Objects as Maps

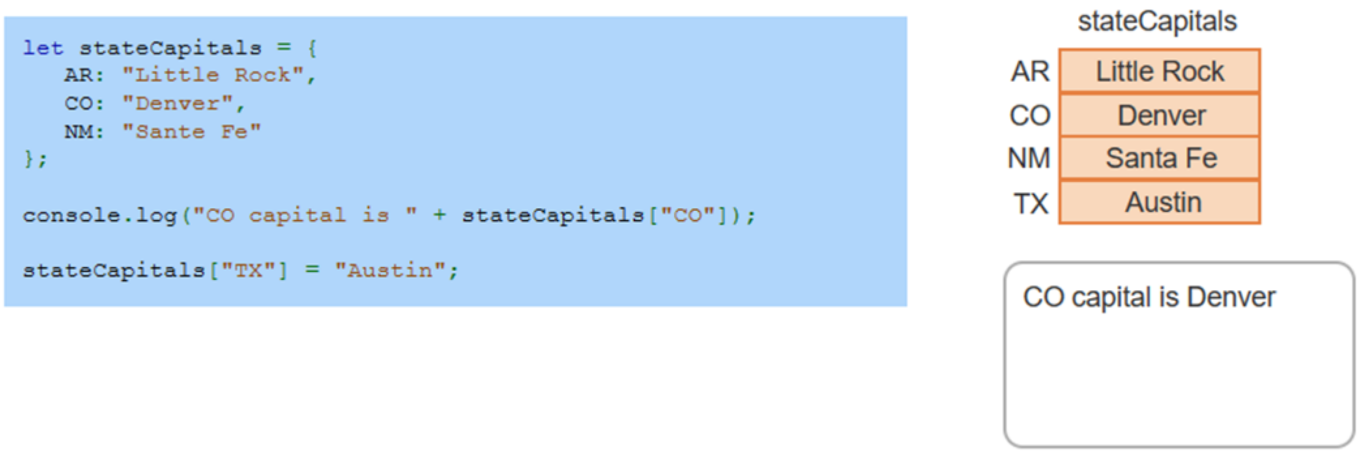
Maps

Objects as maps

A ***map*** or ***associative array*** is a data structure that maps keys to values. Each key/value pair in a map is called an ***element***.

JavaScript objects can be used as maps, in which the key is the object property and the value is the property's value. When an object is used as a map, individual elements are accessed by key using brackets. Ex: myMap["key"].

State capitals in an object map



```
let stateCapitals = {
   AR: "Little Rock",
   CO: "Denver",
   NM: "Sante Fe"
};

console.log("CO capital is " + stateCapitals["CO"]);

stateCapitals["TX"] = "Austin";
```

```
CO capital is Denver
```

1. An object map called stateCapitals is initialized with three key/value pairs, creating three elements.
2. The map's value for key "CO" is "Denver".
3. The capital of Texas, with key "TX" and value "Austin", is added to the map.

For-in loop

The ***for-in loop*** iterates over an object's properties in arbitrary order and is ideal for looping through an object map's elements. The for-in loop declares a variable on the left side of the in keyword and an object on the right. In each iteration, the variable is assigned with each of the object's properties.

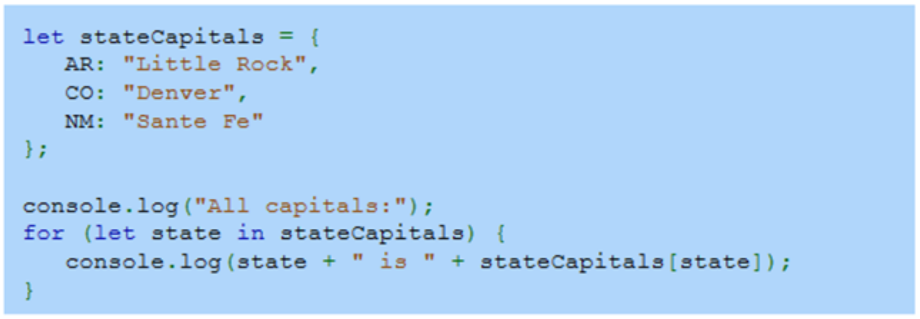
Construct 6.10.1: for-in loop.

for (let variable in object) {

  body

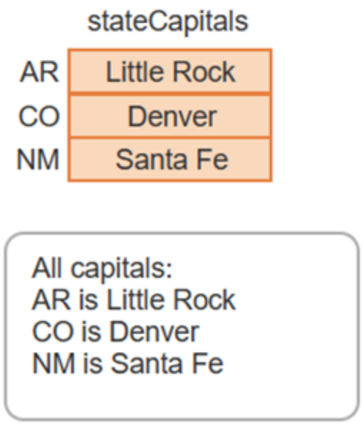
}

Looping through an object map.



```
let stateCapitals = {
   AR: "Little Rock",
   CO: "Denver",
   NM: "Sante Fe"
};

console.log("All capitals:");
for (let state in stateCapitals) {
   console.log(state + " is " + stateCapitals[state]);
}
```



```
All capitals:
AR is Little Rock
CO is Denver
NM is Santa Fe
```

Explanation :

1. An object map called stateCapitals is initialized with three key/value pairs, creating three elements.
2. The for-in loop declares variable state inside the for-in statement.
3. The for-in loop assigns each key to the state variable, one at a time. The loop body outputs each element.

Other object map operations

Other common operations performed on object maps include:

- **Get number of elements.**The ***keys()*** method returns an array of an object's property names. The array's length property returns the number of elements in the object map.
- **Check for key.**The ***in operator*** returns true if an object contains the given property and returns false
- **Remove element.**The ***delete operator*** removes a key/property from a map or object.

Object map operations example.

let stateCapitals = {

  AR: "Little Rock",

  CO: "Denver",

  NM: "Sante Fe"

```
};


let states = Object.keys(stateCapitals);
console.log(states);        // AR,CO,NM
console.log(states.length);   // 3


// Evaluates true
if ("NM" in stateCapitals) {
   console.log("NM exists");
}


// Remove the NM/Santa Fe pair
delete stateCapitals["NM"];


// Evaluates false
if ("NM" in stateCapitals) {
   console.log("NM exists");
}


// Outputs undefined
console.log(stateCapitals["NM"]);
```

Map object

The ***Map object*** is a newer alternative to using objects for storing key/value pairs. Common methods and properties of the Map object include:

- The ***set(key, value)***method sets a key/value pair. If the key is new, a new element is added to the map. If the key already exists, the new value replaces the existing value.
- The ***get(key)***method gets a key's associated value.

- The ***has(key)***method returns true if a map contains a key, false otherwise.
- The ***delete(key)***method removes a map element.
- The ***size***property is the number of elements in the map.

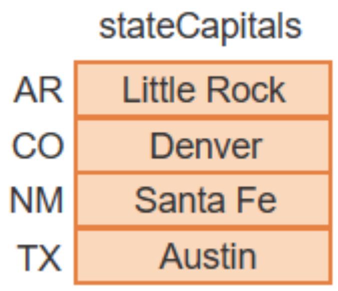
The for-of loop, which is often used to loop through an array, is ideal for looping through a Map. Each of the map's key/value pairs are assigned to the [key, value] variables declared in the for-of loop, as illustrated in the animation below.

State capitals in a Map.

```
let stateCapitals = {
   AR: "Little Rock",
   CO: "Denver",
   NM: "Sante Fe"
};

console.log("CO capital is " + stateCapitals["CO"]);

stateCapitals["TX"] = "Austin";
```



CO Capital is Denver

Explanation :

1. new Map object is created with the Map() constructor.
2. The set() method adds three key/value pairs to the stateCapitals Map.
3. The size property returns 3 because stateCapitals has three key/value pairs.
4. The has() method returns true because "CO" is one of the keys in stateCapitals. The get() method returns the value associated with "CO".
5. The for-of loop assigns each key/value pair to variables state and capital.