Właściwości komponentu (props)

1. Przygotujmy dane

- Utwórzmy plik z danymi do naszego komponentu: data.js.
  Umieścimy w nim dane różnego typu na temat 3 samochodów:
  Marka: string
  Model: string
  Max. Prędkość: int
  Dostępne paliwa: string[]

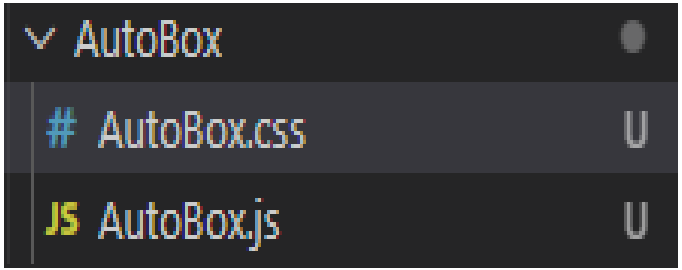
1. Przygotujmy dane cd

- Tworzymy więc przykładowe samochody
  przy założeniach z poprzedniego slajdu
- Utwórzmy też folder na nasz komponent np. AutoBox
- W nim będziemy potrzebowali pliku .js i .css

```
export default [
    {
        marka: "Opel",
        model: "Corsa",
        max_v: 180,
        paliwa: ["Diesel","Benzyna","LPG"]
    },
```

2. Stwórzmy komponent

- Utwórzmy też folder na nasz komponent np. AutoBox
- W nim będziemy potrzebowali pliku .js i .css dla naszego komponentu
- Stwórzmy szkielet AutoBoxa, który później będziemy mogli modyfikować




```
1  import React from 'react';
2  import './AutoBox.css'
3
4  export default function AnimalCard(){
5      return <h2>Autko</h2>
6  }
```

3. Wyświetlmy komponent

- W pliku App.js wyświetlmy nasz komponent. Najpierw dołączmy dane dla komponentu oraz sam komponent.
- Następnie metodą .map wyświetlmy zawartość komponentu

```
import data from './data';
import AutoBox from './Components/AutoBox/AutoBox';
```

```
function App() {
  return (
    <div className="wrapper">
      <h1>Samochody</h1>
      {data.map(auto=>(
        <AutoBox key={auto.marka}/>
      ))}

    </div>
  );
}
```

Samochody

Autko

Autko

Autko

4. Szybkie style

- Te style są oczywiście opcjonalne ale ułatwią nam czytanie zawartości naszego komponentu

```css
.wrapper{
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
  padding: 20px;
}
.wrapper h1{
  text-align: center;
  width: 100%;
}
```

5. Właściwości komponentu

- Wykorzystajmy teraz te dane które zapisaliśmy w pliku data.js.
  Do komponentu przekazujemy je jako obiekt props.

```
export default function AnimalCard(props){
```

A żeby wyciągnąć poszczególne dane musimy odwołać się do nich w nawiasach klamrowych

```
const {marka} = props
```

5. Właściwości komponentu

- Taki zabieg nazywamy dekonstrukcją.
  Ważne aby nazwa w nawiasach klamrowych zgadzała się z tym co zapisaliśmy w tablicy obiektów w pliku data.js
- Wykorzystajmy w końcu dane.
  W pliku AutoBox.js :

W bardziej złożonych przypadkach można używać notacji z kropką aby odwołać się do fragmentu danych np.
<h2>{props.marka}</h2>

```
export default function AnimalCard(props){
    const {marka} = props
    return <h2>{marka}</h2>
}
```

5. Właściwości komponentu

- Teraz musimy jeszcze wyświetlić te dane w naszej aplikacji

```
function App() {
  return (
    <div className="wrapper">
      <h1>Samochody</h1>
      {data.map(auto=>(
        <AutoBox
        key={auto.marka}
        marka={auto.marka}
        />
      ))}

    </div>
  );
}
```

5. Właściwości komponentu

- Marka jest typu string ale możemy przekazać dane dowolnego typu akceptowalnego przez JS. Dodajmy resztę informacji o samochodach
- Możemy to zrobić też jako listę właściwości aby nie dekonstruować każdej danej po kolei

```
export default function AutoBox({
    marka,
    model,
    max_v,
    paliwa
})
    {
    return (
    <div>
        <h2>{marka}</h2>
        <h3>{model}</h3>
        <h3>{max_v} km/h</h3>
        <h3>{paliwa.join(', ')}</h3>
    </div>
    );
}
```

6. Definiowane typu (PropTypes)

- PropTypes to sposób do określenia typu danych podawanych jako właściwość. Typ danych jest sprawdzany w trakcie pracy aplikacji i błędy spowodują jedynie error natomiast aplikacja i tak zostanie wyświetlona.
- Określenie typu jest o tyle ważne, że nigdzie wcześniej w kodzie tego nie zrobiliśmy i aby aplikacja była przejrzysta i możliwa do kontynuacji przez inne osoby warto określić typ danych już na tym etapie

6. Definiowane typu (PropTypes)

- W pliku AutoBox.js zaimportujmy odpowiedni package
- Teraz zdefiniujmy typy dla naszych danych

```
1   import React from 'react';
2   import './AutoBox.css'
3   import PropTypes from 'prop-types';
4
```

```
AutoBox.propTypes ={
    marka: PropTypes.string.isRequired,
    model: PropTypes.string.isRequired,
    max_v: PropTypes.number.isRequired,
    paliwa: PropTypes.arrayOf(PropTypes.string).isRequired
}
```

6. Definiowane typu (PropTypes)

- Marka i model oczywiście są stringami, prędkość w naszym przypadku jest int'em ale PropTypes scala int'y, float'y itp. w jednym typie „number".
- Paliwa w naszym przypadku to tablica stringów więc definiujemy ją nie tylko jako tablicę ale jeszcze tablicę stringów właśnie.
- IsRequired mówi kompilatorowi, że ma się spodziewać tej danej w tym typie i ma poinformować nas jeżeli jej nie dostanie

```
AutoBox.propTypes ={
    marka: PropTypes.string.isRequired,
    model: PropTypes.string.isRequired,
    max_v: PropTypes.number.isRequired,
    paliwa: PropTypes.arrayOf(PropTypes.string).isRequired
}
```

6. Definiowane typu (PropTypes)

- Aby sprawdzić jak to działa zmieńmy typ jednej z danych w pliku data.js

```
{
    marka: "BMW",
    model: "e36",
    max_v: '220',
    paliwa: ["Diesel","Benzyna"],
},
```

Prędkość jest teraz stringiem a konsola w przeglądarce wystosowała odpowiedni error

```
▸Warning: Failed prop type:          react-jsx-dev-runtime.development.js:117
Invalid prop `max_v` of type `string` supplied to `AutoBox`, expected
`number`.
    at AutoBox (http://localhost:3000/static/js/bundle.js:173:5)
    at App
```