Lecture-12

Promises

Prototypes

Coding Blocks - Kartik Mathur

Class Agenda

01	Promises		
02	Prototypes		
03	Bindings	04	-
05	-		
06	-	07	-

Promises

1

Functions

- Promise.race
- Promise.all

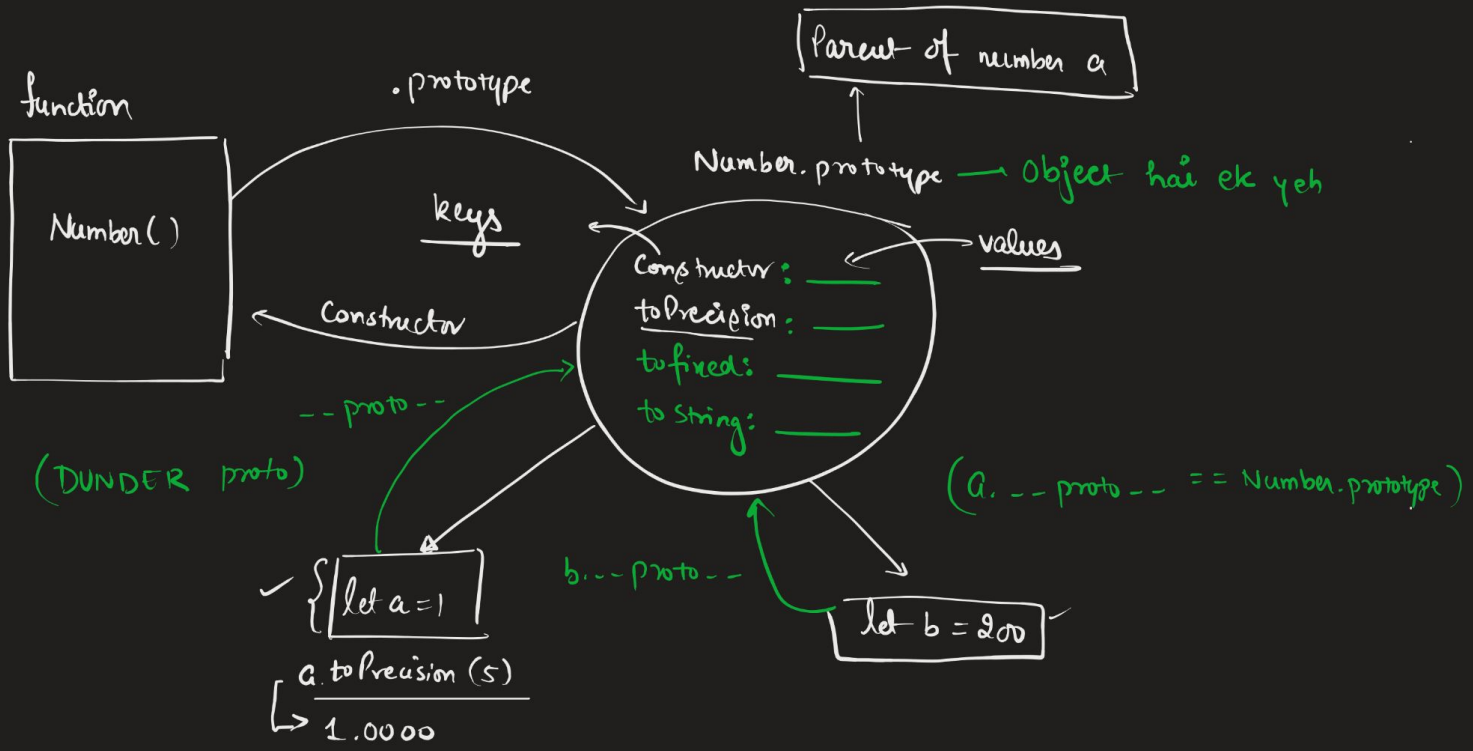
Promises

Prototypes

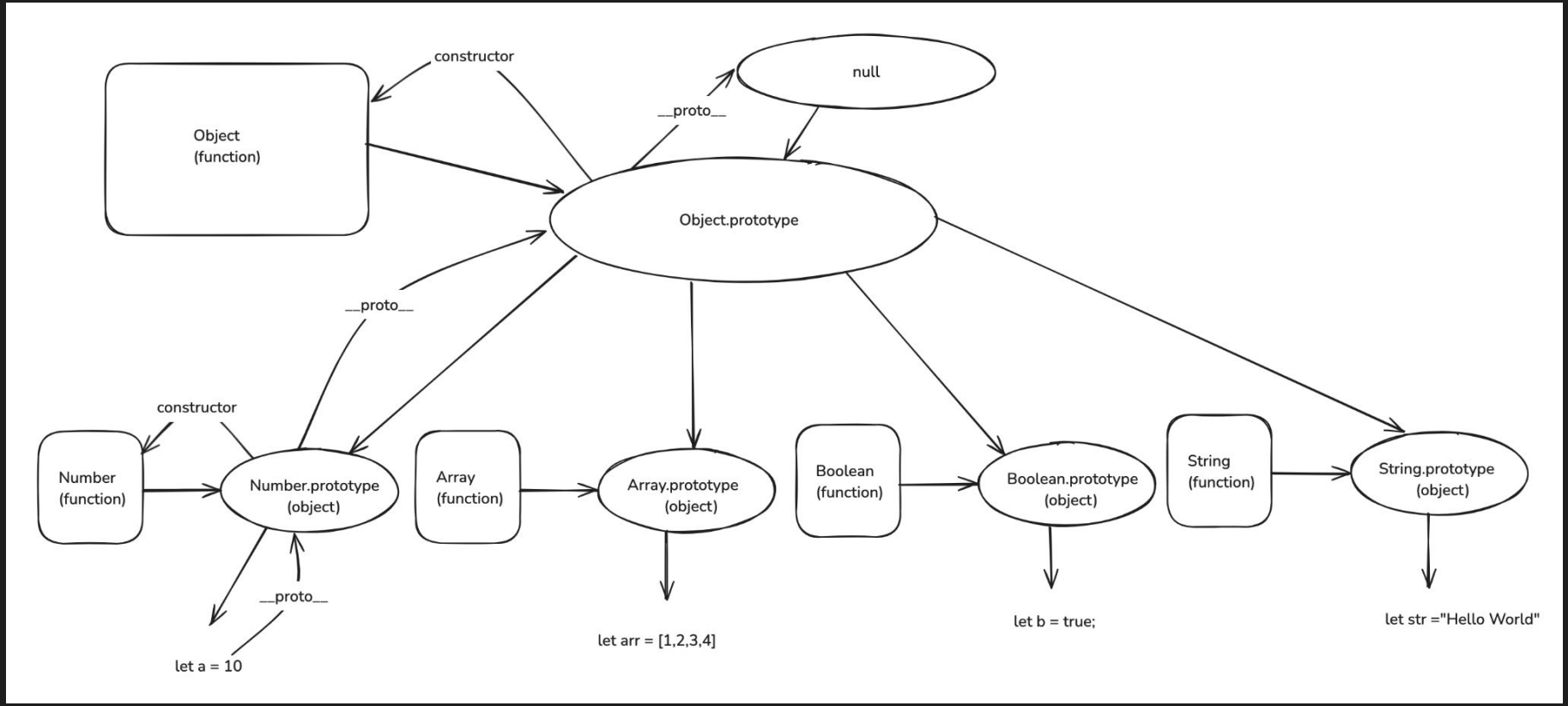
2

Prototypes are the mechanism by which JavaScript objects inherit features from one another.

Prototypes


Parent of number a
function
.prototype
Number.prototype → Object hai ek yeh
Number()
keys
values
Constructor
Constructor:
toPrecision:
toFixed:
toString:
__proto__
(DUNDER proto)
(a.__proto__ == Number.prototype)
b.__proto__
let a = 1
let b = 200
a.toPrecision(5)
1.0000

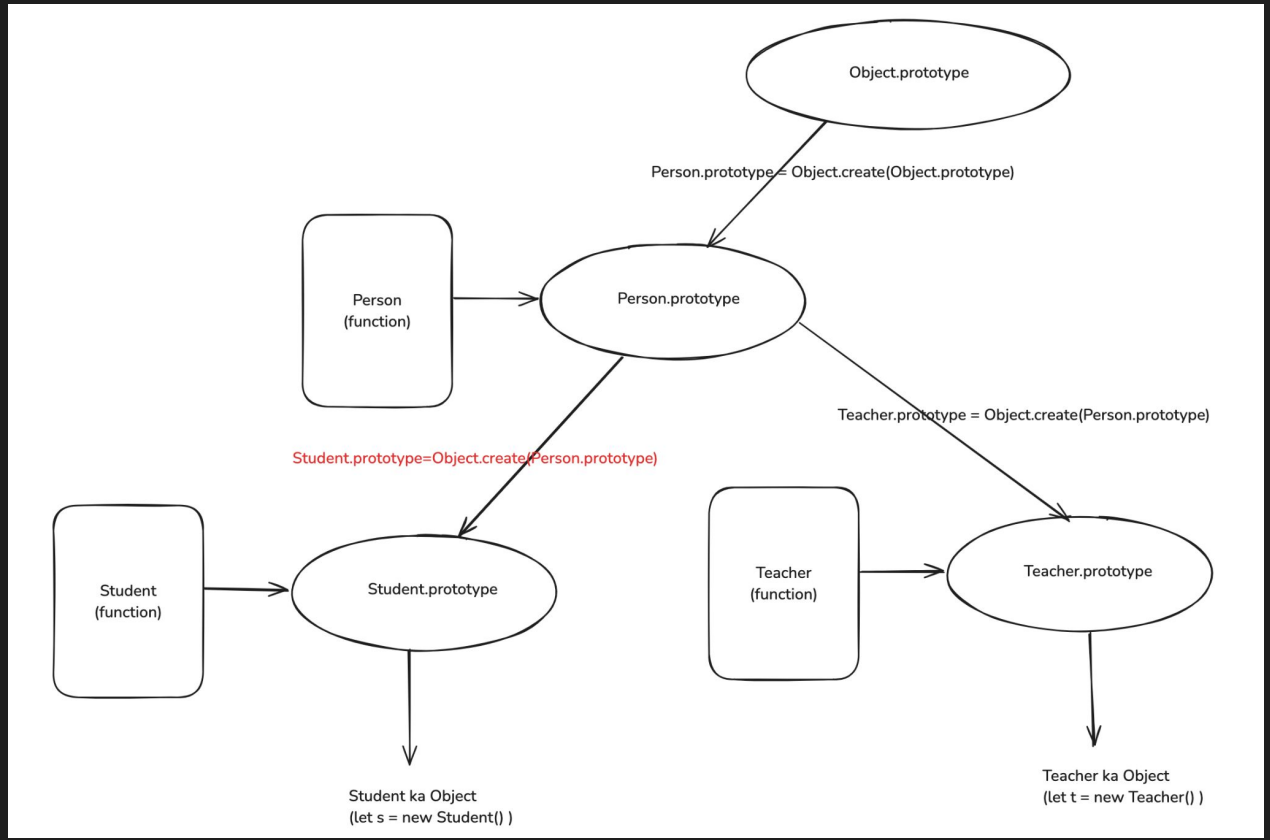
Prototypes


constructor
null
__proto__
Object
(function)
Object.prototype
__proto__
constructor
Number
(function)
Number.prototype
(object)
Array
(function)
Array.prototype
(object)
Boolean
(function)
Boolean.prototype
(object)
String
(function)
String.prototype
(object)
__proto__
let a = 10
let arr = [1,2,3,4]
let b = true;
let str ="Hello World"

Prototypes

Let’s see: Object.create() function usage

Let’s use this for inheritance



Let’s use this for inheritance

Bindings

Binding something in JavaScript means recording that identifier in a specific `Environment Record`. Each `Environment Record` is related to a specific `Execution Context` - and that binds the identifier (variable or function name) to the `this` keyword for that execution context.

There are four type of Bindings:

1. Default binding
2. Implicit Binding: Dot operator binding
3. Explicit Binding: call, apply, bind
4. new keyWord

Implicit binding

This is done with the help of creating an object.

Bindings

Explicit Binding

- Apply: Change the context and call the function immediately.
- Call: Change the context and call the function immediately.
- Bind” Change the context and call the function later.

Bindings

New Bindings

Using new keyword we can create a completely new “Execution-Context”

Bindings

Default Bindings

In whichever Execution context we are currently, it will remain the same.

Bindings

CLASS Syntax

JS is not OOPS based, it simply has classes to handle the prototype chaining part.

Classes