.StartSe
<tech/academy>

Encapsulamento - Atributos e Métodos Privados

Encapsulamento

- Os métodos de uma classe permitem isolar ou esconder os atributos de um objeto para que eles sejam acessados somente através dos métodos
    - dessa forma os dados e o comportamento de um objeto estão encapsulados em uma mesma entidade: o **objeto**

Encapsulamento

- O encapsulamento é uma forma de **esconder** ou **ocultar** os dados de uma classe
    - permite proteger o acesso aos atributos (propriedades) de um objeto
    - ao invés de acessar um atributo diretamente, por exemplo, é necessário utilizar um método para atribuir ou recuperar algum dado

Encapsulamento

- Uma forma de encapsular os dados de uma classe é declarar os atributos dessa classe como **privados** e utilizar métodos para acessá-los
- E JavaScript, utilizamos **#** no início do identificador de um atributo para declará-lo como **privado**

Encapsulamento

Os atributos **titulo** e numPag da classe Livro só podem ser acessados fora da classe através do método **getDados()**

```
class Livro extends Produto {
    #titulo;
    #numPag;
    constructor(tipo,titulo,numPag) {
        super(tipo);
        this.#titulo = titulo;
        this.#numPag = numPag;
    }
    getDados() {
        return(`
        Tipo: ${super.getDados()}
        Titulo: ${this.#titulo}
        Num.Pág.: ${this.#numPag}
        `);
    }
}
```

Encapsulamento

- Também é possível **esconder** ou **ocultar** métodos de uma classe tornando o método privado
    - basta utilizar **#** no início do identificador de um método para declará-lo como privado
    - dessa forma, o **método privado** será acessível apenas dentro da própria classe
        - não poderá ser chamado através de um objeto que foi instanciado a partir da classe que contém o método privado

Encapsulamento

```
1  class Produto {
2      #tipo;
3      #valor;
4      constructor(tipo, valor) {
5          this.#validaValor(valor);
6          this.#tipo = tipo;
7          this.#valor = valor;
8      }
9      #validaValor(valor) {
10         if (typeof(valor) !== 'number') {
11             console.log('Valor inválido!');
12         }
13     }
14     getDados() {
15         return `
16         Tipo: ${this.#tipo}`;
17     }
18 }
19
```

o método **privado** **validaValor** só pode ser acessado dentro da classe **Produto**
(nem as sublcassses nem os objetos instanciados a partir da classe Produto terão acesso a esse método)

.StartSe
<tech/academy>