# Introduction

- JS has first-class functions which means developers can:
    - add them to objects
    - execute them in the context of those objects
    - remove them from their objects,
    - pass them to other functions
    - run them in entirely different contexts
- First class functions initially have no context; they receive one when the program executes them

# The Global Object

- JS creates a global object when it starts running, which serves as the implicit execution context
- In the browser, the global object is the `window` object
    - global functions like `isNan` and `parseInt` are properties of the global object

## Global Object as Implicit Context

- The global object is the implicit context when we evaluate expressions
- If a variable is declared without `var` to declare it as a global variable, it is implicitly evaluated in the context of the global object
    - Now, this variable is actually assigned as a property on the global object with a value of whatever was declared
- Initializeing an undeclared variable automatically creates that variable as a property on the `window` object, since the global object is the implicit context for evaluating expressions

## Global Variables and Function Declarations

- When we declare global variables/functions, JS adds them to the global object as properties
- This looks similar to when you dont declare teh variable, the subtle difference though is you can delete global variables that you don't declare
    - You can't delete declared global variables, but you can delete undeclared global variables
    - A function declaration effectively declares a new variable and assigns a function to it and, since it involves declaration, cannot be deleted
    - You can delete anonymous functions (function expressions)
    - Global variables explicitly added to the global object as properties (foo.a = 'test') can be deleted just like undeclared variables

## Delete variables

```
 var moreFoo = 3;
moreBar = 3;

delete window.moreFoo // false (not deleted)
delete window.moreBar // true  (deleted)
```

## Node vs Browser

- In non-browser JS environments, like Node, the global object is not `window`, it is `global`
- Also, Node introduces an additional Module scope
    - Variables in the module scope are those declared at the top level (not inside a function definition or object) of a node.js file
    - Consequently, module-scoped variables are not added to the global object `global` (`window` in browser)
    - Module-scoped variables are only accessible from anywhere within the file but not accessible anywhere else