Oggetti in JavaScript

📦 Cosa sono gli oggetti in JavaScript?

Gli **oggetti** sono un tipo di dato che consente di **raccogliere coppie chiave-valore**. Ogni valore può essere un dato semplice (stringa, numero...) o complesso (array, altro oggetto, funzione).

Sintassi base:

```
let oggetto = { chiave1: valore1, chiave2: valore2 };
```

Esempio:

```
let aboutMe = { city: 'Madrid, ES', hair: 'brown' };
```

🧭 Accesso alle proprietà

🔹 Notazione con punto (dot notation)

```
aboutMe.city; // 'Madrid, ES' aboutMe.gender; // undefined (non esiste)
```

🔸 Notazione con parentesi (bracket notation)

```
aboutMe['hair']; // 'brown' aboutMe['gender']; // undefined let prop = 'city';
aboutMe[prop]; // 'Madrid, ES'
```

Bracket notation è utile quando il nome della proprietà è dinamico.

🛠️ Modifica degli oggetti

Puoi:

- **Modificare** una proprietà esistente: `aboutMe.hair = 'blue';`
- **Aggiungere** una nuova proprietà: `aboutMe.gender = 'female';`
- **Eliminare** una proprietà:
  `delete aboutMe.gender;`

🧩 Oggetti negli array

Un array può contenere oggetti:

```
let myCats = [ { name: 'Cleo', age: 8 }, { name: 'Simba', age: 1 } ];
```

🧬 Oggetti come argomenti

Gli oggetti possono essere passati alle funzioni:

```
function describeCat(cat) { console.log('This cat is ' + cat.age + ' years old.');
} describeCat(myCat);
```

🔧 Metodi (funzioni negli oggetti)

Un oggetto può avere proprietà che sono funzioni, chiamate **metodi**:

```
let myCat = { meow: function() { console.log('meowww'); }, eat: function(food) {
console.log('Yum, I love ' + food); } };
```

Si invocano così:

```
myCat.meow(); myCat.eat('fish');
```

⚠️ `this` nei metodi

All'interno di un metodo, `this` fa riferimento **all'oggetto che lo contiene**:

```
let myCat = { name: 'Cleo', sayName: function() { console.log('I am ' +
this.name); } }; myCat.sayName(); // I am Cleo
```

🔍 `Object.keys()`

Restituisce un array con i nomi delle proprietà di un oggetto:

```
Object.keys(myCat); // ['age', 'furColor', 'meow', 'sleep']
```

📚 Oggetti built-in (già presenti in JavaScript)

- **Array** – `Array.isArray()`
- **Number** – `Number()`, `parseInt()`, `parseFloat()`
- **Date** – `Date.now()`, `Date.parse()`
- **Math** – contiene molte funzioni matematiche

📖 Approfondimenti:

- MDN - Global Objects

🔄 Copia per valore vs per riferimento

Primitivi (copiati per valore):

```
let a = 5; let b = a; b = 10; console.log(a); // 5
```

Oggetti (copiati per riferimento):

```
let obj1 = { name: 'John' }; let obj2 = obj1; obj2.name = 'Jane';
console.log(obj1.name); // 'Jane'
```

📩 Passaggio di parametri (valore vs riferimento)

```
function updateValues(num, obj, arr) { num = 10; obj.name = 'Jane'; arr.push(4);
}
```

- `num`: primitivo → non cambia fuori dalla funzione
- `obj` e `arr`: oggetti/array → modificati **per riferimento**

🔧 Invocare metodi con notazione tra parentesi

```
car['startEngine']('gasoline');
```

È equivalente a:

```
car.startEngine('gasoline');
```

❄️ Oggetti immutabili

Anche se un oggetto è dichiarato con `const`, le sue proprietà possono essere modificate:

```
const myCat = { name: 'Cleo' }; myCat.name = 'Adam'; // OK
```

Per impedire modifiche:

```
Object.freeze(myCat); myCat.name = 'Sam'; // NON CAMBIA
```