Lecture 6: Objects & Destructuring

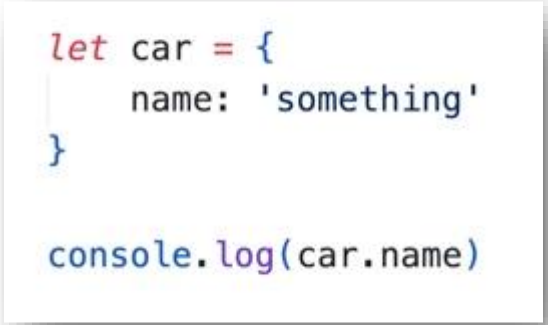
What are Objects?

- **Objects** in JavaScript are collections of key-value pairs. They allow us to store multiple values in a single variable, and each value is associated with a unique key (property).

- Think of **objects** like real-world objects with properties (like a car with properties such as color, make, model).

Object Properties

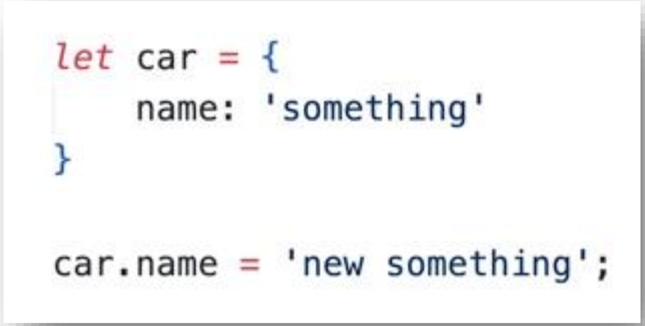
Accessing Object Properties



```
let car = {
    name: 'something'
}

console.log(car.name)
```

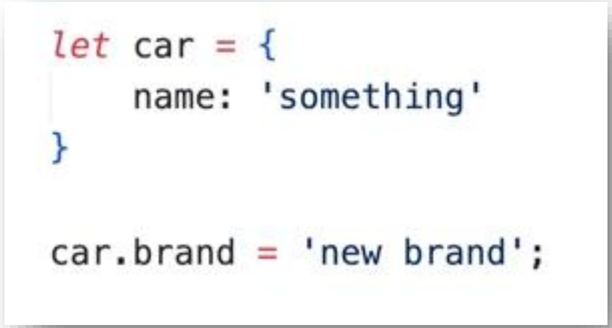
Modifying Object Properties



```
let car = {
    name: 'something'
}

car.name = 'new something';
```

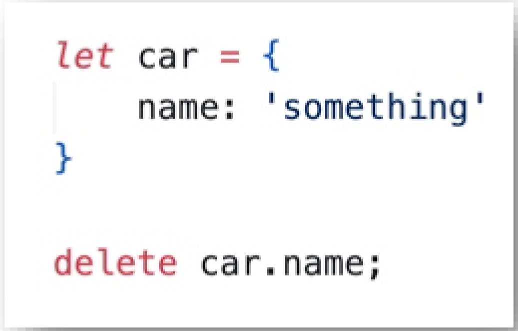
Adding Object Properties



```
let car = {
    name: 'something'
}

car.brand = 'new brand';
```

Deleting Object Properties

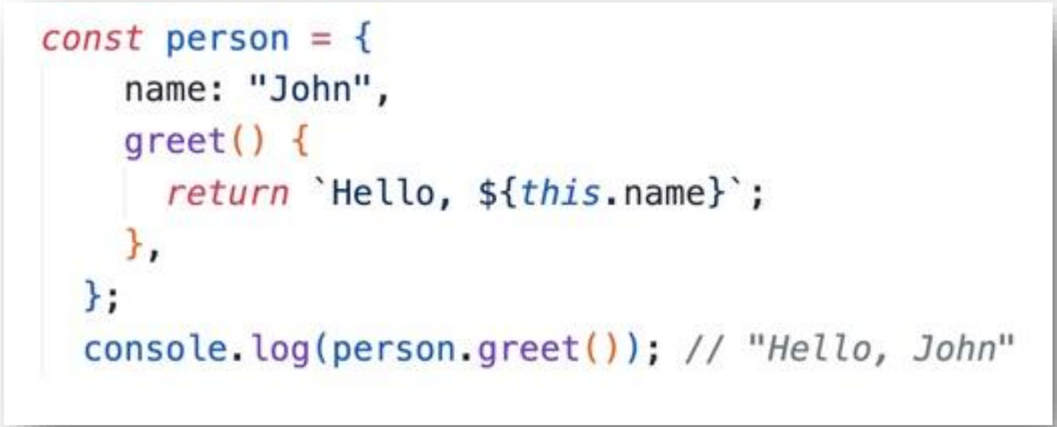


```
let car = {
    name: 'something'
}

delete car.name;
```

Object Methods

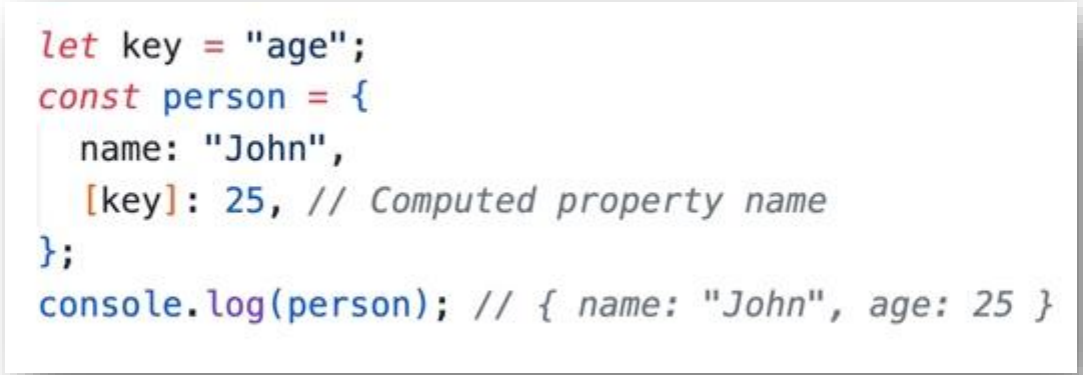
Functions can be properties in objects, known as **methods**



```
const person = {
    name: "John",
    greet() {
      return `Hello, ${this.name}`;
    },
  };
  console.log(person.greet()); // "Hello, John"
```

Computed Property Names

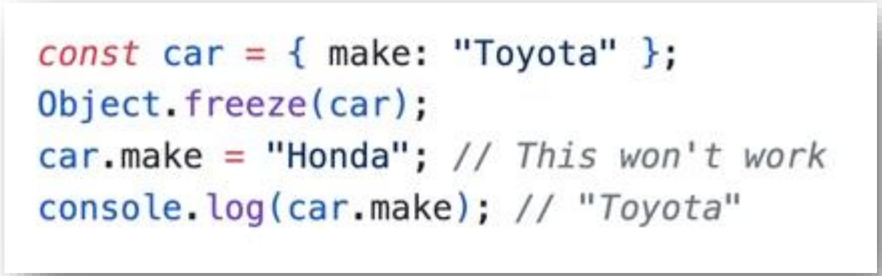
You can use expressions to define property names dynamically.



```
let key = "age";
const person = {
  name: "John",
  [key]: 25, // Computed property name
};
console.log(person); // { name: "John", age: 25 }
```

Object Freezing

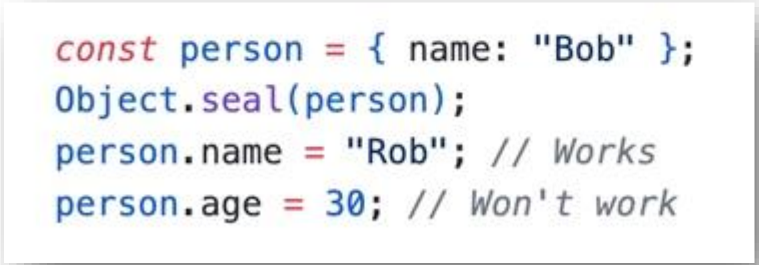
Object freezing prevents any changes made to an object



```
const car = { make: "Toyota" };
Object.freeze(car);
car.make = "Honda"; // This won't work
console.log(car.make); // "Toyota"
```

Object Sealing

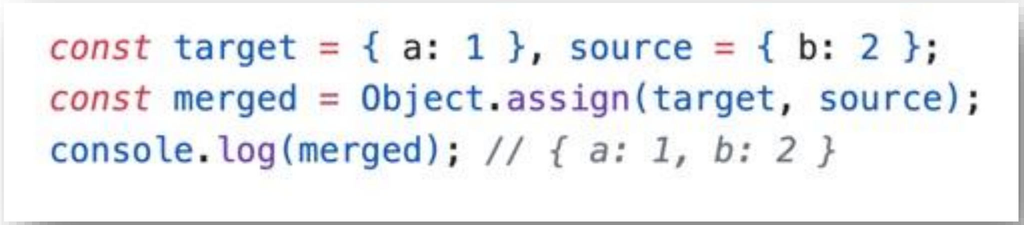
Object Sealing allows modification of existing properties, but prevents adding new ones



```
const person = { name: "Bob" };
Object.seal(person);
person.name = "Rob"; // Works
person.age = 30; // Won't work
```

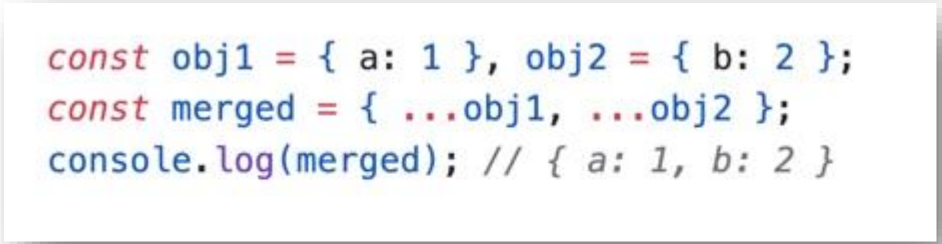
Merging Objects

Using **Object.assign**



```javascript
const target = { a: 1 }, source = { b: 2 };
const merged = Object.assign(target, source);
console.log(merged); // { a: 1, b: 2 }
```

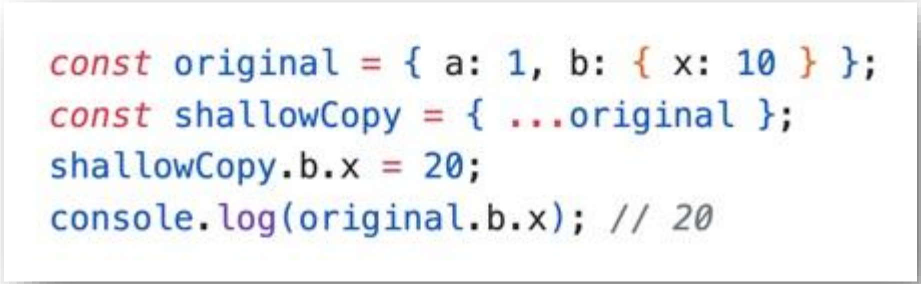
Using **spread operator (...)**



```javascript
const obj1 = { a: 1 }, obj2 = { b: 2 };
const merged = { ...obj1, ...obj2 };
console.log(merged); // { a: 1, b: 2 }
```

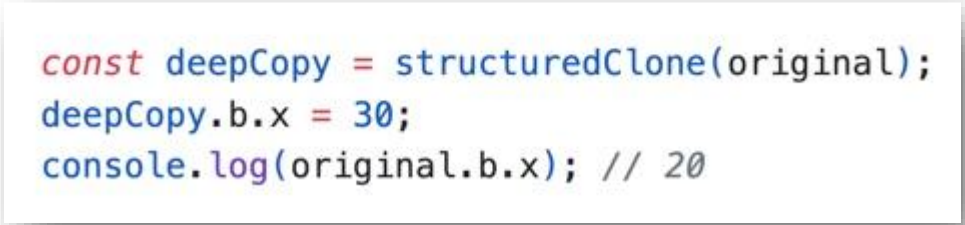
Cloning Objects

Shallow Copy



```
const original = { a: 1, b: { x: 10 } };
const shallowCopy = { ...original };
shallowCopy.b.x = 20;
console.log(original.b.x); // 20
```

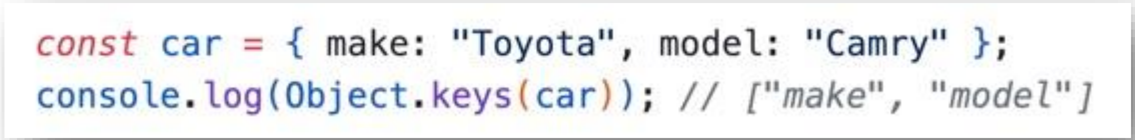
Deep Copy



```
const deepCopy = structuredClone(original);
deepCopy.b.x = 30;
console.log(original.b.x); // 20
```

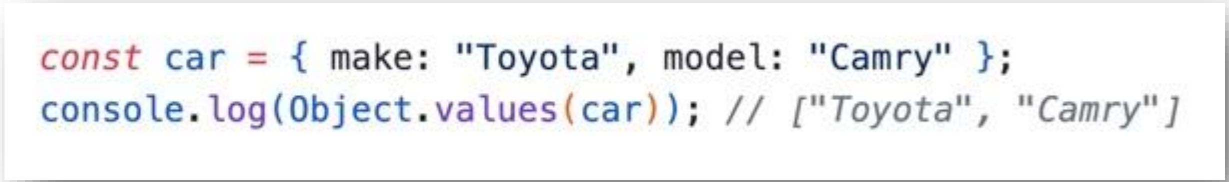
Object Iteration Techniques

Object.keys()



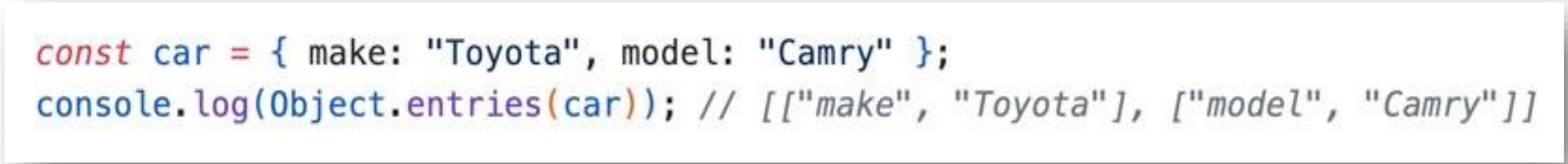
```javascript
const car = { make: "Toyota", model: "Camry" };
console.log(Object.keys(car)); // ["make", "model"]
```

Object.values()



```javascript
const car = { make: "Toyota", model: "Camry" };
console.log(Object.values(car)); // ["Toyota", "Camry"]
```

Object.entries()



```javascript
const car = { make: "Toyota", model: "Camry" };
console.log(Object.entries(car)); // [["make", "Toyota"], ["model", "Camry"]]
```