



# JavaScript

Functional Programming

# Functional Abstraction



Abstraction is the process by which we solve problems permanently.



By giving a complex idea a name, we don't have to spell out the concrete idea every time we need to apply it.



In JavaScript, functions are our most powerful tool for abstraction

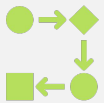


First-class functions, like those in JavaScript, can abstract any idea we can express as a block of code – including operations on other functions!

# First-Class Functions can be:



Stored in variables



Passed as arguments to other functions



Returned by other functions

How does that help us  
solve problems?

# The Process of Functional Abstraction

01

Identify repeated  
code pattern

02

Refactor to be a  
function call

03

Extract the  
function as a  
parameter

# Array Methods

# map

Problem: we need to transform  
each element of an array



```
graph TD; A[Problem: we need to transform each element of an array] --> B[Further Problem: for-loops are repeated code patterns]; B --> C[Solution: Make an abstraction for for-loops that can return an array of results]; C --> D[Sound familiar?];
```

Further Problem: for-loops are  
repeated code patterns

Solution: Make an abstraction for  
for-loops that can return an array of  
results

Sound familiar?

# filter

Problem: we need to choose  
some subset of an array



```
graph TD; A[Problem: we need to choose some subset of an array] --> B[Further Problem: for-loops are repeated code patterns]; B --> C[Solution: Make an abstraction for for-loops that can choose elements from an array];
```

Further Problem: for-loops are  
repeated code patterns

Solution: Make an abstraction  
for for-loops that can choose  
elements from an array



# reduce

Problem: we need to build a value from some operation on each element of an array



```
graph TD; A[Problem: we need to build a value from some operation on each element of an array] --> B[Further Problem: for-loops are repeated code patterns]; B --> C[Solution: Make an abstraction for for-loops that can build a value as it loops];
```

Further Problem: for-loops are repeated code patterns

Solution: Make an abstraction for for-loops that can build a value as it loops

# Transducers

- composable higher order reducers which can reduce over any underlying data type