Laboratorul 12: React Bootstrap

React Bootstrap

Bootstrap [https://getbootstrap.com/] reprezinta un framework open-source de CSS folosit pe scara larga, construit pe principii precum responsiveness, mobile-first, simplitate si flexibilitate. Adaugat la un proiect, acesta ofera din start un styling de baza pentru toate elementele HTML dintr-o pagina (de acolo denumirea de "bootstrap"). Instalarea lui presupune legarea unui stylesheet in head-ul paginilor, plus asigurarea existentei unor biblioteci JS in proiect (e.g. jQuery).

React Bootstrap [https://react-bootstrap.github.io/] este o reimplementare a componentelor de Bootstrap folosing React, rezultand, astfel, componente React, fara a mai fi necesare biblioteci externe pentru functionarea lor.

Pentru a instala React Bootstrap trebuie sa rulam urmatoarea comanda:

```
npm install react-bootstrap bootstrap
```

Legarea CSS-ului se realizeaza, in fisierul App.js sau index.js, astfel:

```
import 'bootstrap/dist/css/bootstrap.min.css';
```

Pentru a facilita incarcarea mai rapida a CSS-ului, precum si asigurarea celei mai recente versiuni, se poate utiliza varianta de pe un Content Delivery Network (CDN):

```
<link
  rel="stylesheet"
  href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
  integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
  crossorigin="anonymous"
/>
```

Bootstrap Grid Layout

Pentru a construi cu usurinta un layout responsive, Bootstrap pune la dispozitie un grid flexbox. Acesta vine cu un sistem de containere, randuri si coloane, pentru a aranja continutul in pagina. Pe fiecare rand pot fi asezate un numar de pana la 12 coloane, cu dimensiuni variabile, in functie de setarile specificate si de device-ul pe care sunt afisate. Sunt disponibile, by default, optiuni pentru 5 clase de device-uri: xs (<576px), sm (≥576px), md (≥768px), lg (≥992px), xl (≥1200px).

Container

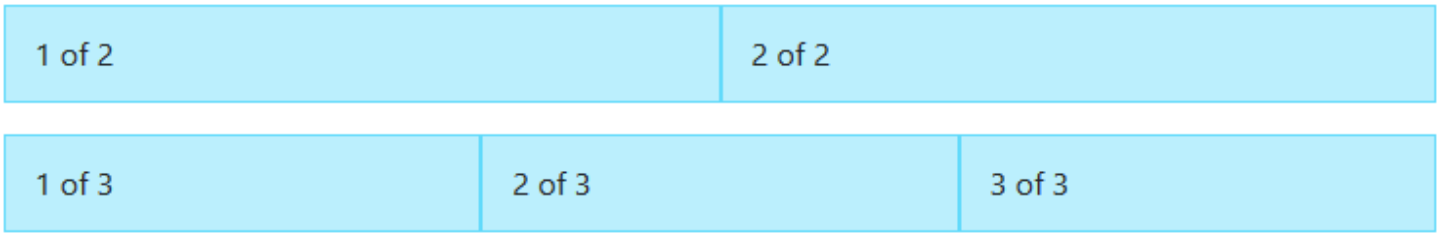
Containerele ofera posibilitatea de a centra si alinia orizontal continutul paginii. Acestea sunt responsive, max-width modificandu-se in functie de dimensiunea viewport-ului. Daca se specifica optiunea "fluid", ele vor avea width 100% intotdeauna.

1 of 1

```
<Container> //<Container fluid> --> width 100%
  <Row>
    <Col>1 of 1</Col>
  </Row>
</Container>
```

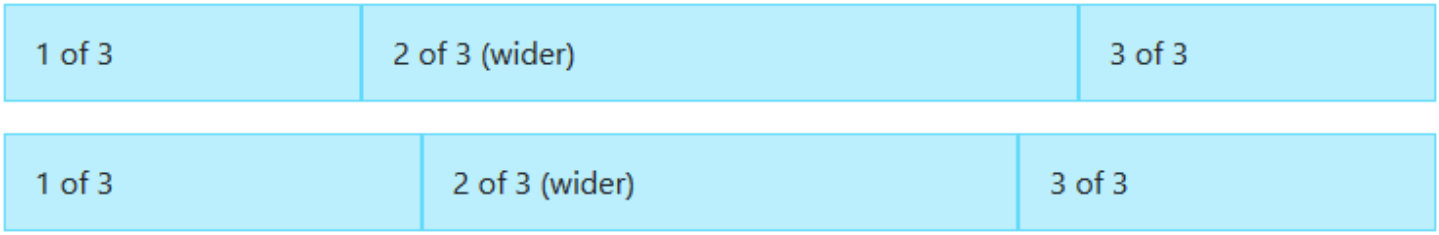
Auto-layout columns

Cand nu se specifica nicio dimensiune, componentele *Col* vor avea dimensiuni egale.




```
<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>
```

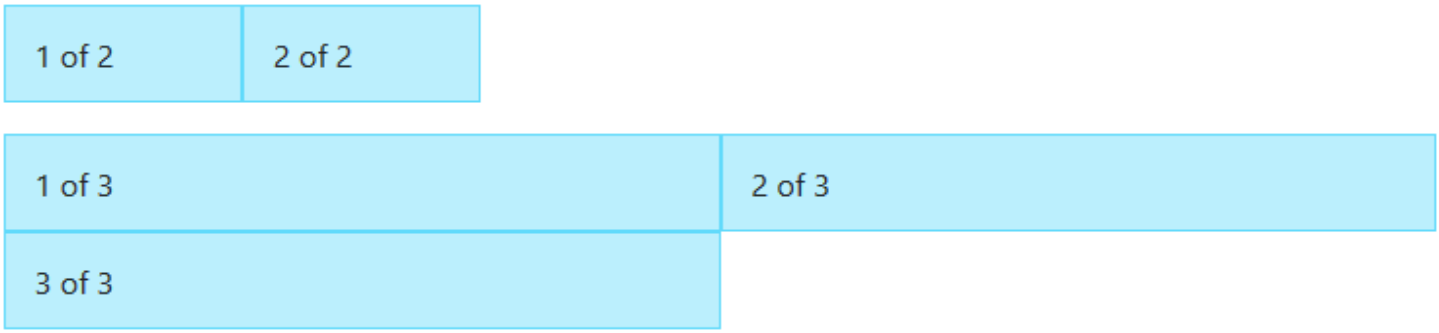
Daca pentru una dintre coloane se specifica o dimensiune, atunci toate celelalte coloane isi vor schimba automat dimensiunea, ramanand egale intre ele.




```
<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>
```

Rows

Componenta *Row* permite setarea dimensiunilor coloanelor din interiorul ei pentru toate cele 5 viewport-uri predefinite (xs, sm, md, lg, xl). Pentru fiecare viewport, se poate specifica numarul de coloane care incap una langa alta.




```
<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>
</Container>
```

Componente

React Bootstrap ofera o serie de componente utile in orice aplicatie web, ce usureaza procesul de dezvoltare. Pentru a utiliza o componenta din Bootstrap, aceasta trebuie mai intai importata. Codul de mai jos importa o componenta de alerta.

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

Dupa ce a fost importata, componenta poate fi utilizata in fisier. Mai jos aveti, complet, un exemplu simplu de alerta.

```
import React, { Component } from 'react';
import Alert from 'react-bootstrap/Alert';

function Example() {
  return (
    <Alert dismissible variant="danger">
      <Alert.Heading>You got an error!</Alert.Heading>
      <p>
        Change this and that and try again.
      </p>
    </Alert>
  )
}
```

Starea poate fi si ea transmisa componentelor React Bootstrap ca si *props*. Exemplu:

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

  return (
    <>
      <Alert show={show} variant="success">
        <Alert.Heading>Alert Header</Alert.Heading>
        <p>
          Alert Content.
        </p>
        <hr />
        <div className="d-flex justify-content-end">
          <Button onClick={() => setShow(false)} variant="outline-success">
            Close!
          </Button>
        </div>
```

```
      </Alert>

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

render(<AlertDismissible />);
```

Variant

Variant este o proprietate a componentelor React Bootstrap ce permite atasarea de stiluri predefinite.

Pentru culori proprietatea **variant** poate avea 8 valori: primary (albastru), secondary (gri), success (verde), danger (rosu), warning (galben), info (grayish cyan), light (light gray), dark (dark gray). In general, culoarea se traduce in proprietatile CSS background-color si color.

Componente de baza

Butoane

Butoanele [https://react-bootstrap.github.io/components/buttons/] sunt reprezentate prin tag-ul **Button**. Pe langa proprietatea **variant**, se mai pot specifica:

- block - latimea butonului va fi maxima
- disabled - pe buton nu se vor executa mouse events
- size (sm sau lg) - dimensiunea butonului (small sau large)
- type (button, reset, submit) - echivalent cu atributul type al butonului HTML

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

<>
  <Button variant="primary" size="lg" block>
    Block level button
  </Button>
</>
```

Imagini

Imaginile [https://react-bootstrap.github.io/components/images/] se definesc prin componenta **Image**. Cele mai folosite proprietati ale unei imagini sunt:

- fluid - imaginea va fi scalata la dimensiunea containerului parinte
- rounded - forma imaginii va fi un dreptunghi cu colturile rotunjite
- roundedCircle - forma imaginii va fi un cerc

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

<>
  <Image src="myFile/myImage.png" roundedCircle />
</>
```

Tables

Tabelul [https://react-bootstrap.github.io/components/table/] este reprezentat prin tag-ul **Table**.

Cele mai folosite proprietati ale unui tabel sunt:

- bordered - adauga border pentru toate celulele tabelului
- borderless - scoate border de pe toate celulele tabelului, includiv din header

- hover - activeaza starea hover pentru toate randurile tabelului
- striped - randurile tabelului vor fi colorate alternant
- variant (valoarea dark) - inverseaza culorile tabelului: text ce culoare deschisa si background inchis

```
import Table from 'react-bootstrap/Table'
<>
  <Table striped bordered hover>
    <thead>
        <tr>
          <th>#</th>
          <th>Username</th>
          <th>Email</th>
        </tr>
    </thead>
    <tbody>
        <tr>
          <td>1</td>
          <td>mark</td>
          <td>mark@gmail.com</td>
        </tr>
        <tr>
          <td>2</td>
          <td>jacob</td>
          <td>jacob@gmail.com</td>
        </tr>
    </tbody>
  </Table>
</>
```

Componente utile

In continuare sunt descrise componentele ce vor fi utilizate si in rezolvarea exercitiilor.

Cards

Componenta Card [https://react-bootstrap.github.io/components/cards/], reprezentata prin tag-ul **Card**, este un container impartit in 3 sectiuni:

- header - <Card.Header>
- body - <Card.Body>
- footer - <Card.Footer>

Folosirea Card.Header si Card.Footer nu este obligatorie.

Pentru introducerea de text se folosesc urmatoarele componente:

- <Card.Title> - titlul principal
- <Card.Subtitle> - subtitlu, cu un font mai mic decat titlul
- <Card.Text> - similar cu un paragraf din HTML
- <Card.Link> - similar cu tagul <a> din HTML, necesita setarea atributului **href**

Imaginile se introduc cu tagul <Card.Img> si pot fi amplasate la inceputul sau la finalul card-ului prin specificarea atributului **variant** cu valorile **top** sau **bottom**.

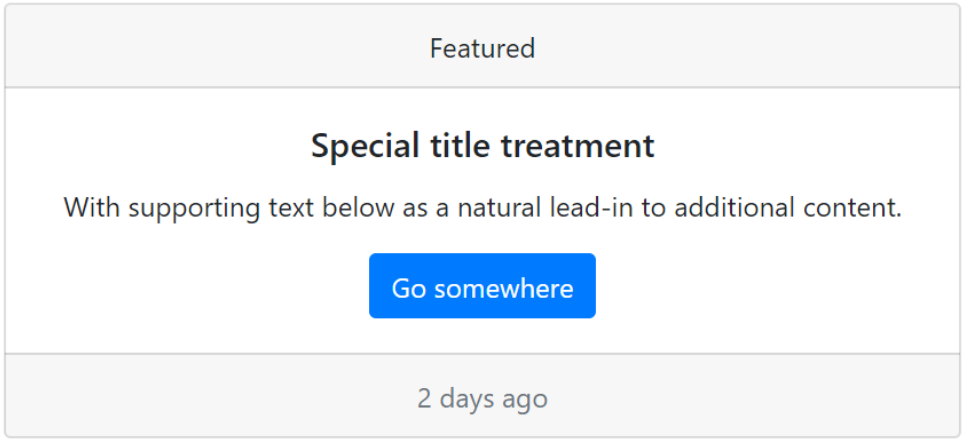
Un exemplu simplu de Card cu header si footer:

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

<>
  <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>
</>
```

Rezultatul codului de mai sus este:



Carousel

Carousel-ul [https://react-bootstrap.github.io/components/carousel/] este folosit pentru galeriile foto. Componenta **Carousel** contine mai multe tag-uri <Carousel.Item> in care se adauga imagini (<img>) si, optional, <Carousel.Caption>.

Pagination

Un exemplu de paginare este disponibil in documentatia oficiala [https://react-bootstrap.github.io/components/pagination/]. React Bootstrap ofera doar partea de prezentare a componentei, logica fiind facuta de developer. Astfel, la fiecare click pe un <Pagination.Item>, continutul vazut de utilizator trebuie sa se schimbe. Prin urmare, pe componenta <Pagination> atasam o functie evenimentului *click* ce va fi executata de fiecare data cand se schimba o pagina.

```
  constructor(props){
      super(props)
      this.pageChanged = this.pageChanged.bind(this);
   }

   pageChanged(e){
        // e.target.text contine numarul paginii
        console.log(e.target.text);
   }

    <Pagination onClick={this.pageChanged}>
       ...
    </Pagination>
```

JSON Placeholder

JSON Placeholder [https://jsonplaceholder.typicode.com/] este un Fake Online REST API ce ne permite testarea si prototiparea aplicatiilor. Adica, ne ofera o serie de endpoint-uri pe care putem sa le apelam din codul aplicatiei noastre. Resurse disponibile:

- /posts [https://jsonplaceholder.typicode.com/posts]
- /comments [https://jsonplaceholder.typicode.com/comments]
- /albums [https://jsonplaceholder.typicode.com/albums]
- /photos [https://jsonplaceholder.typicode.com/photos]

- /todos [https://jsonplaceholder.typicode.com/todos]
- /users [https://jsonplaceholder.typicode.com/users]

In exercitiile urmatoare vom folosi 3 resurse: albums, photos si users.

Pentru a vizualiza toate albumele, vom face un request de tip GET catre endpoint-ul /albums [https://jsonplaceholder.typicode.com/albums]

Pentru filtrarea resurselor folosim nested routes, explicate si in acest ghid [https://jsonplaceholder.typicode.com/guide.html]

Accesarea pozelor din albumul cu id-ul 1, se face prin trimiterea unui request de tip GET catre https://jsonplaceholder.typicode.com/albums/1/photos [https://jsonplaceholder.typicode.com/albums/1/photos]

Exercitii

In cadrul acestui laborator veti implementa o galerie foto pentru vizualizarea unei liste de albume si a pozelor din acestea.

1. [1p] Printr-un request de tip GET accesati toate albumele disponibile pe webserver-ul JSONPlaceholder.
2. [2p] Fiecare album va fi reprezentat intr-o componenta de tip **Card**.
    a. Card title este numele albumului
    b. Card text este informatia despre utilizatorul care a facut albumul, sub forma "by {username} aka {name}"
    c. [Bonus] Card image reprezinta prima poza din album (hint: afisati thumbnailUrl)
3. [3p] Afisati intr-o pagina toate albumele. Folositi componenta **Pagination** pentru a reprezenta cate 10 albume pe pagina.
4. [1p] La click pe un album trimiteti un request de tip GET pentru a accesa pozele albumului respectiv
5. [3p] Afisati pozele intr-o componenta **Carousel**.

pw/laboratoare/12.txt · Last modified: 2020/05/04 11:01 by ciprian.dobre