

# Introduction to Objects

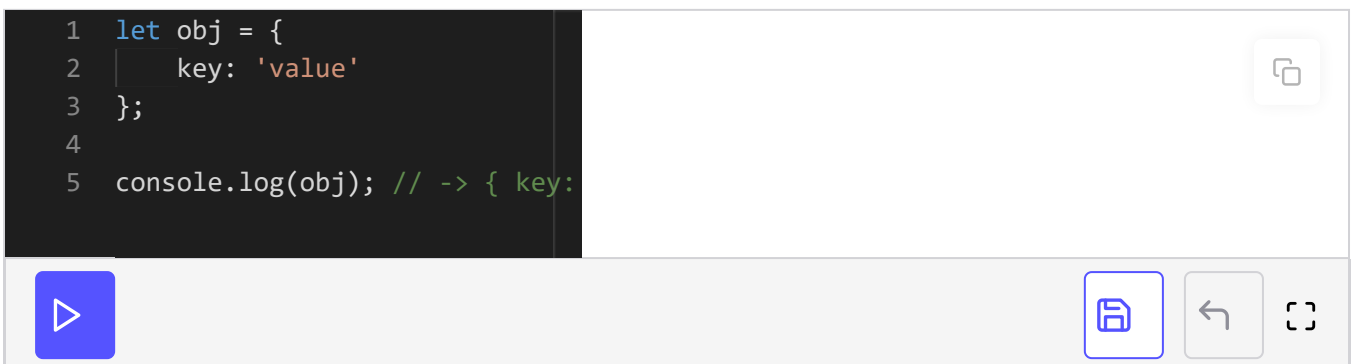
Learn the last data type we'll cover: objects. Objects are like arrays in that they group a set of information. They're more loose, however, and we interact with them differently. They're made up of key-value pairs.

Objects are another way to store a collection of values. While arrays keep everything ordered, objects provide a little more flexibility. They're also a little harder to get the hang of.

## Key-Value

An object is created with curly brackets `{ }`. They're made up of *key-value* pairs, also referred to as *properties*. Let's start with an example.

```
1 let obj = {
2   key: 'value'
3 };
4
5 console.log(obj); // -> { key:
```

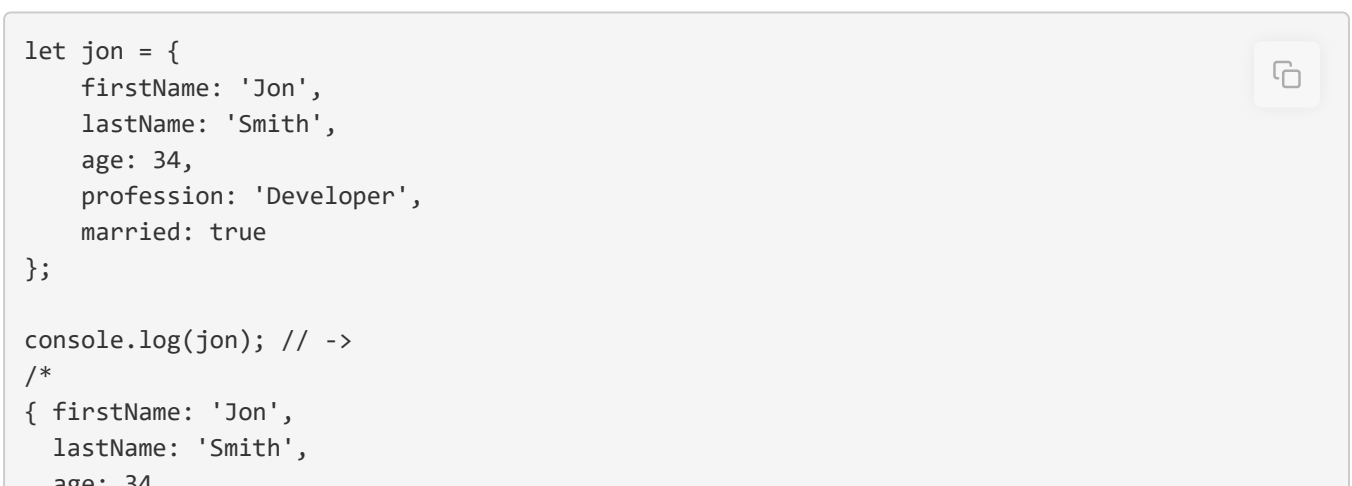


We've created an object with one item inside it, one key-value pair. The key is named `'key'` and the value is the string `'value'`.

Let's show a more complex object.

```
let jon = {
  firstName: 'Jon',
  lastName: 'Smith',
  age: 34,
  profession: 'Developer',
  married: true
};

console.log(jon); // ->
/*
{ firstName: 'Jon',
  lastName: 'Smith',
  age: 34,
```



```
    age: 34,  
    profession: 'Developer',  
    married: true }  
*/
```



As we can see, an object is a good structure to store related pieces of information. We have all the information about Jon inside one object.

Each property we write in an object must be separated by commas.

## Working with Properties

### Bracket Notation `[]`

To access an object's property, we use brackets, like we would for an array.

```
let jon = {  
  firstName: 'Jon',  
  lastName: 'Smith',  
  age: 34,  
  profession: 'Developer',  
  married: true  
};  
  
console.log(jon['firstName']); // -> Jon  
console.log(jon['age']); // -> 34
```



We can pass a variable into the brackets to access a property as well.

```
let jon = {  
  firstName: 'Jon',  
  lastName: 'Smith',  
  age: 34,  
  profession: 'Developer',  
  married: true  
};  
  
let propName = 'firstName';  
console.log(jon[propName]); // -> Jon
```



### Dot Notation

If the property names are valid JavaScript variable names, we can use dot notation instead. It saves us a few characters. All properties on `jon` are valid JavaScript variable names so we can access them completely with dot notation.

Note that we can't use variables this way as we did for bracket notation above. Using dot notation, we can only directly access a property using the property name itself.

```
let jon = {
  firstName: 'Jon',
  lastName: 'Smith',
  age: 34,
  profession: 'Developer',
  married: true
};

console.log(jon.firstName); // -> Jon
console.log(jon.age); // -> 34
```

## Adding Properties

We can add properties using bracket and dot notation as well.

```
let jon = {
  firstName: 'Jon',
  lastName: 'Smith',
  age: 34,
  profession: 'Developer',
  married: true
};

jon.fullName = jon.firstName + jon.lastName;
jon['hasChildren'] = true;

console.log(jon); // ->
/*
{ firstName: 'Jon',
  lastName: 'Smith',
  age: 34,
  profession: 'Developer',
  married: true,
  fullName: 'JonSmith',
  hasChildren: true }
*/
```



# Complex Objects

Objects can have arrays and other objects as properties. To access nested arrays or objects, we *chain* dot and bracket notation.

```
let jon = {
  firstName: 'Jon',
  lastName: 'Smith',
  otherInformation: {
    age: 34,
    profession: 'Developer',
    married: true
  },
  children: ['Anna', 'Jake']
};

console.log(jon.otherInformation.profession); // -> Developer
console.log(jon.children[0]); // -> Anna
```



## undefined

If we attempt to look up a property that doesn't exist, we'll get **undefined**. The property isn't there, so JavaScript is telling us that we're looking for something that's probably missing.

```
let obj = {};
console.log(obj.xyz); // -> undefined
```



# Quiz & Code Challenges

Feel free to test your understanding.

1

What will the following code print?

```
let obj = {};
```

```
let obj = {};  
obj['abcd'] = 'Hello';  
console.log(obj.abcd);
```

- ☐ A) abcd
- ☐ B) Hello
- ☐ C) undefined
- ☐ D) error

2

What will the following code print?

```
let obj = {};  
obj[abcd] = 'Hello';  
console.log(obj.abcd);
```

- ☐ A) abcd
- ☐ B) Hello
- ☐ C) undefined
- ☐ D) error

3

What will the following code print?

```
let obj = { prop: 19 };  
obj.prop = {};  
console.log(obj.prop);
```

- ☐ A) 19
- ☐ B) {}
- ☐ C) undefined
- ☐ D) error

4

What will the following code print?

```
let obj = { abc: 19 };  
obj.innerObj = {};  
console.log(obj.innerObj.abc);
```

- ☐ A) 19
- ☐ B) {}
- ☐ C) undefined
- ☐ D) error

5

What will the following code print?

```
let obj = { obj: {} };  
obj.obj.obj = 19;  
console.log(obj.obj);
```

- ☐ A) 19
- ☐ B) {}
- ☐ C) { obj: 19 }
- ☐ D) undefined
- ☐ E) error

[CHECK ANSWERS](#)

## INSTRUCTIONS

Edit this function such that it takes in an object as a parameter. Return the `'abc'` property from that object.

```
function getABC() {}
```



## INSTRUCTIONS

Edit this function such that it takes in an object as a parameter.

Add a property to the object with a key `'abc'` and a value of `true`.

Return the object.

```
function addABC() {}
```

