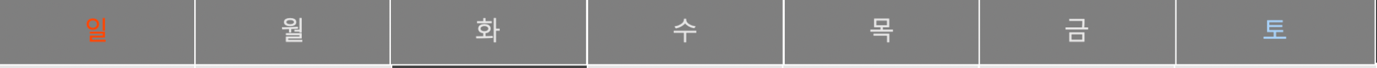
React로 calendar App만들기 -3

## Calendar container 요일 표시하기

이번 작업은 요일을 표시하는 Datehead component를 작업할 예정입니다.



현재 우리의 달력 app은 한글로 이루어져 있지만 사용자에 따라 영어로 sun, mon, tue, wed, thu, fri, sat,로 표현도 가능하고 한자로 日, 月, 火, 水, 木, 金, 土 로 표현 가능합니다. 그렇기에 개발자의 편의성을 위하여 Dateheader component로 요일에 대한 표현을 전달하면 요일을 표시하도록 구상했습니다. 이 과정에서 생각해볼 수 있는 것은 다형성의 원칙으로 인하여 “sun, mon, tue, wed, thu, fri, sat”도 요일을 모은 표시 형식이며, [‘sun’, ‘mon’, ‘tue’, ‘wed’, ‘thu’, ‘fri’, ‘sat’] 도 동일하게 요일을 표시할 수 있는 형식입니다. 그렇기에 두 타입을 모두 지원하도록 header를 구상했습니다.

**/src/Calendar**

class DateHeader extends Component {

dateToArray = (dates) => {

if(Array.isArray(dates)){

return dates

}else if(typeof dates === "string"){

return dates.split(',')

}else{

return ["일", "월", "화", "수", "목", "금", "토"]

}

}

mapArrayToDate = (dateArray) => {

try{

if(dateArray.length !== 7){

console.log(new Error("dates props must be had 7 date"))

dateArray = ["일", "월", "화", "수", "목", "금", "토"]

}

return dateArray.map((date, index) => {

const className = ()=>{

let className = "RCA-calendar-date-component";

if(index === 0){

return className + " date-sun"

}else if(index === 6){

return className + " date-sat"

}else{

return className + " date-weekday"

}

}

return (

<div className={className()} key={"RCA-header-"+date}>

{date}

</div>

)

})

}catch{

throw new Error ("date must be string or component")

}

}

render() {

return (

<div className="RCA-calendar-date-header">

{this.mapArrayToDate(this.dateToArray(this.props.dates))}

</div>

)

}

}

우리는 Array.map 메서드를 이용하여, date를 동일한 컴포넌트로 만들어 줄 것이기에 props로 전달받은 dates의 형식을 구분해야만 합니다. props로 전달하는 dates의 경우는 Date가 array로 전달되는 경우, Date가 string되는 경우, 전달되지 않는 경우 총 3가지 경우 입니다.

dateToArray = (dates) => {

if(Array.isArray(dates)){

return dates

}else if(typeof dates === "string"){

return dates.split(',')

}else{

return ["일", "월", "화", "수", "목", "금", "토"]

}

}

D/H component의 dateToArray 함수는 해당 props가 map을 쓰기 위하여 string이 들어오면 split메서드를 이용하여 스트링을 array로 변환하게 끔 합니다. 혹여 다른 type의 object가 들어오는 경우, 에러를 방지하기 위해서 else문으로 default DateArray를 전달합니다.

mapArrayToDate = (dateArray) => {

try{

if(dateArray.length !== 7){

console.log(new Error("dates props must be had 7 date"))

dateArray = ["일", "월", "화", "수", "목", "금", "토"]

}

return dateArray.map((date, index) => {

const className = ()=>{

let className = "RCA-calendar-date-component";

if(index === 0){

return className + " date-sun"

}else if(index === 6){

return className + " date-sat"

}else{

return className + " date-weekday"

}

}

return (

<div className={className()} key={"RCA-header-"+date}>

{date}

</div>

)

})

}catch{

throw new Error("date must be string or component")

}

}

이렇게 전달 받은 array가 우리가 예상치 못한 에러를 일으킬 수 있는 경우는 내부에 string혹은 React.component가 아닌 다른 type이 존재하는 경우 동작을 멈추게 됩니다. 그렇기에 try/catch문으로 mapArrayToDate의 내부를 감싸줍니다.

이제 우리가 생각하기에 발생할 수 있는 오류로는 dates props가 7일이 아닌 개발자의 단순 실수로 7개의 요일이 아닌 5일 혹은 6일의 요일을 담는 경우가 존재합니다. 그렇기에 dateArray를 map하기 전 array내부에 7개의 요일이 존재하는지 확인해야 합니다.

여기까지 순조로웠다면 이제 dateArray를 map해주면 됩니다. 우리의 컴포넌트 중 일요일, 토요일, 평일은 구분해주어야 할 필요가 있기 때문에, index에 따라 className이 결정됩니다.

(작성하고 보니 className함수에서 index를 map의 콜백에 있는 index 파라메터를 바라보고 있는데, 이를 className함수의 index로 전달 인자로 사용 했으면 코드가 깔끔하지 않았을까 생각해봅니다.)

**/src/RCA.css**

*…*

*/\** calendar component *\*/*

.RCA-calendar-container{

flex-grow: 1;

background-color: rgb(127, 184, 127)

}

*/\** calendar-header *\*/*

.RCA-calendar-date-header{

display: flex;

height: 50px;

background-color: green;

}

.RCA-calendar-date-component{

width: 14.2857143%;

text-align: center;

box-sizing: border-box;

background-color: rgb(107, 107, 108);

border: 0.1px solid white;

font-size : 1.5rem;

line-height: 50px;

}

.RCA-calendar-date-component:nth-child(odd){

border-left : none;

border-right: none;

}

.date-weekday{

color : rgba(255, 255, 255, 0.603)

}

.date-sun{

color: tomato;

}

.date-sat{

color: skyblue;

}

Css까지 완료 했다면 DateHeader를 마운트 시켜봅니다.

**/src/Calendar.js**

class DateHeader extends Component {

…

render() {

return (

<div className="RCA-calendar-date-header">

{this.mapArrayToDate(this.dateToArray(this.props.dates))}

</div>

)

}

}

export default class Calendar extends Component {

render() {

return (

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

<DateHeader />

</div>

)

}

}

**Props를 전달하지 않은 render**

스크린샷, 그리기이(가) 표시된 사진

자동 생성된 설명

**Custom array dates [‘sun’, ‘mon’, ‘tue’, ‘wed’, ‘thu’, ‘fri’, ‘sat’]**

render() {

return (

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

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

</div>

)

}

스크린샷이(가) 표시된 사진

자동 생성된 설명

**Custom string dates “Sun, Mon, Tue, Wed, Thu, Fri, Sat”**

render() {

return (

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

<DateHeader dates={"sun, mon, tue, wed, thu, fri, sat"}/>

</div>

)

}

스크린샷이(가) 표시된 사진

자동 생성된 설명

Error case : under 7 length

render() {

return (

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

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

</div>

)

}

스크린샷이(가) 표시된 사진

자동 생성된 설명