reference:

1. code step by step youtube channel(18:13 video duration)

2. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures> (wow definition some parts)

3.<https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-closure-b2f0d2152b36> ( wow definition some parts)

Q. what is closure in javascript & how does it work & example & interview questions?

javascript closure 90 dependent on lexical scope.

What is lexical scope ?

A function scope’s ability to access variables from the parent scope is known as lexical scope.

Example:

function init() {

var name = "Mozilla"; // name is a local variable created by init

function displayName() {

// displayName() is the inner function, that forms the closure

console.log(name); // use variable declared in the parent function

}

displayName();

}

init();

init() creates a local variable called name and a function called displayName(). The displayName() function is an inner function that is defined inside init() and is available only within the body of the init() function. Note that the displayName() function has no local variables of its own. However, since inner functions have access to the variables of outer functions, displayName() can access the variable name declared in the parent function, init().

.............................................so on......................

let a = 100;

//start laxical scope of a

function abc(){

let b = 10;

//start laxical scope of b

console.log(a)

function def(){

console.log(a)

}

def()

console.log(b)//end laxical scope of b

}

abc()

console.log(a)//end laxical scope of a

What is a Closure?

A closure is the combination of a function bundled together (enclosed) with references to its surrounding state (the lexical environment). In other words, a closure gives you access to an outer function’s scope from an inner function. In JavaScript, closures are created every time a function is created, at function creation time.

To use a closure, define a function inside another function and expose it. To expose a function, return it or pass it to another function.

The inner function will have access to the variables in the outer function scope, even after the outer function has returned.

// Outer function

function outer() {

function create\_Closure(val) {

return function () {

return val;

}

}

let arr = [];

let i;

for (i = 0; i < 4; i++) {

arr[i] = create\_Closure(i);

}

return arr;

}

let get\_arr = outer();

console.log(get\_arr[0]());

console.log(get\_arr[1]());

console.log(get\_arr[2]());

console.log(get\_arr[3]());

//output:

0

1

2

3

JavaScript prototype:

reference:

1. https://www.youtube.com/watch?v=6b601MrVEi0

example\_1:

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 getFullname: function(){

 return this.name + " " + this.lastname

 }

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 getFullname: function(){

 return this.name + " " + this.lastname

 }

}

console.warn(student.getFullname()) // Swarnali Ghosh

console.warn(teacher.getFullname()) // Soma Ghosh

example\_2:

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 birth: 1998,

 getFullname: function(){

 return this.name + " " + this.lastname

 },

 getAge: function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 birth: 1974,

 getFullname: function(){

 return this.name + " " + this.lastname

 },

 getAge: function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

}

console.warn(student.getFullname()) // Swarnali Ghosh

console.warn(teacher.getFullname()) // Soma Ghosh

console.warn(teacher.getAge()) // 49

console.warn(student.getAge()) // 25

example\_3:

let users = { // common function

 getFullname: function(){

 return this.name + " " + this.lastname

 },

 getAge: function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

}

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 birth: 1998,

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 birth: 1974,

}

teacher.\_\_proto\_\_ = users; // it’s not a proper way to use prototype

student.\_\_proto\_\_ = users; // it’s not a proper way to use prototype

console.warn(student.getFullname()) // Swarnali Ghosh

console.warn(teacher.getFullname()) // Soma Ghosh

console.warn(teacher.getAge()) // 49

console.warn(student.getAge()) // 25

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

example\_4:

let users = { // common function

 getFullname: function(){

 return this.name + " " + this.lastname

 },

 getAge: function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

}

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 birth: 1998,

 getAge: users.getAge // it’s not a proper way to use prototype

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 birth: 1974,

 getAge: users.getAge // it’s not a proper way to use prototype

}

console.warn(teacher.getAge()) // 49

console.warn(student.getAge()) // 25

Object.prototype.getFullname = function(){

 return this.name + " " + this.lastname

 }

Object.prototype.getAge = function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 birth: 1998

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 birth: 1974

}

console.log(teacher.getFullname()) // Soma Ghosh

console.log(teacher.getAge()) // 49

console.log(student.getAge()) // 25

//prototype use with object

String.prototype.getCommonString = "is the employee of r.a.tech company"

Object.prototype.getFullname = function(){

 return this.name + " " + this.lastname

 }

Object.prototype.getAge = function(){

 let age = new Date().getFullYear()-this.birth;

 return age

 }

let student = {

 name: 'Swarnali',

 lastname: 'Ghosh',

 birth: 1998

}

let teacher = {

 name: 'Soma',

 lastname: 'Ghosh',

 birth: 1974

}

console.log(teacher.getFullname()) // Soma Ghosh

console.log(teacher.getAge()) // 49

console.log(student.getAge()) // 25

let fullname = student.getFullname()

// prototype use with string

console.log(fullname, student.name.getCommonString)