JS - Data Types

Primitive DT

number, string, boolean, undefined, null, bigint, symbol

Non- Primitive

Object → JS Array or JS Object

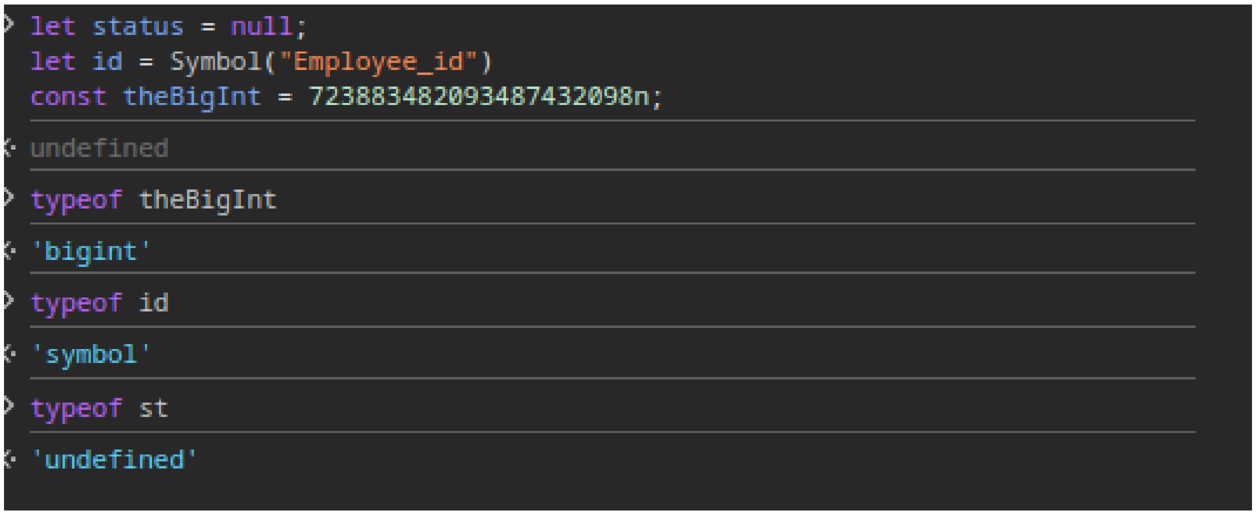
We can find the datatype uisng typeof( )

```
var brand = "Abhay"; //string
var size = 'S'; //string
var price = 327; //number
var rating = 7.8; //number
var availability = true; //boolean
var discount; //undefined

console.log(typeof brand) => string
console.log(typeof size) => string
console.log(typeof price) => number
console.log(typeof rating)
console.log(typeof availability)
console.log(typeof discount)
```

```
let status = null;
let id = Symbol("Employee_id");
```

```
const theBigInt = 7238834820934874320989n;
```



```
let status = null;
let id = Symbol("Employee_id")
const theBigInt = 7238834820934874320989n;
undefined
typeof theBigInt
'bigint'
typeof id
'symbol'
typeof st
'undefined'
```

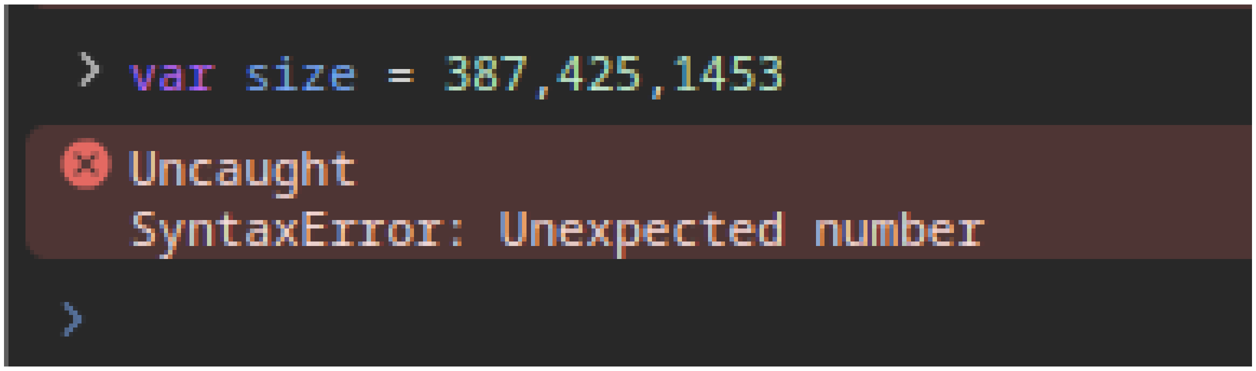
Non-primitive type

```
//js arrays and js object both are object data types
var size = [2,42,532.53]
var specification = {
                                            "fit" : "regular",
                                            "size" : 34
                                        }
var emp = {eid : 011, ename: "rg"}
var name = ["rg", "dsv" , "nm"];

console.log(typeof size)
console.log(typeof specification)
console.log(typeof emp)
console.log(typeof name)

// output
object
```

```
object
object
string
```



```
> var size = 387,425,1453
Uncaught
  SyntaxError: Unexpected number
>
```

JS - Array

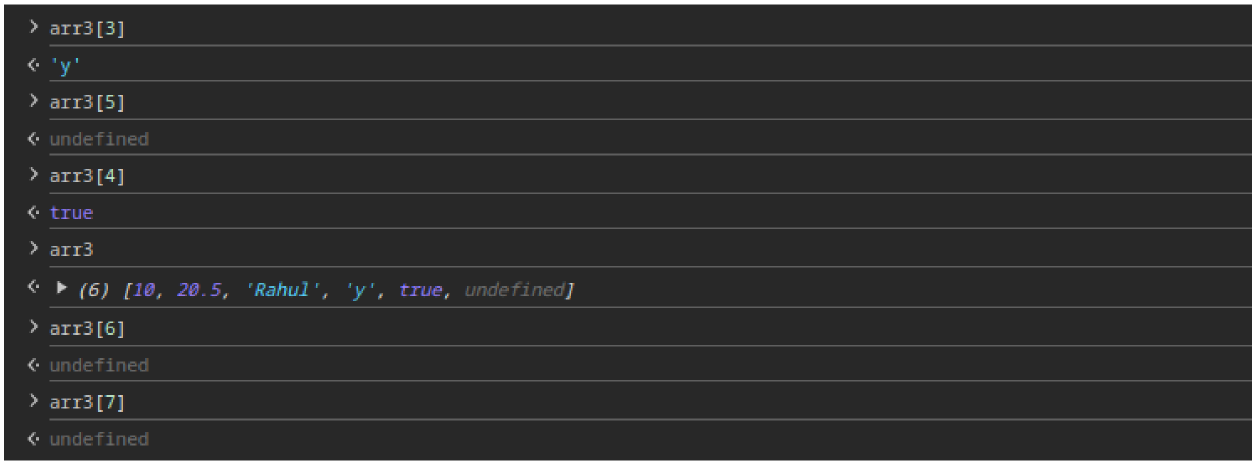
- Collection / Group of values or elements as one entity
- Heterogeneous and duplicate values are allowed
- Elements are stored based on indexing
- Negative indexing is not possible
- It is iterable object → for, while, do - while, for (of)

Create

```
let arr = []
let arr2 = [10, 20, 30]
let arr3 = [10, 20.5, "Rahul", 'y', true, undefined]
```

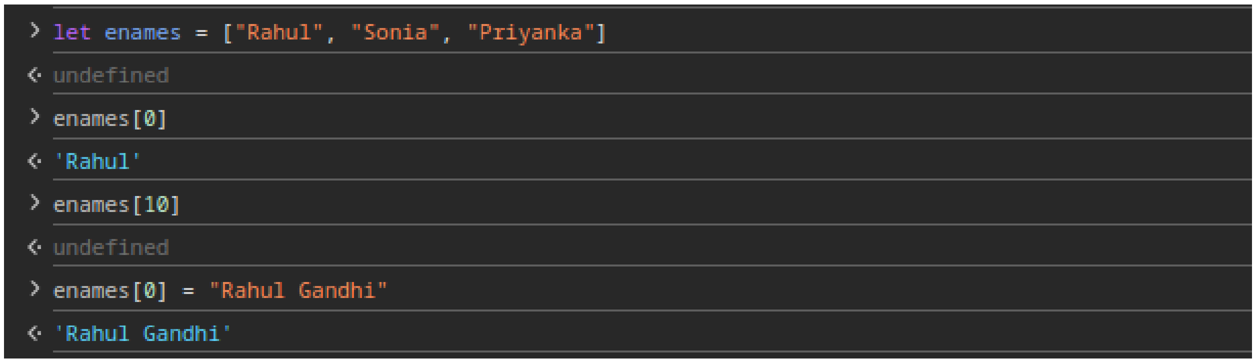
Read

```
console.log(arr3[0]) // 10
console.log(arr3[1]) // 20.5
console.log(arr3[2]) // "Rahul"
console.log(arr3[3]) // 'y'
console.log(arr3[4]) // true
```

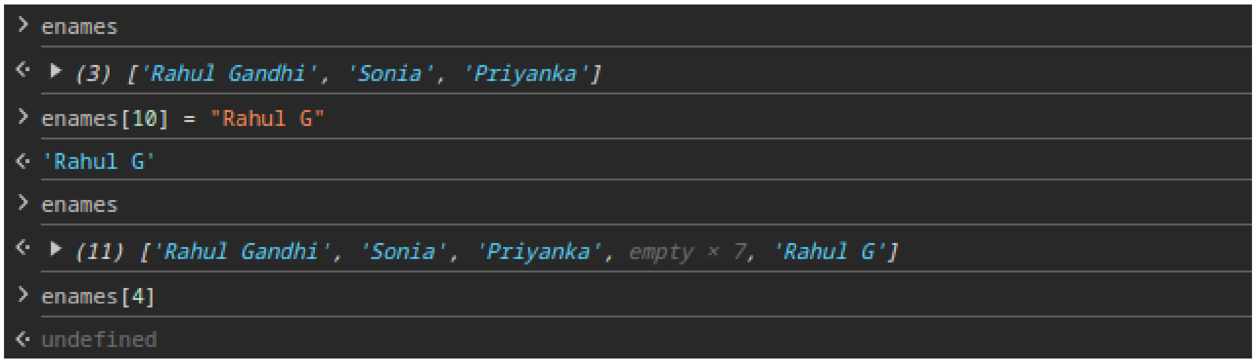


```
> arr3[3]
< 'y'
> arr3[5]
< undefined
> arr3[4]
< true
> arr3
< ▶ (6) [10, 20.5, 'Rahul', 'y', true, undefined]
> arr3[6]
< undefined
> arr3[7]
< undefined
```

Create / Read / Update



```
> let enames = ["Rahul", "Sonia", "Priyanka"]
< undefined
> enames[0]
< 'Rahul'
> enames[10]
< undefined
> enames[0] = "Rahul Gandhi"
< 'Rahul Gandhi'
```

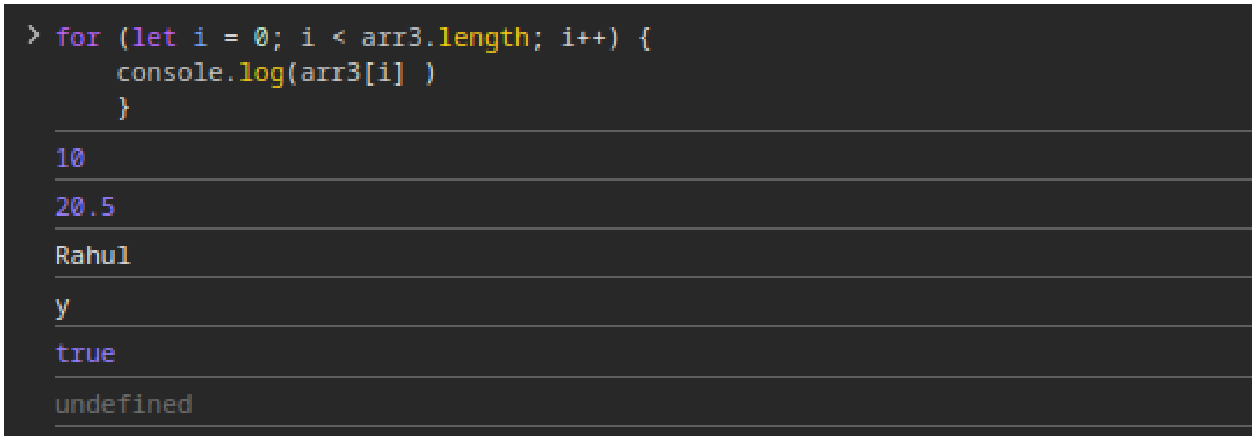


```
> enames
<· ▶ (3) ['Rahul Gandhi', 'Sonia', 'Priyanka']
> enames[10] = "Rahul G"
<· 'Rahul G'
> enames
<· ▶ (11) ['Rahul Gandhi', 'Sonia', 'Priyanka', empty × 7, 'Rahul G']
> enames[4]
<· undefined
```

Delete

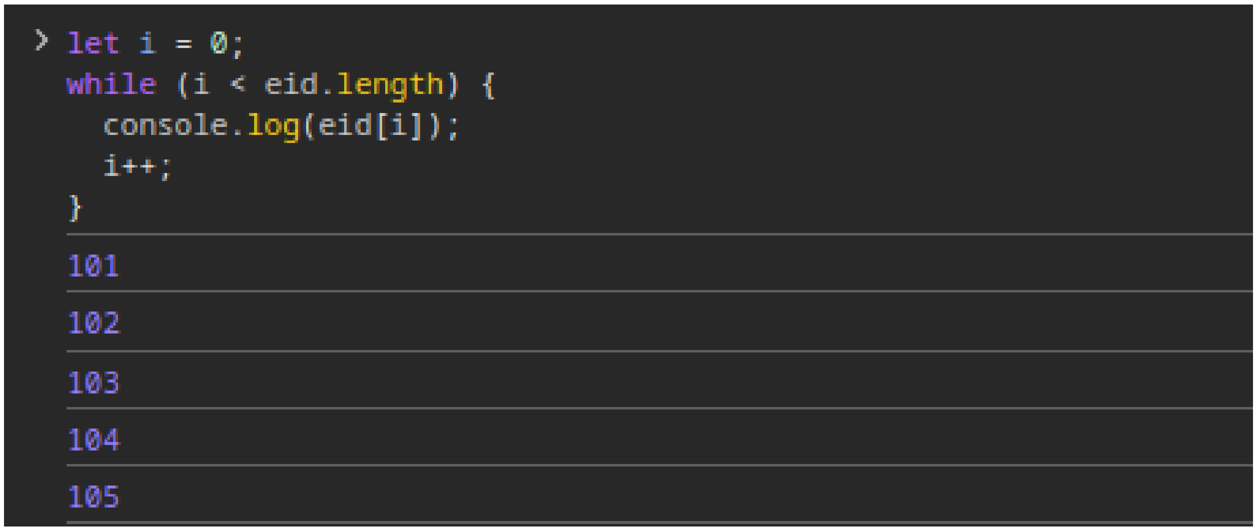
Iteration over JS - Array

For



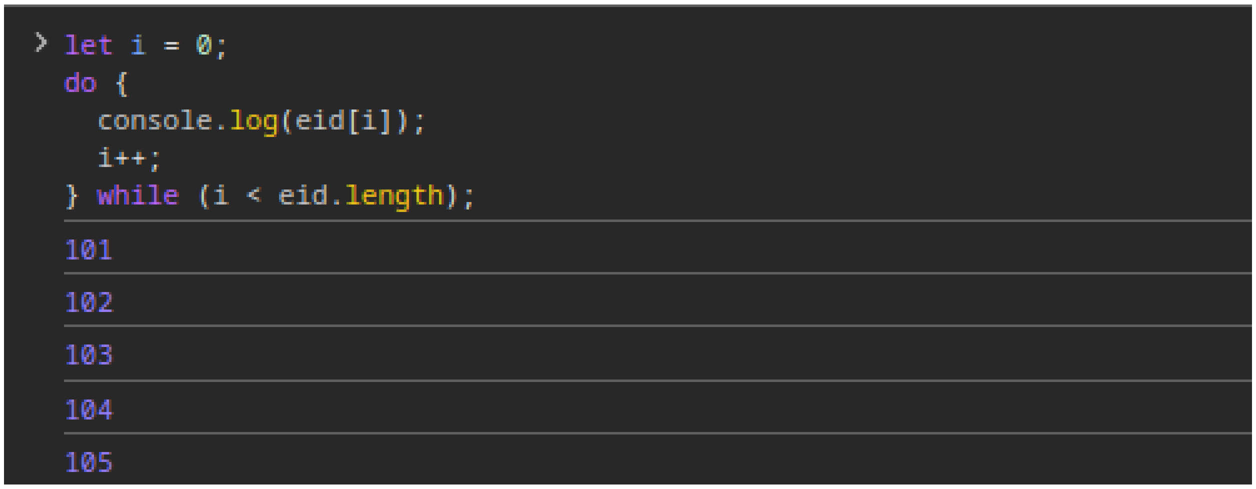
```
> for (let i = 0; i < arr3.length; i++) {
      console.log(arr3[i] )
      }
  10
  20.5
  Rahul
  y
  true
  undefined
```

While



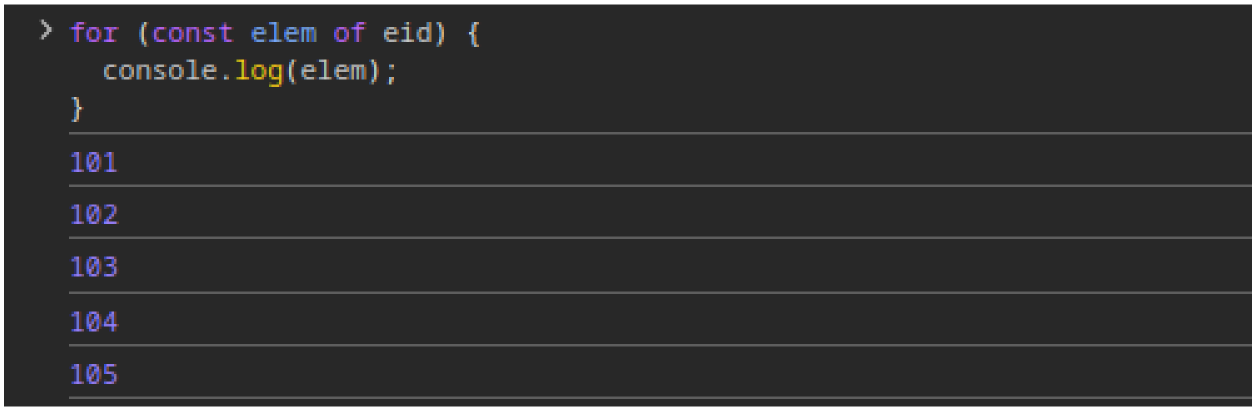
```
> let i = 0;
  while (i < eid.length) {
    console.log(eid[i]);
    i++;
  }
  101
  102
  103
  104
  105
```

do-while



```
> let i = 0;
  do {
    console.log(eid[i]);
    i++;
  } while (i < eid.length);
  101
  102
  103
  104
  105
```

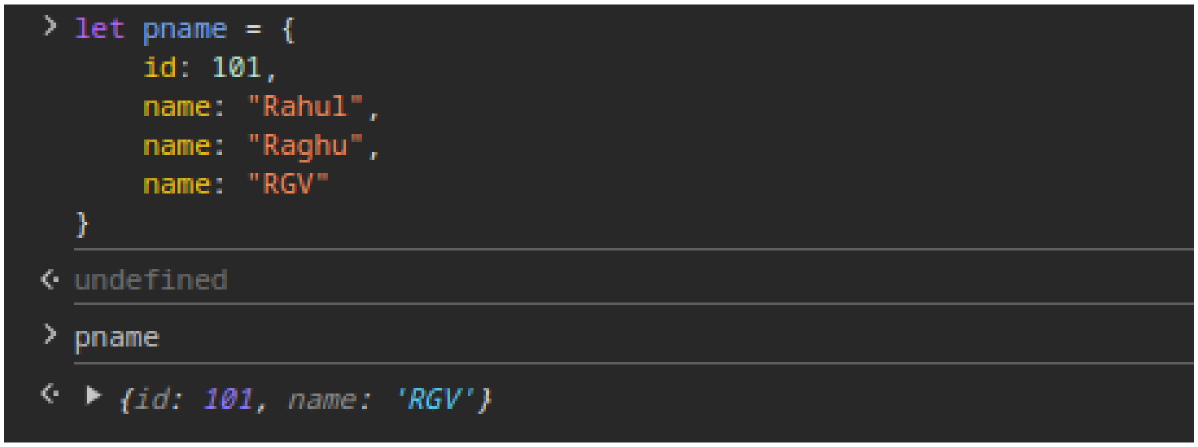
For-of



```
> for (const elem of eid) {
    console.log(elem);
  }
  101
  102
  103
  104
  105
```

JS - Objects

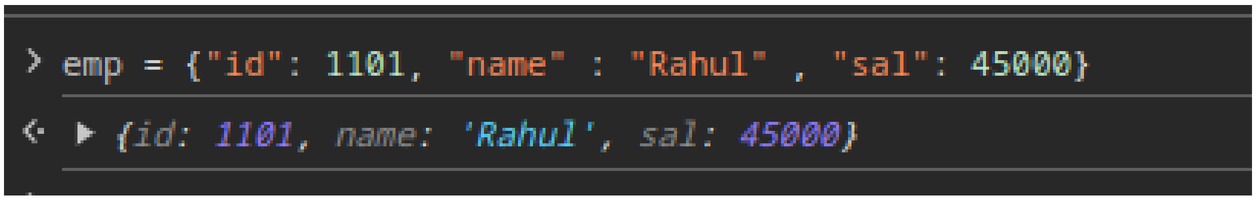
- Group of key-value pairs as one entity
- Group of properties (property name - values) as one entity
- Duplicate keys are not allowed



```
> let pname = {
        id: 101,
        name: "Rahul",
        name: "Raghu",
        name: "RGV"
  }
<· undefined
> pname
<· ▶ {id: 101, name: 'RGV'}
```

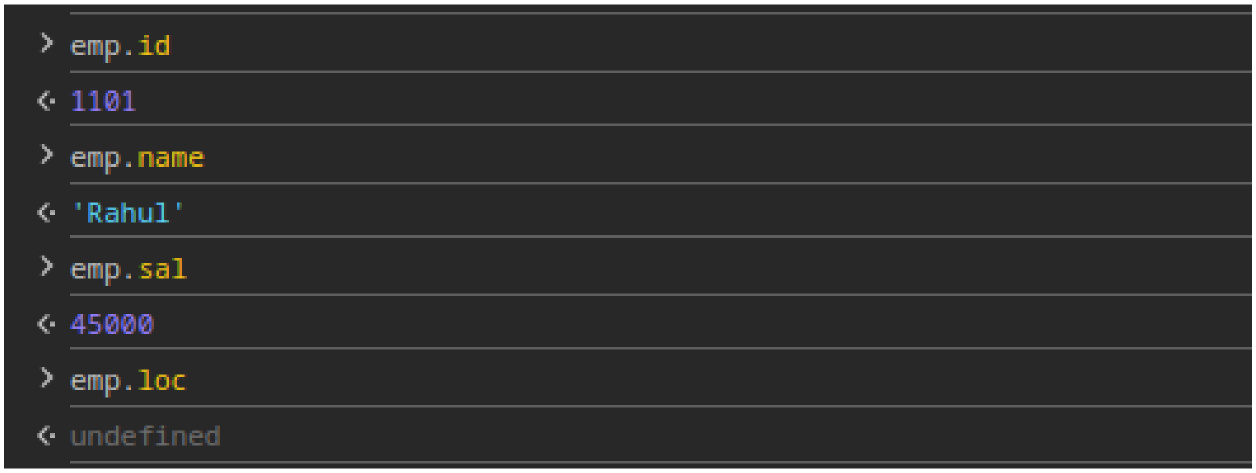
- Indexing concept is not allowed
- Objects are not iterable

Create



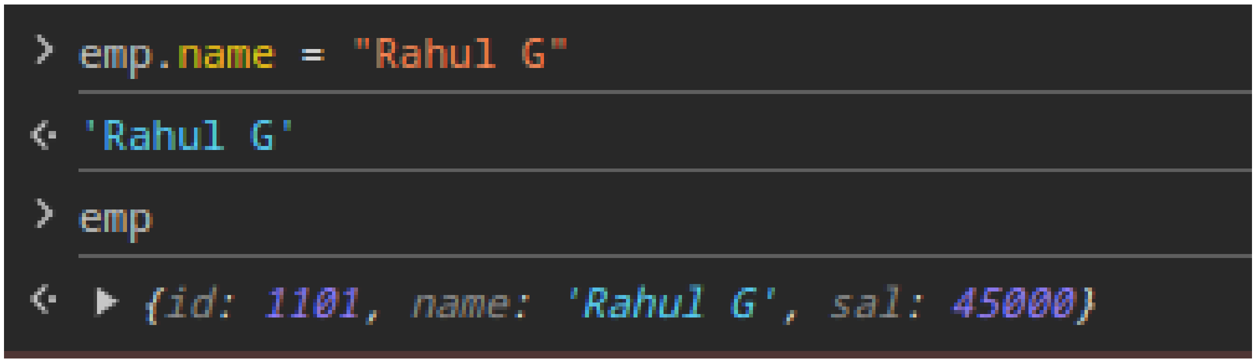
```
> emp = {"id": 1101, "name" : "Rahul" , "sal": 45000}
<· ▶ {id: 1101, name: 'Rahul', sal: 45000}
```

Read



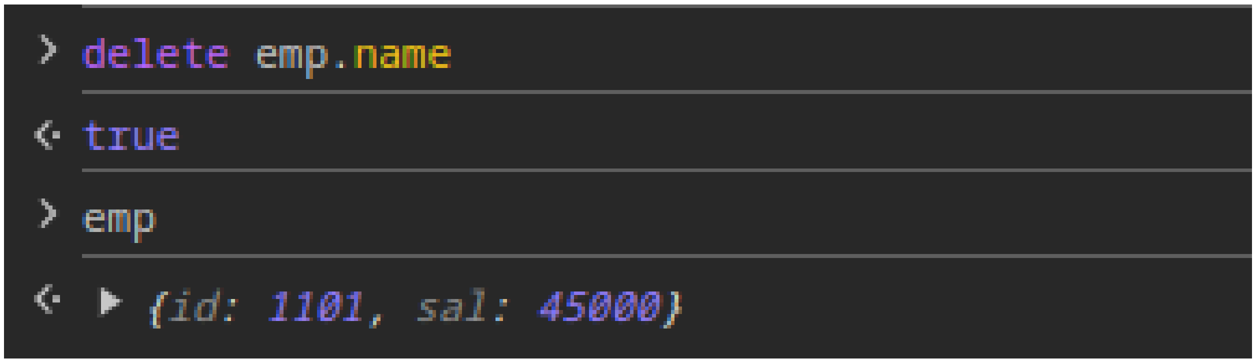
```
> emp.id
<· 1101
> emp.name
<· 'Rahul'
> emp.sal
<· 45000
> emp.loc
<· undefined
```

Update



```
> emp.name = "Rahul G"
<· 'Rahul G'
> emp
<· ▶ {id: 1101, name: 'Rahul G', sal: 45000}
```

Delete



```
> delete emp.name
<· true
> emp
<· ▶ {id: 1101, sal: 45000}
```