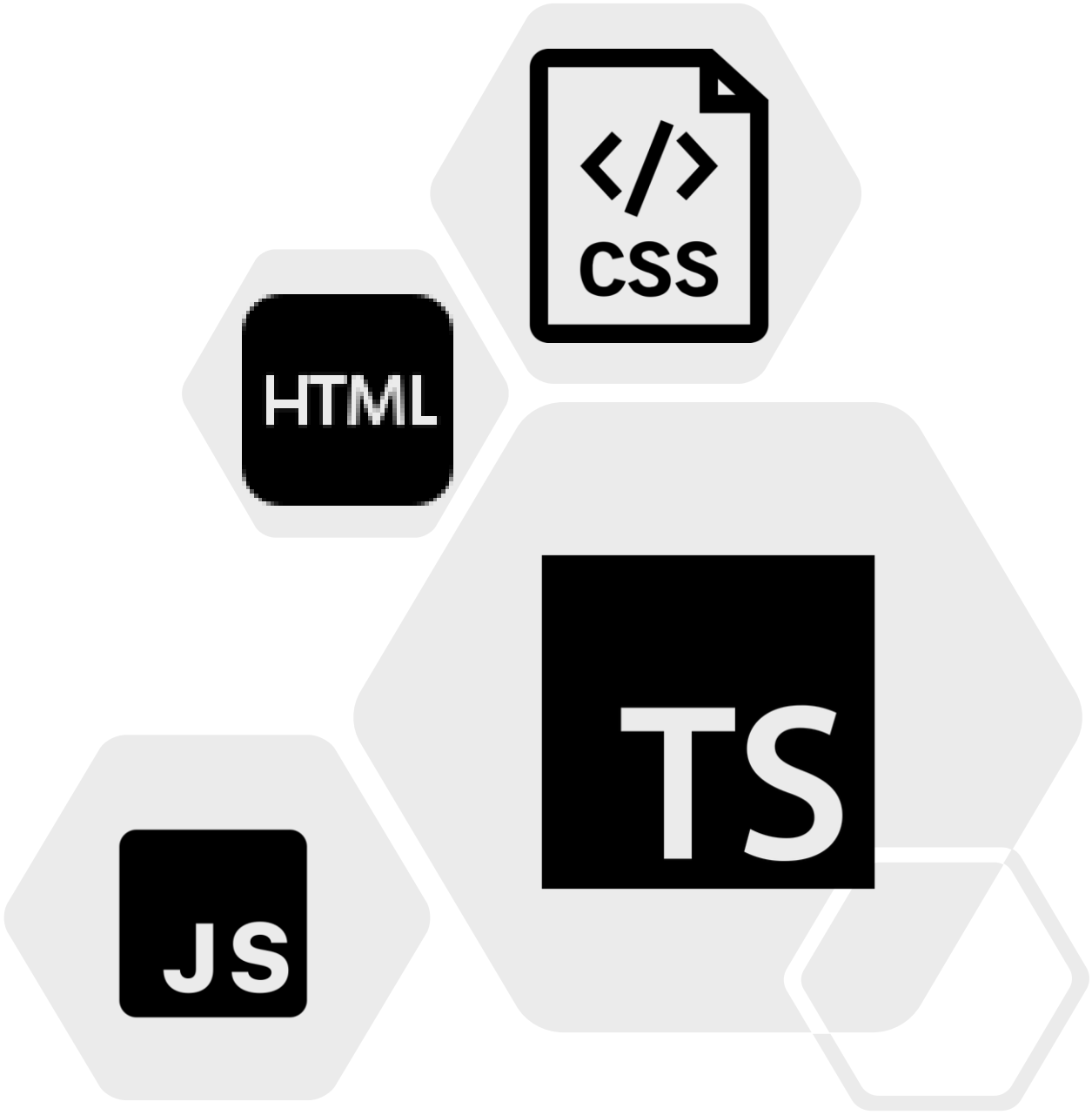


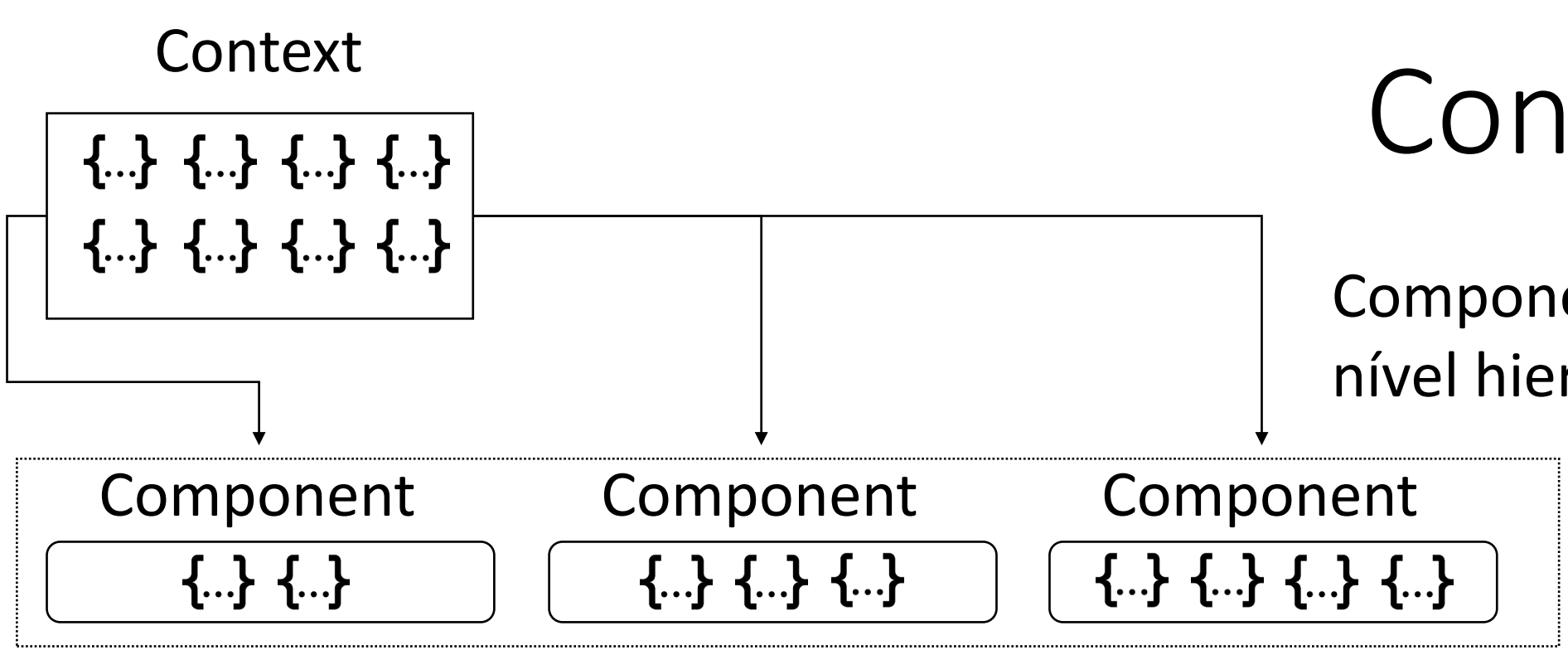
Context

Context (contexto)

O contexto fornece uma maneira de passar dados pela árvore de componentes sem ter que passar props manualmente em todos os níveis.

O “context” foi projetado para compartilhar dados que podem ser considerados “globais” para uma árvore de componentes React, como o usuário autenticado atual, tema ou idioma preferido. O context não armazena apenas um tipo de dado, ele pode armazenar uma estrutura, ou seja, um objetivo com vários dados globais.

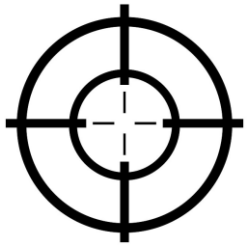
Context



Componentes no mesmo nível hierárquico.

O objeto do context é repassar dados, como propriedades globais para outros componentes do projeto react. Uma vez que o fluxo de envio de propriedades segue sempre de um componente pai para um componente filho, não é possível o repasse de propriedades entre componentes do mesmo nível – o contexto auxilia nestas situações.

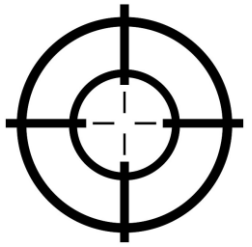
Estrutura de um context



```tsx
import { createContext } from "react";
import contextType from "../../domain/type/contextType";

const AuthenticationContext = createContext<contextType>({
    token: '',
    userApp: null,
    setData: () => { }
})

export default AuthenticationContext
```

```tsx
import UserApp from "../../model/userApp"

type contextType = {
    userApp: UserApp,
    token: string,
    setData: (token: string, userApp: UserApp) => void
}

export default contextType
```

Context Provider



```
const AuthenticationContext = createContext<contextType>
```

Todo objeto context possui um componente “Provider”, que permite que os componentes de consumo recebam as propriedades do contexto.

```
<AuthenticationContext.Provider value={   {...}   }>
    <>
</AuthenticationContext.Provider>
```

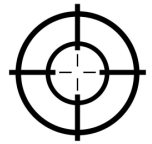
No atributo “value” são passados todos os dados que serão compartilhados entre os componentes, como propriedades.

Componente que receberá os dados passados pelo provider.

```
export default class AuthenticationContextComponent
    constructor(props) {
        super(props)
        this.state = { token: '', userApp: null }
        this.setData = this.setData.bind(this)
    }
    setData(newToken: string, newUserApp: UserApp) {…
    }
    render() {
        let contextData = {
            token: this.state.token,
            userApp: this.state.userApp,
            setData: this.setData
        }
        return (
            <AuthenticationContext.Provider value={contextData}>
            {this.props.child}
            </AuthenticationContext.Provider>
        )
    }
}
```

Para atualizar os dados (propriedades) de um context é necessário utilizar um componente para alterar o valor do provider e repassar os novos dados aos componentes filhos do context.

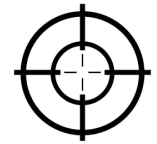
Context Component



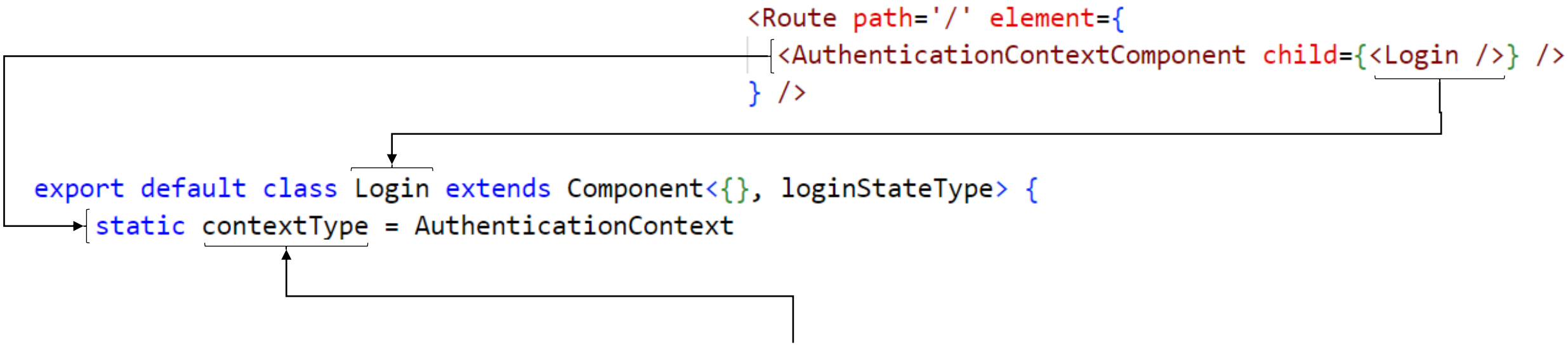
```
export default class App extends Component {
  render() {
    return (
      <BrowserRouter>
        <Routes>
          <Route path='/' element={
            <AuthenticationContextComponent child={<Login />} />
          } />
          <Route path='/home' element={
            <AuthenticationContextComponent child={<Home />} />
          } />
          <Route path='/register' element={
            <AuthenticationContextComponent child={<Register />} />
          } />
          <Route path='*' element={
            <AuthenticationContextComponent child={<NoPage />} />
          } />
        </Routes>
      </BrowserRouter>
    )
  }
}
```

```
export default class AuthenticationContextComponent
```

Context Component



Acessando o context



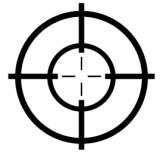
Todo componente do react, possui um campo estático (static), que é utilizado para referenciar ao um context. A partir desse campo, um component pode recuperar (acessar) o contexto onde está envolvido e, assim, acessar os dados fornecidos pelo contexto.

```tsx
export default class Login extends Component<{}, loginStateType> {
    static contextType = AuthenticationContext
    private username: string
    private password: string
    constructor(props) {···
    }

    loadData(event) {···
    }

    authenticate(event) {···
    }

    doComponent() {···
    }

    render() {
        let authenticationContext: any = this.context
        let token = authenticationContext.token
        return (
            <HomeFilter token={token} component={this.doComponent()} />
        )
    }
}
```

O tipo do context precisa ser informado. Ele será passado para o campo estático.

O context é “inserido” na variável contextType, mas é lido pela variável “context”.

Acessando o context



```
export default class AuthenticationContextComponent extends Compo
    constructor(props) {···
    }
    setData(newToken: string, newUserApp: UserApp) {
        let state = { token: newToken, userApp: newUserApp }
        this.setState(state)
    }
    render() {
        let contextData = {
            token: this.state.token,
            userApp: this.state.userApp,
            setData: this.setData
        }
        return (
            <AuthenticationContext.Provider value={contextData}>
                {this.props.child}
            </AuthenticationContext.Provider>
        )
    }
}
```

O context, por padrão, apenas repassa dados aos seus componentes filhos. Portanto, não é possível, diretamente, alterar os dados de um context.

Por isso utiliza-se um outro componente que fornece a alteração ao Provider.

https://github.com/gerson-pn/react-web-app-typescript-spring