Here are 10 methods commonly used with JavaScript objects:

1. **Object.keys()**: Returns an array of a given object's own enumerable property names.
   ```javascript
   const obj = { a: 1, b: 2, c: 3 };
   console.log(Object.keys(obj)); // Outputs: ["a", "b", "c"]
   ```

2. **Object.values()**: Returns an array of a given object's own enumerable property values.
   ```javascript
   const obj = { a: 1, b: 2, c: 3 };
   console.log(Object.values(obj)); // Outputs: [1, 2, 3]
   ```

3. **Object.entries()**: Returns an array of a given object's own enumerable property `[key, value]` pairs.
   ```javascript
   const obj = { a: 1, b: 2, c: 3 };
   console.log(Object.entries(obj)); // Outputs: [["a", 1], ["b", 2], ["c", 3]]
   ```

4. **Object.assign()**: Copies the values of all enumerable own properties from one or more source objects to a target object.
   ```javascript
   const obj1 = { a: 1 };
   const obj2 = { b: 2 };
   const obj3 = Object.assign({}, obj1, obj2);
   console.log(obj3); // Outputs: { a: 1, b: 2 }
   ```

5. **Object.freeze()**: Freezes an object: other code can't delete or change its properties.
   ```javascript
   const obj = { a: 1 };
   Object.freeze(obj);
   obj.a = 2;
   console.log(obj); // Outputs: { a: 1 }
   ```

6. **Object.seal()**: Prevents new properties from being added to an object and marks all existing properties as non-configurable.
   ```javascript
   const obj = { a: 1 };
   Object.seal(obj);
   obj.b = 2;
   ```

```
delete obj.a;
console.log(obj); // Outputs: { a: 1 }
```

7. **Object.hasOwnProperty()**: Returns a boolean indicating whether the object has the specified property as its own property (not inherited).
```javascript
const obj = { a: 1, b: 2 };
console.log(obj.hasOwnProperty("a")); // Outputs: true
console.log(obj.hasOwnProperty("toString")); // Outputs: false
```

8. **Object.getOwnPropertyNames()**: Returns an array of all properties (enumerable or not) found directly upon a given object.
```javascript
const obj = { a: 1, b: 2 };
console.log(Object.getOwnPropertyNames(obj)); // Outputs: ["a", "b"]
```

9. **Object.create()**: Creates a new object with the specified prototype object and properties.
```javascript
const parent = { a: 1 };
const child = Object.create(parent);
child.b = 2;
console.log(child.a); // Outputs: 1
console.log(child.b); // Outputs: 2
```

10. **Object.entries()**: Returns an array of a given object's own enumerable property `[key, value]` pairs.
```javascript
const obj = { a: 1, b: 2, c: 3 };
console.log(Object.entries(obj)); // Outputs: [["a", 1], ["b", 2], ["c", 3]]
```

These methods are commonly used for manipulating and working with objects in JavaScript, enabling tasks such as iterating over properties, copying objects, and checking property existence.