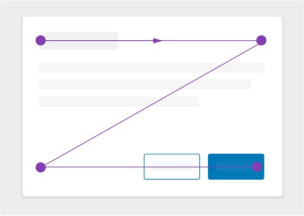
FLAT BUTTON
Use flat buttons when a user is expected to **take an action**.

VS

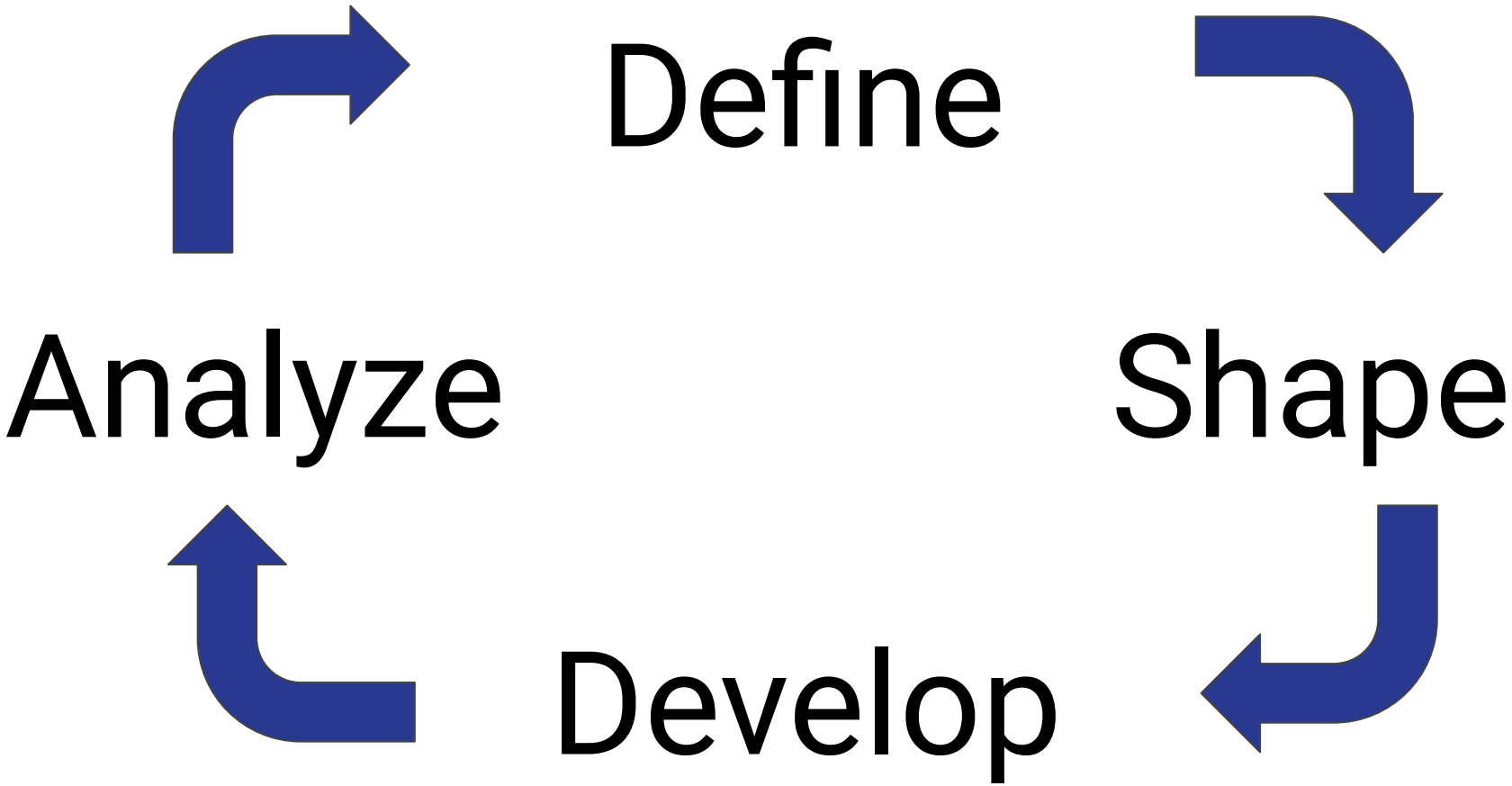
Link
Use a link when a user is expected to be **taken to a different page**.

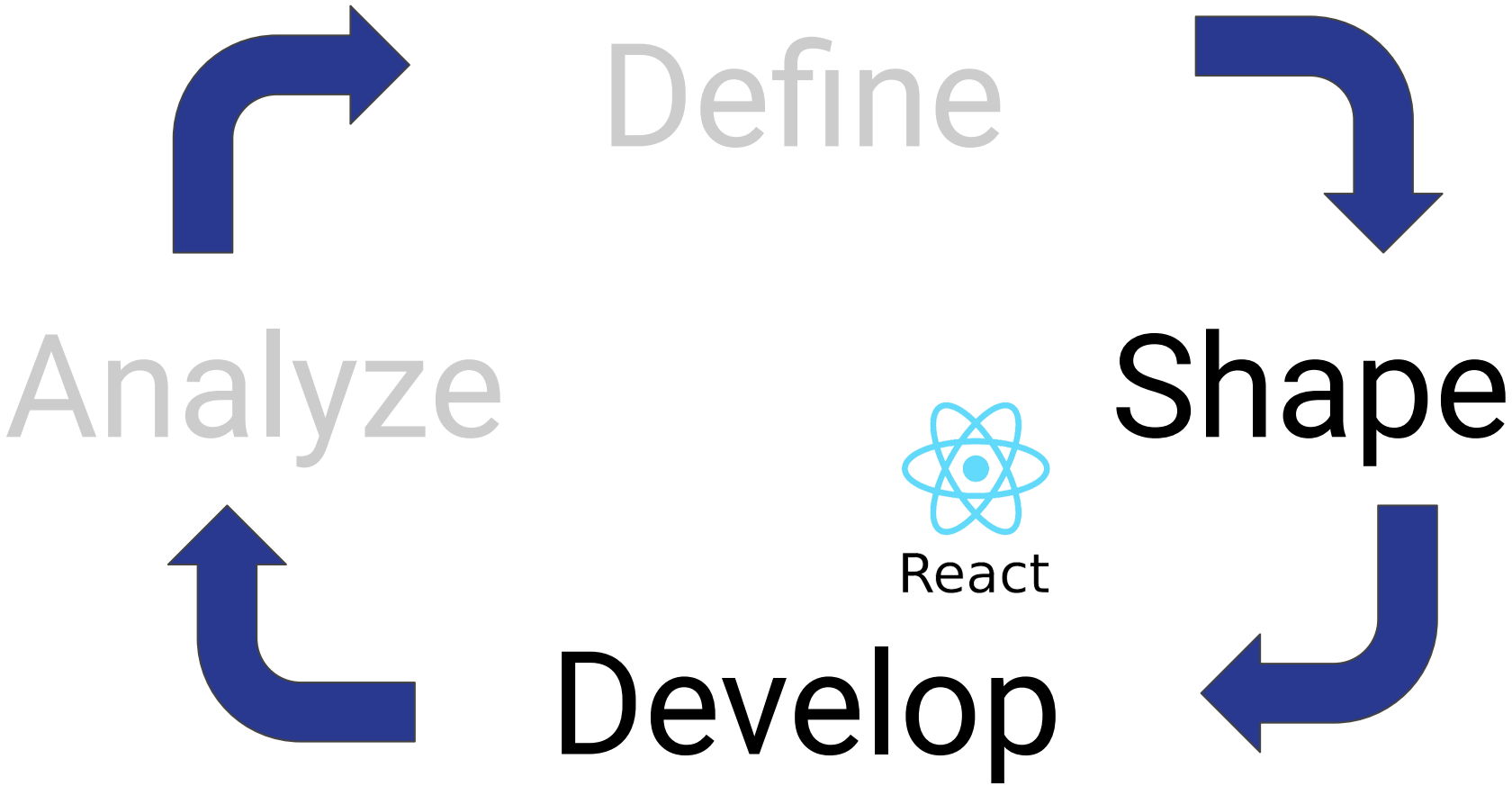
Placement

There are two distinct patterns when it comes to the placement of a button.




Dashboard
View


Define
Shape
Develop
Analyze


Define
Shape
Develop
Analyze
React

Shaping

An interactive, hands-on collaboration for devs and users

Storybook

Welcome to the OGM3 Prototype Storybook.

This storybook provides a place to experiment with new OGM3 experiences. 💻🔨

Copy the `example.proto.tsx` file from the `_prototypes` folder to get started.

Usability Tests

Storybook

Not secure | w10-d-chriswil:6008/iframe.html?id=activity-stream-prototype--hello

2. Test

Facilitator

Describe the scenario to your first user, and let the magic happen.

> "Lets say you receive an email from a customer asking you to *BLAH*. Where would you start?"

Tip - Get Them To Think Out Loud

If the user asks what they are supposed to do, encourage them to think out loud instead of guidi
them to your own answer.

> "Am I supposed to click on BLAH?"

Responses:

> "What do you think will happen when you click on BLAH?"
> "Go ahead and try clicking on BLAH."
> "Tell me more."

Usability Session Worksheet - Activity Stream

Date of Test Session: 2019-07-31

Feature Name: Activity Stream + Descriptions

User Jobs enabled by this Feature: Working collaboratively with your analyst team to maintain a customer.

User Pains relieved by this Feature: Forgetting why things were changed or deleted.

Test Scenarios:

1. You want to leave the contact emails for YourNet in an obvious location, even for people who don't normally manage this customer.
   Doug: doug@yournet.com
   Tom: tom@yournet.com

2. Doug from "YourNet Care Systems" emailed you and wanted to know why his facilities are suddenly able to purchase Pumpkin Sauce. He says they started being able to purchase it 2 days ago.

Use the tools you're comfortable with to drive fast iterations with direct evidence from users

Develop

Making non-UI-wizards successful and happy

TypeScript

```
return <Page title='Verify Onboarding' subtitle={subtitle} breadcrumbs={crumbs}>
    <PageSection title='Verify Locations'>
        <DataTable
            columns={columns}
            data={locationValidation}
            itemId={x => x.id}
            expandContent={x => <ValidationDetails row={x} />}
            onExpand={noop}

        />
    </PageSe
</Page >;
```

```
actions
checked
children
```

```
(JSX attribute) actions?: Reac
t.ReactNode
```

TypeScript

- Crucial for work as a team
- Communicate semantics and intent
- Bad data states fail to compile
- Java/C# devs feel a bit more at home

```
function StatusIcon(props: {
    row: Duck.LocationVerification;
}): JSX.Element

Type '{ rwo: LocationVerification; }' is
not assignable to type
'IntrinsicAttributes & { row:
LocationVerification; }'.
  Property 'rwo' does not exist on type
'IntrinsicAttributes & { row:
LocationVerification; }'. ts(2322)

Peek Problem    No quick fixes available

⇒ <StatusIcon rwo={x} />
```

Give Real Advice

- Real advice is straightforward and actionable
- Give devs concrete examples, not just theory
- Help them understand what NOT to do

File Organization

Folder-by-feature

Features are organized by placing the components, hubs, ducks, tests, and stories together in one folder. This helps a developer quickly locate and identify related code. This is the opposite of folder-by-type, which separates code files by technical category, and leads to jumping around trying to find a file.

Filenames

Filenames have a segment that indicate what kind of file they are (when it's not obvious from the extension).

Kind	File
Component	`my-component.tsx`
Controller	`my-feature.page.tsx`
Duck	`my-feature.duck.ts`
Hub	`my-feature.hub.ts`
Test	`my-feature.spec.ts`

Filenames are lower-kebab-cased.

Less is More

```
return <button style={{backgroundColor: '???'}}>Hello</button>;
```

```
return <MyButton text='Hello' color='' />;
```

- danger
- primary
- secondary
- warning

Less is More

```
return <div style={{padding: '1.68rem'}}>Hello</div>;
```

```
return <PageSection density='' >Hello</PageSection>;
```

- compact
- normal
- touch

Functional CSS

- Past experience and results
  - 👍 from developers that don’t like UI
  - basics of CSS
  - Is not “magical” or “powerful”
- You will write 99-100% less CSS
- Best used in a component-based system
- Easy to migrate bits and pieces

```
const btn1 = <button className='home-page-cta'>Semantic</button>;

const btn2 = <button className='bg-blue border-pink padding-3 rounded'>Functional</button>;
```

Functional CSS

	CSS Written	Reuse
BEM	High	Low
Framework	Medium	Medium
f(css)	Low	High

Storybook

Welcome to the OGM3 Storybook 🎉

This storybook is part component catalog, part visual development tool, part manual regression test, and 💯 percent awesome.

🖼️ Component Catalog

Browse through existing components in OGM3 and see them in various states of action!

🛠️ Visual Development

Copy a page or component story out of the `_boilerplate` folder and start working on new UI!

🧪 Regression Test

Changes to components are easier to verify since data can be experimented with, without running the site!

Component-First Design

- Build a React component in Storybook
- Mock out the interaction with React.state
- Usability test
- Component Props ~= Your API

Debugging - View, or State?

Search (text or /regex/)

- CommonProvider DragDropContext
  - CommonProvider
    - IntlProvider
      - Provider
        - Context.Provider
          - Router
            - RouterContext
              - Layout InjectIntl +1
                - Layout Connect +1
                - Context.Provider

Button

```
props
  data-test-id: "mass-import-button"
  dsxIcon: "upload"
  isHidden: false
  onClick: fn()
  style: "primary-outline"
  value: "Mass Upload"
  new prop : ""
```

Inspector | DSSI OGM

filter... Commit

@@INIT	3:12:13.78
@@router/LOCATION_CHANGE	+00:00.00
@ca:LOADED_CUSTOMERS_PAGE_PEND…	+00:00.02
@loa:SET_SIGN_ON_TYPE	+00:00.01
@ca:LOADED_CUSTOMERS_PAGE_SUCC…	+00:00.09
@ca:SELECTED_CUSTOMER_PENDING	+00:01.06
@@router/LOCATION_CHANGE	+00:00.06
@customers:HOME_PAGE_LOADED_PE…	+00:00.01
LOADING_BLUEPRINTS	+00:00.00
@ca:SELECTED_CUSTOMER_SUCCESS	+00:00.03
@customers:HOME_PAGE_LOADED_SU…	+00:00.03
@ta:TIMER_CLEAR	+00:00.01
@ta:TIMER_START	+00:00.00
@ta:TIMEOUT_REFRESH	+00:00.00
LOADING_BLUEPRINTS_SUCCESS	+00:00:00

Diff | Action | State | Diff | Trace | Test

Tree | Raw

```
▼ blueprintManagement (pin)
  ▼ blueprints (pin)
      inProcess (pin): ~~true~~ => false
    ▶ payload (pin): { 0: {…}, 1: {…}, 2: {…}, … }
      totalResults (pin): 0 => 8
      totalPages (pin): 0 => 1
```

Ducks 🦆

A "duck" is a file that contains

- a reducer
- actions that the reducer can handle
- functions that create the actions
- the state that the reducer operates on
- customers.duck.ts

Explicit, useful tools that improve a dev’s life

- Types
- Small, clear CSS
- Play and experiment with UIs
- Components drive API design
- Spatial awareness

Question Time!

Or, ask me later:

chris.williams@directsupply.com

github.com/chriswxyz