Object Literal

key-value pairs...

Key-Value Pairs

Name	Sensebounce+ SUMMER.RDY Shoes
Manufacturer	Adidas
Gender	Men
Sport	Running
Weight (gram)	366

Key-Value Pairs in Code (1)


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};
```
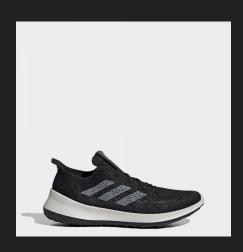
Key-Value Pairs in Code (2)


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};
```
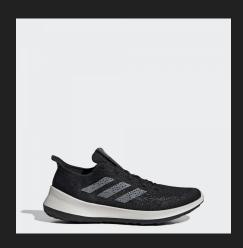
Diapit dengan { }

Key-Value Pairs in Code (3)


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};
```

Masing-masing key-value pair dipisahkan dengan ,

Key-Value Pairs in Code (4)


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};
```

Key ditulis menggunakan camel case

Key-Value Pairs in Code (5)


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};
```

Isi dari value bebas

Array Inside Object Literal


```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366,
  colors: ['Black', 'White']
};
```

Object Literal Inside Object Literal


```javascript
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY
Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366,
  colors: ['Black', 'White'],
  warehouseStock: {
    warehouse1: 0,
    warehouse2: 10,
    warehouse3: 2
  }
};
```

Accessing Object Literal (1)

```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};

console.log(sensebounce.name);

console.log(sensebounce['manufacturer']);

let toAccess = 'gender';
console.log(sensebounce[toAccess]);
```

Menggunakan .

Accessing Object Literal (2)

```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366
};

console.log(sensebounce.name);

console.log(sensebounce['manufacturer']);

let toAccess = 'gender';
console.log(sensebounce[toAccess]);
```

Menggunakan []

Adding New Key-Value Pairs (1)

```javascript
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366,
  colors: ['Black', 'White'],
  warehouseStock: {
    warehouse1: 0,
    warehouse2: 10,
    warehouse3: 2
  }
};

sensebounce.rating = 10;

let toAdd = 'price';
sensebounce[toAdd] = '100 USD';
```

Menggunakan .

Adding New Key-Value Pairs (2)

```
let sensebounce = {
  name: 'Sensebounce+ SUMMER.RDY Shoes',
  manufacturer: 'Adidas',
  gender: 'Men',
  sport: 'Running',
  weightGram: 366,
  colors: ['Black', 'White'],
  warehouseStock: {
    warehouse1: 0,
    warehouse2: 10,
    warehouse3: 2
  }
};

sensebounce.rating = 10;

let toAdd = 'price';
sensebounce[toAdd] = '100 USD';
```

Menggunakan []

Iteration for Object Literal (1)

```
for (const key in sensebounce) {
  console.log(`key: ${key}, value: ${sensebounce[key]}`);
}

const keys = Object.keys(sensebounce);
for (let i = 0; i < keys.length; i++) {
  console.log(`key: ${keys[i]}, value: ${sensebounce[keys[i]]}`);
}
```

Menggunakan `for-in`

Iteration for Object Literal (2)

```
for (const key in sensebounce) {
  console.log(`key: ${key}, value: ${sensebounce[key]}`);
}

const keys = Object.keys(sensebounce);
for (let i = 0; i < keys.length; i++) {
  console.log(`key: ${keys[i]}, value: ${sensebounce[keys[i]]}`);
}
```

Memanfaatkan `Object.keys()`

Deleting a Key-Value Pair (1)

```
delete sensebounce.rating;

let toDelete = 'price';
delete sensebounce[toDelete];
```

Menggunakan .

Deleting a Key-Value Pair (2)

```
delete sensebounce.rating;

let toDelete = 'price';
delete sensebounce[toDelete];
```

Menggunakan [ ]

Pass by Reference

```
let a = { key1: 1, key2: 2, key3: 3 };
let b = a;

a.key1 = 10;
console.log(a, b);
```

Output:
{ key1: 10, key2: 2, key3: 3 } { key1: 10, key2: 2, key3: 3 }
}

Copying an Object Literal

```
let a = { key1: 1, key2: 2, key3: 3 };
let b = {};

for (const key in a) {
  b[key] = a[key];
}

a.key1 = 10;
console.log(a, b);
```

Output:
{ key1: 10, key2: 2, key3: 3 } { key1: 1, key2: 2, key3: 3 }

Copying an Object Literal (not for LC)

```
let a = { key1: 1, key2: 2, key3: 3 };
let b = {};

Object.assign(b, a);

a.key1 = 10;
console.log(a, b);
```

Output:
{ key1: 10, key2: 2, key3: 3 } { key1: 1, key2: 2, key3: 3 }

Copying an Object Literal (not for LC)

```
let a = { key1: 1, key2: 2, key3: 3 };
let b = { ...a };

a.key1 = 10;
console.log(a, b);
```

Output:
{ key1: 10, key2: 2, key3: 3 } { key1: 1, key2: 2, key3: 3 }

Try This

```
let a = { key1: 1, key2: 2, key3: 3 };
let b = [1, 2, 3];

console.log(typeof(a), typeof(b));
```

Output:
object object