

≡ 05

## Será que cabe na cesta?

Vimos o parâmetro `rest` (i.e. resto) nesse capítulo, quando enviamos diversos métodos para serem monitorados no `Proxy`. Agora, considere que um aluno está tentando usar `rest`, mas sem sucesso:

```
// o código abaixo tem um problema, não funciona
class Cesta{
    constructor(tipo, [items...]){
        //lógica
    }
}
```

E em algum outro lugar, alguém que cria uma instância da classe `Cesta`:

```
let cesta = new Cesta('fruta', ['banana', 'tomate', 'maçã']);
```

Vemos que tem algo errado! Como podemos consertar esse código com `rest`, para que possamos passar infinitas frutas, e no `constructor` declarar apenas uma variável para elas?

*Selecione uma alternativa*

**A** Alteramos o `constructor`:

```
constructor(tipo, [...items]) {
    //lógica
}
```

E chamamos:

```
let cesta = new Cesta('fruta', 'banana', 'tomate', 'maçã');
```

**B** Alteramos o `constructor`:

```
constructor(tipo, ...items) {
    //lógica
}
```

E chamamos:

```
let cesta = new Cesta('fruta', ['banana', 'tomate', 'maçã']);
```

**C** Alteramos o `constructor`:

```
constructor(tipo, items...) {
    //lógica
}
```

E chamamos:

```
let cesta = new Cesta('fruta', 'banana', 'tomate', 'maçã');
```

**D** Alteramos o `constructor`:

```
constructor(tipo, ...items) {
    //lógica
}
```

}

E chamamos:

```
let cesta = new Cesta('fruta', 'banana', 'tomate', 'maçã');
```