Breakpoints

Breakpoints are customizable widths that determine how your responsive layout behaves across device or viewport sizes in Bootstrap.

Available breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as *grid tiers*, for building responsively. These breakpoints can be customized if you're using our source Sass files.

Breakpoint	Class infix	Dimensions
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

Custom breakpoints

If you are looking to use custom breakpoints, you must wrap your application with a theme provider and use the `breakpoints` prop to specify the breakpoints you will use. This ensures that components such as `Row` or `Col` can parse the correct custom breakpoint props.

```
<ThemeProvider
  breakpoints={['xxxl', 'xxl', 'xl', 'lg', 'md', 'sm', 'xs', 'xxs']}
  minBreakpoint="xxs"
>
  <div>Your app...</div>
</ThemeProvider>;
```

More information

For more information about breakpoints, see the Bootstrap documentation.

API

ThemeProvider

```
import ThemeProvider from 'react-bootstrap/ThemeProvider'
```

Name	Type	Default	Description
`prefixes`	`object`	`{}`	An object mapping of Bootstrap component classes that map to a custom class.

Name	Type	Default	Description
			**Note: Changing prefixes is an escape hatch and generally shouldn't be used. For more information, see here.
`breakpoints`	`arrayOf`	`['xxl', 'xl', 'lg', 'md', 'sm', 'xs']`	An array of breakpoints that your application supports. Defaults to the standard Bootstrap breakpoints.
`minBreakpoint`	`string`	`'xs'`	The minimum breakpoint used by your application. Defaults to the smallest of the standard Bootstrap breakpoints.
`dir`	`string`		Indicates the directionality of the application's text. Use `rtl` to set text as "right to left".
`children` Required			

Grid system

Bootstrap's grid system uses a series of containers, rows, and columns to layout and align content. It's built with flexbox and is fully responsive. Below is an example and an in-depth look at how the grid comes together.

New to or unfamiliar with flexbox? Read this CSS Tricks flexbox guide for background, terminology, guidelines, and code snippets.

Container

Containers provide a means to center and horizontally pad your site's contents. Use `Container` for a responsive pixel width.

RESULT

1 of 1

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function ContainerExample() {
  return (
    <Container>
      <Row>
        <Col>1 of 1</Col>
      </Row>
    </Container>
  );
}

export default ContainerExample;
```

Fluid Container

You can use `<Container fluid />` for width: 100% across all viewport and device sizes.

RESULT

1 of 1

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function ContainerFluidExample() {
  return (
    <Container fluid>
      <Row>
        <Col>1 of 1</Col>
      </Row>
    </Container>
```

```
  );
}

export default ContainerFluidExample;
```

You can set breakpoints for the `fluid` prop. Setting it to a breakpoint (`sm, md, lg, xl, xxl`) will set the `Container` as fluid until the specified breakpoint.

RESULT

1 of 1

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function ContainerFluidBreakpointExample() {
  return (
    <Container fluid="md">
      <Row>
        <Col>1 of 1</Col>
      </Row>
    </Container>
  );
}

export default ContainerFluidBreakpointExample;
```

Auto-layout columns

When no column widths are specified the `Col` component will render equal width columns

RESULT

1 of 2 | 2 of 2

1 of 3 | 2 of 3 | 3 of 3

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function AutoLayoutExample() {
  return (
    <Container>
      <Row>
        <Col>1 of 2</Col>
        <Col>2 of 2</Col>
      </Row>
      <Row>
        <Col>1 of 3</Col>
        <Col>2 of 3</Col>
        <Col>3 of 3</Col>
      </Row>
    </Container>
  );
}

export default AutoLayoutExample;
```

Setting one column width

Auto-layout for flexbox grid columns also means you can set the width of one column and have the sibling columns automatically resize around it. You may use predefined grid classes (as shown below), grid mixins, or inline widths. Note that the other columns will resize no matter the width of the center column.

RESULT

1 of 3	2 of 3 (wider)	3 of 3
1 of 3	2 of 3 (wider)	3 of 3

LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function AutoLayoutSizingExample() {
  return (
    <Container>
      <Row>
        <Col>1 of 3</Col>
        <Col xs={6}>2 of 3 (wider)</Col>
        <Col>3 of 3</Col>
      </Row>
      <Row>
        <Col>1 of 3</Col>
        <Col xs={5}>2 of 3 (wider)</Col>
        <Col>3 of 3</Col>
      </Row>
    </Container>
  );
}

export default AutoLayoutSizingExample;
```

Variable width content

Set the column value (for any breakpoint size) to `auto` to size columns based on the natural width of their content.

RESULT

1 of 3	Variable width content	3 of 3
1 of 3	Variable width content	3 of 3

LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function AutoLayoutVariableExample() {
  return (
    <Container>
      <Row className="justify-content-md-center">
        <Col xs lg="2">
          1 of 3
        </Col>
        <Col md="auto">Variable width content</Col>
```

```
        <Col xs lg="2">
          3 of 3
        </Col>
      </Row>
      <Row>
        <Col>1 of 3</Col>
        <Col md="auto">Variable width content</Col>
        <Col xs lg="2">
          3 of 3
        </Col>
      </Row>
    </Container>
  );
}

export default AutoLayoutVariableExample;
```

Responsive grids

The `Col` lets you specify column widths across 6 breakpoint sizes (xs, sm, md, lg, xl and xxl). For every breakpoint, you can specify the amount of columns to span, or set the prop to `<Col lg={true} />` for auto layout widths.

RESULT

sm=8	sm=4

sm=true	sm=true	sm=true

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function ResponsiveAutoExample() {
  return (
    <Container>
      <Row>
        <Col sm={8}>sm=8</Col>
        <Col sm={4}>sm=4</Col>
      </Row>
      <Row>
        <Col sm>sm=true</Col>
        <Col sm>sm=true</Col>
        <Col sm>sm=true</Col>
      </Row>
    </Container>
  );
}

export default ResponsiveAutoExample;
```

You can also mix and match breakpoints to create different grids depending on the screen size.

RESULT

xs=12 md=8	xs=6 md=4

xs=6 md=4	xs=6 md=4	xs=6 md=4

xs=6	xs=6

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function ResponsiveExample() {
  return (
    <Container>
      {/* Stack the columns on mobile by making one full-width and the
other half-width */}
      <Row>
        <Col xs={12} md={8}>
          xs=12 md=8
        </Col>
        <Col xs={6} md={4}>
          xs=6 md=4
        </Col>
      </Row>

      {/* Columns start at 50% wide on mobile and bump up to 33.3% wide
on desktop */}
      <Row>
        <Col xs={6} md={4}>
          xs=6 md=4
        </Col>
        <Col xs={6} md={4}>
          xs=6 md=4
        </Col>
        <Col xs={6} md={4}>
          xs=6 md=4
        </Col>
      </Row>

      {/* Columns are always 50% wide, on mobile and desktop */}
      <Row>
        <Col xs={6}>xs=6</Col>
        <Col xs={6}>xs=6</Col>
      </Row>
    </Container>
  );
}

export default ResponsiveExample;
```

The `Col` breakpoint props also have a more complicated `object` prop form: `{span: number, order: number, offset: number}` for specifying offsets and ordering effects.

You can use the `order` property to control the **visual order** of your content.

RESULT

First, but unordered	Second, but last	Third, but second

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function OrderingExample() {
  return (
    <Container>
      <Row>
        <Col xs>First, but unordered</Col>
```

```
        <Col xs={{ order: 12 }}>Second, but last</Col>
        <Col xs={{ order: 1 }}>Third, but second</Col>
      </Row>
    </Container>
  );
}

export default OrderingExample;
```

The `order` property also supports `first` (`order: -1`) and `last` (`order: $columns+1`).

RESULT

Third, but first	Second, but unordered	First, but last

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function OrderingFirstLastExample() {
  return (
    <Container>
      <Row>
        <Col xs={{ order: 'last' }}>First, but last</Col>
        <Col xs>Second, but unordered</Col>
        <Col xs={{ order: 'first' }}>Third, but first</Col>
      </Row>
    </Container>
  );
}

export default OrderingFirstLastExample;
```

For offsetting grid columns you can set an `offset` value or for a more general layout, use the margin class utilities.

RESULT

md=4	md={{ span: 4, offset: 4 }}
md={{ span: 3, offset: 3 }}	md={{ span: 3, offset: 3 }}
md={{ span: 6, offset: 3 }}	

LIVE EDITOR

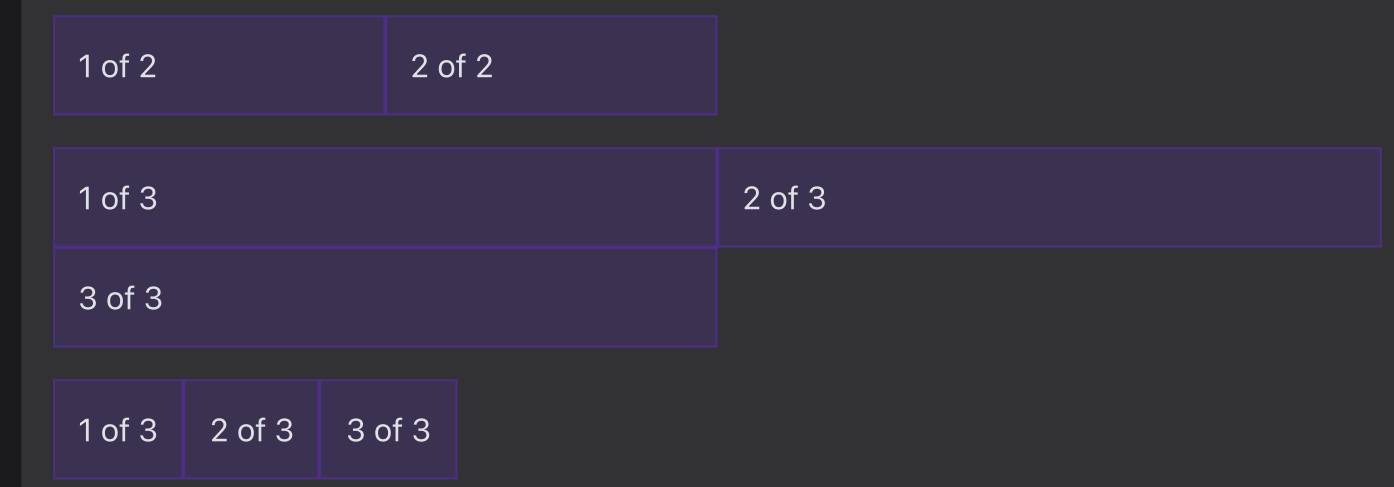
```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function OffsettingExample() {
  return (
    <Container>
      <Row>
        <Col md={4}>md=4</Col>
        <Col md={{ span: 4, offset: 4 }}>{`md={{ span: 4, offset: 4 }}`}
</Col>
      </Row>
      <Row>
        <Col md={{ span: 3, offset: 3 }}>{`md={{ span: 3, offset: 3 }}`}
</Col>
```

```
        <Col md={{ span: 3, offset: 3 }}>{`md={{ span: 3, offset: 3 }}`}
</Col>
      </Row>
      <Row>
        <Col md={{ span: 6, offset: 3 }}>{`md={{ span: 6, offset: 3 }}`}
</Col>
      </Row>
    </Container>
  );
}

export default OffsettingExample;
```

Setting column widths in Row

The `Row` lets you specify column widths across 6 breakpoint sizes (xs, sm, md, lg, xl and xxl). For every breakpoint, you can specify the amount of columns that will fit next to each other. You can also specify `auto` to set the columns to their natural widths.

RESULT



LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function RowColLayoutExample() {
  return (
    <Container>
      <Row xs={2} md={4} lg={6}>
        <Col>1 of 2</Col>
        <Col>2 of 2</Col>
      </Row>
      <Row xs={1} md={2}>
        <Col>1 of 3</Col>
        <Col>2 of 3</Col>
        <Col>3 of 3</Col>
      </Row>
      <Row xs="auto">
        <Col>1 of 3</Col>
        <Col>2 of 3</Col>
        <Col>3 of 3</Col>
      </Row>
    </Container>
  );
}

export default RowColLayoutExample;
```

Note that `Row` column widths will override `Col` widths set on lower breakpoints when viewed on larger screens. The `<Col xs={6} />` size will be overridden by `<Row md={4} />` on medium and larger screens.

RESULT

1 of 3	2 of 3	3 of 3

LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function RowColLayoutColWidthBreakpointExample() {
  return (
    <Container>
      <Row md={4}>
        <Col>1 of 3</Col>
        <Col xs={6}>2 of 3</Col>
        <Col>3 of 3</Col>
      </Row>
    </Container>
  );
}

export default RowColLayoutColWidthBreakpointExample;
```

API

Container

```jsx
import Container from 'react-bootstrap/Container'
```

Name	Type	Default	Description
bsPrefix	string	`'container'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
fluid	`true` | `"sm"` | `"md"` | `"lg"` | `"xl"` | `"xxl"`	`false`	Allow the Container to fill all of its available horizontal space.
as	elementType	`'div'`	You can use a custom element for this component

Row

```jsx
import Row from 'react-bootstrap/Row'
```

Name	Type	Default	Description
bsPrefix	string	`'row'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	elementType	`'div'`	You can use a custom element type for this component.
xs	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on extra small devices (<576px).

Name	Type	Default	Description
			Use `auto` to give columns their natural widths.
sm	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on small devices (≥576px). Use `auto` to give columns their natural widths.
md	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on medium devices (≥768px). Use `auto` to give columns their natural widths.
lg	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on large devices (≥992px). Use `auto` to give columns their natural widths.
xl	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on extra large devices (≥1200px). Use `auto` to give columns their natural widths.
xxl	`number` | `'auto'` | `{ cols: number` | `'auto' }`		The number of columns that will fit next to each other on extra extra large devices (≥1400px). Use `auto` to give columns their natural widths.

Col

```
import Col from 'react-bootstrap/Col'
```

Name	Type	Default	Description
bsPrefix	`string`	`'col'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	`elementType`		You can use a custom element type for this component.
xs	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on extra small devices (<576px)
sm	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on small devices (≥576px)
md	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on medium devices (≥768px)

Name	Type	Default	Description
lg	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on large devices (≥992px)
xl	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on extra large devices (≥1200px)
xxl	`boolean` | `"auto"` | `number` | `{ span: boolean` | `"auto"` | `number, offset: number, order: "first"` | `"last"` | `number }`		The number of columns to span on extra extra large devices (≥1400px)

Stacks

Shorthand helpers that build on top of our flexbox utilities to make component layout faster and easier than ever.

Vertical

Stacks are vertical by default and stacked items are full-width by default. Use the `gap` prop to add space between items.

RESULT

First item

Second item

Third item

LIVE EDITOR

```jsx
import Stack from 'react-bootstrap/Stack';

function VerticalExample() {
  return (
    <Stack gap={3}>
      <div className="p-2">First item</div>
      <div className="p-2">Second item</div>
      <div className="p-2">Third item</div>
    </Stack>
  );
}

export default VerticalExample;
```

Horizontal

Use `direction="horizontal"` for horizontal layouts. Stacked items are vertically centered by default and only take up their necessary width. Use the `gap` prop to add space between items.

RESULT

First item Second item Third item

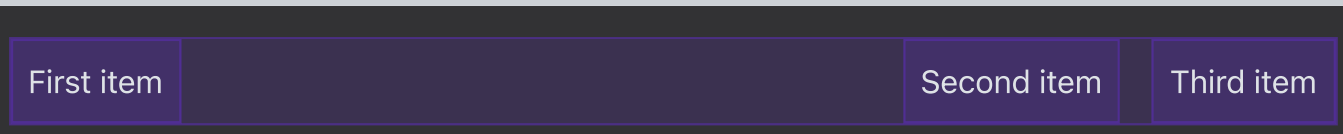
LIVE EDITOR

```jsx
import Stack from 'react-bootstrap/Stack';

function HorizontalExample() {
  return (
    <Stack direction="horizontal" gap={3}>
      <div className="p-2">First item</div>
      <div className="p-2">Second item</div>
      <div className="p-2">Third item</div>
    </Stack>
  );
```

```
}

export default HorizontalExample;
```

Using horizontal margin utilities like `.ms-auto` as spacers:

RESULT

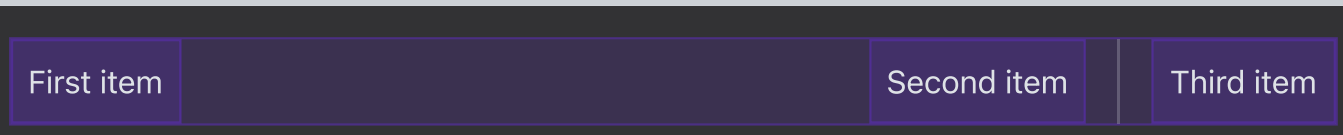


LIVE EDITOR

```
import Stack from 'react-bootstrap/Stack';

function HorizontalMarginStartExample() {
  return (
    <Stack direction="horizontal" gap={3}>
      <div className="p-2">First item</div>
      <div className="p-2 ms-auto">Second item</div>
      <div className="p-2">Third item</div>
    </Stack>
  );
}

export default HorizontalMarginStartExample;
```

And with vertical rules:

RESULT



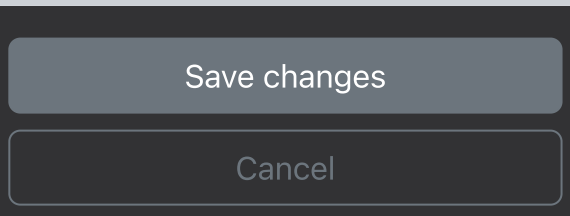
LIVE EDITOR

```
import Stack from 'react-bootstrap/Stack';

function HorizontalVerticalRulesExample() {
  return (
    <Stack direction="horizontal" gap={3}>
      <div className="p-2">First item</div>
      <div className="p-2 ms-auto">Second item</div>
      <div className="vr" />
      <div className="p-2">Third item</div>
    </Stack>
  );
}

export default HorizontalVerticalRulesExample;
```

Examples

Use a vertical `Stack` to stack buttons and other elements:

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Stack from 'react-bootstrap/Stack';
```

```jsx
function ButtonsExample() {
  return (
    <Stack gap={2} className="col-md-5 mx-auto">
      <Button variant="secondary">Save changes</Button>
      <Button variant="outline-secondary">Cancel</Button>
    </Stack>
  );
}

export default ButtonsExample;
```

Create an inline form with a horizontal `Stack`:

RESULT

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import Stack from 'react-bootstrap/Stack';

function FormExample() {
  return (
    <Stack direction="horizontal" gap={3}>
      <Form.Control className="me-auto" placeholder="Add your item
here..." />
      <Button variant="secondary">Submit</Button>
      <div className="vr" />
      <Button variant="outline-danger">Reset</Button>
    </Stack>
  );
}

export default FormExample;
```

API

Stack

```jsx
import Stack from 'react-bootstrap/Stack'
```

Name	Type	Default	Description
bsPrefix	string	`'hstack | vstack'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css. Defaults to `hstack` if direction is `horizontal` or `vstack` if direction is `vertical`.
gap	custom		Sets the spacing between each item. Valid values are `0-5`.
as		`'div'`	You can use a custom element type for this component.

Forms

Examples and usage guidelines for form control styles, layout options, and custom components for creating a wide variety of forms.

Overview

The `<FormControl>` component renders a form control with Bootstrap styling. The `<FormGroup>` component wraps a form control with proper spacing, along with support for a label, help text, and validation state. To ensure accessibility, set `controlId` on `<FormGroup>`, and use `<FormLabel>` for the label.

RESULT

Email address

Enter email

We'll never share your email with anyone else.

Password

Password

Check me out

Submit

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';

function BasicExample() {
  return (
    <Form>
      <Form.Group className="mb-3" controlId="formBasicEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control type="email" placeholder="Enter email" />
        <Form.Text className="text-muted">
          We'll never share your email with anyone else.
        </Form.Text>
      </Form.Group>

      <Form.Group className="mb-3" controlId="formBasicPassword">
        <Form.Label>Password</Form.Label>
        <Form.Control type="password" placeholder="Password" />
      </Form.Group>
      <Form.Group className="mb-3" controlId="formBasicCheckbox">
        <Form.Check type="checkbox" label="Check me out" />
      </Form.Group>
      <Button variant="primary" type="submit">
        Submit
      </Button>
    </Form>
  );
```

```
}

export default BasicExample;
```

The `<FormControl>` component directly renders the `<input>` or other specified component. If you need to access the value of an uncontrolled `<FormControl>`, attach a `ref` to it as you would with an uncontrolled input, then call `ReactDOM.findDOMNode(ref)` to get the DOM node. You can then interact with that node as you would with any other uncontrolled input.

If your application contains a large number of form groups, we recommend building a higher-level component encapsulating a complete field group that renders the label, the control, and any other necessary components. We don't provide this out-of-the-box, because the composition of those field groups is too specific to an individual application to admit a good one-size-fits-all solution.

Disabled forms

Add the `disabled` boolean attribute on an input to prevent user interactions and make it appear lighter.

RESULT

Disabled input

Disabled input

Disabled select menu

Disabled select

Can't check this

LIVE EDITOR

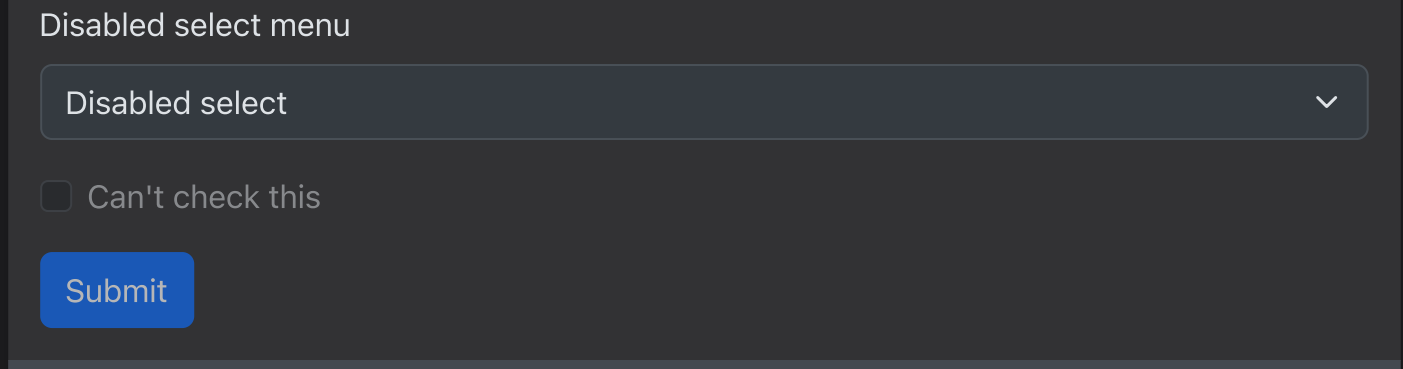
```
import Form from 'react-bootstrap/Form';

function FormDisabledInputExample() {
  return (
    <>
      <Form.Group className="mb-3">
        <Form.Label>Disabled input</Form.Label>
        <Form.Control placeholder="Disabled input" disabled />
      </Form.Group>
      <Form.Group className="mb-3">
        <Form.Label>Disabled select menu</Form.Label>
        <Form.Select disabled>
          <option>Disabled select</option>
        </Form.Select>
      </Form.Group>
      <Form.Group className="mb-3">
        <Form.Check type="checkbox" label="Can't check this" disabled />
      </Form.Group>
    </>
  );
}

export default FormDisabledInputExample;
```

Add the `disabled` attribute to a `<fieldset>` to disable all the controls within.

RESULT

Disabled input

Disabled input



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';

function FormDisabledExample() {
  return (
    <Form>
      <fieldset disabled>
        <Form.Group className="mb-3">
          <Form.Label htmlFor="disabledTextInput">Disabled
input</Form.Label>
          <Form.Control id="disabledTextInput" placeholder="Disabled
input" />
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Label htmlFor="disabledSelect">Disabled select
menu</Form.Label>
          <Form.Select id="disabledSelect">
            <option>Disabled select</option>
          </Form.Select>
        </Form.Group>
        <Form.Group className="mb-3">
          <Form.Check
            type="checkbox"
            id="disabledFieldsetCheck"
            label="Can't check this"
          />
        </Form.Group>
        <Button type="submit">Submit</Button>
      </fieldset>
    </Form>
  );
}

export default FormDisabledExample;
```

Browsers treat all native form controls (`<input>`, `<select>` and `<button>` elements) inside `<fieldset disabled>` as disabled, preventing both keyboard and mouse interactions on them.

However, if your form also includes custom button-like elements such as `<a ... class="btn btn-*">`, these will only be given a style of `pointer-events: none`, meaning they are still focusable and operable using the keyboard. In this case, you must manually modify these controls by adding `tabindex="-1"` to prevent them from receiving focus and `aria-disabled="disabled"` to signal their state to assistive technologies.

API

Form

```
import Form from 'react-bootstrap/Form'
```

Name	Type	Default	Description
_ref	`ReactRef`		The Form `ref` will be forwarded to the underlying element, which means, unless it's rendered `as` a composite component, it will be a DOM node, when resolved.
validated	bool		Mark a form as having been validated. Setting it to `true` will toggle any validation styles on the forms elements.
as	elementType	`'form'`	You can use a custom element type for this component.

FormLabel

```
import FormLabel from 'react-bootstrap/FormLabel'
```

Name	Type	Default	Description
bsPrefix	string	`'form-label'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
htmlFor	string		Uses `controlId` from `<FormGroup>` if not explicitly specified.
column	`bool` | `enum`	`false`	Renders the FormLabel as a `<Col>` component (accepting all the same props), as well as adding additional styling for horizontal forms.
_ref	`ReactRef`		The FormLabel `ref` will be forwarded to the underlying element. Unless the FormLabel is rendered `as` a composite component, it will be a DOM node, when resolved.
visuallyHidden	bool	`false`	Hides the label visually while still allowing it to be read by assistive technologies.
as	elementType	`'label'`	Set a custom element for this component

Form controls

Give textual form controls like <input>s and <textarea>s an upgrade with custom styles, sizing, focus states, and more.

Example

For textual form controls—like `input`s and `textarea`s—use the `FormControl` component. FormControl adds some additional styles for general appearance, focus state, sizing, and more.

RESULT

Email address

name@example.com

Example textarea

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function TextControlsExample() {
  return (
    <Form>
      <Form.Group className="mb-3" controlId="exampleForm.ControlInput1">
        <Form.Label>Email address</Form.Label>
        <Form.Control type="email" placeholder="name@example.com" />
      </Form.Group>
      <Form.Group className="mb-3"
controlId="exampleForm.ControlTextarea1">
        <Form.Label>Example textarea</Form.Label>
        <Form.Control as="textarea" rows={3} />
      </Form.Group>
    </Form>
  );
}

export default TextControlsExample;
```

Sizing

Use `size` on `<FormControl>` to change the size of the input.

RESULT

Large text

Normal text

Small text

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function InputSizesExample() {
  return (
    <>
      <Form.Control size="lg" type="text" placeholder="Large text" />
      <br />
      <Form.Control type="text" placeholder="Normal text" />
      <br />
      <Form.Control size="sm" type="text" placeholder="Small text" />
    </>
  );
}

export default InputSizesExample;
```

Disabled

Add the `disabled` prop on an input to give it a grayed out appearance and remove pointer events.

RESULT

Disabled input

Disabled readonly input

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function FormControlDisabledExample() {
  return (
    <>
      <Form.Control
        type="text"
        placeholder="Disabled input"
        aria-label="Disabled input example"
        disabled
        readOnly
      />
      <br />
      <Form.Control
        type="text"
        placeholder="Disabled readonly input"
        aria-label="Disabled input example"
        readOnly
      />
    </>
  );
}

export default FormControlDisabledExample;
```

Readonly

Add the `readOnly` prop on an input to prevent modification of the input's value. Read-only inputs appear lighter (just like disabled inputs), but retain the standard cursor.

RESULT

Readonly input here...

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function InputReadOnlyExample() {
  return (
    <Form.Control type="text" placeholder="Readonly input here..."
readOnly />
  );
}

export default InputReadOnlyExample;
```

Readonly plain text

If you want to have readonly elements in your form styled as plain text, use the `plaintext` prop on FormControls to remove the default form field styling and preserve the correct margin and padding.

RESULT

Email	email@example.com
Password	Password

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function PlaintextExample() {
  return (
    <Form>
      <Form.Group as={Row} className="mb-3"
controlId="formPlaintextEmail">
        <Form.Label column sm="2">
          Email
        </Form.Label>
        <Col sm="10">
          <Form.Control plaintext readOnly
defaultValue="email@example.com" />
        </Col>
      </Form.Group>

      <Form.Group as={Row} className="mb-3"
controlId="formPlaintextPassword">
        <Form.Label column sm="2">
          Password
        </Form.Label>
        <Col sm="10">
          <Form.Control type="password" placeholder="Password" />
        </Col>
      </Form.Group>
    </Form>
  );
}
```

```
export default PlaintextExample;
```

File input

RESULT

Default file input example

Chọn tệp | Không có tệp nào được chọn

Multiple files input example

Chọn tệp | Không có tệp nào được chọn

Disabled file input example

Chọn tệp | Không có tệp nào được chọn

Small file input example

Chọn tệp | Không có tệp nào được chọn

Large file input example

Chọn tệp | Không có tệp nào được chọn

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function FormFileExample() {
  return (
    <>
      <Form.Group controlId="formFile" className="mb-3">
        <Form.Label>Default file input example</Form.Label>
        <Form.Control type="file" />
      </Form.Group>
      <Form.Group controlId="formFileMultiple" className="mb-3">
        <Form.Label>Multiple files input example</Form.Label>
        <Form.Control type="file" multiple />
      </Form.Group>
      <Form.Group controlId="formFileDisabled" className="mb-3">
        <Form.Label>Disabled file input example</Form.Label>
        <Form.Control type="file" disabled />
      </Form.Group>
      <Form.Group controlId="formFileSm" className="mb-3">
        <Form.Label>Small file input example</Form.Label>
        <Form.Control type="file" size="sm" />
      </Form.Group>
      <Form.Group controlId="formFileLg" className="mb-3">
        <Form.Label>Large file input example</Form.Label>
        <Form.Control type="file" size="lg" />
      </Form.Group>
    </>
  );
}

export default FormFileExample;
```

Color

RESULT

Color picker

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function ColorPickerExample() {
  return (
    <>
      <Form.Label htmlFor="exampleColorInput">Color picker</Form.Label>
      <Form.Control
        type="color"
        id="exampleColorInput"
        defaultValue="#563d7c"
        title="Choose your color"
      />
    </>
  );
}

export default ColorPickerExample;
```

API

FormControl

```jsx
import FormControl from 'react-bootstrap/FormControl'
```

Name	Type	Default	Description
bsPrefix	`string`	`{'form-control'}`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
_ref	`ReactRef`		The FormControl `ref` will be forwarded to the underlying input element, which means unless `as` is a composite component, it will be a DOM node, when resolved.
size	`'sm'` | `'lg'`		Input size variants
htmlSize	`number`		The size attribute of the underlying HTML element. Specifies the visible width in characters if `as` is `'input'`.
as	`'input'` | `'textarea'` | `elementType`	`'input'`	The underlying HTML element to use when rendering the FormControl.
plaintext	`bool`		Render the input as plain text. Generally used along side `readOnly`.
readOnly	`bool`		Make the control readonly
disabled	`bool`		Make the control disabled
value	`string` | `arrayOf` | `number`		*controlled by: `onChange`, initial prop: `defaultValue`* The `value` attribute of underlying input
onChange	`func`		A callback fired when the `value` prop changes

Name	Type	Default	Description
`type`	`string`		The HTML input `type`, which is only relevant if `as` is `'input'` (the default).
`id`	`string`		Uses `controlId` from `<FormGroup>` if not explicitly specified.
`isValid`	`bool`	`false`	Add "valid" validation styles to the control
`isInvalid`	`bool`	`false`	Add "invalid" validation styles to the control and accompanying label

Form text

Create block-level or inline-level form text.

Overview

Block-level help text in forms can be created using `<Form.Text>`. Inline help text can be flexibly implemented using any inline HTML element and utility classes like `.text-muted`.

Associating help text with form controls

Help text should be explicitly associated with the form control it relates to using the `aria-describedby` attribute. This will ensure that assistive technologies—such as screen readers—will announce this help text when the user focuses or enters the control.

Form text below inputs can be styled with `<Form.Text>`. This component includes `display: block` and adds some top margin for easy spacing from the inputs above.

RESULT

Password

Your password must be 8-20 characters long, contain letters and numbers, and must not contain spaces, special characters, or emoji.

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function FormTextExample() {
  return (
    <>
      <Form.Label htmlFor="inputPassword5">Password</Form.Label>
      <Form.Control
        type="password"
        id="inputPassword5"
        aria-describedby="passwordHelpBlock"
      />
      <Form.Text id="passwordHelpBlock" muted>
        Your password must be 8-20 characters long, contain letters and
numbers,
        and must not contain spaces, special characters, or emoji.
      </Form.Text>
    </>
  );
}

export default FormTextExample;
```

API

FormText

```
import FormText from 'react-bootstrap/FormText'
```

Name	Type	Default	Description
bsPrefix	string	`'form-text'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
_ref	`ReactRef`		The FormText `ref` will be forwarded to the underlying element. Unless the FormText is rendered `as` a composite component, it will be a DOM node, when resolved.
muted	bool		A convenience prop for add the `text-muted` class, since it's so commonly used here.
as	elementType	`'small'`	You can use a custom element type for this component.

Select

Customize the native <select> with custom CSS that changes the element's initial appearance.

Default

RESULT

Open this select menu

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function SelectBasicExample() {
  return (
    <Form.Select aria-label="Default select example">
      <option>Open this select menu</option>
      <option value="1">One</option>
      <option value="2">Two</option>
      <option value="3">Three</option>
    </Form.Select>
  );
}

export default SelectBasicExample;
```

Sizing

You may also choose from small and large custom selects to match our similarly sized text inputs.

RESULT

Large select

Default select

Small select

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function SelectSizesExample() {
  return (
    <>
      <Form.Select size="lg">
        <option>Large select</option>
```

```
      </Form.Select>
      <br />
      <Form.Select>
        <option>Default select</option>
      </Form.Select>
      <br />
      <Form.Select size="sm">
        <option>Small select</option>
      </Form.Select>
    </>
  );
}

export default SelectSizesExample;
```

API

FormSelect

```
import FormSelect from 'react-bootstrap/FormSelect'
```

Name	Type	Default	Description
bsPrefix	string	{'form-select'}	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
size	'sm' | 'lg'		Size variants
htmlSize	number		The size attribute of the underlying HTML element. Specifies the number of visible options.
disabled	bool		Make the control disabled
value	string | arrayOf | number		*controlled by:* onChange, *initial prop:* defaultValue The value attribute of underlying input
onChange	func		A callback fired when the value prop changes
isValid	bool	false	Add "valid" validation styles to the control
isInvalid	bool	false	Add "invalid" validation styles to the control and accompanying label

Checks and radios

Create consistent cross-browser and cross-device checkboxes and radios with our completely rewritten checks component.

For the non-textual checkbox and radio controls, `FormCheck` provides a single component for both types that adds some additional styling and improved layout.

Default (stacked)

By default, any number of checkboxes and radios that are immediate sibling will be vertically stacked and appropriately spaced with FormCheck.

RESULT

- [ ] default checkbox
- [ ] disabled checkbox

- ( ) default radio
- ( ) disabled radio

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function CheckExample() {
  return (
    <Form>
      {['checkbox', 'radio'].map((type) => (
        <div key={`default-${type}`} className="mb-3">
          <Form.Check // prettier-ignore
            type={type}
            id={`default-${type}`}
            label={`default ${type}`}
          />

          <Form.Check
            disabled
            type={type}
            label={`disabled ${type}`}
            id={`disabled-default-${type}`}
          />
        </div>
      ))}
    </Form>
  );
}

export default CheckExample;
```

Switches

A switch has the markup of a custom checkbox but uses `type="switch"` to render a toggle switch. Switches also support the same customizable children as `<FormCheck>`.

RESULT

Check this switch

disabled switch

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function SwitchExample() {
  return (
    <Form>
      <Form.Check // prettier-ignore
        type="switch"
        id="custom-switch"
        label="Check this switch"
      />
      <Form.Check // prettier-ignore
        disabled
        type="switch"
        label="disabled switch"
        id="disabled-custom-switch"
      />
    </Form>
  );
}

export default SwitchExample;
```

Tip

You can also use the `<Form.Switch>` alias which encapsulates the above, in a very small component wrapper.

Inline

Group checkboxes or radios on the same horizontal row by adding the `inline` prop.

RESULT

1 2 3 (disabled)

1 2 3 (disabled)

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function CheckInlineExample() {
  return (
    <Form>
      {['checkbox', 'radio'].map((type) => (
        <div key={`inline-${type}`} className="mb-3">
          <Form.Check
            inline
            label="1"
            name="group1"
            type={type}
            id={`inline-${type}-1`}
          />
          <Form.Check
```

```
            inline
            label="2"
            name="group1"
            type={type}
            id={`inline-${type}-2`}
          />
          <Form.Check
            inline
            disabled
            label="3 (disabled)"
            type={type}
            id={`inline-${type}-3`}
          />
        </div>
      ))}
    </Form>
  );
}

export default CheckInlineExample;
```

Reverse

Put your checkboxes, radios, and switches on the opposite side with the `reverse` prop.

RESULT

1
2
3 (disabled)

1
2
3 (disabled)

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function CheckReverseExample() {
  return (
    <Form>
      {['checkbox', 'radio'].map((type) => (
        <div key={`reverse-${type}`} className="mb-3">
          <Form.Check
            reverse
            label="1"
            name="group1"
            type={type}
            id={`reverse-${type}-1`}
          />
          <Form.Check
            reverse
            label="2"
            name="group1"
            type={type}
            id={`reverse-${type}-2`}
          />
          <Form.Check
            reverse
            disabled
            label="3 (disabled)"
            type={type}
            id={`reverse-${type}-3`}
          />
        </div>
      ))}
```

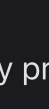
```
    </Form>
  );
}

export default CheckReverseExample;
```

Without labels

When you render a FormCheck without a label (no `children`) some additional styling is applied to keep the inputs from collapsing.

Remember to add an `aria-label` when omitting labels!

RESULT



LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function NoLabelExample() {
  return (
    <>
      <Form.Check aria-label="option 1" />
      <Form.Check type="radio" aria-label="radio 1" />
    </>
  );
}

export default NoLabelExample;
```

Customizing FormCheck rendering

When you need tighter control, or want to customize how the `FormCheck` component renders, it may better to use its constituent parts directly.

By provided `children` to the `FormCheck` you can forgo the default rendering and handle it yourself. (You can still provide an `id` to the `FormCheck` or `FormGroup` and have it propagate to the label and input).

RESULT

Custom api checkbox

You did it!

Custom api radio

You did it!

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function CheckApiExample() {
  return (
    <Form>
      {['checkbox', 'radio'].map((type) => (
        <div key={type} className="mb-3">
          <Form.Check type={type} id={`check-api-${type}`}>
            <Form.Check.Input type={type} isValid />
            <Form.Check.Label>{`Custom api ${type}`}</Form.Check.Label>
            <Form.Control.Feedback type="valid">
              You did it!
```

```
            </Form.Control.Feedback>
          </Form.Check>
        </div>
      ))}
    </Form>
  );
}

export default CheckApiExample;
```

API

FormCheck

```
import FormCheck from 'react-bootstrap/FormCheck'
```

Name	Type	Default	Description
bsPrefix	string	`'form-check'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
bsSwitchPrefix	string	`'form-switch'`	bsPrefix override for the base switch class.
_ref	`ReactRef`		The FormCheck `ref` will be forwarded to the underlying input element, which means it will be a DOM node, when resolved.
as	`'input'` | `elementType`	`'input'`	The underlying HTML element to use when rendering the FormCheck.
id	string		A HTML id attribute, necessary for proper form accessibility. An id is recommended for allowing label clicks to toggle the check control. This is **required** when `type="switch"` due to how they are rendered.
children	node		Provide a function child to manually handle the layout of the FormCheck's inner components. `<FormCheck> <FormCheck.Input isInvalid type={radio} /> <FormCheck.Label>Allow us to contact you? </FormCheck.Label> <Feedback type="invalid">Yo this is required</Feedback> </FormCheck>`
inline	bool	`false`	Groups controls horizontally with other `FormCheck`s.
reverse	bool	`false`	Put your checkboxes, radios, and switches on the opposite side.
disabled	bool	`false`	Disables the control.

Name	Type	Default	Description
title	string	''	title attribute for the underlying FormCheckLabel.
label	node		Label for the control.
type	'radio' | 'checkbox' | 'switch'	'checkbox'	The type of checkable.
isValid	bool	false	Manually style the input as valid
isInvalid	bool	false	Manually style the input as invalid
feedbackTooltip	bool	false	Display feedback as a tooltip.
feedback	node		A message to display when the input is in a validation state
feedbackType	'valid' | 'invalid'		Specify whether the feedback is for valid or invalid fields

FormCheckInput

```
import FormCheckInput from 'react-bootstrap/FormCheckInput'
```

Name	Type	Default	Description
bsPrefix	string	'form-check-input'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	'input' | elementType	'input'	The underlying HTML element to use when rendering the FormCheckInput.
id	string		A HTML id attribute, necessary for proper form accessibility.
type	'radio' | 'checkbox'	'checkbox'	The type of checkable.
isValid	bool	false	Manually style the input as valid
isInvalid	bool	false	Manually style the input as invalid

FormCheckLabel

```
import FormCheckLabel from 'react-bootstrap/FormCheckLabel'
```

Name	Type	Default	Description
bsPrefix	string	'form-check-label'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
htmlFor	string		The HTML for attribute for associating the label with an input

Range

Use our custom range inputs for consistent cross-browser styling and built-in customization.

Overview

Create custom `<input type="range">` controls with `<FormRange>`. The track (the background) and thumb (the value) are both styled to appear the same across browsers. As only Firefox supports "filling" their track from the left or right of the thumb as a means to visually indicate progress, we do not currently support it.

RESULT

Range

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function RangeExample() {
  return (
    <>
      <Form.Label>Range</Form.Label>
      <Form.Range />
    </>
  );
}

export default RangeExample;
```

API

FormRange

```jsx
import FormRange from 'react-bootstrap/FormRange'
```

Name	Type	Default	Description
bsPrefix	string	{'form-range'}	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
disabled	bool		Make the control disabled
value	string | arrayOf | number		*controlled by: onChange, initial prop: defaultValue* The value attribute of underlying input

Name	Type	Default	Description
`onChange`	`func`		A callback fired when the `value` prop changes
`id`	`string`		Uses `controlId` from `<FormGroup>` if not explicitly specified.

Input Group

Easily extend form controls by adding text, buttons, or button groups on either side of textual inputs, custom selects, and custom file inputs.

Example

Place one add-on or button on either side of an input. You may also place one on both sides of an input. Remember to place `<label>`s outside the input group.

RESULT

@ Username

Recipient's username @example.com

Your vanity URL

https://example.com/users/

$ .00

With textarea

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function BasicExample() {
  return (
    <>
      <InputGroup classNam="mb-3">
        <InputGroup.Text id="basic-addon1">@</InputGroup.Text>
        <Form.Control
          placeholder="Username"
          aria-label="Username"
          aria-describedby="basic-addon1"
        />
      </InputGroup>

      <InputGroup classNam="mb-3">
        <Form.Control
          placeholder="Recipient's username"
          aria-label="Recipient's username"
          aria-describedby="basic-addon2"
        />
        <InputGroup.Text id="basic-addon2">@example.com</InputGroup.Text>
      </InputGroup>

      <Form.Label htmlFor="basic-url">Your vanity URL</Form.Label>
      <InputGroup classNam="mb-3">
        <InputGroup.Text id="basic-addon3">
```

```
          https://example.com/users/
        </InputGroup.Text>
        <Form.Control id="basic-url" aria-describedby="basic-addon3" />
      </InputGroup>

      <InputGroup className="mb-3">
        <InputGroup.Text>$</InputGroup.Text>
        <Form.Control aria-label="Amount (to the nearest dollar)" />
        <InputGroup.Text>.00</InputGroup.Text>
      </InputGroup>

      <InputGroup>
        <InputGroup.Text>With textarea</InputGroup.Text>
        <Form.Control as="textarea" aria-label="With textarea" />
      </InputGroup>
    </>
  );
}

export default BasicExample;
```

Sizing

Add the relative form sizing classes to the `InputGroup` and contents within will automatically resize—no need for repeating the form control size classes on each element.

RESULT

Small

Default

Large

LIVE EDITOR

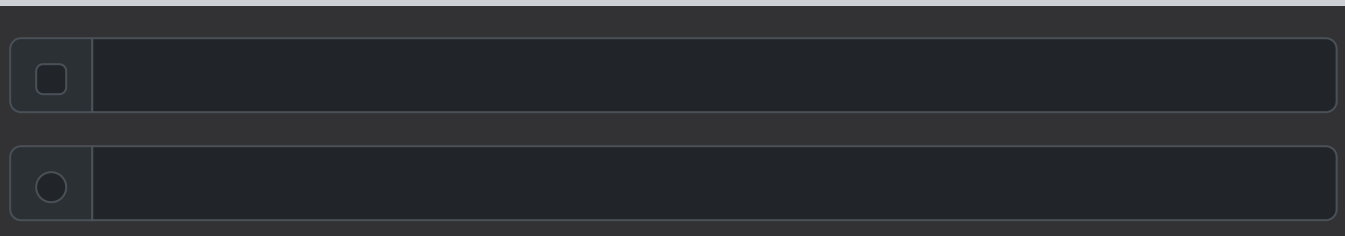
```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function SizesExample() {
  return (
    <>
      <InputGroup size="sm" className="mb-3">
        <InputGroup.Text id="inputGroup-sizing-
sm">Small</InputGroup.Text>
        <Form.Control
          aria-label="Small"
          aria-describedby="inputGroup-sizing-sm"
        />
      </InputGroup>
      <br />
      <InputGroup className="mb-3">
        <InputGroup.Text id="inputGroup-sizing-default">
          Default
        </InputGroup.Text>
        <Form.Control
          aria-label="Default"
          aria-describedby="inputGroup-sizing-default"
        />
      </InputGroup>
      <br />
      <InputGroup size="lg">
        <InputGroup.Text id="inputGroup-sizing-
lg">Large</InputGroup.Text>
        <Form.Control
```

```
          aria-label="Large"
          aria-describedby="inputGroup-sizing-sm"
        />
      </InputGroup>
    </>
  );
}

export default SizesExample;
```

Checkboxes and radios

Use the `InputGroup.Radio` or `InputGroup.Checkbox` to add options to an input group.

RESULT



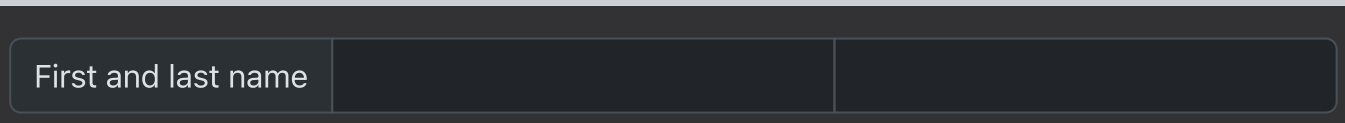
LIVE EDITOR

```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function CheckboxesExample() {
  return (
    <>
      <InputGroup classNam="mb-3">
        <InputGroup.Checkbox aria-label="Checkbox for following text
input" />
        <Form.Control aria-label="Text input with checkbox" />
      </InputGroup>
      <InputGroup>
        <InputGroup.Radio aria-label="Radio button for following text
input" />
        <Form.Control aria-label="Text input with radio button" />
      </InputGroup>
    </>
  );
}

export default CheckboxesExample;
```

Multiple inputs

While multiple inputs are supported visually, validation styles are only available for input groups with a single input.

RESULT



LIVE EDITOR

```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function MultipleInputsExample() {
  return (
    <InputGroup classNam="mb-3">
      <InputGroup.Text>First and last name</InputGroup.Text>
      <Form.Control aria-label="First name" />
```

```
      <Form.Control aria-label="Last name" />
    </InputGroup>
  );
}

export default MultipleInputsExample;
```

Multiple addons

Multiple add-ons are supported and can be mixed with checkbox and radio input versions.

RESULT

$ 0.00

0.00 $

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function MultipleAddonsExample() {
  return (
    <>
      <InputGroup classNam="mb-3">
        <InputGroup.Text>$</InputGroup.Text>
        <InputGroup.Text>0.00</InputGroup.Text>
        <Form.Control aria-label="Dollar amount (with dot and two decimal
places)" />
      </InputGroup>
      <InputGroup>
        <Form.Control aria-label="Dollar amount (with dot and two decimal
places)" />
        <InputGroup.Text>$</InputGroup.Text>
        <InputGroup.Text>0.00</InputGroup.Text>
      </InputGroup>
    </>
  );
}

export default MultipleAddonsExample;
```

Button addons

RESULT

Button

Recipient's username Button

Button Button

Recipient's username Button Button

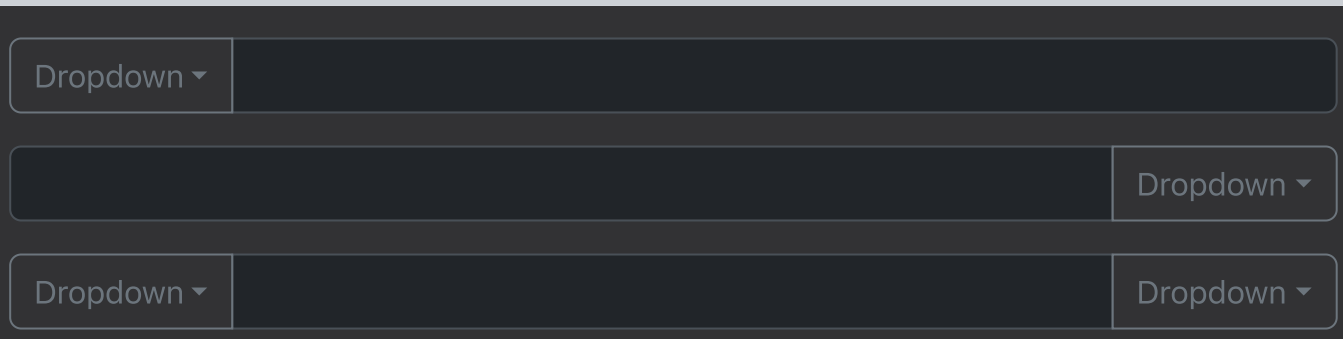
LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function ButtonsExample() {
```

```
  return (
    <>
      <InputGroup className="mb-3">
        <Button variant="outline-secondary" id="button-addon1">
          Button
        </Button>
        <Form.Control
          aria-label="Example text with button addon"
          aria-describedby="basic-addon1"
        />
      </InputGroup>

      <InputGroup className="mb-3">
        <Form.Control
          placeholder="Recipient's username"
          aria-label="Recipient's username"
          aria-describedby="basic-addon2"
        />
        <Button variant="outline-secondary" id="button-addon2">
          Button
        </Button>
      </InputGroup>

      <InputGroup className="mb-3">
        <Button variant="outline-secondary">Button</Button>
        <Button variant="outline-secondary">Button</Button>
        <Form.Control aria-label="Example text with two button addons" />
      </InputGroup>

      <InputGroup>
        <Form.Control
          placeholder="Recipient's username"
          aria-label="Recipient's username with two button addons"
        />
        <Button variant="outline-secondary">Button</Button>
        <Button variant="outline-secondary">Button</Button>
      </InputGroup>
    </>
  );
}

export default ButtonsExample;
```

Buttons with Dropdowns

RESULT



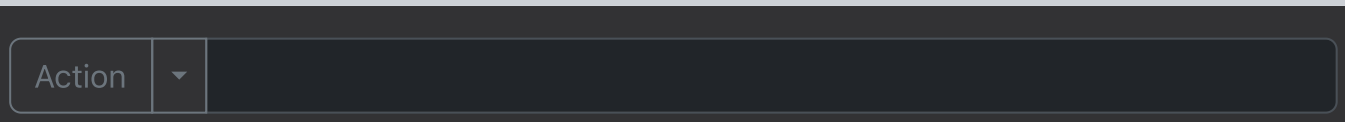
LIVE EDITOR

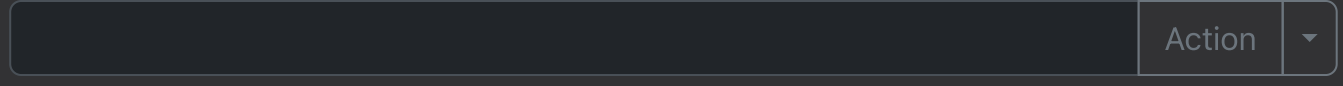
```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function ButtonDropdownsExample() {
  return (
    <>
      <InputGroup className="mb-3">
        <DropdownButton
          variant="outline-secondary"
          title="Dropdown"
```

```
          id="input-group-dropdown-1"
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </DropdownButton>
        <Form.Control aria-label="Text input with dropdown button" />
      </InputGroup>

      <InputGroup className="mb-3">
        <Form.Control aria-label="Text input with dropdown button" />

        <DropdownButton
          variant="outline-secondary"
          title="Dropdown"
          id="input-group-dropdown-2"
          align="end"
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </DropdownButton>
      </InputGroup>

      <InputGroup>
        <DropdownButton
          variant="outline-secondary"
          title="Dropdown"
          id="input-group-dropdown-3"
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </DropdownButton>
        <Form.Control aria-label="Text input with 2 dropdown buttons" />
        <DropdownButton
          variant="outline-secondary"
          title="Dropdown"
          id="input-group-dropdown-4"
          align="end"
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </DropdownButton>
      </InputGroup>
    </>
  );
}

export default ButtonDropdownsExample;
```

Segmented buttons

RESULT





LIVE EDITOR

```jsx
import Dropdown from 'react-bootstrap/Dropdown';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import SplitButton from 'react-bootstrap/SplitButton';

function SegmentedButtonDropdownsExample() {
  return (
    <>
      <InputGroup className="mb-3">
        <SplitButton
          variant="outline-secondary"
          title="Action"
          id="segmented-button-dropdown-1"
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </SplitButton>
        <Form.Control aria-label="Text input with dropdown button" />
      </InputGroup>

      <InputGroup className="mb-3">
        <Form.Control aria-label="Text input with dropdown button" />
        <SplitButton
          variant="outline-secondary"
          title="Action"
          id="segmented-button-dropdown-2"
          alignRight
        >
          <Dropdown.Item href="#">Action</Dropdown.Item>
          <Dropdown.Item href="#">Another action</Dropdown.Item>
          <Dropdown.Item href="#">Something else here</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#">Separated link</Dropdown.Item>
        </SplitButton>
      </InputGroup>
    </>
  );
}

export default SegmentedButtonDropdownsExample;
```

API

InputGroup

```jsx
import InputGroup from 'react-bootstrap/InputGroup'
```

Name	Type	Default	Description
bsPrefix	string	`'input-group'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
size	`'sm'` | `'lg'`		Control the size of buttons and form elements from the top-level.
hasValidation	bool		Handles the input's rounded corners when using form validation.

Name	Type	Default	Description
			Use this when your input group contains both an input and feedback element.
as	elementType	`'div'`	You can use a custom element type for this component.

Floating labels

Create beautifully simple form labels that float over your input fields.

Example

Wrap a `<Form.Control>` element in `<FloatingLabel>` to enable floating labels with Bootstrap's textual form fields. A `placeholder` is required on each `<Form.Control>` as our method of CSS-only floating labels uses the `:placeholder-shown` pseudo-element.

RESULT

Email address

Password

LIVE EDITOR

```jsx
import FloatingLabel from 'react-bootstrap/FloatingLabel';
import Form from 'react-bootstrap/Form';

function FormFloatingBasicExample() {
  return (
    <>
      <FloatingLabel
        controlId="floatingInput"
        label="Email address"
        className="mb-3"
      >
        <Form.Control type="email" placeholder="name@example.com" />
      </FloatingLabel>
      <FloatingLabel controlId="floatingPassword" label="Password">
        <Form.Control type="password" placeholder="Password" />
      </FloatingLabel>
    </>
  );
}

export default FormFloatingBasicExample;
```

Textareas

By default, `<textarea>`s will be the same height as `<input>`s. To set a custom height on your `<textarea>`, do not use the `rows` attribute. Instead, set an explicit `height` (either inline or via custom CSS).

RESULT

Comments

Comments

LIVE EDITOR

```jsx
import FloatingLabel from 'react-bootstrap/FloatingLabel';
import Form from 'react-bootstrap/Form';

function FormFloatingTextareaExample() {
  return (
    <>
      <FloatingLabel
        controlId="floatingTextarea"
        label="Comments"
        className="mb-3"
      >
        <Form.Control as="textarea" placeholder="Leave a comment here" />
      </FloatingLabel>
      <FloatingLabel controlId="floatingTextarea2" label="Comments">
        <Form.Control
          as="textarea"
          placeholder="Leave a comment here"
          style={{ height: '100px' }}
        />
      </FloatingLabel>
    </>
  );
}

export default FormFloatingTextareaExample;
```

Selects

Other than `<Form.Control>`, floating labels are only available on `<Form.Select>`s. They work in the same way, but unlike `<input>`s, they'll always show the `<label>` in its floated state.

RESULT

Works with selects
Open this select menu

LIVE EDITOR

```jsx
import FloatingLabel from 'react-bootstrap/FloatingLabel';
import Form from 'react-bootstrap/Form';

function FormFloatingSelectExample() {
  return (
    <FloatingLabel controlId="floatingSelect" label="Works with selects">
      <Form.Select aria-label="Floating label select example">
        <option>Open this select menu</option>
        <option value="1">One</option>
        <option value="2">Two</option>
        <option value="3">Three</option>
      </Form.Select>
    </FloatingLabel>
  );
}

export default FormFloatingSelectExample;
```

Layout

When working with the Bootstrap grid system, be sure to place form elements within column classes.

RESULT

Email address

Works with selects
Open this select menu

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import FloatingLabel from 'react-bootstrap/FloatingLabel';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function FormFloatingLayoutExample() {
  return (
    <Row className="g-2">
      <Col md>
        <FloatingLabel controlId="floatingInputGrid" label="Email address">
          <Form.Control type="email" placeholder="name@example.com" />
        </FloatingLabel>
      </Col>
      <Col md>
        <FloatingLabel
          controlId="floatingSelectGrid"
          label="Works with selects"
        >
          <Form.Select aria-label="Floating label select example">
            <option>Open this select menu</option>
            <option value="1">One</option>
            <option value="2">Two</option>
            <option value="3">Three</option>
          </Form.Select>
        </FloatingLabel>
      </Col>
    </Row>
  );
}

export default FormFloatingLayoutExample;
```

Customizing rendering

If you need greater control over the rendering, use the `<FormFloating>` component to wrap your input and label. Also note that the `<Form.Control>` must come first so we can utilize a sibling selector (e.g., ~).

RESULT

Email address

Password

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';

function FormFloatingCustom() {
  return (
```

```
    <>
      <Form.Floating className="mb-3">
        <Form.Control
          id="floatingInputCustom"
          type="email"
          placeholder="name@example.com"
        />
        <label htmlFor="floatingInputCustom">Email address</label>
      </Form.Floating>
      <Form.Floating>
        <Form.Control
          id="floatingPasswordCustom"
          type="password"
          placeholder="Password"
        />
        <label htmlFor="floatingPasswordCustom">Password</label>
      </Form.Floating>
    </>
  );
}

export default FormFloatingCustom;
```

API

FormFloating

```
import FormFloating from 'react-bootstrap/FormFloating'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

FloatingLabel

```
import FloatingLabel from 'react-bootstrap/FloatingLabel'
```

Name	Type	Default	Description
as	elementType		You can use a custom element type for this component.
controlId	string		Sets `id` on `<FormControl>` and `htmlFor` on `<label>`.
label Required	node		Form control label.

Layout

Give your forms some structure—from inline to horizontal to custom grid implementations—with our form layout options.

Forms

Every group of form fields should reside in a `<Form>` element. Bootstrap provides no default styling for the `<Form>` element, but there are some powerful browser features that are provided by default.

- New to browser forms? Consider reviewing the MDN form docs for an overview and complete list of available attributes.
- `<button>`s within a `<Form>` default to `type="submit"`, so strive to be specific and always include a `type`.
- You can disable all controls within a form by wrapping them in a `<fieldset>` and setting the `disabled` attribute on it.

Since Bootstrap applies `display: block` and `width: 100%` to almost all our form controls, forms will by default stack vertically. Additional classes can be used to vary this layout on a per-form basis.

Form groups

The `FormGroup` component is the easiest way to add some structure to forms. It provides a flexible container for grouping of labels, controls, optional help text, and form validation messaging. Use it with `fieldset`s, `div`s, or nearly any other element.

You also add the `controlId` prop to accessibly wire the nested label and input together via the `id`.

RESULT

Email address

Enter email

Password

Password

LIVE EDITOR

```jsx
import Form from 'react-bootstrap/Form';

function FormGroupExample() {
  return (
    <Form>
      <Form.Group className="mb-3" controlId="formGroupEmail">
        <Form.Label>Email address</Form.Label>
        <Form.Control type="email" placeholder="Enter email" />
      </Form.Group>
```

```
      <Form.Group className="mb-3" controlId="formGroupPassword">
        <Form.Label>Password</Form.Label>
        <Form.Control type="password" placeholder="Password" />
      </Form.Group>
    </Form>
  );
}

export default FormGroupExample;
```

Form grid

More complex forms can be built using the grid components. Use these for form layouts that require multiple columns, varied widths, and additional alignment options.

RESULT

First name | Last name

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function GridBasicExample() {
  return (
    <Form>
      <Row>
        <Col>
          <Form.Control placeholder="First name" />
        </Col>
        <Col>
          <Form.Control placeholder="Last name" />
        </Col>
      </Row>
    </Form>
  );
}

export default GridBasicExample;
```

More complex layouts can also be created with the grid system.

RESULT

Email | Password

Enter email | Password

Address

1234 Main St

Address 2

Apartment, studio, or floor

City | State | Zip

Choose...

Check me out

Submit

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function GridComplexExample() {
  return (
    <Form>
      <Row className="mb-3">
        <Form.Group as={Col} controlId="formGridEmail">
          <Form.Label>Email</Form.Label>
          <Form.Control type="email" placeholder="Enter email" />
        </Form.Group>

        <Form.Group as={Col} controlId="formGridPassword">
          <Form.Label>Password</Form.Label>
          <Form.Control type="password" placeholder="Password" />
        </Form.Group>
      </Row>

      <Form.Group className="mb-3" controlId="formGridAddress1">
        <Form.Label>Address</Form.Label>
        <Form.Control placeholder="1234 Main St" />
      </Form.Group>

      <Form.Group className="mb-3" controlId="formGridAddress2">
        <Form.Label>Address 2</Form.Label>
        <Form.Control placeholder="Apartment, studio, or floor" />
      </Form.Group>

      <Row className="mb-3">
        <Form.Group as={Col} controlId="formGridCity">
          <Form.Label>City</Form.Label>
          <Form.Control />
        </Form.Group>

        <Form.Group as={Col} controlId="formGridState">
          <Form.Label>State</Form.Label>
          <Form.Select defaultValue="Choose...">
            <option>Choose...</option>
            <option>...</option>
          </Form.Select>
        </Form.Group>

        <Form.Group as={Col} controlId="formGridZip">
          <Form.Label>Zip</Form.Label>
          <Form.Control />
        </Form.Group>
      </Row>

      <Form.Group className="mb-3" id="formGridCheckbox">
        <Form.Check type="checkbox" label="Check me out" />
      </Form.Group>

      <Button variant="primary" type="submit">
        Submit
      </Button>
    </Form>
  );
}

export default GridComplexExample;
```

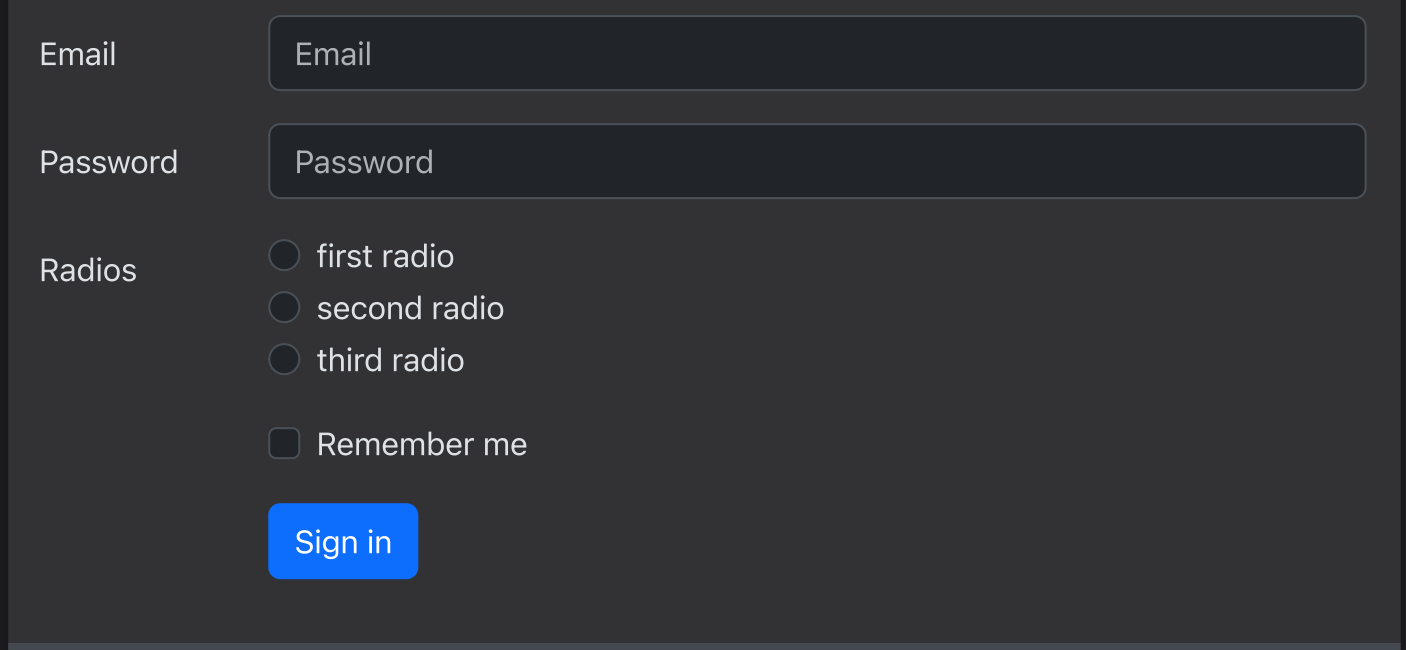
Horizontal form

Create horizontal forms with the grid by adding `as={Row}` to form groups and using `Col` to specify the width of your labels and controls. Be sure to add the `column` prop to your

`FormLabel`s as well so they're vertically centered with their associated form controls.

At times, you maybe need to use margin or padding utilities to create that perfect alignment you need. For example, we've removed the `padding-top` on our stacked radio inputs label to better align the text baseline.

RESULT



LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function HorizontalExample() {
  return (
    <Form>
      <Form.Group as={Row} className="mb-3"
controlId="formHorizontalEmail">
        <Form.Label column sm={2}>
          Email
        </Form.Label>
        <Col sm={10}>
          <Form.Control type="email" placeholder="Email" />
        </Col>
      </Form.Group>

      <Form.Group as={Row} className="mb-3"
controlId="formHorizontalPassword">
        <Form.Label column sm={2}>
          Password
        </Form.Label>
        <Col sm={10}>
          <Form.Control type="password" placeholder="Password" />
        </Col>
      </Form.Group>
      <fieldset>
        <Form.Group as={Row} className="mb-3">
          <Form.Label as="legend" column sm={2}>
            Radios
          </Form.Label>
          <Col sm={10}>
            <Form.Check
              type="radio"
              label="first radio"
              name="formHorizontalRadios"
              id="formHorizontalRadios1"
            />
            <Form.Check
              type="radio"
              label="second radio"
              name="formHorizontalRadios"
              id="formHorizontalRadios2"
            />
            <Form.Check
```

```
            type="radio"
            label="third radio"
            name="formHorizontalRadios"
            id="formHorizontalRadios3"
          />
        </Col>
      </Form.Group>
    </fieldset>
    <Form.Group as={Row} className="mb-3"
controlId="formHorizontalCheck">
      <Col sm={{ span: 10, offset: 2 }}>
        <Form.Check label="Remember me" />
      </Col>
    </Form.Group>

    <Form.Group as={Row} className="mb-3">
      <Col sm={{ span: 10, offset: 2 }}>
        <Button type="submit">Sign in</Button>
      </Col>
    </Form.Group>
  </Form>
 );
}

export default HorizontalExample;
```

Horizontal form label sizing

You can size the `<FormLabel>` using the `column` prop as shown.

RESULT

Large Text

Large text

Normal Text

Normal text

Small Text

Small text

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function FormLabelSizingExample() {
  return (
    <>
      <Row>
        <Form.Label column="lg" lg={2}>
          Large Text
        </Form.Label>
        <Col>
          <Form.Control size="lg" type="text" placeholder="Large text" />
        </Col>
      </Row>
      <br />
      <Row>
        <Form.Label column lg={2}>
          Normal Text
        </Form.Label>
        <Col>
```

```
          <Form.Control type="text" placeholder="Normal text" />
        </Col>
      </Row>
      <br />
      <Row>
        <Form.Label column="sm" lg={2}>
          Small Text
        </Form.Label>
        <Col>
          <Form.Control size="sm" type="text" placeholder="Small text" />
        </Col>
      </Row>
    </>
  );
}

export default FormLabelSizingExample;
```

Column sizing

As shown in the previous examples, our grid system allows you to place any number of `<Col>`s within a `<Row>`. They'll split the available width equally between them. You may also pick a subset of your columns to take up more or less space, while the remaining `<Col>`s equally split the rest, with specific column classes like `<Col xs={7}>`.

RESULT

City | State | Zip

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import Row from 'react-bootstrap/Row';

function GridColSizesExample() {
  return (
    <Form>
      <Row>
        <Col xs={7}>
          <Form.Control placeholder="City" />
        </Col>
        <Col>
          <Form.Control placeholder="State" />
        </Col>
        <Col>
          <Form.Control placeholder="Zip" />
        </Col>
      </Row>
    </Form>
  );
}

export default GridColSizesExample;
```

Auto-sizing

The example below uses a flexbox utility to vertically center the contents and changes `<Col>` to `<Col xs="auto">` so that your columns only take up as much space as needed. Put another way, the column sizes itself based on the contents.

RESULT

Jane Doe | @ | Username | Remember me

Submit

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';

function GridAutoSizingExample() {
  return (
    <Form>
      <Row className="align-items-center">
        <Col xs="auto">
          <Form.Label htmlFor="inlineFormInput" visuallyHidden>
            Name
          </Form.Label>
          <Form.Control
            className="mb-2"
            id="inlineFormInput"
            placeholder="Jane Doe"
          />
        </Col>
        <Col xs="auto">
          <Form.Label htmlFor="inlineFormInputGroup" visuallyHidden>
            Username
          </Form.Label>
          <InputGroup className="mb-2">
            <InputGroup.Text>@</InputGroup.Text>
            <Form.Control id="inlineFormInputGroup"
placeholder="Username" />
          </InputGroup>
        </Col>
        <Col xs="auto">
          <Form.Check
            type="checkbox"
            id="autoSizingCheck"
            className="mb-2"
            label="Remember me"
          />
        </Col>
        <Col xs="auto">
          <Button type="submit" className="mb-2">
            Submit
          </Button>
        </Col>
      </Row>
    </Form>
  );
}

export default GridAutoSizingExample;
```

You can then remix that once again with size-specific column classes.

RESULT

Jane Doe @ Username Remember me Submit

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';
```

```jsx
function GridAutoSizingColMixExample() {
  return (
    <Form>
      <Row className="align-items-center">
        <Col sm={3} className="my-1">
          <Form.Label htmlFor="inlineFormInputName" visuallyHidden>
            Name
          </Form.Label>
          <Form.Control id="inlineFormInputName" placeholder="Jane Doe"
/>
        </Col>
        <Col sm={3} className="my-1">
          <Form.Label htmlFor="inlineFormInputGroupUsername"
visuallyHidden>
            Username
          </Form.Label>
          <InputGroup>
            <InputGroup.Text>@</InputGroup.Text>
            <Form.Control
              id="inlineFormInputGroupUsername"
              placeholder="Username"
            />
          </InputGroup>
        </Col>
        <Col xs="auto" className="my-1">
          <Form.Check
            type="checkbox"
            id="autoSizingCheck2"
            label="Remember me"
          />
        </Col>
        <Col xs="auto" className="my-1">
          <Button type="submit">Submit</Button>
        </Col>
      </Row>
    </Form>
  );
}

export default GridAutoSizingColMixExample;
```

API

FormGroup

```jsx
import FormGroup from 'react-bootstrap/FormGroup'
```

Name	Type	Default	Description
as	elementType	`'div'`	You can use a custom element type for this component.
controlId	string		Sets `id` on `<FormControl>` and `htmlFor` on `<FormGroup.Label>`.
_ref	`ReactRef`		The FormGroup `ref` will be forwarded to the underlying element. Unless the FormGroup is rendered `as` a composite component, it will be a DOM node, when resolved.

Validation

Provide valuable, actionable feedback to your users with HTML5 form validation, via browser default behaviors or custom styles and JavaScript.

Native HTML5 form validation

For native HTML form validation–available in all our supported browsers, the `:valid` and `:invalid` pseudo selectors are used to apply validation styles as well as display feedback messages.

Bootstrap scopes the `:valid` and `:invalid` styles to parent `.was-validated` class, usually applied to the `<Form>` (you can use the `validated` prop as a shortcut). Otherwise, any required field without a value shows up as invalid on page load. This way, you may choose when to activate them (typically after form submission is attempted).

Watch out!

Browsers provide their own validation UI by default on `form`s. You can disable the default UI by adding the HTML `noValidate` attribute to your `<Form>` or `<form>` element.

RESULT

First name: Mark

Last name: Otto

Username: @ Username

City: City

State: State

Zip: Zip

Agree to terms and conditions

Submit form

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';

function FormExample() {
  const [validated, setValidated] = useState(false);

  const handleSubmit = (event) => {
    const form = event.currentTarget;
    if (form.checkValidity() === false) {
      event.preventDefault();
      event.stopPropagation();
    }
```

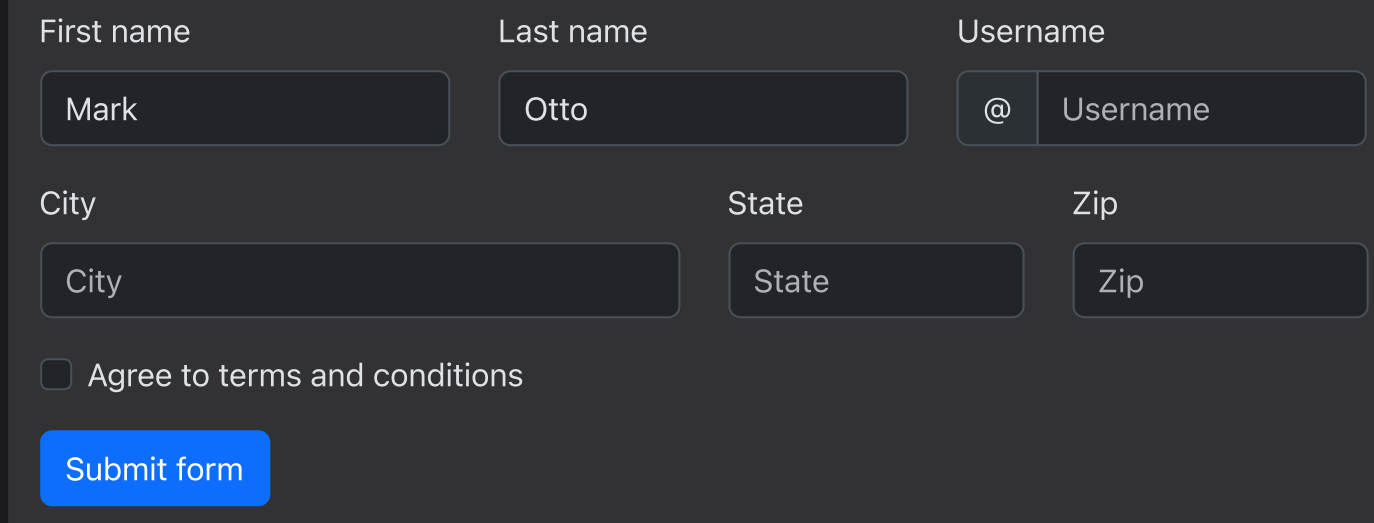
```
  setValidated(true);
};

return (
  <Form noValidate validated={validated} onSubmit={handleSubmit}>
    <Row className="mb-3">
      <Form.Group as={Col} md="4" controlId="validationCustom01">
        <Form.Label>First name</Form.Label>
        <Form.Control
          required
          type="text"
          placeholder="First name"
          defaultValue="Mark"
        />
        <Form.Control.Feedback>Looks good!</Form.Control.Feedback>
      </Form.Group>
      <Form.Group as={Col} md="4" controlId="validationCustom02">
        <Form.Label>Last name</Form.Label>
        <Form.Control
          required
          type="text"
          placeholder="Last name"
          defaultValue="Otto"
        />
        <Form.Control.Feedback>Looks good!</Form.Control.Feedback>
      </Form.Group>
      <Form.Group as={Col} md="4" controlId="validationCustomUsername">
        <Form.Label>Username</Form.Label>
        <InputGroup hasValidation>
          <InputGroup.Text id="inputGroupPrepend">@</InputGroup.Text>
          <Form.Control
            type="text"
            placeholder="Username"
            aria-describedby="inputGroupPrepend"
            required
          />
          <Form.Control.Feedback type="invalid">
            Please choose a username.
          </Form.Control.Feedback>
        </InputGroup>
      </Form.Group>
    </Row>
    <Row className="mb-3">
      <Form.Group as={Col} md="6" controlId="validationCustom03">
        <Form.Label>City</Form.Label>
        <Form.Control type="text" placeholder="City" required />
        <Form.Control.Feedback type="invalid">
          Please provide a valid city.
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group as={Col} md="3" controlId="validationCustom04">
        <Form.Label>State</Form.Label>
        <Form.Control type="text" placeholder="State" required />
        <Form.Control.Feedback type="invalid">
          Please provide a valid state.
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group as={Col} md="3" controlId="validationCustom05">
        <Form.Label>Zip</Form.Label>
        <Form.Control type="text" placeholder="Zip" required />
        <Form.Control.Feedback type="invalid">
          Please provide a valid zip.
        </Form.Control.Feedback>
      </Form.Group>
    </Row>
    <Form.Group className="mb-3">
      <Form.Check
        required
        label="Agree to terms and conditions"
        feedback="You must agree before submitting."
        feedbackType="invalid"
      />
```

```
      </Form.Group>
      <Button type="submit">Submit form</Button>
    </Form>
  );
}

export default FormExample;
```

Form libraries and server-rendered styles

It's often beneficial (especially in React) to handle form validation via a library like Formik, or react-formal. In those cases, `isValid` and `isInvalid` props can be added to form controls to manually apply validation styles. Below is a quick example integrating with Formik.

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';
import * as formik from 'formik';
import * as yup from 'yup';

function FormExample() {
  const { Formik } = formik;

  const schema = yup.object().shape({
    firstName: yup.string().required(),
    lastName: yup.string().required(),
    username: yup.string().required(),
    city: yup.string().required(),
    state: yup.string().required(),
    zip: yup.string().required(),
    terms: yup.bool().required().oneOf([true], 'Terms must be accepted'),
  });

  return (
    <Formik
      validationSchema={schema}
      onSubmit={console.log}
      initialValues={{
        firstName: 'Mark',
        lastName: 'Otto',
        username: '',
        city: '',
        state: '',
        zip: '',
        terms: false,
      }}
    >
      {({ handleSubmit, handleChange, values, touched, errors }) => (
        <Form noValidate onSubmit={handleSubmit}>
          <Row className="mb-3">
```

```
<Form.Group as={Col} md="4" controlId="validationFormik01">
  <Form.Label>First name</Form.Label>
  <Form.Control
    type="text"
    name="firstName"
    value={values.firstName}
    onChange={handleChange}
    isValid={touched.firstName && !errors.firstName}
  />
  <Form.Control.Feedback>Looks good!</Form.Control.Feedback>
</Form.Group>
<Form.Group as={Col} md="4" controlId="validationFormik02">
  <Form.Label>Last name</Form.Label>
  <Form.Control
    type="text"
    name="lastName"
    value={values.lastName}
    onChange={handleChange}
    isValid={touched.lastName && !errors.lastName}
  />

  <Form.Control.Feedback>Looks good!</Form.Control.Feedback>
</Form.Group>
<Form.Group as={Col} md="4"
controlId="validationFormikUsername">
            <Form.Label>Username</Form.Label>
            <InputGroup hasValidation>
              <InputGroup.Text
id="inputGroupPrepend">@</InputGroup.Text>
              <Form.Control
                type="text"
                placeholder="Username"
                aria-describedby="inputGroupPrepend"
                name="username"
                value={values.username}
                onChange={handleChange}
                isInvalid={!!errors.username}
              />
              <Form.Control.Feedback type="invalid">
                {errors.username}
              </Form.Control.Feedback>
            </InputGroup>
          </Form.Group>
        </Row>
        <Row className="mb-3">
          <Form.Group as={Col} md="6" controlId="validationFormik03">
            <Form.Label>City</Form.Label>
            <Form.Control
              type="text"
              placeholder="City"
              name="city"
              value={values.city}
              onChange={handleChange}
              isInvalid={!!errors.city}
            />

            <Form.Control.Feedback type="invalid">
              {errors.city}
            </Form.Control.Feedback>
          </Form.Group>
          <Form.Group as={Col} md="3" controlId="validationFormik04">
            <Form.Label>State</Form.Label>
            <Form.Control
              type="text"
              placeholder="State"
              name="state"
              value={values.state}
              onChange={handleChange}
              isInvalid={!!errors.state}
            />
            <Form.Control.Feedback type="invalid">
              {errors.state}
```

```
            </Form.Control.Feedback>
          </Form.Group>
          <Form.Group as={Col} md="3" controlId="validationFormik05">
            <Form.Label>Zip</Form.Label>
            <Form.Control
              type="text"
              placeholder="Zip"
              name="zip"
              value={values.zip}
              onChange={handleChange}
              isInvalid={!!errors.zip}
            />

            <Form.Control.Feedback type="invalid">
              {errors.zip}
            </Form.Control.Feedback>
          </Form.Group>
        </Row>
        <Form.Group className="mb-3">
          <Form.Check
            required
            name="terms"
            label="Agree to terms and conditions"
            onChange={handleChange}
            isInvalid={!!errors.terms}
            feedback={errors.terms}
            feedbackType="invalid"
            id="validationFormik0"
          />
        </Form.Group>
        <Button type="submit">Submit form</Button>
      </Form>
    )}
  </Formik>
 );
}

export default FormExample;
```

Tooltips

If your form layout allows it, you can use the `tooltip` prop to display validation feedback in a styled tooltip. Be sure to have a parent with `position: relative` on it for tooltip positioning. In the example below, our column classes have this already, but your project may require an alternative setup.

RESULT

First name: Mark

Last name: Otto

Username: @ Username

City: City

State: State

Zip: Zip

File: Chọn tệp | Không có tệp nào được chọn

Agree to terms and conditions

Submit form

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
```

```jsx
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';
import * as formik from 'formik';
import * as yup from 'yup';

function FormExample() {
  const { Formik } = formik;

  const schema = yup.object().shape({
    firstName: yup.string().required(),
    lastName: yup.string().required(),
    username: yup.string().required(),
    city: yup.string().required(),
    state: yup.string().required(),
    zip: yup.string().required(),
    file: yup.mixed().required(),
    terms: yup.bool().required().oneOf([true], 'terms must be accepted'),
  });

  return (
    <Formik
      validationSchema={schema}
      onSubmit={console.log}
      initialValues={{
        firstName: 'Mark',
        lastName: 'Otto',
        username: '',
        city: '',
        state: '',
        zip: '',
        file: null,
        terms: false,
      }}
    >
      {({ handleSubmit, handleChange, values, touched, errors }) => (
        <Form noValidate onSubmit={handleSubmit}>
          <Row className="mb-3">
            <Form.Group
              as={Col}
              md="4"
              controlId="validationFormik101"
              className="position-relative"
            >
              <Form.Label>First name</Form.Label>
              <Form.Control
                type="text"
                name="firstName"
                value={values.firstName}
                onChange={handleChange}
                isValid={touched.firstName && !errors.firstName}
              />
              <Form.Control.Feedback tooltip>Looks good!
</Form.Control.Feedback>
            </Form.Group>
            <Form.Group
              as={Col}
              md="4"
              controlId="validationFormik102"
              className="position-relative"
            >
              <Form.Label>Last name</Form.Label>
              <Form.Control
                type="text"
                name="lastName"
                value={values.lastName}
                onChange={handleChange}
                isValid={touched.lastName && !errors.lastName}
              />

              <Form.Control.Feedback tooltip>Looks good!
</Form.Control.Feedback>
```

```
            </Form.Group>
            <Form.Group as={Col} md="4"
controlId="validationFormikUsername2">
              <Form.Label>Username</Form.Label>
              <InputGroup hasValidation>
                <InputGroup.Text
id="inputGroupPrepend">@</InputGroup.Text>
                <Form.Control
                  type="text"
                  placeholder="Username"
                  aria-describedby="inputGroupPrepend"
                  name="username"
                  value={values.username}
                  onChange={handleChange}
                  isInvalid={!!errors.username}
                />
                <Form.Control.Feedback type="invalid" tooltip>
                  {errors.username}
                </Form.Control.Feedback>
              </InputGroup>
            </Form.Group>
          </Row>
          <Row className="mb-3">
            <Form.Group
              as={Col}
              md="6"
              controlId="validationFormik103"
              className="position-relative"
            >
              <Form.Label>City</Form.Label>
              <Form.Control
                type="text"
                placeholder="City"
                name="city"
                value={values.city}
                onChange={handleChange}
                isInvalid={!!errors.city}
              />

              <Form.Control.Feedback type="invalid" tooltip>
                {errors.city}
              </Form.Control.Feedback>
            </Form.Group>
            <Form.Group
              as={Col}
              md="3"
              controlId="validationFormik104"
              className="position-relative"
            >
              <Form.Label>State</Form.Label>
              <Form.Control
                type="text"
                placeholder="State"
                name="state"
                value={values.state}
                onChange={handleChange}
                isInvalid={!!errors.state}
              />
              <Form.Control.Feedback type="invalid" tooltip>
                {errors.state}
              </Form.Control.Feedback>
            </Form.Group>
            <Form.Group
              as={Col}
              md="3"
              controlId="validationFormik105"
              className="position-relative"
            >
              <Form.Label>Zip</Form.Label>
              <Form.Control
                type="text"
                placeholder="Zip"
```

```
            name="zip"
            value={values.zip}
            onChange={handleChange}
            isInvalid={!!errors.zip}
          />

          <Form.Control.Feedback type="invalid" tooltip>
            {errors.zip}
          </Form.Control.Feedback>
        </Form.Group>
      </Row>
      <Form.Group className="position-relative mb-3">
        <Form.Label>File</Form.Label>
        <Form.Control
          type="file"
          required
          name="file"
          onChange={handleChange}
          isInvalid={!!errors.file}
        />
        <Form.Control.Feedback type="invalid" tooltip>
          {errors.file}
        </Form.Control.Feedback>
      </Form.Group>
      <Form.Group className="position-relative mb-3">
        <Form.Check
          required
          name="terms"
          label="Agree to terms and conditions"
          onChange={handleChange}
          isInvalid={!!errors.terms}
          feedback={errors.terms}
          feedbackType="invalid"
          id="validationFormik106"
          feedbackTooltip
        />
      </Form.Group>
      <Button type="submit">Submit form</Button>
    </Form>
  )}
</Formik>
);
}

export default FormExample;
```

Input group validation

To properly show rounded corners in an `<InputGroup>` with validation, the `<InputGroup>` requires the `hasValidation` prop.

RESULT

@

Please choose a username.

LIVE EDITOR

```
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function ValidationInputGroupExample() {
  return (
    <InputGroup hasValidation>
      <InputGroup.Text>@</InputGroup.Text>
      <Form.Control type="text" required isInvalid />
      <Form.Control.Feedback type="invalid">
        Please choose a username.
```

```
      </Form.Control.Feedback>
    </InputGroup>
  );
}

export default ValidationInputGroupExample;
```

API

Feedback

```
import Feedback from 'react-bootstrap/Feedback'
```

Name	Type	Default	Description
type	`'valid'` | `'invalid'`	`'valid'`	Specify whether the feedback is for valid or invalid fields
tooltip	bool	`false`	Display feedback as a tooltip.
as	elementType	`'div'`	You can use a custom element type for this component.

Accordion

Build vertically collapsing accordions in combination with the Collapse component

Examples

Click the accordions below to expand/collapse the accordion content.

Basic Example

RESULT

Accordion Item #1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Accordion Item #2

LIVE EDITOR

```
import Accordion from 'react-bootstrap/Accordion';

function BasicExample() {
  return (
    <Accordion defaultActiveKey="0">
      <Accordion.Item eventKey="0">
        <Accordion.Header>Accordion Item #1</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
      <Accordion.Item eventKey="1">
        <Accordion.Header>Accordion Item #2</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
```

```
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
    </Accordion>
  );
}

export default BasicExample;
```

Fully Collapsed State

If you want your `Accordion` to start in a fully-collapsed state, then simply don't pass in a `defaultActiveKey` prop to `Accordion`.

RESULT

Accordion Item #1

Accordion Item #2

LIVE EDITOR

```
import Accordion from 'react-bootstrap/Accordion';

function AllCollapseExample() {
  return (
    <Accordion>
      <Accordion.Item eventKey="0">
        <Accordion.Header>Accordion Item #1</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
      <Accordion.Item eventKey="1">
        <Accordion.Header>Accordion Item #2</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
    </Accordion>
  );
}
```

```
export default AllCollapseExample;
```

Flush

Add `flush` to remove the default background-color, some borders, and some rounded corners to render accordions edge-to-edge with their parent container.

RESULT

Accordion Item #1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Accordion Item #2

LIVE EDITOR

```
import Accordion from 'react-bootstrap/Accordion';

function FlushExample() {
  return (
    <Accordion defaultActiveKey="0" flush>
      <Accordion.Item eventKey="0">
        <Accordion.Header>Accordion Item #1</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
      <Accordion.Item eventKey="1">
        <Accordion.Header>Accordion Item #2</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
    </Accordion>
  );
}

export default FlushExample;
```

Always open

You can make accordion items stay open when another item is opened by using the `alwaysOpen` prop. If you're looking to control the component, you must use an array of strings for `activeKey` or `defaultActiveKey`.

RESULT

Accordion Item #1

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Accordion Item #2

LIVE EDITOR

```jsx
import Accordion from 'react-bootstrap/Accordion';

function AlwaysOpenExample() {
  return (
    <Accordion defaultActiveKey={['0']} alwaysOpen>
      <Accordion.Item eventKey="0">
        <Accordion.Header>Accordion Item #1</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
      <Accordion.Item eventKey="1">
        <Accordion.Header>Accordion Item #2</Accordion.Header>
        <Accordion.Body>
          Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
          eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
enim ad
          minim veniam, quis nostrud exercitation ullamco laboris nisi ut
          aliquip ex ea commodo consequat. Duis aute irure dolor in
          reprehenderit in voluptate velit esse cillum dolore eu fugiat
nulla
          pariatur. Excepteur sint occaecat cupidatat non proident, sunt
in
          culpa qui officia deserunt mollit anim id est laborum.
        </Accordion.Body>
      </Accordion.Item>
    </Accordion>
  );
}

export default AlwaysOpenExample;
```

Custom Accordions

You can still create card-based accordions like those in Bootstrap 4. You can hook into the Accordion toggle functionality via `useAccordionButton` to make custom toggle components.

Custom Toggle

RESULT

Click me!

Hello! I'm the body

Click me!

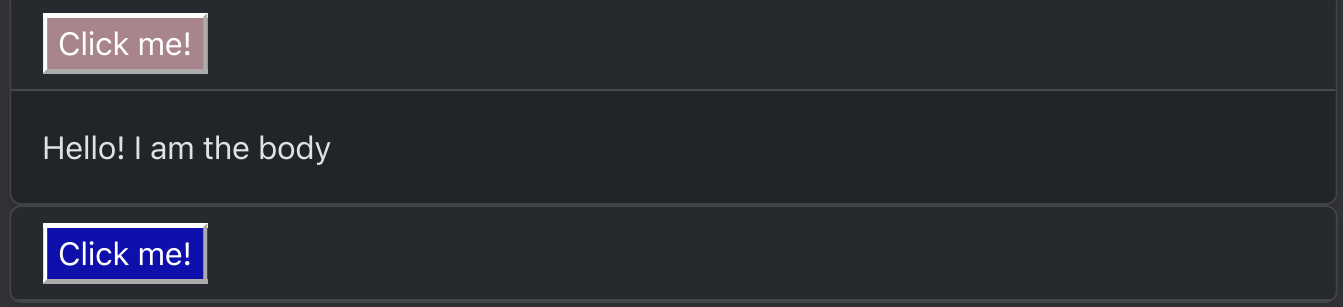
LIVE EDITOR

```jsx
import Accordion from 'react-bootstrap/Accordion';
import { useAccordionButton } from 'react-bootstrap/AccordionButton';
import Card from 'react-bootstrap/Card';

function CustomToggle({ children, eventKey }) {
  const decoratedOnClick = useAccordionButton(eventKey, () =>
    console.log('totally custom!'),
  );

  return (
    <button
      type="button"
      style={{ backgroundColor: 'pink' }}
      onClick={decoratedOnClick}
    >
      {children}
    </button>
  );
}

function Example() {
  return (
    <Accordion defaultActiveKey="0">
      <Card>
        <Card.Header>
          <CustomToggle eventKey="0">Click me!</CustomToggle>
        </Card.Header>
        <Accordion.Collapse eventKey="0">
          <Card.Body>Hello! I'm the body</Card.Body>
        </Accordion.Collapse>
      </Card>
      <Card>
        <Card.Header>
          <CustomToggle eventKey="1">Click me!</CustomToggle>
        </Card.Header>
        <Accordion.Collapse eventKey="1">
          <Card.Body>Hello! I'm another body</Card.Body>
        </Accordion.Collapse>
      </Card>
    </Accordion>
  );
}

render(<Example />);
```

Custom Toggle with Expansion Awareness

You may wish to have different styles for the toggle if it's associated section is expanded, this can be achieved with a custom toggle that is context aware and also takes advantage of the `useAccordionButton` hook.

RESULT



LIVE EDITOR

```jsx
import { useContext } from 'react';
import Accordion from 'react-bootstrap/Accordion';
import AccordionContext from 'react-bootstrap/AccordionContext';
import { useAccordionButton } from 'react-bootstrap/AccordionButton';
import Card from 'react-bootstrap/Card';

const PINK = 'rgba(255, 192, 203, 0.6)';
const BLUE = 'rgba(0, 0, 255, 0.6)';

function ContextAwareToggle({ children, eventKey, callback }) {
  const { activeEventKey } = useContext(AccordionContext);

  const decoratedOnClick = useAccordionButton(
    eventKey,
    () => callback && callback(eventKey),
  );

  const isCurrentEventKey = activeEventKey === eventKey;

  return (
    <button
      type="button"
      style={{ backgroundColor: isCurrentEventKey ? PINK : BLUE }}
      onClick={decoratedOnClick}
    >
      {children}
    </button>
  );
}

function Example() {
  return (
    <Accordion defaultActiveKey="0">
      <Card>
        <Card.Header>
          <ContextAwareToggle eventKey="0">Click me!</ContextAwareToggle>
        </Card.Header>
        <Accordion.Collapse eventKey="0">
          <Card.Body>Hello! I am the body</Card.Body>
        </Accordion.Collapse>
      </Card>
      <Card>
        <Card.Header>
          <ContextAwareToggle eventKey="1">Click me!</ContextAwareToggle>
        </Card.Header>
        <Accordion.Collapse eventKey="1">
          <Card.Body>Hello! I am another body</Card.Body>
        </Accordion.Collapse>
      </Card>
    </Accordion>
  );
}

render(<Example />);
```

API

Accordion

```
import Accordion from 'react-bootstrap/Accordion'
```

Name	Type	Default	Description
as	elementType		Set a custom element for this component
bsPrefix	string	`'accordion'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
activeKey	`string` | `array`		The current active key that corresponds to the currently expanded card
defaultActiveKey	`string` | `array`		The default active key that is expanded on start
onSelect	func		*controls* `activeIndex` Callback fired when the active item changes. `(eventKey: string | string[] | null, event: Object) => void`
flush	bool		Renders accordion edge-to-edge with its parent container
alwaysOpen	bool		Allow accordion items to stay open when another item is opened

AccordionItem

```
import AccordionItem from 'react-bootstrap/AccordionItem'
```

Name	Type	Default	Description
as	elementType	`'div'`	Set a custom element for this component
bsPrefix	string	`'accordion-item'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
eventKey Required	string		A unique key used to control this item's collapse/expand.

AccordionHeader

```
import AccordionHeader from 'react-bootstrap/AccordionHeader'
```

Name	Type	Default	Description
as	elementType	`'h2'`	Set a custom element for this component
bsPrefix	string	`'accordion-header'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.

Name	Type	Default	Description
onClick	func		Click handler for the `AccordionButton` element

AccordionBody

```
import AccordionBody from 'react-bootstrap/AccordionBody'
```

Name	Type	Default	Description
as	elementType	`'div'`	Set a custom element for this component
bsPrefix	string	`'accordion-body'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
onEnter	func		Callback fired before the component expands
onEntering	func		Callback fired after the component starts to expand
onEntered	func		Callback fired after the component has expanded
onExit	func		Callback fired before the component collapses
onExiting	func		Callback fired after the component starts to collapse
onExited	func		Callback fired after the component has collapsed

AccordionButton

```
import AccordionButton from 'react-bootstrap/AccordionButton'
```

Name	Type	Default	Description
as	elementType	`'button'`	Set a custom element for this component
bsPrefix	string	`'accordion-button'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
onClick	func		A callback function for when this component is clicked

AccordionCollapse

```
import AccordionCollapse from 'react-bootstrap/AccordionCollapse'
```

This component accepts all of `Collapse`'s props.

Name	Type	Default	Description
as	elementType	`'div'`	Set a custom element for this component

Name	Type	Default	Description
`eventKey` Required	`string`		A key that corresponds to the toggler that triggers this collapse's expand or collapse.
`children` Required	`element`		Children prop should only contain a single child, and is enforced as such

useAccordionButton

```
import { useAccordionButton } from 'react-bootstrap/AccordionButton';

const decoratedOnClick = useAccordionButton(eventKey, onClick);
```

Alerts

Provide contextual feedback messages for typical user actions with the handful of available and flexible alert messages.

Examples

Alerts are available for any length of text, as well as an optional dismiss button. For proper styling, use one of the eight `variant`s.

RESULT

This is a primary alert—check it out!

This is a secondary alert—check it out!

This is a success alert—check it out!

This is a danger alert—check it out!

This is a warning alert—check it out!

This is a info alert—check it out!

This is a light alert—check it out!

This is a dark alert—check it out!

LIVE EDITOR

```jsx
import Alert from 'react-bootstrap/Alert';

function BasicExample() {
  return (
    <>
      {[
        'primary',
        'secondary',
        'success',
        'danger',
        'warning',
        'info',
        'light',
        'dark',
```

```
      ].map((variant) => (
        <Alert key={variant} variant={variant}>
          This is a {variant} alert—check it out!
        </Alert>
      ))}
    </>
  );
}

export default BasicExample;
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

Links

For links, use the `<Alert.Link>` component to provide matching colored links within any alert.

RESULT

This is a primary alert with **an example link**. Give it a click if you like.

This is a secondary alert with **an example link**. Give it a click if you like.

This is a success alert with **an example link**. Give it a click if you like.

This is a danger alert with **an example link**. Give it a click if you like.

This is a warning alert with **an example link**. Give it a click if you like.

This is a info alert with **an example link**. Give it a click if you like.

This is a light alert with **an example link**. Give it a click if you like.

This is a dark alert with **an example link**. Give it a click if you like.

LIVE EDITOR

```
import Alert from 'react-bootstrap/Alert';

function LinksExample() {
  return (
    <>
      {[
        'primary',
        'secondary',
        'success',
        'danger',
        'warning',
        'info',
```

```
        'light',
        'dark',
      ].map((variant) => (
        <Alert key={variant} variant={variant}>
          This is a {variant} alert with{' '}
          <Alert.Link href="#">an example link</Alert.Link>. Give it a
click if
          you like.
        </Alert>
      ))}
    </>
  );
}

export default LinksExample;
```

Additional content

Alerts can contain whatever content you like. Headers, paragraphs, dividers, go crazy.

RESULT

Hey, nice to see you

Aww yeah, you successfully read this important alert message. This example text is going to run a bit longer so that you can see how spacing within an alert works with this kind of content.

Whenever you need to, be sure to use margin utilities to keep things nice and tidy.

LIVE EDITOR

```
import Alert from 'react-bootstrap/Alert';

function AdditionalContentExample() {
  return (
    <Alert variant="success">
      <Alert.Heading>Hey, nice to see you</Alert.Heading>
      <p>
        Aww yeah, you successfully read this important alert message.
This
        example text is going to run a bit longer so that you can see how
        spacing within an alert works with this kind of content.
      </p>
      <hr />
      <p className="mb-0">
        Whenever you need to, be sure to use margin utilities to keep
things
        nice and tidy.
      </p>
    </Alert>
  );
}

export default AdditionalContentExample;
```

Dismissing

Add the `dismissible` prop to add a functioning dismiss button to the Alert.

RESULT

Oh snap! You got an error!

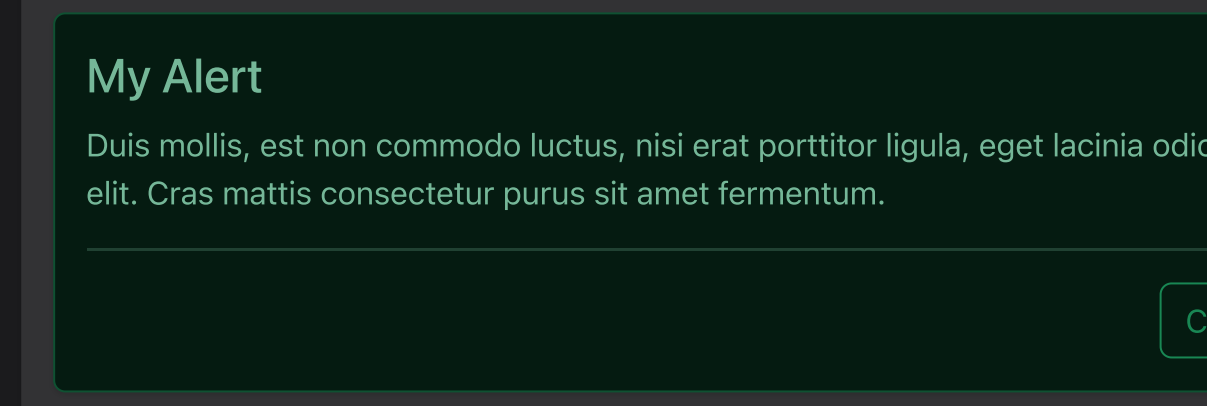
Change this and that and try again. Duis mollis, est non commodo luctus, nisi erat porttitor ligula, eget lacinia odio sem nec elit. Cras mattis consectetur purus sit amet fermentum.

LIVE EDITOR

```jsx
import { useState } from 'react';
import Alert from 'react-bootstrap/Alert';
import Button from 'react-bootstrap/Button';

function AlertDismissibleExample() {
  const [show, setShow] = useState(true);

  if (show) {
    return (
      <Alert variant="danger" onClose={() => setShow(false)} dismissible>
        <Alert.Heading>Oh snap! You got an error!</Alert.Heading>
        <p>
          Change this and that and try again. Duis mollis, est non
commodo
          luctus, nisi erat porttitor ligula, eget lacinia odio sem nec
elit.
          Cras mattis consectetur purus sit amet fermentum.
        </p>
      </Alert>
    );
  }
  return <Button onClick={() => setShow(true)}>Show Alert</Button>;
}

export default AlertDismissibleExample;
```

You can also control the visual state directly which is great if you want to build more complicated alerts.

RESULT



LIVE EDITOR

```jsx
import { useState } from 'react';
import Alert from 'react-bootstrap/Alert';
import Button from 'react-bootstrap/Button';

function AlertDismissible() {
  const [show, setShow] = useState(true);

  return (
    <>
      <Alert show={show} variant="success">
        <Alert.Heading>My Alert</Alert.Heading>
        <p>
          Duis mollis, est non commodo luctus, nisi erat porttitor
ligula, eget
          lacinia odio sem nec elit. Cras mattis consectetur purus sit
amet
          fermentum.
        </p>
        <hr />
```

```
        <div className="d-flex justify-content-end">
          <Button onClick={() => setShow(false)} variant="outline-
success">
            Close me
          </Button>
        </div>
      </Alert>

      {!show && <Button onClick={() => setShow(true)}>Show
Alert</Button>}
    </>
  );
}

export default AlertDismissible;
```

API

Alert

```
import Alert from 'react-bootstrap/Alert'
```

Name	Type	Default	Description
bsPrefix	string	`'alert'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'dark'` | `'light'`		The Alert visual variant
dismissible	bool		Renders a properly aligned

show	bool		*controlled by:* `onClose`, *initial prop:* `defaultShow` Controls the visual state of the Alert.
onClose	func		*controls* `show` Callback fired when alert is closed.
closeLabel	string		Sets the text for alert close button.
closeVariant	`'white'`		Sets the variant for close button.
transition	`bool` | `elementType`		Animate the alert dismissal. Defaults to using `<Fade>` animation or use `false` to disable. A custom `react-transition-group` Transition can also be provided.

AlertHeading

```
import AlertHeading from 'react-bootstrap/AlertHeading'
```

Name	Type	Default	Description
as		`divWithClassName('h4')`	You can use a custom element type for this component.

AlertLink

```
import AlertLink from 'react-bootstrap/AlertLink'
```

Name	Type	Default	Description
as		`Anchor`	You can use a custom element type for this component.

Badges

Documentation and examples for badges, our small count and labeling component.

Examples

Badges scale to match the size of the immediate parent element by using relative font sizing and em units.

RESULT

Example heading New

Example heading New

Example heading New

Example heading New

Example heading New

Example heading New

LIVE EDITOR

```jsx
import Badge from 'react-bootstrap/Badge';

function BasicExample() {
  return (
    <div>
      <h1>
        Example heading <Badge bg="secondary">New</Badge>
      </h1>
      <h2>
        Example heading <Badge bg="secondary">New</Badge>
      </h2>
      <h3>
        Example heading <Badge bg="secondary">New</Badge>
      </h3>
      <h4>
        Example heading <Badge bg="secondary">New</Badge>
      </h4>
      <h5>
        Example heading <Badge bg="secondary">New</Badge>
      </h5>
      <h6>
        Example heading <Badge bg="secondary">New</Badge>
      </h6>
    </div>
  );
}

export default BasicExample;
```

Badges can be used as part of links or buttons to provide a counter.

RESULT

Profile 9

LIVE EDITOR

```
import Badge from 'react-bootstrap/Badge';
import Button from 'react-bootstrap/Button';

function ButtonExample() {
  return (
    <Button variant="primary">
      Profile <Badge bg="secondary">9</Badge>
      <span className="visually-hidden">unread messages</span>
    </Button>
  );
}

export default ButtonExample;
```

Note that depending on how they are used, badges may be confusing for users of screen readers and similar assistive technologies. While the styling of badges provides a visual cue as to their purpose, these users will simply be presented with the content of the badge. Depending on the specific situation, these badges may seem like random additional words or numbers at the end of a sentence, link, or button. Unless the context is clear, consider including additional context with a visually hidden piece of additional text.

Contextual variations

Add any of the below mentioned modifier classes to change the appearance of a badge.

RESULT

Primary Secondary Success Danger Warning Info Light Dark

LIVE EDITOR

```
import Badge from 'react-bootstrap/Badge';
import Stack from 'react-bootstrap/Stack';

function VariationsExample() {
  return (
    <Stack direction="horizontal" gap={2}>
      <Badge bg="primary">Primary</Badge>
      <Badge bg="secondary">Secondary</Badge>
      <Badge bg="success">Success</Badge>
      <Badge bg="danger">Danger</Badge>
      <Badge bg="warning" text="dark">
        Warning
      </Badge>
      <Badge bg="info">Info</Badge>
      <Badge bg="light" text="dark">
        Light
      </Badge>
      <Badge bg="dark">Dark</Badge>
    </Stack>
  );
}

export default VariationsExample;
```

Pill badges

Use the `pill` modifier class to make badges more rounded (with a larger `border-radius`). Useful if you miss the badges from v3.

RESULT

Primary Secondary Success Danger Warning Info Light Dark

LIVE EDITOR

```jsx
import Badge from 'react-bootstrap/Badge';
import Stack from 'react-bootstrap/Stack';

function PillExample() {
  return (
    <Stack direction="horizontal" gap={2}>
      <Badge pill bg="primary">
        Primary
      </Badge>
      <Badge pill bg="secondary">
        Secondary
      </Badge>
      <Badge pill bg="success">
        Success
      </Badge>
      <Badge pill bg="danger">
        Danger
      </Badge>
      <Badge pill bg="warning" text="dark">
        Warning
      </Badge>
      <Badge pill bg="info">
        Info
      </Badge>
      <Badge pill bg="light" text="dark">
        Light
      </Badge>
      <Badge pill bg="dark">
        Dark
      </Badge>
    </Stack>
  );
}

export default PillExample;
```

API

Badge

```jsx
import Badge from 'react-bootstrap/Badge'
```

Name	Type	Default	Description
bsPrefix	string	`'badge'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
bg	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'light'` | `'dark'`	`'primary'`	The visual style of the badge
pill	bool	`false`	Add the `pill` modifier to make badges more rounded with some

Name	Type	Default	Description
			additional horizontal padding
text	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'light'` | `'dark'`		Sets badge text color
as	elementType	`span`	You can use a custom element type for this component.

Breadcrumbs

Indicate the current page's location within a navigational hierarchy that automatically adds separators via CSS.

Example

Add `active` prop to the active `Breadcrumb.Item`. Do not set both `active` and `href` attributes. `active` overrides `href` and `span` element is rendered instead of `a`.

RESULT

Home / Library / Data

LIVE EDITOR

```
import Breadcrumb from 'react-bootstrap/Breadcrumb';

function BreadcrumbExample() {
  return (
    <Breadcrumb>
      <Breadcrumb.Item href="#">Home</Breadcrumb.Item>
      <Breadcrumb.Item
href="https://getbootstrap.com/docs/4.0/components/breadcrumb/">
        Library
      </Breadcrumb.Item>
      <Breadcrumb.Item active>Data</Breadcrumb.Item>
    </Breadcrumb>
  );
}

export default BreadcrumbExample;
```

API

Breadcrumb

```
import Breadcrumb from 'react-bootstrap/Breadcrumb'
```

Name	Type	Default	Description
bsPrefix	string	`'breadcrumb'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
label	string	`'breadcrumb'`	ARIA label for the nav element https://www.w3.org/TR/wai-aria-practices/#breadcrumb

Name	Type	Default	Description
`listProps`	`object`	`{}`	Additional props passed as-is to the underlying `<ol>` element
`as`	`elementType`	`'nav'`	You can use a custom element type for this component.

BreadcrumbItem

```
import BreadcrumbItem from 'react-bootstrap/BreadcrumbItem'
```

Name	Type	Default	Description
`bsPrefix`	`string`	`'breadcrumb-item'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
`active`	`bool`	`false`	Adds a visual "active" state to a Breadcrumb Item and disables the link.
`href`	`string`		`href` attribute for the inner `a` element
`linkAs`	`elementType`	`Anchor`	You can use a custom element type for this component's inner link.
`title`	`node`		`title` attribute for the inner `a` element
`target`	`string`		`target` attribute for the inner `a` element
`linkProps`	`object`	`{}`	Additional props passed as-is to the underlying link for non-active items.
`as`	`elementType`	`'li'`	You can use a custom element type for this component.

Button group

Group a series of buttons together on a single line or stack them in a vertical column.

Basic example

Wrap a series of `<Button>`s in a `<ButtonGroup>`.

RESULT

Left Middle Right

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';

function BasicExample() {
  return (
    <ButtonGroup aria-label="Basic example">
      <Button variant="secondary">Left</Button>
      <Button variant="secondary">Middle</Button>
      <Button variant="secondary">Right</Button>
    </ButtonGroup>
  );
}

export default BasicExample;
```

Button toolbar

Combine sets of `<ButtonGroup>`s into a `<ButtonToolbar>` for more complex components.

RESULT

1 2 3 4 5 6 7 8

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import ButtonToolbar from 'react-bootstrap/ButtonToolbar';

function ToolbarBasicExample() {
  return (
    <ButtonToolbar aria-label="Toolbar with button groups">
      <ButtonGroup className="me-2" aria-label="First group">
        <Button>1</Button> <Button>2</Button> <Button>3</Button>{' '}
        <Button>4</Button>
      </ButtonGroup>
      <ButtonGroup className="me-2" aria-label="Second group">
```

```
        <Button>5</Button> <Button>6</Button> <Button>7</Button>
      </ButtonGroup>
      <ButtonGroup aria-label="Third group">
        <Button>8</Button>
      </ButtonGroup>
    </ButtonToolbar>
  );
}

export default ToolbarBasicExample;
```

Feel free to mix input groups with button groups in your toolbars. Similar to the example above, you'll likely need some utilities though to space things properly.

RESULT

1 2 3 4 @ Input group example

1 2 3 4 @ Input group example

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import ButtonToolbar from 'react-bootstrap/ButtonToolbar';
import Form from 'react-bootstrap/Form';
import InputGroup from 'react-bootstrap/InputGroup';

function ToolbarExample() {
  return (
    <>
      <ButtonToolbar className="mb-3" aria-label="Toolbar with Button
groups">
        <ButtonGroup className="me-2" aria-label="First group">
          <Button variant="secondary">1</Button>{' '}
          <Button variant="secondary">2</Button>{' '}
          <Button variant="secondary">3</Button>{' '}
          <Button variant="secondary">4</Button>
        </ButtonGroup>
        <InputGroup>
          <InputGroup.Text id="btnGroupAddon">@</InputGroup.Text>
          <Form.Control
            type="text"
            placeholder="Input group example"
            aria-label="Input group example"
            aria-describedby="btnGroupAddon"
          />
        </InputGroup>
      </ButtonToolbar>

      <ButtonToolbar
        className="justify-content-between"
        aria-label="Toolbar with Button groups"
      >
        <ButtonGroup aria-label="First group">
          <Button variant="secondary">1</Button>{' '}
          <Button variant="secondary">2</Button>{' '}
          <Button variant="secondary">3</Button>{' '}
          <Button variant="secondary">4</Button>
        </ButtonGroup>
        <InputGroup>
          <InputGroup.Text id="btnGroupAddon2">@</InputGroup.Text>
          <Form.Control
            type="text"
            placeholder="Input group example"
            aria-label="Input group example"
            aria-describedby="btnGroupAddon2"
          />
        </InputGroup>
```

```
      </ButtonToolbar>
    </>
  );
}

export default ToolbarExample;
```

Sizing

Instead of applying button sizing props to every button in a group, just add `size` prop to the `<ButtonGroup>`.

RESULT

Left Middle Right

Left Middle Right

Left Middle Right

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';

function SizesExample() {
  return (
    <>
      <ButtonGroup size="lg" className="mb-2">
        <Button>Left</Button>
        <Button>Middle</Button>
        <Button>Right</Button>
      </ButtonGroup>
      <br />
      <ButtonGroup className="mb-2">
        <Button>Left</Button>
        <Button>Middle</Button>
        <Button>Right</Button>
      </ButtonGroup>
      <br />
      <ButtonGroup size="sm">
        <Button>Left</Button>
        <Button>Middle</Button>
        <Button>Right</Button>
      </ButtonGroup>
    </>
  );
}

export default SizesExample;
```

Nesting

You can place other button types within the `<ButtonGroup>` like `<DropdownButton>`s.

RESULT

1 2 Dropdown

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import DropdownButton from 'react-bootstrap/DropdownButton';
import Dropdown from 'react-bootstrap/Dropdown';
```

```
function NestedExample() {
  return (
    <ButtonGroup>
      <Button>1</Button>
      <Button>2</Button>

      <DropdownButton as={ButtonGroup} title="Dropdown" id="bg-nested-
dropdown">
        <Dropdown.Item eventKey="1">Dropdown link</Dropdown.Item>
        <Dropdown.Item eventKey="2">Dropdown link</Dropdown.Item>
      </DropdownButton>
    </ButtonGroup>
  );
}

export default NestedExample;
```

Vertical variation

Make a set of buttons appear vertically stacked rather than horizontally, by adding `vertical` to the `<ButtonGroup>`. **Split button dropdowns are not supported here.**

RESULT

Button
Button
Dropdown
Button
Button
Dropdown
Dropdown

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function VerticalExample() {
  return (
    <ButtonGroup vertical>
      <Button>Button</Button>
      <Button>Button</Button>

      <DropdownButton
        as={ButtonGroup}
        title="Dropdown"
        id="bg-vertical-dropdown-1"
      >
        <Dropdown.Item eventKey="1">Dropdown link</Dropdown.Item>
        <Dropdown.Item eventKey="2">Dropdown link</Dropdown.Item>
      </DropdownButton>

      <Button>Button</Button>
      <Button>Button</Button>

      <DropdownButton
        as={ButtonGroup}
        title="Dropdown"
        id="bg-vertical-dropdown-2"
      >
        <Dropdown.Item eventKey="1">Dropdown link</Dropdown.Item>
```

```
        <Dropdown.Item eventKey="2">Dropdown link</Dropdown.Item>
      </DropdownButton>

      <DropdownButton
        as={ButtonGroup}
        title="Dropdown"
        id="bg-vertical-dropdown-3"
      >
        <Dropdown.Item eventKey="1">Dropdown link</Dropdown.Item>
        <Dropdown.Item eventKey="2">Dropdown link</Dropdown.Item>
      </DropdownButton>
    </ButtonGroup>
  );
}

export default VerticalExample;
```

API

ButtonGroup

```
import ButtonGroup from 'react-bootstrap/ButtonGroup'
```

Name	Type	Default	Description
bsPrefix	string	`'btn-group'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
size	`'sm'` | `'lg'`		Sets the size for all Buttons in the group.
vertical	bool	`false`	Make the set of Buttons appear vertically stacked.
role	string	`'group'`	An ARIA role describing the button group. Usually the default "group" role is fine. An `aria-label` or `aria-labelledby` prop is also recommended.
as	elementType	`'div'`	You can use a custom element type for this component.

ButtonToolbar

```
import ButtonToolbar from 'react-bootstrap/ButtonToolbar'
```

Name	Type	Default	Description
bsPrefix	string	`'btn-toolbar'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
role	string	`'toolbar'`	The ARIA role describing the button toolbar. Generally the default "toolbar" role is correct. An `aria-label` or `aria-labelledby` prop is also recommended.

Buttons

Use Bootstrap's custom button styles for actions in forms, dialogs, and more with support for multiple sizes, states, and more.

Examples

Use any of the available button style types to quickly create a styled button. Just modify the `variant` prop.

RESULT

Primary Secondary Success Warning Danger Info Light Dark

Link

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';

function TypesExample() {
  return (
    <>
      <Button variant="primary">Primary</Button>{' '}
      <Button variant="secondary">Secondary</Button>{' '}
      <Button variant="success">Success</Button>{' '}
      <Button variant="warning">Warning</Button>{' '}
      <Button variant="danger">Danger</Button>{' '}
      <Button variant="info">Info</Button>{' '}
      <Button variant="light">Light</Button>{' '}
      <Button variant="dark">Dark</Button>
      <Button variant="link">Link</Button>
    </>
  );
}

export default TypesExample;
```

Outline buttons

For a lighter touch, Buttons also come in `outline-*` variants with no background color.

RESULT

Primary Secondary Success Warning Danger Info Light Dark

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';

function OutlineTypesExample() {
  return (
```

```
    <>
      <Button variant="outline-primary">Primary</Button>{' '}
      <Button variant="outline-secondary">Secondary</Button>{' '}
      <Button variant="outline-success">Success</Button>{' '}
      <Button variant="outline-warning">Warning</Button>{' '}
      <Button variant="outline-danger">Danger</Button>{' '}
      <Button variant="outline-info">Info</Button>{' '}
      <Button variant="outline-light">Light</Button>{' '}
      <Button variant="outline-dark">Dark</Button>
    </>
  );
}

export default OutlineTypesExample;
```

Button tags

Normally `<Button>` components will render a HTML `<button>` element. However you can render whatever you'd like, adding a `href` prop will automatically render an `<a />` element. You can use the `as` prop to render whatever your heart desires. React Bootstrap will take care of the proper ARIA roles for you.

RESULT

Link Button Input Submit Reset

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';

function TagTypesExample() {
  return (
    <>
      <Button href="#">Link</Button> <Button
type="submit">Button</Button>{' '}
      <Button as="input" type="button" value="Input" />{' '}
      <Button as="input" type="submit" value="Submit" />{' '}
      <Button as="input" type="reset" value="Reset" />
    </>
  );
}

export default TagTypesExample;
```

Sizes

Fancy larger or smaller buttons? Add `size="lg"`, `size="sm"` for additional sizes.

RESULT

Large button Large button

Small button Small button

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';

function SizesExample() {
  return (
    <>
      <div className="mb-2">
        <Button variant="primary" size="lg">
          Large button
```

```
        </Button>{' '}
        <Button variant="secondary" size="lg">
          Large button
        </Button>
      </div>
      <div>
        <Button variant="primary" size="sm">
          Small button
        </Button>{' '}
        <Button variant="secondary" size="sm">
          Small button
        </Button>
      </div>
    </>
  );
}

export default SizesExample;
```

Block buttons

Create responsive stacks of full-width, “block buttons” like those in Bootstrap 4 with a mix of our display and gap utilities.

RESULT

Block level button

Block level button

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';

function BlockExample() {
  return (
    <div className="d-grid gap-2">
      <Button variant="primary" size="lg">
        Block level button
      </Button>
      <Button variant="secondary" size="lg">
        Block level button
      </Button>
    </div>
  );
}

export default BlockExample;
```

Active state

To set a button's active state simply set the component's `active` prop.

RESULT

Primary button Button

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';

function ActiveExample() {
  return (
```

```
    <>
      <Button variant="primary" size="lg" active>
        Primary button
      </Button>{' '}
      <Button variant="secondary" size="lg" active>
        Button
      </Button>
    </>
  );
}

export default ActiveExample;
```

Disabled state

Make buttons look inactive by adding the `disabled` prop to.

RESULT

Primary button Button Link

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';

function DisabledExample() {
  return (
    <>
      <Button variant="primary" size="lg" disabled>
        Primary button
      </Button>{' '}
      <Button variant="secondary" size="lg" disabled>
        Button
      </Button>{' '}
      <Button href="#" variant="secondary" size="lg" disabled>
        Link
      </Button>
    </>
  );
}

export default DisabledExample;
```

Watch out! `<a>` elements don't naturally support a `disabled` attribute. In browsers that support it this is handled with a `point-events: none` style but not all browsers support it yet.

React Bootstrap will prevent any `onClick` handlers from firing regardless of the rendered element.

Button loading state

When activating an asynchronous action from a button it is a good UX pattern to give the user feedback as to the loading state, this can easily be done by updating your `<Button />`s props from a state change like below.

RESULT

Click to load

LIVE EDITOR

```
import { useEffect, useState } from 'react';
import Button from 'react-bootstrap/Button';
```

```jsx
function LoadingButton() {
  const [isLoading, setLoading] = useState(false);

  useEffect(() => {
    function simulateNetworkRequest() {
      return new Promise((resolve) => setTimeout(resolve, 2000));
    }

    if (isLoading) {
      simulateNetworkRequest().then(() => {
        setLoading(false);
      });
    }
  }, [isLoading]);

  const handleClick = () => setLoading(true);

  return (
    <Button
      variant="primary"
      disabled={isLoading}
      onClick={!isLoading ? handleClick : null}
    >
      {isLoading ? 'Loading…' : 'Click to load'}
    </Button>
  );
}

export default LoadingButton;
```

Checkbox / Radio

Buttons can also be used to style `checkbox` and `radio` form elements. This is helpful when you want a toggle button that works neatly inside an HTML form.

RESULT

Checked

Active Radio Radio

Checked

Active Radio Radio

LIVE EDITOR

```jsx
import { useState } from 'react';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import ToggleButton from 'react-bootstrap/ToggleButton';

function ToggleButtonExample() {
  const [checked, setChecked] = useState(false);
  const [radioValue, setRadioValue] = useState('1');

  const radios = [
    { name: 'Active', value: '1' },
    { name: 'Radio', value: '2' },
    { name: 'Radio', value: '3' },
  ];

  return (
    <>
      <ButtonGroup className="mb-2">
        <ToggleButton
          id="toggle-check"
          type="checkbox"
```

```
        variant="secondary"
        checked={checked}
        value="1"
        onChange={(e) => setChecked(e.currentTarget.checked)}
      >
        Checked
      </ToggleButton>
    </ButtonGroup>
    <br />
    <ButtonGroup className="mb-2">
      {radios.map((radio, idx) => (
        <ToggleButton
          key={idx}
          id={`radio-${idx}`}
          type="radio"
          variant="secondary"
          name="radio"
          value={radio.value}
          checked={radioValue === radio.value}
          onChange={(e) => setRadioValue(e.currentTarget.value)}
        >
          {radio.name}
        </ToggleButton>
      ))}
    </ButtonGroup>
    <br />
    <ToggleButton
      className="mb-2"
      id="toggle-check"
      type="checkbox"
      variant="outline-primary"
      checked={checked}
      value="1"
      onChange={(e) => setChecked(e.currentTarget.checked)}
    >
      Checked
    </ToggleButton>
    <br />
    <ButtonGroup>
      {radios.map((radio, idx) => (
        <ToggleButton
          key={idx}
          id={`radio-${idx}`}
          type="radio"
          variant={idx % 2 ? 'outline-success' : 'outline-danger'}
          name="radio"
          value={radio.value}
          checked={radioValue === radio.value}
          onChange={(e) => setRadioValue(e.currentTarget.value)}
        >
          {radio.name}
        </ToggleButton>
      ))}
    </ButtonGroup>
  </>
  );
}

export default ToggleButtonExample;
```

The above handles styling, But requires manually controlling the `checked` state for each radio or checkbox in the group.

For a nicer experience with checked state management use the `<ToggleButtonGroup>` instead of a `<ButtonGroup>` component. The group behaves as a form component, where the `value` is an array of the selected `value`s for a named checkbox group or the single toggled `value` in a similarly named radio group.

Uncontrolled

RESULT

Checkbox 1 (pre-checked) Checkbox 2 Checkbox 3 (pre-checked)

Radio 1 (pre-checked) Radio 2 Radio 3

LIVE EDITOR

```
import ToggleButton from 'react-bootstrap/ToggleButton';
import ToggleButtonGroup from 'react-bootstrap/ToggleButtonGroup';

function ToggleButtonGroupUncontrolled() {
  return (
    <>
      <ToggleButtonGroup type="checkbox" defaultValue={[1, 3]}
className="mb-2">
        <ToggleButton id="tbg-check-1" value={1}>
          Checkbox 1 (pre-checked)
        </ToggleButton>
        <ToggleButton id="tbg-check-2" value={2}>
          Checkbox 2
        </ToggleButton>
        <ToggleButton id="tbg-check-3" value={3}>
          Checkbox 3 (pre-checked)
        </ToggleButton>
      </ToggleButtonGroup>
      <br />
      <ToggleButtonGroup type="radio" name="options" defaultValue={1}>
        <ToggleButton id="tbg-radio-1" value={1}>
          Radio 1 (pre-checked)
        </ToggleButton>
        <ToggleButton id="tbg-radio-2" value={2}>
          Radio 2
        </ToggleButton>
        <ToggleButton id="tbg-radio-3" value={3}>
          Radio 3
        </ToggleButton>
      </ToggleButtonGroup>
    </>
  );
}

export default ToggleButtonGroupUncontrolled;
```

Controlled

RESULT

Option 1 Option 2 Option 3

LIVE EDITOR

```
import { useState } from 'react';
import ToggleButton from 'react-bootstrap/ToggleButton';
import ToggleButtonGroup from 'react-bootstrap/ToggleButtonGroup';

function ToggleButtonGroupControlled() {
  const [value, setValue] = useState([1, 3]);

  /*
   * The second argument that will be passed to
   * `handleChange` from `ToggleButtonGroup`
   * is the SyntheticEvent object, but we are
   * not using it in this example so we will omit it.
   */
  const handleChange = (val) => setValue(val);
```

```
  return (
    <ToggleButtonGroup type="checkbox" value={value} onChange=
{handleChange}>
      <ToggleButton id="tbg-btn-1" value={1}>
        Option 1
      </ToggleButton>
      <ToggleButton id="tbg-btn-2" value={2}>
        Option 2
      </ToggleButton>
      <ToggleButton id="tbg-btn-3" value={3}>
        Option 3
      </ToggleButton>
    </ToggleButtonGroup>
  );
}

export default ToggleButtonGroupControlled;
```

API

Button

```
import Button from 'react-bootstrap/Button'
```

Name	Type	Default	Description
bsPrefix	string	`'btn'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.

variant	string	`'primary'`	One or more button variant combinations
			
			buttons may be one of a variety of visual variants such as:
			
			`'primary', 'secondary', 'success', 'danger', 'warning', 'info', 'dark', 'light', 'link'`
			
			as well as "outline" versions (prefixed by 'outline-*')
			
			`'outline-primary', 'outline-secondary', 'outline-success', 'outline-danger', 'outline-warning', 'outline-info', 'outline-dark', 'outline-light'`

onClick	func		Callback fired when the button is clicked.
size	`'sm'` | `'lg'`		Specifies a large or small button.
active	bool	`false`	Manually set the visual state of the button to `:active`
disabled	bool	`false`	Disables the Button, preventing mouse events, even if the underlying component is an `<a>` element
href	string		Providing a `href` will render an `<a>` element, *styled* as a button.
type	`'button'` | `'reset'` |	`'button'`	Defines HTML button type attribute.

Name	Type	Default	Description
	`'submit'` | `null`		
as	elementType		You can use a custom element type for this component.

ToggleButtonGroup

```
import ToggleButtonGroup from 'react-bootstrap/ToggleButtonGroup'
```

Name	Type	Default	Description
name	string		An HTML `<input>` name for each child button. **Required if `type` is set to `'radio'`**
value	any		*controlled by: `onChange`, initial prop: `defaultValue`* The value, or array of values, of the active (pressed) buttons
onChange	func		*controls `value`* Callback fired when a button is pressed, depending on whether the `type` is `'radio'` or `'checkbox'`, `onChange` will be called with the value or array of active values
type Required	`'checkbox'` | `'radio'`		The input `type` of the rendered buttons, determines the toggle behavior of the buttons
size	`'sm'` | `'lg'`		Sets the size for all Buttons in the group.
vertical	bool		Make the set of Buttons appear vertically stacked.

ToggleButton

```
import ToggleButton from 'react-bootstrap/ToggleButton'
```

Name	Type	Default	Description
bsPrefix	string	`'btn-check'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
type	`'checkbox'` | `'radio'`		The `<input>` element `type`
name	string		The HTML input name, used to group like checkboxes or radio buttons together semantically
checked	bool		The checked state of the input, managed by `<ToggleButtonGroup>` automatically
disabled	bool		The disabled state of both the label and input
id Required	string		`id` is required for button clicks to toggle input.
onChange	func		A callback fired when the underlying input element changes. This is passed directly to the

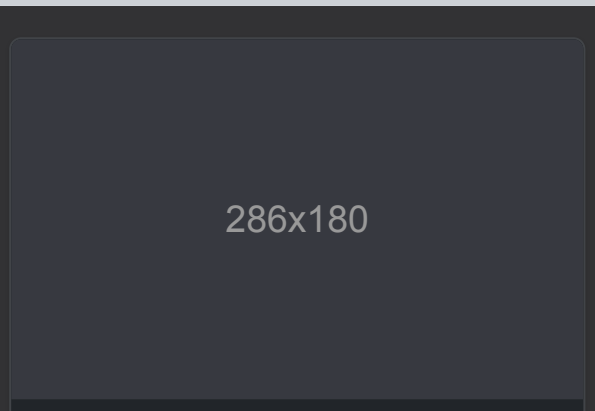
Name	Type	Default	Description
			`<input>` so shares the same signature as a native `onChange` event.
`value` Required	`string` | `arrayOf` | `number`		The value of the input, should be unique amongst its siblings when nested in a `ToggleButtonGroup`.
`inputRef`	`ReactRef`		A ref attached to the `<input>` element

Cards

Bootstrap's cards provide a flexible and extensible content container with multiple variants and options.

Basic Example

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';

function BasicExample() {
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Img variant="top" src="holder.js/100px180" />
      <Card.Body>
        <Card.Title>Card Title</Card.Title>
        <Card.Text>
          Some quick example text to build on the card title and make up
the
          bulk of the card's content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default BasicExample;
```

Content types

Body

Use `<Card.Body>` to pad content inside a `<Card>`.

RESULT

This is some text within a card body.

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';

function BodyOnlyExample() {
  return (
    <Card>
      <Card.Body>This is some text within a card body.</Card.Body>
    </Card>
  );
}

export default BodyOnlyExample;
```

Alternatively, you can use this shorthand version for Cards with body only, and no other children

RESULT

This is some text within a card body.

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';

function BodyShorthandExample() {
  return <Card body>This is some text within a card body.</Card>;
}

export default BodyShorthandExample;
```

Title, text, and links

Using `<Card.Title>`, `<Card.Subtitle>`, and `<Card.Text>` inside the `<Card.Body>` will line them up nicely. `<Card.Link>`s are used to line up links next to each other.

`<Card.Text>` outputs `<p>` tags around the content, so you can use multiple `<Card.Text>`s to create separate paragraphs.

RESULT

Card Title

Card Subtitle

Some quick example text to build on the card title and make up the bulk of the card's content.

Card Link Another Link

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';

function TextExample() {
```

```
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Body>
        <Card.Title>Card Title</Card.Title>
        <Card.Subtitle className="mb-2 text-muted">Card
Subtitle</Card.Subtitle>
        <Card.Text>
          Some quick example text to build on the card title and make up
the
          bulk of the card's content.
        </Card.Text>
        <Card.Link href="#">Card Link</Card.Link>
        <Card.Link href="#">Another Link</Card.Link>
      </Card.Body>
    </Card>
  );
}

export default TextExample;
```

List Groups

Create lists of content in a card with a flush list group.

RESULT

- Cras justo odio
- Dapibus ac facilisis in
- Vestibulum at eros

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';
import ListGroup from 'react-bootstrap/ListGroup';

function ListGroupExample() {
  return (
    <Card style={{ width: '18rem' }}>
      <ListGroup variant="flush">
        <ListGroup.Item>Cras justo odio</ListGroup.Item>
        <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
        <ListGroup.Item>Vestibulum at eros</ListGroup.Item>
      </ListGroup>
    </Card>
  );
}

export default ListGroupExample;
```

RESULT

- Featured
- Cras justo odio
- Dapibus ac facilisis in
- Vestibulum at eros

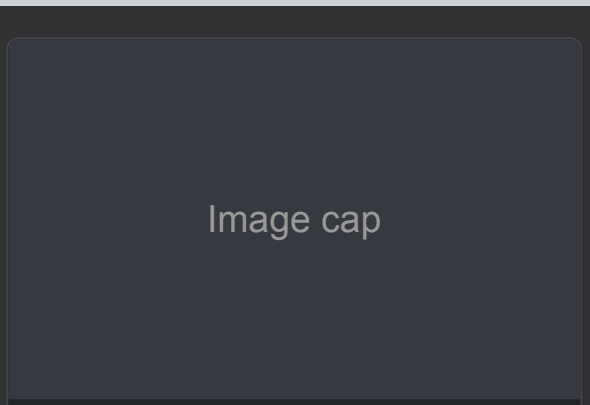
LIVE EDITOR

```
import Card from 'react-bootstrap/Card';
import ListGroup from 'react-bootstrap/ListGroup';

function ListGroupWithHeaderExample() {
```

```
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Header>Featured</Card.Header>
      <ListGroup variant="flush">
        <ListGroup.Item>Cras justo odio</ListGroup.Item>
        <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
        <ListGroup.Item>Vestibulum at eros</ListGroup.Item>
      </ListGroup>
    </Card>
  );
}

export default ListGroupWithHeaderExample;
```

Kitchen Sink

RESULT



Card Title

Some quick example text to build on the card title and make up the bulk of the card's content.

Cras justo odio

Dapibus ac facilisis in

Vestibulum at eros

Card Link Another Link

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';
import ListGroup from 'react-bootstrap/ListGroup';

function KitchenSinkExample() {
  return (
    <Card style={{ width: '18rem' }}>
      <Card.Img variant="top" src="holder.js/100px180?text=Image cap" />
      <Card.Body>
        <Card.Title>Card Title</Card.Title>
        <Card.Text>
          Some quick example text to build on the card title and make up
the
          bulk of the card's content.
        </Card.Text>
      </Card.Body>
      <ListGroup className="list-group-flush">
        <ListGroup.Item>Cras justo odio</ListGroup.Item>
        <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
        <ListGroup.Item>Vestibulum at eros</ListGroup.Item>
      </ListGroup>
      <Card.Body>
        <Card.Link href="#">Card Link</Card.Link>
        <Card.Link href="#">Another Link</Card.Link>
      </Card.Body>
    </Card>
  );
```

```
}

export default KitchenSinkExample;
```

Header and Footer

You may add a header by adding a `<Card.Header>` component.

RESULT

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

Go somewhere

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';

function WithHeaderExample() {
  return (
    <Card>
      <Card.Header>Featured</Card.Header>
      <Card.Body>
        <Card.Title>Special title treatment</Card.Title>
        <Card.Text>
          With supporting text below as a natural lead-in to additional
content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default WithHeaderExample;
```

A `<CardHeader>` can be styled by passing a heading element through the `<as>` prop

RESULT

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

Go somewhere

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';

function WithHeaderStyledExample() {
  return (
    <Card>
      <Card.Header as="h5">Featured</Card.Header>
      <Card.Body>
        <Card.Title>Special title treatment</Card.Title>
        <Card.Text>
```

```
          With supporting text below as a natural lead-in to additional
content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default WithHeaderStyledExample;
```

RESULT

Quote

> Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer posuere erat a ante.
> — Someone famous in *Source Title*

LIVE EDITOR

```
import Card from 'react-bootstrap/Card';

function WithHeaderAndQuoteExample() {
  return (
    <Card>
      <Card.Header>Quote</Card.Header>
      <Card.Body>
        <blockquote className="blockquote mb-0">
          <p>
            {' '}
            Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Integer
            posuere erat a ante.{' '}
          </p>
          <footer className="blockquote-footer">
            Someone famous in <cite title="Source Title">Source
Title</cite>
          </footer>
        </blockquote>
      </Card.Body>
    </Card>
  );
}

export default WithHeaderAndQuoteExample;
```

RESULT

Featured

Special title treatment

With supporting text below as a natural lead-in to additional content.

Go somewhere

2 days ago

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';

function HeaderAndFooterExample() {
```

```
  return (
    <Card className="text-center">
      <Card.Header>Featured</Card.Header>
      <Card.Body>
        <Card.Title>Special title treatment</Card.Title>
        <Card.Text>
          With supporting text below as a natural lead-in to additional
content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
      <Card.Footer className="text-muted">2 days ago</Card.Footer>
    </Card>
  );
}

export default HeaderAndFooterExample;
```

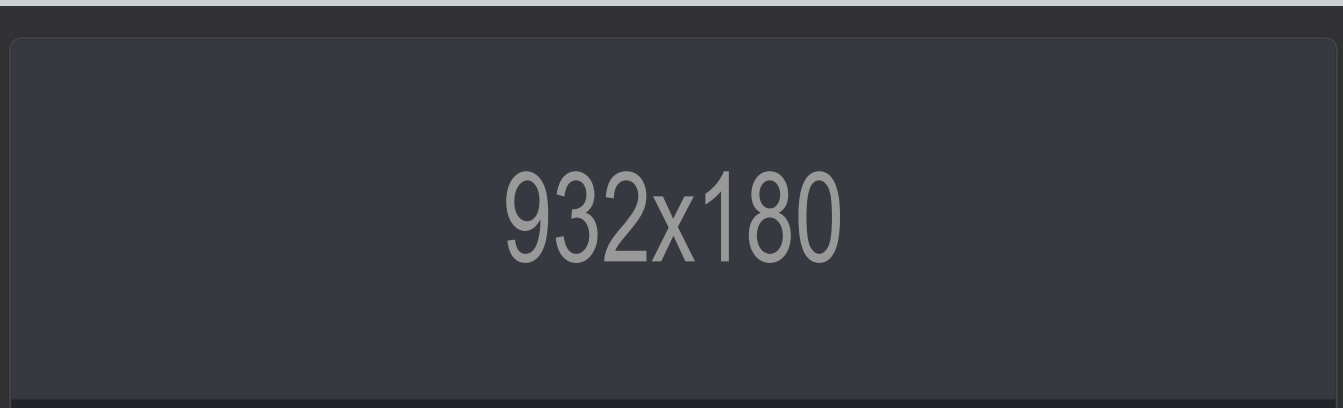
Images

Cards include a few options for working with images. Choose from appending “image caps” at either end of a card, overlaying images with card content, or simply embedding the image in a card.

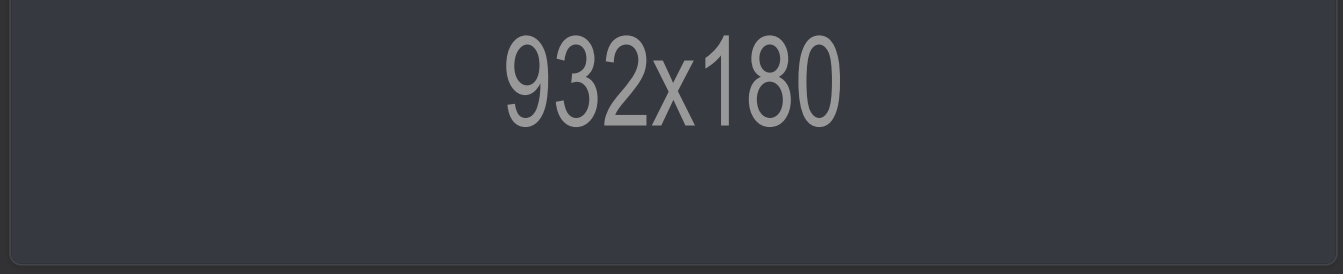
Image caps

Similar to headers and footers, cards can include top and bottom “image caps”—images at the top or bottom of a card.

RESULT



Some quick example text to build on the card title and make up the bulk of the card's content.



LIVE EDITOR

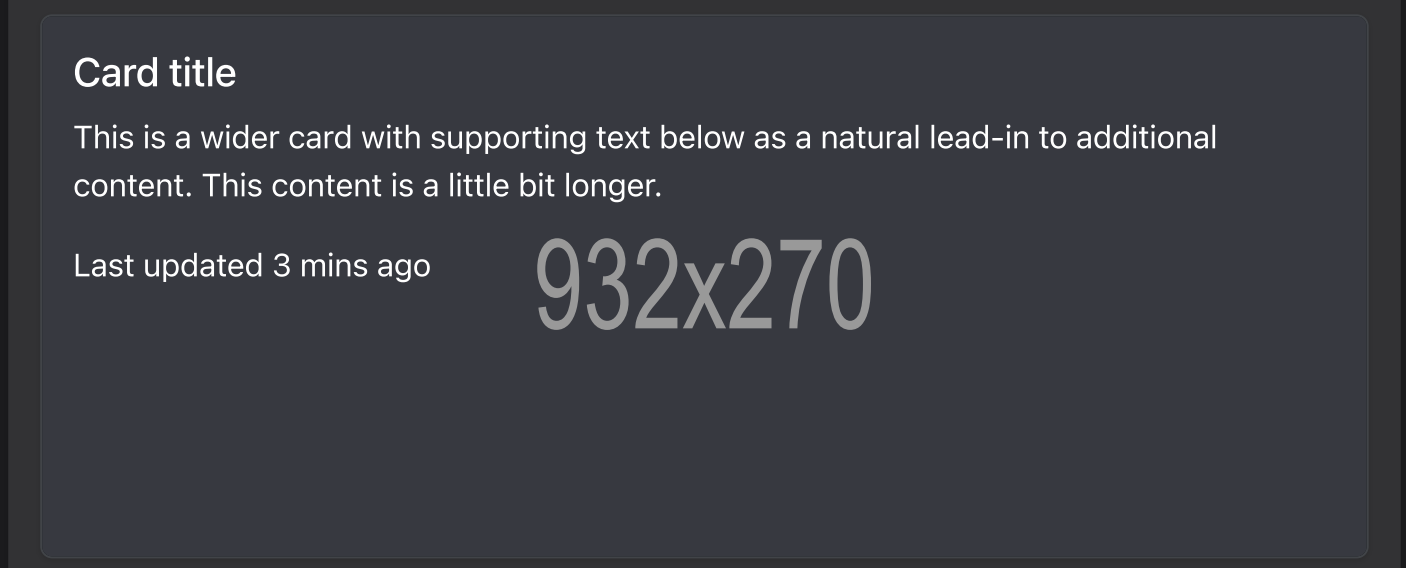
```
import Card from 'react-bootstrap/Card';

function ImageAndTextExample() {
  return (
    <>
      <Card>
        <Card.Img variant="top" src="holder.js/100px180" />
        <Card.Body>
          <Card.Text>
```

```
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />
      <Card>
        <Card.Body>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
        <Card.Img variant="bottom" src="holder.js/100px180" />
      </Card>
    </>
  );
}

export default ImageAndTextExample;
```

Image Overlays

Turn an image into a card background and overlay your card's text. Depending on the image, you may or may not need additional styles or utilities.

RESULT



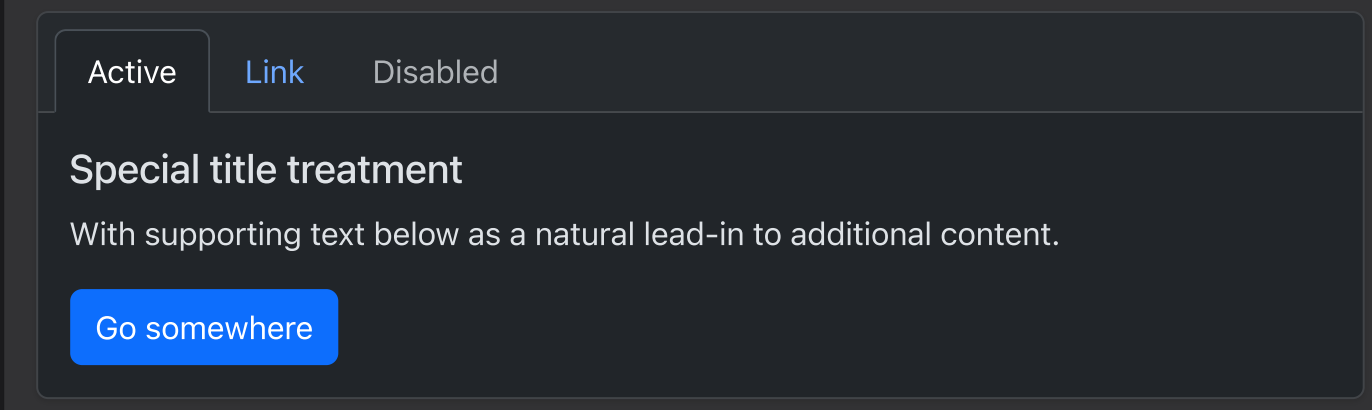
LIVE EDITOR

```
import Card from 'react-bootstrap/Card';

function ImgOverlayExample() {
  return (
    <Card classNam="bg-dark text-white">
      <Card.Img src="holder.js/100px270" alt="Card image" />
      <Card.ImgOverlay>
        <Card.Title>Card title</Card.Title>
        <Card.Text>
          This is a wider card with supporting text below as a natural
lead-in
          to additional content. This content is a little bit longer.
        </Card.Text>
        <Card.Text>Last updated 3 mins ago</Card.Text>
      </Card.ImgOverlay>
    </Card>
  );
}

export default ImgOverlayExample;
```

Navigation

Add some navigation to a card's header (or block) with React Bootstrap's Nav components.

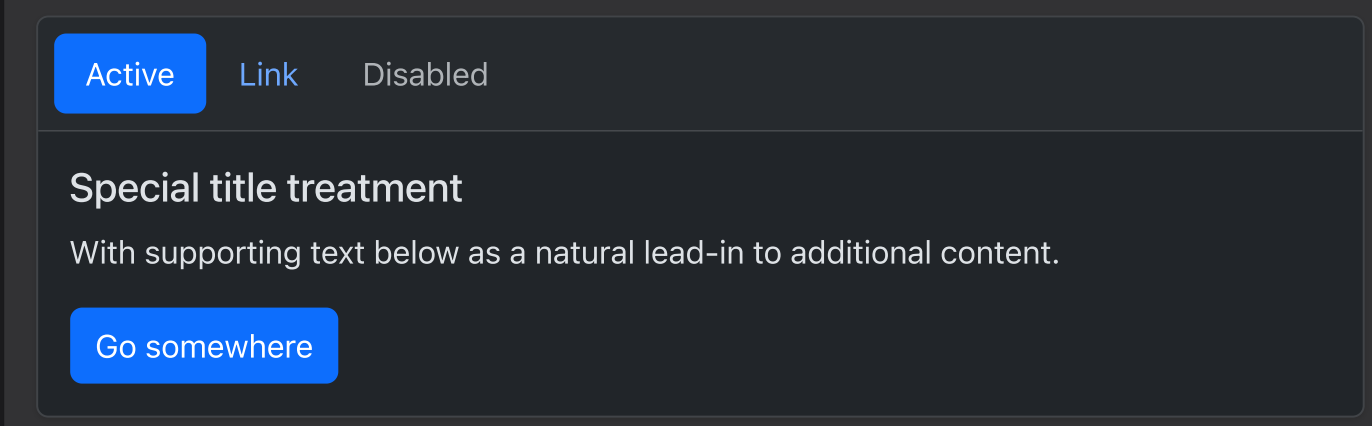
RESULT



LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Nav from 'react-bootstrap/Nav';

function NavTabsExample() {
  return (
    <Card>
      <Card.Header>
        <Nav variant="tabs" defaultActiveKey="#first">
          <Nav.Item>
            <Nav.Link href="#first">Active</Nav.Link>
          </Nav.Item>
          <Nav.Item>
            <Nav.Link href="#link">Link</Nav.Link>
          </Nav.Item>
          <Nav.Item>
            <Nav.Link href="#disabled" disabled>
              Disabled
            </Nav.Link>
          </Nav.Item>
        </Nav>
      </Card.Header>
      <Card.Body>
        <Card.Title>Special title treatment</Card.Title>
        <Card.Text>
          With supporting text below as a natural lead-in to additional
content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default NavTabsExample;
```

RESULT



LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Nav from 'react-bootstrap/Nav';
```

```
function NavPillsExample() {
  return (
    <Card>
      <Card.Header>
        <Nav variant="pills" defaultActiveKey="#first">
          <Nav.Item>
            <Nav.Link href="#first">Active</Nav.Link>
          </Nav.Item>
          <Nav.Item>
            <Nav.Link href="#link">Link</Nav.Link>
          </Nav.Item>
          <Nav.Item>
            <Nav.Link href="#disabled" disabled>
              Disabled
            </Nav.Link>
          </Nav.Item>
        </Nav>
      </Card.Header>
      <Card.Body>
        <Card.Title>Special title treatment</Card.Title>
        <Card.Text>
          With supporting text below as a natural lead-in to additional
content.
        </Card.Text>
        <Button variant="primary">Go somewhere</Button>
      </Card.Body>
    </Card>
  );
}

export default NavPillsExample;
```

Card Styles

Background Color

You can change a card's appearance by changing their `<bg>`, and `<text>` props.

RESULT

Header

Primary Card Title

Some quick example text to build on the card title and make up the bulk of the card's content.

Header

Secondary Card Title

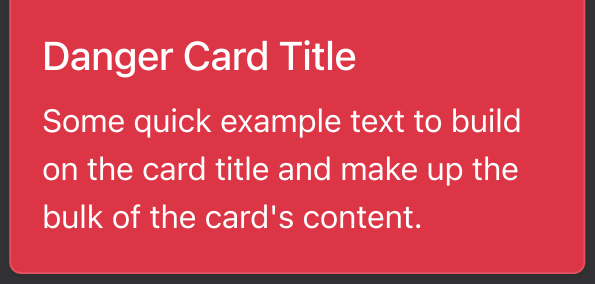
Some quick example text to build on the card title and make up the bulk of the card's content.

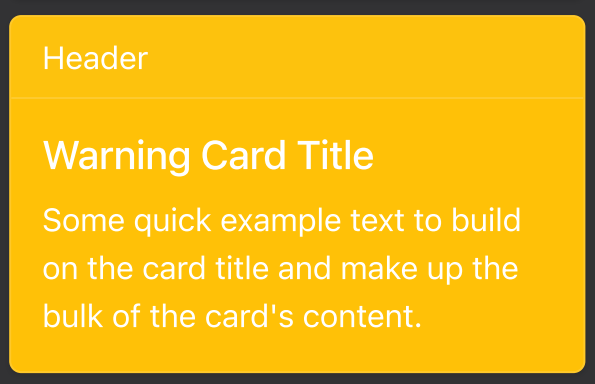
Header

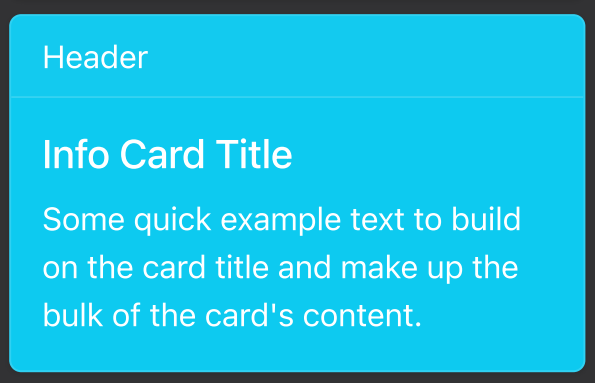
Success Card Title

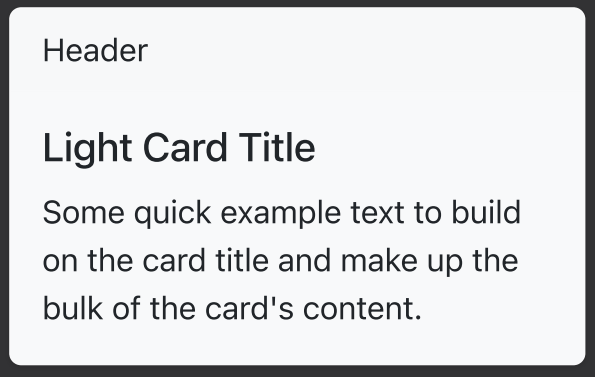
Some quick example text to build on the card title and make up the bulk of the card's content.

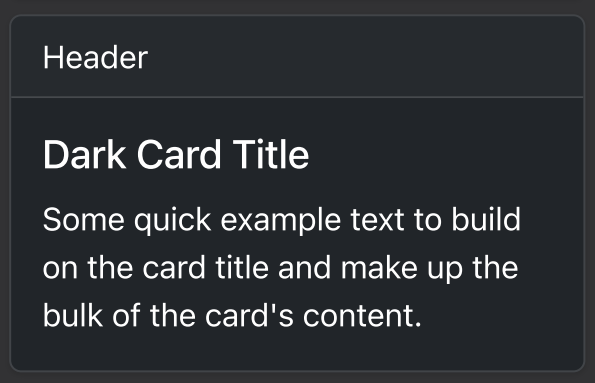
Header











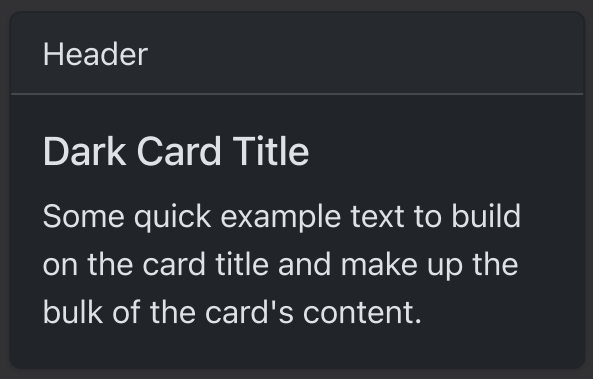
LIVE EDITOR

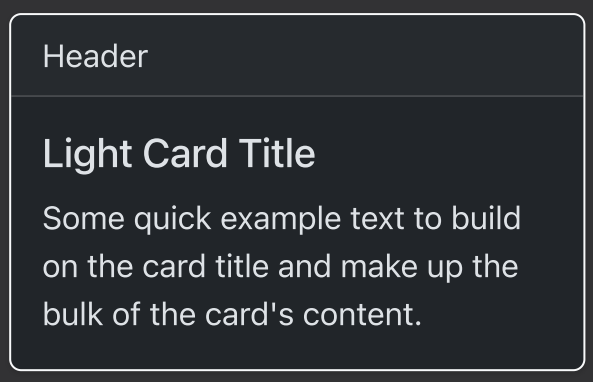
```jsx
import Card from 'react-bootstrap/Card';

function BgColorExample() {
  return (
    <>
      {[
        'Primary',
        'Secondary',
        'Success',
        'Danger',
        'Warning',
        'Info',
        'Light',
        'Dark',
      ].map((variant) => (
        <Card
          bg={variant.toLowerCase()}
          key={variant}
          text={variant.toLowerCase() === 'light' ? 'dark' : 'white'}
          style={{ width: '18rem' }}
          className="mb-2"
        >
          <Card.Header>Header</Card.Header>
          <Card.Body>
            <Card.Title>{variant} Card Title </Card.Title>
            <Card.Text>
              Some quick example text to build on the card title and make
up the
```

```
              bulk of the card's content.
            </Card.Text>
          </Card.Body>
        </Card>
      ))}
    </>
  );
}

export default BgColorExample;
```

Border Color

RESULT





LIVE EDITOR

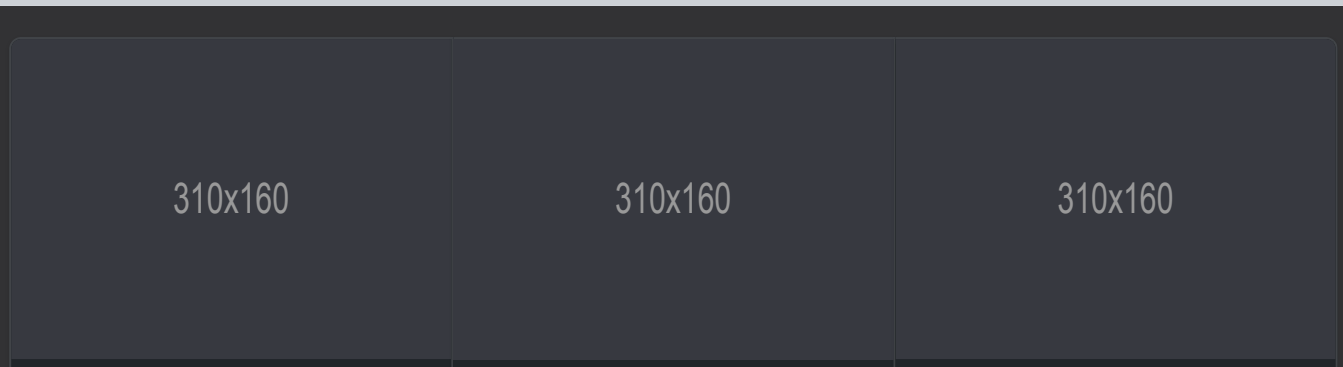
```jsx
import Card from 'react-bootstrap/Card';

function BorderExample() {
  return (
    <>
      <Card border="primary" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Primary Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="secondary" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Secondary Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="success" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Success Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="danger" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Danger Card Title</Card.Title>
          <Card.Text>
```

```
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="warning" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Warning Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="info" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Info Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="dark" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Dark Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />

      <Card border="light" style={{ width: '18rem' }}>
        <Card.Header>Header</Card.Header>
        <Card.Body>
          <Card.Title>Light Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
          </Card.Text>
        </Card.Body>
      </Card>
      <br />
    </>
  );
}

export default BorderExample;
```

Card layout

Card Groups

RESULT



LIVE EDITOR

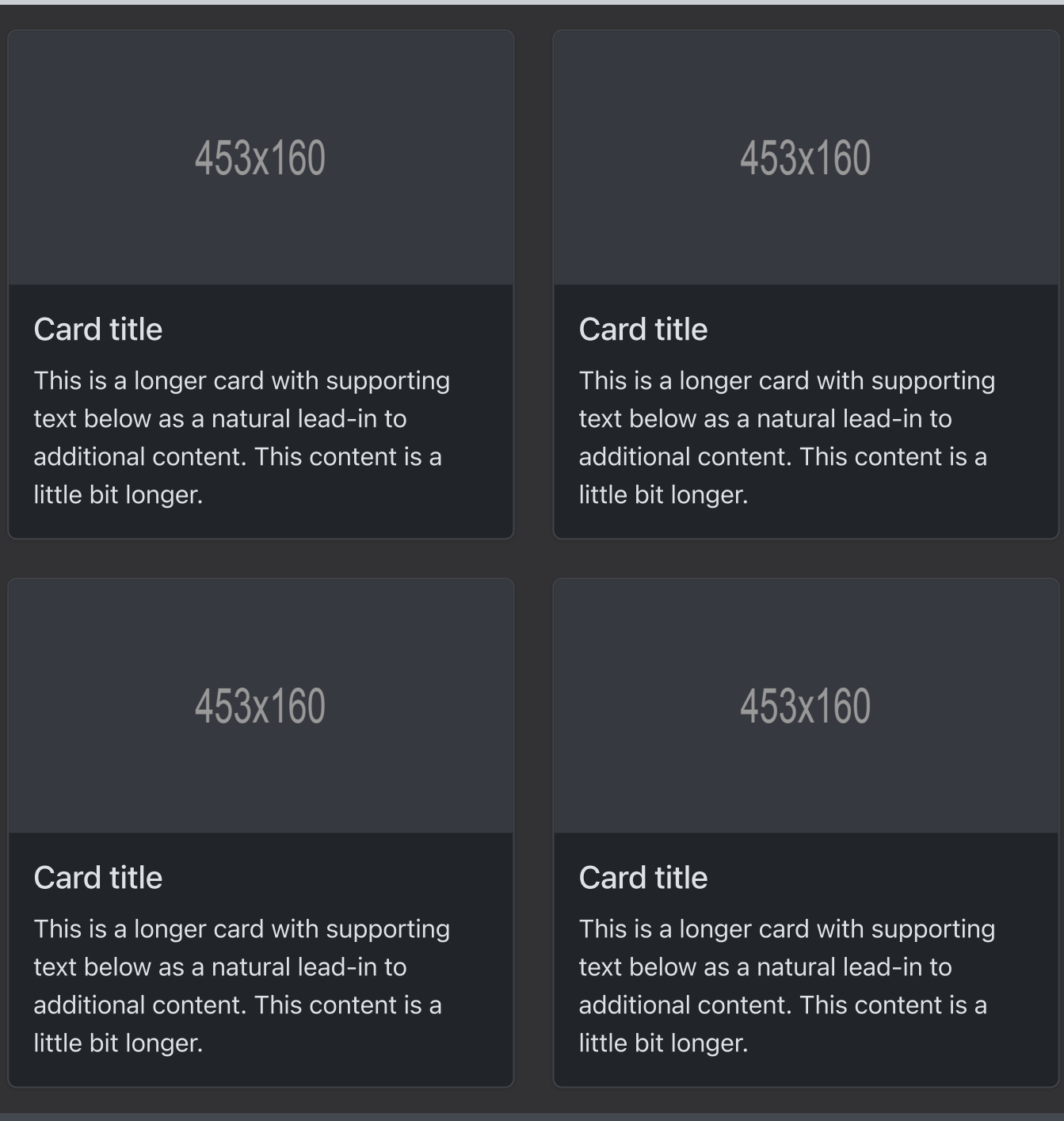
```
import Card from 'react-bootstrap/Card';
import CardGroup from 'react-bootstrap/CardGroup';

function GroupExample() {
  return (
    <CardGroup>
      <Card>
        <Card.Img variant="top" src="holder.js/100px160" />
        <Card.Body>
          <Card.Title>Card title</Card.Title>
          <Card.Text>
            This is a wider card with supporting text below as a natural
lead-in
            to additional content. This content is a little bit longer.
          </Card.Text>
        </Card.Body>
        <Card.Footer>
          <small className="text-muted">Last updated 3 mins ago</small>
        </Card.Footer>
      </Card>
      <Card>
        <Card.Img variant="top" src="holder.js/100px160" />
        <Card.Body>
          <Card.Title>Card title</Card.Title>
          <Card.Text>
            This card has supporting text below as a natural lead-in to
            additional content.{' '}
          </Card.Text>
        </Card.Body>
        <Card.Footer>
          <small className="text-muted">Last updated 3 mins ago</small>
        </Card.Footer>
      </Card>
      <Card>
        <Card.Img variant="top" src="holder.js/100px160" />
        <Card.Body>
          <Card.Title>Card title</Card.Title>
          <Card.Text>
            This is a wider card with supporting text below as a natural
lead-in
            to additional content. This card has even longer content than
the
            first to show that equal height action.
          </Card.Text>
        </Card.Body>
```

```
        <Card.Footer>
          <small className="text-muted">Last updated 3 mins ago</small>
        </Card.Footer>
      </Card>
    </CardGroup>
  );
}

export default GroupExample;
```

Grid cards

Use `Row`'s grid column props to control how many cards to show per row.

RESULT



LIVE EDITOR

```
import Card from 'react-bootstrap/Card';
import Col from 'react-bootstrap/Col';
import Row from 'react-bootstrap/Row';

function GridExample() {
  return (
    <Row xs={1} md={2} className="g-4">
      {Array.from({ length: 4 }).map((_, idx) => (
        <Col key={idx}>
          <Card>
            <Card.Img variant="top" src="holder.js/100px160" />
            <Card.Body>
              <Card.Title>Card title</Card.Title>
              <Card.Text>
                This is a longer card with supporting text below as a
natural
                lead-in to additional content. This content is a little
bit
                longer.
              </Card.Text>
```

```
            </Card.Body>
          </Card>
        </Col>
      ))}
    </Row>
  );
}

export default GridExample;
```

API

Card

```
import Card from 'react-bootstrap/Card'
```

Name	Type	Default	Description
bsPrefix	string	`'card'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
bg	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'dark'` | `'light'`		Sets card background
text	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'dark'` | `'light'` | `'white'` | `'muted'`		Sets card text color
border	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'dark'` | `'light'`		Sets card border color
body	bool	`false`	When this prop is set, it creates a Card with a Card.Body inside passing the children directly to it
as	elementType	`'div'`	You can use a custom element type for this component.

CardBody

```
import CardBody from 'react-bootstrap/CardBody'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

CardFooter

```
import CardFooter from 'react-bootstrap/CardFooter'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

CardHeader

```
import CardHeader from 'react-bootstrap/CardHeader'
```

Name	Type	Default	Description
bsPrefix	string	`'card-header'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	elementType	`'div'`	You can use a custom element type for this component.

CardImg

```
import CardImg from 'react-bootstrap/CardImg'
```

Name	Type	Default	Description
bsPrefix	string	`'card-img'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'top'` | `'bottom'`		Defines image position inside the card.
as	elementType	`'img'`	You can use a custom element type for this component.

CardImgOverlay

```
import CardImgOverlay from 'react-bootstrap/CardImgOverlay'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

CardLink

```
import CardLink from 'react-bootstrap/CardLink'
```

Name	Type	Default	Description
as		`'a'`	You can use a custom element type for this component.

CardSubtitle

```
import CardSubtitle from 'react-bootstrap/CardSubtitle'
```

Name	Type	Default	Description
as		`divWithClassName('h6')`	You can use a custom element type for this component.

CardText

```
import CardText from 'react-bootstrap/CardText'
```

Name	Type	Default	Description
as		`'p'`	You can use a custom element type for this component.

CardTitle

```
import CardTitle from 'react-bootstrap/CardTitle'
```

Name	Type	Default	Description
as		`divWithClassName('h5')`	You can use a custom element type for this component.

CardGroup

```
import CardGroup from 'react-bootstrap/CardGroup'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

Carousels

A slideshow component for cycling through elements—images or slides of text—like a carousel.

Example

Carousels don't automatically normalize slide dimensions. As such, you may need to use additional utilities or custom styles to appropriately size content. While carousels support previous/next controls and indicators, they're not explicitly required. Add and customize as you see fit.

RESULT

LIVE EDITOR

```
import Carousel from 'react-bootstrap/Carousel';
import ExampleCarouselImage from 'components/ExampleCarouselImage';

function UncontrolledExample() {
  return (
    <Carousel>
      <Carousel.Item>
        <ExampleCarouselImage text="First slide" />
        <Carousel.Caption>
          <h3>First slide label</h3>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Second slide" />
        <Carousel.Caption>
          <h3>Second slide label</h3>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
      </Carousel.Item>
```

```
      <Carousel.Item>
        <ExampleCarouselImage text="Third slide" />
        <Carousel.Caption>
          <h3>Third slide label</h3>
          <p>
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
}

export default UncontrolledExample;
```

Controlled

You can also *control* the Carousel state, via the `activeIndex` prop and `onSelect` handler.

RESULT

LIVE EDITOR

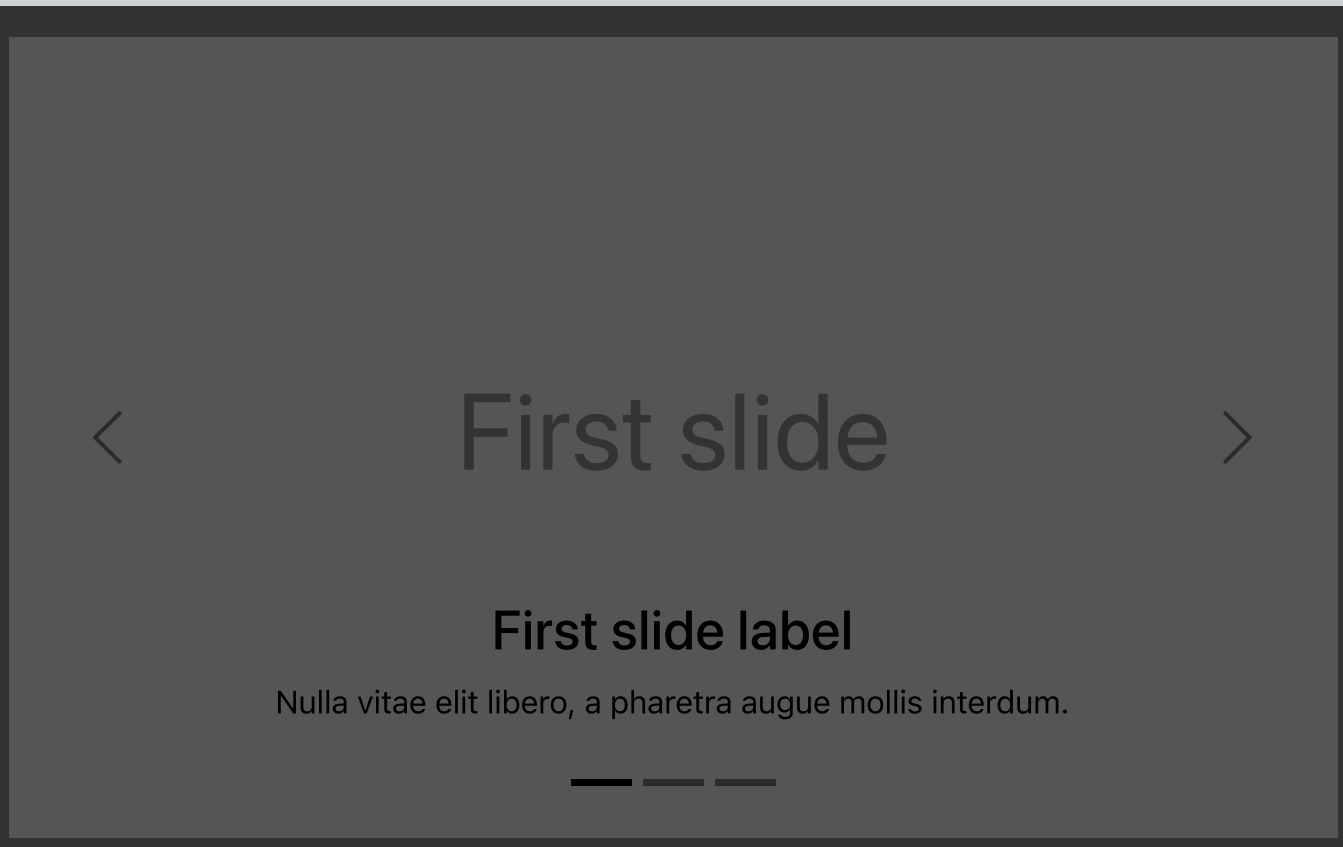
```
import { useState } from 'react';
import Carousel from 'react-bootstrap/Carousel';
import ExampleCarouselImage from 'components/ExampleCarouselImage';

function ControlledCarousel() {
  const [index, setIndex] = useState(0);

  const handleSelect = (selectedIndex) => {
    setIndex(selectedIndex);
  };

  return (
    <Carousel activeIndex={index} onSelect={handleSelect}>
      <Carousel.Item>
        <ExampleCarouselImage text="First slide" />
        <Carousel.Caption>
          <h3>First slide label</h3>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Second slide" />
        <Carousel.Caption>
          <h3>Second slide label</h3>
```

```
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Third slide" />
        <Carousel.Caption>
          <h3>Third slide label</h3>
          <p>
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
}

export default ControlledCarousel;
```

Crossfade

Add the `fade` prop to your carousel to animate slides with a fade transition instead of a slide.

RESULT



LIVE EDITOR

```
import Carousel from 'react-bootstrap/Carousel';
import ExampleCarouselImage from 'components/ExampleCarouselImage';

function CarouselFadeExample() {
  return (
    <Carousel fade>
      <Carousel.Item>
        <ExampleCarouselImage text="First slide" />
        <Carousel.Caption>
          <h3>First slide label</h3>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Second slide" />
        <Carousel.Caption>
          <h3>Second slide label</h3>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
```

```
        <ExampleCarouselImage text="Third slide" />
        <Carousel.Caption>
          <h3>Third slide label</h3>
          <p>
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
}

export default CarouselFadeExample;
```

No transition animation

Set the `slide` prop to false to disable the transition animation between slides.

RESULT

LIVE EDITOR

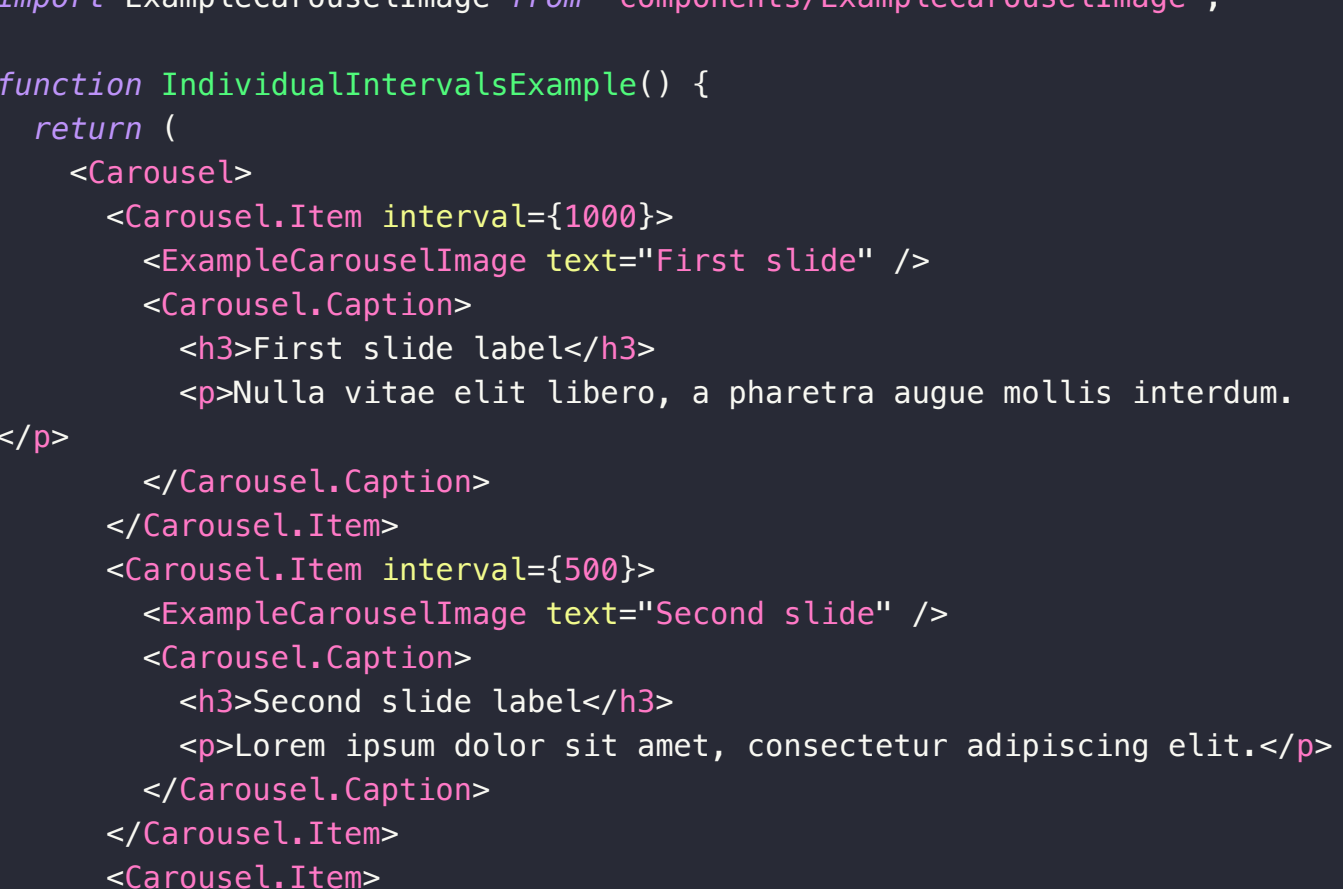
```
import Carousel from 'react-bootstrap/Carousel';
import ExampleCarouselImage from 'components/ExampleCarouselImage';

function NoTransitionExample() {
  return (
    <Carousel slide={false}>
      <Carousel.Item>
        <ExampleCarouselImage text="First slide" />
        <Carousel.Caption>
          <h3>First slide label</h3>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Second slide" />
        <Carousel.Caption>
          <h3>Second slide label</h3>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Third slide" />
        <Carousel.Caption>
          <h3>Third slide label</h3>
          <p>
```

```
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
}

export default NoTransitionExample;
```

Individual Item Intervals

You can specify individual intervals for each carousel item via the `interval` prop.

RESULT



LIVE EDITOR

```
import Carousel from 'react-bootstrap/Carousel';
import ExampleCarouselImage from 'components/ExampleCarouselImage';

function IndividualIntervalsExample() {
  return (
    <Carousel>
      <Carousel.Item interval={1000}>
        <ExampleCarouselImage text="First slide" />
        <Carousel.Caption>
          <h3>First slide label</h3>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item interval={500}>
        <ExampleCarouselImage text="Second slide" />
        <Carousel.Caption>
          <h3>Second slide label</h3>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <ExampleCarouselImage text="Third slide" />
        <Carousel.Caption>
          <h3>Third slide label</h3>
          <p>
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
```

```
      </Carousel.Item>
    </Carousel>
  );
}

export default IndividualIntervalsExample;
```

Dark variant

Add `variant="dark"` to the `Carousel` for darker controls, indicators, and captions.

> **Heads up!**
>
> Dark variants for components were deprecated in Bootstrap v5.3.0 with the introduction of color modes. Instead of adding `variant="dark"`, set `data-bs-theme="dark"` on the root element, a parent wrapper, or the component itself.

RESULT

LIVE EDITOR

```
import Carousel from 'react-bootstrap/Carousel';

function DarkVariantExample() {
  return (
    <Carousel data-bs-theme="dark">
      <Carousel.Item>
        <img
          className="d-block w-100"
          src="holder.js/800x400?text=First slide&bg=f5f5f5"
          alt="First slide"
        />
        <Carousel.Caption>
          <h5>First slide label</h5>
          <p>Nulla vitae elit libero, a pharetra augue mollis interdum.
</p>
        </Carousel.Caption>
      </Carousel.Item>
      <Carousel.Item>
        <img
          className="d-block w-100"
          src="holder.js/800x400?text=Second slide&bg=eee"
          alt="Second slide"
        />
        <Carousel.Caption>
          <h5>Second slide label</h5>
          <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit.</p>
        </Carousel.Caption>
```

```
      </Carousel.Item>
      <Carousel.Item>
        <img
          className="d-block w-100"
          src="holder.js/800x400?text=Third slide&bg=e5e5e5"
          alt="Third slide"
        />
        <Carousel.Caption>
          <h5>Third slide label</h5>
          <p>
            Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
          </p>
        </Carousel.Caption>
      </Carousel.Item>
    </Carousel>
  );
}

export default DarkVariantExample;
```

API

Carousel

```
import Carousel from 'react-bootstrap/Carousel'
```

Name	Type	Default	Description
bsPrefix	string	`'carousel'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	elementType		You can use a custom element type for this component.
slide	bool		Enables animation on the Carousel as it transitions between slides.
fade	bool		Animates slides with a crossfade animation instead of the default slide animation
controls	bool		Show the Carousel previous and next arrows for changing the current slide
indicators	bool		Show a set of slide position indicators
indicatorLabels	array		An array of labels for the indicators. Defaults to "Slide #" if not provided.
activeIndex	number		*controlled by:* `onSelect`, *initial prop:* `defaultActiveIndex` Controls the current visible slide
onSelect	func		*controls* `activeIndex` Callback fired when the active item changes. `(eventKey: number, event: Object | null) => void`

Name	Type	Default	Description
onSlide	func		Callback fired when a slide transition starts. `(eventKey: number, direction: 'left' | 'right') => void`
onSlid	func		Callback fired when a slide transition ends. `(eventKey: number, direction: 'left' | 'right') => void`
interval	`number` | `enum`		The amount of time to delay between automatically cycling an item. If `null`, carousel will not automatically cycle.
keyboard	bool		Whether the carousel should react to keyboard events.
pause	`'hover'` | `false`		If set to `"hover"`, pauses the cycling of the carousel on `mouseenter` and resumes the cycling of the carousel on `mouseleave`. If set to `false`, hovering over the carousel won't pause it. On touch-enabled devices, when set to `"hover"`, cycling will pause on `touchend` (once the user finished interacting with the carousel) for two intervals, before automatically resuming. Note that this is in addition to the above mouse behavior.
wrap	bool		Whether the carousel should cycle continuously or have hard stops.
touch	bool		Whether the carousel should support left/right swipe interactions on touchscreen devices.
prevIcon	node		Override the default button icon for the "previous" control
prevLabel	string		Label shown to screen readers only, can be used to show the previous element in the carousel. Set to null to deactivate.
nextIcon	node		Override the default button icon for the "next" control
nextLabel	string		Label shown to screen readers only, can be used to show the next element in the carousel. Set to null to deactivate.
variant	`'dark'`		Color variant that controls the colors of the controls, indicators and captions.

Name	Type	Default	Description
defaultActiveIndex		`0`	

CarouselItem

```
import CarouselItem from 'react-bootstrap/CarouselItem'
```

Name	Type	Default	Description
as	elementType	`'div'`	Set a custom element for this component
bsPrefix	string	`'carousel-item'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
interval	number		The amount of time to delay between automatically cycling this specific item. Will default to the Carousel's `interval` prop value if none is specified.

CarouselCaption

```
import CarouselCaption from 'react-bootstrap/CarouselCaption'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

Close Button

A generic close button for dismissing content such as modals and alerts.

Example

RESULT

LIVE EDITOR

```jsx
import CloseButton from 'react-bootstrap/CloseButton';

function BasicExample() {
  return <CloseButton />;
}

export default BasicExample;
```

Disabled state

Bootstrap adds relevant styling to a disabled close button to prevent user interactions.

RESULT

LIVE EDITOR

```jsx
import CloseButton from 'react-bootstrap/CloseButton';

function DisabledExample() {
  return <CloseButton disabled />;
}

export default DisabledExample;
```

Variants

Change the default dark color to white using `variant="white"`.

Heads up!

Dark variants for components were deprecated in Bootstrap v5.3.0 with the introduction of color modes. Instead of adding `variant="white"`, set `data-bs-theme="dark"` on the root element, a parent wrapper, or the component itself.

RESULT

LIVE EDITOR

```jsx
import CloseButton from 'react-bootstrap/CloseButton';

function VariantsExample() {
  return (
    <div data-bs-theme="dark" className='bg-dark p-2'>
      <CloseButton />
      <CloseButton disabled />
    </div>
  );
}

export default VariantsExample;
```

Accessibility

To ensure the maximum accessibility for Close Button components, it is recommended that you provide relevant text for screen readers. The example below provides an example of accessible usage of this component by way of the `aria-label` property.

RESULT

LIVE EDITOR

```jsx
import CloseButton from 'react-bootstrap/CloseButton';

function LabelledExample() {
  return <CloseButton aria-label="Hide" />;
}

export default LabelledExample;
```

API

CloseButton

```jsx
import CloseButton from 'react-bootstrap/CloseButton'
```

Name	Type	Default	Description
`aria-label`	`string`	`'Close'`	An accessible label indicating the relevant information about the Close Button.
`onClick`	`func`		A callback fired after the Close Button is clicked.
`variant`	`'white'`		Render different color variant for the button. Omitting this will render the default dark color.

Dropdowns

Toggle contextual overlays for displaying lists of links and more with the Bootstrap dropdown plugin

Overview

Dropdowns are toggleable, contextual overlays for displaying lists of links and more. Like overlays, Dropdowns are built using a third-party library Popper.js, which provides dynamic positioning and viewport detection.

Accessibility

The WAI ARIA standard defines a `role="menu"` widget, but it's very specific to a certain kind of menu. ARIA menus, must only contain `role="menuitem"`, `role="menuitemcheckbox"`, or `role="menuitemradio"`.

On the other hand, Bootstrap's dropdowns are designed to more generic and application in a variety of situations. For this reason we don't automatically add the menu roles to the markup. We do implement some basic keyboard navigation, and if you do provide the "menu" role, react-bootstrap will do its best to ensure the focus management is compliant with the ARIA authoring guidelines for menus.

Examples

Single button dropdowns

The basic Dropdown is composed of a wrapping `Dropdown` and inner `<DropdownMenu>`, and `<DropdownToggle>`. By default the `<DropdownToggle>` will render a `Button` component and accepts all the same props.

RESULT

Dropdown Button

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';

function BasicExample() {
  return (
    <Dropdown>
      <Dropdown.Toggle variant="success" id="dropdown-basic">
        Dropdown Button
      </Dropdown.Toggle>

      <Dropdown.Menu>
        <Dropdown.Item href="#/action-1">Action</Dropdown.Item>
        <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
        <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
```

```
      </Dropdown.Menu>
    </Dropdown>
  );
}

export default BasicExample;
```

Since the above is such a common configuration react-bootstrap provides the `<DropdownButton>` component to help reduce typing. Provide a `title` prop and some `<DropdownItem>`s and you're ready to go.

RESULT

Dropdown button

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function BasicButtonExample() {
  return (
    <DropdownButton id="dropdown-basic-button" title="Dropdown button">
      <Dropdown.Item href="#/action-1">Action</Dropdown.Item>
      <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
      <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
    </DropdownButton>
  );
}

export default BasicButtonExample;
```

DropdownButton will forward Button props to the underlying Toggle component

RESULT

Primary Secondary Success Info Warning Danger

LIVE EDITOR

```
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function VariantsExample() {
  return (
    <>
      {['Primary', 'Secondary', 'Success', 'Info', 'Warning',
'Danger'].map(
        (variant) => (
          <DropdownButton
            as={ButtonGroup}
            key={variant}
            id={`dropdown-variants-${variant}`}
            variant={variant.toLowerCase()}
            title={variant}
          >
            <Dropdown.Item eventKey="1">Action</Dropdown.Item>
            <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
            <Dropdown.Item eventKey="3" active>
              Active Item
            </Dropdown.Item>
            <Dropdown.Divider />
            <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
          </DropdownButton>
        ),
      )}
```

```
    </>
  );
}

export default VariantsExample;
```

Split button dropdowns

Similarly, You create a split dropdown by combining the Dropdown components with another Button and a ButtonGroup.

RESULT

Split Button

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import Dropdown from 'react-bootstrap/Dropdown';

function SplitBasicExample() {
  return (
    <Dropdown as={ButtonGroup}>
      <Button variant="success">Split Button</Button>

      <Dropdown.Toggle split variant="success" id="dropdown-split-basic"
/>

      <Dropdown.Menu>
        <Dropdown.Item href="#/action-1">Action</Dropdown.Item>
        <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
        <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
      </Dropdown.Menu>
    </Dropdown>
  );
}

export default SplitBasicExample;
```

As with DropdownButton, `SplitButton` is provided as convenience component.

RESULT

Primary Secondary Success Info Warning Danger

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import SplitButton from 'react-bootstrap/SplitButton';

function SplitVariantExample() {
  return (
    <>
      {['Primary', 'Secondary', 'Success', 'Info', 'Warning',
'Danger'].map(
        (variant) => (
          <SplitButton
            key={variant}
            id={`dropdown-split-variants-${variant}`}
            variant={variant.toLowerCase()}
            title={variant}
          >
            <Dropdown.Item eventKey="1">Action</Dropdown.Item>
            <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
            <Dropdown.Item eventKey="3" active>
```

```
              Active Item
            </Dropdown.Item>
            <Dropdown.Divider />
            <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
          </SplitButton>
        ),
      )}
    </>
  );
}

export default SplitVariantExample;
```

Sizing

Dropdowns work with buttons of all sizes.

RESULT

Drop large Drop large

Drop small Drop small

LIVE EDITOR

```
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';
import SplitButton from 'react-bootstrap/SplitButton';

function ButtonSizesExample() {
  return (
    <>
      <div className="mb-2">
        {[DropdownButton, SplitButton].map((DropdownType, idx) => (
          <DropdownType
            as={ButtonGroup}
            key={idx}
            id={`dropdown-button-drop-${idx}`}
            size="lg"
            title="Drop large"
          >
            <Dropdown.Item eventKey="1">Action</Dropdown.Item>
            <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
            <Dropdown.Item eventKey="3">Something else
here</Dropdown.Item>
            <Dropdown.Divider />
            <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
          </DropdownType>
        ))}
      </div>
      <div>
        {[DropdownButton, SplitButton].map((DropdownType, idx) => (
          <DropdownType
            as={ButtonGroup}
            key={idx}
            id={`dropdown-button-drop-${idx}`}
            size="sm"
            variant="secondary"
            title="Drop small"
          >
            <Dropdown.Item eventKey="1">Action</Dropdown.Item>
            <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
            <Dropdown.Item eventKey="3">Something else
here</Dropdown.Item>
            <Dropdown.Divider />
            <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
          </DropdownType>
```

```
        ))}
      </div>
    </>
  );
}

export default ButtonSizesExample;
```

Dark dropdowns

Opt into darker dropdowns to match a dark navbar or custom style by adding `variant="dark"` onto an existing `DropdownMenu`. Alternatively, use `menuVariant="dark"` when using the `DropdownButton` component.

Heads up!

Dark variants for components were deprecated in Bootstrap v5.3.0 with the introduction of color modes. Instead of adding `variant="dark"`, set `data-bs-theme="dark"` on the root element, a parent wrapper, or the component itself.

RESULT

Dropdown Button

Dropdown button

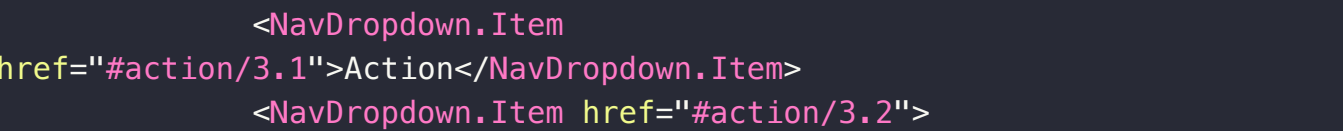
LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function ButtonDarkExample() {
  return (
    <>
      <Dropdown data-bs-theme="dark">
        <Dropdown.Toggle id="dropdown-button-dark-example1"
variant="secondary">
          Dropdown Button
        </Dropdown.Toggle>

        <Dropdown.Menu>
          <Dropdown.Item href="#/action-1" active>
            Action
          </Dropdown.Item>
          <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
          <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item href="#/action-4">Separated link</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>

      <DropdownButton
        id="dropdown-button-dark-example2"
        variant="secondary"
        title="Dropdown button"
        className="mt-2"
        data-bs-theme="dark"
      >
        <Dropdown.Item href="#/action-1" active>
          Action
        </Dropdown.Item>
        <Dropdown.Item href="#/action-2">Another action</Dropdown.Item>
        <Dropdown.Item href="#/action-3">Something else</Dropdown.Item>
        <Dropdown.Divider />
        <Dropdown.Item href="#/action-4">Separated link</Dropdown.Item>
      </DropdownButton>
```

```
    </>
  );
}

export default ButtonDarkExample;
```

Using `menuVariant="dark"` in a `NavDropdown`:

RESULT



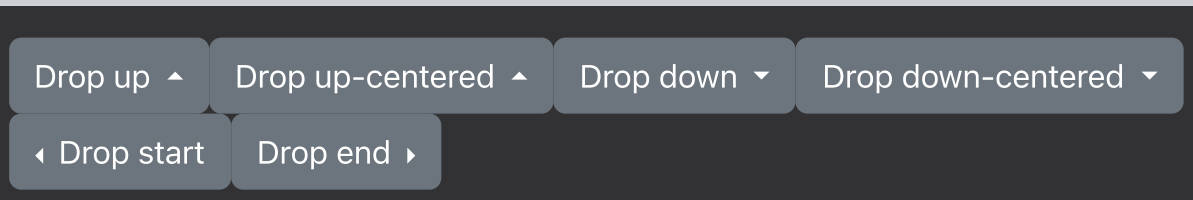
LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';

function NavbarDarkExample() {
  return (
    <Navbar variant="dark" bg="dark" expand="lg">
      <Container fluid>
        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>
        <Navbar.Toggle aria-controls="navbar-dark-example" />
        <Navbar.Collapse id="navbar-dark-example">
          <Nav>
            <NavDropdown
              id="nav-dropdown-dark-example"
              title="Dropdown"
              menuVariant="dark"
            >
              <NavDropdown.Item
href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Item
href="#action/3.3">Something</NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action/3.4">
                Separated link
              </NavDropdown.Item>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default NavbarDarkExample;
```

Drop directions

Trigger dropdown menus above, below, left, or to the right of their toggle elements, with the `drop` prop.

RESULT



Drop up | Drop up-centered | Drop down | Drop down-centered

Drop start | Drop end

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';
import SplitButton from 'react-bootstrap/SplitButton';

function DropDirectioExample() {
  return (
    <>
      <div className="mb-2">
        {['up', 'up-centered', 'down', 'down-centered', 'start',
'end'].map(
          (direction) => (
            <DropdownButton
              as={ButtonGroup}
              key={direction}
              id={`dropdown-button-drop-${direction}`}
              drop={direction}
              variant="secondary"
              title={` Drop ${direction} `}
            >
              <Dropdown.Item eventKey="1">Action</Dropdown.Item>
              <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
              <Dropdown.Item eventKey="3">Something else
here</Dropdown.Item>
              <Dropdown.Divider />
              <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
            </DropdownButton>
          ),
        )}
      </div>

      <div>
        {['up', 'up-centered', 'down', 'down-centered', 'start',
'end'].map(
          (direction) => (
            <SplitButton
              key={direction}
              id={`dropdown-button-drop-${direction}`}
              drop={direction}
              variant="secondary"
              title={`Drop ${direction}`}
            >
              <Dropdown.Item eventKey="1">Action</Dropdown.Item>
              <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
              <Dropdown.Item eventKey="3">Something else
here</Dropdown.Item>
              <Dropdown.Divider />
              <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
            </SplitButton>
          ),
        )}
      </div>
    </>
  );
}

export default DropDirectioExample;
```

Dropdown items

Historically dropdown menu contents had to be links, but that's no longer the case with v4. Now you can optionally use `<button>` elements in your dropdowns instead of just `<a>`s.

You can also create non-interactive dropdown items with `<Dropdown.ItemText>`. Feel free to style further with custom CSS or text utilities.

RESULT

Dropdown button

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function DropdownItemTagsExample() {
  return (
    <DropdownButton id="dropdown-item-button" title="Dropdown button">
      <Dropdown.ItemText>Dropdown item text</Dropdown.ItemText>
      <Dropdown.Item as="button">Action</Dropdown.Item>
      <Dropdown.Item as="button">Another action</Dropdown.Item>
      <Dropdown.Item as="button">Something else</Dropdown.Item>
    </DropdownButton>
  );
}

export default DropdownItemTagsExample;
```

Menu alignment

By default, a dropdown menu is aligned to the left, but you can switch it by passing `align="end"` to a `<Dropdown>`, `<DropdownButton>`, or `<SplitButton>`.

RESULT

Dropdown end

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';

function MenuAlignEndExample() {
  return (
    <DropdownButton
      align="end"
      title="Dropdown end"
      id="dropdown-menu-align-end"
    >
      <Dropdown.Item eventKey="1">Action</Dropdown.Item>
      <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
      <Dropdown.Item eventKey="3">Something else here</Dropdown.Item>
      <Dropdown.Divider />
      <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
    </DropdownButton>
  );
}

export default MenuAlignEndExample;
```

Responsive alignment

If you want to use responsive menu alignment, pass an object containing a breakpoint to the `align` prop on the `<DropdownMenu>`, `<DropdownButton>`, or `<SplitButton>`. You can specify `start` or `end` for the various breakpoints.

Warning

Using responsive alignment will disable Popper usage so any dynamic positioning features such as `flip` will not work.

RESULT

Left-aligned but right aligned when large screen

Right-aligned but left aligned when large screen

LIVE EDITOR

```jsx
import Dropdown from 'react-bootstrap/Dropdown';
import DropdownButton from 'react-bootstrap/DropdownButton';
import SplitButton from 'react-bootstrap/SplitButton';

function MenuAlignResponsiveExample() {
  return (
    <>
      <div>
        <DropdownButton
          as={ButtonGroup}
          align={{ lg: 'end' }}
          title="Left-aligned but right aligned when large screen"
          id="dropdown-menu-align-responsive-1"
        >
          <Dropdown.Item eventKey="1">Action 1</Dropdown.Item>
          <Dropdown.Item eventKey="2">Action 2</Dropdown.Item>
        </DropdownButton>
      </div>
      <div className="mt-2">
        <SplitButton
          align={{ lg: 'start' }}
          title="Right-aligned but left aligned when large screen"
          id="dropdown-menu-align-responsive-2"
        >
          <Dropdown.Item eventKey="1">Action 1</Dropdown.Item>
          <Dropdown.Item eventKey="2">Action 2</Dropdown.Item>
        </SplitButton>
      </div>
    </>
  );
}

export default MenuAlignResponsiveExample;
```

Menu headers

Add a header to label sections of actions.

RESULT

Dropdown header

Another action

Something else here

LIVE EDITOR

```jsx
import Dropdown from 'react-bootstrap/Dropdown';

function MenuHeadersExample() {
  return (
```

```
    <Dropdown.Menu show>
      <Dropdown.Header>Dropdown header</Dropdown.Header>
      <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
      <Dropdown.Item eventKey="3">Something else here</Dropdown.Item>
    </Dropdown.Menu>
  );
}

export default MenuHeadersExample;
```

Menu dividers

Separate groups of related menu items with a divider.

RESULT

Action

Another action

Something else here

Separated link

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';

function MenuDividersExample() {
  return (
    <Dropdown.Menu show>
      <Dropdown.Item eventKey="1">Action</Dropdown.Item>
      <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
      <Dropdown.Item eventKey="3">Something else here</Dropdown.Item>
      <Dropdown.Divider />
      <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
    </Dropdown.Menu>
  );
}

export default MenuDividersExample;
```

AutoClose

By default, the dropdown menu is closed when selecting a menu item or clicking outside of the dropdown menu. This behaviour can be changed by using the `autoClose` property.

By default, `autoClose` is set to the default value `true` and behaves like expected. By choosing `false`, the dropdown menu can only be toggled by clicking on the dropdown button. `inside` makes the dropdown disappear **only** by choosing a menu item and `outside` closes the dropdown menu **only** by clicking outside.

Notice how the dropdown is toggled in each scenario by clicking on the button.

RESULT

Default Dropdown Clickable Outside Clickable Inside Manual Close

LIVE EDITOR

```
import Dropdown from 'react-bootstrap/Dropdown';
```

```jsx
function AutoCloseExample() {
  return (
    <>
      <Dropdown className="d-inline mx-2">
        <Dropdown.Toggle id="dropdown-autoclose-true">
          Default Dropdown
        </Dropdown.Toggle>

        <Dropdown.Menu>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>

      <Dropdown className="d-inline mx-2" autoClose="inside">
        <Dropdown.Toggle id="dropdown-autoclose-inside">
          Clickable Outside
        </Dropdown.Toggle>

        <Dropdown.Menu>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>

      <Dropdown className="d-inline mx-2" autoClose="outside">
        <Dropdown.Toggle id="dropdown-autoclose-outside">
          Clickable Inside
        </Dropdown.Toggle>

        <Dropdown.Menu>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>

      <Dropdown className="d-inline mx-2" autoClose={false}>
        <Dropdown.Toggle id="dropdown-autoclose-false">
          Manual Close
        </Dropdown.Toggle>

        <Dropdown.Menu>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
          <Dropdown.Item href="#">Menu Item</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>
    </>
  );
}

export default AutoCloseExample;
```

Customization

If the default handling of the dropdown menu and toggle components aren't to your liking, you can customize them, by using the more basic `<Dropdown>` Component to explicitly specify the Toggle and Menu components

RESULT

Pow! Zoom! mix it up style-wise

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import ButtonGroup from 'react-bootstrap/ButtonGroup';
import Dropdown from 'react-bootstrap/Dropdown';

function ButtonCustomExample() {
  return (
    <>
      <Dropdown as={ButtonGroup}>
        <Dropdown.Toggle id="dropdown-custom-1">Pow! Zoom!
</Dropdown.Toggle>
        <Dropdown.Menu className="super-colors">
          <Dropdown.Item eventKey="1">Action</Dropdown.Item>
          <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
          <Dropdown.Item eventKey="3" active>
            Active Item
          </Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>{' '}
      <Dropdown as={ButtonGroup}>
        <Button variant="info">mix it up style-wise</Button>
        <Dropdown.Toggle split variant="success" id="dropdown-custom-2"
/>
        <Dropdown.Menu className="super-colors">
          <Dropdown.Item eventKey="1">Action</Dropdown.Item>
          <Dropdown.Item eventKey="2">Another action</Dropdown.Item>
          <Dropdown.Item eventKey="3" active>
            Active Item
          </Dropdown.Item>
          <Dropdown.Divider />
          <Dropdown.Item eventKey="4">Separated link</Dropdown.Item>
        </Dropdown.Menu>
      </Dropdown>
    </>
  );
}

export default ButtonCustomExample;
```

Custom Dropdown Components

For those that want to customize everything, you can forgo the included Toggle and Menu components, and create your own. By providing custom components to the `as` prop, you can control how each component behaves. Custom toggle and menu components **must** be able to accept refs.

RESULT

Custom toggle▼

LIVE EDITOR

```
import React, { useState } from 'react';
import Dropdown from 'react-bootstrap/Dropdown';
import Form from 'react-bootstrap/Form';

// The forwardRef is important!!
// Dropdown needs access to the DOM node in order to position the Menu
const CustomToggle = React.forwardRef(({ children, onClick }, ref) => (
  <a
    href=""
    ref={ref}
    onClick={(e) => {
      e.preventDefault();
      onClick(e);
    }}
  >
```

```
    {children}
    &#x25bc;
  </a>
));

// forwardRef again here!
// Dropdown needs access to the DOM of the Menu to measure it
const CustomMenu = React.forwardRef(
  ({ children, style, className, 'aria-labelledby': labeledBy }, ref) =>
{
    const [value, setValue] = useState('');

    return (
      <div
        ref={ref}
        style={style}
        className={className}
        aria-labelledby={labeledBy}
      >
        <Form.Control
          autoFocus
          className="mx-3 my-2 w-auto"
          placeholder="Type to filter..."
          onChange={(e) => setValue(e.target.value)}
          value={value}
        />
        <ul className="list-unstyled">
          {React.Children.toArray(children).filter(
            (child) =>
              !value ||
child.props.children.toLowerCase().startsWith(value),
          )}
        </ul>
      </div>
    );
  },
);

render(
  <Dropdown>
    <Dropdown.Toggle as={CustomToggle} id="dropdown-custom-components">
      Custom toggle
    </Dropdown.Toggle>

    <Dropdown.Menu as={CustomMenu}>
      <Dropdown.Item eventKey="1">Red</Dropdown.Item>
      <Dropdown.Item eventKey="2">Blue</Dropdown.Item>
      <Dropdown.Item eventKey="3" active>
        Orange
      </Dropdown.Item>
      <Dropdown.Item eventKey="1">Red-Orange</Dropdown.Item>
    </Dropdown.Menu>
  </Dropdown>,
);
```

API

DropdownButton

```
import DropdownButton from 'react-bootstrap/DropdownButton'
```

A convenience component for simple or general use dropdowns. Renders a `Button` toggle and all `children` are passed directly to the default `Dropdown.Menu`. This component accepts all of `Dropdown`'s props.

All unknown props are passed through to the `Dropdown` component. Only the Button `variant`, `size` and `bsPrefix` props are passed to the toggle, along with menu-related props are passed

to the `Dropdown.Menu`

Name	Type	Default	Description
`id`	`string`		An html id attribute for the Toggle button, necessary for assistive technologies, such as screen readers.
`href`	`string`		An `href` passed to the Toggle component
`onClick`	`func`		An `onClick` handler passed to the Toggle component
`title` Required	`node`		The content of the non-toggle Button.
`disabled`	`bool`		Disables both Buttons
`align`	`"start" | "end" | { sm: "start" | "end" } | { md: "start" | "end" } | { lg: "start" | "end" } | { xl: "start" | "end"} | { xxl: "start" | "end"}`		Aligns the dropdown menu. *see DropdownMenu for more details*
`menuRole`	`string`		An ARIA accessible role applied to the Menu component. When set to 'menu', The dropdown
`renderMenuOnMount`	`bool`		Whether to render the dropdown menu in the DOM before the first time it is shown
`rootCloseEvent`	`string`		Which event when fired outside the component will cause it to be closed. *see DropdownMenu for more details*
`menuVariant`	`'dark'`		Menu color variant. Omitting this will use the default light color.
`flip`	`bool`		Allow Dropdown to flip in case of an overlapping on the reference element. For more information refer to Popper.js's flip docs.
`bsPrefix`	`string`		Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
`variant`	`string`		Component visual or contextual style variants.

Name	Type	Default	Description
size	string		Component size variations.

SplitButton

```
import SplitButton from 'react-bootstrap/SplitButton'
```

A convenience component for simple or general use split button dropdowns. Renders a `ButtonGroup` containing a `Button` and a `Button` toggle for the `Dropdown`. All `children` are passed directly to the default `Dropdown.Menu`. This component accepts all of `Dropdown`'s props.

All unknown props are passed through to the `Dropdown` component. The Button `variant`, `size` and `bsPrefix` props are passed to the button and toggle, and menu-related props are passed to the `Dropdown.Menu`

Name	Type	Default	Description
id	`string`		An html id attribute for the Toggle button, necessary for assistive technologies, such as screen readers.
toggleLabel	string	`'Toggle dropdown'`	Accessible label for the toggle; the value of `title` if not specified.
href	string		An `href` passed to the non-toggle Button
target	string		An anchor `target` passed to the non-toggle Button
onClick	func		An `onClick` handler passed to the non-toggle Button
title Required	node		The content of the non-toggle Button.
type	string	`'button'`	A `type` passed to the non-toggle Button
disabled	bool		Disables both Buttons
align	`"start"` | `"end"` | `{ sm: "start"` | `"end" }` | `{ md: "start"` | `"end" }` | `{ lg: "start"` | `"end" }` | `{ xl: "start"` | `"end"}` | `{ xxl: "start"` | `"end"}`		Aligns the dropdown menu. *see DropdownMenu for more details*
menuRole	string		An ARIA accessible role applied to the Menu component. When set to 'menu', The dropdown
renderMenuOnMount	bool		Whether to render the dropdown menu in the DOM before the first time it is shown
rootCloseEvent	string		Which event when fired outside the component will

Name	Type	Default	Description
			cause it to be closed. *see DropdownMenu for more details*
`flip`	`bool`		Allow Dropdown to flip in case of an overlapping on the reference element. For more information refer to Popper.js's flip docs.
`bsPrefix`	`string`		Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
`variant`	`string`		Component visual or contextual style variants.
`size`	`string`		Component size variations.

Dropdown

```
import Dropdown from 'react-bootstrap/Dropdown'
```

Name	Type	Default	Description
`bsPrefix`	`string`	`'dropdown'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
`drop`	`'up'` | `'up-centered'` | `'start'` | `'end'` | `'down'` | `'down-centered'`		Determines the direction and location of the Menu in relation to it's Toggle.
`as`	`elementType`		You can use a custom element type for this component.
`align`	`"start"` | `"end"` | `{ sm: "start" | "end" }` | `{ md: "start" | "end" }` | `{ lg: "start" | "end" }` | `{ xl: "start" | "end"}` | `{ xxl: "start" | "end"}`		Aligns the dropdown menu to the specified side of the Dropdown toggle. You can also align the men responsively for breakpoints starting at `sm` and up. The alignment direction will affect the specified breakpoint or larger. *Note: Using responsive alignment will disable Popper usage for positioning.*
`show`	`bool`		*controlled by: `onToggle`, initial prop: `defaultShow`*

Name	Type	Default	Description
			Whether or not the Dropdown is visible.

onToggle	func		*controls* `show`
			
			A callback fired when the Dropdown wishes to change visibility. Called with the requested `show` value, the DOM event, and the source that fired it: `'click'`, `'keydown'`, `'rootClose'` or `'select'`.
			
			`function(`
			`  nextShow: boolean,`
			`  meta: ToggleMetadata,`
			`): void`

onSelect	func		A callback fired when a menu item is selected. `(eventKey: any, event: Object => any`
focusFirstItemOnShow	`false` | `true` | `'keyboard'`		Controls the focus behavior for when the Dropdown is opened. Se to `true` to always focus the first menu item, `keyboard` to focus onl when navigating via the keyboard, or `false` to disable completely The Default behavior is `false` **unless** the Menu has a `role="menu"` where it will default to `keyboard` to match the recommended ARIA Authoring practices.
autoClose	`true` | `'outside'` | `'inside'` | `false`		Controls the auto close behaviour of the dropdown when clicking outside of the button or the list.

DropdownToggle

```
import DropdownToggle from 'react-bootstrap/DropdownToggle'
```

Name	Type	Default	Description
bsPrefix	string	`'dropdown-toggle'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
id	`string` | `number`		An html id attribute, necessary for assistive technologies, such as screen readers.
split	bool		
as	elementType	`Button`	You can use a custom element type for this component.

DropdownMenu

```
import DropdownMenu from 'react-bootstrap/DropdownMenu'
```

Name	Type	Default	Description
bsPrefix	string	`'dropdown-menu'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
show	bool		Controls the visibility of the Dropdown menu
renderOnMount	bool		Whether to render the dropdown menu in the DOM before the first time it is shown
flip	bool	`true`	Have the dropdown switch to it's opposite placement when necessary to stay on screen.
align	`"start"` | `"end"` | `{ sm: "start"` | `"end" }` | `{ md: "start"` | `"end" }` | `{ lg: "start"` | `"end" }` | `{ xl: "start"` | `"end"}` | `{ xxl: "start"` | `"end"}`		Aligns the dropdown menu to the specified side of the container. You can also align the menu responsively for breakpoints starting at `sm` and up. The alignment direction will affect the specified breakpoint or larger. *Note: Using responsive alignment will disable Popper usage for positioning.*
rootCloseEvent	`'click'` | `'mousedown'`		Which event when fired outside the component will cause it to be closed *Note: For custom dropdown components, you will have to pass the `rootCloseEvent` to `<RootCloseWrapper>` in your custom dropdown menu component (similarly to how it is implemented in `<Dropdown.Menu>`).*
as	elementType	`'div'`	Control the rendering of the DropdownMenu. All non-menu props (listed here) are passed through to the `as` Component. If providing a custom, non DOM, component. the `show`, `close` and `align` props are also injected and should be handled appropriately.
popperConfig	object		A set of popper options and props passed directly to Popper.

Name	Type	Default	Description
variant	string		Menu color variant. Omitting this will use the default light color.

DropdownItem

```
import DropdownItem from 'react-bootstrap/DropdownItem'
```

Name	Type	Default	Description
bsPrefix	string	`'dropdown-item'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
active	bool		Highlight the menu item as active.
disabled	bool	`false`	Disable the menu item, making it unselectable.
eventKey	`string` | `number`		Value passed to the `onSelect` handler, useful for identifying the selected menu item.
href	string		HTML `href` attribute corresponding to `a.href`.
onClick	func		Callback fired when the menu item is clicked.
as	elementType	`Anchor`	You can use a custom element type for this component.

DropdownHeader

```
import DropdownHeader from 'react-bootstrap/DropdownHeader'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.
role		`'heading'`	

DropdownDivider

```
import DropdownDivider from 'react-bootstrap/DropdownDivider'
```

Name	Type	Default	Description
as		`'hr'`	You can use a custom element type for this component.
role		`'separator'`	

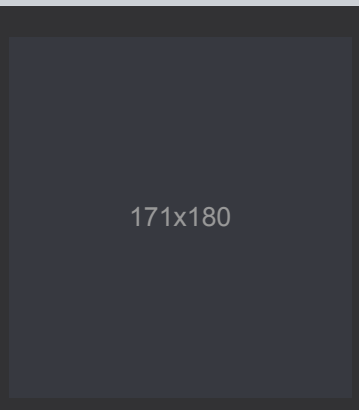
Figures

Documentation and examples for displaying related images and text with the figure component in Bootstrap.

Figure

Displaying related images and text with the Figure component.

RESULT



Nulla vitae elit libero, a pharetra augue mollis interdum.

LIVE EDITOR

```jsx
import Figure from 'react-bootstrap/Figure';

function FigureExample() {
  return (
    <Figure>
      <Figure.Image
        width={171}
        height={180}
        alt="171x180"
        src="holder.js/171x180"
      />
      <Figure.Caption>
        Nulla vitae elit libero, a pharetra augue mollis interdum.
      </Figure.Caption>
    </Figure>
  );
}

export default FigureExample;
```

API

Figure

```jsx
import Figure from 'react-bootstrap/Figure'
```

Name	Type	Default	Description
as		`'figure'`	You can use a custom element type for this component.

FigureImage

```
import FigureImage from 'react-bootstrap/FigureImage'
```

Name	Type	Default	Description
fluid		`true`	

FigureCaption

```
import FigureCaption from 'react-bootstrap/FigureCaption'
```

Name	Type	Default	Description
as		`'figcaption'`	You can use a custom element type for this component.

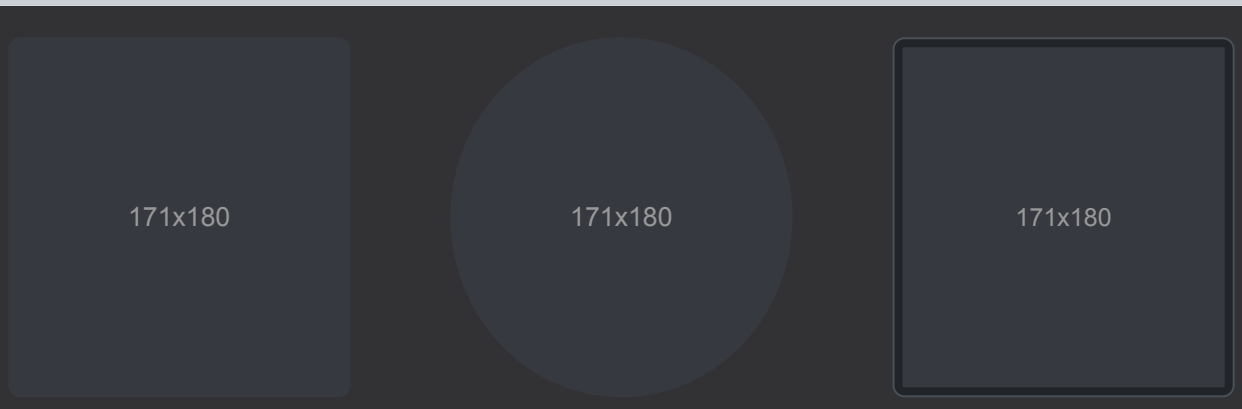
Images

Documentation and examples for opting images into responsive behavior (so they never become wider than their parent) and add lightweight styles to them—all via classes.

Shape

Use the `rounded`, `roundedCircle` and `thumbnail` props to customise the image.

RESULT



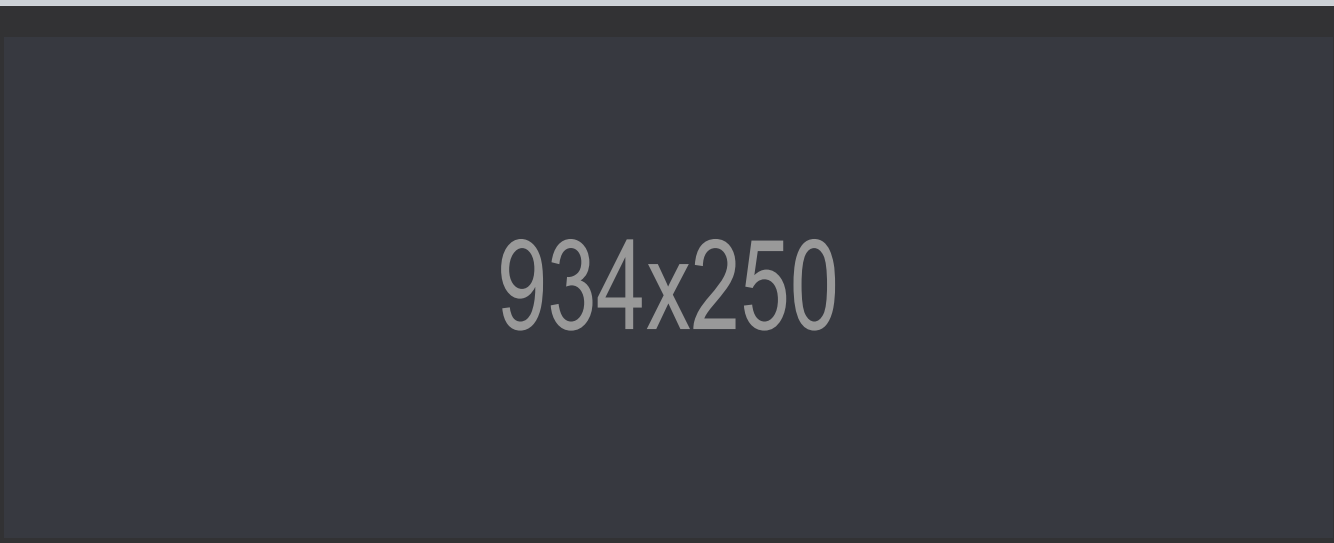
LIVE EDITOR

```jsx
import Col from 'react-bootstrap/Col';
import Container from 'react-bootstrap/Container';
import Image from 'react-bootstrap/Image';
import Row from 'react-bootstrap/Row';

function ShapeExample() {
  return (
    <Container>
      <Row>
        <Col xs={6} md={4}>
          <Image src="holder.js/171x180" rounded />
        </Col>
        <Col xs={6} md={4}>
          <Image src="holder.js/171x180" roundedCircle />
        </Col>
        <Col xs={6} md={4}>
          <Image src="holder.js/171x180" thumbnail />
        </Col>
      </Row>
    </Container>
  );
}

export default ShapeExample;
```

Fluid

Use the `fluid` to scale image nicely to the parent element.

RESULT



LIVE EDITOR

```jsx
import Image from 'react-bootstrap/Image';

function FluidExample() {
  return <Image src="holder.js/100px250" fluid />;
}

export default FluidExample;
```

API

Image

```jsx
import Image from 'react-bootstrap/Image'
```

Name	Type	Default	Description
bsPrefix	string	'img'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
fluid	bool	false	Sets image as fluid image.
rounded	bool	false	Sets image shape as rounded.
roundedCircle	bool	false	Sets image shape as circle.
thumbnail	bool	false	Sets image shape as thumbnail.

List groups

List groups are a flexible and powerful component for displaying a series of content. Modify and extend them to support just about any content within.

Basic Example

RESULT

- Cras justo odio
- Dapibus ac facilisis in
- Morbi leo risus
- Porta ac consectetur ac
- Vestibulum at eros

LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function DefaultExample() {
  return (
    <ListGroup>
      <ListGroup.Item>Cras justo odio</ListGroup.Item>
      <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
      <ListGroup.Item>Morbi leo risus</ListGroup.Item>
      <ListGroup.Item>Porta ac consectetur ac</ListGroup.Item>
      <ListGroup.Item>Vestibulum at eros</ListGroup.Item>
    </ListGroup>
  );
}

export default DefaultExample;
```

Active items

Set the `active` prop to indicate the list groups current active selection.

RESULT

- **Cras justo odio**
- Dapibus ac facilisis in
- Morbi leo risus
- Porta ac consectetur ac

LIVE EDITOR

```jsx
import ListGroup from 'react-bootstrap/ListGroup';

function ActiveExample() {
  return (
    <ListGroup as="ul">
      <ListGroup.Item as="li" active>
        Cras justo odio
      </ListGroup.Item>
      <ListGroup.Item as="li">Dapibus ac facilisis in</ListGroup.Item>
      <ListGroup.Item as="li" disabled>
        Morbi leo risus
      </ListGroup.Item>
      <ListGroup.Item as="li">Porta ac consectetur ac</ListGroup.Item>
    </ListGroup>
  );
}

export default ActiveExample;
```

Disabled items

Set the `disabled` prop to prevent actions on a `<ListGroup.Item>`. For elements that aren't naturally disable-able (like anchors) `onClick` handlers are added that call `preventDefault` to mimick disabled behavior.

RESULT

- Cras justo odio
- Dapibus ac facilisis in
- Morbi leo risus
- Porta ac consectetur ac

LIVE EDITOR

```jsx
import ListGroup from 'react-bootstrap/ListGroup';

function DisabledExample() {
  return (
    <ListGroup>
      <ListGroup.Item disabled>Cras justo odio</ListGroup.Item>
      <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
      <ListGroup.Item>Morbi leo risus</ListGroup.Item>
      <ListGroup.Item>Porta ac consectetur ac</ListGroup.Item>
    </ListGroup>
  );
}

export default DisabledExample;
```

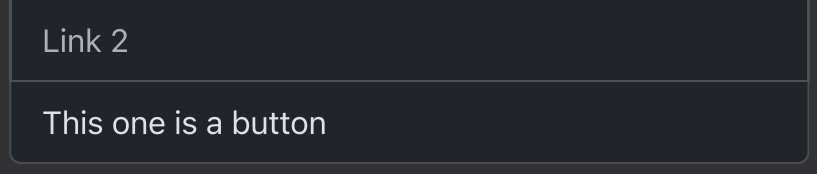
Actionable items

Toggle the `action` prop to create *actionable* list group items, with disabled, hover and active styles. List item actions will render a `<button>` or `<a>` (depending on the presence of an `href`) by default but can be overridden by setting the `as` prop as usual.

List items `actions` are distinct from plain items to ensure that click or tap affordances aren't applied to non-interactive items.

RESULT

- Link 1



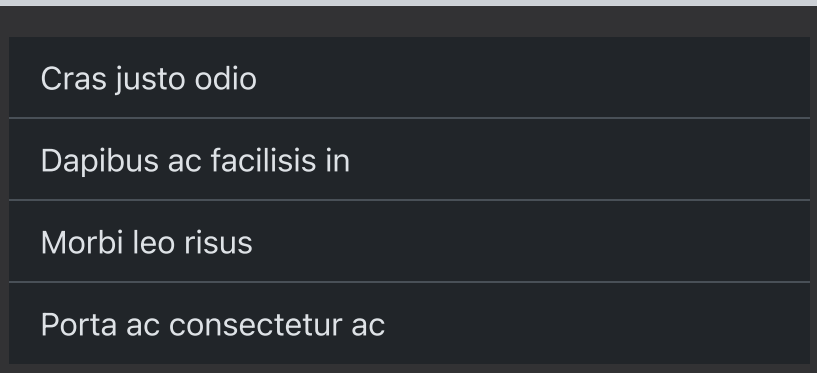
LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function LinkedExample() {
  const alertClicked = () => {
    alert('You clicked the third ListGroupItem');
  };

  return (
    <ListGroup defaultActiveKey="#link1">
      <ListGroup.Item action href="#link1">
        Link 1
      </ListGroup.Item>
      <ListGroup.Item action href="#link2" disabled>
        Link 2
      </ListGroup.Item>
      <ListGroup.Item action onClick={alertClicked}>
        This one is a button
      </ListGroup.Item>
    </ListGroup>
  );
}

export default LinkedExample;
```

Flush

Add the `flush` variant to remove outer borders and rounded corners to render list group items edge-to-edge in a parent container such as a `Card`.

RESULT



LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function FlushExample() {
  return (
    <ListGroup variant="flush">
      <ListGroup.Item>Cras justo odio</ListGroup.Item>
      <ListGroup.Item>Dapibus ac facilisis in</ListGroup.Item>
      <ListGroup.Item>Morbi leo risus</ListGroup.Item>
      <ListGroup.Item>Porta ac consectetur ac</ListGroup.Item>
    </ListGroup>
  );
}

export default FlushExample;
```

Numbered

Add the `numbered` prop to opt into numbered list group items. Numbers are generated via CSS (as opposed to a `<ol>`s default browser styling) for better placement inside list group items and

to allow for better customization.

RESULT

1. Cras justo odio
2. Cras justo odio
3. Cras justo odio

LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function NumberedExample() {
  return (
    <ListGroup as="ol" numbered>
      <ListGroup.Item as="li">Cras justo odio</ListGroup.Item>
      <ListGroup.Item as="li">Cras justo odio</ListGroup.Item>
      <ListGroup.Item as="li">Cras justo odio</ListGroup.Item>
    </ListGroup>
  );
}

export default NumberedExample;
```

These work great with custom content as well.

RESULT

1. **Subheading** 14
   Cras justo odio
2. **Subheading** 14
   Cras justo odio
3. **Subheading** 14
   Cras justo odio

LIVE EDITOR

```
import Badge from 'react-bootstrap/Badge';
import ListGroup from 'react-bootstrap/ListGroup';

function DefaultExample() {
  return (
    <ListGroup as="ol" numbered>
      <ListGroup.Item
        as="li"
        className="d-flex justify-content-between align-items-start"
      >
        <div className="ms-2 me-auto">
          <div className="fw-bold">Subheading</div>
          Cras justo odio
        </div>
        <Badge bg="primary" pill>
          14
        </Badge>
      </ListGroup.Item>
      <ListGroup.Item
        as="li"
        className="d-flex justify-content-between align-items-start"
      >
        <div className="ms-2 me-auto">
          <div className="fw-bold">Subheading</div>
          Cras justo odio
        </div>
```

```
        <Badge bg="primary" pill>
          14
        </Badge>
      </ListGroup.Item>
      <ListGroup.Item
        as="li"
        className="d-flex justify-content-between align-items-start"
      >
        <div className="ms-2 me-auto">
          <div className="fw-bold">Subheading</div>
          Cras justo odio
        </div>
        <Badge bg="primary" pill>
          14
        </Badge>
      </ListGroup.Item>
    </ListGroup>
  );
}

export default DefaultExample;
```

Horizontal

Use the `horizontal` prop to make the ListGroup render horizontally. Currently **horizontal list groups cannot be combined with flush list groups.**

RESULT

This ListGroup renders horizontally!

LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function HorizontalExample() {
  return (
    <ListGroup horizontal>
      <ListGroup.Item>This</ListGroup.Item>
      <ListGroup.Item>ListGroup</ListGroup.Item>
      <ListGroup.Item>renders</ListGroup.Item>
      <ListGroup.Item>horizontally!</ListGroup.Item>
    </ListGroup>
  );
}

export default HorizontalExample;
```

There are responsive variants to `horizontal`: setting it to `{sm|md|lg|xl|xxl}` makes the list group horizontal starting at that breakpoint's `min-width`.

RESULT

This ListGroup	renders horizontally	on sm	and above!

This ListGroup	renders horizontally	on md	and above!

This ListGroup
renders horizontally
on lg

and above!

This ListGroup

renders horizontally

on xl

and above!

This ListGroup

renders horizontally

on xxl

and above!

LIVE EDITOR

```
import ListGroup from 'react-bootstrap/ListGroup';

function HorizontalResponsiveExample() {
  return (
    <>
      {['sm', 'md', 'lg', 'xl', 'xxl'].map((breakpoint) => (
        <ListGroup key={breakpoint} horizontal={breakpoint}
className="my-2">
          <ListGroup.Item>This ListGroup</ListGroup.Item>
          <ListGroup.Item>renders horizontally</ListGroup.Item>
          <ListGroup.Item>on {breakpoint}</ListGroup.Item>
          <ListGroup.Item>and above!</ListGroup.Item>
        </ListGroup>
      ))}
    </>
  );
}

export default HorizontalResponsiveExample;
```

Contextual classes

Use contextual variants on `<ListGroup.Item>`s to style them with a stateful background and color.

RESULT

No style

Primary

Secondary

Success

Danger

Warning

Info

Light

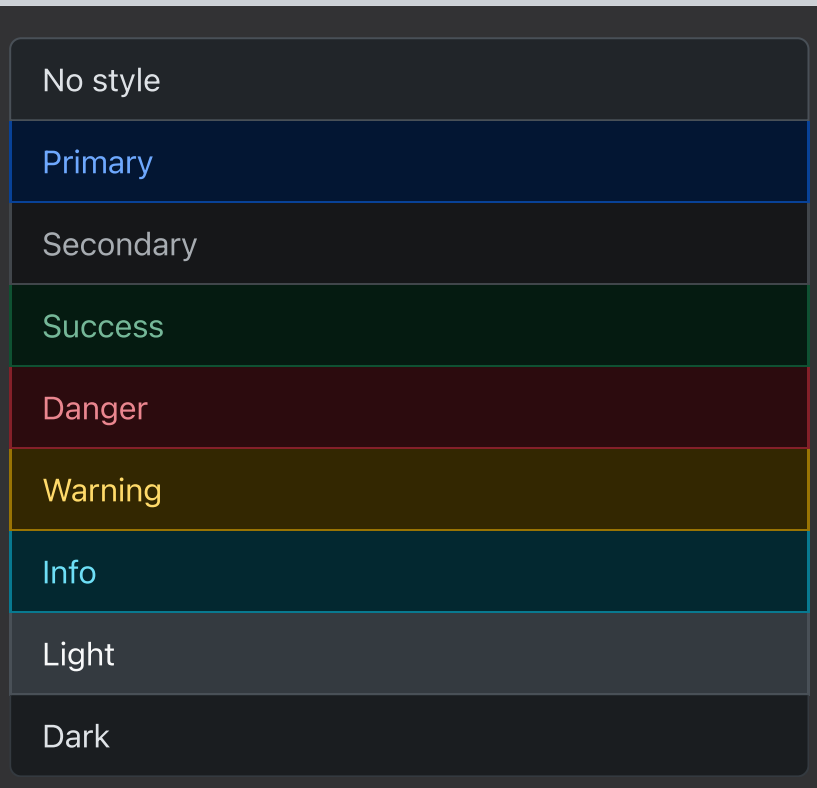
Dark

LIVE EDITOR

```jsx
import ListGroup from 'react-bootstrap/ListGroup';

function StyleExample() {
  return (
    <ListGroup>
      <ListGroup.Item>No style</ListGroup.Item>
      <ListGroup.Item variant="primary">Primary</ListGroup.Item>
      <ListGroup.Item variant="secondary">Secondary</ListGroup.Item>
      <ListGroup.Item variant="success">Success</ListGroup.Item>
      <ListGroup.Item variant="danger">Danger</ListGroup.Item>
      <ListGroup.Item variant="warning">Warning</ListGroup.Item>
      <ListGroup.Item variant="info">Info</ListGroup.Item>
      <ListGroup.Item variant="light">Light</ListGroup.Item>
      <ListGroup.Item variant="dark">Dark</ListGroup.Item>
    </ListGroup>
  );
}

export default StyleExample;
```

When paired with `action`s, additional hover and active styles apply.

RESULT



LIVE EDITOR

```jsx
import ListGroup from 'react-bootstrap/ListGroup';

function StyleActionsExample() {
  return (
    <ListGroup>
      <ListGroup.Item>No style</ListGroup.Item>
      <ListGroup.Item variant="primary">Primary</ListGroup.Item>
      <ListGroup.Item action variant="secondary">
        Secondary
      </ListGroup.Item>
      <ListGroup.Item action variant="success">
        Success
      </ListGroup.Item>
      <ListGroup.Item action variant="danger">
        Danger
      </ListGroup.Item>
      <ListGroup.Item action variant="warning">
        Warning
      </ListGroup.Item>
      <ListGroup.Item action variant="info">
        Info
      </ListGroup.Item>
      <ListGroup.Item action variant="light">
        Light
      </ListGroup.Item>
      <ListGroup.Item action variant="dark">
```

```
        Dark
      </ListGroup.Item>
    </ListGroup>
  );
}

export default StyleActionsExample;
```

Conveying meaning to assistive technologies

Using color to add meaning only provides a visual indication, which will not be conveyed to users of assistive technologies – such as screen readers. Ensure that information denoted by the color is either obvious from the content itself (e.g. the visible text), or is included through alternative means, such as additional text hidden with the `.visually-hidden` class.

Tabbed Interfaces

You can also use the [Tab][tabs] components to create ARIA compliant tabbable interfaces with the `<ListGroup>` component. Swap out the `<Nav>` component for the list group and you are good to go.

RESULT

Link 1
Link 2

Tab pane content 1

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import ListGroup from 'react-bootstrap/ListGroup';
import Row from 'react-bootstrap/Row';
import Tab from 'react-bootstrap/Tab';

function TabsExample() {
  return (
    <Tab.Container id="list-group-tabs-example"
defaultActiveKey="#link1">
      <Row>
        <Col sm={4}>
          <ListGroup>
            <ListGroup.Item action href="#link1">
              Link 1
            </ListGroup.Item>
            <ListGroup.Item action href="#link2">
              Link 2
            </ListGroup.Item>
          </ListGroup>
        </Col>
        <Col sm={8}>
          <Tab.Content>
            <Tab.Pane eventKey="#link1">Tab pane content 1</Tab.Pane>
            <Tab.Pane eventKey="#link2">Tab pane content 2</Tab.Pane>
          </Tab.Content>
        </Col>
      </Row>
    </Tab.Container>
  );
}

export default TabsExample;
```

API

ListGroup

```
import ListGroup from 'react-bootstrap/ListGroup'
```

Name	Type	Default	Description
bsPrefix	string	`'list-group'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'flush'`		Adds a variant to the list-group
horizontal	`true` | `'sm'` | `'md'` | `'lg'` | `'xl'` | `'xxl'`		Changes the flow of the list group items from vertical to horizontal. A value of `null` (the default) sets it to vertical for all breakpoints; Just including the prop sets it for all breakpoints, while `{sm|md|lg|xl|xxl}` makes the list group horizontal starting at that breakpoint's `min-width`.
numbered	bool		Generate numbered list items.
as	elementType		You can use a custom element type for this component.

ListGroupItem

```
import ListGroupItem from 'react-bootstrap/ListGroupItem'
```

Name	Type	Default	Description
bsPrefix	string	`'list-group-item'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'dark'` | `'light'`		Sets contextual classes for list item.
action	bool		Marks a ListGroupItem as actionable, applying additional hover, active and disabled styles for links and buttons.
active	bool		Sets list item as active.
disabled	bool		Sets list item state as disabled.
eventKey	`string` | `number`		A unique identifier for the Component, the `eventKey` makes it distinguishable from others in a set. Similar to React's `key` prop, in that it only needs to be unique amongst the Components siblings, not globally.
onClick	func		A callback function for when this component is clicked.
href	string		Providing a `href` and setting `action` to `true`, it will render the

Name	Type	Default	Description
			ListGroup.Item as an `<a>` element (unless `as` is provided).
as	elementType	`{'div' | 'a' | 'button'}`	You can use a custom element type for this component. For none `action` items, items render as `li`. For actions the default is an anchor or button element depending on whether a `href` is provided.

Modals

Add dialogs to your site for lightboxes, user notifications, or completely custom content.

Overview

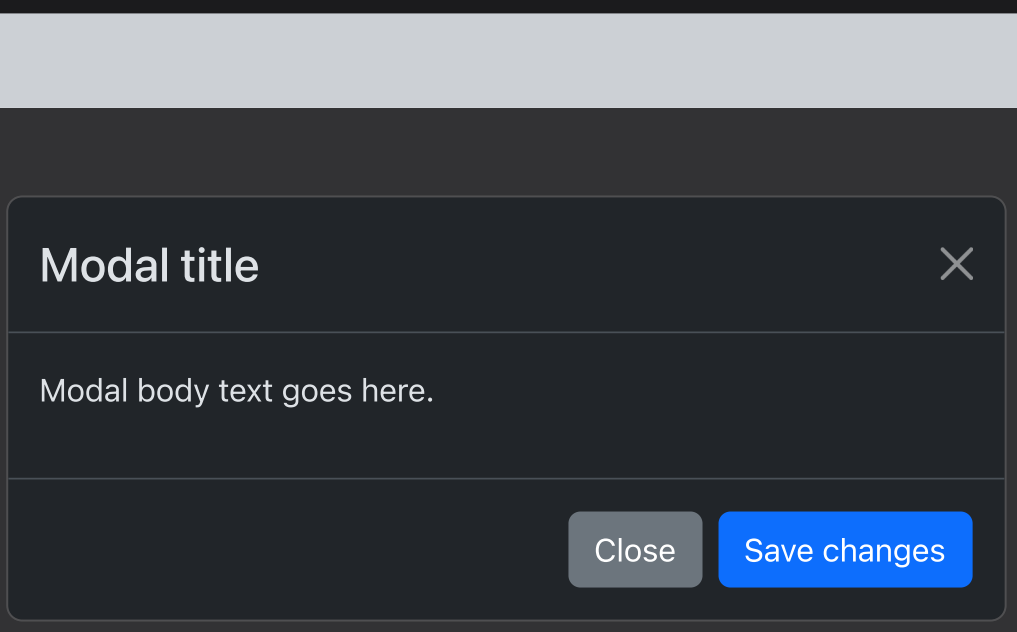
- Modals are positioned over everything else in the document and remove scroll from the `<body>` so that modal content scrolls instead.
- Modals are *unmounted* when closed.
- Bootstrap only supports **one** modal window at a time. Nested modals aren't supported, but if you really need them, the underlying `@restart/ui` library can support them if you're willing.
- Modal's "trap" focus in them, ensuring the keyboard navigation cycles through the modal, and not the rest of the page.
- Unlike vanilla Bootstrap, `autoFocus` works in Modals because React handles the implementation.

Examples

Static Markup

Below is a *static* modal dialog (without the positioning) to demonstrate the look and feel of the Modal.

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function StaticExample() {
  return (
    <div
      className="modal show"
      style={{ display: 'block', position: 'initial' }}
    >
```

```
      <Modal.Dialog>
        <Modal.Header closeButton>
          <Modal.Title>Modal title</Modal.Title>
        </Modal.Header>

        <Modal.Body>
          <p>Modal body text goes here.</p>
        </Modal.Body>

        <Modal.Footer>
          <Button variant="secondary">Close</Button>
          <Button variant="primary">Save changes</Button>
        </Modal.Footer>
      </Modal.Dialog>
    </div>
  );
}

export default StaticExample;
```

Live demo

A modal with header, body, and set of actions in the footer. Use `<Modal/>` in combination with other components to show or hide your Modal. The `<Modal/>` Component comes with a few convenient "sub components": `<Modal.Header/>`, `<Modal.Title/>`, `<Modal.Body/>`, and `<Modal.Footer/>`, which you can use to build the Modal content.

RESULT

Launch demo modal

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Launch demo modal
      </Button>

      <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
          <Modal.Title>Modal heading</Modal.Title>
        </Modal.Header>
        <Modal.Body>Woohoo, you are reading this text in a modal!
</Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Close
          </Button>
          <Button variant="primary" onClick={handleClose}>
            Save Changes
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
}
```

```
export default Example;
```

Static backdrop

When backdrop is set to static, the modal will not close when clicking outside it. Click the button below to try it.

RESULT

Launch static backdrop modal

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Launch static backdrop modal
      </Button>

      <Modal
        show={show}
        onHide={handleClose}
        backdrop="static"
        keyboard={false}
      >
        <Modal.Header closeButton>
          <Modal.Title>Modal title</Modal.Title>
        </Modal.Header>
        <Modal.Body>
          I will not close if you click outside me. Do not even try to
press
          escape key.
        </Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Close
          </Button>
          <Button variant="primary">Understood</Button>
        </Modal.Footer>
      </Modal>
    </>
  );
}

export default Example;
```

Without Animation

A Modal can also be without an animation. For that set the `animation` prop to `false`.

RESULT

Launch demo modal

LIVE EDITOR

```jsx
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Launch demo modal
      </Button>

      <Modal show={show} onHide={handleClose} animation={false}>
        <Modal.Header closeButton>
          <Modal.Title>Modal heading</Modal.Title>
        </Modal.Header>
        <Modal.Body>Woohoo, you are reading this text in a modal!
</Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Close
          </Button>
          <Button variant="primary" onClick={handleClose}>
            Save Changes
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
}

export default Example;
```

Additional Import Options

The Modal Header, Title, Body, and Footer components are available as static properties the `<Modal/>` component, but you can also, import them directly like: `require("react-bootstrap/ModalHeader")`.

Vertically centered

You can vertically center a modal by passing the `centered` prop.

RESULT

Launch vertically centered modal

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function MyVerticallyCenteredModal(props) {
  return (
    <Modal
      {...props}
      size="lg"
      aria-labelledby="contained-modal-title-vcenter"
      centered
    >
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">
```

```
          Modal heading
        </Modal.Title>
      </Modal.Header>
      <Modal.Body>
        <h4>Centered Modal</h4>
        <p>
          Cras mattis consectetur purus sit amet fermentum. Cras justo
odio,
          dapibus ac facilisis in, egestas eget quam. Morbi leo risus,
porta ac
          consectetur ac, vestibulum at eros.
        </p>
      </Modal.Body>
      <Modal.Footer>
        <Button onClick={props.onHide}>Close</Button>
      </Modal.Footer>
    </Modal>
  );
}

function App() {
  const [modalShow, setModalShow] = React.useState(false);

  return (
    <>
      <Button variant="primary" onClick={() => setModalShow(true)}>
        Launch vertically centered modal
      </Button>

      <MyVerticallyCenteredModal
        show={modalShow}
        onHide={() => setModalShow(false)}
      />
    </>
  );
}

render(<App />);
```

Using the grid

You can use grid layouts within a model using regular grid components inside the modal content.

RESULT

Launch modal with grid

LIVE EDITOR

```
import React, { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Container from 'react-bootstrap/Container';
import Modal from 'react-bootstrap/Modal';
import Row from 'react-bootstrap/Row';

function MydModalWithGrid(props) {
  return (
    <Modal {...props} aria-labelledby="contained-modal-title-vcenter">
      <Modal.Header closeButton>
        <Modal.Title id="contained-modal-title-vcenter">
          Using Grid in Modal
        </Modal.Title>
      </Modal.Header>
      <Modal.Body className="grid-example">
        <Container>
          <Row>
```

```
          <Col xs={12} md={8}>
            .col-xs-12 .col-md-8
          </Col>
          <Col xs={6} md={4}>
            .col-xs-6 .col-md-4
          </Col>
        </Row>

        <Row>
          <Col xs={6} md={4}>
            .col-xs-6 .col-md-4
          </Col>
          <Col xs={6} md={4}>
            .col-xs-6 .col-md-4
          </Col>
          <Col xs={6} md={4}>
            .col-xs-6 .col-md-4
          </Col>
        </Row>
      </Container>
    </Modal.Body>
    <Modal.Footer>
      <Button onClick={props.onHide}>Close</Button>
    </Modal.Footer>
  </Modal>
  );
}

function App() {
  const [modalShow, setModalShow] = useState(false);

  return (
    <>
      <Button variant="primary" onClick={() => setModalShow(true)}>
        Launch modal with grid
      </Button>

      <MydModalWithGrid show={modalShow} onHide={() =>
setModalShow(false)} />
    </>
  );
}

render(<App />);
```

Focus on specific element

You can focus on an element inside the modal using `autoFocus` attribute on the element.

RESULT

Launch demo modal

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Form from 'react-bootstrap/Form';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
```

```
      <Button variant="primary" onClick={handleShow}>
        Launch demo modal
      </Button>

      <Modal show={show} onHide={handleClose}>
        <Modal.Header closeButton>
          <Modal.Title>Modal heading</Modal.Title>
        </Modal.Header>
        <Modal.Body>
          <Form>
            <Form.Group className="mb-3"
controlId="exampleForm.ControlInput1">
              <Form.Label>Email address</Form.Label>
              <Form.Control
                type="email"
                placeholder="name@example.com"
                autoFocus
              />
            </Form.Group>
            <Form.Group
              className="mb-3"
              controlId="exampleForm.ControlTextarea1"
            >
              <Form.Label>Example textarea</Form.Label>
              <Form.Control as="textarea" rows={3} />
            </Form.Group>
          </Form>
        </Modal.Body>
        <Modal.Footer>
          <Button variant="secondary" onClick={handleClose}>
            Close
          </Button>
          <Button variant="primary" onClick={handleClose}>
            Save Changes
          </Button>
        </Modal.Footer>
      </Modal>
    </>
  );
}

export default Example;
```

Optional Sizes

You can specify a Bootstrap large or small modal by using the `size` prop.

RESULT

Small modal Large modal

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [smShow, setSmShow] = useState(false);
  const [lgShow, setLgShow] = useState(false);

  return (
    <>
      <Button onClick={() => setSmShow(true)} className="me-2">
        Small modal
      </Button>
      <Button onClick={() => setLgShow(true)}>Large modal</Button>
      <Modal
```

```
        size="sm"
        show={smShow}
        onHide={() => setSmShow(false)}
        aria-labelledby="example-modal-sizes-title-sm"
      >
        <Modal.Header closeButton>
          <Modal.Title id="example-modal-sizes-title-sm">
            Small Modal
          </Modal.Title>
        </Modal.Header>
        <Modal.Body>...</Modal.Body>
      </Modal>
      <Modal
        size="lg"
        show={lgShow}
        onHide={() => setLgShow(false)}
        aria-labelledby="example-modal-sizes-title-lg"
      >
        <Modal.Header closeButton>
          <Modal.Title id="example-modal-sizes-title-lg">
            Large Modal
          </Modal.Title>
        </Modal.Header>
        <Modal.Body>...</Modal.Body>
      </Modal>
    </>
  );
}

export default Example;
```

Fullscreen Modal

You can use the `fullscreen` prop to make the modal fullscreen. Specifying a breakpoint will only set the modal as fullscreen **below** the breakpoint size.

RESULT

Full screen | Full screen below sm | Full screen below md | Full screen below lg | Full screen below xl | Full screen below xxl

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const values = [true, 'sm-down', 'md-down', 'lg-down', 'xl-down', 'xxl-down'];
  const [fullscreen, setFullscreen] = useState(true);
  const [show, setShow] = useState(false);

  function handleShow(breakpoint) {
    setFullscreen(breakpoint);
    setShow(true);
  }

  return (
    <>
      {values.map((v, idx) => (
        <Button key={idx} className="me-2 mb-2" onClick={() => handleShow(v)}>
          Full screen
          {typeof v === 'string' && `below ${v.split('-')[0]}`}
        </Button>
      ))}
```

```
      <Modal show={show} fullscreen={fullscreen} onHide={() =>
setShow(false)}>
        <Modal.Header closeButton>
          <Modal.Title>Modal</Modal.Title>
        </Modal.Header>
        <Modal.Body>Modal body content</Modal.Body>
      </Modal>
    </>
  );
}

export default Example;
```

Sizing modals using custom CSS

You can apply custom css to the modal dialog div using the `dialogClassName` prop. Example is using a custom css class with width set to 90%.

RESULT

Custom Width Modal

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Modal from 'react-bootstrap/Modal';

function Example() {
  const [show, setShow] = useState(false);

  return (
    <>
      <Button variant="primary" onClick={() => setShow(true)}>
        Custom Width Modal
      </Button>

      <Modal
        show={show}
        onHide={() => setShow(false)}
        dialogClassName="modal-90w"
        aria-labelledby="example-custom-modal-styling-title"
      >
        <Modal.Header closeButton>
          <Modal.Title id="example-custom-modal-styling-title">
            Custom Modal Styling
          </Modal.Title>
        </Modal.Header>
        <Modal.Body>
          <p>
            Ipsum molestiae natus adipisci modi eligendi? Debitis amet
quae unde
            commodi aspernatur enim, consectetur. Cumque deleniti
temporibus
            ipsam atque a dolores quisquam quisquam adipisci possimus
            laboriosam. Quibusdam facilis doloribus debitis! Sit quasi
quod
            accusamus eos quod. Ab quos consequuntur eaque quo rem!
Mollitia
            reiciendis porro quo magni incidunt dolore amet atque facilis
ipsum
            deleniti rem!
          </p>
        </Modal.Body>
      </Modal>
    </>
  );
}
```

```
export default Example;
```

API

Modal

```
import Modal from 'react-bootstrap/Modal'
```

Name	Type	Default	Description
bsPrefix	string	'modal'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
size	'sm' | 'lg' | 'xl'		Render a large, extra large or small modal. When not provided, the modal is rendered with medium (default) size.
fullscreen	true | 'sm-down' | 'md-down' | 'lg-down' | 'xl-down' | 'xxl-down'		Renders a fullscreen modal. Specifying a breakpoint will render the modal as fullscreen **below** the breakpoint size.
centered	bool		vertically center the Dialog in the window
backdrop	'static' | true | false	true	Include a backdrop component. Specify 'static' for a backdrop that doesn't trigger an "onHide" when clicked.
backdropClassName	string		Add an optional extra class name to .modal-backdrop It could end up looking like class="modal-backdrop foo-modal-backdrop in".
keyboard	bool	true	Close the modal when escape key is pressed
scrollable	bool		Allows scrolling the <Modal.Body> instead of the entire Modal when overflowing.
animation	bool	true	Open and close the Modal with a slide and fade animation.
dialogClassName	string		A css class to apply to the Modal dialog DOM node.

Name	Type	Default	Description
`contentClassName`	`string`		Add an optional extra class name to .modal-content
`dialogAs`	`elementType`	`ModalDialog`	A Component type that provides the modal content Markup. This is a useful prop when you want to use your own styles and markup to create a custom modal component.
`autoFocus`	`bool`	`true`	When `true` The modal will automatically shift focus to itself when it opens, and replace it to the last focused element when it closes. Generally this should never be set to false as it makes the Modal less accessible to assistive technologies, like screen-readers.
`enforceFocus`	`bool`	`true`	When `true` The modal will prevent focus from leaving the Modal while open. Consider leaving the default value here, as it is necessary to make the Modal work well with assistive technologies, such as screen readers.
`restoreFocus`	`bool`	`true`	When `true` The modal will restore focus to previously focused element once modal is hidden
`restoreFocusOptions`	`shape`		Options passed to focus function when `restoreFocus` is set to `true`
`show`	`bool`	`false`	When `true` The modal will show itself.
`onShow`	`func`		A callback fired when the Modal is opening.
`onHide`	`func`		A callback fired when the header closeButton or non-static backdrop is clicked. Required if either are specified.
`onEscapeKeyDown`	`func`		A callback fired when the escape key, if specified in `keyboard`, is pressed.
`onEnter`	`func`		Callback fired before the Modal transitions in
`onEntering`	`func`		Callback fired as the Modal begins to transition in
`onEntered`	`func`		Callback fired after the Modal finishes transitioning

Name	Type	Default	Description
			in
`onExit`	`func`		Callback fired right before the Modal transitions out
`onExiting`	`func`		Callback fired as the Modal begins to transition out
`onExited`	`func`		Callback fired after the Modal finishes transitioning out
`manager`	`object`		A ModalManager instance used to track and manage the state of open Modals. Useful when customizing how modals interact within a container
`data-bs-theme`	`string`		
`aria-labelledby`	`string`		
`aria-describedby`	`string`		
`aria-label`	`string`		

ModalDialog

```
import ModalDialog from 'react-bootstrap/ModalDialog'
```

Name	Type	Default	Description
`bsPrefix`	`string`	`'modal'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
`contentClassName`	`string`		
`size`	`'sm'` | `'lg'` | `'xl'`		Render a large, extra large or small modal.
`fullscreen`	`true` | `'sm-down'` | `'md-down'` | `'lg-down'` | `'xl-down'` | `'xxl-down'`		Renders a fullscreen modal. Specifying a breakpoint will render the modal as fullscreen **below** the breakpoint size.
`centered`	`bool`		Specify whether the Component should be vertically centered
`scrollable`	`bool`		Allows scrolling the `<Modal.Body>` instead of the entire Modal when overflowing.

ModalHeader

```
import ModalHeader from 'react-bootstrap/ModalHeader'
```

Name	Type	Default	Description
bsPrefix	string	`'modal-header'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
closeLabel	string	`'Close'`	Provides an accessible label for the close button. It is used for Assistive Technology when the label text is not readable.
closeVariant	`'white'`		Sets the variant for close button.
closeButton	bool	`false`	Specify whether the Component should contain a close button
onHide	func		A Callback fired when the close button is clicked. If used directly inside a Modal component, the onHide will automatically be propagated up to the parent Modal `onHide`.

ModalTitle

```
import ModalTitle from 'react-bootstrap/ModalTitle'
```

Name	Type	Default	Description
as		`divWithClassName('h4')`	You can use a custom element type for this component.

ModalBody

```
import ModalBody from 'react-bootstrap/ModalBody'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

ModalFooter

```
import ModalFooter from 'react-bootstrap/ModalFooter'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

Navbars

A powerful, responsive navigation header, the navbar. Includes support for branding, navigation, and more.

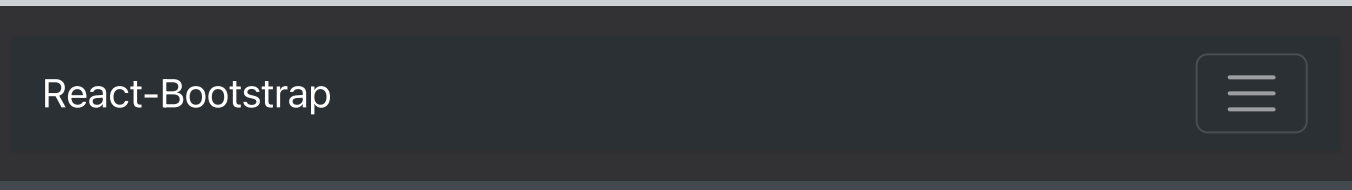
Overview

Here's what you need to know before getting started with the Navbar:

- Use the `expand` prop to allow for collapsing the `Navbar` at lower breakpoints.
- `Navbar`s and their contents are fluid by default. Use optional containers to limit their horizontal width.
- Use spacing and flex utilities to size and position content

A responsive navigation header, including support for branding, navigation, and more. Here's an example of all the sub-components included in a responsive light-themed navbar that automatically collapses at the lg (large) breakpoint.

RESULT



LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';

function BasicExample() {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#link">Link</Nav.Link>
            <NavDropdown title="Dropdown" id="basic-nav-dropdown">
              <NavDropdown.Item
href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Item
href="#action/3.3">Something</NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action/3.4">
                Separated link
              </NavDropdown.Item>
            </NavDropdown>
          </Nav>
```

```
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default BasicExample;
```

Brand

A simple flexible branding component. Images are supported but will likely require custom styling to work well.

RESULT

Brand link

Brand text

React Bootstrap

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Navbar from 'react-bootstrap/Navbar';

function BrandExample() {
  return (
    <>
      <Navbar className="bg-body-tertiary">
        <Container>
          <Navbar.Brand href="#home">Brand link</Navbar.Brand>
        </Container>
      </Navbar>
      <br />
      <Navbar className="bg-body-tertiary">
        <Container>
          <Navbar.Brand>Brand text</Navbar.Brand>
        </Container>
      </Navbar>
      <br />
      <Navbar className="bg-body-tertiary">
        <Container>
          <Navbar.Brand href="#home">
            <img
              src="/img/logo.svg"
              width="30"
              height="30"
              className="d-inline-block align-top"
              alt="React Bootstrap logo"
            />
          </Navbar.Brand>
        </Container>
      </Navbar>
      <br />
      <Navbar className="bg-body-tertiary">
        <Container>
          <Navbar.Brand href="#home">
            <img
              alt=""
```

```
            src="/img/logo.svg"
            width="30"
            height="30"
            className="d-inline-block align-top"
          />{' '}
          React Bootstrap
        </Navbar.Brand>
      </Container>
    </Navbar>
  </>
);
}

export default BrandExample;
```

Forms

Use `<Form inline>` and your various form controls within the Navbar. Align the contents as needed with utility classes.

RESULT

@ Username Search Submit

LIVE EDITOR

```
import Navbar from 'react-bootstrap/Navbar';
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button';
import InputGroup from 'react-bootstrap/InputGroup';
import Row from 'react-bootstrap/Row';
import Col from 'react-bootstrap/Col';

function FormExample() {
  return (
    <Navbar className="bg-body-tertiary justify-content-between">
      <Form inline>
        <InputGroup>
          <InputGroup.Text id="basic-addon1">@</InputGroup.Text>
          <Form.Control
            placeholder="Username"
            aria-label="Username"
            aria-describedby="basic-addon1"
          />
        </InputGroup>
      </Form>
      <Form inline>
        <Row>
          <Col xs="auto">
            <Form.Control
              type="text"
              placeholder="Search"
              className=" mr-sm-2"
            />
          </Col>
          <Col xs="auto">
            <Button type="submit">Submit</Button>
          </Col>
        </Row>
      </Form>
    </Navbar>
  );
}

export default FormExample;
```

Text and Non-nav links

Loose text and links can be wrapped `Navbar.Text` in order to correctly align it vertically.

RESULT

LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Navbar from 'react-bootstrap/Navbar';

function TextLinkExample() {
  return (
    <Navbar className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">Navbar with text</Navbar.Brand>
        <Navbar.Toggle />
        <Navbar.Collapse className="justify-content-end">
          <Navbar.Text>
            Signed in as: <a href="#login">Mark Otto</a>
          </Navbar.Text>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default TextLinkExample;
```

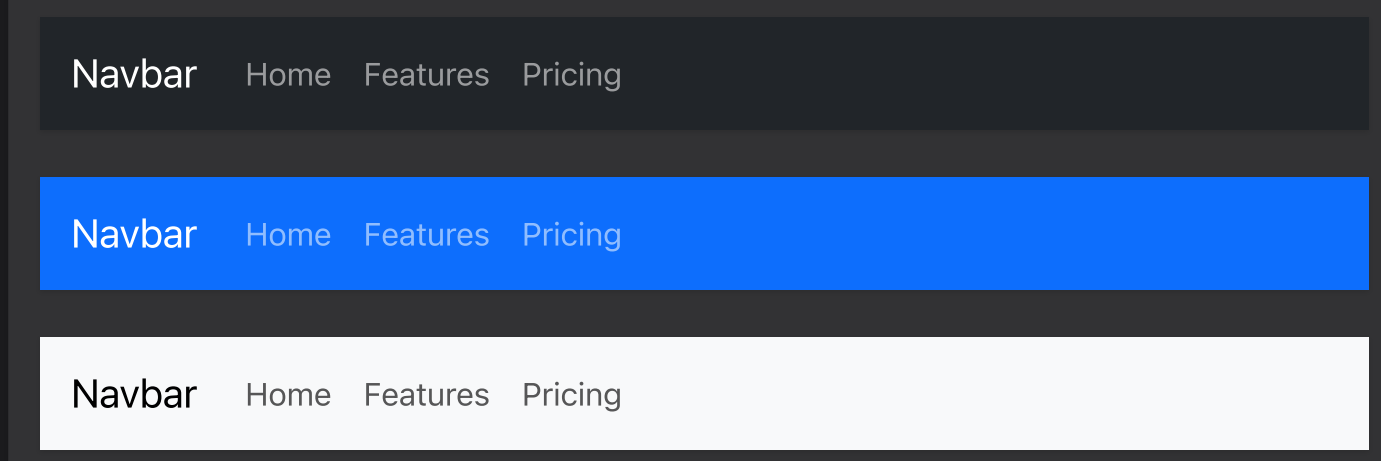
Color schemes

Theming the navbar has never been easier thanks to the combination of theming classes and background-color utilities. Choose from `variant="light"` for use with light background colors, or `variant="dark"` for dark background colors. Then, customize with the `bg` prop or any custom css!

Heads up!

Dark variants for components were deprecated in Bootstrap v5.3.0 with the introduction of color modes. Instead of adding `variant="dark"`, set `data-bs-theme="dark"` on the `Navbar`.

RESULT



LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
```

```jsx
function ColorSchemesExample() {
  return (
    <>
      <Navbar bg="dark" data-bs-theme="dark">
        <Container>
          <Navbar.Brand href="#home">Navbar</Navbar.Brand>
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#features">Features</Nav.Link>
            <Nav.Link href="#pricing">Pricing</Nav.Link>
          </Nav>
        </Container>
      </Navbar>
      <br />
      <Navbar bg="primary" data-bs-theme="dark">
        <Container>
          <Navbar.Brand href="#home">Navbar</Navbar.Brand>
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#features">Features</Nav.Link>
            <Nav.Link href="#pricing">Pricing</Nav.Link>
          </Nav>
        </Container>
      </Navbar>
      <br />
      <Navbar bg="light" data-bs-theme="light">
        <Container>
          <Navbar.Brand href="#home">Navbar</Navbar.Brand>
          <Nav className="me-auto">
            <Nav.Link href="#home">Home</Nav.Link>
            <Nav.Link href="#features">Features</Nav.Link>
            <Nav.Link href="#pricing">Pricing</Nav.Link>
          </Nav>
        </Container>
      </Navbar>
    </>
  );
}

export default ColorSchemesExample;
```

Containers

While not required, you can wrap the Navbar in a `<Container>` component to center it on a page, or add one within to only center the contents of a fixed or static top navbar.

RESULT

Navbar

LIVE EDITOR

```jsx
import Container from 'react-bootstrap/Container';
import Navbar from 'react-bootstrap/Navbar';

function ContainerOutsideExample() {
  return (
    <Container>
      <Navbar expand="lg" className="bg-body-tertiary">
        <Container>
          <Navbar.Brand href="#">Navbar</Navbar.Brand>
        </Container>
      </Navbar>
    </Container>
  );
}
```

```
export default ContainerOutsideExample;
```

When the container is within your navbar, its horizontal padding is removed at breakpoints lower than your specified `expand={'sm' | 'md' | 'lg' | 'xl' | 'xxl'}` prop. This ensures we're not doubling up on padding unnecessarily on lower viewports when your navbar is collapsed.

RESULT

Navbar

LIVE EDITOR

```
import Container from 'react-bootstrap/Container';
import Navbar from 'react-bootstrap/Navbar';

function ContainerInsideExample() {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#">Navbar</Navbar.Brand>
      </Container>
    </Navbar>
  );
}

export default ContainerInsideExample;
```

Placement

You can use Bootstrap's position utilities to place navbars in non-static positions. Choose from fixed to the top, fixed to the bottom, or stickied to the top (scrolls with the page until it reaches the top, then stays there), or stickied to the bottom (scrolls with the page until it reaches the bottom, then stays there).

Fixed navbars use `position: fixed`, meaning they're pulled from the normal flow of the DOM and may require custom CSS (e.g., padding-top on the `<body>`) to prevent overlap with other elements.

Since these positioning needs are so common for navbars, we've added convenience props for them.

Fixed top

```
<Navbar fixed="top" />
```

Fixed bottom

```
<Navbar fixed="bottom" />
```

Sticky top

```
<Navbar sticky="top" />
```

Sticky bottom

```
<Navbar sticky="bottom" />
```

Scrolling

You can use the `navbarScroll` prop in a `<Nav>` to enable vertical scrolling within the toggleable contents of a collapsed navbar. See the Bootstrap docs for more information.

RESULT

Navbar scroll

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Container from 'react-bootstrap/Container';
import Form from 'react-bootstrap/Form';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';

function NavScrollExample() {
  return (
    <Navbar expand="lg" className="bg-body-tertiary">
      <Container fluid>
        <Navbar.Brand href="#">Navbar scroll</Navbar.Brand>
        <Navbar.Toggle aria-controls="navbarScroll" />
        <Navbar.Collapse id="navbarScroll">
          <Nav
            className="me-auto my-2 my-lg-0"
            style={{ maxHeight: '100px' }}
            navbarScroll
          >
            <Nav.Link href="#action1">Home</Nav.Link>
            <Nav.Link href="#action2">Link</Nav.Link>
            <NavDropdown title="Link" id="navbarScrollingDropdown">
              <NavDropdown.Item href="#action3">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action4">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action5">
                Something else here
              </NavDropdown.Item>
            </NavDropdown>
            <Nav.Link href="#" disabled>
              Link
            </Nav.Link>
          </Nav>
          <Form className="d-flex">
            <Form.Control
              type="search"
              placeholder="Search"
              className="me-2"
              aria-label="Search"
            />
            <Button variant="outline-success">Search</Button>
          </Form>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default NavScrollExample;
```

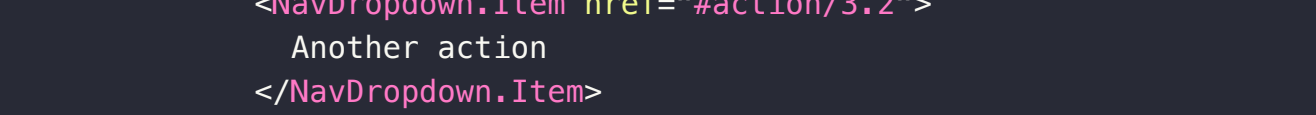
Responsive behaviors

Use the `expand` prop as well as the `Navbar.Toggle` and `Navbar.Collapse` components to control when content collapses behind a button.

Set the `defaultExpanded` prop to make the Navbar start expanded. Set `collapseOnSelect` to make the Navbar collapse automatically when the user selects an item. You can also finely control the collapsing behavior by using the `expanded` and `onToggle` props.

> **Caution**
>
> Watch out! You **need** to provide a breakpoint value to `expand` in order for the Navbar to collapse at all.

RESULT



LIVE EDITOR

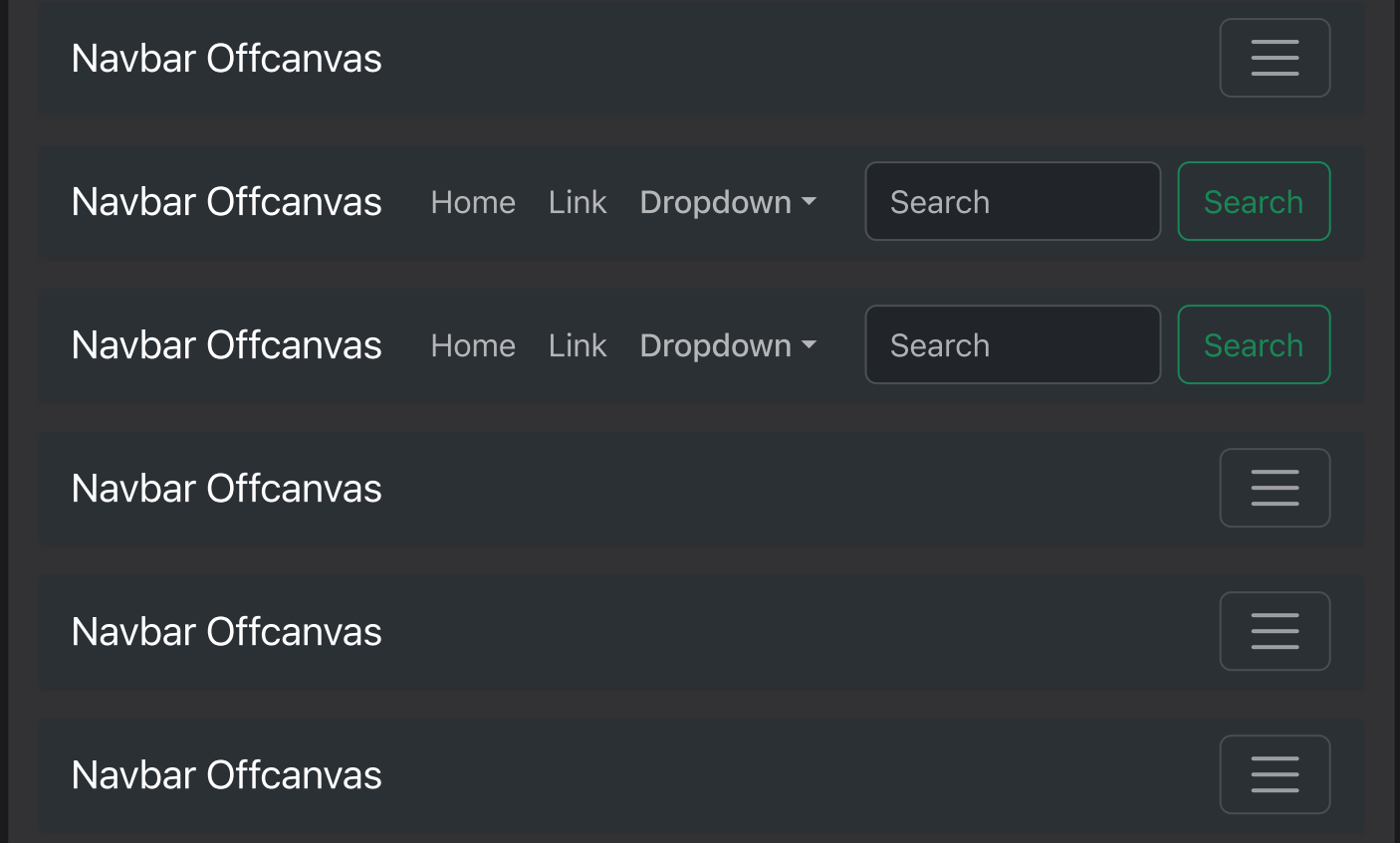
```jsx
import Container from 'react-bootstrap/Container';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';

function CollapsibleExample() {
  return (
    <Navbar collapseOnSelect expand="lg" className="bg-body-tertiary">
      <Container>
        <Navbar.Brand href="#home">React-Bootstrap</Navbar.Brand>
        <Navbar.Toggle aria-controls="responsive-navbar-nav" />
        <Navbar.Collapse id="responsive-navbar-nav">
          <Nav className="me-auto">
            <Nav.Link href="#features">Features</Nav.Link>
            <Nav.Link href="#pricing">Pricing</Nav.Link>
            <NavDropdown title="Dropdown" id="collapsible-nav-dropdown">
              <NavDropdown.Item
href="#action/3.1">Action</NavDropdown.Item>
              <NavDropdown.Item href="#action/3.2">
                Another action
              </NavDropdown.Item>
              <NavDropdown.Item
href="#action/3.3">Something</NavDropdown.Item>
              <NavDropdown.Divider />
              <NavDropdown.Item href="#action/3.4">
                Separated link
              </NavDropdown.Item>
            </NavDropdown>
          </Nav>
          <Nav>
            <Nav.Link href="#deets">More deets</Nav.Link>
            <Nav.Link eventKey={2} href="#memes">
              Dank memes
            </Nav.Link>
          </Nav>
        </Navbar.Collapse>
      </Container>
    </Navbar>
  );
}

export default CollapsibleExample;
```

Offcanvas

Transform your expanding and collapsing navbar into an offcanvas drawer with the offcanvas component. We extend both the offcanvas default styles and use the `expand` prop to create a dynamic and flexible navigation sidebar.

In the example below, to create an offcanvas navbar that is always collapsed across all breakpoints, set the `expand` prop to `false`.

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Container from 'react-bootstrap/Container';
import Form from 'react-bootstrap/Form';
import Nav from 'react-bootstrap/Nav';
import Navbar from 'react-bootstrap/Navbar';
import NavDropdown from 'react-bootstrap/NavDropdown';
import Offcanvas from 'react-bootstrap/Offcanvas';

function OffcanvasExample() {
  return (
    <>
      {[false, 'sm', 'md', 'lg', 'xl', 'xxl'].map((expand) => (
        <Navbar key={expand} expand={expand} className="bg-body-tertiary
mb-3">
          <Container fluid>
            <Navbar.Brand href="#">Navbar Offcanvas</Navbar.Brand>
            <Navbar.Toggle aria-controls={`offcanvasNavbar-
expand-${expand}`} />
            <Navbar.Offcanvas
              id={`offcanvasNavbar-expand-${expand}`}
              aria-labelledby={`offcanvasNavbarLabel-expand-${expand}`}
              placement="end"
            >
              <Offcanvas.Header closeButton>
                <Offcanvas.Title id={`offcanvasNavbarLabel-
expand-${expand}`}>
                  Offcanvas
                </Offcanvas.Title>
              </Offcanvas.Header>
              <Offcanvas.Body>
                <Nav className="justify-content-end flex-grow-1 pe-3">
                  <Nav.Link href="#action1">Home</Nav.Link>
                  <Nav.Link href="#action2">Link</Nav.Link>
                  <NavDropdown
                    title="Dropdown"
                    id={`offcanvasNavbarDropdown-expand-${expand}`}
                  >
                    <NavDropdown.Item
href="#action3">Action</NavDropdown.Item>
                    <NavDropdown.Item href="#action4">
                      Another action
                    </NavDropdown.Item>
                    <NavDropdown.Divider />
```

```
                  <NavDropdown.Item href="#action5">
                    Something else here
                  </NavDropdown.Item>
                </NavDropdown>
              </Nav>
              <Form className="d-flex">
                <Form.Control
                  type="search"
                  placeholder="Search"
                  className="me-2"
                  aria-label="Search"
                />
                <Button variant="outline-success">Search</Button>
              </Form>
            </Offcanvas.Body>
          </Navbar.Offcanvas>
        </Container>
      </Navbar>
    ))}
  </>
);
}

export default OffcanvasExample;
```

API

Navbar

```
import Navbar from 'react-bootstrap/Navbar'
```

Name	Type	Default	Description
bsPrefix	string	`'navbar'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'light'` | `'dark'`		The general visual variant a the Navbar. Use in combination with the `bg` prop, `background-color` utilities, or your own background styles.
expand	`bool` | `string`		The breakpoint, below which, the Navbar will collapse. When `true` the Navbar will always be expanded regardless of screen size.
bg	string		A convenience prop for adding `bg-*` utility classes since they are so commonly used here. `light` and `dark` are common choices but any `bg-*` class is supported, including any custom ones you might define. Pairs nicely with the `variant` prop.

Name	Type	Default	Description
fixed	`'top'` | `'bottom'`		Create a fixed navbar along the top or bottom of the screen, that scrolls with the page. A convenience prop for the `fixed-*` positioning classes.
sticky	`'top'` | `'bottom'`		Position the navbar at the top or bottom of the viewport, but only after scrolling past it. A convenience prop for the `sticky-*` positioning classes.
as	elementType		Set a custom element for this component.
onToggle	func		*controls* `expanded` A callback fired when the `<Navbar>` body collapses or expands. Fired when a `<Navbar.Toggle>` is clicked and called with the new `expanded` boolean value.
onSelect	func		A callback fired when a descendant of a child `<Nav>` is selected. Should be used to execute complex closing or other miscellaneous actions desired after selecting a descendant of `<Nav>`. Does nothing if no `<Nav>` or `<Nav>` descendants exist. The callback is called with an eventKey, which is a prop from the selected `<Nav>` descendant, and an event. `function ( eventKey: mixed, event?: SyntheticEvent )` For basic closing behavior after all `<Nav>` descendant onSelect events in mobile viewports, try using collapseOnSelect. Note: If you are manually closing the navbar using this `OnSelect` prop, ensure that you are setting `expanded` to false and not *toggling* between true and false.
collapseOnSelect	bool		Toggles `expanded` to `false` after the onSelect event of a descendant of a child `<Nav>` fires. Does nothing if no `<Nav>` or `<Nav>` descendants exist. `<NavLink>` descendants of `<Nav>` will not trigger the `onSelect` event unless an `eventKey` or `href` prop is defined.

Name	Type	Default	Description
			Manually controlling `expanded` via the onSelect callback is recommended instead, for more complex operations that need to be executed after the `select` event of `<Nav>` descendants.
expanded	bool		*controlled by: `onToggle`, initial prop: `defaultExpanded`* Controls the visibility of the navbar body
role	string	`'navigation'`	The ARIA role for the navbar, will default to 'navigation' for Navbars whose `as` is something other than `<nav>`.

NavbarBrand

```
import NavbarBrand from 'react-bootstrap/NavbarBrand'
```

Name	Type	Default	Description
bsPrefix	string	`'navbar'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
href	string		An href, when provided the Brand will render as an `<a>` element (unless `as` is provided).
as	elementType		Set a custom element for this component.

NavbarToggle

```
import NavbarToggle from 'react-bootstrap/NavbarToggle'
```

Name	Type	Default	Description
bsPrefix	string	`'navbar-toggler'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
label	string	`'Toggle navigation'`	An accessible ARIA label for the toggler button.
children	node		The toggle content. When empty, the default toggle will be rendered.
as	elementType	`'button'`	You can use a custom element type for this component.

NavbarCollapse

```
import NavbarCollapse from 'react-bootstrap/NavbarCollapse'
```

Name	Type	Default	Description
bsPrefix	string	`'navbar-collapse'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.

Navs and tabs

Documentation and examples for how to use Bootstrap's included navigation components.

Base Nav

Navigation bits in Bootstrap all share a general `Nav` component and styles. Swap `variant`s to switch between each style. The base `Nav` component is built with flexbox and provide a strong foundation for building all types of navigation components.

> **Caution**
>
> The basic, variant-less, `Nav` component does not include any `active` prop styling!

RESULT

Active Link Link Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function BasicExample() {
  return (
    <Nav
      activeKey="/home"
      onSelect={(selectedKey) => alert(`selected ${selectedKey}`)}
    >
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Link</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-2">Link</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default BasicExample;
```

`<Nav>` markup is very flexible and styling is controlled via classes so you can use whatever elements you like to build your navs. By default `<Nav>` and `<Nav.Item>` both render `<div>`s instead of `<ul>` and `<li>` elements respectively. This because it's possible (and common) to

leave off the `<Nav.Item>`'s and render a `<Nav.Link>` directly, which would create invalid markup by default (`ul > a`).

When a `<ul>` is appropriate you can render one via the `as` prop; be sure to also set your items to `<li>` as well!

RESULT

Active Link Link

LIVE EDITOR

```
import Nav from 'react-bootstrap/Nav';

function ListExample() {
  return (
    <Nav defaultActiveKey="/home" as="ul">
      <Nav.Item as="li">
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item as="li">
        <Nav.Link eventKey="link-1">Link</Nav.Link>
      </Nav.Item>
      <Nav.Item as="li">
        <Nav.Link eventKey="link-2">Link</Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default ListExample;
```

Available styles

You can control the the direction and orientation of the `Nav` by making use of the flexbox utility classes. By default, navs are left-aligned, but that is easily changed to center or right-aligned.

RESULT

Active Link Link Disabled

Or right-aligned

Active Link Link Disabled

LIVE EDITOR

```
import Nav from 'react-bootstrap/Nav';

function AlignmentExample() {
  return (
    <>
      <Nav className="justify-content-center" activeKey="/home">
        <Nav.Item>
          <Nav.Link href="/home">Active</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="link-1">Link</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="link-2">Link</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="disabled" disabled>
```

```
            Disabled
          </Nav.Link>
        </Nav.Item>
      </Nav>
      <p className="text-center mt-4 mb-4">Or right-aligned</p>
      <Nav className="justify-content-end" activeKey="/home">
        <Nav.Item>
          <Nav.Link href="/home">Active</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="link-1">Link</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="link-2">Link</Nav.Link>
        </Nav.Item>
        <Nav.Item>
          <Nav.Link eventKey="disabled" disabled>
            Disabled
          </Nav.Link>
        </Nav.Item>
      </Nav>
    </>
  );
}

export default AlignmentExample;
```

Vertical

Create stacked navs by changing the flex item direction with the `.flex-column` class, or your own css. You can even use the responsive versions to stack in some viewports but not others (e.g. `.flex-sm-column`).

RESULT

Active

Link

Link

Disabled

LIVE EDITOR

```
import Nav from 'react-bootstrap/Nav';

function StackedExample() {
  return (
    <Nav defaultActiveKey="/home" className="flex-column">
      <Nav.Link href="/home">Active</Nav.Link>
      <Nav.Link eventKey="link-1">Link</Nav.Link>
      <Nav.Link eventKey="link-2">Link</Nav.Link>
      <Nav.Link eventKey="disabled" disabled>
        Disabled
      </Nav.Link>
    </Nav>
  );
}

export default StackedExample;
```

Tabs

Visually represent nav items as "tabs". This style pairs nicely with tabbable regions created by our Tab components.

Note: creating a vertical nav (`.flex-column`) with tabs styling is unsupported by Bootstrap's default stylesheet.

RESULT

Active Option 2 Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function TabsExample() {
  return (
    <Nav variant="tabs" defaultActiveKey="/home">
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Option 2</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default TabsExample;
```

Pills

RESULT

Active Option 2 Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function PillsExample() {
  return (
    <Nav variant="pills" defaultActiveKey="/home">
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Option 2</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default PillsExample;
```

Underline

RESULT

Active Option 2 Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function UnderlineExample() {
  return (
    <Nav variant="underline" defaultActiveKey="/home">
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Option 2</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default UnderlineExample;
```

Fill and justify

Force the contents of your nav to extend the full available width. To proportionately fill the space use `fill`. Notice that the nav is the entire width but each nav item is a different size.

RESULT

Active Loooonger NavLink Link Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function FillExample() {
  return (
    <Nav fill variant="tabs" defaultActiveKey="/home">
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Loooonger NavLink</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-2">Link</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default FillExample;
```

If you want each NavItem to be the same size use `justify`.

RESULT

Active | Loooonger NavLink | Link | Disabled

LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';

function JustifiedExample() {
  return (
    <Nav justify variant="tabs" defaultActiveKey="/home">
      <Nav.Item>
        <Nav.Link href="/home">Active</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-1">Loooonger NavLink</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="link-2">Link</Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="disabled" disabled>
          Disabled
        </Nav.Link>
      </Nav.Item>
    </Nav>
  );
}

export default JustifiedExample;
```

Using dropdowns

You can mix and match the Dropdown components with the NavLink and NavItem components to create a Dropdown that plays well in a Nav component

RESULT

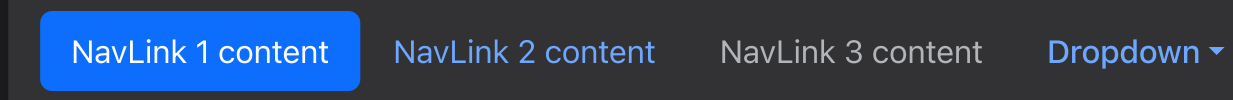
Click to see more...

LIVE EDITOR

```jsx
import Dropdown from 'react-bootstrap/Dropdown';
import NavItem from 'react-bootstrap/NavItem';
import NavLink from 'react-bootstrap/NavLink';

function DropdownImplExample() {
  return (
    <Dropdown as={NavItem}>
      <Dropdown.Toggle as={NavLink}>Click to see more…</Dropdown.Toggle>
      <Dropdown.Menu>
        <Dropdown.Item>Hello there!</Dropdown.Item>
      </Dropdown.Menu>
    </Dropdown>
  );
}

export default DropdownImplExample;
```

The above demonstrates how flexible the component model can be. But if you didn't want to roll your own versions we've included a straight-forward `<NavDropdown>` that works for most cases.

RESULT



LIVE EDITOR

```jsx
import Nav from 'react-bootstrap/Nav';
import NavDropdown from 'react-bootstrap/NavDropdown';

function NavDropdownExample() {
  const handleSelect = (eventKey) => alert(`selected ${eventKey}`);

  return (
    <Nav variant="pills" activeKey="1" onSelect={handleSelect}>
      <Nav.Item>
        <Nav.Link eventKey="1" href="#/home">
          NavLink 1 content
        </Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="2" title="Item">
          NavLink 2 content
        </Nav.Link>
      </Nav.Item>
      <Nav.Item>
        <Nav.Link eventKey="3" disabled>
          NavLink 3 content
        </Nav.Link>
      </Nav.Item>
      <NavDropdown title="Dropdown" id="nav-dropdown">
        <NavDropdown.Item eventKey="4.1">Action</NavDropdown.Item>
        <NavDropdown.Item eventKey="4.2">Another
action</NavDropdown.Item>
        <NavDropdown.Item eventKey="4.3">Something else
here</NavDropdown.Item>
        <NavDropdown.Divider />
        <NavDropdown.Item eventKey="4.4">Separated
link</NavDropdown.Item>
      </NavDropdown>
    </Nav>
  );
}

export default NavDropdownExample;
```

API

Nav

```jsx
import Nav from 'react-bootstrap/Nav'
```

Name	Type	Default	Description
bsPrefix	string	`'nav'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'tabs'` | `'pills'` | `'underline'`		The visual variant of the nav items.
activeKey	`string` | `number`		Marks the NavItem with a matching `eventKey` (or `href` if present) as active.

Name	Type	Default	Description
defaultActiveKey	`string` | `number`		The default active key that is selected on start.
fill	bool		Have all `NavItem`s proportionately fill all available width.
justify	`boolean`		Have all `NavItem`s evenly fill all available width.
onSelect	func		A callback fired when a NavItem is selected. `function ( Any eventKey, SyntheticEvent event? )`
role	string		ARIA role for the Nav, in the context of a TabContainer, the default will be set to "tablist", but can be overridden by the Nav when set explicitly. When the role is "tablist", NavLink focus is managed according to the ARIA authoring practices for tabs: https://www.w3.org/TR/2013/WD-wai-aria-practices-20130307/#tabpanel
navbar	bool		Apply styling an alignment for use in a Navbar. This prop will be set automatically when the Nav is used inside a Navbar.
navbarScroll	bool		Enable vertical scrolling within the toggleable contents of a collapsed Navbar.
as	elementType		You can use a custom element type for this component.

NavItem

```
import NavItem from 'react-bootstrap/NavItem'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

NavLink

```
import NavLink from 'react-bootstrap/NavLink'
```

Name	Type	Default	Description
bsPrefix	string	`'nav-link'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.

Name	Type	Default	Description
active	bool		The active state of the NavItem item.
disabled	bool	false	The disabled state of the NavItem item.
role	string		The ARIA role for the NavLink, In the context of a 'tablist' parent Nav, the role defaults to 'tab'
href	string		The HTML href attribute for the NavLink. Used as the unique identifier for the NavLink if an eventKey is not provided.
eventKey	string | number		Uniquely identifies the NavItem amongst its siblings, used to determine and control the active state of the parent Nav as well as onSelect behavior of a parent Navbar.
as	elementType	'a'	You can use a custom element type for this component.

NavDropdown

```
import NavDropdown from 'react-bootstrap/NavDropdown'
```

Name	Type	Default	Description
id	string		An html id attribute for the Toggle button, necessary for assistive technologies, such as screen readers.
onClick	func		An onClick handler passed to the Toggle component
title Required	node		The content of the non-toggle Button.
disabled	bool		Disables the toggle NavLink
active	bool		Style the toggle NavLink as active
menuRole	string		An ARIA accessible role applied to the Menu component. When set to 'menu', The dropdown
renderMenuOnMount	bool		Whether to render the dropdown menu in the DOM before the first time it is shown
rootCloseEvent	string		Which event when fired outside the component will cause it to be closed. *see DropdownMenu for more details*
menuVariant	'dark'		Menu color variant. Omitting this will use the default light color.
bsPrefix	string		Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.

Offcanvas

Build hidden sidebars into your project for navigation, shopping carts, and more.

Examples

Offcanvas includes support for a header with a close button and an optional body class for some initial padding. We suggest that you include offcanvas headers with dismiss actions whenever possible, or provide an explicit dismiss action.

Basic Example

RESULT

Launch

LIVE EDITOR

```jsx
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Offcanvas from 'react-bootstrap/Offcanvas';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow}>
        Launch
      </Button>

      <Offcanvas show={show} onHide={handleClose}>
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Offcanvas</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          Some text as placeholder. In real life you can have the
elements you
          have chosen. Like, text, images, lists, etc.
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}

export default Example;
```

Responsive

Responsive offcanvas classes hide content outside the viewport from a specified breakpoint and down. Above that breakpoint, the contents within will behave as usual.

RESULT

Launch

LIVE EDITOR

```jsx
import { useState } from 'react';
import Alert from 'react-bootstrap/Alert';
import Button from 'react-bootstrap/Button';
import Offcanvas from 'react-bootstrap/Offcanvas';

function ResponsiveExample() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" className="d-lg-none" onClick={handleShow}>
        Launch
      </Button>

      <Alert variant="info" className="d-none d-lg-block">
        Resize your browser to show the responsive offcanvas toggle.
      </Alert>

      <Offcanvas show={show} onHide={handleClose} responsive="lg">
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Responsive offcanvas</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          <p className="mb-0">
            This is content within an <code>.offcanvas-lg</code>.
          </p>
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}

export default ResponsiveExample;
```

Placement

Offcanvas supports a few different placements:

- `start` places offcanvas on the left of the viewport
- `end` places offcanvas on the right of the viewport
- `top` places offcanvas on the top of the viewport
- `bottom` places offcanvas on the bottom of the viewport

RESULT

start end top bottom

LIVE EDITOR

```jsx
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Offcanvas from 'react-bootstrap/Offcanvas';
```

```
function OffCanvasExample({ name, ...props }) {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
      <Button variant="primary" onClick={handleShow} className="me-2">
        {name}
      </Button>
      <Offcanvas show={show} onHide={handleClose} {...props}>
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Offcanvas</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          Some text as placeholder. In real life you can have the
elements you
          have chosen. Like, text, images, lists, etc.
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}

function Example() {
  return (
    <>
      {['start', 'end', 'top', 'bottom'].map((placement, idx) => (
        <OffCanvasExample key={idx} placement={placement} name=
{placement} />
      ))}
    </>
  );
}

render(<Example />);
```

Backdrop

Scrolling the `<body>` element is disabled when an offcanvas and its backdrop are visible. Use the `scroll` prop to toggle `<body>` scrolling and the `backdrop` prop to toggle the backdrop.

RESULT

Enable backdrop (default) | Disable backdrop | Enable body scrolling | Enable both scrolling & backdrop

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Offcanvas from 'react-bootstrap/Offcanvas';

const options = [
  {
    name: 'Enable backdrop (default)',
    scroll: false,
    backdrop: true,
  },
  {
    name: 'Disable backdrop',
    scroll: false,
    backdrop: false,
  },
  {
    name: 'Enable body scrolling',
    scroll: true,
```

```
    backdrop: false,
  },
  {
    name: 'Enable both scrolling & backdrop',
    scroll: true,
    backdrop: true,
  },
];

function OffCanvasExample({ name, ...props }) {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const toggleShow = () => setShow((s) => !s);

  return (
    <>
      <Button variant="primary" onClick={toggleShow} className="me-2">
        {name}
      </Button>
      <Offcanvas show={show} onHide={handleClose} {...props}>
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Offcanvas</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          Some text as placeholder. In real life you can have the
elements you
          have chosen. Like, text, images, lists, etc.
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}

function Example() {
  return (
    <>
      {options.map((props, idx) => (
        <OffCanvasExample key={idx} {...props} />
      ))}
    </>
  );
}

render(<Example />);
```

Static backdrop

When `backdrop` is set to `static`, the offcanvas will not close when clicking outside of it.

RESULT

Toggle static offcanvas

LIVE EDITOR

```
import { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Offcanvas from 'react-bootstrap/Offcanvas';

function Example() {
  const [show, setShow] = useState(false);

  const handleClose = () => setShow(false);
  const handleShow = () => setShow(true);

  return (
    <>
```

```
      <Button variant="primary" onClick={handleShow}>
        Toggle static offcanvas
      </Button>

      <Offcanvas show={show} onHide={handleClose} backdrop="static">
        <Offcanvas.Header closeButton>
          <Offcanvas.Title>Offcanvas</Offcanvas.Title>
        </Offcanvas.Header>
        <Offcanvas.Body>
          I will not close if you click outside of me.
        </Offcanvas.Body>
      </Offcanvas>
    </>
  );
}

export default Example;
```

API

Offcanvas

```
import Offcanvas from 'react-bootstrap/Offcanvas'
```

Name	Type	Default	Description
bsPrefix	string	`'offcanvas'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
backdrop	`'static'` | `true` | `false`	`true`	Include a backdrop component. Specify 'static' for a backdrop that doesn't trigger an "onHide" when clicked.
backdropClassName	string		Add an optional extra class name to .offcanvas-backdrop.
keyboard	bool	`true`	Closes the offcanvas when escape key is pressed.
scroll	bool	`false`	Allow body scrolling while offcanvas is open.
placement	`'start'` | `'end'` | `'top'` | `'bottom'`	`'start'`	Which side of the viewport the offcanvas will appear from.
responsive	`"sm"` | `"md"` | `"lg"` | `"xl"` | `"xxl"`		Hide content outside the viewport from a specified breakpoint and down.
autoFocus	bool	`true`	When `true` The offcanvas will automatically shift focus to itself when it opens, and replace it to the last focused element when it closes. Generally this should never be set to false as it makes the offcanvas less accessible to

Name	Type	Default	Description
			assistive technologies, like screen-readers.
`enforceFocus`	`bool`	`true`	When `true` The offcanvas will prevent focus from leaving the offcanvas while open. Consider leaving the default value here, as it is necessary to make the offcanvas work well with assistive technologies, such as screen readers.
`restoreFocus`	`bool`	`true`	When `true` The offcanvas will restore focus to previously focused element once offcanvas is hidden
`restoreFocusOptions`	`shape`		Options passed to focus function when `restoreFocus` is set to `true`
`show`	`bool`	`false`	When `true` The offcanvas will show itself.
`onShow`	`func`		A callback fired when the offcanvas is opening.
`onHide`	`func`		A callback fired when the header closeButton or backdrop is clicked. Required if either are specified.
`onEscapeKeyDown`	`func`		A callback fired when the escape key, if specified in `keyboard`, is pressed.
`onEnter`	`func`		Callback fired before the offcanvas transitions in
`onEntering`	`func`		Callback fired as the offcanvas begins to transition in
`onEntered`	`func`		Callback fired after the offcanvas finishes transitioning in
`onExit`	`func`		Callback fired right before the offcanvas transitions out
`onExiting`	`func`		Callback fired as the offcanvas begins to transition out
`onExited`	`func`		Callback fired after the offcanvas finishes transitioning out
`aria-labelledby`	`string`		

OffcanvasHeader

```
import OffcanvasHeader from 'react-bootstrap/OffcanvasHeader'
```

Name	Type	Default	Description
bsPrefix	string	`'offcanvas-header'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
closeLabel	string	`'Close'`	Provides an accessible label for the close button. It is used for Assistive Technology when the label text is not readable.
closeVariant	`'white'`		Sets the variant for close button.
closeButton	bool	`false`	Specify whether the Component should contain a close button
onHide	func		A Callback fired when the close button is clicked. If used directly inside a Offcanvas component, the onHide will automatically be propagated up to the parent Offcanvas `onHide`.

OffcanvasTitle

```
import OffcanvasTitle from 'react-bootstrap/OffcanvasTitle'
```

Name	Type	Default	Description
as		`divWithClassName('h5')`	You can use a custom element type for this component.

OffcanvasBody

```
import OffcanvasBody from 'react-bootstrap/OffcanvasBody'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

Overlay

A set of components for positioning beautiful overlays, tooltips, popovers, and anything else you need.

Overview

Things to know about the React-Bootstrap Overlay components.

- Overlays rely on the third-party library Popper.js. It's included automatically with React-Bootstrap, but you should reference the API for more advanced use cases.
- The `<Tooltip>` and `<Popover>` components do not position themselves. Instead the `<Overlay>` (or `<OverlayTrigger>`) components, inject `ref` and `style` props.
- Tooltip expects specific props injected by the `<Overlay>` component.
- Tooltips for `disabled` elements must be triggered on a wrapper element.

Overlay

`Overlay` is the fundamental component for positioning and controlling tooltip visibility. It's a wrapper around Popper.js, that adds support for transitions, and visibility toggling.

Creating an Overlay

Overlays consist of at least two elements, the "overlay", the element to be positioned, as well as a "target", the element the overlay is positioned in relation to. You can also have an "arrow" element, like the tooltips and popovers, but that is optional. Be sure to check out the Popper documentation for more details about the injected props.

RESULT

Click me to see

LIVE EDITOR

```jsx
import { useState, useRef } from 'react';
import Button from 'react-bootstrap/Button';
import Overlay from 'react-bootstrap/Overlay';

function Example() {
  const [show, setShow] = useState(false);
  const target = useRef(null);

  return (
    <>
      <Button variant="danger" ref={target} onClick={() =>
setShow(!show)}>
        Click me to see
      </Button>
      <Overlay target={target.current} show={show} placement="right">
        {({
          placement: _placement,
          arrowProps: _arrowProps,
```

```
          show: _show,
          popper: _popper,
          hasDoneInitialMeasure: _hasDoneInitialMeasure,
          ...props
        }) => (
          <div
            {...props}
            style={{
              position: 'absolute',
              backgroundColor: 'rgba(255, 100, 100, 0.85)',
              padding: '2px 10px',
              color: 'white',
              borderRadius: 3,
              ...props.style,
            }}
          >
            Simple tooltip
          </div>
        )}
      </Overlay>
    </>
  );
}

export default Example;
```

Customizing Overlay rendering

The `Overlay` injects a number of props that you can use to customize the rendering behavior. There is a case where you would need to show the overlay before `Popper` can measure and position it properly. In React-Bootstrap, tooltips and popovers sets the opacity and position to avoid issues where the initial positioning of the overlay is incorrect. See the Tooltip implementation for an example on how this is done.

OverlayTrigger

Since the above pattern is pretty common, but verbose, we've included `<OverlayTrigger>` component to help with common use-cases. It even has functionality to delayed show or hides, and a few different "trigger" events you can mix and match.

Note that triggering components must be able to accept a ref since `<OverlayTrigger>` will attempt to add one. You can use forwardRef() for function components.

RESULT

Hover me to see

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Tooltip from 'react-bootstrap/Tooltip';

function TriggerExample() {
  const renderTooltip = (props) => (
    <Tooltip id="button-tooltip" {...props}>
      Simple tooltip
    </Tooltip>
  );

  return (
    <OverlayTrigger
      placement="right"
      delay={{ show: 250, hide: 400 }}
      overlay={renderTooltip}
    >
```

```
      <Button variant="success">Hover me to see</Button>
    </OverlayTrigger>
  );
}

export default TriggerExample;
```

Customizing trigger behavior

For more advanced behaviors `<OverlayTrigger>` accepts a function child that passes in the injected `ref` and event handlers that correspond to the configured `trigger` prop.

You can manually apply the props to any element you want or split them up. The example below shows how to position the overlay to a different element than the one that triggers its visibility.

> **Pro Tip**
>
> Using the function form of OverlayTrigger avoids a `React.findDOMNode` call, for those trying to be strict mode compliant.

RESULT

J Hover to see

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Image from 'react-bootstrap/Image';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';

function TriggerRendererProp() {
  return (
    <OverlayTrigger
      placement="bottom"
      overlay={<Tooltip id="button-tooltip-2">Check out this
avatar</Tooltip>}
    >
      {({ ref, ...triggerHandler }) => (
        <Button
          variant="light"
          {...triggerHandler}
          className="d-inline-flex align-items-center"
        >
          <Image
            ref={ref}
            roundedCircle
            src="holder.js/20x20?text=J&bg=28a745&fg=FFF"
          />
          <span className="ms-1">Hover to see</span>
        </Button>
      )}
    </OverlayTrigger>
  );
}

export default TriggerRendererProp;
```

Tooltips

A tooltip component for a more stylish alternative to that anchor tag `title` attribute.

Examples

Hover over the links below to see tooltips.

RESULT

Tight pants next level keffiyeh you probably haven't heard of them. Farm-to-table seitan, mcsweeney's fixie sustainable quinoa 8-bit american apparel have a terry richardson vinyl chambray. Beard stumptown, cardigans banh mi lomo thundercats. Tofu biodiesel williamsburg marfa, four loko mcsweeney's cleanse vegan chambray. A really ironic artisan whatever keytar, scenester farm-to-table banksy Austin twitter handle freegan cred raw denim single-origin coffee viral.

LIVE EDITOR

```jsx
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Tooltip from 'react-bootstrap/Tooltip';

function TooltipInCopyExample() {
  const Link = ({ id, children, title }) => (
    <OverlayTrigger overlay={<Tooltip id={id}>{title}</Tooltip>}>
      <a href="#">{children}</a>
    </OverlayTrigger>
  );

  return (
    <p>
      Tight pants next level keffiyeh{' '}
      <Link title="Default title" id="t-1">
        you probably
      </Link>{' '}
      haven't heard of them. Farm-to-table seitan, mcsweeney's fixie sustainable
      quinoa 8-bit american apparel{' '}
      <Link id="t-2" title="Another one">
        have a
      </Link>{' '}
      terry richardson vinyl chambray. Beard stumptown, cardigans banh mi lomo
      thundercats. Tofu biodiesel williamsburg marfa, four loko mcsweeney's
      cleanse vegan chambray. A really ironic artisan{' '}
      <Link title="Another one here too" id="t-3">
        whatever keytar
      </Link>
      , scenester farm-to-table banksy Austin{' '}
      <Link title="The last tip!" id="t-4">
        twitter handle
      </Link>{' '}
      freegan cred raw denim single-origin coffee viral.
    </p>
  );
}

export default TooltipInCopyExample;
```

You can pass the `Overlay` injected props directly to the Tooltip component.

RESULT

Click me!

LIVE EDITOR

```jsx
import { useState, useRef } from 'react';
import Button from 'react-bootstrap/Button';
import Overlay from 'react-bootstrap/Overlay';
import Tooltip from 'react-bootstrap/Tooltip';
```

```jsx
function Example() {
  const [show, setShow] = useState(false);
  const target = useRef(null);

  return (
    <>
      <Button ref={target} onClick={() => setShow(!show)}>
        Click me!
      </Button>
      <Overlay target={target.current} show={show} placement="right">
        {(props) => (
          <Tooltip id="overlay-example" {...props}>
            My Tooltip
          </Tooltip>
        )}
      </Overlay>
    </>
  );
}

export default Example;
```

Or pass a Tooltip element to `OverlayTrigger` instead.

RESULT

Tooltip on top | Tooltip on right | Tooltip on bottom | Tooltip on left

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Tooltip from 'react-bootstrap/Tooltip';

function TooltipPositionedExample() {
  return (
    <>
      {['top', 'right', 'bottom', 'left'].map((placement) => (
        <OverlayTrigger
          key={placement}
          placement={placement}
          overlay={
            <Tooltip id={`tooltip-${placement}`}>
              Tooltip on <strong>{placement}</strong>.
            </Tooltip>
          }
        >
          <Button variant="secondary">Tooltip on {placement}</Button>
        </OverlayTrigger>
      ))}
    </>
  );
}

export default TooltipPositionedExample;
```

Popovers

A popover component, like those found in iOS.

Examples

RESULT

Click me to see

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Popover from 'react-bootstrap/Popover';

const popover = (
  <Popover id="popover-basic">
    <Popover.Header as="h3">Popover right</Popover.Header>
    <Popover.Body>
      And here's some <strong>amazing</strong> content. It's very
engaging.
      right?
    </Popover.Body>
  </Popover>
);

const Example = () => (
  <OverlayTrigger trigger="click" placement="right" overlay={popover}>
    <Button variant="success">Click me to see</Button>
  </OverlayTrigger>
);

render(<Example />);
```

As with `<Tooltip>`s, you can control the placement of the Popover.

RESULT

Popover on top | Popover on right | Popover on bottom | Popover on left

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Popover from 'react-bootstrap/Popover';

function PopoverPositionedExample() {
  return (
    <>
      {['top', 'right', 'bottom', 'left'].map((placement) => (
        <OverlayTrigger
          trigger="click"
          key={placement}
          placement={placement}
          overlay={
            <Popover id={`popover-positioned-${placement}`}>
              <Popover.Header as="h3">{`Popover ${placement}`}
</Popover.Header>
              <Popover.Body>
                <strong>Holy guacamole!</strong> Check this info.
              </Popover.Body>
            </Popover>
          }
        >
          <Button variant="secondary">Popover on {placement}</Button>
        </OverlayTrigger>
      ))}
    </>
  );
}

export default PopoverPositionedExample;
```

Disabled elements

Elements with the `disabled` attribute aren't interactive, meaning users cannot hover or click them to trigger a popover (or tooltip). As a workaround, you'll want to trigger the overlay from a wrapper `<div>` or `<span>` and override the `pointer-events` on the disabled element.

RESULT

Disabled button

LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Tooltip from 'react-bootstrap/Tooltip';

function DisabledExample() {
  return (
    <OverlayTrigger overlay={<Tooltip id="tooltip-disabled">Tooltip!
</Tooltip>}>
      <span className="d-inline-block">
        <Button disabled style={{ pointerEvents: 'none' }}>
          Disabled button
        </Button>
      </span>
    </OverlayTrigger>
  );
}

export default DisabledExample;
```

Changing containers

You can specify a `container` to control the DOM element the overlay is appended to. This is especially useful when styles conflict with your Overlays.

RESULT

Holy guacamole!

LIVE EDITOR

```
import { useState, useRef } from 'react';
import Button from 'react-bootstrap/Button';
import Overlay from 'react-bootstrap/Overlay';
import Popover from 'react-bootstrap/Popover';

function Example() {
  const [show, setShow] = useState(false);
  const [target, setTarget] = useState(null);
  const ref = useRef(null);

  const handleClick = (event) => {
    setShow(!show);
    setTarget(event.target);
  };

  return (
```

```
    <div ref={ref}>
      <Button onClick={handleClick}>Holy guacamole!</Button>

      <Overlay
        show={show}
        target={target}
        placement="bottom"
        container={ref}
        containerPadding={20}
      >
        <Popover id="popover-contained">
          <Popover.Header as="h3">Popover bottom</Popover.Header>
          <Popover.Body>
            <strong>Holy guacamole!</strong> Check this info.
          </Popover.Body>
        </Popover>
      </Overlay>
    </div>
  );
}

export default Example;
```

Updating position dynamically

Since we can't know every time your overlay changes size, to reposition it, you need to take manual action if you want to update the position of an Overlay in response to a change.

For this, the Overlay component also injects a `popper` prop with a `scheduleUpdate()` method that an overlay component can use to reposition itself.

RESULT

Holy guacamole!

LIVE EDITOR

```
import React, { useEffect, useState } from 'react';
import Button from 'react-bootstrap/Button';
import OverlayTrigger from 'react-bootstrap/OverlayTrigger';
import Popover from 'react-bootstrap/Popover';

const UpdatingPopover = React.forwardRef(
  ({ popper, children, show: _, ...props }, ref) => {
    useEffect(() => {
      console.log('updating!');
      popper.scheduleUpdate();
    }, [children, popper]);

    return (
      <Popover ref={ref} body {...props}>
        {children}
      </Popover>
    );
  },
);

const longContent = `
  Very long
  Multiline content
  that is engaging and what-not
`;
const shortContent = 'Short and sweet!';

function Example() {
  const [content, setContent] = useState(shortContent);

  useEffect(() => {
```

```
    const timerId = setInterval(() => {
      setContent(content === shortContent ? longContent : shortContent);
    }, 3000);

    return () => clearInterval(timerId);
  });

  return (
    <OverlayTrigger
      trigger="click"
      overlay={
        <UpdatingPopover id="popover-contained">{content}
</UpdatingPopover>
      }
    >
      <Button>Holy guacamole!</Button>
    </OverlayTrigger>
  );
}

render(<Example />);
```

API

Overlay

```
import Overlay from 'react-bootstrap/Overlay'
```

Name	Type	Default	Description
container	`custom` | `func`		A component instance, DOM node, or function that returns either. The `container` element will have the Overlay appended to it via a React portal.
target	`custom` | `func`		A component instance, DOM node, or function that returns either. The overlay will be positioned in relation to the `target`
show	bool	`false`	Set the visibility of the Overlay
popperConfig	object	`{}`	A set of popper options and props passed directly to Popper.
rootClose	bool	`false`	Specify whether the overlay should trigger onHide when the user clicks outside the overlay
rootCloseEvent	`'click'` | `'mousedown'`		Specify event for triggering a "root close" toggle.
onHide	func		A callback invoked by the overlay when it wishes to be hidden. Required if `rootClose` is specified.

Name	Type	Default	Description
`transition`	`bool` | `elementType`	`Fade`	Animate the entering and exiting of the Overlay. `true` will use the `<Fade>` transition, or a custom react-transition-group `<Transition>` component can be provided.
`onEnter`	`func`		Callback fired before the Overlay transitions in
`onEntering`	`func`		Callback fired as the Overlay begins to transition in
`onEntered`	`func`		Callback fired after the Overlay finishes transitioning in
`onExit`	`func`		Callback fired right before the Overlay transitions out
`onExiting`	`func`		Callback fired as the Overlay begins to transition out
`onExited`	`func`		Callback fired after the Overlay finishes transitioning out
`placement`	`'auto-start'` | `'auto'` | `'auto-end'` | `'top-start'` | `'top'` | `'top-end'` | `'right-start'` | `'right'` | `'right-end'` | `'bottom-end'` | `'bottom'` | `'bottom-start'` | `'left-end'` | `'left'` | `'left-start'`	`'top'`	The placement of the Overlay in relation to it's `target`.

OverlayTrigger

```
import OverlayTrigger from 'react-bootstrap/OverlayTrigger'
```

Name	Type	Default	Description
`children` Required	`element` | `func`		
`trigger`	`'hover'` | `'click'` | `'focus'` | `Array<'hover'` | `'click'` | `'focus'>`	`['hover', 'focus']`	Specify which action or actions trigger Overlay visibility The `click` trigger ignores the configured `delay`.
`delay`	`number` | `shape`		A millisecond delay amount to show and hide the Overlay once triggered

Name	Type	Default	Description
`show`	`bool`		*controlled by: `onToggle`, initial prop: `defaultShow`* The visibility of the Overlay. `show` is a *controlled* prop so should be paired with `onToggle` to avoid breaking user interactions. Manually toggling `show` does **not** wait for `delay` to change the visibility.
`defaultShow`	`bool`	`false`	The initial visibility state of the Overlay.
`onToggle`	`func`		*controls `show`* A callback that fires when the user triggers a change in tooltip visibility. `onToggle` is called with the desired next `show`, and generally should be passed back to the `show` prop. `onToggle` fires *after* the configured `delay`
`flip`	`bool`	`placement && placement.indexOf('auto') !== -1`	The initial flip state of the Overlay.
`overlay` Required	`func` | `element`		An element or text to overlay next to the target.
`popperConfig`	`object`	`{}`	A Popper.js config object passed to the underlying popper instance.

Name	Type	Default	Description
placement	`'auto-start'` | `'auto'` | `'auto-end'` | `'top-start'` | `'top'` | `'top-end'` | `'right-start'` | `'right'` | `'right-end'` | `'bottom-end'` | `'bottom'` | `'bottom-start'` | `'left-end'` | `'left'` | `'left-start'`		The placement of the Overlay in relation to it's `target`.

Tooltip

```
import Tooltip from 'react-bootstrap/Tooltip'
```

Name	Type	Default	Description
bsPrefix	string	`'tooltip'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
id	`string`		An html id attribute, necessary for accessibility
placement	`'auto-start'` | `'auto'` | `'auto-end'` | `'top-start'` | `'top'` | `'top-end'` | `'right-start'` | `'right'` | `'right-end'` | `'bottom-end'` | `'bottom'` | `'bottom-start'` | `'left-end'` | `'left'` | `'left-start'`	`'right'`	Sets the direction the Tooltip is positioned towards. > This is generally provided by the `Overlay` component positioning the tooltip
arrowProps	`{ ref: ReactRef, style: Object }`		An Overlay injected set of props for positioning the tooltip arrow. > This is generally provided by the `Overlay` component positioning the tooltip
hasDoneInitialMeasure	bool		Whether or not Popper has done its initial

Name	Type	Default	Description
			measurement and positioning.

Popover

```
import Popover from 'react-bootstrap/Popover'
```

Name	Type	Default	Description
bsPrefix	string	`'popover'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
id	`string`		An html id attribute, necessary for accessibility
placement	`'auto-start'` | `'auto'` | `'auto-end'` | `'top-start'` | `'top'` | `'top-end'` | `'right-start'` | `'right'` | `'right-end'` | `'bottom-end'` | `'bottom'` | `'bottom-start'` | `'left-end'` | `'left'` | `'left-start'`	`'right'`	Sets the direction the Popover is positioned towards. > This is generally provided by the `Overlay` component positioning the popover
arrowProps	shape		An Overlay injected set of props for positioning the popover arrow. > This is generally provided by the `Overlay` component positioning the popover
body	bool		When this prop is set, it creates a Popover with a Popover.Body inside passing the children directly to it
hasDoneInitialMeasure	bool		Whether or not Popper has done its initial measurement and positioning.

PopoverBody

```
import PopoverBody from 'react-bootstrap/PopoverBody'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

PopoverHeader

```
import PopoverHeader from 'react-bootstrap/PopoverHeader'
```

Name	Type	Default	Description
as		`'div'`	You can use a custom element type for this component.

Pagination

A set of presentational components for building pagination UI.

Example

RESULT

LIVE EDITOR

```
import Pagination from 'react-bootstrap/Pagination';

let active = 2;
let items = [];
for (let number = 1; number <= 5; number++) {
  items.push(
    <Pagination.Item key={number} active={number === active}>
      {number}
    </Pagination.Item>,
  );
}

const paginationBasic = (
  <div>
    <Pagination>{items}</Pagination>
    <br />

    <Pagination size="lg">{items}</Pagination>
    <br />

    <Pagination size="sm">{items}</Pagination>
  </div>
);

render(paginationBasic);
```

More options

For building more complex pagination UI, there are few convenient sub-components for adding "First", "Previous", "Next", and "Last" buttons, as well as an "Ellipsis" item for indicating previous or continuing results.

RESULT

« ‹ 1 ... 10 11 12 13 14 ... 20 › »

LIVE EDITOR

```jsx
import Pagination from 'react-bootstrap/Pagination';

function AdvancedExample() {
  return (
    <Pagination>
      <Pagination.First />
      <Pagination.Prev />
      <Pagination.Item>{1}</Pagination.Item>
      <Pagination.Ellipsis />

      <Pagination.Item>{10}</Pagination.Item>
      <Pagination.Item>{11}</Pagination.Item>
      <Pagination.Item active>{12}</Pagination.Item>
      <Pagination.Item>{13}</Pagination.Item>
      <Pagination.Item disabled>{14}</Pagination.Item>

      <Pagination.Ellipsis />
      <Pagination.Item>{20}</Pagination.Item>
      <Pagination.Next />
      <Pagination.Last />
    </Pagination>
  );
}

export default AdvancedExample;
```

API

Pagination

```jsx
import Pagination from 'react-bootstrap/Pagination'
```

Name	Type	Default	Description
bsPrefix	string	`'pagination'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
size	`'sm'` | `'lg'`		Sets the size of all PageItems.

PageItem

```jsx
import PageItem from 'react-bootstrap/PageItem'
```

Name	Type	Default	Description
disabled	bool	`false`	Disables the PageItem
active	bool	`false`	Styles PageItem as active, and renders a `<span>` instead of an `<a>`.
activeLabel	string	`'(current)'`	An accessible label indicating the active state.

Name	Type	Default	Description
`href`	`string`		The HTML href attribute for the `PageItem`.
`onClick`	`func`		A callback function for when this component is clicked.
`linkStyle`	`object`		custom style for the inner component of the PageItem
`linkClassName`	`string`		custom className for the inner component of the PageItem
`as`		`Anchor`	You can use a custom element type for this component.

Placeholders

Use loading placeholders for your components or pages to indicate something may still be loading.

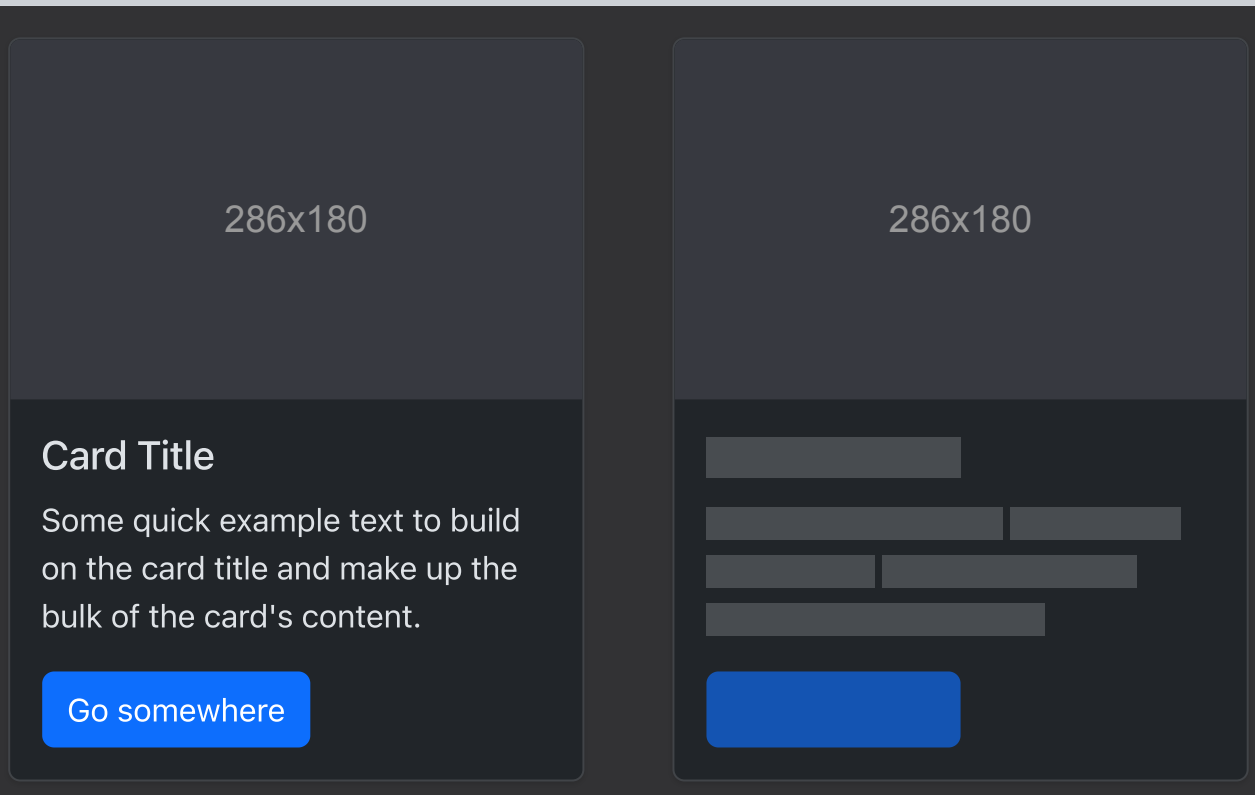
About

Placeholders can be used to enhance the experience of your application. They're built only with HTML and CSS, meaning you don't need any JavaScript to create them. You will, however, need some custom JavaScript to toggle their visibility. Their appearance, color, and sizing can be easily customized with our utility classes.

Example

In the example below, we take a typical card component and recreate it with placeholders applied to create a "loading card". Size and proportions are the same between the two.

RESULT



LIVE EDITOR

```
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import Placeholder from 'react-bootstrap/Placeholder';

function CardExample() {
  return (
    <div className="d-flex justify-content-around">
      <Card style={{ width: '18rem' }}>
        <Card.Img variant="top" src="holder.js/100px180" />
        <Card.Body>
          <Card.Title>Card Title</Card.Title>
          <Card.Text>
            Some quick example text to build on the card title and make
up the
            bulk of the card's content.
```

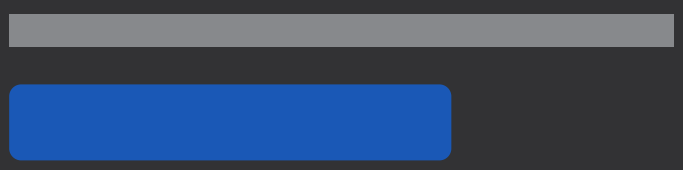
```
          </Card.Text>
          <Button variant="primary">Go somewhere</Button>
        </Card.Body>
      </Card>

      <Card style={{ width: '18rem' }}>
        <Card.Img variant="top" src="holder.js/100px180" />
        <Card.Body>
          <Placeholder as={Card.Title} animation="glow">
            <Placeholder xs={6} />
          </Placeholder>
          <Placeholder as={Card.Text} animation="glow">
            <Placeholder xs={7} /> <Placeholder xs={4} /> <Placeholder
xs={4} />{' '}
            <Placeholder xs={6} /> <Placeholder xs={8} />
          </Placeholder>
          <Placeholder.Button variant="primary" xs={6} />
        </Card.Body>
      </Card>
    </div>
  );
}

export default CardExample;
```

How it works

Create placeholders with the `Placeholder` component and a grid column prop (e.g., `xs={6}`) to set the `width`. They can replace the text inside an element or be added to an existing component via the `as` prop.

Additional styling is applied to `PlaceholderButton`s via `::before` to ensure the `height` is respected. You may extend this pattern for other situations as needed, or add a ` ` within the element to reflect the height when actual text is rendered in its place.

RESULT



LIVE EDITOR

```
import Placeholder from 'react-bootstrap/Placeholder';

function BasicExample() {
  return (
    <>
      <p aria-hidden="true">
        <Placeholder xs={6} />
      </p>

      <Placeholder.Button xs={4} aria-hidden="true" />
    </>
  );
}

export default BasicExample;
```

Info

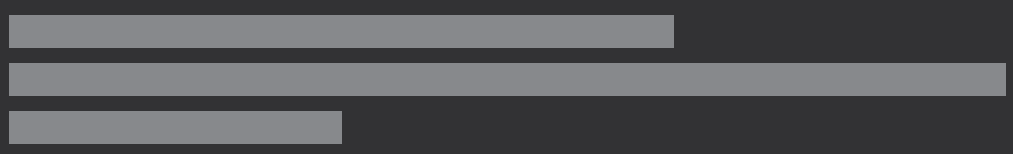
The use of `aria-hidden="true"` only indicates that the element should be hidden to screen readers. The *loading* behaviour of the placeholder depends on how authors will actually use the placeholder styles, how they plan to update things, etc. Some JavaScript

code may be needed to *swap* the state of the placeholder and inform AT users of the update.

Width

You can change the `width` through grid column classes, width utilities, or inline styles.

RESULT



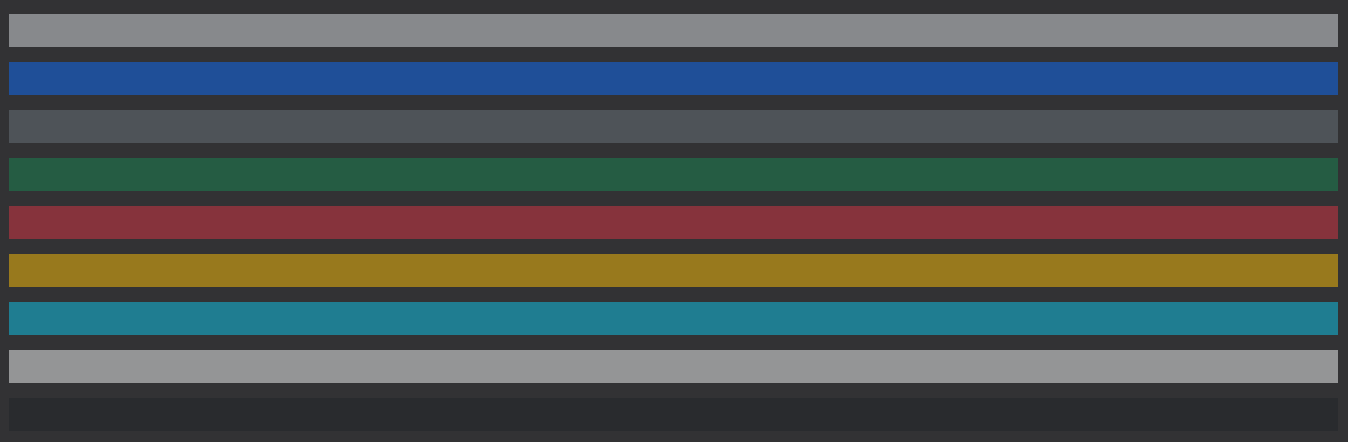
LIVE EDITOR

```
import Placeholder from 'react-bootstrap/Placeholder';

function WidthExample() {
  return (
    <>
      <Placeholder xs={6} />
      <Placeholder className="w-75" /> <Placeholder style={{ width: '25%'
}} />
    </>
  );
}

export default WidthExample;
```

Color

By default, the `Placeholder` uses `currentColor`. This can be overridden with a custom color or utility class.

RESULT



LIVE EDITOR

```
import Placeholder from 'react-bootstrap/Placeholder';

function ColorExample() {
  return (
    <>
      <Placeholder xs={12} />

      <Placeholder xs={12} bg="primary" />
      <Placeholder xs={12} bg="secondary" />
      <Placeholder xs={12} bg="success" />
      <Placeholder xs={12} bg="danger" />
      <Placeholder xs={12} bg="warning" />
      <Placeholder xs={12} bg="info" />
      <Placeholder xs={12} bg="light" />
      <Placeholder xs={12} bg="dark" />
    </>
```

```
  );
}

export default ColorExample;
```

Sizing

The size of `Placeholder`s are based on the typographic style of the parent element. Customize them with sizing props: `lg`, `sm`, or `xs`.

RESULT

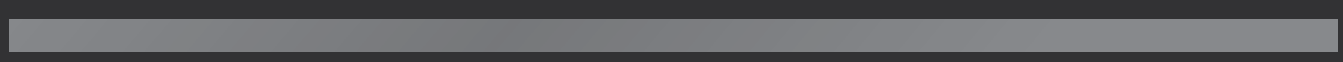
LIVE EDITOR

```
import Placeholder from 'react-bootstrap/Placeholder';

function SizeExample() {
  return (
    <>
      <Placeholder xs={12} size="lg" />
      <Placeholder xs={12} />
      <Placeholder xs={12} size="sm" />
      <Placeholder xs={12} size="xs" />
    </>
  );
}

export default SizeExample;
```

Animation

Animate placeholders by setting the prop `animation` to `glow` or `wave` to better convey the perception of something being *actively* loaded.

RESULT



LIVE EDITOR

```
import Placeholder from 'react-bootstrap/Placeholder';

function AnimationExample() {
  return (
    <>
      <Placeholder as="p" animation="glow">
        <Placeholder xs={12} />
      </Placeholder>
      <Placeholder as="p" animation="wave">
        <Placeholder xs={12} />
      </Placeholder>
    </>
  );
}

export default AnimationExample;
```

API

Placeholder

```
import Placeholder from 'react-bootstrap/Placeholder'
```

Name	Type	Default	Description
bsPrefix	string	'placeholder'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
animation	'glow' | 'wave'		Changes the animation of the placeholder.
bg	'primary' | 'secondary' | 'success' | 'danger' | 'warning' | 'info' | 'light' | 'dark'		Change the background color of the placeholder.
size	'xs' | 'sm' | 'lg'		Component size variations.
as		'span'	You can use a custom element type for this component.

PlaceholderButton

```
import PlaceholderButton from 'react-bootstrap/PlaceholderButton'
```

Name	Type	Default	Description
bsPrefix	string	'placeholder'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
animation	'glow' | 'wave'		Changes the animation of the placeholder.
size	'xs' | 'sm' | 'lg'		Component size variations.
variant	string		Button variant.

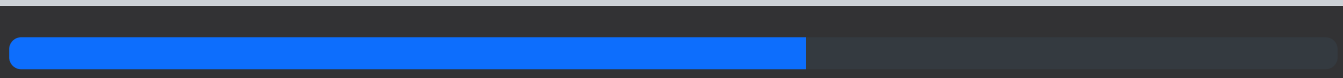
Progress bars

Provide up-to-date feedback on the progress of a workflow or action with simple yet flexible progress bars.

Example

Default progress bar.

RESULT



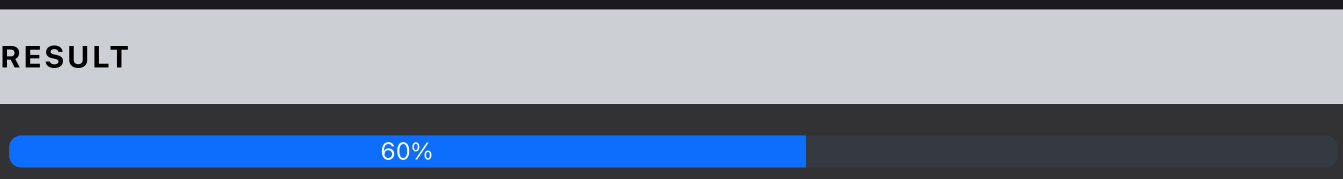
LIVE EDITOR

```jsx
import ProgressBar from 'react-bootstrap/ProgressBar';

function BasicExample() {
  return <ProgressBar now={60} />;
}

export default BasicExample;
```

With label

Add a `label` prop to show a visible percentage. For low percentages, consider adding a min-width to ensure the label's text is fully visible.

RESULT



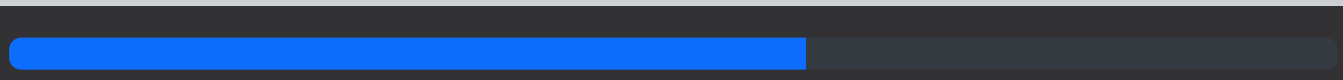
LIVE EDITOR

```jsx
import ProgressBar from 'react-bootstrap/ProgressBar';

function WithLabelExample() {
  const now = 60;
  return <ProgressBar now={now} label={`${now}%`} />;
}

export default WithLabelExample;
```

Screenreader only label

Add a `visuallyHidden` prop to hide the label visually.

RESULT



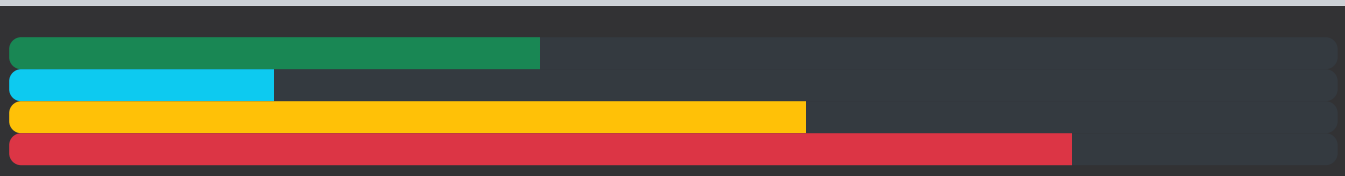
LIVE EDITOR

```
import ProgressBar from 'react-bootstrap/ProgressBar';

function ScreenreaderLabelExample() {
  const now = 60;
  return <ProgressBar now={now} label={`${now}%`} visuallyHidden />;
}

export default ScreenreaderLabelExample;
```

Contextual alternatives

Progress bars use some of the same button and alert classes for consistent styles.

RESULT



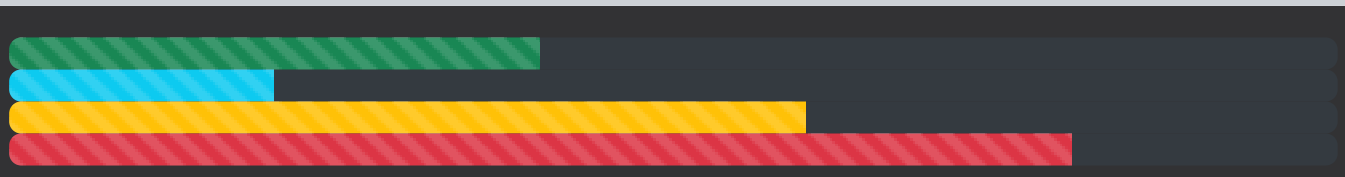
LIVE EDITOR

```
import ProgressBar from 'react-bootstrap/ProgressBar';

function ContextualExample() {
  return (
    <div>
      <ProgressBar variant="success" now={40} />
      <ProgressBar variant="info" now={20} />
      <ProgressBar variant="warning" now={60} />
      <ProgressBar variant="danger" now={80} />
    </div>
  );
}

export default ContextualExample;
```

Striped

Uses a gradient to create a striped effect.

RESULT



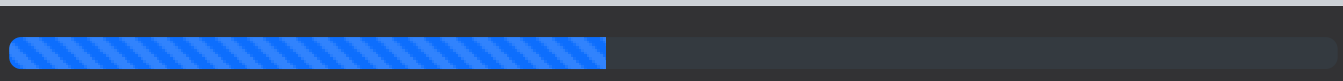
LIVE EDITOR

```
import ProgressBar from 'react-bootstrap/ProgressBar';

function StripedExample() {
  return (
    <div>
      <ProgressBar striped variant="success" now={40} />
      <ProgressBar striped variant="info" now={20} />
      <ProgressBar striped variant="warning" now={60} />
      <ProgressBar striped variant="danger" now={80} />
    </div>
  );
}

export default StripedExample;
```

Animated

Add `animated` prop to animate the stripes right to left.

RESULT



LIVE EDITOR

```jsx
import ProgressBar from 'react-bootstrap/ProgressBar';

function AnimatedExample() {
  return <ProgressBar animated now={45} />;
}

export default AnimatedExample;
```

Stacked

Nest `<ProgressBar />`s to stack them.

RESULT

LIVE EDITOR

```jsx
import ProgressBar from 'react-bootstrap/ProgressBar';

function StackedExample() {
  return (
    <ProgressBar>
      <ProgressBar striped variant="success" now={35} key={1} />
      <ProgressBar variant="warning" now={20} key={2} />
      <ProgressBar striped variant="danger" now={10} key={3} />
    </ProgressBar>
  );
}

export default StackedExample;
```

API

ProgressBar

```jsx
import ProgressBar from 'react-bootstrap/ProgressBar'
```

Name	Type	Default	Description
`min`	`number`		Minimum value progress can begin from
`now`	`number`		Current value of progress
`max`	`number`		Maximum value progress can reach
`label`	`node`		Show label that represents visual percentage. EG. 60%
`visuallyHidden`	`bool`		Hide's the label visually.

Name	Type	Default	Description
`striped`	`bool`		Uses a gradient to create a striped effect.
`animated`	`bool`		Animate's the stripes from right to left
`variant`	`'success'` | `'danger'` | `'warning'` | `'info'`		Sets the background class of the progress bar.
`children`	`custom`		Child elements (only allows elements of type)

Spinners

Spinners can be used to show the loading state in your projects.

Example

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function BasicExample() {
  return (
    <Spinner animation="border" role="status">
      <span className="visually-hidden">Loading...</span>
    </Spinner>
  );
}

export default BasicExample;
```

Animations

Bootstrap offers two animation styles for spinners. The animation style can be configured with the `animation` property. An animation style must always be provided when creating a spinner.

** Border Spinner - `border` **

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function BorderExample() {
  return <Spinner animation="border" />;
}

export default BorderExample;
```

** Grow Spinner - `grow` **

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function GrowExample() {
  return <Spinner animation="grow" />;
}

export default GrowExample;
```

Variants

All standard visual variants are available for both animation styles by setting the `variant` property. Alternatively spinners can be custom sized with the `style` property, or custom CSS classes.

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function VariantsExample() {
  return (
    <>
      <Spinner animation="border" variant="primary" />
      <Spinner animation="border" variant="secondary" />
      <Spinner animation="border" variant="success" />
      <Spinner animation="border" variant="danger" />
      <Spinner animation="border" variant="warning" />
      <Spinner animation="border" variant="info" />
      <Spinner animation="border" variant="light" />
      <Spinner animation="border" variant="dark" />
      <Spinner animation="grow" variant="primary" />
      <Spinner animation="grow" variant="secondary" />
      <Spinner animation="grow" variant="success" />
      <Spinner animation="grow" variant="danger" />
      <Spinner animation="grow" variant="warning" />
      <Spinner animation="grow" variant="info" />
      <Spinner animation="grow" variant="light" />
      <Spinner animation="grow" variant="dark" />
    </>
  );
}

export default VariantsExample;
```

Sizing

In addition to the standard size, a smaller additional preconfigured size is available by configuring the `size` property to `sm`.

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function SizesExample() {
  return (
    <>
      <Spinner animation="border" size="sm" />
      <Spinner animation="border" />
      <Spinner animation="grow" size="sm" />
      <Spinner animation="grow" />
    </>
  );
}

export default SizesExample;
```

Buttons

Like the original Bootstrap spinners, these can also be used with buttons. To use this component out-of-the-box it is recommended you change the element type to `span` by configuring the `as` property when using spinners inside buttons.

RESULT

Loading...

LIVE EDITOR

```jsx
import Button from 'react-bootstrap/Button';
import Spinner from 'react-bootstrap/Spinner';

function ButtonExample() {
  return (
    <>
      <Button variant="primary" disabled>
        <Spinner
          as="span"
          animation="border"
          size="sm"
          role="status"
          aria-hidden="true"
        />
        <span className="visually-hidden">Loading...</span>
      </Button>{' '}
      <Button variant="primary" disabled>
        <Spinner
          as="span"
          animation="grow"
          size="sm"
          role="status"
          aria-hidden="true"
        />
        Loading...
      </Button>
    </>
  );
}

export default ButtonExample;
```

Accessibility

To ensure the maximum accessibility for spinner components it is recommended you provide a relevant ARIA `role` property, and include screenreader-only readable text representation of the spinner's meaning inside the component using Bootstrap's `visually-hidden` class.

The example below provides an example of accessible usage of this component.

RESULT

LIVE EDITOR

```jsx
import Spinner from 'react-bootstrap/Spinner';

function BasicExample() {
  return (
    <Spinner animation="border" role="status">
      <span className="visually-hidden">Loading...</span>
    </Spinner>
  );
}

export default BasicExample;
```

API

Spinner

```jsx
import Spinner from 'react-bootstrap/Spinner'
```

Name	Type	Default	Description
bsPrefix	string	`'spinner'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
variant	`'primary'` | `'secondary'` | `'success'` | `'danger'` | `'warning'` | `'info'` | `'light'` | `'dark'`		The visual color style of the spinner
animation	`'border'` | `'grow'`	`true`	Changes the animation style of the spinner.
size	`'sm'`		Component size variations.
children	element		This component may be used to wrap child elements or components.
role	string		An ARIA accessible role applied to the Menu component. This should generally be set to 'status'
as	elementType	`div`	You can use a custom element type for this component.

Tables

Example

Use the `striped`, `bordered` and `hover` props to customise the table.

RESULT

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

LIVE EDITOR

```jsx
import Table from 'react-bootstrap/Table';

function BasicExample() {
  return (
    <Table striped bordered hover>
      <thead>
        <tr>
          <th>#</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Username</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Mark</td>
          <td>Otto</td>
          <td>@mdo</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Jacob</td>
          <td>Thornton</td>
          <td>@fat</td>
        </tr>
        <tr>
          <td>3</td>
          <td colSpan={2}>Larry the Bird</td>
          <td>@twitter</td>
        </tr>
      </tbody>
    </Table>
  );
}

export default BasicExample;
```

Small Table

Use `size="sm"` to make tables compact by cutting cell padding in half.

RESULT

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';

function SmallExample() {
  return (
    <Table striped bordered hover size="sm">
      <thead>
        <tr>
          <th>#</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Username</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Mark</td>
          <td>Otto</td>
          <td>@mdo</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Jacob</td>
          <td>Thornton</td>
          <td>@fat</td>
        </tr>
        <tr>
          <td>3</td>
          <td colSpan={2}>Larry the Bird</td>
          <td>@twitter</td>
        </tr>
      </tbody>
    </Table>
  );
}

export default SmallExample;
```

Dark Table

Use `variant="dark"` to invert the colors of the table and get light text on a dark background.

RESULT

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat

#	First Name	Last Name	Username
3	Larry the Bird		@twitter

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';

function DarkExample() {
  return (
    <Table striped bordered hover variant="dark">
      <thead>
        <tr>
          <th>#</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Username</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Mark</td>
          <td>Otto</td>
          <td>@mdo</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Jacob</td>
          <td>Thornton</td>
          <td>@fat</td>
        </tr>
        <tr>
          <td>3</td>
          <td colSpan={2}>Larry the Bird</td>
          <td>@twitter</td>
        </tr>
      </tbody>
    </Table>
  );
}

export default DarkExample;
```

Striped rows

Use `striped` to add zebra-striping to any table row within the table.

RESULT

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';

function StripedRowExample() {
  return (
    <Table striped>
      <thead>
```

```
        <tr>
          <th>#</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Username</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Mark</td>
          <td>Otto</td>
          <td>@mdo</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Jacob</td>
          <td>Thornton</td>
          <td>@fat</td>
        </tr>
        <tr>
          <td>3</td>
          <td colSpan={2}>Larry the Bird</td>
          <td>@twitter</td>
        </tr>
      </tbody>
    </Table>
  );
}

export default StripedRowExample;
```

Striped columns

Use `striped="columns"` to add zebra-striping to any table column.

RESULT

#	First Name	Last Name	Username
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';

function StripedColumnsExample() {
  return (
    <Table striped="columns">
      <thead>
        <tr>
          <th>#</th>
          <th>First Name</th>
          <th>Last Name</th>
          <th>Username</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Mark</td>
          <td>Otto</td>
          <td>@mdo</td>
```

```
        </tr>
        <tr>
          <td>2</td>
          <td>Jacob</td>
          <td>Thornton</td>
          <td>@fat</td>
        </tr>
        <tr>
          <td>3</td>
          <td colSpan={2}>Larry the Bird</td>
          <td>@twitter</td>
        </tr>
      </tbody>
    </Table>
  );
}

export default StripedColumnsExample;
```

Responsive

Responsive tables allow tables to be scrolled horizontally with ease.

Always Responsive

Across every breakpoint, use `responsive` for horizontally scrolling tables. Responsive tables are wrapped automatically in a `div`. The following example has 12 columns that are scrollable horizontally.

RESULT

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell 0	Table cell 1	Table cell 2	Table cell 3	Table cell 4	Table cell 5	Table cell 6	Table cell 7
2	Table cell 0	Table cell 1	Table cell 2	Table cell 3	Table cell 4	Table cell 5	Table cell 6	Table cell 7
3	Table cell 0	Table cell 1	Table cell 2	Table cell 3	Table cell 4	Table cell 5	Table cell 6	Table cell 7

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';

function ResponsiveExample() {
  return (
    <Table responsive>
      <thead>
        <tr>
          <th>#</th>
          {Array.from({ length: 12 }).map((_, index) => (
            <th key={index}>Table heading</th>
          ))}
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          {Array.from({ length: 12 }).map((_, index) => (
            <td key={index}>Table cell {index}</td>
          ))}
        </tr>
```

```
        <tr>
          <td>2</td>
          {Array.from({ length: 12 }).map((_, index) => (
            <td key={index}>Table cell {index}</td>
          ))}
        </tr>
        <tr>
          <td>3</td>
          {Array.from({ length: 12 }).map((_, index) => (
            <td key={index}>Table cell {index}</td>
          ))}
        </tr>
      </tbody>
    </Table>
  );
}

export default ResponsiveExample;
```

Breakpoint specific

Use `responsive="sm"`, `responsive="md"`, `responsive="lg"`, or `responsive="xl"` as needed to create responsive tables up to a particular breakpoint. From that breakpoint and up, the table will behave normally and not scroll horizontally.

RESULT

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

#	Table heading	Table heading	Table heading	Table heading	Table heading	Table heading
1	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
2	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell
3	Table cell	Table cell	Table cell	Table cell	Table cell	Table cell

LIVE EDITOR

```
import Table from 'react-bootstrap/Table';
```

```jsx
function ResponsiveBreakpointsExample() {
  return (
    <div>
      <Table responsive="sm">
        <thead>
          <tr>
            <th>#</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>1</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
          <tr>
            <td>2</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
          <tr>
            <td>3</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
        </tbody>
      </Table>
      <Table responsive="md">
        <thead>
          <tr>
            <th>#</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
            <th>Table heading</th>
          </tr>
        </thead>
        <tbody>
          <tr>
            <td>1</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
          <tr>
            <td>2</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
```

```
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
        </tr>
        <tr>
          <td>3</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
        </tr>
      </tbody>
    </Table>
    <Table responsive="lg">
      <thead>
        <tr>
          <th>#</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
        </tr>
        <tr>
          <td>2</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
        </tr>
        <tr>
          <td>3</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
          <td>Table cell</td>
        </tr>
      </tbody>
    </Table>
    <Table responsive="xl">
      <thead>
        <tr>
          <th>#</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
          <th>Table heading</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>1</td>
```

```
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
          <tr>
            <td>2</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
          <tr>
            <td>3</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
            <td>Table cell</td>
          </tr>
        </tbody>
      </Table>
    </div>
  );
}

export default ResponsiveBreakpointsExample;
```

API

Table

```
import Table from 'react-bootstrap/Table'
```

Name	Type	Default	Description
bsPrefix	string	`'table'`	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
striped	`bool` | `string`		Adds zebra-striping to any table row within the `<tbody>`. Use `columns` to add zebra-striping to any table column.
bordered	bool		Adds borders on all sides of the table and cells.
borderless	bool		Removes all borders on the table and cells, including table header.
hover	bool		Enable a hover state on table rows within a `<tbody>`.
size	string		Make tables more compact by cutting cell padding in half by setting size as `sm`.
variant	string		Invert the colors of the table — with light text on dark backgrounds by setting variant as `dark`.
responsive	`bool` | `string`		Responsive tables allow tables to be scrolled horizontally with ease. Across every breakpoint, use

Name	Type	Default	Description
			`responsive` for horizontally scrolling tables. Responsive tables are wrapped automatically in a `div`. Use `responsive="sm"`, `responsive="md"`, `responsive="lg"`, or `responsive="xl"` as needed to create responsive tables up to a particular breakpoint. From that breakpoint and up, the table will behave normally and not scroll horizontally.

Tabbed components

Dynamic tabbed interfaces

Examples

Create dynamic tabbed interfaces, as described in the WAI ARIA Authoring Practices. `Tabs` is a higher-level component for quickly creating a `Nav` matched with a set of `TabPane`s.

RESULT

Home | Profile | Contact

Tab content for Profile

LIVE EDITOR

```jsx
import Tab from 'react-bootstrap/Tab';
import Tabs from 'react-bootstrap/Tabs';

function UncontrolledExample() {
  return (
    <Tabs
      defaultActiveKey="profile"
      id="uncontrolled-tab-example"
      className="mb-3"
    >
      <Tab eventKey="home" title="Home">
        Tab content for Home
      </Tab>
      <Tab eventKey="profile" title="Profile">
        Tab content for Profile
      </Tab>
      <Tab eventKey="contact" title="Contact" disabled>
        Tab content for Contact
      </Tab>
    </Tabs>
  );
}

export default UncontrolledExample;
```

Controlled

`Tabs` can be controlled directly when you want to handle the selection logic personally.

RESULT

Home | Profile | Contact

Tab content for Home

LIVE EDITOR

```
import { useState } from 'react';
import Tab from 'react-bootstrap/Tab';
import Tabs from 'react-bootstrap/Tabs';

function ControlledTabsExample() {
  const [key, setKey] = useState('home');

  return (
    <Tabs
      id="controlled-tab-example"
      activeKey={key}
      onSelect={(k) => setKey(k)}
      className="mb-3"
    >
      <Tab eventKey="home" title="Home">
        Tab content for Home
      </Tab>
      <Tab eventKey="profile" title="Profile">
        Tab content for Profile
      </Tab>
      <Tab eventKey="contact" title="Contact" disabled>
        Tab content for Contact
      </Tab>
    </Tabs>
  );
}

export default ControlledTabsExample;
```

No animation

Set the `transition` prop to `false`.

RESULT

Home Profile Contact

Tab content for Home

LIVE EDITOR

```
import Tab from 'react-bootstrap/Tab';
import Tabs from 'react-bootstrap/Tabs';

function NoAnimationExample() {
  return (
    <Tabs
      defaultActiveKey="home"
      transition={false}
      id="noanim-tab-example"
      className="mb-3"
    >
      <Tab eventKey="home" title="Home">
        Tab content for Home
      </Tab>
      <Tab eventKey="profile" title="Profile">
        Tab content for Profile
      </Tab>
      <Tab eventKey="contact" title="Contact" disabled>
        Tab content for Contact
      </Tab>
    </Tabs>
  );
}
```

```
export default NoAnimationExample;
```

Fill and justify

Similar to the `Nav` component, you can force the contents of your `Tabs` to extend the full available width. To proportionately fill the space use `fill`. Notice that the `Tabs` is the entire width but each `Tab` item is a different size.

RESULT

Home | Profile | Loooonger Tab | Contact

Tab content for Profile

LIVE EDITOR

```
import Tab from 'react-bootstrap/Tab';
import Tabs from 'react-bootstrap/Tabs';

function FillExample() {
  return (
    <Tabs
      defaultActiveKey="profile"
      id="fill-tab-example"
      className="mb-3"
      fill
    >
      <Tab eventKey="home" title="Home">
        Tab content for Home
      </Tab>
      <Tab eventKey="profile" title="Profile">
        Tab content for Profile
      </Tab>
      <Tab eventKey="longer-tab" title="Loooonger Tab">
        Tab content for Loooonger Tab
      </Tab>
      <Tab eventKey="contact" title="Contact" disabled>
        Tab content for Contact
      </Tab>
    </Tabs>
  );
}

export default FillExample;
```

If you want each `Tab` to be the same size use `justify`.

RESULT

Home | Profile | Loooonger Tab | Contact

Tab content for Profile

LIVE EDITOR

```
import Tab from 'react-bootstrap/Tab';
import Tabs from 'react-bootstrap/Tabs';

function JustifiedExample() {
  return (
    <Tabs
      defaultActiveKey="profile"
      id="justify-tab-example"
      className="mb-3"
      justify
```

```
    >
      <Tab eventKey="home" title="Home">
        Tab content for Home
      </Tab>
      <Tab eventKey="profile" title="Profile">
        Tab content for Profile
      </Tab>
      <Tab eventKey="longer-tab" title="Loooonger Tab">
        Tab content for Loooonger Tab
      </Tab>
      <Tab eventKey="contact" title="Contact" disabled>
        Tab content for Contact
      </Tab>
    </Tabs>
  );
}

export default JustifiedExample;
```

Dropdowns?

Dynamic tabbed interfaces should not contain dropdown menus, as this causes both usability and accessibility issues. From a usability perspective, the fact that the currently displayed tab's trigger element is not immediately visible (as it's inside the closed dropdown menu) can cause confusion. From an accessibility point of view, there is currently no sensible way to map this sort of construct to a standard WAI ARIA pattern, meaning that it cannot be easily made understandable to users of assistive technologies.

That said, it Dropdowns do work technically (sans focus management), but we don't make any claims about support.

Custom Tab Layout

For more complex layouts the flexible `TabContainer`,

`TabContent`, and `TabPane` components along with any style of `Nav` allow you to quickly piece together your own Tabs component with additional markup needed.

Create a set of NavItems each with an `eventKey` corresponding to the eventKey of a `TabPane`. Wrap the whole thing in a `TabContainer` and you have fully functioning custom tabs component. Check out the below example making use of the grid system, pills and underline.

Pills

RESULT

Tab 1

Tab 2

First tab content

LIVE EDITOR

```
import Col from 'react-bootstrap/Col';
import Nav from 'react-bootstrap/Nav';
import Row from 'react-bootstrap/Row';
import Tab from 'react-bootstrap/Tab';

function LeftTabsExample() {
  return (
    <Tab.Container id="left-tabs-example" defaultActiveKey="first">
      <Row>
        <Col sm={3}>
          <Nav variant="pills" className="flex-column">
            <Nav.Item>
```

```
            <Nav.Link eventKey="first">Tab 1</Nav.Link>
          </Nav.Item>
          <Nav.Item>
            <Nav.Link eventKey="second">Tab 2</Nav.Link>
          </Nav.Item>
        </Nav>
      </Col>
      <Col sm={9}>
        <Tab.Content>
          <Tab.Pane eventKey="first">First tab content</Tab.Pane>
          <Tab.Pane eventKey="second">Second tab content</Tab.Pane>
        </Tab.Content>
      </Col>
    </Row>
  </Tab.Container>
);
}

export default LeftTabsExample;
```

API

Tabs

```
import Tabs from 'react-bootstrap/Tabs'
```

Name	Type	Default	Description
activeKey	`string` | `number`		*controlled by:* `onSelect`, *initial prop:* `defaultActiveKey` Mark the Tab with a matching `eventKey` as active.
defaultActiveKey	`string` | `number`		The default active key that is selected on start
variant	`'tabs'` | `'pills'` | `'underline'`		Navigation style
transition	`Transition` | `false`	`{Fade}`	Sets a default animation strategy for all children `<TabPane>`s.<tbcont Defaults to `<Fade>` animation, else use `false` to disable or a react-transition-group `<Transition/>` component.
id	`string`		HTML id attribute, required if no `generateChildId` prop is specified.
onSelect	`func`		*controls* `activeKey` Callback fired when a Tab is selected. `function ( Any eventKey, SyntheticEvent event? )`
mountOnEnter	`bool`		Wait until the first "enter" transition to mount tabs (add them to the DOM)
unmountOnExit	`bool`		Unmount tabs (remove it from the DOM) when it is no longer visible

Name	Type	Default	Description
`fill`	`bool`		Have all `Tabs`s proportionately fill all available width.
`justify`	`bool`		Have all `Tab`s evenly fill all available width.

Tab

TabContainer

```
import TabContainer from 'react-bootstrap/TabContainer'
```

Name	Type	Default	Description
`id`	`string`		HTML id attribute, required if no `generateChildId` prop is specified.
`transition`	`{Transition | false}`	`{Fade}`	Sets a default animation strategy for all children `<TabPane>`s. Defaults to `<Fade>` animation; else, use `false` to disable, or a custom react-transition-group `<Transition/>` component.
`mountOnEnter`	`bool`		Wait until the first "enter" transition to mount tabs (add them to the DOM)
`unmountOnExit`	`bool`		Unmount tabs (remove it from the DOM) when they are no longer visible
`generateChildId`	`func`		A function that takes an `eventKey` and `type` and returns a unique id for child tab `<NavItem>`s and `<TabPane>`s. The function *must* be a pure function, meaning it should always return the *same* id for the same set of inputs. The default value requires that an `id` to be set for the `<TabContainer>`. The `type` argument will either be `"tab"` or `"pane"`.
`onSelect`	`func`		*controls `activeKey`* A callback fired when a tab is selected.
`activeKey`	`string | number`		*controlled by: `onSelect`, initial prop: `defaultActiveKey`* The `eventKey` of the currently active tab.

TabContent

```
import TabContent from 'react-bootstrap/TabContent'
```

Name	Type	Default	Description
`as`		`'div'`	You can use a custom element type for this component.

TabPane

```
import TabPane from 'react-bootstrap/TabPane'
```

Name	Type	Default	Description
bsPrefix	string	'tab-pane'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
as	elementType		You can use a custom element type for this component.
eventKey	string | number		A key that associates the TabPane with it's controlling NavLink.
active	bool		Toggles the active state of the TabPane, this is generally controlled by a TabContainer.
transition	bool | elementType		Use animation when showing or hiding <TabPane>s. Defaults to <Fade> animation, else use false to disable or a react-transition-group <Transition/> component.
onEnter	func		Transition onEnter callback when animation is not false
onEntering	func		Transition onEntering callback when animation is not false
onEntered	func		Transition onEntered callback when animation is not false
onExit	func		Transition onExit callback when animation is not false
onExiting	func		Transition onExiting callback when animation is not false
onExited	func		Transition onExited callback when animation is not false
mountOnEnter	bool		Wait until the first "enter" transition to mount the tab (add it to the DOM)
unmountOnExit	bool		Unmount the tab (remove it from the DOM) when it is no longer visible
id	string		
aria-labelledby	string		

Toasts

Push notifications to your visitors with a toast, a lightweight and easily customizable alert message.

Toasts are lightweight notifications designed to mimic the push notifications that have been popularized by mobile and desktop operating systems. They're built with flexbox, so they're easy to align and position.

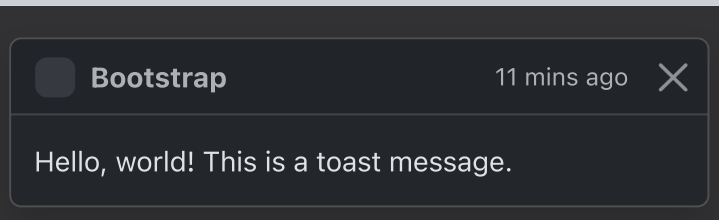
Examples

Basic

To encourage extensible and predictable toasts, we recommend a header and body. Toast headers use display: flex, allowing easy alignment of content thanks to our margin and flexbox utilities.

Toasts are as flexible as you need and have very little required markup. At a minimum, we require a single element to contain your "toasted" content and strongly encourage a dismiss button.

RESULT

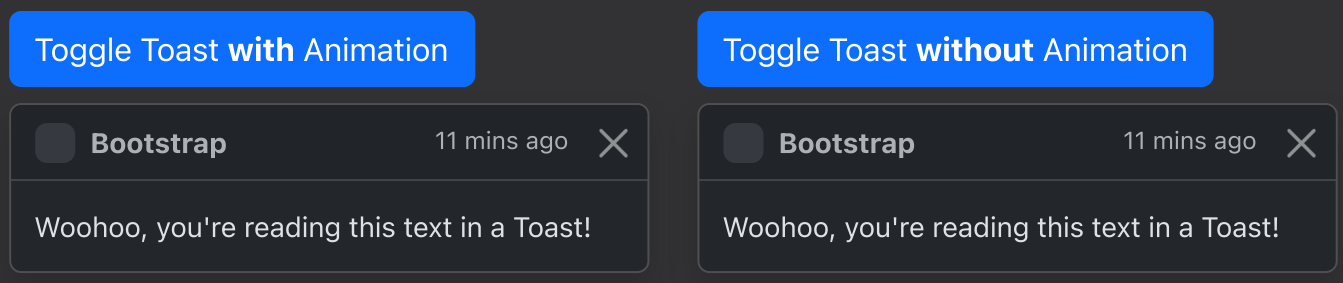


LIVE EDITOR

```jsx
import Toast from 'react-bootstrap/Toast';

function BasicExample() {
  return (
    <Toast>
      <Toast.Header>
        <img src="holder.js/20x20?text=%20" className="rounded me-2"
alt="" />
        <strong className="me-auto">Bootstrap</strong>
        <small>11 mins ago</small>
      </Toast.Header>
      <Toast.Body>Hello, world! This is a toast message.</Toast.Body>
    </Toast>
  );
}

export default BasicExample;
```

Dismissible

RESULT



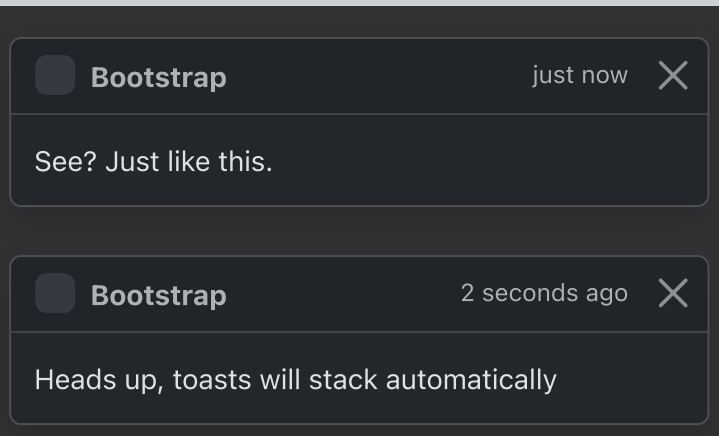
LIVE EDITOR

```jsx
import React, { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Row from 'react-bootstrap/Row';
import Toast from 'react-bootstrap/Toast';

function DismissibleExample() {
  const [showA, setShowA] = useState(true);
  const [showB, setShowB] = useState(true);

  const toggleShowA = () => setShowA(!showA);
  const toggleShowB = () => setShowB(!showB);

  return (
    <Row>
      <Col md={6} className="mb-2">
        <Button onClick={toggleShowA} className="mb-2">
          Toggle Toast <strong>with</strong> Animation
        </Button>
        <Toast show={showA} onClose={toggleShowA}>
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small>11 mins ago</small>
          </Toast.Header>
          <Toast.Body>Woohoo, you're reading this text in a Toast!
</Toast.Body>
        </Toast>
      </Col>
      <Col md={6} className="mb-2">
        <Button onClick={toggleShowB} className="mb-2">
          Toggle Toast <strong>without</strong> Animation
        </Button>
        <Toast onClose={toggleShowB} show={showB} animation={false}>
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small>11 mins ago</small>
          </Toast.Header>
          <Toast.Body>Woohoo, you're reading this text in a Toast!
</Toast.Body>
        </Toast>
      </Col>
    </Row>
  );
}

export default DismissibleExample;
```

Stacking

When you have multiple toasts, we default to vertically stacking them in a readable manner.

RESULT



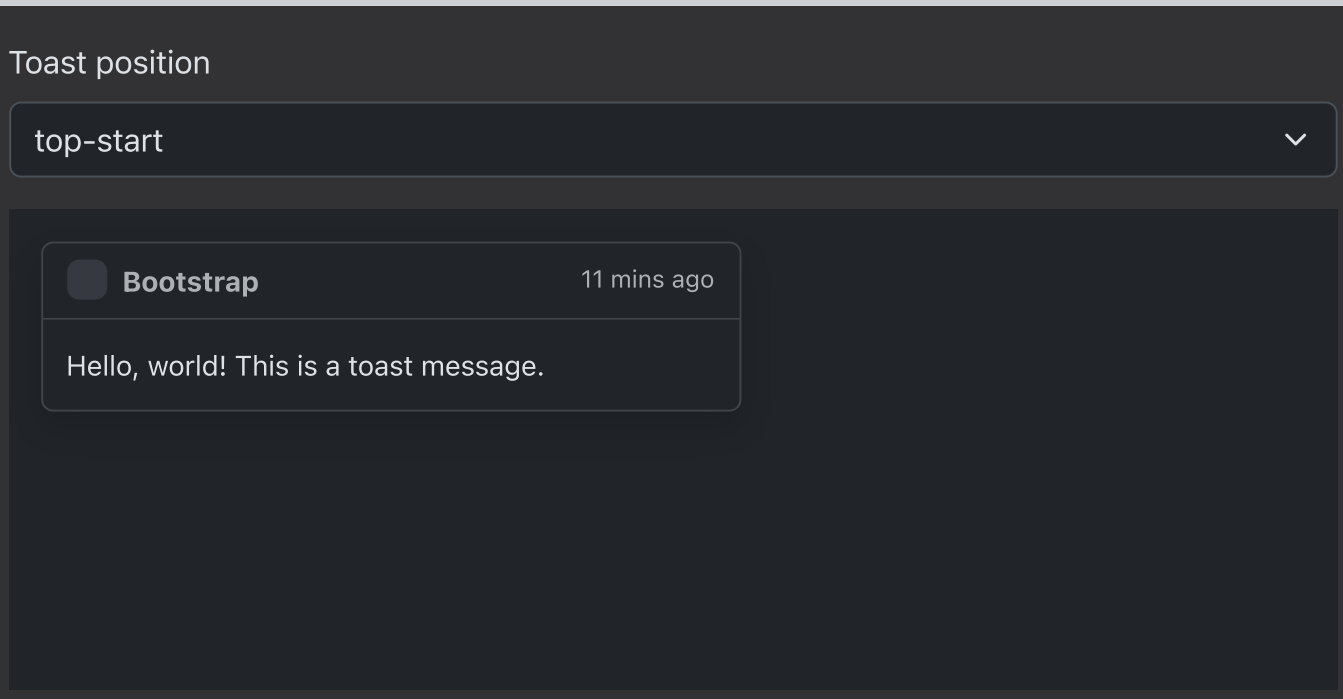
LIVE EDITOR

```jsx
import Toast from 'react-bootstrap/Toast';
import ToastContainer from 'react-bootstrap/ToastContainer';

function StackingExample() {
  return (
    <ToastContainer className="position-static">
      <Toast>
        <Toast.Header>
          <img src="holder.js/20x20?text=%20" className="rounded me-2"
alt="" />
          <strong className="me-auto">Bootstrap</strong>
          <small className="text-muted">just now</small>
        </Toast.Header>
        <Toast.Body>See? Just like this.</Toast.Body>
      </Toast>
      <Toast>
        <Toast.Header>
          <img src="holder.js/20x20?text=%20" className="rounded me-2"
alt="" />
          <strong className="me-auto">Bootstrap</strong>
          <small className="text-muted">2 seconds ago</small>
        </Toast.Header>
        <Toast.Body>Heads up, toasts will stack
automatically</Toast.Body>
      </Toast>
    </ToastContainer>
  );
}

export default StackingExample;
```

Placement

Place toasts by setting a `position` in a `ToastContainer`. The top right is often used for notifications, as is the top middle.

RESULT

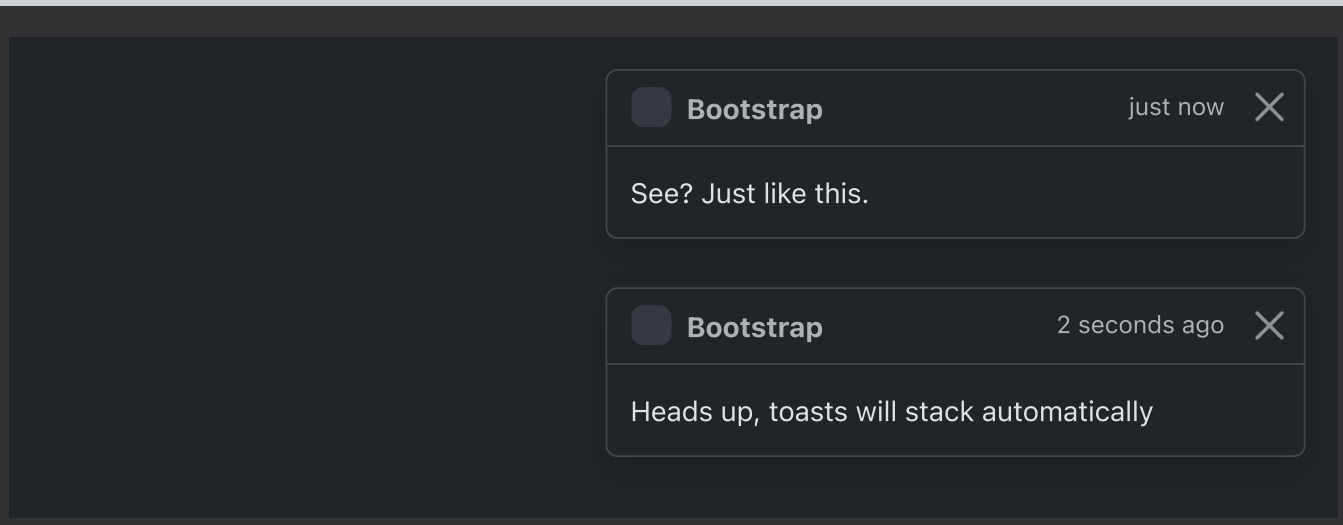


LIVE EDITOR

```jsx
import { useState } from 'react';
import Form from 'react-bootstrap/Form';
import Toast from 'react-bootstrap/Toast';
import ToastContainer from 'react-bootstrap/ToastContainer';

function PlacementExample() {
  const [position, setPosition] = useState('top-start');

  return (
    <>
      <div className="mb-3">
        <label htmlFor="selectToastPlacement">Toast position</label>
        <Form.Select
          id="selectToastPlacement"
          className="mt-2"
          onChange={(e) => setPosition(e.currentTarget.value)}
        >
          {[
            'top-start',
            'top-center',
            'top-end',
            'middle-start',
            'middle-center',
            'middle-end',
            'bottom-start',
            'bottom-center',
            'bottom-end',
          ].map((p) => (
            <option key={p} value={p}>
              {p}
            </option>
          ))}
        </Form.Select>
      </div>

      <div
        aria-live="polite"
        aria-atomic="true"
        className="bg-dark position-relative"
        style={{ minHeight: '240px' }}
      >
        <ToastContainer
          className="p-3"
          position={position}
          style={{ zIndex: 1 }}
        >
          <Toast>
            <Toast.Header closeButton={false}>
              <img
                src="holder.js/20x20?text=%20"
                className="rounded me-2"
                alt=""
              />
              <strong className="me-auto">Bootstrap</strong>
              <small>11 mins ago</small>
            </Toast.Header>
            <Toast.Body>Hello, world! This is a toast message.
</Toast.Body>
          </Toast>
        </ToastContainer>
      </div>
    </>
  );
}

export default PlacementExample;
```

For systems that generate more notifications, consider using a wrapping element so they can easily stack.

RESULT



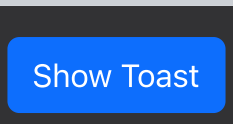
LIVE EDITOR

```jsx
import Toast from 'react-bootstrap/Toast';
import ToastContainer from 'react-bootstrap/ToastContainer';

function PlacementMultiExample() {
  return (
    <div
      aria-live="polite"
      aria-atomic="true"
      className="bg-dark position-relative"
      style={{ minHeight: '240px' }}
    >
      <ToastContainer position="top-end" className="p-3" style={{ zIndex: 1 }}>
        <Toast>
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small className="text-muted">just now</small>
          </Toast.Header>
          <Toast.Body>See? Just like this.</Toast.Body>
        </Toast>
        <Toast>
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small className="text-muted">2 seconds ago</small>
          </Toast.Header>
          <Toast.Body>Heads up, toasts will stack automatically</Toast.Body>
        </Toast>
      </ToastContainer>
    </div>
  );
}

export default PlacementMultiExample;
```

Autohide

A Toast can also automatically hide after X milliseconds using the `autohide` prop with the `delay` prop to specify the delay. To open the toast, manually change the show property.

RESULT



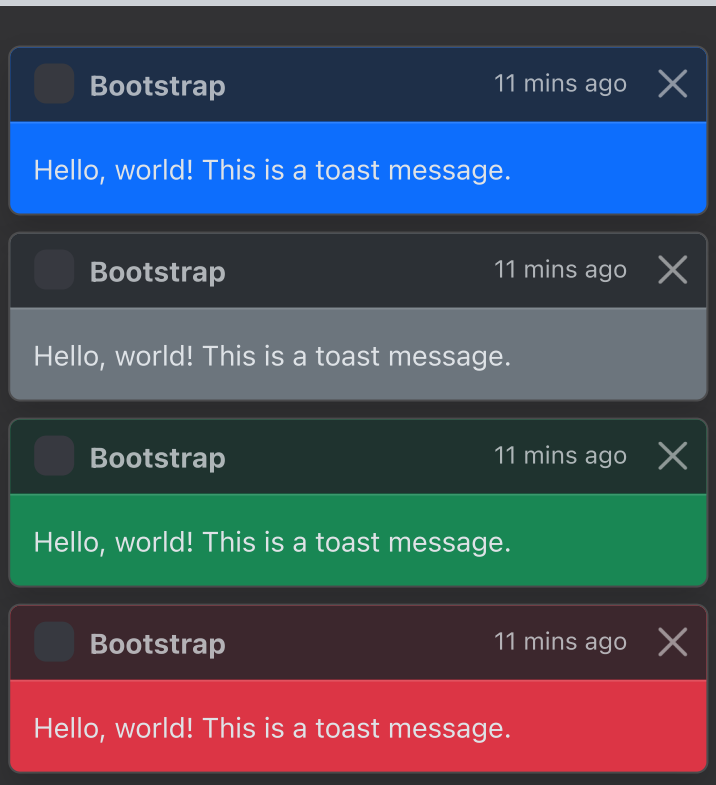
LIVE EDITOR

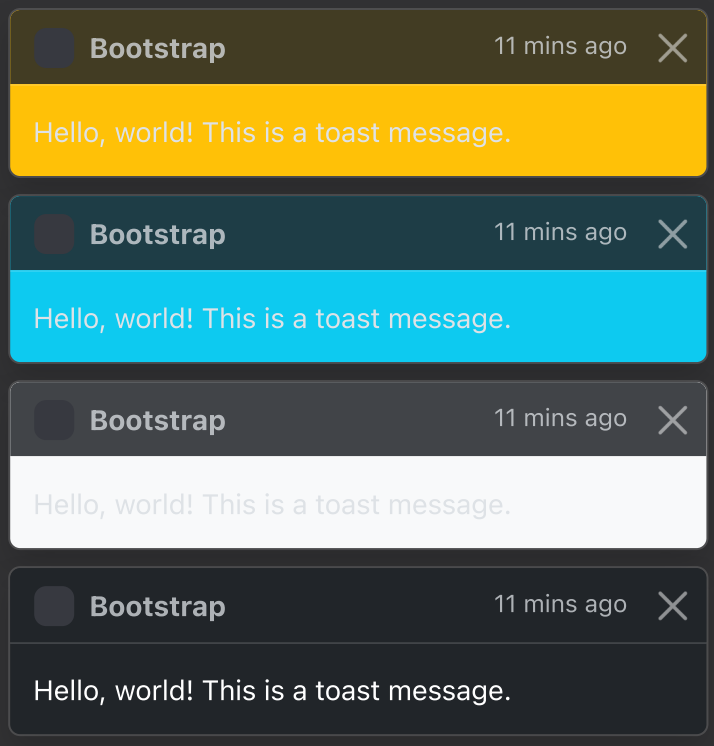
```jsx
import React, { useState } from 'react';
import Button from 'react-bootstrap/Button';
import Col from 'react-bootstrap/Col';
import Row from 'react-bootstrap/Row';
import Toast from 'react-bootstrap/Toast';

function AutohideExample() {
  const [show, setShow] = useState(false);

  return (
    <Row>
      <Col xs={6}>
        <Toast onClose={() => setShow(false)} show={show} delay={3000}
autohide>
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small>11 mins ago</small>
          </Toast.Header>
          <Toast.Body>Woohoo, you're reading this text in a Toast!
</Toast.Body>
        </Toast>
      </Col>
      <Col xs={6}>
        <Button onClick={() => setShow(true)}>Show Toast</Button>
      </Col>
    </Row>
  );
}

export default AutohideExample;
```

Contextual variations

Add any of the below mentioned modifier classes to change the appearance of a toast.

RESULT





LIVE EDITOR

```jsx
import Toast from 'react-bootstrap/Toast';

function ContextualExample() {
  return (
    <>
      {[
        'Primary',
        'Secondary',
        'Success',
        'Danger',
        'Warning',
        'Info',
        'Light',
        'Dark',
      ].map((variant, idx) => (
        <Toast
          className="d-inline-block m-1"
          bg={variant.toLowerCase()}
          key={idx}
        >
          <Toast.Header>
            <img
              src="holder.js/20x20?text=%20"
              className="rounded me-2"
              alt=""
            />
            <strong className="me-auto">Bootstrap</strong>
            <small>11 mins ago</small>
          </Toast.Header>
          <Toast.Body className={variant === 'Dark' && 'text-white'}>
            Hello, world! This is a toast message.
          </Toast.Body>
        </Toast>
      ))}
    </>
  );
}

export default ContextualExample;
```

API

Toast

```jsx
import Toast from 'react-bootstrap/Toast'
```

Name	Type	Default	Description
bsPrefix	string	'toast'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
animation	bool	true	Apply a CSS fade transition to the toast
autohide	bool	false	Auto hide the toast
delay	number	5000	Delay hiding the toast (ms)
onClose	func		A Callback fired when the close button is clicked.
onEnter	func		Callback fired before the toast transitions in
onEntering	func		Callback fired as the toast begins to transition in
onEntered	func		Callback fired after the toast finishes transitioning in
onExit	func		Transition onExit callback when animation is not `false`
onExiting	func		Transition onExiting callback when animation is not `false`
onExited	func		Transition onExited callback when animation is not `false`
show	bool	true	When `true` The toast will show itself.
transition	elementType	ToastFade	A `react-transition-group` Transition component used to animate the Toast on dismissal.
bg	'primary' | 'secondary' | 'success' | 'danger' | 'warning' | 'info' | 'dark' | 'light'		Sets Toast background

ToastHeader

```
import ToastHeader from 'react-bootstrap/ToastHeader'
```

Name	Type	Default	Description
bsPrefix	string		Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
closeLabel	string	'Close'	Provides an accessible label for the close button. It is used for Assistive Technology when the label text is not readable.
closeVariant	'white'		Sets the variant for close button.

Name	Type	Default	Description
closeButton	bool	true	Specify whether the Component should contain a close button

ToastBody

```
import ToastBody from 'react-bootstrap/ToastBody'
```

Name	Type	Default	Description
as		'div'	You can use a custom element type for this component.

ToastContainer

```
import ToastContainer from 'react-bootstrap/ToastContainer'
```

Name	Type	Default	Description
bsPrefix	string	'toast-container'	Change the underlying component CSS base class name and modifier class names prefix. **This is an escape hatch** for working with heavily customized bootstrap css.
position	'top-start' | 'top-center' | 'top-end' | 'middle-start' | 'middle-center' | 'middle-end' | 'bottom-start' | 'bottom-center' | 'bottom-end'		Where the toasts will be placed within the container.
containerPosition	string		Specify the positioning method for the container.
as		'div'	You can use a custom element type for this component.