2. Functions are Objects

* These two pillars are unique to JS no language has it, that’s why JS is so powerful.
* functions() and arrays are objects.

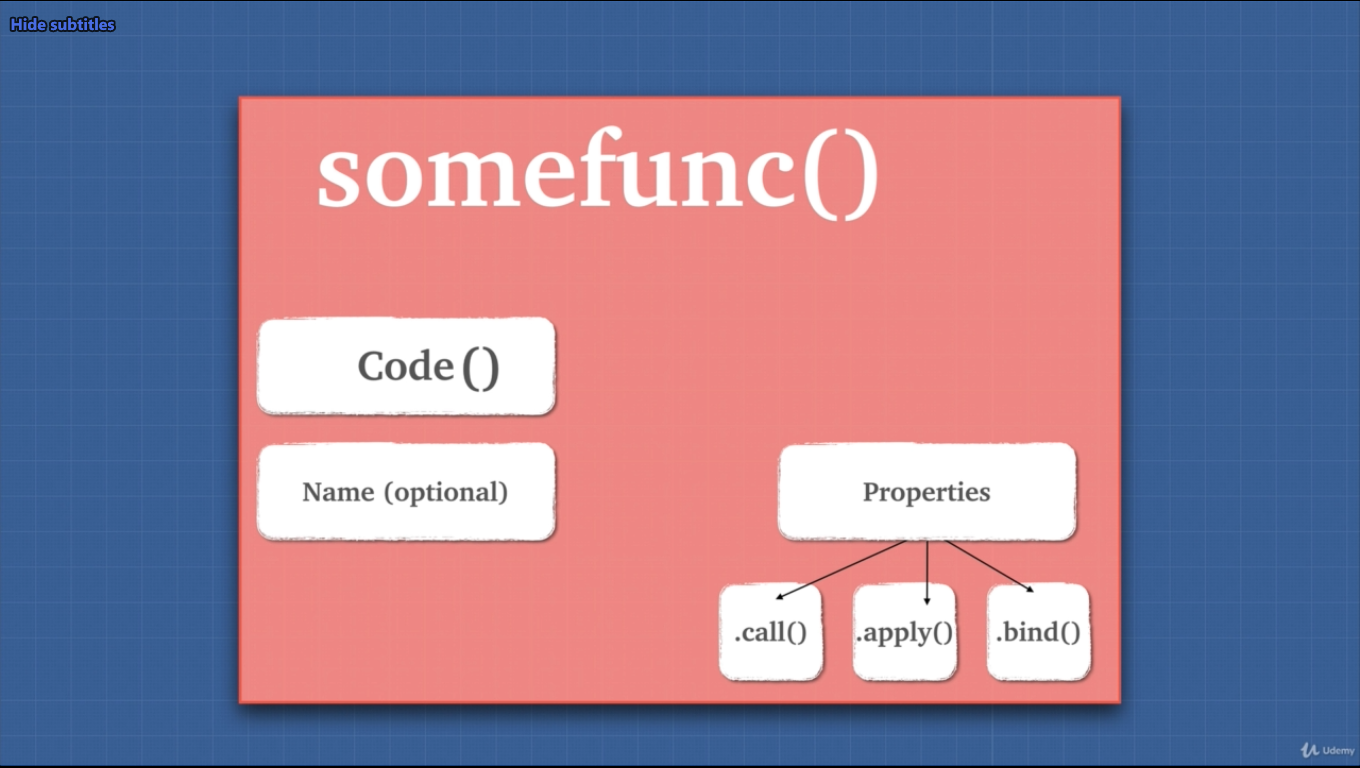
functions():

* When we invoke or call a function, we get this and argument keyword.
* argument is an array like object that has some wired behavior for looping and iteration.
* We can grab parameters using argument keyword despite of not defining any parameter at first. Using this (…args).
* When we define our function, compiler looks at our code lexically to determine what variable we have access to in our variable environment. It also add scope chain.

Ways of invoking a function:

1. Normal way, foo().
2. Inside of an object, a method(), obj.foo(). In this case the this keyword will be updated to the current object.
3. Using call() and apply() method, foo.call().
4. Using the built-in object Function(), it’s a constructor. It also accepts parameter.

* In JS functions are objects it is not common in other languages.
* We can move them around, store them as data.



3. First Class Citizens

* functions are first class citizens is JS.

Behavior of functions:

1. Assigning to a variable or a method inside of an object.
2. We can pass a function as a parameter.
3. We can return functions as values from other functions.

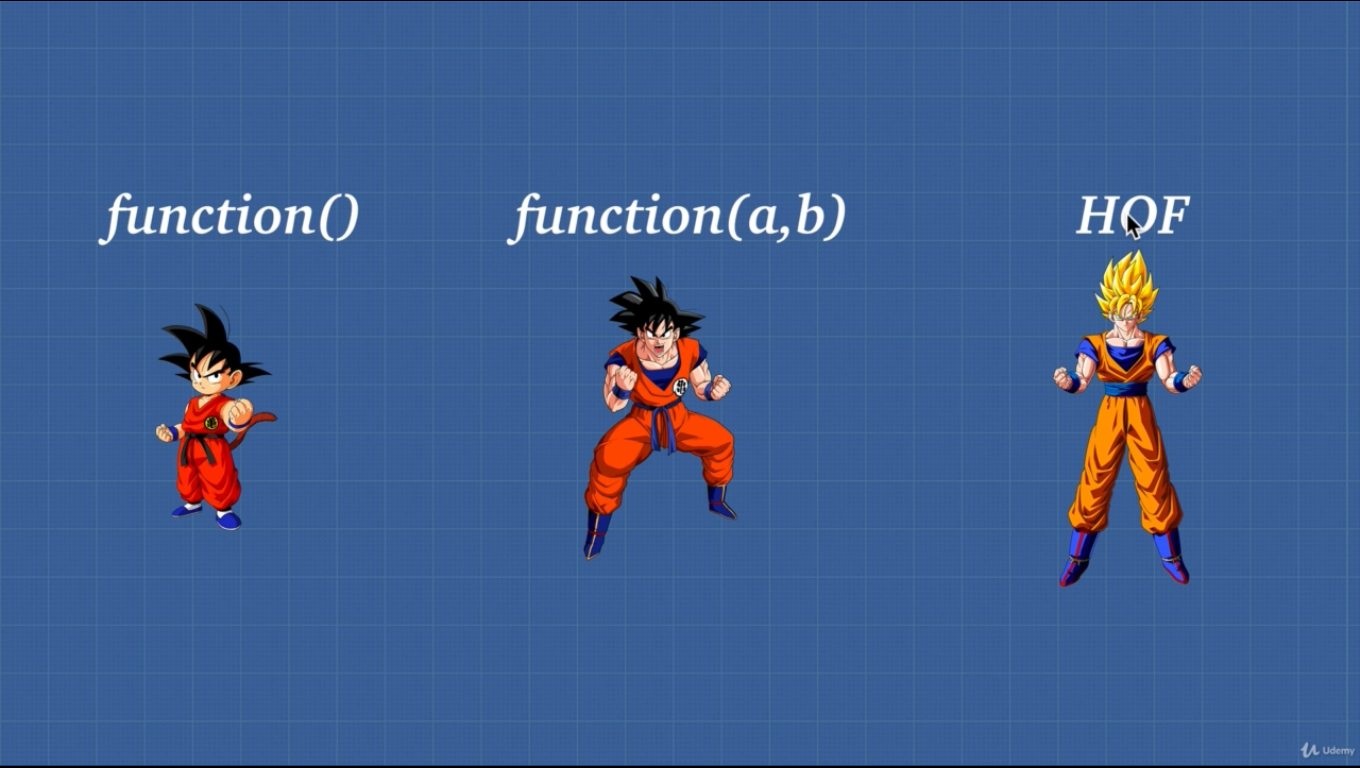
These behaviors makes functions as if they are JS types. Anything we can do with other types we can do with functions. This idea of first-class citizen property, makes JS a functional programming language.

4. Extra Bits Functions

Careful using function:

* Don’t initialize function inside of a loop, instead we need to move them somewhere else.
* Is variable accessible or not. To avoid this, we can check if variable is present or give a default param (ES6).

5. Higher Order Functions (HOF)



* HOF is a function that can take another function as an argument or a function that returns another function.

Three types of function use cases:

1. A normal function that is, every time we tell it what to do and it’s not dry.
2. Function with param that we tell the function what data to use later on. It can also be not dry.
3. HOF which can do both of these things and takes and/or returns another function.

6. Exercise Higher Order Functions

* Arrow function is cleaner visually.

7. Closures

Two things for closures:

1. In JS functions are first class citizen (HOF).
2. Lexical scope, where the code is written.

Closures is combination of function and the lexical environment from where it was declared.

* Closures is one of the most confusing things but important topic.
* When we execute a function, it is popped off from the stack and its variable environment is cleared.
  + Even though the function is popped off form the stack and variable environment is cleared, since another function is referencing some variables of the cleared function, closures takes the variables and stores them in the heap, so that the referencing function can have the variables that it needs.
  + JS engine will keep anything that is being refenced, inside of the closures.
* Closures are also called lexical scoping.
  + Lexical, where it is written.
  + Scoping, what variable it has access to.
* JS is lexically/statically scoped.

8. Exercise Closures

* If we use setTimeout() closures will work.
* Even if we declare variable after the setTimeout() closures will have the variable for us.