Conditional Rendering

Komponen Anda sering kali perlu menampilkan hal yang berbeda tergantung pada kondisi tertentu. Di React, Anda dapat melakukan perenderan secara kondisional menggunakan sintaksis JavaScript seperti pernyataan `if`, operator &&, dan operator ? :

Anda akan belajar

- Cara mengembalikan JSX yang berbeda tergantung pada suatu kondisi
- Cara memasukkan atau mengecualikan bagian JSX secara bersyarat
- Pintasan sintaksis kondisional umum yang akan Anda temui dalam basis kode React

Melakukan perenderan secara bersyarat

Buatlah file baru bernama `PackingList.js` yang dibuat di dalam folder `components`.

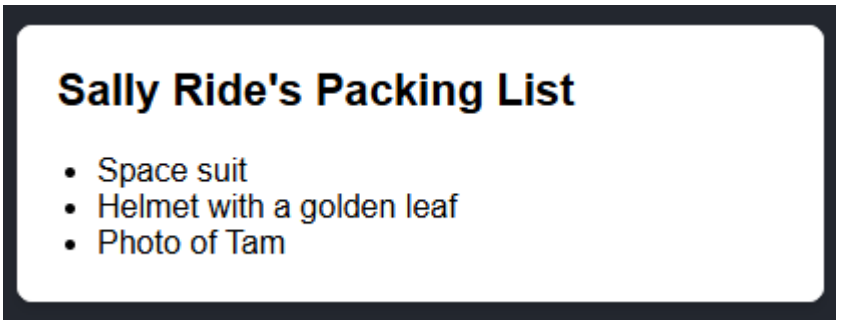
```
function Item({ name, isPacked }) {
  return <li className="item">{name}</li>;
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
```

```
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Lalu di dalam folder app pada bagian file page.js buat menjadi seperti berikut:

```
import PackingList from "../components/PackingList";
export default function Home() {
  return (
    <>
      <PackingList />
    </>
  );
}
```

Hasil:



Perhatikan bahwa beberapa `Item`komponen memiliki `isPacked` properti yang ditetapkan ke `true` bukan `false`. Anda ingin menambahkan tanda centang (✅) ke item yang dikemas jika `isPacked={true}`.

Anda dapat menuliskannya sebagai pernyataan if/else seperti ini:

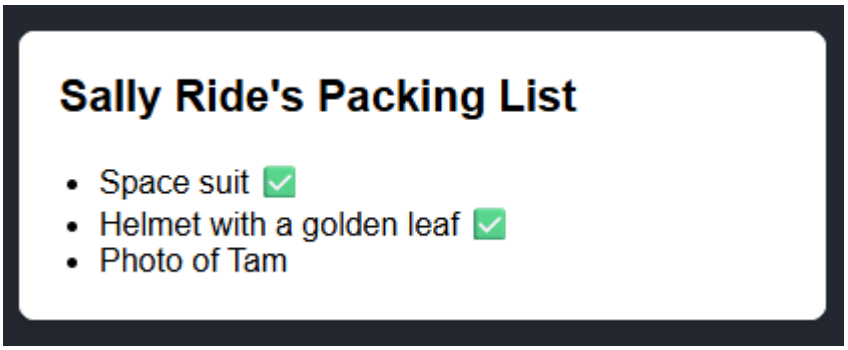
```
if (isPacked) {
  return <li className="item">{name} ✅</li>;
}
return <li className="item">{name}</li>;
```

Jika isPacked prop adalah true, kode ini mengembalikan pohon JSX yang berbeda. Dengan perubahan ini, beberapa item akan mendapatkan tanda centang di bagian akhir: Aplikasi.js

```
function Item({ name, isPacked }) {
  if (isPacked) {
    return <li className="item">{name} ✅</li>;
  }
  return <li className="item">{name}</li>;
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
```

```
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Hasil:



Cobalah mengedit apa yang dikembalikan dalam kedua kasus, dan lihat bagaimana hasilnya berubah!

Perhatikan bagaimana Anda membuat logika percabangan dengan JavaScript if dan return pernyataan. Dalam React, aliran kontrol (seperti kondisi) ditangani oleh JavaScript.

Secara kondisional tidak mengembalikan apa pun dengan null.

Dalam beberapa situasi, Anda tidak ingin merender apa pun sama sekali. Misalnya, katakanlah Anda tidak ingin menampilkan item yang dikemas sama sekali. Suatu komponen harus menampilkan sesuatu. Dalam kasus ini, Anda dapat menampilkan nya dengan null:

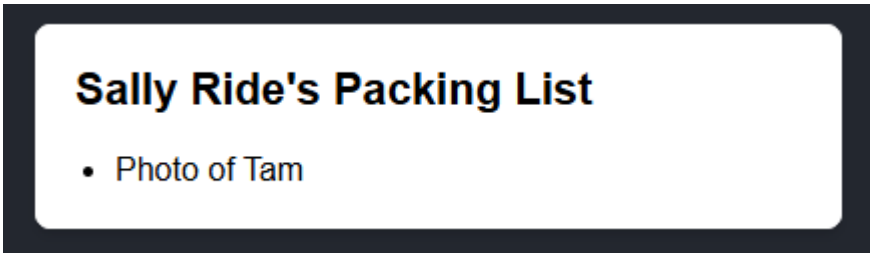
```
if (isPacked) {
  return null;
}
return <li className="item">{name}</li>;
```

Jika isPacked benar, komponen tidak akan mengembalikan apa pun null. Jika tidak, komponen akan mengembalikan JSX untuk dirender.

```
function Item({ name, isPacked }) {
  if (isPacked) {
    return null;
  }
  return <li className="item">{name}</li>;
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
```

```
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Hasil:



Dalam praktiknya, mengembalikan `null` dari suatu komponen tidaklah umum karena dapat membingungkan pengembang lain yang mencoba menggunakannya. Lebih sering, komponen akan ditampilkan atau tidak ditampilkan berdasarkan kondisi tertentu dari dalam komponen induknya. Berikut cara melakukannya!

Menampilkan komponen secara bersyarat

Pada contoh sebelumnya, Anda mengendalikan bagian tampilan mana (jika ada!) yang akan dikembalikan oleh komponen. Anda mungkin telah melihat beberapa duplikasi dalam hasil yang ditampilkan.

```
<li className="item">{name} ✅</li>
```

Sangat mirip dengan kode berikut:

```
<li className="item">{name}</li>
```

Kedua cabang kondisional mengembalikan <li className="item">...:

```
if (isPacked) {
  return <li className="item">{name} ✅</li>;
}
return <li className="item">{name}</li>;
```

Meskipun duplikasi ini tidak berbahaya, hal tersebut dapat membuat kode Anda lebih sulit untuk dikelola. Bagaimana jika Anda ingin mengubah `className`? Anda harus melakukannya di dua tempat dalam kode Anda! Dalam situasi seperti ini, Anda dapat menampilkan elemen atau komponen tertentu secara kondisional untuk membuat kode Anda lebih ringkas dan tidak berulang DRY.

Operator kondisional (terner) (? :)

JavaScript memiliki sintaksis yang ringkas untuk menulis ekspresi kondisional — operator kondisional atau “operator terner”.

Sebagai gantinya:

```
if (isPacked) {
  return <li className="item">{name} ✅</li>;
}
return <li className="item">{name}</li>;
```

Anda dapat menulis ini:

```
return (
  <li className="item">
    {isPacked ? name + ' ✅' : name}
  </li>
);
```

Anda dapat membacanya sebagai “jika *isPacked* benar, maka (?) *render name + ' ✅'*, jika tidak (:) render *name*”.

Apakah kedua contoh ini sepenuhnya setara?

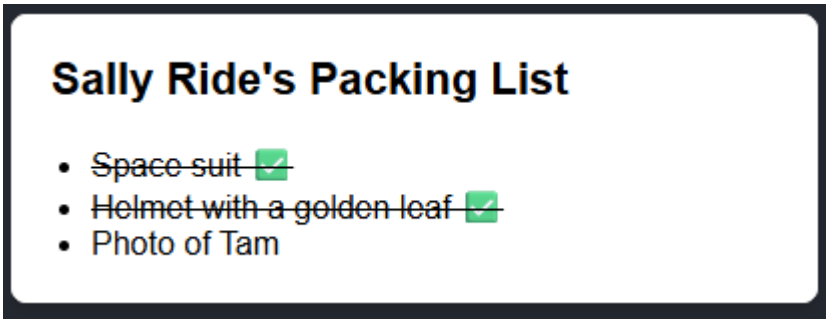
Jika Anda berasal dari latar belakang pemrograman berorientasi objek (OOP), Anda mungkin berasumsi bahwa kedua contoh di atas sedikit berbeda karena salah satunya dapat membuat dua "contoh" yang berbeda dari <li>. Namun, elemen JSX bukanlah "contoh" karena tidak memiliki status internal apa pun dan bukan simpul DOM yang sebenarnya. Elemen-elemen tersebut adalah deskripsi yang ringan, seperti cetak biru. Jadi, kedua contoh ini, pada kenyataannya, sepenuhnya setara. Preserving and Resetting State menjelaskan secara terperinci tentang cara kerjanya.

Sekarang, katakanlah Anda ingin membungkus teks item yang sudah selesai ke dalam tag HTML lain, seperti <del>mencoretnya. Anda dapat menambahkan lebih banyak baris baru dan tanda kurung sehingga lebih mudah untuk menyarangkan lebih banyak di setiap kasus:

```
function Item({ name, isPacked }) {
  return (
    <li className="item">
      {isPacked ? (
        <del>
          {name + ' ✅'}
        </del>
      ) : (
        name
      )}
    </li>
  );
}

export default function PackingList() {
  return (
    <section>
```

```
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Hasil



Gaya ini berfungsi baik untuk kondisi sederhana, tetapi gunakan secukupnya. Jika komponen Anda menjadi berantakan karena terlalu banyak markup kondisional bersarang, pertimbangkan untuk mengekstrak komponen anak untuk merapikannya. Di React, markup adalah bagian dari kode Anda, jadi Anda dapat menggunakan alat seperti variabel dan fungsi untuk merapikan ekspresi yang rumit.

Operator logika AND (&&)

Pintasan umum lainnya yang sering digunakan dalam JavaScript adalah operator logika AND (&&). Dalam komponen React, ini biasanya digunakan untuk menampilkan elemen tertentu hanya jika suatu kondisi bernilai benar, dan tidak menampilkan apa pun jika kondisinya salah. Misalnya, Anda bisa menampilkan tanda centang hanya jika nilai `isPacked` adalah `true`.

```
return (
  <li className="item">
    {name} {isPacked && '✅'}
  </li>
);
```

Anda dapat membacanya sebagai “jika isPacked, maka (&&) berikan tanda centang, jika tidak, tidak berikan apa pun”.
Berikut ini aksinya:

```
function Item({ name, isPacked }) {
  return (
    <li className="item">
      {name} {isPacked && '✅'}
    </li>
  );
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
```

```
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Ekspresi JavaScript `&&` akan menghasilkan nilai di sebelah kanan (dalam hal ini, tanda centang) jika nilai di sebelah kirinya (kondisi yang dicek) adalah `true`. Namun jika kondisinya `false`, maka seluruh ekspresi juga bernilai `false`. Dalam React, nilai `false` dianggap seperti nilai kosong, sehingga tidak akan ditampilkan apa pun di layar.

Pitfall

Jangan menaruh angka di sisi kiri &&.
Untuk menguji kondisi tersebut, JavaScript secara otomatis mengonversi sisi kiri menjadi boolean. Namun, jika sisi kiri adalah 0, maka seluruh ekspresi akan mendapatkan nilai tersebut (0), dan React akan dengan senang hati melakukan render 0daripada tidak melakukan apa pun. Misalnya, kesalahan umum adalah menulis kode seperti messageCount && New messages . Mudah untuk berasumsi bahwa kode tidak akan menghasilkan apa pun saat messageCount, 0tetapi kode tersebut benar-benar menghasilkan 0dirinya sendiri!. Untuk memperbaikinya, buat sisi kiri menjadi boolean: messageCount > 0 && New messages .

Menetapkan konten tampilan secara bersyarat ke dalam sebuah variabel.

Bila pintasan menghalangi penulisan kode biasa, cobalah menggunakan if pernyataan dan variabel. Anda dapat menetapkan ulang variabel yang didefinisikan dengan let, jadi mulailah dengan menyediakan konten default yang ingin Anda tampilkan, yaitu nama:

```
let itemContent = name;
```

Gunakan pernyataan `if` untuk menetapkan ulang nilai `itemContent` jika `isPacked` bernilai `true`.

```
if (isPacked) {
  itemContent = name + " ✅";
}
```

Tanda kurung kurawal membuka 'jendela ke dalam JavaScript'. Anda bisa menyisipkan variabel di dalam struktur tampilan yang dikembalikan dengan menggunakan kurung kurawal, sehingga ekspresi yang sudah dihitung sebelumnya bisa langsung ditampilkan.

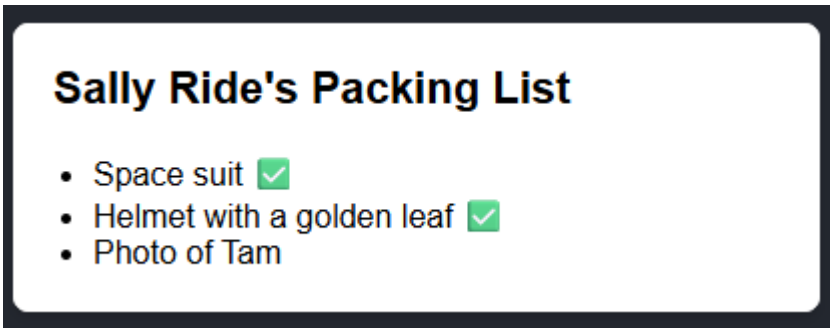
```
<li className="item">
  {itemContent}
</li>
```

Gaya ini adalah yang paling bertele-tele, tetapi juga paling fleksibel. Berikut ini cara kerjanya:

```
function Item({ name, isPacked }) {
  let itemContent = name;
  if (isPacked) {
```

```
    itemContent = name + " ✅";
  }
  return (
    <li className="item">
      {itemContent}
    </li>
  );
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Hasil

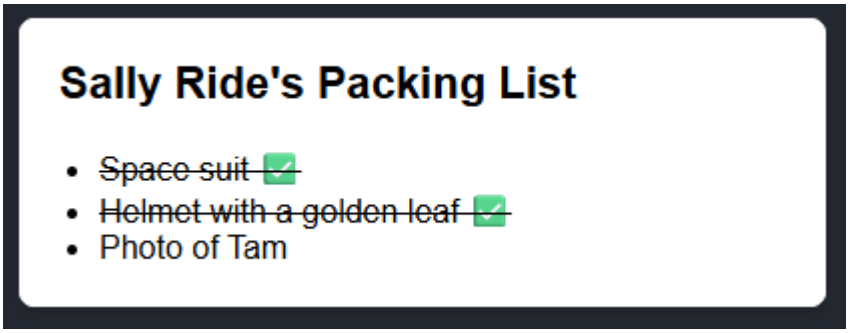


Seperti sebelumnya, cara ini tidak hanya berlaku untuk teks, tetapi juga untuk berbagai elemen tampilan lainnya.

```
function Item({ name, isPacked }) {
  let itemContent = name;
  if (isPacked) {
    itemContent = (
      <del>
        {name + " ✅"}
      </del>
    );
  }
  return (
    <li className="item">
      {itemContent}
    </li>
  );
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
```

```
          name="Space suit"
        />
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Hasil



Jika Anda tidak familier dengan JavaScript, berbagai gaya ini mungkin tampak membingungkan pada awalnya. Namun, mempelajarinya akan membantu Anda membaca dan menulis kode JavaScript apa pun — dan bukan hanya komponen React! Pilih yang Anda sukai sebagai permulaan, lalu lihat referensi ini lagi jika Anda lupa cara kerja yang lain.

Rekap

Di React, Anda mengendalikan logika percabangan dengan JavaScript. Anda bisa menampilkan konten secara kondisional menggunakan pernyataan `if`. Anda juga dapat menyimpan elemen yang ingin ditampilkan ke dalam variabel terlebih dahulu,

lalu menampilkannya sesuai kebutuhan dengan menyisipkannya menggunakan tanda kurung kurawal. Misalnya, ekspresi `{cond ? A : B}` berarti 'jika `cond` bernilai benar, tampilkan `A`; jika tidak, tampilkan `B`'. Sedangkan `{cond && A}` berarti 'jika `cond` benar, tampilkan `A`; jika tidak, tidak menampilkan apa pun'. Pintasan seperti ini umum digunakan, namun Anda tetap bisa memilih menulis logika dengan cara biasa menggunakan `if` jika dirasa lebih jelas.

Cobalah beberapa tantangan berikut :

Buatlah tiga file JavaScript dengan nama `Tantangan1.js`, `Tantangan2.js`, dan `Tantangan3.js` di dalam folder `components`. Setelah itu, impor dan panggil ketiga komponen tersebut di file `page.js` yang berada dalam folder `app`.

Tantangan 1

Menampilkan ikon untuk item yang tidak lengkap dengan ? :
Gunakan operator kondisional (cond ? a : b) untuk merender ❌ jika isPacked bukan true.

```
function Item({ name, isPacked }) {
  return (
    <li className="item">
      {name} {isPacked && '✅'}
    </li>
  );
}

export default function Tantangan1() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
```

```
        <Item
          isPacked={true}
          name="Helmet with a golden leaf"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Tantangan 2

Tunjukkan pentingnya item dengan &&

Dalam contoh ini, masing-masing Itemmenerima prop numerik importance. Gunakan &&operator untuk membuat “ (Importance: X) ” dalam huruf miring, tetapi hanya untuk item yang memiliki kepentingan bukan nol. Daftar item Anda akan terlihat seperti ini:

- Pakaian antariksa (Pentingnya: 9)
- Helm dengan daun emas
- Foto Tam (Pentingnya: 6)

Jangan lupa menambahkan spasi di antara kedua label tersebut!

```
function Item({ name, importance }) {

  return (

    <li className="item">

      {name}

    </li>

  );
```

```
}


export default function Tantangan2() {
  return (
    <section role="banner">
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          importance={9}
          name="Space suit"
        />
        <Item
          importance={0}
          name="Helmet with a golden leaf"
        />
        <Item
          importance={6}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}
```

Tantangan 3

Refraktor serangkaian variabel ? : ke if dan

Komponen ini Drink menggunakan serangkaian ? : kondisi untuk menampilkan informasi yang berbeda, tergantung pada apakah nameprop tersebut "tea" atau "coffee". Masalahnya adalah informasi tentang setiap minuman tersebar di beberapa kondisi. Ubah kode ini untuk menggunakan satu if pernyataan, bukan tiga ? : kondisi.

```
function Drink({ name }) {
  return (
    <section>
      <h1>{name}</h1>
      <dl>
        <dt>Part of plant</dt>
        <dd>{name === 'tea' ? 'leaf' : 'bean'}</dd>
        <dt>Caffeine content</dt>
        <dd>{name === 'tea' ? '15-70 mg/cup' : '80-185
mg/cup'}</dd>
        <dt>Age</dt>
        <dd>{name === 'tea' ? '4,000+ years' : '1,000+
years'}</dd>
      </dl>
    </section>
  );
```

```
}


export default function Tantangan3() {
  return (
    <div>
      <Drink name="tea" />
      <Drink name="coffee" />
    </div>
  );
}
```