

React Local Storage

Paweł Leszczyński

Session vs Local Storage

Session Storage

- ▶ Dane są przechowywane przez przeglądarkę, tak długo, jak otwarta jest karta, zawierająca je.

Local Storage

- ▶ Dane są przechowywane lokalnie, nawet po zamknięciu przeglądarki.

Session Storage w todos

- ▶ W tworzonej aplikacji „todos”, każdorazowe odświeżenie strony, spowoduje utratę wprowadzonych przez nas zmian

todos

Items will persist in the browser local storage

<input checked="" type="checkbox"/> <i>Setup development environment</i>	Edit	Delete
<input type="checkbox"/> Develop website and add content	Edit	Delete
<input type="checkbox"/> Deploy to live server	Edit	Delete
<input type="checkbox"/> refresh tab	Edit	Delete

todos

Items will persist in the browser local storage

<input checked="" type="checkbox"/> <i>Setup development environment</i>	Edit	Delete
<input type="checkbox"/> Develop website and add content	Edit	Delete
<input type="checkbox"/> Deploy to live server	Edit	Delete

?

Local Storage

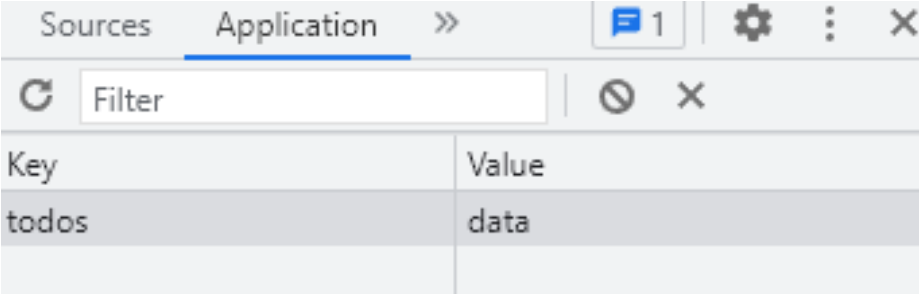
- ▶ Wpisując w konsoli "localStorage" lub "window.localStorage", uzyskamy informację o przechowywanych lokalnie danych.

```
window.localStorage
▼ Storage {length: 0} ⓘ
  length: 0
  ▼ [[Prototype]]: Storage
    ▶ clear: f clear()
    ▶ getItem: f getItem()
    ▶ key: f key()
    length: (...)
    ▶ removeItem: f removeItem()
    ▶ setItem: f setItem()
    ▶ constructor: f Storage()
    Symbol(Symbol.toStringTag): "Storage"
  ▼ get length: f length()
    length: 0
    name: "get length"
    arguments: (...)
    caller: (...)
    ▶ [[Prototype]]: f ()
    ▶ [[Scopes]]: Scopes[0]
  ▼ [[Prototype]]: Object
    ▶ constructor: f Object()
    ▶ hasOwnProperty: f hasOwnProperty()
    ▶ isPrototypeOf: f isPrototypeOf()
    ▶ propertyIsEnumerable: f propertyIsEnumerable()
    ▶ toLocaleString: f toLocaleString()
    ▶ toString: f toString()
    ▶ valueOf: f valueOf()
    ▶ __defineGetter__: f __defineGetter__()
    ▶ __defineSetter__: f __defineSetter__()
    ▶ __lookupGetter__: f __lookupGetter__()
    ▶ __lookupSetter__: f __lookupSetter__()
    __proto__: (...)
    ▶ get __proto__: f __proto__()
    ▶ set __proto__: f __proto__()
```

Set Item

- Za pomocą metody "setItem" jesteśmy w stanie dodać item'y do obiektu "Storage"
- Storage akceptuje tylko wartości typu "string".
- Przekazując dane do Storage'u, użyjemy metody: `JSON.stringify()`.

```
localStorage.setItem("todos", "data")
```

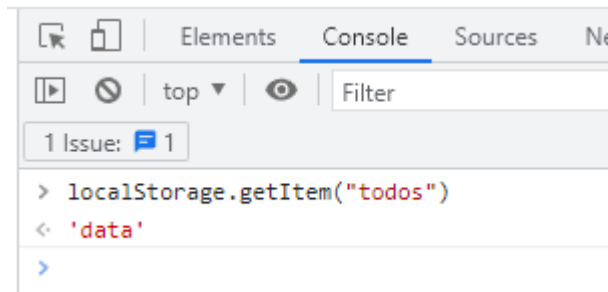


Sources Application >> 1 [Settings] [Close]	
Filter [X]	
Key	Value
todos	data

Get Item

- Za pomocą metody "getItem" za pomocą klucza, możemy znaleźć przypisaną mu wartość.
- Do konwersji odczytanego string'a do obiektu JS, użyjemy metody "JSON.parse()"

```
localStorage.getItem("todos")
```



Remove Item

Metoda "RemoveItem" umożliwia usunięcie item'u ze Storage'u, przy użyciu klucza("key")

Clear

Metoda "clear" wyczyści Storage ze wszystkich przechowywanych danych.

Local Storage w todos

W celu przeniesienia danych do local storage, musimy stworzyć Hook "useEffect" w "TodosLogic".

```
useEffect(()=>{  
  const temp=JSON.stringify(todos);  
  localStorage.setItem('todos',temp)  
},[todos]);
```

Local Storage w todos

- Następnie tworzymy funkcję, która będzie zwracać ze storage'a, umieszczone tam item'y.
- Stworzonej funkcji, musimy jeszcze użyć do przypisania wartości zmiennym.

```
function getInitialTodos(){  
  const temp=localStorage.getItem('todos');  
  const savedTodos=JSON.parse(temp);  
  return savedTodos || [];  
}
```

```
const [todos,setTodos]=useState(getInitialTodos());
```



Od teraz, lista z naszej aplikacji, nie będzie znikać po każdym odświeżeniu strony.