JavaScript Array Methods Cheat Sheet: filter() & map()

filter() - Creates a new array with elements that pass a test

Basic Syntax

javascript

```
array.filter(callback(element, index, array), thisArg)
```

Examples

Filter numbers greater than 5:

javascript

```
const numbers = [1, 4, 7, 2, 9, 3];
const filtered = numbers.filter(num => num > 5);
// Result: [7, 9]
```

Filter even numbers:

javascript

```
const numbers = [1, 2, 3, 4, 5, 6];
const evens = numbers.filter(num => num % 2 === 0);
// Result: [2, 4, 6]
```

Filter objects by property:

javascript

```
const users = [
  { name: 'Alice', age: 25 },
  { name: 'Bob', age: 17 },
  { name: 'Charlie', age: 30 }
];

const adults = users.filter(user => user.age >= 18);
// Result: [{ name: 'Alice', age: 25 }, { name: 'Charlie', age: 30 }]
```

Filter truthy values:

javascript

```
const mixed = [0, 'hello', '', 42, null, undefined, false, 'world'];
const truthy = mixed.filter(Boolean);
// Result: ['hello', 42, 'world']
```

map() - Creates a new array by transforming every element

Basic Syntax

javascript

```
array.map(callback(element, index, array), thisArg)
```

Examples

Double numbers:

javascript

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num => num * 2);
// Result: [2, 4, 6, 8]
```

Extract properties:

javascript

```
const users = [
  { name: 'Alice', age: 25 },
  { name: 'Bob', age: 17 }
];

const names = users.map(user => user.name);
// Result: ['Alice', 'Bob']
```

Transform objects:

javascript

```
const products = [
  { name: 'apple', price: 1.2 },
  { name: 'banana', price: 0.8 }
];

const withTax = products.map(product => ({
  ...product,
  priceWithTax: product.price * 1.1
}));
// Result: [{ name: 'apple', price: 1.2, priceWithTax: 1.32 }, ...]
```

Convert to different format:

javascript

```
const numbers = [1, 2, 3];
const strings = numbers.map(num => num.toString());
// Result: ['1', '2', '3']
```

Combining filter() and map()

Get names of adults only:

javascript

```
const users = [
  { name: 'Alice', age: 25 },
  { name: 'Bob', age: 17 },
  { name: 'Charlie', age: 30 }
];

const adultNames = users
  .filter(user => user.age >= 18)
  .map(user => user.name);
// Result: ['Alice', 'Charlie']
```

Get squared values of even numbers:

javascript

```
const numbers = [1, 2, 3, 4, 5, 6];
const evenSquares = numbers
  .filter(num => num % 2 === 0)
  .map(num => num * num);
// Result: [4, 16, 36]
```

Process and transform data:

javascript

```
const employees = [
  { name: 'John', department: 'IT', salary: 50000 },
  { name: 'Jane', department: 'HR', salary: 45000 },
  { name: 'Doe', department: 'IT', salary: 60000 }
];

const itEmployeeNames = employees
  .filter(emp => emp.department === 'IT')
  .map(emp => emp.name.toUpperCase());
// Result: ['JOHN', 'DOE']
```

Key Differences

Method	Returns	Purpose	Original Array
`filter()`	New array with matching elements	Filter/select elements	Unchanged
`map()`	New array with transformed elements	Transform elements	Unchanged

Common Patterns

Chaining Multiple Operations

javascript

```
const result = array
  .filter(condition)
  .map(transformation)
  .filter(anotherCondition);
```

Using with Array Methods

javascript

```
// Find and transform
const found = array
  .filter(item => item.id === targetId)
  .map(item => item.name);

// Count after filtering
const count = array.filter(condition).length;
```

Performance Tips

- Chain `filter()` before `map()` when possible to reduce operations
- Use descriptive variable names for clarity
- Consider using `find()` instead of `filter()[0]` when you need only one element

This cheat sheet covers the most common use cases for `filter()` and `map()` - two of the most frequently used array methods in JavaScript!