# JavaScript

Switch statement

var job = 'teacher';

switch (job){

    case 'teacher' :

         console.log('he teacher');

         break;

    case ("driver") :

        console.log("he drives");

        break;

    default : console.log('he does nothing');

}

* Falsy values -> ‘ ’ ,‘null’, undefined,0 Nan

Arrays

* Arrays can take any type of data as below

var johnArray = ['john','30',1990]

**Methods**

johnArray.push()---- add a new element at the end of array

johnArray.pop() ---- removes one element at the end of array

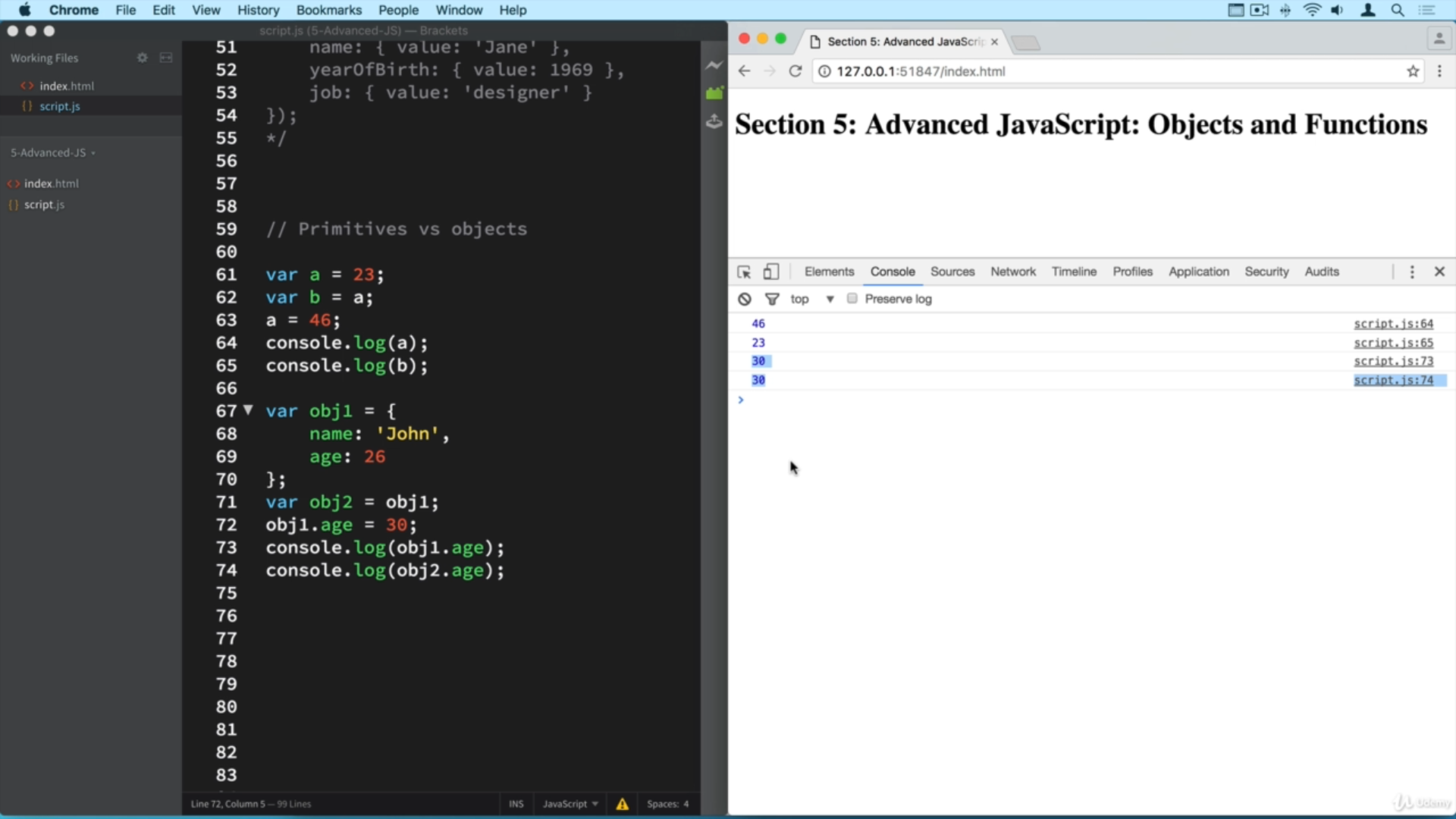
johnArray.unshift()---- add a new element at the start of array

johnArray.shift()---- removes an element at the start of array

Primitives & Objects

Primitives – hold the copy of data.

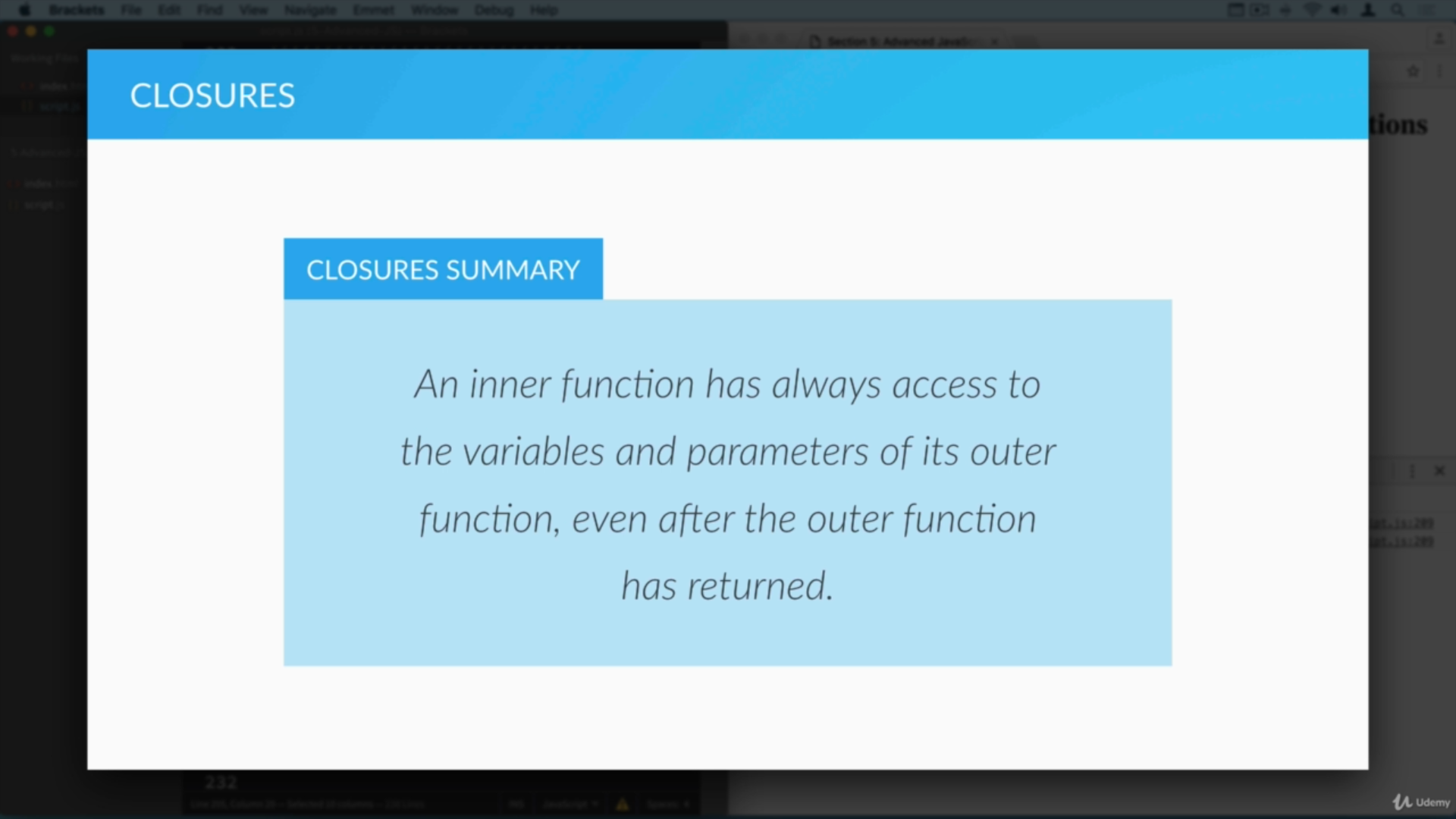
Objects – hold the reference to the object but no the real value



Passing Objects & Primitives to a function

When we pass primitive values to function, a copy of primitive is set as argument. While when objects are passed as arguments, we pass a reference of obj to functions as arguments.

Closures





Prototype

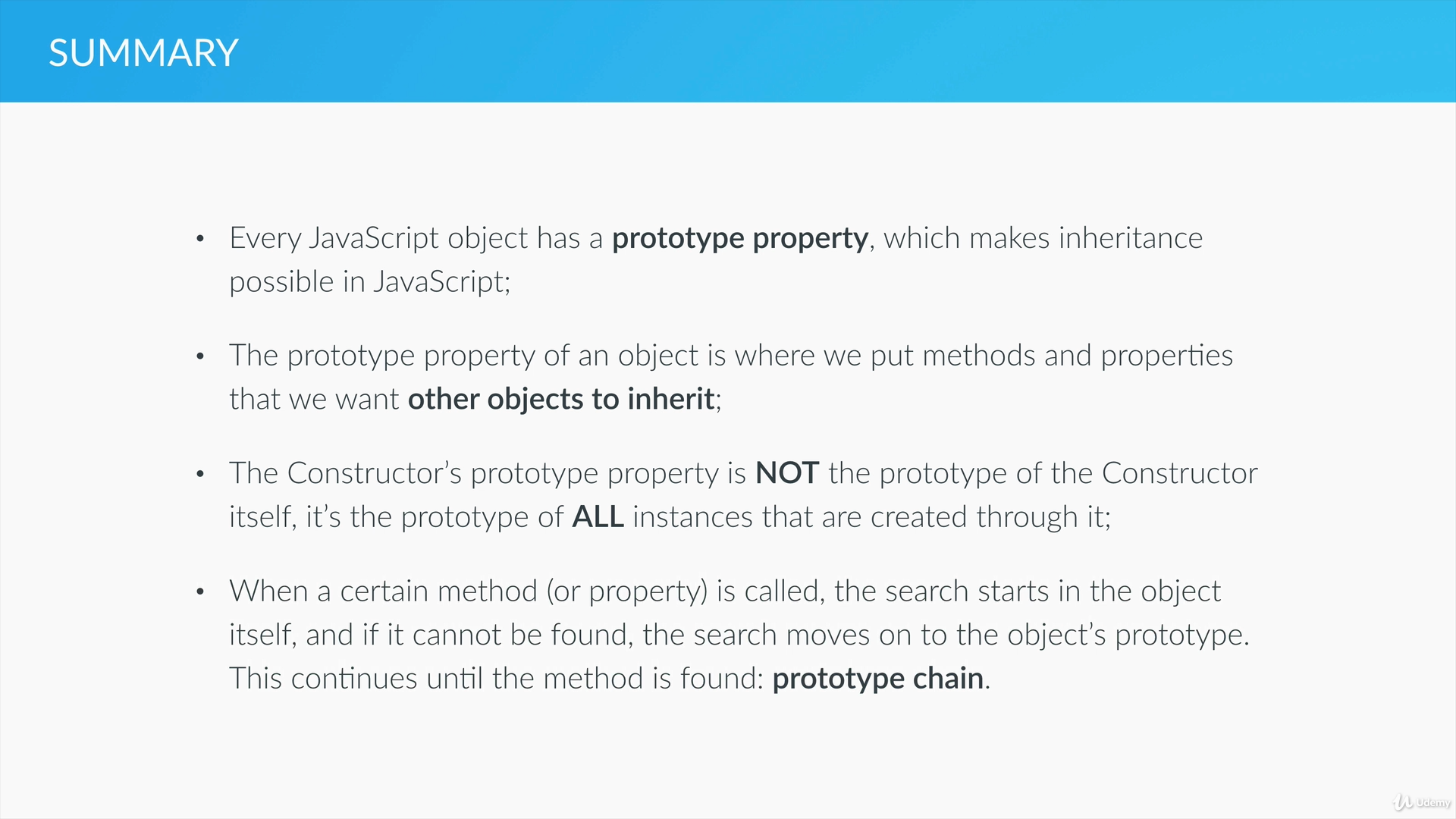
Inheritance in JavaScript works by some **property** called *prototype*

* *Prototype* property of an object is where you put properties and methods for other objects to inhertit.
* **Prototype** is not just prototype of an object itself but it belongs to all the instances created using objects blueprint

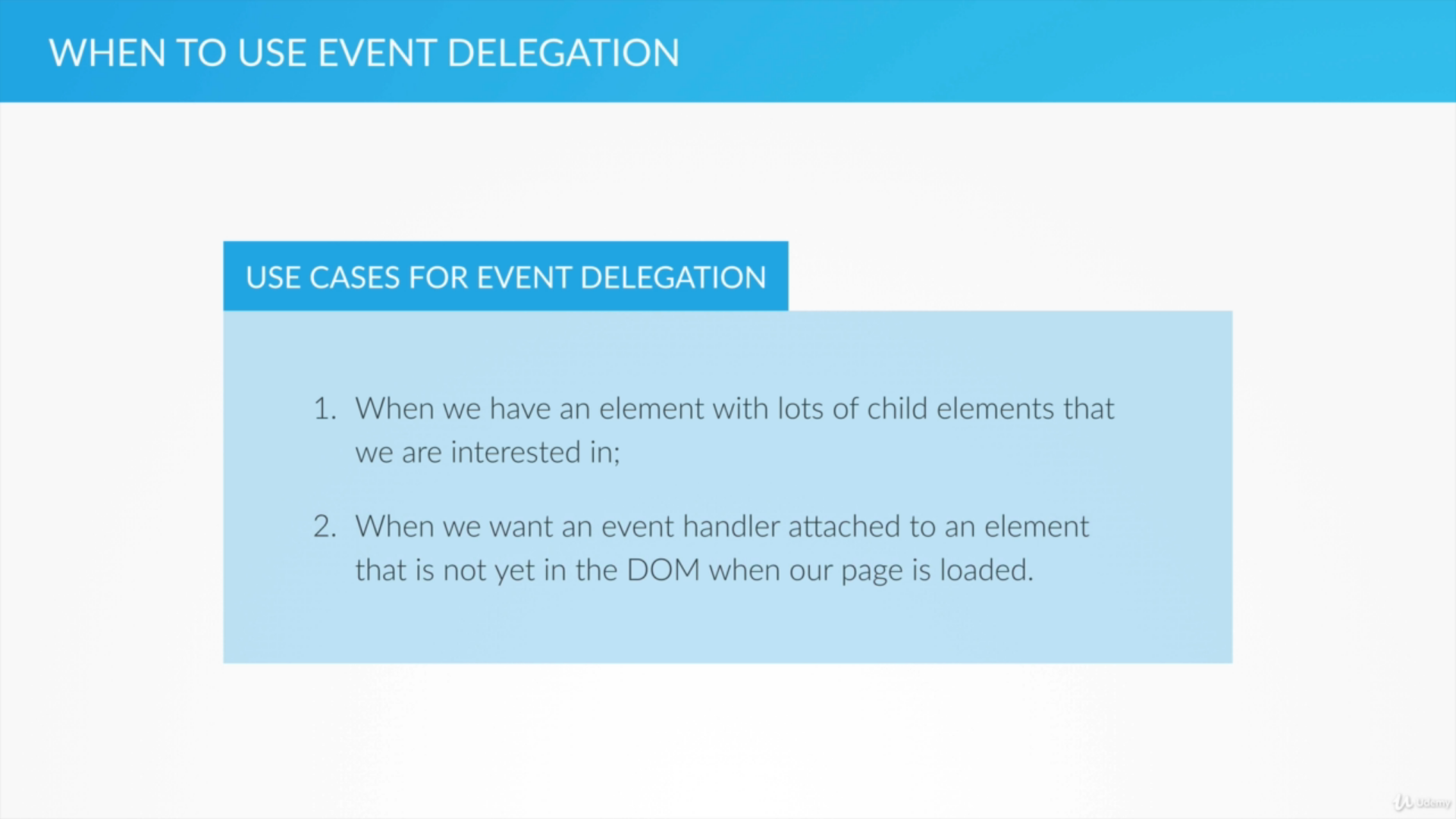
Objects can be created using

1. Function constructors
2. Object.create()

Main difference in creating objects using above 2 ways is that if we create objects using **1** the new instances created inherit from the constructors prototype property. While the **2** enables us to inherit from the first argument of Object.create(prototype \_\_proto\_\_)



Event Delegation



Arrow Functions

Arrow functions don’t have ‘**this** ’ keyword. They just use the ‘**this**’ of the function in which they are part of.

In method call, **this** refers to current objects properties & in case of regular function call, **this** refers to document object

In forEach, map loops we cant use **continue & break.** These key words can be used only with traditional for loop as of ES5

But in ES6, with **forOf** loop both **continue** & **break** can be used

* REST & SPREAD – both used to convert the arrays into separate objects & vice versa
* SPREAD is used in function call where as REST is used in function declaration which has arbitrary no of arguments.

Default Parameters

Function constructorName(arg1,arg2 = ‘Lorel Ipsum’){

…..

…..

}

When we create a new instance using ***constructorName*** we can send as many params as needed – some times even less no of params

So when we send less # of params, in order to assign a default value to them we use above default parameters way in ES6 rather than manually specifying it in a *if /else* block

Maps

In objects , only strings can be used as keys whereas in maps – we can use primitives, objects, functions as keys



Objects are not iterable, where as maps are

Promises