# Week 6: Objects & Arrays

**วัตถุประสงค์**

1. จัดการ objects ขั้นพื้นฐาน
2. ใช้ array methods: map, filter, reduce
3. ใช้ spread operator { ...obj }
4. วิเคราะห์ข้อมูลด้วย JavaScript

---

## Activity 1: Objects & Basic Methods

### ขั้นตอน 1: สร้าง User Objects

```
// สร้าง user objects
const users = [
  {
    id: 1,
    name: "John",
    email: "john@example.com",
    role: "admin",
  },
  {
    id: 2,
    name: "Jane",
    email: "jane@example.com",
    role: "user",
  },
  {
    id: 3,
    name: "Bob",
    email: "bob@example.com",
    role: "admin",
  },
];
```

**Tasks**:

1. ดึงชื่อ admins ออกมา
2. ดึง email ของ user ทั้งหมด
3. หา user ที่มี id = 2

**โจทย์**:

```
// TODO 1: สร้าง admins array โดยใช้ filter()
const admins = users.filter(/* ??? */);
console.log(admins);

// TODO 2: สร้าง emails array โดยใช้ map()
const emails = users.map(/* ??? */);
console.log(emails);

// TODO 3: หา user ที่มี id = 2 โดยใช้ find()
const userWithId2 = users.find(/* ??? */);
console.log(userWithId2);
```

**เฉลย:**

```
const admins = users.filter((u) => u.role === "admin");
const emails = users.map((u) => u.email);
const userWithId2 = users.find((u) => u.id === 2);
```

## ขั้นตอน 2: Spread Operator & Immutable Update

```
// Original user
const user = {
  id: 1,
  name: "John",
  role: "user",
};

// ทำการอัปเดต role เป็น admin โดยไม่เปลี่ยน original
const promoted = {
  ...user,
  role: "admin",
};

console.log(user.role); // "user"  (ไม่เปลี่ยน)
console.log(promoted.role); // "admin" (ใหม่)
```

**Tasks:**

- สร้าง object ใหม่ที่ copied จาก user แต่เปลี่ยน role
- ตรวจสอบว่า original user ไม่เปลี่ยน

**โจทย์:**

```
const users = [
  { id: 1, name: "John", role: "user" },
  { id: 2, name: "Jane", role: "user" },
];

// TODO: อัปเดต user id=1 ให้เป็น admin (ไม่เปลี่ยน original)
const updated = users.map((u) => (u.id === 1 ? { ...u, role: "admin" } : u));

console.log("Original:", users[0].role); // "user"
console.log("Updated:", updated[0].role); // "admin"
```

## ขั้นตอน 3: Group & Count ด้วย reduce()

```
const users = [
  { name: "John", role: "admin" },
  { name: "Jane", role: "user" },
  { name: "Bob", role: "admin" },
];

// นับจำนวน user แต่ละ role
const count = users.reduce((acc, u) => {
  acc[u.role] = (acc[u.role] || 0) + 1;
  return acc;
}, {});

console.log(count);
// Output: { admin: 2, user: 1 }
```

**โจทย์:**

```
// TODO: จัดกลุ่ม users ตามบทบาท (role)
const grouped = users.reduce((acc, u) => {
  if (!acc[u.role]) acc[u.role] = [];
  acc[u.role].push(u);
  return acc;
}, {});

console.log(grouped);
// Output: {
//   admin: [ { name: 'John', role: 'admin' }, ... ],
//   user: [ { name: 'Jane', role: 'user' } ]
// }
```

## Activity 2: Array Methods

### ขั้นตอน 1: Transform with map()

```
const products = [
  { name: "Laptop", price: 50000 },
  { name: "Book", price: 500 },
  { name: "Shirt", price: 1000 },
];

// ดึงชื่อสินค้า
const names = products.map((p) => p.name);
console.log(names);
// Output: ['Laptop', 'Book', 'Shirt']

// ลดราคา 20%
const discounted = products.map((p) => ({
  ...p,
  price: Math.round(p.price * 0.8),
}));
console.log(discounted);
```

### โจทย์:

```
// TODO 1: สร้าง display strings
const display = products.map((p) => `${p.name} - ฿${p.price}`);
console.log(display);

// TODO 2: แปลงสินค้าที่แพงเป็น "expensive" และถูกเป็น "cheap"
const labeled = products.map((p) => ({
  ...p,
  category: p.price > 5000 ? "expensive" : "cheap",
}));
console.log(labeled);
```

### ขั้นตอน 2: Filter & Chain

```
const products = [
  { name: "Laptop", price: 50000, category: "electronics" },
  { name: "Book", price: 500, category: "books" },
  { name: "Phone", price: 25000, category: "electronics" },
];

// สินค้าใน electronics category
const electronics = products.filter((p) => p.category === "electronics");
console.log(electronics);

// สินค้าราคาน้อยกว่า 10000
const affordable = products.filter((p) => p.price < 10000);
console.log(affordable);

// Chaining: electronics ราคาน้อยกว่า 30000
const result = products
  .filter((p) => p.category === "electronics")
  .filter((p) => p.price < 30000)
  .map((p) => p.name);
console.log(result);
// Output: ['Phone']
```

**โจทย์:**

```
// TODO 1: สินค้า books ที่ราคาน้อยกว่า 1000
const cheapBooks = products.filter(/* ??? */).filter(/* ??? */);

// TODO 2: ชื่อสินค้า electronics ที่ราคาน้อยกว่า 30000
const phoneNames = products.filter(/* ??? */).filter(/* ??? */).map(/* ??? */);
```

### ขั้นตอน 3: Reduce & Aggregate

```
const products = [
  { name: "Laptop", price: 50000 },
  { name: "Book", price: 500 },
  { name: "Shirt", price: 1000 },
];

// รวมราคาทั้งหมด
const total = products.reduce((acc, p) => acc + p.price, 0);
console.log(total);
// Output: 51500

// สินค้าที่แพงที่สุด
const mostExpensive = products.reduce((max, p) =>
  p.price > max.price ? p : max
);
console.log(mostExpensive);
// Output: { name: 'Laptop', price: 50000 }
```

**โจทย์:**

```
// TODO 1: ราคาเฉลี่ย
const average = products.reduce((acc, p) => acc + p.price, 0) / products.length;

// TODO 2: สินค้าที่ถูกที่สุด
const cheapest = products.reduce((min, p) => (p.price < min.price ? p : min));

console.log("Average:", average);
console.log("Cheapest:", cheapest);
```

## Activity 3: Data Analysis

### ขั้นตอน 1: Basic Statistics

```
const sales = [
  { date: "2024-01-01", amount: 1000, region: "North", status: "completed" },
  { date: "2024-01-01", amount: 1500, region: "South", status: "completed" },
  { date: "2024-01-02", amount: 2000, region: "North", status: "completed" },
  { date: "2024-01-02", amount: 500, region: "East", status: "pending" },
  { date: "2024-01-03", amount: 3000, region: "South", status: "completed" },
  { date: "2024-01-03", amount: 800, region: "West", status: "completed" },
];

// รวมยอดขาย
const total = sales.reduce((acc, s) => acc + s.amount, 0);
console.log("Total:", total);

// ยอดขายเฉลี่ย
const average = total / sales.length;
console.log("Average:", average);

// ยอดขายสูงสุด
const highest = sales.reduce((max, s) => (s.amount > max.amount ? s : max));
console.log("Highest:", highest);
```

```
// นับ completed
const completedCount = sales.filter((s) => s.status === "completed").length;
console.log("Completed count:", completedCount);
```

### ขั้นตอน 2: Group & Summarize

```
// จัดกลุ่มตามภูมิภาค (region)
const byRegion = sales.reduce((acc, s) => {
  if (!acc[s.region]) {
    acc[s.region] = { total: 0, count: 0 };
  }
  acc[s.region].total += s.amount;
  acc[s.region].count += 1;
  return acc;
}, {});

console.log(byRegion);
// Output: {
//   North: { total: 2800, count: 2 },
//   South: { total: 4500, count: 2 },
//   East: { total: 500, count: 1 },
//   West: { total: 800, count: 1 }
// }
```

### โจทย์:

```
// TODO: เพิ่ม average ให้กับแต่ละ region
const byRegionWithAvg = sales.reduce((acc, s) => {
  if (!acc[s.region]) {
    acc[s.region] = { total: 0, count: 0, average: 0 };
  }
  acc[s.region].total += s.amount;
  acc[s.region].count += 1;
  acc[s.region].average = acc[s.region].total / acc[s.region].count;
  return acc;
}, {});

console.log(byRegionWithAvg);
```

### ขั้นตอน 3: Filter & Top

```
// Top 2 ยอดขายสูงสุด
const top2 = sales.sort((a, b) => b.amount - a.amount).slice(0, 2);
console.log("Top 2:", top2);

// ยอดขาย completed เฉพาะ
const completedSales = sales.filter((s) => s.status === "completed");
console.log("Completed:", completedSales.length);

// ยอดขาย North ทั้งหมด
const northSales = sales.filter((s) => s.region === "North");
const northTotal = northSales.reduce((acc, s) => acc + s.amount, 0);
console.log("North total:", northTotal);

// Pending ที่ต้องจัดการ
const pending = sales.filter((s) => s.status === "pending");
console.log("Pending count:", pending.length);
```

โจทย์:

```
// TODO 1: ยอดขาย completed ในภูมิภาค South
const southCompleted = sales
  .filter((s) => s.region === "South" && s.status === "completed")
  .reduce((acc, s) => acc + s.amount, 0);
```

```
// TODO 2: ยอดขายเฉลี่ยต่อวัน (daily average)
const days = [...new Set(sales.map((s) => s.date))].length;
const dailyAverage = total / days;

console.log("South completed:", southCompleted);
console.log("Daily average:", dailyAverage);
```

### ขั้นตอน 4: ตัวอย่างการใช้งาน

```
// สรุป Report
const report = {
  totalSales: total,
  averageSale: average.toFixed(2),
  completedCount: completedCount,
  pendingCount: pending.length,
  bestRegion: "South", // จากข้อมูล
  salesByRegion: byRegionWithAvg,
};

console.log("=== SALES REPORT ===");
console.log(report);
```

## สิ่งที่ต้องเข้าใจ

### Array Methods

| Method | ใช้สำหรับ | ตัวอย่าง |
|---|---|---|
| `map()` | แปลง element | `users.map(u => u.name)` |
| `filter()` | เลือก element | `users.filter(u => u.active)` |
| `find()` | หา 1 element | `users.find(u => u.id === 5)` |
| `reduce()` | รวม/จัดกลุ่ม | `items.reduce((sum, i) => sum + i.price, 0)` |
| `sort()` | เรียงลำดับ | `items.sort((a, b) => b.price - a.price)` |

### Spread Operator

```
// Copy object
const copy = { ...obj };

// Update property
const updated = { ...user, role: "admin" };

// Merge objects
const merged = { ...obj1, ...obj2 };
```

### Reduce Pattern

```
// รูปแบบพื้นฐาน
array.reduce((accumulator, currentItem) => {
  // ทำสิ่งที่ต้องการ
  return accumulator; // ส่งกลับ accumulator
}, initialValue);
```

### Checklist

- [ ] ข้อ 1 ออกมาถูก (filter, map, find)
- [ ] ข้อ 2 ใช้ spread operator และ original ไม่เปลี่ยน
- [ ] ข้อ 3 reduce() จัดกลุ่ม
- [ ] ข้อ 4 map() transform ข้อมูล
- [ ] ข้อ 5 filter() + chain
- [ ] ข้อ 6 reduce() รวมและ find()
- [ ] ข้อ 7 reduce() จัดกลุ่มข้อมูล
- [ ] ข้อ 8 filter() + reduce() รวม
- [ ] ข้อ 9 sort() และ slice()

### เทคนิคการเขียน

ดี: ฟังก์ชันเล็กๆ

```
const adults = users.filter((u) => u.age >= 18);
const names = adults.map((a) => a.name);
```

### ดี: Chain methods

```
const result = users.filter((u) => u.age >= 18).map((u) => u.name);
```

### อย่าทำ: Loop ธรรมดา

```
// Avoid
const result = [];
for (let u of users) {
  if (u.age >= 18) {
    result.push(u.name);
  }
}
```

---