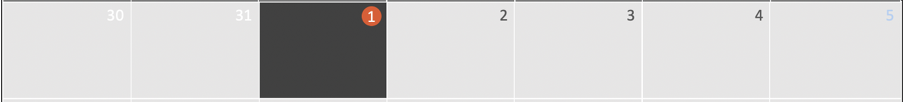
React 로 Calendar app만들기 - 5

# ­­클릭 한 날짜 표시하기



금일은 클릭하거나 최초 RCA 마운트 시 오늘에 해당하는 날에 Day컴포넌트에 색상을 다르게 해주는 작업을 할 것입니다. 그리고 캘린더에서 이번 달이 아닌 부분을 클릭하게 되면 해당 달력으로 이동되게 끔 할 예정입니다.

**/src/App.js**

import React, { Component } from 'react'

import Header from './Header'

import Calendar from './Calendar'

import moment from 'moment'

import './style/RCA.css'

export default class App extends Component {

state = {

calendarYM : moment(),

today : moment(),

selected : moment().format("YYYY-MM-DD")

}

static defaultProps = {

clickFn : ()=>{}

}

moveMonth = (month) => {

this.setState({

calendarYM : this.state.calendarYM.add(month,'M')

})

}

changeSelected = (clickedDate) =>{

if(moment(clickedDate).isSame(this.state.selected,'day')){

this.props.clickFn(clickedDate);

return;

}

this.setState({

selected : clickedDate

})

this.props.clickFn(clickedDate)

if(moment(clickedDate).isBefore(this.state.calendarYM,'month')){

this.moveMonth(-1)

}else if(moment(clickedDate).isAfter(this.state.calendarYM,'month')){

this.moveMonth(1)

}

}

render() {

return (

<div className="test-layout">

<div className="RCA-app-container">

<Header calendarYM={this.state.calendarYM.format("YYYY년 MM월")}

today={this.state.today.format("현재 YYYY - MM - DD")}

moveMonth={this.moveMonth}

/>

<Calendar YM={this.state.calendarYM.format("YYYY-MM-DD")}

selected={this.state.selected}

changeSelected={this.changeSelected}

/>

</div>

</div>

)

}

}

Selected에 관한 부분은 App에서 관리하게 됩니다. 그러므로 State에 selected를 추가하여 오늘의 날짜를 YYYY-MM-DD로 표시하게 끔 작성합니다.

state = {

calendarYM : moment(),

today : moment(),

selected : moment().format("YYYY-MM-DD")

}

다음으로 selected된 Day의 정보를 이용하여, App.state.selected를 변경하고, 만약 이번 달이 아닌 Day component를 클릭하였으면 해당 달의 정보로 App.state.calendarYM를 변경하는 함수를 작성해야 합니다 .

**/src/App.js**

changeSelected = (clickedDate) =>{

if(moment(clickedDate).isSame(this.state.selected,'day')){

this.props.clickFn(clickedDate);

return;

}

this.setState({

selected : clickedDate

})

this.props.clickFn(clickedDate)

if(moment(clickedDate).isBefore(this.state.calendarYM,'month')){

this.moveMonth(-1)

}else if(moment(clickedDate).isAfter(this.state.calendarYM,'month')){

this.moveMonth(1)

}

}

만약 Day Component들이 클릭 되게 된다면 위 함수의 파라미터로 DayComponent의 날짜 정보를 전달하게 됩니다. 이 경우 바로 setState 함수를 호출 하기 전에 생각해야 하는 것이 있는데, 만약 날짜가 같더라도 클릭을 하게 되면 App.js의 state변화로 props가 변화되는 계층 아래 component들이 다시 render함수를 호출 하게 됩니다. 쓸 때 없는 render는 자원의 낭비이니 만약 클릭한 day가 기존 selected된 날짜와 같은 경우 App.js로 props로 전달되는 함수를 실행만 시키고 함수를 종료하게 됩니다.

기존과 다른 day component의 날짜가 들어온 경우, 예정대로 App.state.selected를 변경 시키고, 기존 달과의 일치 여부에 따라 달을 변경 시키면 됩니다.

**/src/App.js**

render() {

return (

<div className="test-layout">

<div className="RCA-app-container">

<Header calendarYM={this.state.calendarYM.format("YYYY년 MM월")}

today={this.state.today.format("현재 YYYY - MM - DD")}

moveMonth={this.moveMonth}

/>

<Calendar YM={this.state.calendarYM.format("YYYY-MM-DD")}

selected={this.state.selected}

changeSelected={this.changeSelected}

/>

</div>

</div>

)

}

이제 selected와 changeSelected함수를 calendar component로 전달합니다.

**/src/calendar.js**

export default class Calendar extends Component {

Weeks = (monthYear,selected,clickFn) => {

const firstDayOfMonth = moment(monthYear).startOf('month');

const firstDateOfMonth = firstDayOfMonth.get('d');

const firstDayOfWeek = firstDayOfMonth.clone().add('d', -firstDateOfMonth);

const \_Weeks = [];

for (let i = 0; i < 6; i++) {

\_Weeks.push((

<Week key={`RCA-calendar-week-${i}`}

weekIndex={i}

ymOfThisCalendar={firstDayOfMonth.format("YYYY-MM")}

firstDayOfThisWeekformat={firstDayOfWeek.clone().add('d', i \* 7).format("YYYY-MM-DD")}

selected={selected}

fn={clickFn}

/>

))

}

return \_Weeks

}

render() {

return (

<div className="RCA-calendar-container">

<DateHeader dates={"Sun, Mon, Tue, Wed, Thu, Fri, Sat"} />

{this.Weeks(this.props.YM,this.props.selected,this.props.changeSelected)}

</div>

)

}

}

이제 전달받은 내용을 DayComponent로 전달하는 경로를 만들어주시면 됩니다.

**/src/calendar.js Week 내부**

class Week extends Component {

state = {}

Days = (firstDayFormat,weekIndex) => {

const \_days = [];

for (let i = 0; i < 7; i++) {

const Day = moment(firstDayFormat).add('d', i);

\_days.push({

yearMonthDayFormat: Day.format("YYYY-MM-DD"),

getDay: Day.format('D'),

isHolyDay: false,

weekIndex

});

}

return \_days;

}

mapDaysToComponents = (Days,calendarMonthYear ,selectedDayFormat ,fn = () => { }) => {

const thisMonth = moment(calendarMonthYear);

return Days.map((dayInfo, i) => {

let className = "date-weekday-label";

if (!thisMonth.isSame(dayInfo.yearMonthDayFormat,'month')) {

className = "date-notThisMonth";

} else if (i === 0) {

className = "date-sun"

}else if(i===6){

className ="date-sat"

}

if(moment(dayInfo.yearMonthDayFormat).isSame(selectedDayFormat,'day')){

className = "selected"

}

return (

<div className={"RCA-calendar-day " + className} key={`RCA-${dayInfo.weekIndex}-${i}-day`}onClick={() => fn(dayInfo.yearMonthDayFormat)}>

<label className="RCA-calendar-day-label">

{dayInfo.getDay}

</label>

</div>

)

})

}

render() {

return (

<div className="RCA-calendar-week">

{this.mapDaysToComponents(this.Days(this.props.firstDayOfThisWeekformat,this.props.weekIndex),

this.props.ymOfThisCalendar,

this.props.selected,

this.props.fn

)}

</div>

)

}

}

Day Component의 div에 selected 클래스를 추가 해주어야 하므로, DaysToComponent함수에 Calendar component로 부터 전달 받은 App.state.selected 내용과 App.changeSelected 함수를 파라메터로 전달합니다. Fn함수를 세번째 파라메터로 변경하고 2번째 파라메터로 selected내용을 전달합니다. 그리고 map함수 내부에서 dayInfo 객체를 통해 component를 만들 때 마다 해당 날짜가 전달받은 selected와 같은지를 마지막으로 체크하면 정상적으로 우리가 만들고자 했던 RCA가 완성됩니다.

**/src/RCA.css**

…

.selected{

background-color: rgb(49,50,50);

}

.selected label{

font-size: 1.2rem;

color: white;

width: 25px;

height: 25px;

background-color: rgb(224, 62, 42);

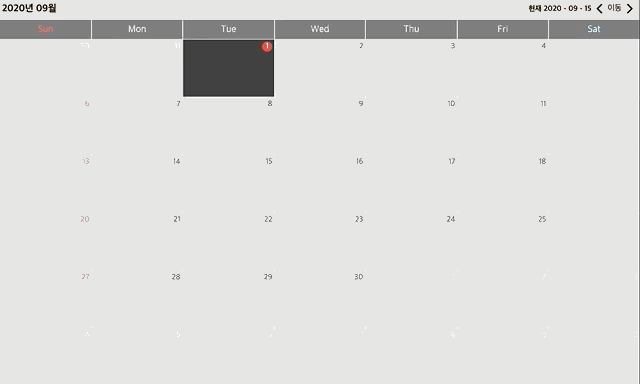
border-radius: 100%;

}

**결과**

사진, 앉아있는, 덮여있는, 대형이(가) 표시된 사진

자동 생성된 설명

****