Spread

SPREAD

La sintassi dello spread consente di passare gli elementi di un array come argomenti ad una funzione.

```
function sum(x, y, z) {
  return x + y + z;
}
const numbers = [1, 2, 3];
sum(...numbers); // 6
```

SPREAD

- DOMANDA cosa succede in questo caso?

```
function sum(x, y, z) {
return x + y + z;
}
sum(...[1, 2]);
```

  - 3
  - 5
  - `"1,2undefined"`
  - NaN
  - TypeError

SPREAD

- DOMANDA cosa succede in questo caso?

```
function sum(x, y, z) {
return x + y + z;
}
sum(...[1, 2]);
```

  - 3
  - 5
  - `"1,2undefined"`
  - NaN
  - TypeError

> RISPOSTA
> NaN, ovvero il risultato di una operazione matematica andata "male".
>
> È uno degli argomenti più spinosi di JavaScript, tanto caro ai suoi detrattori, ma in realtà comune a tutti i linguaggi e piattaforme che usano la specifica IEEE 754 per i `float`!

SPREAD

Non abbiamo ancora visto queste funzioni ma secondo voi che succede?

```
new Array(10).fill(1 + undefined).join("") + " Batman!";
```

SPREAD

- DOMANDA

```
let sum = (a, b, c) => a + b;
sum(1, ...[2, 3]);
```

  - 5
  - 6
  - undefined
  - 3

SPREAD

- DOMANDA

```
let sum = (a, b, c) => a + b;
sum(1, ...[2, 3]);
```

  - 5
  - 6
  - undefined
  - 3

RISPOSTA: 3

SPREAD

- La sintassi spread può essere usata per concatenazione, copia e inserimenti negli array

```
const a = [1, 2];
[0, ...a, 3]; // [0, 1, 2, 3]
```

```
const a = [1, 2];
const b = [3, 4];
[...a, ...b]; // [1, 2, 3, 4]
```

```
const a = [1, 2, 3];
const b = [...a];
b; // [1, 2, 3]
```

SPREAD

- La sintassi spread può essere usata per la creazine di nuovi oggetti che includono tutte le proprietà di uno o più altri oggetti:

```
const original = { a: 1, b: 2 };
const clone = { ...original };

original === clone;
// false – anche se hanno proprietà “uguali” sono oggetti diversi
```

Attenzione che la “clonazione” avviene solo sul primo livello di profondità degli oggetti:

```
const original = { a: 1, nested: { c: 2 } };
const clone = { ...original };

original === clone; // false – sono oggetti diversi

clone.nested.c = 3;
original; // { a: 1, nested: { c: 3 } }

original.nested === clone.nested; // true!!!!
```

SPREAD

Negli oggetti fondere lo spread operator con le normali definizioni di proprietà ci permette di creare “copie diverse”:

```
const original = { a: 1, b: 2, c: 3 };
const modified = {
  ...original,
  b: 9999
};

modified; // { a: 1, b: 9999, c: 3 }
```

Questo è il modo per “modificare” oggetti senza “mutarli”, rispettando i vincoli di immutabilità!