1. Declaring an Empty Array

In JavaScript, you can declare an array, which is essentially a collection of data (can be of any type, like numbers, strings, objects, etc.). An empty array can be declared like this:

```
var pets = [];
```

Here:

- `var` is a keyword used to declare a variable.
- `pets` is the name of the variable.
- `[]` is the syntax for an empty array. The square brackets `[]` represent an array literal.

At this point, the `pets` array is empty, and it doesn't have any elements yet. It's just a placeholder to store future values.

2. Assigning Values to an Array

You can assign values to specific positions (or indices) in the array after it has been declared. Arrays in JavaScript are zero-indexed, meaning the first element is at index `0`, the second element at index `1`, and so on.

Here’s how you can assign values to specific indices:

```
pets[0] = "dog";  // Assigns "dog" to the first position (index 0)
pets[1] = "cat";  // Assigns "cat" to the second position (index 1)
pets[2] = "bird"; // Assigns "bird" to the third position (index 2)
```

Now, the `pets` array looks like this:

```
["dog", "cat", "bird"]
```

- `pets[0]` is `"dog"`
- `pets[1]` is `"cat"`
- `pets[2]` is `"bird"`

3. Leaving Gaps in an Array

JavaScript allows you to skip over indices in an array, and this creates "gaps." If you don't assign a value to a certain index, JavaScript will leave it as `undefined`.

For example, you can assign values to indices 3 and 6 in the array, leaving the indices in between (i.e., 4 and 5) undefined:

```
pets[3] = "lizard";  // Assigns "lizard" to the 4th position (index 3)
pets[6] = "snake";   // Assigns "snake" to the 7th position (index 6)
```

Now, the pets array looks like this:

```
["dog", "cat", "bird", "lizard", <1 empty slot>, <1 empty slot>, "snake"]
```

Here:

- pets[3] is "lizard"
- pets[6] is "snake"
- pets[4] and pets[5] are undefined because no values were assigned to those indices.

Although the array technically has 7 elements, it contains gaps (undefined values) in the positions where values were not assigned.

4. Adding More Values to an Array

You can continue adding new elements to an array after it has already been initialized and populated with some values. This will increase the size of the array.

For example, continuing from the previous example where we had ["dog", "cat", "bird", "lizard", undefined, undefined, "snake"]:

```
pets[4] = "fish";    // Adds "fish" at index 4
pets[5] = "gerbil";  // Adds "gerbil" at index 5
```

Now, the array looks like this:

```
["dog", "cat", "bird", "lizard", "fish", "gerbil", "snake"]
```

Notice how we "filled in" the previous gaps by assigning values to indices 4 and 5. Now the array has 7 elements, each containing a value, and there are no undefined values.

5. Using pop() to Remove the Last Element of an Array

The pop() method is used to remove the last element from an array. This method modifies the array in place and returns the value that was removed.

For example, if we have the following array:

```
let pets = ["dog", "cat", "bird", "lizard", "fish", "gerbil", "snake"];
```

If we call `pop()` on this array:

```
pets.pop(); // Removes the last element, "snake"
```

Now, the `pets` array looks like this:

```
["dog", "cat", "bird", "lizard", "fish", "gerbil"]
```

As you can see, `"snake"` was removed from the array, and the size of the array decreased by 1.

6. Using `push()` to Add One or More Elements to the End of an Array

The `push()` method is used to add one or more elements to the end of an array. It also modifies the array in place and returns the new length of the array.

For example:

```
let pets = ["dog", "cat", "bird"];
pets.push("fish", "ferret"); // Adds "fish" and "ferret" to the end of the array
```

After calling `push()`, the `pets` array will look like this:

```
["dog", "cat", "bird", "fish", "ferret"]
```

Here, we added two new elements, `"fish"` and `"ferret"`, to the end of the array. The size of the array is now 5.

Summary of Array Methods and Concepts

- **Declaring an array**: You can declare an empty array using `[]` or initialize it with values, e.g., `var pets = ["dog", "cat"]`.
- **Assigning values**: You can assign values to specific indices like `pets[0] = "dog"`.
- **Leaving gaps**: It's legal to leave gaps in an array by assigning values to non-contiguous indices, but this may result in `undefined` values in those positions.
- **`pop()`**: This method removes the last element of the array and returns it.
- **`push()`**: This method adds one or more elements to the end of the array.