Objects

methods

The Object **class** represents one of JavaScript's data types.

It is used to store various keyed collections and more complex entities.

▼ What is the Object constructor?

```
new Object(value)
```

▼ What is the object initializer / literal notation?

```
const object1 = { a: 'foo', b: 42, c: {} };
```

▼ What is Object.create()?

The Object.create() method creates a new object, using an existing object as the prototype of the newly created object.

```
const person = {
  isHuman: false,
  printIntroduction: function() {
    console.log(`My name is ${this.name}. Am I human? ${this.isHuman}`);
  }
};

const me = Object.create(person);
```

▼ What does Object.assign() do?

The Object.assign() method copies all enumerable own properties from one or more source objects to a target object.

It returns the modified target object.

```
const target = { a: 1, b: 2 };
const source = { b: 4, c: 5 };

const returnedTarget = Object.assign(target, source);

console.log(target);
// expected output: Object { a: 1, b: 4, c: 5 }

console.log(returnedTarget);
// expected output: Object { a: 1, b: 4, c: 5 }
```

▼ What does Object.defineProperty() do?

The static method Object.defineProperty() defines a new property directly on an object, or modifies an existing property on an object, and returns the object.

```
const object1 = {};

Object.defineProperty(object1, 'property1', {
  value: 42,
  writable: false
});

object1.property1 = 77;
// throws an error in strict mode

console.log(object1.property1);
// expected output: 42
```

▼ What does Object.defineProperties() do?

The Object.defineProperties() method defines new or modifies existing properties directly on an object, returning the object.

```
const object1 = {};

Object.defineProperties(object1, {
  property1: {
    value: 42,
    writable: true
```

```
  },
  property2: {}
});

console.log(object1.property1);
// expected output: 42
```

▼ What does Object.entries() do?

The Object.entries() method returns an array of a given object's own enumerable string-keyed property [key, value] pairs, in the same order as that provided by a for...in loop.

```
const object1 = {
  a: 'somestring',
  b: 42
};

for (const [key, value] of Object.entries(object1)) {
  console.log(`${key}: ${value}`);
}

// expected output:
// "a: somestring"
// "b: 42"
// order is not guaranteed
```

▼ What does Object.freeze() do?

The Object.freeze() method freezes an object. A frozen object can no longer be changed.

```
const obj = {
  prop: 42
};

Object.freeze(obj);

obj.prop = 33;
// Throws an error in strict mode

console.log(obj.prop);
// expected output: 42
```

▼ What does Object.fromEntries() do?

The Object.fromEntries() method transforms a list of key-value pairs into an object.

```
const entries = new Map([
  ['foo', 'bar'],
  ['baz', 42]
]);

const obj = Object.fromEntries(entries);

console.log(obj);
// expected output: Object { foo: "bar", baz: 42 }
```

▼ What does Object.getOwnPropertyNames() do?

The Object.getOwnPropertyNames() method returns an array of all properties (including non-enumerable properties except for those which use Symbol) found directly in a given object.

```
const object1 = {
  a: 1,
  b: 2,
  c: 3
};

console.log(Object.getOwnPropertyNames(object1));
// expected output: Array ["a", "b", "c"]
```

▼ What does Object.getPrototypeOf() do?

The Object.getPrototypeOf() method returns the prototype (i.e. the value of the internal [[Prototype]] property) of the specified object.

```
const prototype1 = {};
const object1 = Object.create(prototype1);

console.log(Object.getPrototypeOf(object1) === prototype1);
// expected output: true
```

▼ What does Object.is() do?

The Object.is() method determines whether two values are the same **value**.

```
Object.is(value1, value2);
```

▼ What does Object.isFrozen() do?

The Object.isFrozen() determines if an object is frozen.

```
const object1 = {
  property1: 42
};

console.log(Object.isFrozen(object1));
// expected output: false

Object.freeze(object1);

console.log(Object.isFrozen(object1));
// expected output: true
```

▼ What does Object.keys() do?

The Object.keys() method returns an array of a given object's own enumerable property names, iterated in the same order that a normal loop would.

```
const object1 = {
  a: 'somestring',
  b: 42,
  c: false
};

console.log(Object.keys(object1));
// expected output: Array ["a", "b", "c"]
```

▼ What does Object.values() do?

The Object.values() method returns an array of a given object's own enumerable property values, in the same order as that provided by a for...in loop.

```
const object1 = {
  a: 'somestring',
```

```
  b: 42,
  c: false
};

console.log(Object.values(object1));
// expected output: Array ["somestring", 42, false]
```