attrecto
NEXT TECH DIGITAL SOLUTIONS

Frontend - React

ISO 9001 Certified

FT FINANCIAL TIMES 1000 Europe's Fastest Growing Companies

50™ Technology Fast 50 2016 CENTRAL EUROPE

500 Technology Fast 500 2016 EMEA

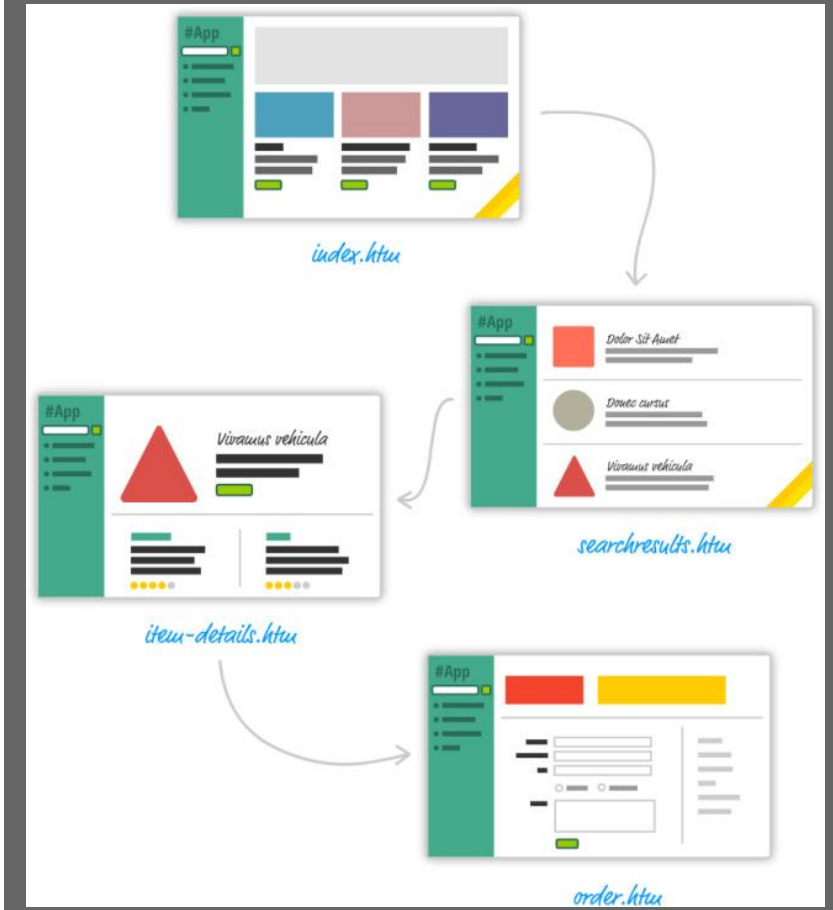
Frontend régen

Multi-Page Design

- független, különálló oldalak
- navigációnál teljesen új oldalra visz a böngésző
- minden alkalommal betöltődik a teljes oldal
- felesleges műveletek
- nem túl jó felhasználó élmény

Előnye:

- az alkalmazás állapota és a felhasználói felület szinkronizációjával nem kell foglalkozni



Frontend most

Single-Page App

Jellemzői:

- csak egyszer van oldalletöltés
- valójában nem navigálunk soha másik oldalra
- minden változás ugyanabban az oldalban fog megjelenni

Mire kell figyelni:

- az adatok szinkronban legyenek a UI-jal
- a DOM manipuláció lassú
- sablonok / template-ek

Elvárások:

- szép
- gyors
- egyszerű

React

by Facebook

SPA esetén megoldandó probléma:

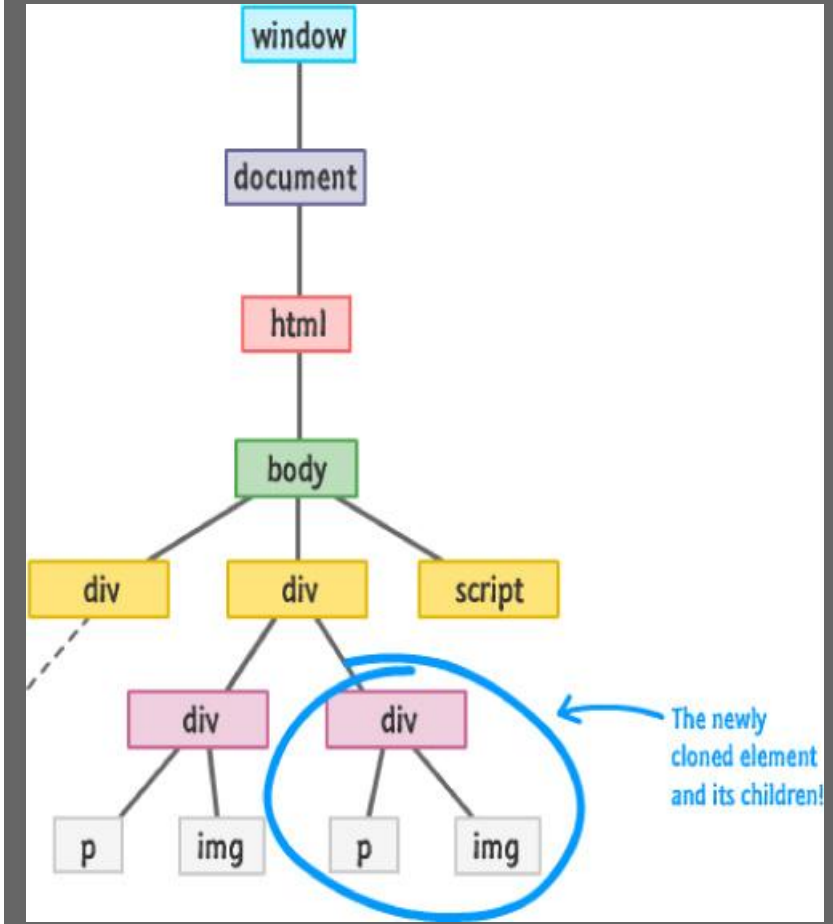
- a UI állapotának követése nehéz és időigényes
- a legfonotsabb az aktuális/legutolsó állapot

Erre nyújt megoldást a React:

- csak az aktuális állapottal kell törődnünk

A többit elintézi nekünk a React:

- automatikus UI update
- gyors DOM manipuláció:
  - virtual DOM
  - real DOM



Komponensek

A felhasználói felületet osszuk több kisebb egységre, un. komponensre:

- moduláris
- kompakt
- önálló egységek

Az egyes komponensek implementáslásának módja lehet:

- klasszikus template megoldás:
  - template-be ágyazott javascript
  - framework által korlátozott eszköztár
- React megközelítés:
  - javascript-be ágyazott UI
  - minden js feature rendelkezésre áll
  - jsx

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}


class Welcome extends React.Component {
    render() {
        return <h1>Hello, {this.props.name}</h1>;
    }
}


ReactDOM.render(
    <Welcome name="React" />,
    document.getElementById('root')
);
```

Komponensek

Komponensek típusai:

- functional / class
- stateless / statefull
- container / presentational

Jellemzői:

- PascalCase elnevezés
- className, htmlFor
- style object camelCase
- style prefix WebkitTransition, msTransition
- onClick, onChange
- babel-polyfill, whatwg-fetch

```
const Hi = (props) => {
   return <h1>Hi, {props.name}</h1>;
};

const Hello = props => <Hi name={props.name}/>;

class WelcomeComponent extends React.Component {
   render() {
       return <Hello name="Zoli"/>;
   }
}
```

JSX

Jellemzői:

- olyan, mint a HTML
- a Javascript nyelv szintaxisának kiterjesztése
- olyan, mintha template nyelv lenne
- a Javascript teljes eszköztára használható vele

viszont:

- a böngészők nem ismerik
- meg kell értetni a böngészővel:
  - babel
  - fejlesztői környezet: build
  - fordítás böngészőben runtime

```
ReactDOM.render(
    React.createElement(
        "div",
        null,
        React.createElement(
            "h1",
            null,
            "Hello, React!"
        ),
    ),
    document.getElementById('root')
);

ReactDOM.render(
    <div>
        <h1>
            Hello, React!
        </h1>
    </div>,
    document.getElementById('root')
);
```

props

- a szülőtől/hívótól kapja a komponens
- jsx-ben hasonló a szintaxisa, mint a html attribútumoknak
- this.props.children egy speciális props
- a komponens nem változtathatja a saját props-szait

```
const Hi = (props) => {
    return <h1>Hi, {props.name}</h1>;
};

class Welcome extends React.Component {
    render() {
        return (
            <Hi
                name="Zoli"
                valami={this.props.valami}
                megvalami={'még valami'}
            />
        );
    }
}

class Parent extends React.Component {
    render() {
        return <Welcome valami="Valami" />;
    }
}
```

state

- adatok, amelyek folyamatosan változnak a komponens életciklusa során
- az értékeket a render-ben használjuk
- plain object
- setState()

```
class Welcome extends React.Component {
    state = {
        fieldname: null
    };

    onClick = () => {
      this.setState({fieldname: 'Mező neve'});
    };

    render() {
        return (
            <Hi
                name="Zoli"
                valami={this.props.valami}
                megvalami={this.state.fieldname}
                onClick={this.onClick}
            />
        );
    }
}
```

Lifecycle - mount

- constructor(props)
  - mount előtt fut
  - default state inicializálásra
  - super(props)
  - eseménykezelők bind-olása az instance-hoz
  - setState nem hívható
- componentWillMount()
  - render előtt fut
  - setState hívható, de nem triggerel újabb render-t
  - server side rendering
- render()
  - pure function
  - kötelező
- componentDidMount()
  - render-elés után fut
  - itt javasolt pl. ws-ek hívása

```
class Welcome extends React.Component {
    constructor(props) {
        super(props);
        this.state = {
            fieldname: null
        };
        this.onHiClick.bind(this);
    }

    componentDidMount() {
        this.setState({fieldname: 'Mező neve'});
    }

    onHiClick() {
        this.setState({fieldname: 'Mező'});
    }

    render() {
        return
        <Hi name="Zoli"
            fieldname={this.state.fieldname}
            onClick={this.onHiClick}/>;
    }
}
```

Lifecycle - update

- componentWillReceiveProps(nextProps)
  - mielőtt a komponens megkapja az új props-okat
  - akkor is fut, ha a props nem változik
- shouldComponentUpdate(nextProps, nextState)
  - történjen e renderelés
- componentWillUpdate(nextProps, nextState)
  - miután a komponens megkapta az új props-okat, és state-et
  - setState nem hívható
- render()
- componentDidUpdate(prevProps, prevState)
  - render után

```
class Welcome extends React.Component {
    componentDidUpdate() {
        // do something
    }

    render() {
        return (
            <div/>
        );
    }
}
```

Lifecycle - others

Unmount:

- componentWillUnmount()
  - mielőtt a komponens megszűnik

Hibakezelés

- componentDidCatch(error, info)

State:

- setState(updater[, callback])
  - state változtatása
  - mindig újra render-elés az eredménye
  - aszinkron
  - updater function / object

Force update

- forceUpdate(callback)

```
this.setState({
   fieldname: 'Mező neve'
});
```

```
this.setState((prevState, props) => {
   return {
        counter: prevState.counter + props. step
   };
});
```

create-react-app

Git beállítása:

- klónozzuk le a repot
- nevezzük át a mappát, hogy ne legyen benne nagybetű

create-react-app jellemzői:

- alapból feltelepít egy fejlesztői környezetet
- beállítja a szükséges dependenciákat
- létrehozza az alakalmazás belépési pontját
- létrehoz egy alap komponenst
- egy egyszerű paranccsal tudjuk a kódunkat futtatni
- van benne hot-reload
- azonnal használható fejlesztésre
- lehetőség van production build-et létrehozni

```
git clone --branch 00-init
https://github.com/attrecto/Attrecto-Academy-JS-Fro
ntend.git

npm install -g create-react-app

npx create-react-app attrecto-academy-js-frontend

create-react-app attrecto-academy-js-frontend


yarn start
http://localhost:3000/

yarn build
```

Csomagok

Szükséges csomagok:

- bootstrap@3
- react-bootstrap
- react-router-dom
- react-router-bootstrap

```
yarn add bootstrap@3 react-bootstrap
react-router-dom react-router-bootstrap
```

Köszönöm a figyelmet!

Attrecto Zrt.
Attrecto Next Tech Digital Solutions

H-9024 Győr, Wesselényi str. 6.
info@attrecto.com

Öteletek a továbbfejlesztésre

- Dashboard menüpont
  - legtöbb badge-dzsel rendelkező felhasználók
  - melyik badge van meg a legtöbb felhasználónak
  - statisztikák
  - stb...
- badge adatlap:
  - adott badge-el rendelkező user-ek listája
  - user hozzárendelése az adott beadge-hez

Köszönöm a figyelmet!

Attrecto Zrt.
Attrecto Next Tech Digital Solutions

H-9024 Győr, Wesselényi str. 6.
info@attrecto.com

HOGYAN HASZNÁLD A TEMPLATET?

- Jobb klikkel adj hozzá új diát az oldalsávon
- és válaszd ki a szöveghez illő elendezést (Lesz kép? Ha igen, akkor ez fél oldalas, vagy teljes oldalas? Hosszú a szöveg? Két hasábba kell tördelni?)
- Ha másolod a szövegeket, CTRL+ Shift+ V kombinációval illeszd be, hogy a szövegmező formázását kapja. Így nem esik szét a prezentáció, amikor kivetíted vagy letöltöd pdf formátumban.

EGYSOROS CÍM

Használd ki az alcím helyét, ha szükség van rá

Figyelj rá, hogy a címeket végig nagybetűvel írd
(Kijelölés/ Formázás/ Nagybetűs írás/ NAGYBETŰ)

A cím alatti kék csíkot annak megfelelően illeszd, hogy hány soros a cím. (Innen és a következő diáról másolhatod.)

NAGYON HOSSZÚ KÉTSOROS CÍM

Válassz alcím nélküli diatípust, ha nincs szükséged rá

Ki tudsz emelni fontos információkat, idézeteket egész oldalas képek használatával

MIT ÉRDEMES MÉG TUDNI?

- Figyelj rá, ha világos képet választasz, akkor sötét legyen a logó,
- ha sötét képet, akkor fehér legyen a logó
- figyelj, hogy olvasható legyen a szöveged, akkor is ha kivetíted (méret, szín)

A használt nyelvnek megfelelő referencia és management oldalakat is be tudsz illeszteni

Stock képek forrása:

https://drive.google.com/drive/folders/0B2nU7ytUu_TWc3RFZTF4bGtFdHc?usp=sharing