S2 Z6

Advanced Topics

- Proxy
- Iterators
- Generator

Proxy

```
const target = {};
const handler = {};
const proxy = new Proxy(target, handler); // proxy
console.log(proxy);
```

Proxy
[[Handler]]:Object
[[IsRevoked]]:false
[[Target]]:Object

Proxy

```
const target = {}; // target
const handler = {};
const proxy = new Proxy(target, handler); // proxy
target.prop1 = "prop1";
proxy.prop2 = "prop2";
console.log(proxy); // proxy and target
console.log(target);
```

Proxy {prop1: 'prop1', prop2: 'prop2'}
{prop1: 'prop1', prop2: 'prop2'}

Proxy

```
let target = -{};
let handler = {
  get: () => {
    console.log("Object is being accessed");
  },
};
let proxy = new Proxy(target, handler); // proxy
target.prop1 = "prop1";
proxy.prop2 = "prop2";
console.log(target.prop1);
console.log(proxy.prop2); //??
console.log(proxy);
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
prop1
Object is being accessed
undefined
> Proxy {prop1: 'prop1', prop2: 'prop2'}
```

Proxy

```
let target = {}; // target
let handler = {
  get: (obj, prop) => {
    console.log("Object is being accessed");
    return obj[prop];
  },
};
let proxy = new Proxy(target, handler); // proxy
target.prop1 = "prop1";
proxy.prop2 = "prop2";
console.log(target.prop1);
console.log(proxy.prop2);
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
prop1
Object is being accessed
prop2
```

Proxy

```
let test = { prop1: "pro1 Value", hidden: "secret :)" };
test = new Proxy(test, {
  get: (target, property, receiver) => {
    if (!target[property])
      throw new Error(`${property} not found on an object`);
    if (property === "hidden") return "Hidden property";
    console.log("Accessing:", property);
    return target[property];
  },
});
console.log(test.hidden);
console.log(test.hammer);
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
Hidden property
> Uncaught Error: hammer not found on an object
```

Proxy

```
"use strict";

let test = { prop1: "prop1 Value" };
test = new Proxy(test, {
  set: (target, property, value, receiver) => {
    target[property] = value;
  },
});

console.log((test.prop1 = "new Value"));
console.log(test.prop1);
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
> Uncaught TypeError: 'set' on proxy: trap returned falsish for property 'prop1'
Process exited with code 1
```

Proxy

```
"use strict";

let test = { prop1: "prop1 Value" };
test = new Proxy(test, {
  set: (target, property, value, receiver) => {
    target[property] = value;
    return true;
  },
});

console.log((test.prop1 = "new Value"));
console.log(test.prop1);
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
new Value
new Value
```

Proxy

```
let test = { prop1: "prop1 Value", _const: "constVal" };
test = new Proxy(test, {
  set: (target, property, value, receiver) => {
    if (/^_/.test(property))
      throw new Error("Can't change internat property: " + property);

    target[property] = value;
    return true;
  },
});

console.log((test.prop1 = "new Value"));
console.log(test.prop1);
console.log((test._const = "changed Const"));
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
new Value
new Value
> Uncaught Error: Can't change internat property: _const
Process exited with code 1
```

Proxy

```
class BaseClass {
  constructor(id) {
    this.id = id;
    return new Proxy(this, {});
  }
}

class SpecialClass extends BaseClass {
  constructor(id, prop1, prop2) {
    super(id);
    this.prop1 = prop1;
    this.prop2 = prop2;
  }
}

console.log(new BaseClass("uniquID"));
console.log(new SpecialClass("uniquID", "PROP1", "PROP2"));
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
> Proxy {id: 'uniquID'}
> Proxy {id: 'uniquID', prop1: 'PROP1', prop2: 'PROP2'}
```

Proxy

```
let myLog = (prop1, prop2) => {
  console.log(prop1, prop2);
};

myLog = new Proxy(myLog, {
  apply: (fn, contex, args) => {
    console.log("FUNCTION: " + fn.toString());
    console.log("CONTEX: " + contex);
    console.log("ARGS: " + args.toString());
    fn(...args);
  },
});

myLog("P1", "P2");
```

```
C:\Program Files\nodejs\node.exe .\1_Proxy.js
FUNCTION: (prop1, prop2) => {
  console.log(prop1, prop2);
}
CONTEX: undefined
ARGS: P1,P2
P1 P2
```

Generators

```
function* timestampGenerator() {
  console.log(Date.now());
}

const iterator = timestampGenerator();
iterator.next();
```

```
C:\Program Files\nodejs\node.exe .\2_Generator.js
1621497117045
```

Generators

```
function* getRandomNumber() {
    while (true) {
        yield Math.floor(Math.random() * 100);
    }
}

const randomNumberIterator = getRandomNumber();
console.log(randomNumberIterator.next());
console.log(randomNumberIterator.next());
console.log(randomNumberIterator.next());
console.log(randomNumberIterator.next());
```

```
C:\Program Files\nodejs\node.exe .\2_Generator.js
> {value: 52, done: false}
> {value: 99, done: false}
> {value: 80, done: false}
> {value: 53, done: false}
```

Generators

```
function* timestampGenerator() {
  var ts = Date.now();
  console.log("first ts" + ts);
  yield ts;
  console.log("go go");
  var extraTime = yield;
  console.log("extra Time" + extraTime);
  if (extraTime) ts += extraTime;
  console.log("current ts" + ts);
}

const iterator = timestampGenerator();
const firstTS = iterator.next();
console.log(firstTS);
iterator.next();
iterator.next(1000);
```

```
C:\Program Files\nodejs\node.exe .\2_Generator.js
first ts1621499154732
> {value: 1621499154732, done: false}
go go
extra Time1000
current ts1621499155732
```

Generators

```
function* gen1() {
  yield 1;
  yield 2;
}

function* gen2() {
  yield* gen1();
  yield 3;
}

const iterator = gen2();
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
```

```
C:\Program Files\nodejs\node.exe .\2_Generator.js
> {value: 1, done: false}
> {value: 2, done: false}
> {value: 3, done: false}
```

Generators

```
function* gen1() {
  yield 1;
  yield 2;
  return 4;
}

function* gen2() {
  const val = yield* gen1();
  yield 3;
  yield val;
}

const iterator = gen2();
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
console.log(iterator.next());
```

```
C:\Program Files\nodejs\node.exe .\2_Generator.js
> {value: 1, done: false}
> {value: 2, done: false}
> {value: 3, done: false}
> {value: 4, done: false}
```

Iterators

```
const array = [1, 2, 3];
const iterator = array[Symbol.iterator]();
let result = iterator.next();
while (!result.done) {
    console.log(result);
    result = iterator.next();
}
```

```
C:\Program Files\nodejs\node.exe .\3_Iterators.js
> {value: 1, done: false}
> {value: 2, done: false}
> {value: 3, done: false}
```

Iterators

```
const map = new Map();
map.set("key1", "val1");
map.set("key2", "val2");
const iterator = map[Symbol.iterator]();
let result = iterator.next();
while (!result.done) {
  console.log(result);
  result = iterator.next();
}
```

```
C:\Program Files\nodejs\node.exe .\3_Iterators.js
{value: Array(2), done: false}
  done: false
> value: (2) ['key1', 'val1']
> __proto__: Object
{value: Array(2), done: false}
  done: false
> value: (2) ['key2', 'val2']
> __proto__: Object
```

Iterators

```
const map = new Map();
map.set("key1", "val1");
map.set("key2", "val2");
for (const [key, value] of map) {
  console.log(`${key} and ${value}`);
}
```

```
C:\Program Files\nodejs\node.exe .\3_Iterators.js
key1 and val1
key2 and val2
```

Iterators 1/2

```
function myIterator(start, finish) {
  let index = start;
  let count = 0;

  return {
    next() {
      let result;
      if (index < finish) {
        result = { value: index, done: false };
        index++;
        count++;
        return result;
      }

      return {
        value: count,
        done: true,
      };
    },
  };
}
```

Iterators

```
let iterator = myIterator(0, 10);
let result = iterator.next();
while (!result.done) {
  console.log(result.value);
  result = iterator.next();
}
```

```
C:\Program Files\nodejs\node.exe .\3_Iterators.js
0
1
2
3
4
5
6
7
8
9
```