- Object Creation Functions
- Inheritance
- Properties
- Methods
- Instantiation

OBJECTS IN JAVASCRIPT

Everything is an object.

(Well, almost everything)

PRIMITIVES

- Numbers
- Strings
- Booleans
- Undefined
- Null

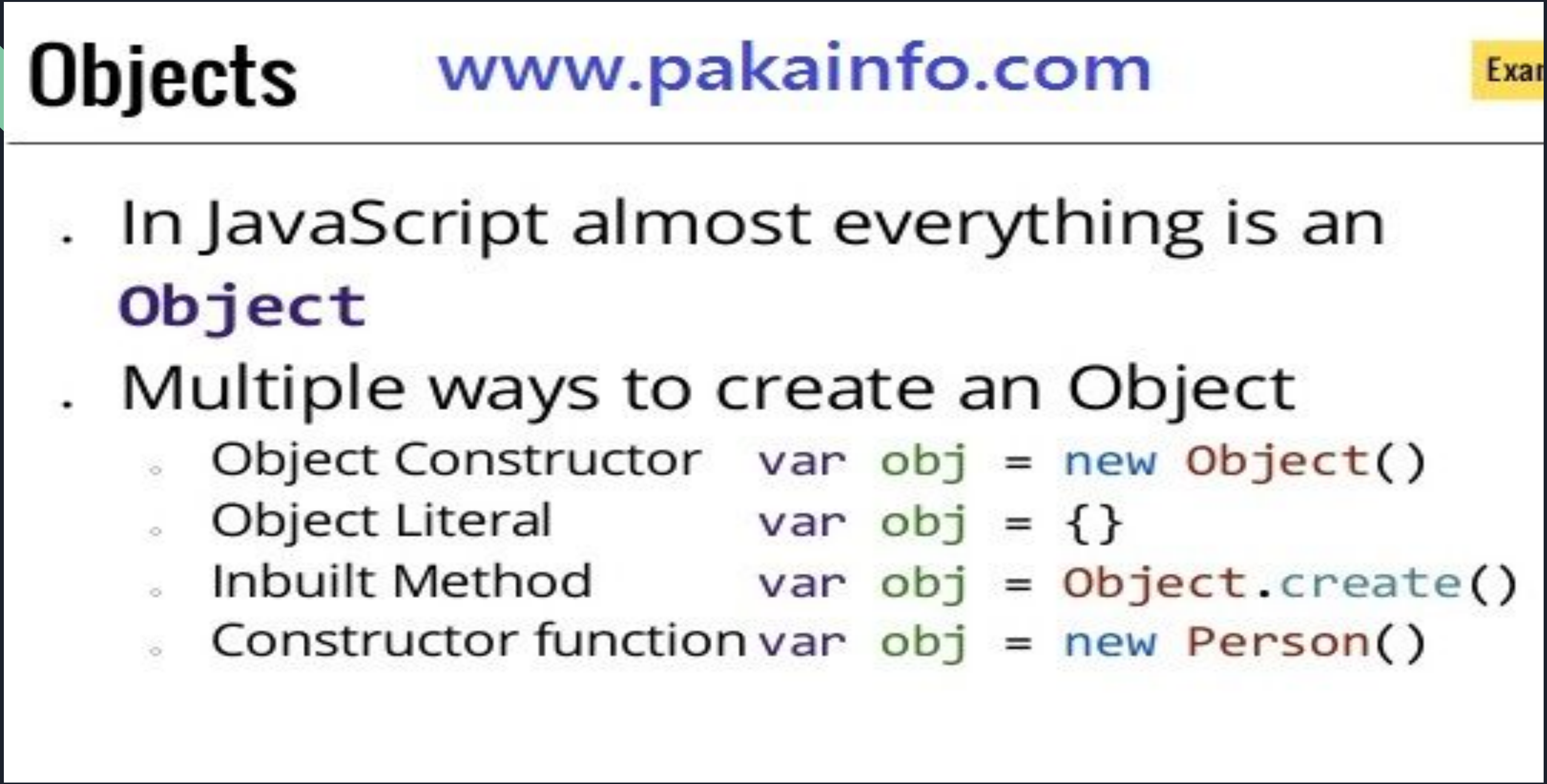
EVERYTHING ELSE ...

- Arrays
- Functions
- Objects
- Dates
- Wrappers for Numbers, Strings, Booleans

... IS AN OBJECT

JavaScript Objects

- JavaScript is an *object-based* programming language
  - Programmer focuses on objects needed to solve a problem
- An *object* represents a real-world entity
- An *object* contains properties and methods
  - *Properties* are attributes that distinguish one object from another
  - *Methods* are functions or actions you want an object perform



Objects

www.pakainfo.com

Exar

- In JavaScript almost everything is an **Object**
- Multiple ways to create an Object

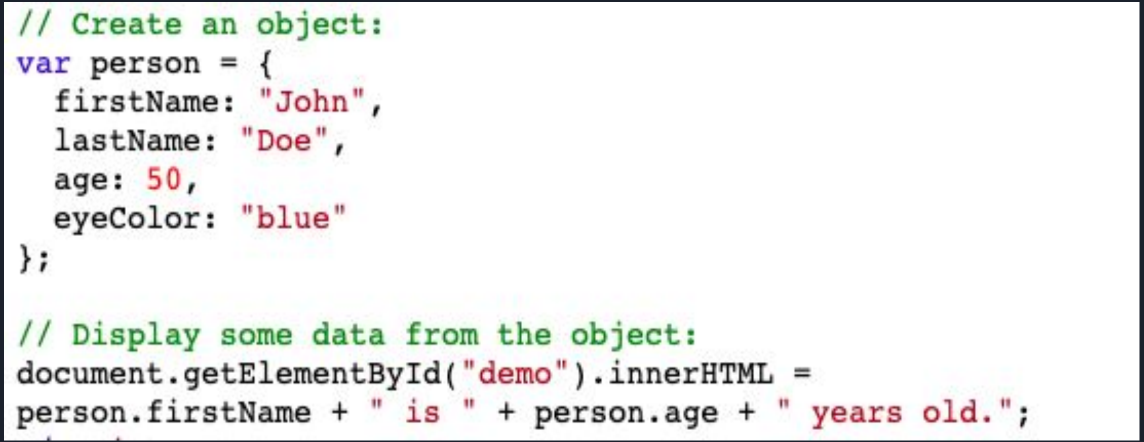
Object Constructor	`var obj = new Object()`
Object Literal	`var obj = {}`
Inbuilt Method	`var obj = Object.create()`
Constructor function	`var obj = new Person()`

Object	Properties	Methods
	car.name = Fiat	car.start()
	car.model = 500	car.drive()
	car.weight = 850kg	car.brake()
	car.color = white	car.stop()

All cars have the same **properties**, but the property **values** differ from car to car.

All cars have the same **methods**, but the methods are performed **at different times**.

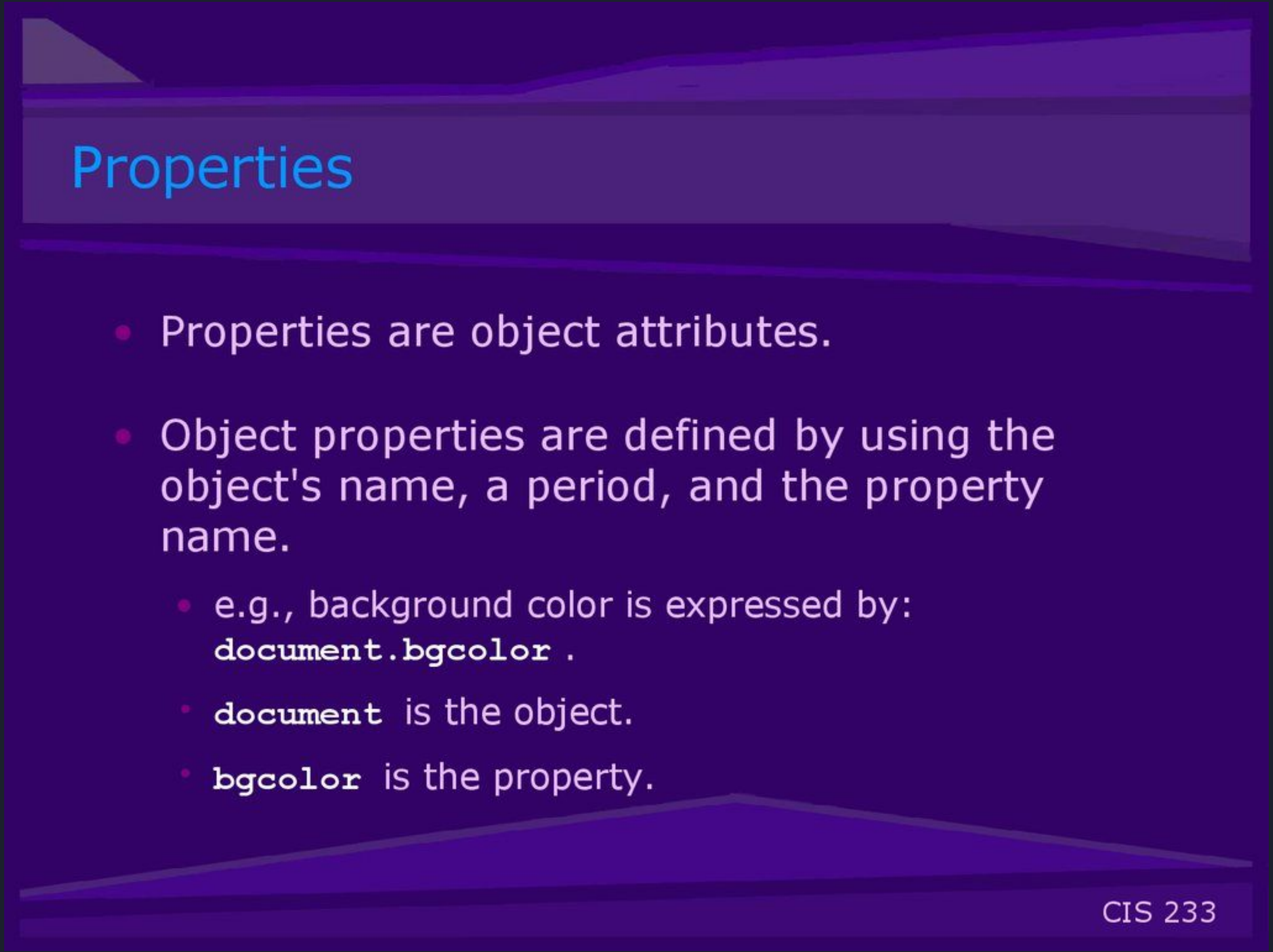
How to create an Object



```
// Create an object:
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};

// Display some data from the object:
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";
```

Output: John is 50 years old



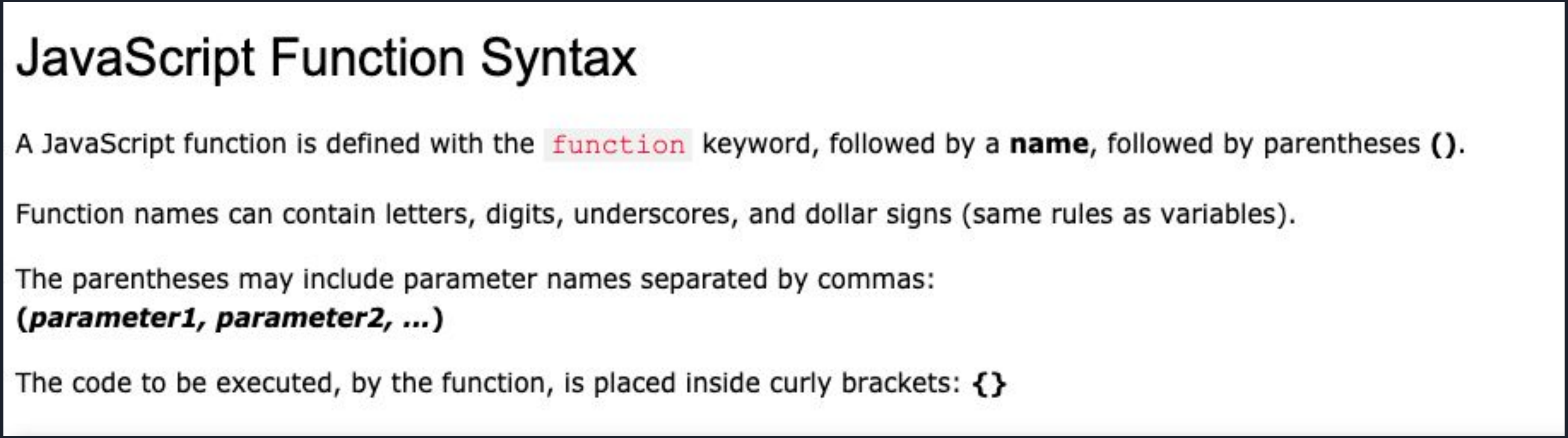
Properties

- Properties are object attributes.
- Object properties are defined by using the object's name, a period, and the property name.
  - e.g., background color is expressed by: **`document.bgcolor`** .
  - **`document`** is the object.
  - **`bgcolor`** is the property.

Object	Properties

	Person.name = Pankaj
	Person.fname = Mr. Manoj
	Person.phNumber = 2212
	Person.Address = INDIA
	Person.getInfor() = All info

Functions



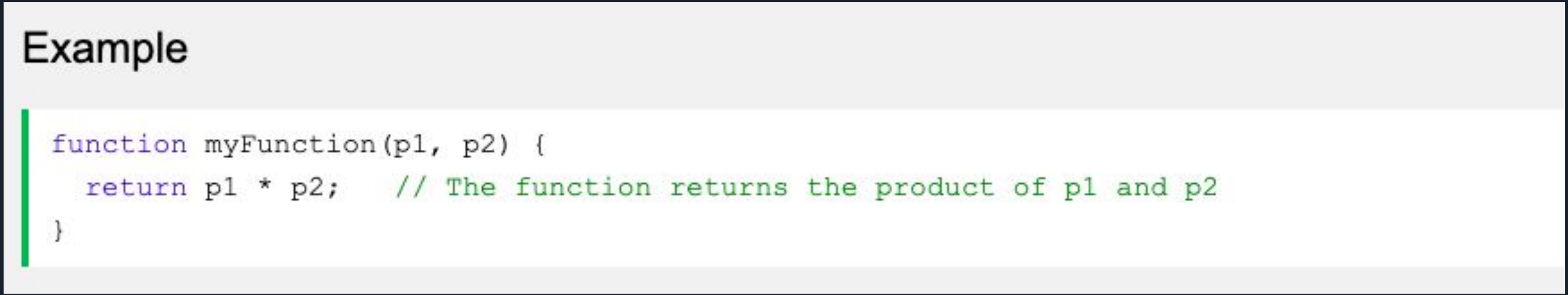
JavaScript Function Syntax

A JavaScript function is defined with the `function` keyword, followed by a **name**, followed by parentheses **()**.

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas:
(*parameter1, parameter2, ...*)

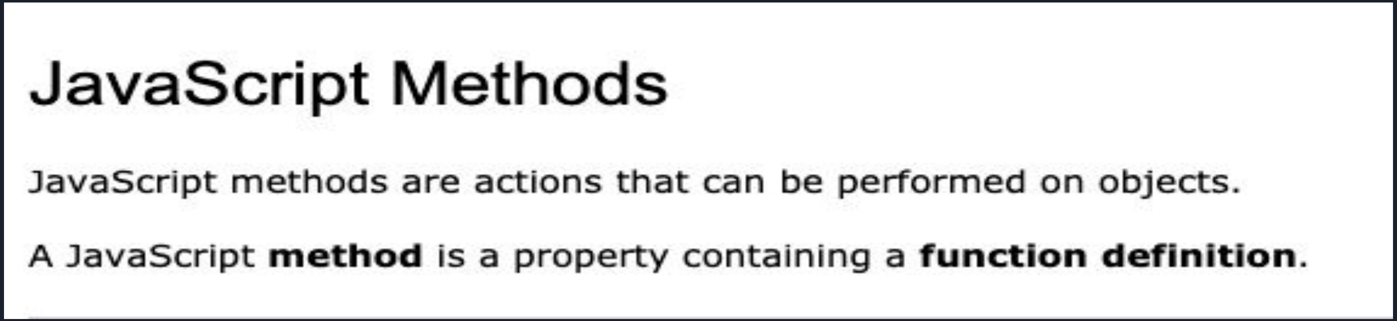
The code to be executed, by the function, is placed inside curly brackets: **{}**



Example

```
function myFunction(p1, p2) {
  return p1 * p2;   // The function returns the product of p1 and p2
}
```

Methods

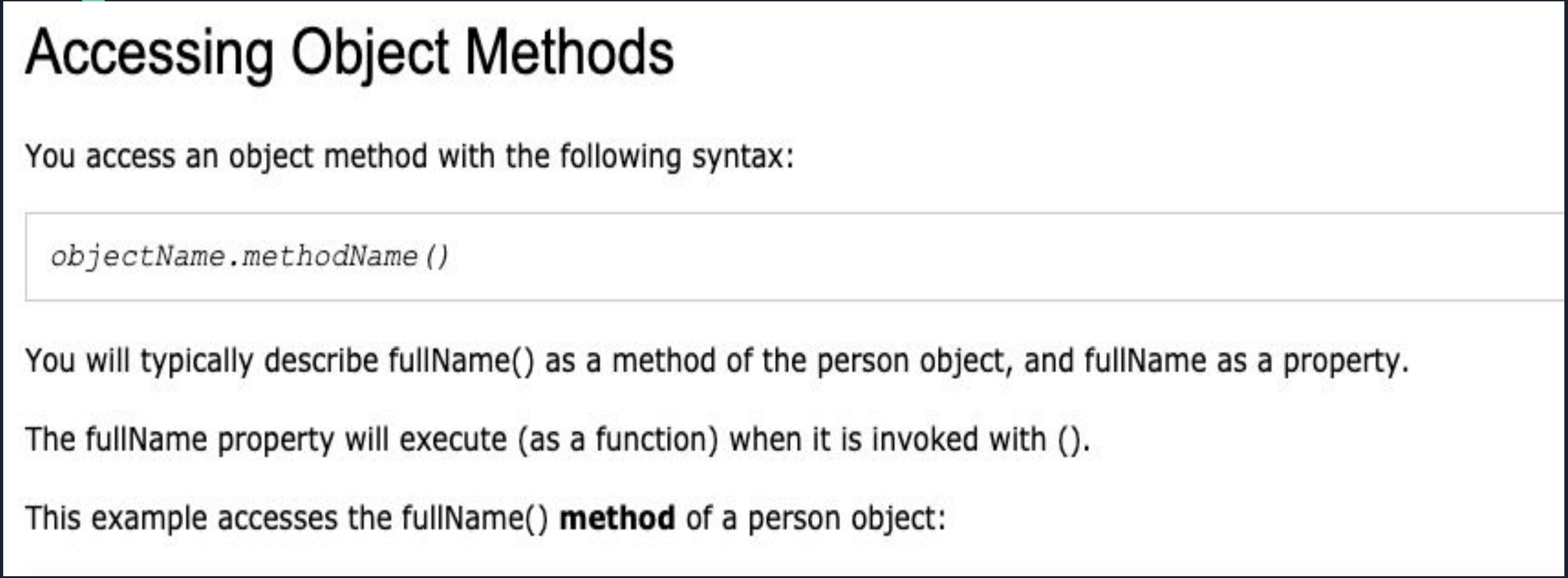


JavaScript Methods

JavaScript methods are actions that can be performed on objects.

A JavaScript **method** is a property containing a **function definition**.

Methods



Accessing Object Methods

You access an object method with the following syntax:

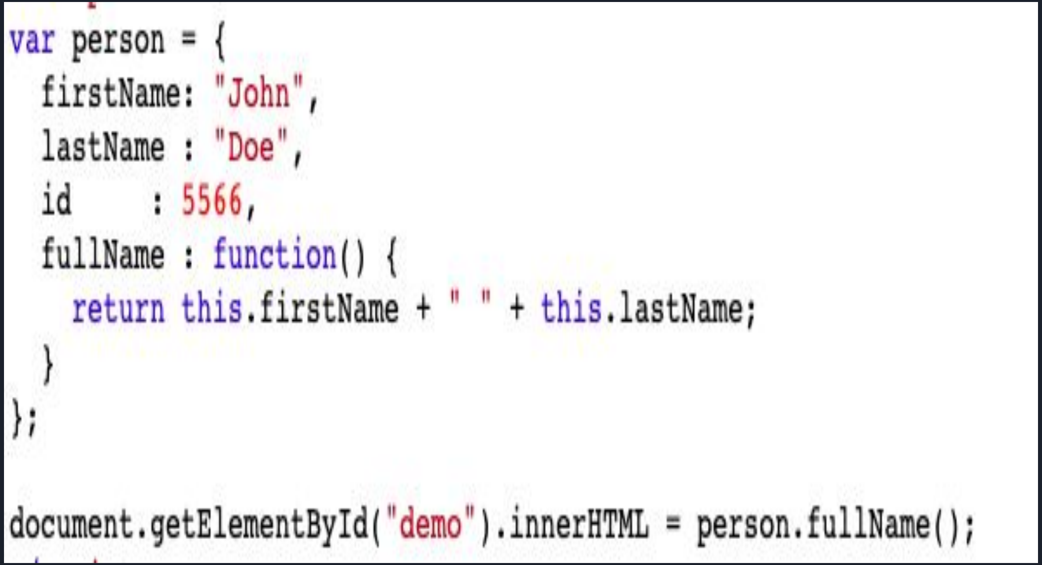
```
objectName.methodName()
```

You will typically describe fullName() as a method of the person object, and fullName as a property.

The fullName property will execute (as a function) when it is invoked with ().

This example accesses the fullName() **method** of a person object:

Example

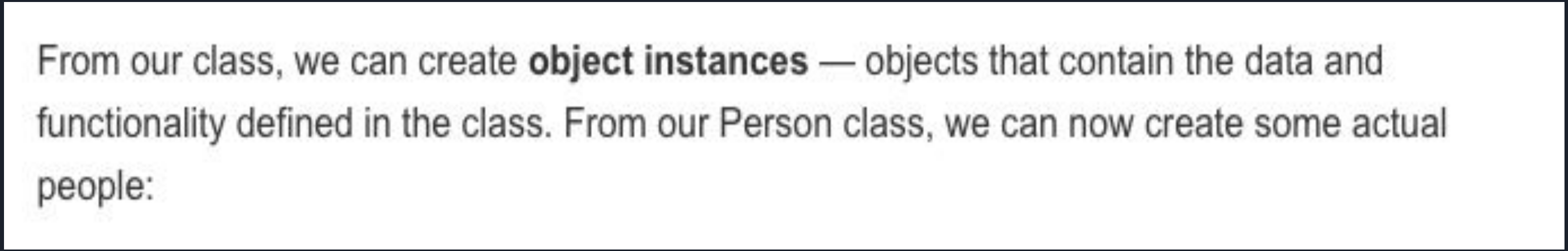


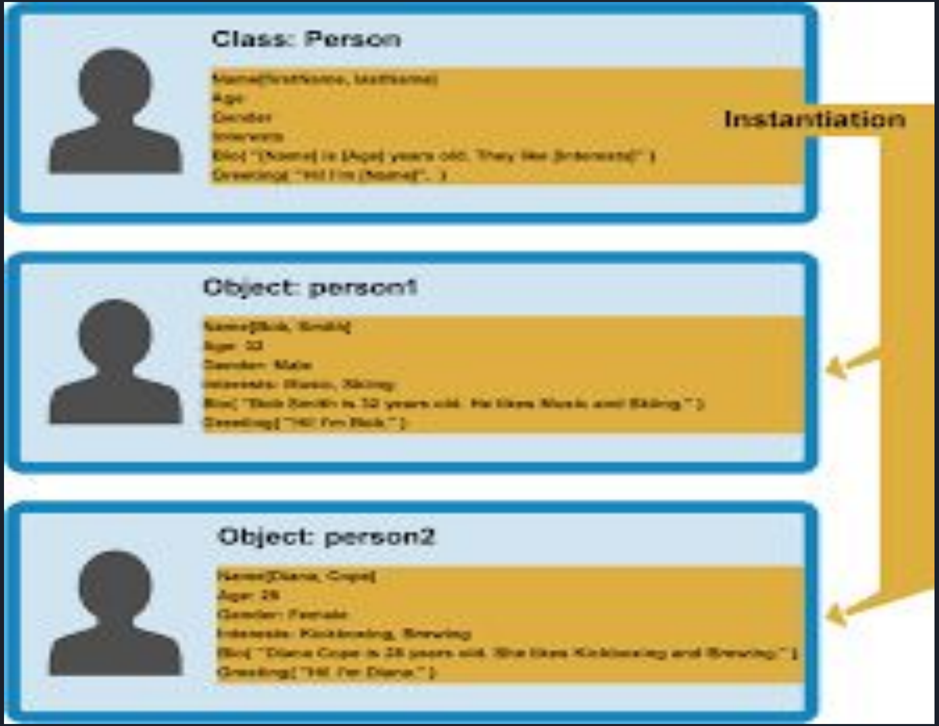
```
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};

document.getElementById("demo").innerHTML = person.fullName();
```

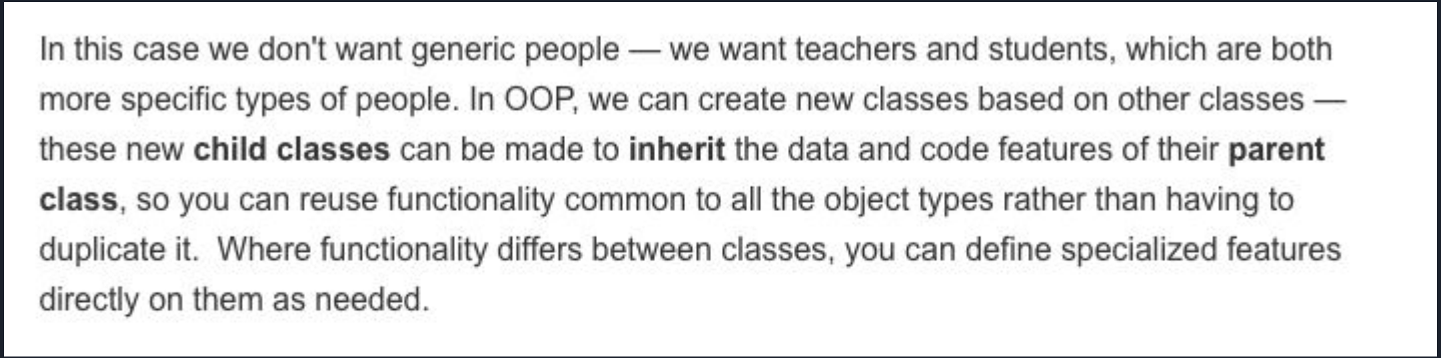
Output: John Doe

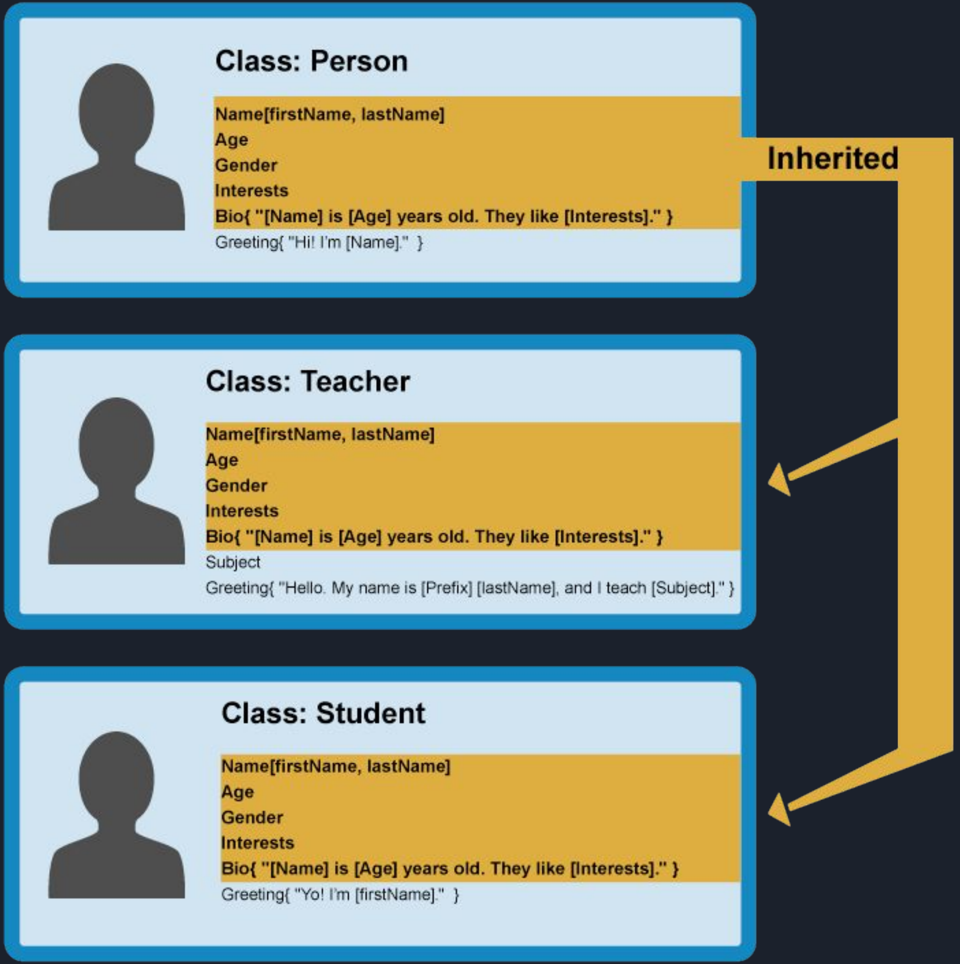
Instantiation


From our class, we can create **object instances** — objects that contain the data and functionality defined in the class. From our Person class, we can now create some actual people:


Class: Person
Name[firstName, lastName]
Age
Gender
Interests
Bio["[Name] is [Age] years old. They like [Interests]"]
Greeting["Hi! I'm [Name]".]
Instantiation
Object: person1
Name[Bob, Smith]
Age: 32
Gender: Male
Interests: Music, Skiing
Bio["Bob Smith is 32 years old. He likes Music and Skiing."]
Greeting["Hi! I'm Bob."]
Object: person2
Name[Diana, Cope]
Age: 28
Gender: Female
Interests: Kickboxing, Brewing
Bio["Diana Cope is 28 years old. She likes Kickboxing and Brewing."]
Greeting["Hi! I'm Diana."]

When an object instance is created from a class, the class's **constructor function** is run to create it. This process of creating an object instance from a class is called **instantiation** — the object instance is **instantiated** from the class.


In this case we don't want generic people — we want teachers and students, which are both more specific types of people. In OOP, we can create new classes based on other classes — these new **child classes** can be made to **inherit** the data and code features of their **parent class**, so you can reuse functionality common to all the object types rather than having to duplicate it. Where functionality differs between classes, you can define specialized features directly on them as needed.



Class: Person

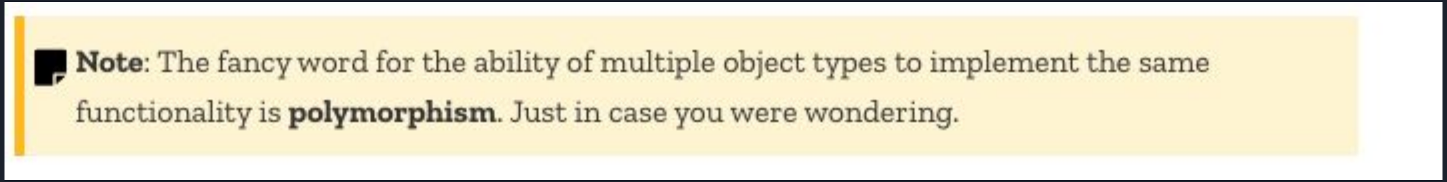
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Hi! I'm [Name]." }

Class: Teacher

Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Subject
Greeting{ "Hello. My name is [Prefix] [lastName], and I teach [Subject]." }

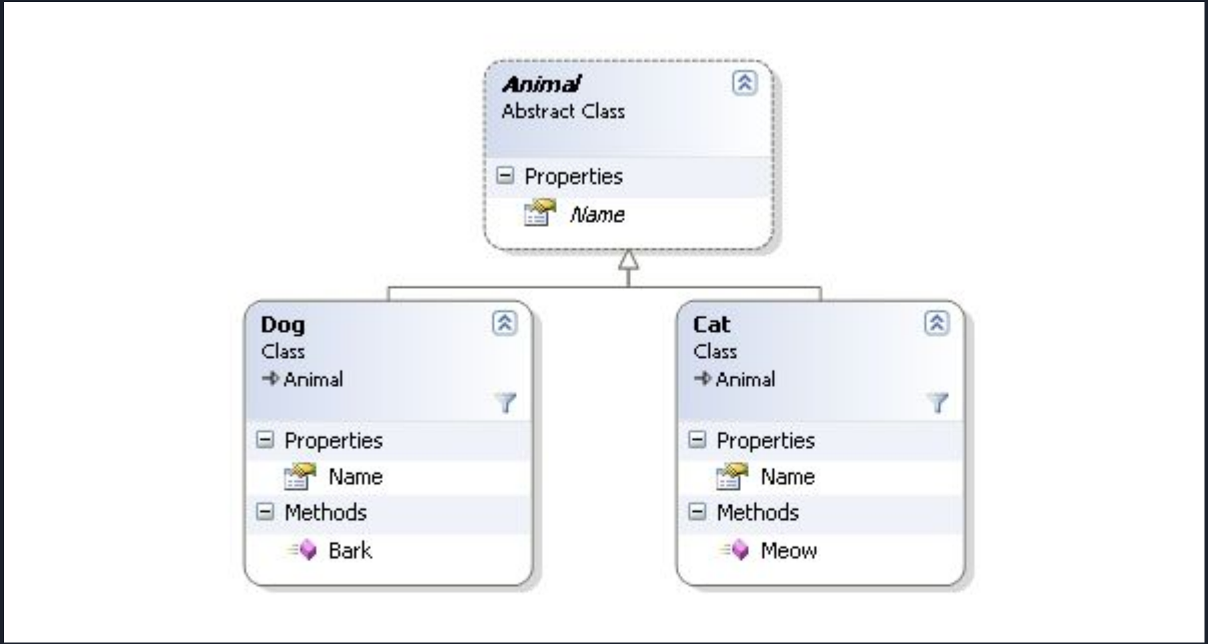
Class: Student

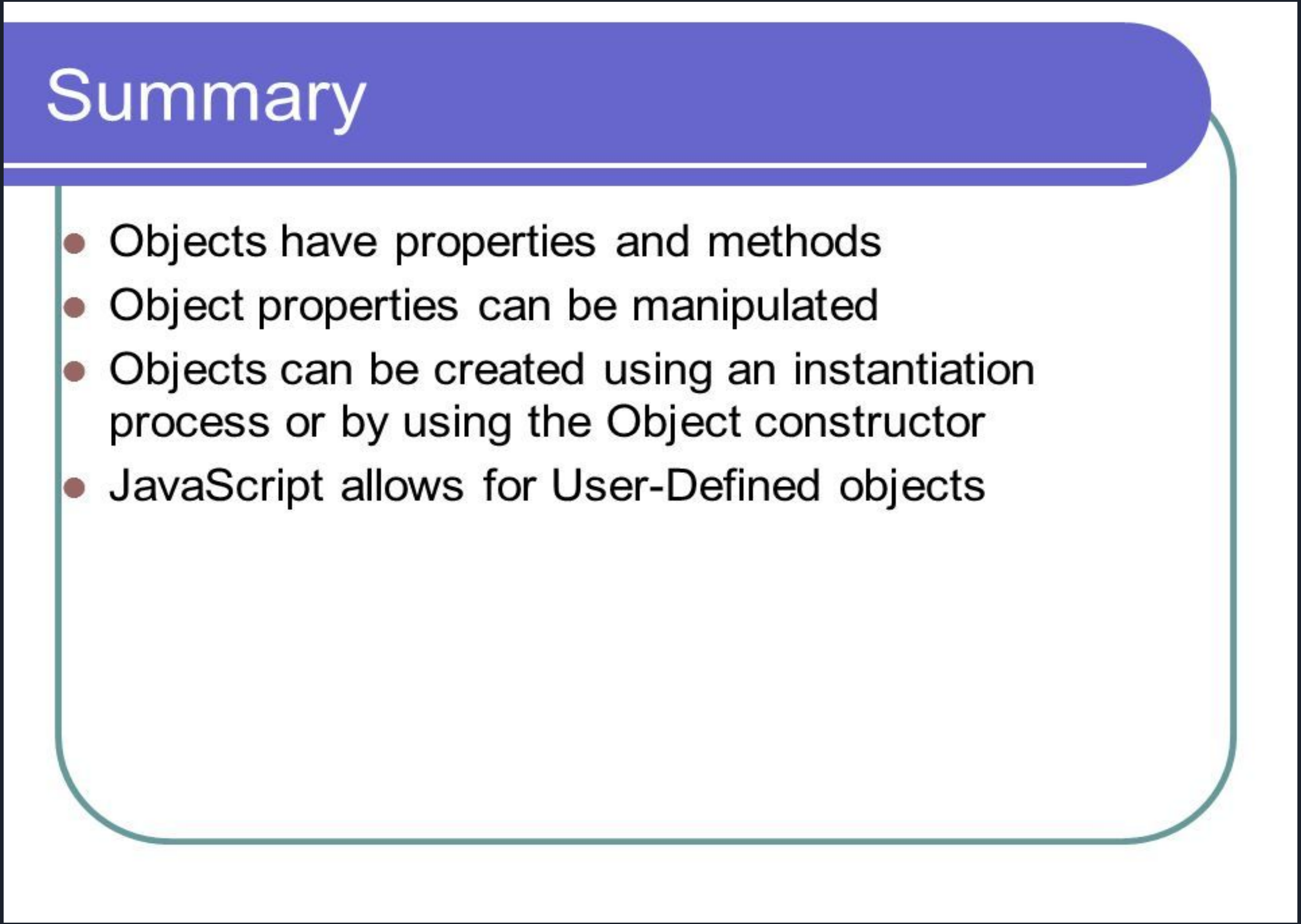
Name[firstName, lastName]
Age
Gender
Interests
Bio{ "[Name] is [Age] years old. They like [Interests]." }
Greeting{ "Yo! I'm [firstName]." }



Note: The fancy word for the ability of multiple object types to implement the same functionality is **polymorphism**. Just in case you were wondering.

Inheritance

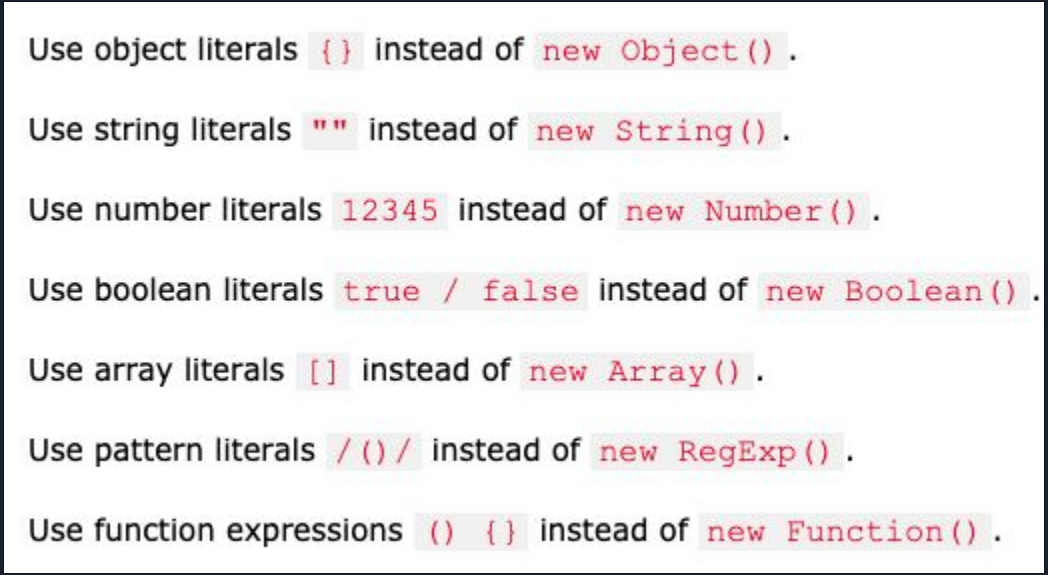




Summary

- Objects have properties and methods
- Object properties can be manipulated
- Objects can be created using an instantiation process or by using the Object constructor
- JavaScript allows for User-Defined objects

Recommendations



Use object literals `{}` instead of `new Object()`.

Use string literals `""` instead of `new String()`.

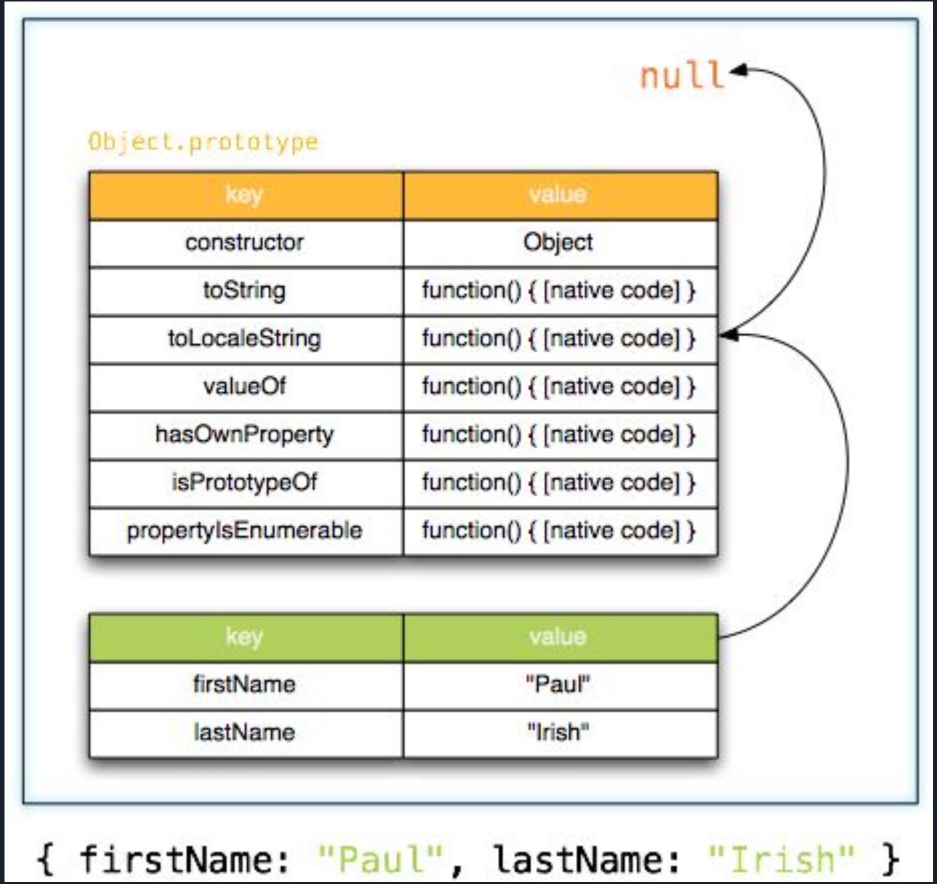
Use number literals `12345` instead of `new Number()`.

Use boolean literals `true / false` instead of `new Boolean()`.

Use array literals `[]` instead of `new Array()`.

Use pattern literals `/()/` instead of `new RegExp()`.

Use function expressions `() {}` instead of `new Function()`.



key	value
constructor	Object
toString	function() { [native code] }
toLocaleString	function() { [native code] }
valueOf	function() { [native code] }
hasOwnProperty	function() { [native code] }
isPrototypeOf	function() { [native code] }
propertyIsEnumerable	function() { [native code] }

key	value
firstName	"Paul"
lastName	"Irish"

```
{ firstName: "Paul", lastName: "Irish" }
```

References

https://www.w3schools.com/js/js_object_constructors.asp

https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS