c l a s s e s

THIAGO DELGADO PINTO
thiago_dp (at) yahoo (dot) com (dot) br

versão: 2020.01.20

notação de classe

introduzida no ES6

substitui a sintaxe de funções como classes

- internamente, a representação anterior continua

ainda possui limitações

- não suporta destrutor, encapsulamento, etc.

sintaxe básica

```
class NomeDaClasse {
  constructor( ... ) { ... }
  metodo1( ... ) { ... }
  metodo2( ... ) { ... }
  metodoN( ... ) { ... }
}
```

exemplo 1

```
class Pessoa {
  constructor( nome ) {
    this.nome = nome;
  }
}
const p = new Pessoa( 'Ana' );
console.log( p.nome );
```

exemplo 2

```
class Pessoa {
  nome = '';
  constructor( nome ) {
    this.nome = nome;
  }
}
const p = new Pessoa( 'Ana' );
console.log( p.nome );
```

exemplo 3

```
class Pessoa {
  nome = '';
  sobrenome = '';
  constructor( nome, sobrenome ) {
    this.nome = nome;
    this.sobrenome = sobrenome;
  }
  nomeCompleto() {
    return this.nome + ' ' + this.sobrenome;
  }
}
const p = new Pessoa( 'Clarice', 'Linspector' );
console.log( p.nomeCompleto() );
```

expressão de classe

permite declaração dinâmica, similar à uma função

exemplo

```
const Pessoa = class {
  constructor( nome ) {
    this.nome = nome;
  }
};
const p = new Pessoa( 'Ana' );
console.log( p.nome );
```

métodos como propriedades

ocorre através do uso das palavras reservadas **`get`** ou **`set`**

permite definir uma função para ser chamada como um atributo

ela não poderá ser chamada como uma função!

`get` faz com que uma função seja invocada para obter um valor

`set` faz com que uma função seja invocada para definir um valor

métodos como propriedades – exemplo 1

```
class Pessoa {
  _nome = '';
  constructor( nome ) { this._nome = nome; }
  get nome() { return this._nome; }
}
const p = new Pessoa( 'Ana' );
console.log( p.nome ); // Ana
console.log( p.nome() ); // Error
```

métodos como propriedades – exemplo 2

```
class Pessoa {
  _nome = '';
  constructor( nome ) { this._nome = nome; }
  get nome() { return this._nome; }
  set nome( valor ) { this._nome = valor; }
}
const p = new Pessoa( 'Ana' );
console.log( p.nome ); // Ana
p.nome = 'Sérgio';
console.log( p.nome ); // Sérgio
```

métodos como propriedades – exemplo 3

```
class Pessoa {
  _nome = '';
  constructor( nome ) { this.nome = nome; } // Chama setter
  get nome() { return this._nome; }
  set nome( valor ) { this._nome = nome; }
}
const p = new Pessoa( 'Ana' );
console.log( p.nome ); // Ana
p.nome = 'Sérgio';
console.log( p.nome ); // Sérgio
```

herança – exemplo 1

```
class Documento {
  validar() { return false; }
}

class CPF extends Documento {
  constructor( numero ) { this.numero = numero; }
  validar() { /* ...realiza validação... */ }
}
```

herança – exemplo 2

```
class Cliente {
  constructor( nome, email ) {
    this.nome = nome;
    this.email = email;
  }
}

class ClientePJ extends Cliente {
  constructor( nome, email, cnpj ) {
    super( nome, email );
    this.cnpj = cnpj;
  }
}

const clientes = [ new Cliente( 'Ana', 'ana@site.com' ),
  new ClientePJ( 'Acme', 'contato@acme.com', '38.987.303/0001-12' ) ];
```

herança – exemplo 3

```
class Cliente {
  constructor( nome, email ) { this.nome = nome; this.email = email; }
  validar() { /* ... */ }
}

class ClientePJ extends Cliente {
  constructor( nome, email, cnpj ) { super( nome, email ); this.cnpj = cnpj; }

  validar() {
    super.validar();
    if ( ! this.cnpj || this.cnpj.length != 14 ) {
      throw new Error( 'CNPJ deve ter 14 dígitos numéricos.' );
    }
    const cnpj = new CNPJ( this.cnpj );
    if ( ! cnpj.validar() ) {
      throw new Error( 'CNPJ inválido.' );
    }
  }
}
```

métodos estáticos

```
class Calculo {
  static soma( x, y ) {
    return x + y;
  }
  static media( x, y ) {
    return Calculo.soma( x, y ) / 2;
  }
}

Calculo.soma( 10, 20 ); // 30
Calculo.media( 10, 20 ); // 15
```

referências usadas

MDN. **Referência JavaScript**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference

fim

Versão 1: 2020.01.20