The For In Loop

The JavaScript `for in` statement loops through the properties of an Object:

Syntax

```
for (key in object) {
  // code block to be executed
}
```

Example

```
const person = {fname:"John", lname:"Doe", age:25};

let text = "";
for (let x in person) {
  text += person[x];
}
```

Example Explained

- The **for in** loop iterates over a **person** object
- Each iteration returns a **key** (x)
- The key is used to access the **value** of the key
- The value of the key is **person[x]**

For In Over Arrays

The JavaScript `for in` statement can also loop over the properties of an Array:

Syntax

```
for (variable in array) {
  code
}
```

Example

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
for (let x in numbers) {
  txt += numbers[x];
}
```

Note:

- Do not use **for in** over an Array if the index **order** is important.
- The index order is implementation-dependent, and array values may not be accessed in the order you expect.
- It is better to use a **for** loop, a **for of** loop, or **Array.forEach()** when the order is important.

Array.forEach()

The `forEach()` method calls a function (a callback function) once for each array element.

Example

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
numbers.forEach(myFunction);

function myFunction(value, index, array) {
  txt += value;
}
```

Note that the function takes 3 arguments:

- The item value
- The item index
- The array itself

The example above uses only the value parameter. It can be rewritten to:

Example

```
const numbers = [45, 4, 9, 16, 25];

let txt = "";
numbers.forEach(myFunction);

function myFunction(value) {
  txt += value;
}
```