Object

1. Tổng quan về object
2. Một số thao tác cơ bản với object
3. Tham trị và tham chiếu
4. Duyệt key của object
5. Bài tập thực hành

1. Tổng quan về Object:

Trong JavaScript, "`object`" là một kiểu dữ liệu phổ biến được sử dụng để biểu diễn và lưu trữ dữ liệu dưới dạng các cặp key-value (khóa-giá trị). Mỗi giá trị trong một `object` có thể là một kiểu dữ liệu khác nhau, bao gồm chuỗi, số, mảng, hoặc thậm chí là một `object` khác.

Khai báo object

- Với các loại dữ liệu mình đã biết như number, string, boolean, nó chỉ là một giá trị đơn giản.
- Nhưng với object là kiểu dữ liệu có thể chứa nhiều dữ liệu khác nhau thông qua các cặp key, value
- value có thể là kiểu dữ liệu bất kỳ: number, string, boolean, object, array, function, ...

```
object syntax
{
 key1: value1,
 key2: value2,
 ...
}
```

```
const student = {
 id: 1,
 name: 'Van A',
 name: 'Van B', // same key come later will take precedence
 isHero: true,
 'key has space': 'super', // key with spaces should be wrapped in quotes
 sayHi() {
 console.log('Hello!');
 }
}
```

Lấy value của key:

- Dùng dot operator để truy cập key của object
- Dùng square brackets để truy cập dynamic key của object, kể cả key có space

```
const student = {
 id: 1,
 name: 'Van A',
 isHero: true,
 'avg mark': 9,
}
console.log(student.name);
console.log(student.avg mark); // Syntax Error
console.log(student['avg mark']); // 9
const key = 'avg mark';
console.log(student.key); // undefined
console.log(student[key]); // 9
```

Thêm key mới cho object

```
const student = {
 id: 1,
 name: 'Van A',
 isHero: true,
}
// update value of a key
student.name = 'Van B';
// simply set new key for object
student.age = 18;
student['mark'] = 10;
console.log(student.age); // 18
console.log(student.mark); // 10
```

Xoá một key

Để xoá một key ra khỏi object hiện tại, dùng delete operator

```
const student = {
 id: 1,
 name: 'Van A',
 isHero: true,
}
// Remove "name" key
delete student.name;
console.log(student.name); // undefined
```

2. Một số thao tác cơ bản với object

Đặt tên cho key

- Với tên biến / tên function thì không được dùng reserved keywords.
- Còn với tên key của object thì thoải mái, kể cả reserved keywords, nhưng không khuyến khích sử dụng

```
const student = {
 name: 'Long Do',
 const: 'haha',
 function: 'it works',
 true: 'work too',
}
```

Property value shorthand

```
const name = 'Long Do';
const age = 18;
const student = {
 name: name, // key and value variable have the same name
 age: age, // key and value variable have the same name
}
// shorthand (recommended)
const student = {
 name,
 age,
}
```

Object destructuring

```
const student = {
 name: 'Long Do',
 age: 18,
}
// old way
const name = student.name;
const age = student.age;
// new way usign object destructuring
const { name, age } = student; // recommended
```

Kiểm tra key có trong object không?

Dùng `in` operator để kiểm tra một key có tồn tại trong object không

```
const student = {
 name: 'Long Do',
 age: 18,
}
'name' in student // true
'age' in student // true
'isHero' in student // false
```

Clone object

```
const student = {
 name: 'Long Do',
 age: 18,
}

const moreProps = {
 isHero: true,
 gender: 'male',
}
// v1: using Object.assign()
const clonedStudent = Object.assign({}, student, moreProps);
// v2: using spread operator (shorter, easier to read)
const clonedStudent2 = {
 ...student,
 ...moreProps,
}
```

Deep clone object

```
const student = {
 name: 'Long Do',
 age: 18,
 mark: {
 math: 10,
 english: 7,
 }
}
const clonedStudent = {
 ...student,
}
clonedStudent.mark.math = 1;
console.log(student.mark.math); // 1  haha
// solution, clone nested levels if any
const clonedStudent = {
 ...student,
 mark: {
 ...student.mark,
 }
}
clonedStudent.mark.math = 1;
console.log(student.mark.math); // 10 works now
```

3. Tham trị và tham chiếu

#	Data	Type

#	Data	Type
1	Primitive Type / Value Type	Primitive Type / Value Type
2	Reference Type	object, array, function

Phân biệt tham trị với tham chiếu

```
// value type, the variable store the value
const name = 'Long Do';
const age = 18;
const isHero = true;
const selectedStudent = null;
// |----------------------------|
// | name = 'Long Do' |
// |____________________________|
// reference type, the variable store the address/ref of the value
// so in this example
// object value { name, age, ... } will be store at address 123456
(somewhere in memory)
// and the student variable just hold the address 123456 (that's why we
call reference)
const student = {
 name: 'Long Do',
 age: 18,
}
// |----------------------| | ADDRESS: 123456 |
// | student = 123456 | ---> | VALUE: { name, age, ... } |
// |______________________| |_____________________________|
```

Phép gán với object

```
// primitive type
const a = 5;
let b = a;
b = 10;
console.log(a); // 5
```

```
// reference type
const student1 = { name: 'Att Lab', };
const student2 = student1;****
student2.name = 'Long Do';
console.log(student1.name); // Long Do ???
```

So sánh object

Khi so sánh tham chiếu (object, array, function) thì địa chỉ tham chiếu sẽ được đem ra so sánh. Nếu cùng trỏ về một địa chỉ tham chiếu thì sẽ trả về true, còn lại là false.

```
const student1 = { name: 'Long Do', };
const student2 = student1; // copy reference from student1 to student2
student1 === student2; // true as they both point to the same reference
```

```
const student1 = { name: 'Long Do', }; // reference 1
const student2 = { name: 'Long Do', }; // reference 2
student1 === student2; // false as they are different refs
```

> Trường hợp bạn cần so sánh 2 objects, có thể xem xét so sánh các key primitive types của 2 objects.

Pass by value vs Pass by reference

```
function changePrimitive(name, age) {
 name = 'Long Do';
 age = 18;
}
let name = 'Long';
let age = 17;
changePrimitive(name, age);
console.log(name); // 'Long'
console.log(age); // 17
```

```
function changeReference(student) {
 student.name = 'Long Do';
 student.age = 18;
}
const student = {
 name: 'Long',
 age: 17,
}
changeReference(student);
console.log(student.name); // 'Long Do'
console.log(student.age); // 18
```

4. Duyệt keys của object

> Nên dùng for...in để duyệt keys của object

```
const student = {
 id: 1,
```

```
 name: 'Van A',
 isHero: true,
}
const keyList = Object.keys(student); // ['id', 'name', 'isHero']
for (let i = 0; i < keyList.length; i++) {
 const key = keyList[i];
 console.log('key:', key); // id, name, isHero
 console.log('value:', student[key]); // 1, 'Van A', true
}
// or a similar way using forEach
Object.keys(student).forEach(key => {
 console.log('key:', key); // id, name, isHero
 console.log('value:', student[key]); // 1, 'Van A', true
})
```

```
const student = {
 id: 1,
 name: 'Van A',
 isHero: true,
}
// recommended
for (let key in student) {
 console.log('key:', key); // id, name, isHero
 console.log('value:', student[key]); // 1, 'Van A', true
}
```

5. Bài tập về object đơn giản:

```
// 1. Create an object student with name is Do Long and age is 18.
const student = {};
student.name = 'Do Long';
student.age = 18;
console.log(student);
// ----
const student = {
 name: 'Do Long',
 name: 'Att Lab',
 age: 18,
};
console.log(student);
```

```
// 2. Check if an object is empty (means have no keys)
// { } --> no keys --> length of key list is 0
function isEmpty(obj) {
 return Object.keys(obj).length === 0;
}
console.log(isEmpty({}));
```

```
console.log(isEmpty({ id: 1 }));
// let data = {};
// data = { id: 1 }
// data && data.id
// data?.id
```

```
// 3. Get average mark of an object
// calcAvgMark({ math: 10, english: 8 }) --> 9
function calcAvgMark(mark) {
 if (!mark) return -1;
 // avg = sum / length
 const length = Object.keys(mark).length;
 let sum = 0;
 for (const key in mark) {
 const value = mark[key];
 sum += value;
 }
 return (sum / length).toFixed(1);
}
console.log(calcAvgMark({ math: 10, english: 7 }));
```

```
// 4. Viết hàm function cloneObject(obj) để clone một object obj truyền vào, và trả về là một object mới có đầy đủ các keys của object truyền vào.

// Lưu ý: Không sử dụng Object.assign() và spread operator

// Ví dụ:

const studentA = { name: 'Bob', math: 9 };
const studentB = cloneObject(studentA);

console.log(studentA === studentB); // should be false
console.log(studentB.name); // Bob
console.log(studentB.math); // 9

// Giải:

function cloneObject(obj) {
    const newObj = {};

    for(let key in obj) {
        newObj.key = obj[key];
    }

    return newObj;
}
```

```
// 5. Viết hàm isEqual(obj1, obj2) nhận vào 2 objects và trả về: true nếu số lượng keys của 2 objects bằng nhau, và giá trị của từng key cũng bằng nhau (dùng === để so sánh), ngược lại là false

// Ví dụ:

isEqual({}, {}) // --> true

isEqual({ name: 'Bob' }, { name: 'Alice' }) // --> false

isEqual({ name: 'Bob' }, { name: 'Bob' }) // --> true

isEqual({ name: 'Bob' }, { name: 'Bob', age: 18 }) // --> false

// Giả sử kiểu dữ liệu của các thuộc tính của cả 2 objects đều là kiểu dữ liệu primitive.

// Giải bài tập:

function isEqual(obj1, obj2) {
    if(Object.keys(obj1).length === 0 && Object.keys(obj2) === 0) return true;

    const key1List = Object.keys(obj1);
    const key2List = Object.keys(obj2);
    if(key1List.length !== key2List.length) return false;
    for(let i=0; i< key1List.length - 1; i++) {
        if(key1List[i] !== key2List[i]) return false;
        if(obj1[key1List[i]] !== obj2[key2List[i]]) return false;
    }

    return true;
}
```