1. Create an object to represent a book with properties:

```
const book = {
  title: "The Great Gatsby",
  author: "F. Scott Fitzgerald",
  pages: 180,
  isRead: true
};
```

---

2. Function to return the value of a given property:

```
function getPropertyValue(obj, property) {
  return obj[property];
}

// Example usage:
const book = { title: "1984", author: "George Orwell" };
console.log(getPropertyValue(book, "author")); // "George Orwell"
```

---

3. Function to add and update properties in an object:

```
function modifyObject(obj, newPublisher, newPages) {
  obj.publisher = newPublisher; // Add new property
  obj.pages = newPages; // Update existing property
}

// Example usage:
const book = { title: "1984", pages: 200 };
modifyObject(book, "Penguin", 250);
console.log(book); // { title: '1984', pages: 250, publisher: 'Penguin' }
```

---

4. Function to check if a key exists in an object:

```
function hasKey(obj, key) {
  return obj.hasOwnProperty(key);
}

// Example usage:
const user = { name: "Alice", age: 25 };
console.log(hasKey(user, "name")); // true
console.log(hasKey(user, "gender")); // false
```

---

5. Function to delete a specific property:

```
function deleteProperty(obj, property) {
  delete obj[property];
}

// Example usage:
```

```
const user = { username: "JohnDoe", password: "12345" };
deleteProperty(user, "password");
console.log(user); // { username: 'JohnDoe' }
```

---

6. Function to loop through keys and log key-value pairs:

```
function logObjectKeysAndValues(obj) {
  for (const key in obj) {
    console.log(`Key: ${key}, Value: ${obj[key]}`);
  }
}

// Example usage:
const user = { name: "Alice", age: 30 };
logObjectKeysAndValues(user);
// Output:
// Key: name, Value: Alice
// Key: age, Value: 30
```

---

7. Access `city` and modify `subjects`:

```
const student = {
  name: "Alex",
  address: { city: "New York", zip: "10001" },
  subjects: ["Math", "Science"]
};

function handleStudent() {
  // Access city
  console.log(student.address.city); // "New York"

  // Add new subject
  student.subjects.push("History");
  console.log(student.subjects); // ["Math", "Science", "History"]
}

handleStudent();
```

---

8. Create shallow and deep copies:

```
const person = {
  name: "Alice",
  details: { age: 30, city: "Hyderabad" }
};

function createCopies(obj) {
  // Shallow copy
  const shallowCopy = { ...obj };

  // Deep copy
  const deepCopy = JSON.parse(JSON.stringify(obj));

  return { shallowCopy, deepCopy };
}
```

```
// Example usage:
const copies = createCopies(person);
console.log(copies.shallowCopy);
console.log(copies.deepCopy);
```

---

9. Merge two objects with overriding:

```
function mergeObjects(obj1, obj2) {
  return { ...obj1, ...obj2 };
}

// Example usage:
const obj1 = { a: 1, b: 2 };
const obj2 = { b: 3, c: 4 };
const mergedObj = mergeObjects(obj1, obj2);
console.log(mergedObj); // { a: 1, b: 3, c: 4 }
```

---

10. Convert an object to an array of key-value pairs:

```
function objectToArray(obj) {
  return Object.entries(obj).map(([key, value]) => ({ key, value }));
}

// Example usage:
const user = { id: 101, name: "John", age: 30 };
console.log(objectToArray(user));
// Output:
// [
//   { key: 'id', value: 101 },
//   { key: 'name', value: 'John' },
//   { key: 'age', value: 30 }
// ]
```