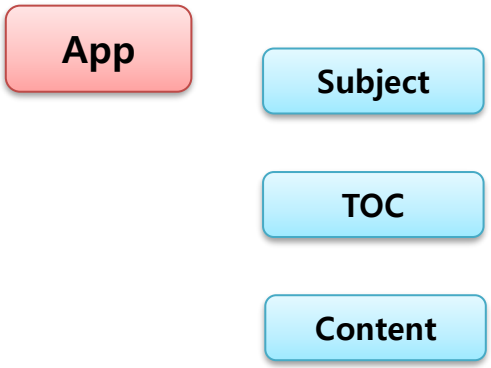
CRUD 구현

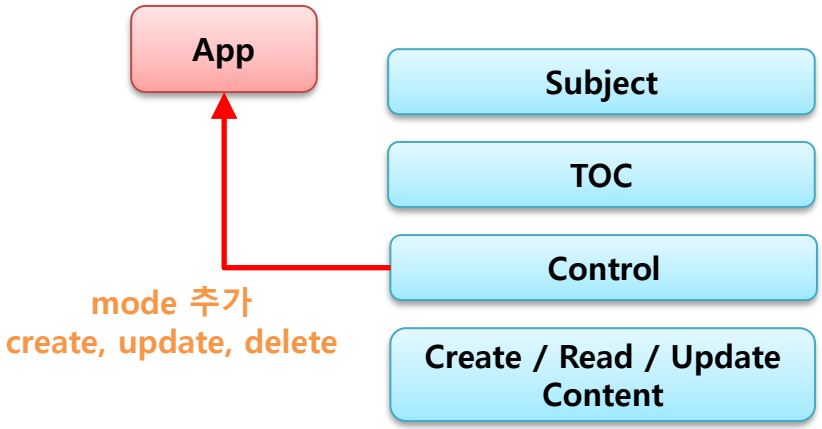
1. Create
2. Read
3. Update
4. Delete

■ CRUD

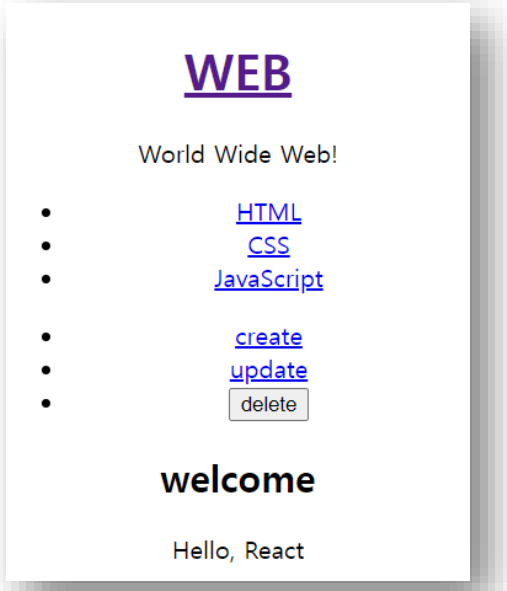
- 입력, 수정, 삭제 기능 추가
- 입력 (Create)
  - <form>, <input>, <textarea> 등을 이용하여 데이터 추가
- 수정 (Update)
  - selected_content_id 에 해당하는 데이터를 읽은 후 Create와 같은 요소에 데이터 출력, 데이터 수정 후 저장
- 삭제 (Delete)
  - selected_content_id 에 해당하는 데이터 삭제

■ CRUD


App
Subject
TOC
Content


App
Subject
TOC
Control
Create / Read / Update
Content
mode 추가
create, update, delete


WEB
World Wide Web!
HTML2
CSS2
JavaScript2
welcome
Hello, React


WEB
World Wide Web!
HTML
CSS
JavaScript
create
update
delete
welcome
Hello, React

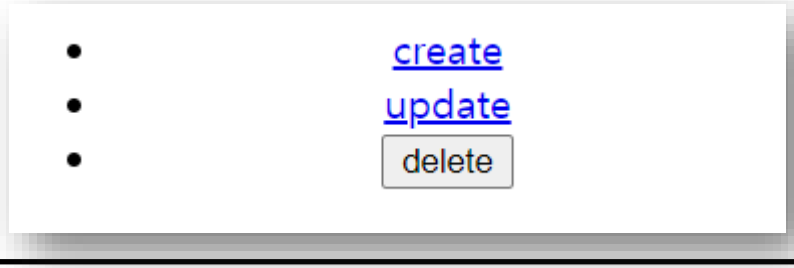
■ CRUD

● Control.js 추가

```
import React, {Component} from 'react';

class Control extends Component {
  render() {
    return (
      <ul>
        <li><a href='/create'>create</a></li>
        <li><a href='/update'>update</a></li>
        <li><button>delete</button></li>
      </ul>
    )
  }
}

export default Control;
```



■ CRUD

● App.js

```
import Control from './Control';

...

  render() {
    …
    return (
      <div className="App">
        <Subject … ></Subject>
        <TOC … ></TOC>

        <Control></Control>

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

■ CRUD

● Control에서 선택한 값에 따라 화면이 변경될 수 있도록 이벤트 작성

- App.js

```
render() {
  …
  return (
    <div className="App">
      …
      <Control onChangeMode={function(mode) {
        this.setState({mode: mode});
      }.bind(this)}></Control>
      …
    </div>
  );
}
```

■ CRUD

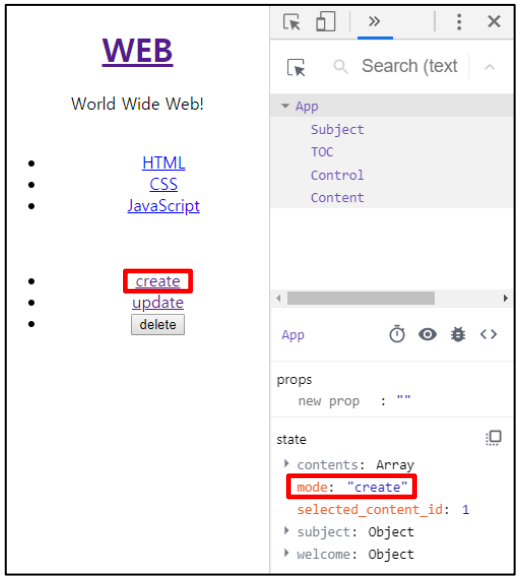
● Control에서 선택한 값에 따라 화면이 변경될 수 있도록 이벤트 작성

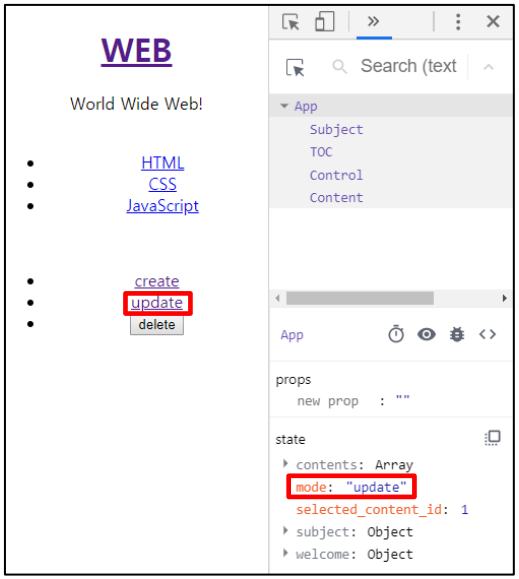
- Control.js

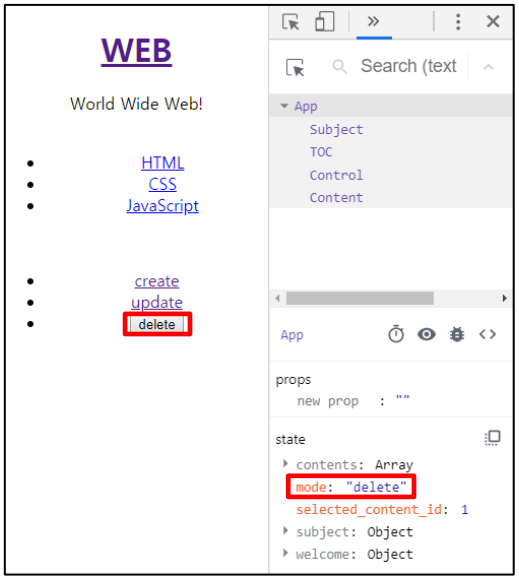
```
<ul>
  <li>
    <a href='/create' onClick={function(e) {
      e.preventDefault();
      this.props.onChangeMode('create');
    }.bind(this)}>create</a>
  </li>
  <li><a href='/update' onClick={function(e) {
      e.preventDefault();
      this.props.onChangeMode('update');
    }.bind(this)}>update</a></li>
  <li><button onClick={function(e) {
      e.preventDefault();
      this.props.onChangeMode('delete');
    }.bind(this)}>delete</button></li>
</ul>
```

■ CRUD

● Control에서 선택한 값에 따라 화면이 변경될 수 있도록 이벤트 작성







■ CRUD - Create

● Create 선택 시 입력 form 요소 출력

- CreateContent.js

```
import React, {Component} from 'react';

class CreateContent extends Component {
  render() {
    return (
      <article>
        <h2>Create Content</h2>
        <form action='/create_process' method='post'>
          <p><input type='text' name='title'></input></p>
          <p><textarea name='desc'></textarea></p>
          <p><input type='submit'></input></p>
        </form>
      </article>
    )
  }
}

export default CreateContent;
```

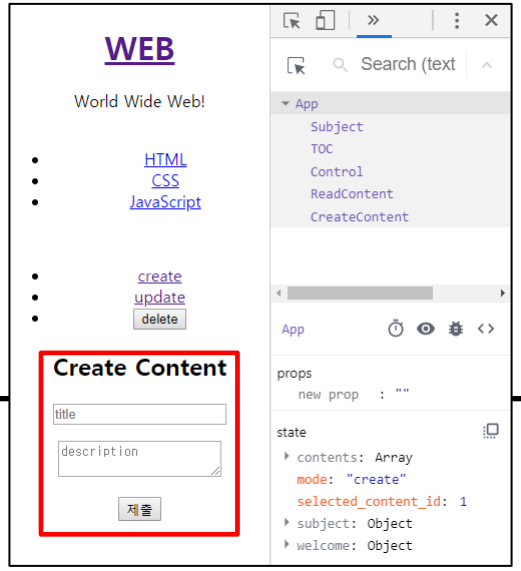
■ CRUD - Create

● Create 선택 시 입력 form 요소 출력

- App.js

```
import CreateContent from './CreateContent';
…
  render() {
    let article;
    …
    } else if(this.state.mode === 'create') {
      article = <CreateContent></CreateContent>
    }

    return (
      <div className="App">
        …
        {article}
      </div>
    );
```



■ CRUD - Create

● Submit 이벤트 처리

- input, textarea 요소에 입력된 값을 App.js로 전달

- CreateContent.js

```
<form action='/create_process' method='post'
  onSubmit={function(e) {
    e.preventDefault();
    const title = e.target.title.value;
    const desc = e.target.desc.value;
    this.props.onSubmit(title, desc);
  }.bind(this)}>
  <p><input type='text' name='title'></input></p>
  <p><textarea name='desc'></textarea></p>
  <p><input type='submit'></input></p>
</form>
```

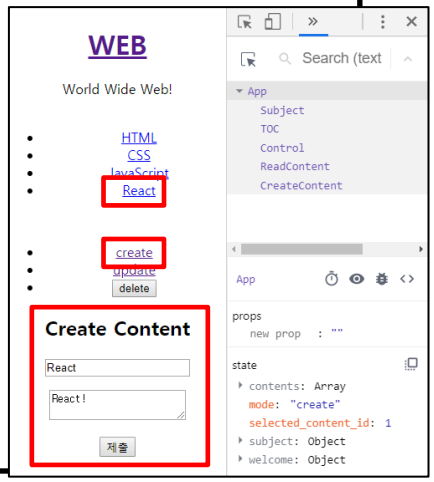
■ CRUD - Create

● Submit 이벤트 처리

- 입력 값을 기존 데이터 contents에 추가

- App.js

```
render() {
  let article;
  …
  } else if(this.state.mode === 'create') {
    article = <CreateContent
      onSubmit={function(title, desc) {
        console.log(title, desc);
        this.state.contents.push({
          id: this.state.contents.length + 1,
          title: title,
          desc: desc
        });
        this.setState({
          contents: this.state.contents
        })
      }.bind(this)}></CreateContent>
  }
```



■ CRUD - Read

● Content.js ➔ ReadContent.js 파일명 및 관련 코드 변경

- **ReadContent.js (파일명 수정)**

```
import React, {Component} from 'react';

class ReadContent extends Component {
  render() {
    return (
      <article>
        <h2>{this.props.title}</h2>
        {this.props.desc}
      </article>
    )
  }
}

export default ReadContent;
```

■ CRUD - Read

● Content.js ➔ ReadContent.js 파일명 및 관련 코드 변경

- App.js

```
import ReadContent from './ReadContent';

  …

  render() {
    …
    return (
      <div className="App">
        …
        <ReadContent title={title} desc={desc}></ReadContent>
        {article}
      </div>
    );
  }
```

■ CRUD - Update

● Update 선택 시 입력 form 요소에 해당 값 출력

- App.js

```
import UpdateContent from './UpdateContent';

  ...

    } else if(this.state.mode === 'update') {
      const content =
        this.findContentById(this.state.selected_content_id);
      title = content.title;
      desc = content.desc;
      article = <UpdateContent
        title={title} desc={desc}></UpdateContent>
    }
```

■ CRUD - Update

● Update 선택 시 입력 form 요소에 해당 값 출력

- UpdateContent.js

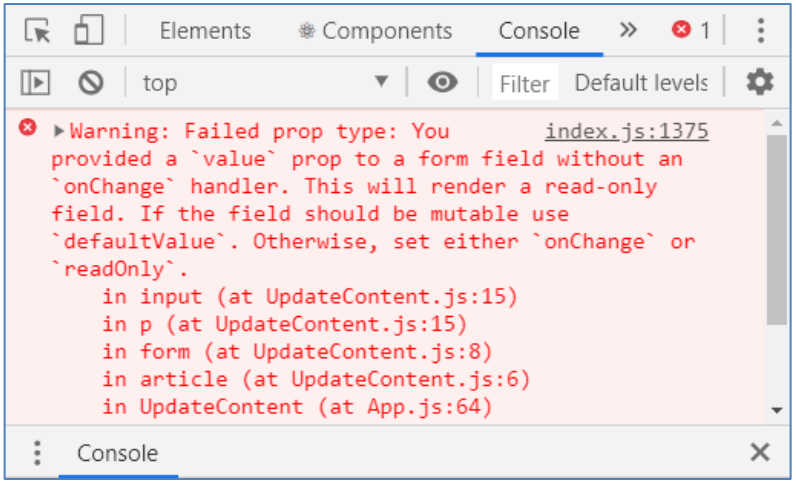
```
import React, {Component} from 'react';

class UpdateContent extends Component {
  render() {
    return (
      <article>
        <h2>Update Content</h2>
        <form action='/update_process' method='post'>
          <p><input type='text' name='title'
               value={this.props.title}></input></p>
          <p><textarea name='desc'
               value={this.props.desc}></textarea></p>
          <p><input type='submit'></input></p>
        </form>
      </article>
    )
  }
}

export default UpdateContent;
```

■ CRUD - Update

- props의 데이터는 수정 불가
- 수정이 가능한 HTML 요소의 value로 props 사용시 오류 메시지 출력



■ CRUD - Update

- React의 규칙으로 form 요소의 값을 변경 할 때는 onChange() 사용
- 상위 컴포넌트에서 넘겨준 props를 하위 컴포넌트 state로 저장
- 하위 컴포넌트에서 변경되는 내용을 setState()를 이용하여 처리

  - UpdateContent.js ➔ constructor()

```
constructor(props) {
  super(props);
  this.state = {
    title: this.props.title,
    desc: this.props.desc
  }
}
```

■ CRUD - Update

- React의 규칙으로 form 요소의 값을 변경 할 때는 onChange() 사용
- 상위 컴포넌트에서 넘겨준 props를 하위 컴포넌트 state로 저장
- 하위 컴포넌트에서 변경되는 내용을 setState()를 이용하여 처리

  - UpdateContent.js ➔ render() <form>

```
<form action='/update_process' method='post'>
  <p><input type='text' name='title'
       value={this.state.title}
       onChange={function(e) {
         this.setState({title: e.target.value});
       }.bind(this)}></input></p>
  <p><textarea name='desc'
       value={this.state.desc}
       onChange={function(e) {
         this.setState({desc: e.target.value});
       }.bind(this)}></textarea></p>
  <p><input type='submit'></input></p>
</form>
```

■ CRUD - Update

● Submit 이벤트 처리

- input, textarea 요소에 입력된(수정된) 값을 App.js로 전달

- UpdateContent.js

```
<form action='/update_process' method='post'
  onSubmit={function(e) {
    e.preventDefault();
    const title = e.target.title.value;
    const desc = e.target.desc.value;
    this.props.onSubmit(title, desc);
  }.bind(this)}>
```

■ CRUD - Update

● Submit 이벤트 처리

- input, textarea 요소에 입력된(수정된) 값을 App.js로 전달

- App.js

```
    } else if(this.state.mode === 'update') {
      const content =
        this.findContentById(this.state.selected_content_id);
      title = content.title;
      desc = content.desc;
      article = <UpdateContent title={title} desc={desc}
        onSubmit={function(title, desc) {
          content.title = title;
          content.desc = desc;
          this.setState({mode: 'read'});
        }.bind(this)}></UpdateContent>
    }
```

■ CRUD - Delete

● Delete 선택 시 selected_content_id에 지정되어 있는 content 삭제

- 선택된 요소에 따라 App.js의 onChangeMode() 호출

```
<Control onChangeMode={function (mode) {
  if (mode === 'delete') {
    const contents = this.state.contents;
    if (window.confirm('really?')) {
      for (let i = 0; i < contents.length; i++) {
        if (contents[i].id === this.state.selected_content_id) {
          contents.splice(i, 1);
        }
      }
    }
    this.setState({
      mode: 'welcome', contents: contents
    });
  } else {
    this.setState({ mode: mode });
  }
}.bind(this)}></Control>
```

■ 연습문제

● react-sample 프로젝트에 아래의 **DescInput 컴포넌트를 추가**하고 DescInput의 input 요소에서 **입력된 내용이 Footer 컴포넌트에서 보이도록 DescInput.js 와 App.js 수정**하기

1. props와 state 활용 (state 수정은 setState 함수로)
2. onClick / onChange 이벤트 사용

DescInput.js

```
import './DescInput.css';
import {Component} from 'react';

class DescInput extends Component {
  render() {
    return (
      <form className='desc-input'>
        <input type='text' />
        <button type='submit'>추가</button>
      </form>
    );
  }
}

export default DescInput;
```

DescInput.css

```
.desc-input { display: flex; }
.desc-input input {
  background: orange;
  border: none;
  padding: 0.5rem; font-size: 1.125rem;
  line-height: 1.5;
}
.desc-input button {
  outline: none; border: none;
  background: #868e96;
  color: white;
  padding-left: 1rem;
  padding-right: 1rem;
  font-size: 1.5rem;
  transition: 0.1s background ease-in;
}
.desc-input button:hover { background: #adb5bd; }
```

■ 연습문제

- react-sample 프로젝트에 아래의 **DescInput 컴포넌트를 추가**하고 DescInput의 input 요소에서 **입력된 내용이 Footer 컴포넌트에서 보이도록 DescInput.js 와 App.js 수정**하기

횟수	숫자	판정
1	8 3 0	아웃
2	6 5 9	0S 1B
3	2 6 4	1S 1B
4	1 2 6	1S 2B
5	2 1 6	3S 0B

1. 830 - 들어맞는 숫자가 아예 없으므로 아웃. 이때부터 0, 3, 8이 후보에서 빠지므로 남는 숫자는 1, 2, 4, 5, 6, 7, 9다.

2. 659 - 6이 있지만 위치가 다르므로 1볼. 게임 상으로는 어떤 숫자가 맞는지 모르기 때문에 가장 난감하다.

3. 264 - 2가 있고 위치가 맞으며, 6이 있지만 위치가 다르므로 1스트라이크 1볼.

4. 126 - 숫자는 전부 맞지만 위치는 6만 맞고 나머지 둘은 다르므로 1스트라이크 2볼.

5. 216 - 전부 맞으므로 승리.

6. 379 - 아웃 추가

횟수	숫자	판정
1	8 3 0	아웃
2	6 5 9	0S 1B
3	2 6 4	1S 1B
4	1 2 6	1S 2B
5	2 1 6	3S 0B

1. 830 - 들어맞는 숫자가 아예 없으므로 아웃. 이때부터 0, 3, 8이 후보에서 빠지므로 남는 숫자는 1, 2, 4, 5, 6, 7, 9다.

2. 659 - 6이 있지만 위치가 다르므로 1볼. 게임 상으로는 어떤 숫자가 맞는지 모르기 때문에 가장 난감하다.

3. 264 - 2가 있고 위치가 맞으며, 6이 있지만 위치가 다르므로 1스트라이크 1볼.

4. 126 - 숫자는 전부 맞지만 위치는 6만 맞고 나머지 둘은 다르므로 1스트라이크 2볼.

5. 216 - 전부 맞으므로 승리.

6. 379 - 아웃

추가