


React pre začiatočníkov V. Lifting state up!

Ing. Slavomír Kožár

10. 4. 2024



Obsah

- ToDo - zmena stavu tasku
- ToDo - filter
- context

ToDo Filter

```
<ul className="nav nav-pills mb-4">
  <li className="nav-item">
    <button className="nav-link active">all</button>
  </li>
  <li className="nav-item">
    <button className="nav-link">active</button>
  </li>
  <li className="nav-item">
    <button className="nav-link">completed</button>
  </li>
</ul>
```

ToDo Filter

Klikaním na tlačidlá chceme prepínať filter medzi jednou z týchto hodnôt:

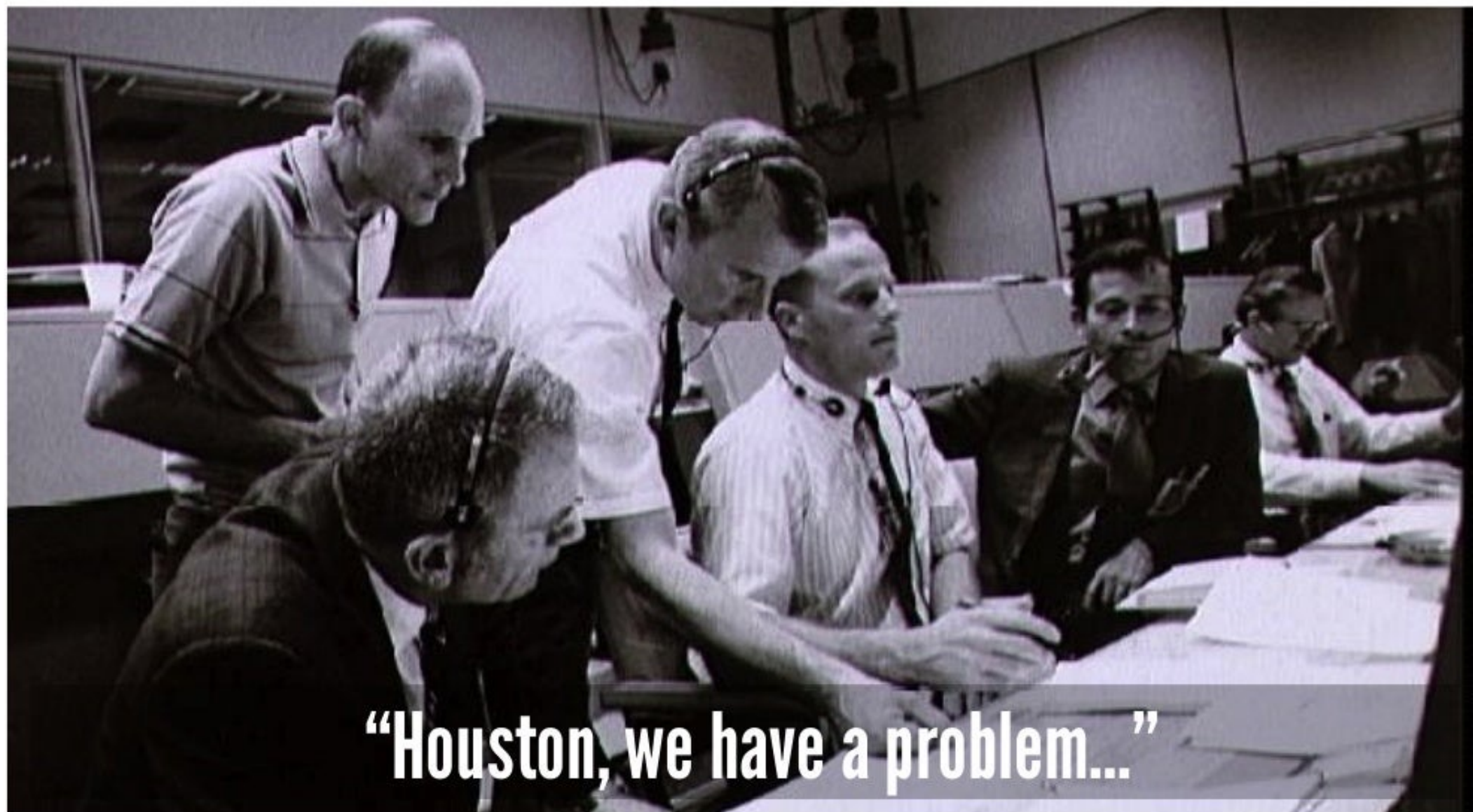
- `all`
- `active`
- `completed`

Aktuálne vybraný filter sa indikuje pridaním triedy `active`:

```
<li className="nav-item">  
  <button className="nav-link active">all</button>  
</li>
```

ToDo Filter - state

1. zdefinujeme state atribút filter s defaultnou hodnotou `'all'`
2. elementom `<button>` pridáme event listener `onClick`, ktorý bude meniť state atribút filter
3. `console.log(filter)`
4. aktívnemu tlačidlu pridáme triedu `'active'`



DRY

Don't Repeat Yourself

FilterItem

Vytvorme novú komponentu `FilterItem` s jedným parametrom - `value`.

```
<ul className="nav nav-pills mb-4">  
  <FilterItem value="all" />  
  <FilterItem value="active" />  
  <FilterItem value="all" />  
</ul>
```


FilterItem

```
<li className="nav-item">
  <button
    onClick={ () => setFilter( props.value ) }
    className={`nav-link ${filter === props.value? 'active' : ''}`}
    >{ props.value }</button>
</li>
```

Sú premenná `filter` a metóda `setFilter` definované vo `FilterItem` ?

FilterItem

Ak zadefinujeme `filter` a `setFilter` vo `FilterItem`...

Ako sa tieto komponenty medzi sebou „dohodnú“ ktorá z nich je aktívna?

Lifting State Up

V reacte je často potrebné, aby viaceré komponenty reflektovali rovnaké atribúty stavu.

Riešením je "pozdvihnúť" tento stav do ich najbližšieho spoločného predka.

FilterItem – zdieľaný stav

```
const [filter, setFilter] = useState('all');
```

```
<ul className="nav nav-pills mb-4">
```

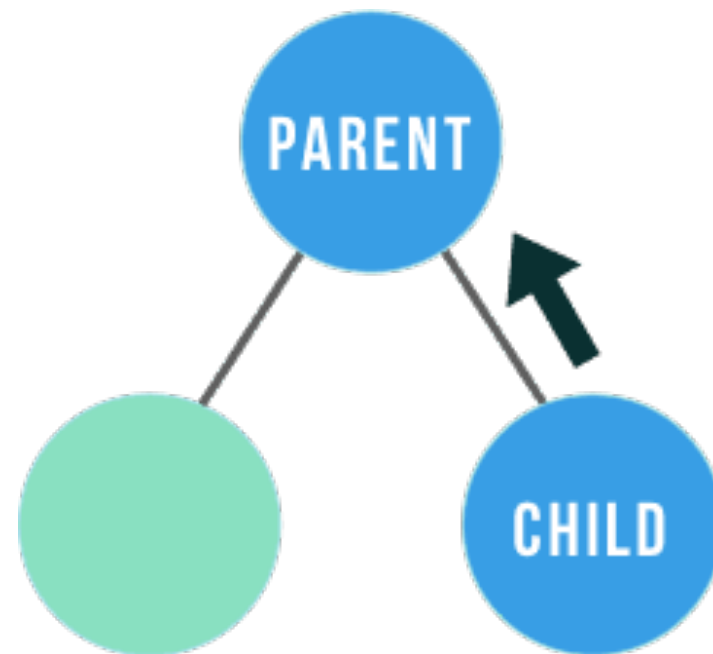
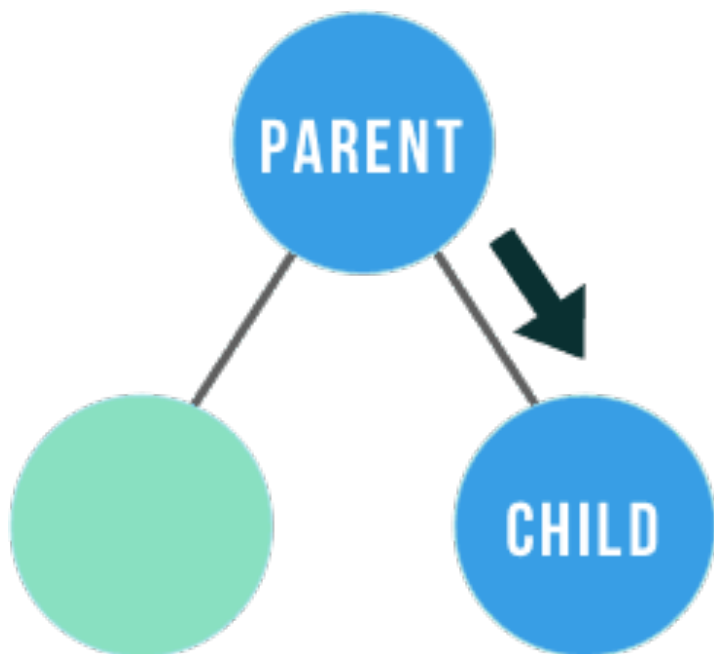
```
  <FilterItem value="all" />
```

```
  <FilterItem value="active" />
```

```
  <FilterItem value="all" />
```

```
</ul>
```

FilterItem – zdieľanie stavu



Zdzielanie stavu DOLE

```
const [filter, setFilter] = useState('all');
```

```
<FilterItem
```

```
  value="all"
```

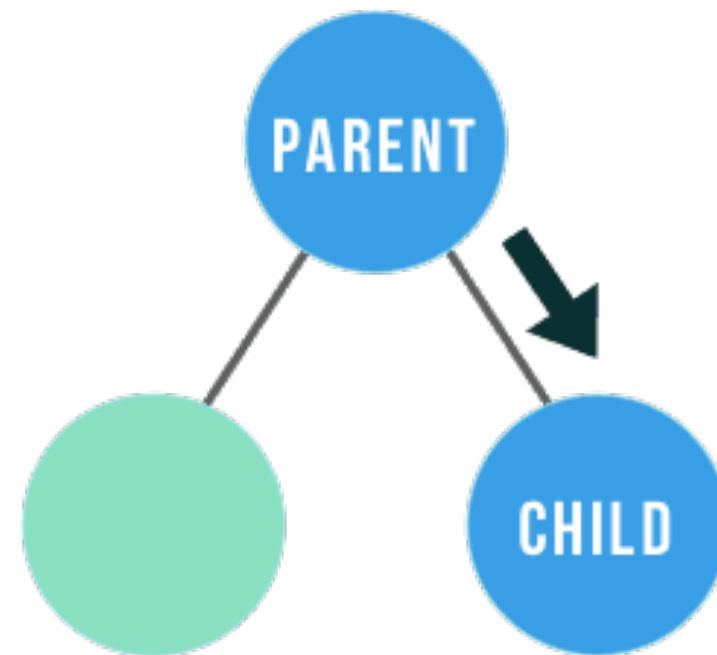
```
  filter={filter}
```

```
/>
```

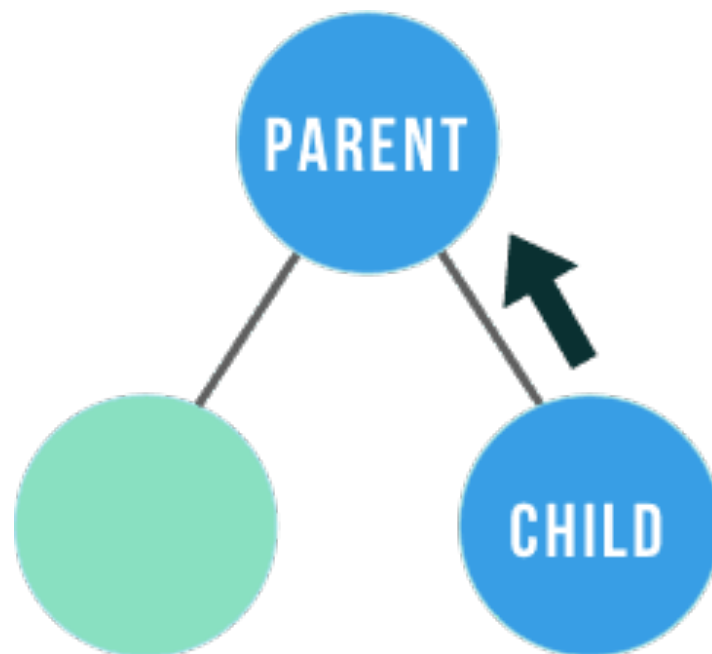
```
function FilterItem(props) {
```

```
  { props.filter }
```

```
}
```

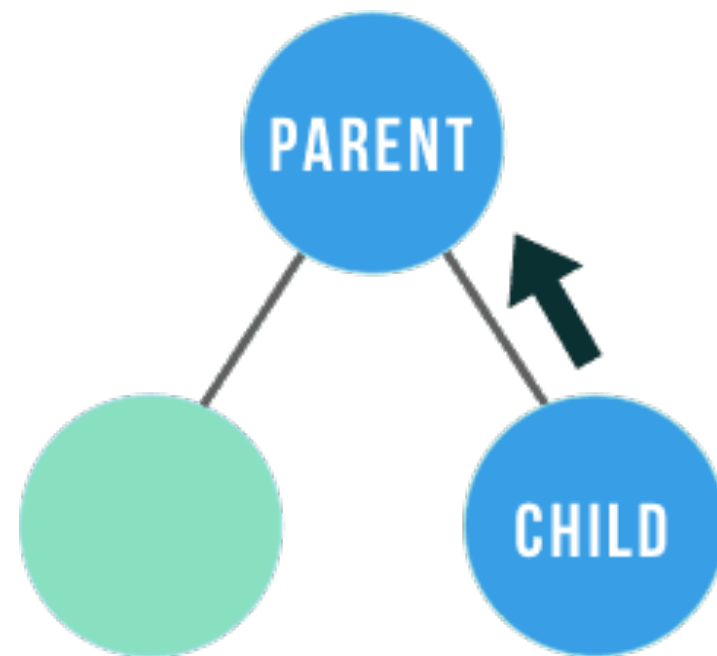


Zdzielanie stavu HORE



Zdieľanie stavu HORE

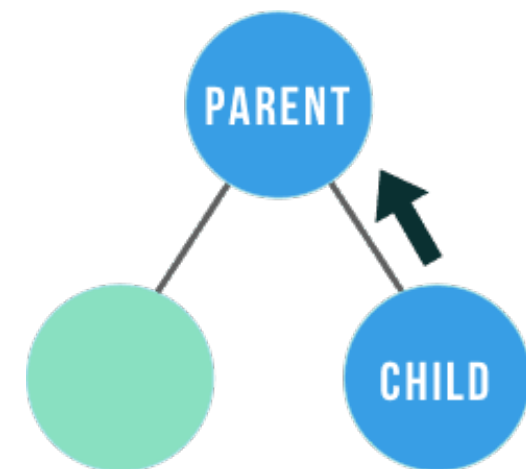
```
const [filter, setFilter] = useState('all');  
  
<FilterItem  
  value="all"  
  setFilter={setFilter}  
>  
  
function FilterItem(props) {  
  <button onClick={ props.setFilter('all') }  
}
```



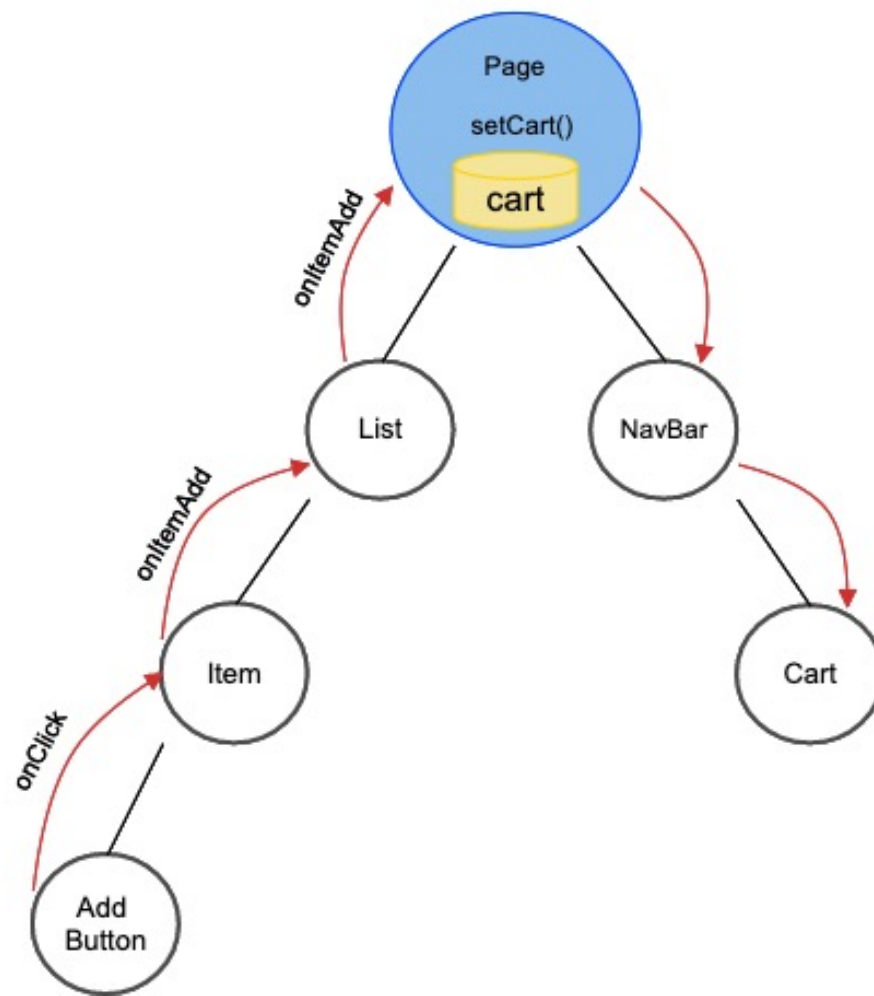
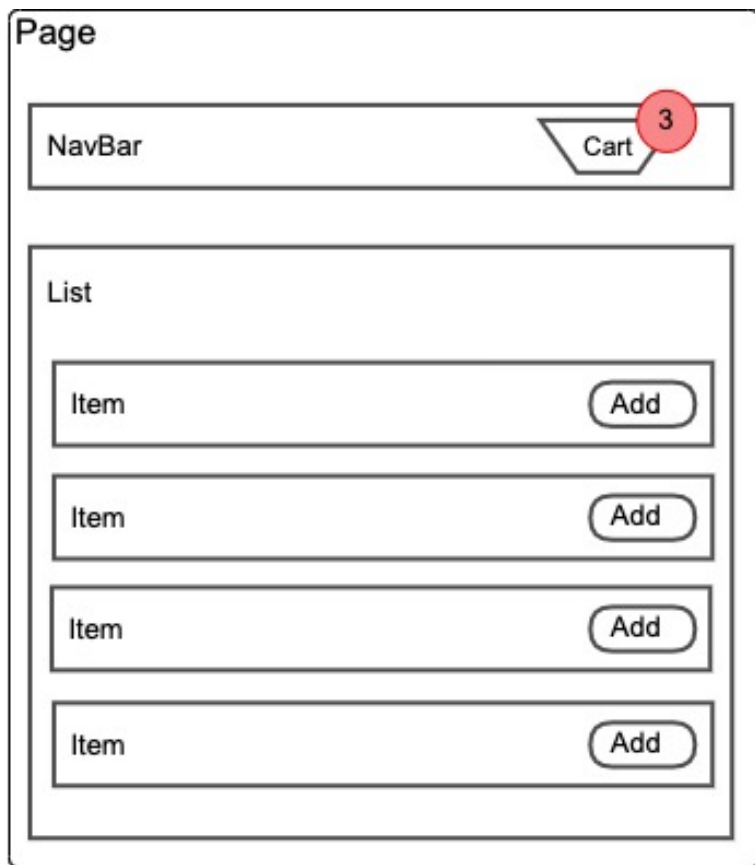
Zdieľanie stavu HORE

```
function ParentComponent() {
  return (
    <ChildComponent
      handleClick={() => { // any code to modify parent }}
    />
  ) }
```

```
function ChildComponent(props) {
  <button onClick={ props.handleClick }
  }
```



Props Drilling



Props Drilling

```
function AComponent() {  
  const [value, setValue] = useState();  
  return ( <BComponent value={value}/> );  
}  
  
function BComponent(props) {  
  return ( <CComponent value={props.value}/> );  
}  
  
...  
  
function XComponent(props) {  
  return ( <div>{ props.value }</div> );  
}
```

React Context

Kontext poskytuje spôsob, ako preniesť údaje cez strom komponentov bez toho, aby ste museli manuálne odovzdávať props na každej úrovni.

Context ~~ *Globálny State*

React Context

1. Creating the context
2. Providing the context
3. Consuming the context

Creating the context

Vytvorenie kontextu je veľmi podobné vytvoreniu stavu komponenty

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

```
const MyContext = createContext('Default Value');
```

Providing the context

Komponenty, ktoré majú mať možnosť získať hodnotu kontextu musia byť umiestnené v komponente **Context Provider**.

```
const MyContext = createContext('Default');
```

```
<MyContext.Provider value={ 'Value' }>
```

```
  <ChildComponent/>
```


```
</MyContext.Provider>
```

Consuming the context

```
import {useContext} from 'react'  
import {MyContext} from 'XYZComponent'  
  
function ChildComponent() {  
  const myContextValue = useContext(MyContext);  
  
  return (  
    <div>{value}</div>  
  )  
}
```


Congratulations!



 homm.fun

meme-arsenal.ru

ĎAKUJEME !

Ing. Slavomír Kožár

slavomir.kozar@innovis.sk

www.innovis.sk

