<select>

내장 브라우저 `<select>` 컴포넌트는 옵션을 포함하는 select box를 렌더링합니다.

```
<select>
  <option value="someOption">Some option</option>
  <option value="otherOption">Other option</option>
</select>
```

레퍼런스

`<select>`

select box를 표시하려면 내장 브라우저 `<select>` 컴포넌트를 렌더링합니다.

```
<select>
```

```
    <option value="someOption">Some option</option>
    <option value="otherOption">Other option</option>
  </select>
```

아래 예시를 참고하세요.

Props

`<select>` 는 모든 공통 엘리먼트 Props를 지원합니다.

select box를 제어하려면 `value` Prop을 전달하세요.

- `value` : 문자열 타입 (또는 `multiple={true}` 에서 사용하는 문자열 배열)이며 어떤 옵션을 선택할지 제어합니다. `value` 는 `<select>` 내부에 중첩된 `<option>` 의 `value` 와 일치합니다.

`value` 를 전달할 때, 전달된 `value` 를 업데이트하는 `onChange` 핸들러를 전달해야 합니다.

`<select>` 가 제어되지 않는다면, `defaultValue` Prop을 전달합니다.

- `defaultValue` : 문자열 타입 (또는 `multiple={true}` 에서 사용하는 문자열 배열)이며 초기 선택 옵션을 지정합니다.

`<select>` Props는 제어되지 않는 상태와 제어되는 상태 모두에 적용됩니다.

- `autoComplete` : 문자열 타입. 가능한 자동완성 동작 중 하나를 지정합니다.
- `autoFocus` : 불리언 타입. `true` 이면 React가 마운트 시 해당 요소에 포커싱합니다.
- `children` : `<select>` 는 자식으로 `<option>` , `<optgroup>` , `<datalist>` 컴포넌트를 받습니다. 또한 허용되는 컴포넌트 중 하나를 렌더링하기만 한다면 사용자 정의 컴포넌트를 전달할 수도 있습니다. 최종적으로 `<option>` 태그를 렌더링하는 사용자 정의 컴포넌트를 전달한다면, 렌더링되는 모든 `<option>` 은 `value` 를 가져야 합니다.
- `disabled` : 불리언 타입. `true` 일 경우 select box는 상호작용할 수 없게 되고 흐리게 표시됩니다.
- `form` : 문자열 타입. 이 select box가 속하는 `<form>` 의 `id` 를 지정합니다. 생략되면 가장 가까운 부모 폼으로 지정됩니다.
- `multiple` : 불리언 타입. `true` 이면 브라우저가 다중 선택을 허용합니다.
- `name` : 문자열 타입. 양식과 함께 제출되는 select box의 이름을 지정합니다.
- `onChange` : `Event` 핸들러 함수. 제어되는 select box에 필수적입니다. 사용자가 다른 옵션을 선택하는 즉시 실행됩니다. 브라우저의 `input` 이벤트처럼 동작합니다.
- `onChangeCapture` : 캡처 단계에서 실행되는 버전의 `onChange` .

- `onInput`: `Event` 핸들러 함수. 사용자에 의해 값이 변경되는 즉시 실행됩니다. 역사적인 이유로 React에서는 유사하게 동작하는 `onChange` 를 대신 사용하는 것이 관용적입니다.
- `onInputCapture`: 캡처 단계에서 실행되는 버전의 `inputCapture`.
- `onInvalid`: `Event` 핸들러 함수. 양식 제출 시 입력 유효성 검사에 실패하면 실행됩니다. 내장 `invalid` 이벤트와 다르게, React의 `onInvalid` 이벤트는 버블링됩니다.
- `onInvalidCapture`: 캡처 단계에서 실행되는 버전의 `invalidCapture`.
- `required`: 불리언 타입. `true` 일 경우 양식을 제출할 때 값이 반드시 제공되어야 합니다.
- `size`: 숫자 타입. `multiple={true}` 인 select에 대해, 초기에 표시되는 기본 항목의 수를 지정합니다.

주의 사항

- HTML과는 달리, `selected` 어트리뷰트를 `<option>` 에 전달하는 것은 지원하지 않습니다. 대신, 제어되지 않는 select box인 경우 `<select defaultValue>` 를 사용하고, 제어되어야 하는 select box인 경우 `<select value>` 를 사용해야 합니다.
- `<select>` 에 `value` Prop이 전달된다면, 제어되는 것으로 간주합니다.
- select box는 제어 상태와 비제어 상태를 동시에 행할 수 없습니다. 둘 중 하나의 상태만 가질 수 있습니다.
- select box는 생명주기 동안 처음 설정한 제어 상태를 변경할 수 없습니다.
- 제어되는 모든 select box는 제공되는 값을 동기적으로 업데이트하는 `onChange` 이벤트 핸들러가 필요합니다.

사용법

옵션이 담긴 select box 표시

`<select>` 는 `<option>` 컴포넌트의 리스트를 포함하는 `<select>` 를 렌더링합니다. 각 `<option>` 에는 폼과 함께 제출되는 데이터인 `value` 를 지정합니다.

App.js 다운로드 새로고침 Clear 포크

```
export default function FruitPicker() {
  return (
    <label>
      Pick a fruit:
      <select name="selectedFruit">
```

```
            <option value="apple">Apple</option>
            <option value="banana">Banana</option>
            <option value="orange">Orange</option>
          </select>
        </label>
      );
    `
```

select box가 포함된 레이블 제공

레이블이 해당 select box와 연결되어 있음을 브라우저에 알리기 위해 `<label>` 태그 안에 `<select>` 를 배치합니다. 사용자가 레이블을 클릭하면 브라우저는 자동으로 select box에 초점을 맞춥니다. 또한, 접근성을 위해 필수적입니다. 사용자가 select box에 초점을 맞추면 스크린 리더가 레이블 캡션을 알립니다.

`<select>` 를 `<label>` 안에 중첩 시킬 수 없다면, 같은 ID를 `<select id>` 와 `<label htmlFor>` 에 전달하여 연결해야 합니다. 한 컴포넌트에서 여러 인스턴스 간 충돌을 피하려면 `useId` 를 사용하여 ID를 생성하세요.

App.js 다운로드 새로고침 Clear 포크

```jsx
import { useId } from 'react';

export default function Form() {
  const vegetableSelectId = useId();
  return (
    <>
      <label>
        Pick a fruit:
        <select name="selectedFruit">
          <option value="apple">Apple</option>
          <option value="banana">Banana</option>
          <option value="orange">Orange</option>
```

자세히 보기

초기 선택 옵션 제공

기본적으로 브라우저는 목록에서 첫 번째 `<option>` 을 선택합니다. 다른 옵션을 기본값으로 선택하려면 `<select>` 요소 `<option>` 의 `value` 를 `defaultValue` 로 전달해야 합니다.

App.js 다운로드 새로고침 Clear 포크

```jsx
export default function FruitPicker() {
  return (
    <label>
      Pick a fruit:
      <select name="selectedFruit" defaultValue="orange">
        <option value="apple">Apple</option>
        <option value="banana">Banana</option>
        <option value="orange">Orange</option>
      </select>
    </label>
  );
}
```

주의하세요!

HTML과는 달리 개별 `<option>`에 `selected` 어트리뷰트를 전달하는 것은 지원되지 않습니다.

다중 선택 활성화

사용자가 여러 옵션을 선택할 수 있도록 `<select>` 에 `multiple={true}` 를 전달해야 합니다. 초기 선택 옵션을 선택하려면 `defaultValue` 를 배열로 지정해야 합니다.

App.js 다운로드 새로고침 Clear 포크

```
export default function FruitPicker() {
  return (
    <label>
      Pick some fruits:
      <select
        name="selectedFruit"
        defaultValue={['orange', 'banana']}
        multiple={true}
      >
        <option value="apple">Apple</option>
        <option value="banana">Banana</option>
        <option value="orange">Orange</option>
```

자세히 보기

폼을 제출할 때 선택 상자에서 제공하는 값 읽기

내부에 `<button type="submit">` 이 있는 select box 주변에 `<form>` 을 추가하면 `<form onSubmit>` 이벤트 핸들러를 호출해 값을 전달할 수 있습니다. 아무런 설정이 되어 있지 않다면 브라우저는 양식 데이터를 현재 URL로 보내고 페이지를 새로 고칩니다. `e.preventDefault()` 를 호출하여 해당 동작을 재정의할 수 있습니다. `new FormData(e.target)` 로 양식 데이터 읽는 방법은 다음과 같습니다.

App.js

다운로드 새로고침 Clear 포크

```
export default function EditPost() {
  function handleSubmit(e) {
    // 브라우저가 페이지를 다시 로드하지 않도록 합니다.
    e.preventDefault();
    // 폼 데이터를 읽습니다.
    const form = e.target;
    const formData = new FormData(form);
    // formData를 fetch의 본문으로 직접 전달할 수 있습니다.
    fetch('/some-api', { method: form.method, body: formData });
    // 브라우저의 기본 동작처럼 URL을 생성할 수 있습니다.
    console.log(new URLSearchParams(formData).toString());
    // 일반 오브젝트로 작업할 수 있습니다.
```

자세히 보기

중요합니다!

`<select>` 에 `name` 을 지정해야 합니다. (예: `<select name="selectedFruit" />`) 지정한 `name` 은 폼 데이터에서 키로 사용됩니다. (예: `{ selectedFruit: "orange" }`)

`<select multiple={true}>` 를 사용하면 `FormData` 에서 각 선택한 값을 별도의 이름-값 쌍으로 포함합니다. 위의 예시에서 콘솔 로그를 확인해 주세요.

주의하세요!

기본적으로 `<form>` 내부의 *모든* `<button>` 은 select box의 값을 제출합니다. 의도치 않은 동작으로 인해 당황할 수 있습니다! 사용자 정의 `Button` React 컴포넌트가 있다면 `<button>` 대신 `<button type="button">` 을 반환하는 것을 고려해야 합니다. 그런 다음 명시적으로 폼을 제출해야 하는 곳에 `<button type="submit">` 을 사용해 주세요.

State 변수와 함께 select box 제어

`<select>` 와 같은 select box는 *제어되지 않습니다.* `<select defaultValue="orange" />` 와 같이 처음에 선택한 값을 전달하더라도 JSX는 현재 값이 아닌 초기 값만 지정합니다.

제어된 select box를 렌더링하려면 `value` Prop을 전달해야 합니다. React는 select box가 항상 전달한 `value` 를 갖도록 강제합니다. 보통 State 변수로 선언하여 선택 상자를 제어합니다.

```
function FruitPicker() {
  const [selectedFruit, setSelectedFruit] = useState('orange'); // State 변수를 선언합니
  // ...
  return (
    <select
      value={selectedFruit} // ...select의 값이 State 변수와 일치하도록 강제합니다....
      onChange={e => setSelectedFruit(e.target.value)} // ... 변경 사항이 있을 때마다 Sta
```

```
    >
      <option value="apple">Apple</option>
      <option value="banana">Banana</option>
      <option value="orange">Orange</option>
    </select>
  );
}
```

모든 선택에 대한 응답으로 UI 일부를 다시 렌더링하려는 경우 유용합니다.

App.js

다운로드 새로고침 Clear 포크

```
import { useState } from 'react';

export default function FruitPicker() {
  const [selectedFruit, setSelectedFruit] = useState('orange');
  const [selectedVegs, setSelectedVegs] = useState(['corn', 'tomato']);
  return (
    <>
      <label>
        Pick a fruit:
        <select
          value={selectedFruit}
          onChange={e => setSelectedFruit(e.target.value)}
```

자세히 보기

주의하세요!

`onChange` 없이 `value` 를 전달하면 옵션을 선택할 수 없습니다. `value` 를 전달하여 select box를 제어하면 전달한 값이 항상 있도록 *강제*합니다. 따라서 `value` 를 State 변수로 전달했지만 `onChange` 이벤트 핸들러에서 State 변수를 동기적으로 업데이트하지 않으면 React는 키를 누를 때마다 select box를 지정한 `value` 로 되돌립니다.

HTML과는 달리 개별 `<option>` 에 `selected` 어트리뷰트를 전달하는 것은 지원하지 않습니다.

이전
<progress>

다음
<textarea>