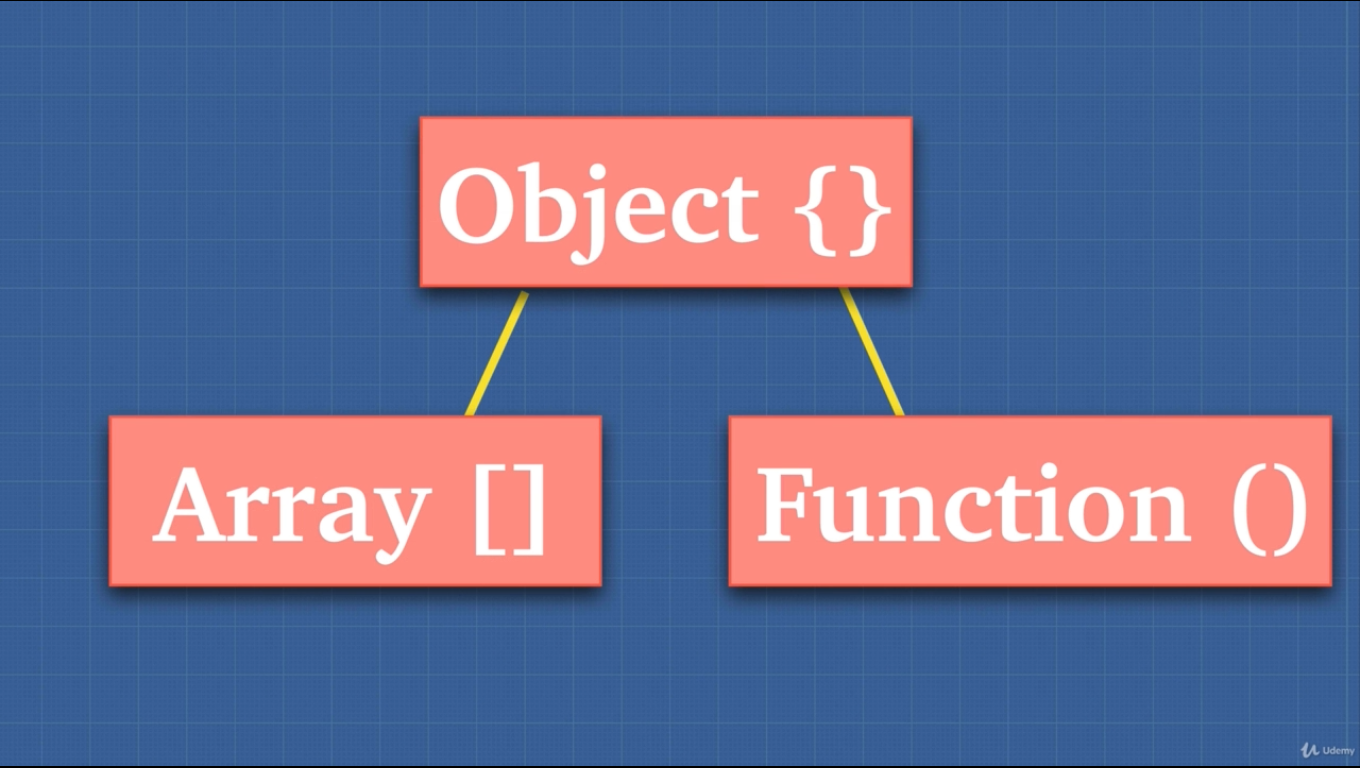
2. Javascript Types

Types in JS can get a little bit tricky.

There are 7 types of data in JS:

1. Integer.
2. Boolean.
3. String.
4. Undefined.
5. Null.
6. Symbol.
7. Object.

* Symbols are used for object properties so that the object properties are unique.
* **Undefined** is the absence of definition.
  + Function doesn’t return anything.
  + There is a variable there but nothing in it.
  + Missing property of an object.
* **Null** is absence of value.
* In JS 2 types of data type:
  + *Primitive data type*: Data only represents a single value.
    1. Integer.
    2. Boolean.
    3. String.
    4. Undefined.
    5. Null.
    6. Symbol.
  + *Non-primitive data type*: Doesn’t contain the actual value directly.
    1. Object.
    2. Array.
    3. Function.



* Object doesn’t contain the actual value directly. It has a reference to its value somewhere to the memory.
* A common term that “Everything in JS is an object”. Well… not really.
  + There are a lot of built in objects that we can use.
  + Primitive types have object wrapper around them 😐.

3. Array.isArray()

* Underneath the hood **array**
* is an object in JS.
* Then how to check if something is an array. Using **.isArray()** method

4. Pass by Value vs Pass by Reference

* Primitive types are passed by value. When we do **b = a** we copy what was in **a**.
* **Objects** are passed by reference. When we do **b = a** we simply give the address.
* Difference:
  + Copying primitive types takes up memory.
  + Object doesn’t. Accidently we may change the object.
* **Array** is an object.
  + To copy or clone array we can use **.concate()** method.
* To clone an object, we can use:
  + Shallow clone (only clone the first layer).
    - **Object.assign()** method.
    - Spread operator **copy = {…obj}**.
  + Deep clone:
    - **JSON.parse(JSON.stringify(obj))** This has performance issue when it’s very deep.
* Each object gets passed by reference, even nested object.

7. Type Coercion

* **Type Coercion** means that the language converting a certain type to another type.
  + 1 == “1” true (Shouldn’t use this it’s confusing).
  + 1 === “1” false