Ctrl + ñ: terminal (visual code)

Lección 1

Visual code: texto plano: ctrl+k +m para cambiarlo

Entrar a carpeta (comand git hub)

cd '/c/Users/quique don wolf/Desktop/Curso.Prep.Henry'

CLI: command line interface (terminal para ejecutar pruebas etc)

\_Nombre del comando –espacio- opciones del comando –espacio- y argumentos del comando.

$: promp (ahí se escriben comandos)

Pwd: enq carpeta estas

Ls: carpetas q hay // ls –la (archivos ocultos muestra)

Date: fecha

Clear: limpia

cd.. : moverse entre carpetas (anterior)

cd (tab va pasando llasas carpetas) si pones letra autocompleta.

mkdir: crea carpeta (mkdir ejemplo)

rm: eliminar archivo (rm ejemplo) si no se borra

rm –rf: borra sin permisos (rm –rf ejemplo)

Entrar a carpetas

## **En Windows:**

Para desplazarnos por los directorios o carpetas en la consola de Windows utilizamos el comando «cd» seguido por la ruta de la carpeta a la que queramos acceder. Por ejemplo, si queremos acceder a la carpeta Windows escribiríamos el siguiente comando en la consola:

cd c:/windows

Si el nombre de la carpeta tiene espacios podemos utilizar la barra invertida (\) para indicarle a la terminal que el espacio en blanco siguiente es parte del nombre del directorio. Por ejemplo:

cd c:/Archivos\ de\ Programa

GIT: sistema de control de versión

Hacer un commit: guardar una foto de tu versión

git commit -m'titulo del comentario que quieras hacer'

Git init . inicias un repositorio

Git status

Git add “nombre del archivo”: meter archivos modificados en la siguiente “foto!

Git add . : mete todos los archivos modificados a la prox

Git log: muestar historia del repositorio (cambios)

Git push origin master: sincronizar los cambios con la nube

Npm test ‘nombredelarchivo’ para hacer una prueba

Prueba: grabar siempre antes de chequear

Agregar .only para testear un solo ejercicio

Agregar x al ppio para saltear

Lección 2

En el test colocar .only dsp de describe para testear solo un caso

Lenguajes

Variable –

Var nombre

Nombre = ‘valores’;

Const… nombre = asigna un valor inmodificable

Let: variables que funcionan con scope (futuro…)

// escribir comentario que no influyen en la prog

Tipos de datos

Strings (hilo de caracteres): palabras van entre ‘ ‘; (si no lo pones enter ‘’ se confunde con variable)

Números:

Boolean : V o F true or false

Null: reservada para la nada misma.

Undefined: no definido (se muestra cuando nombramos una variable pero no la definimos)

Operadores matemáticos

= es la operación de asignación… asigna un vaor a una variable

Modulo: resto de la division

Java script cambia datos para interpretar los pedidos del q escribe: 1 + ‘1’ convierte en string el primer número (casteo de datos)

NaN: not a number (cuando le das a sumar cosas q no son números)

Funciones

No escribir código que haga dos veces lo mismo. Agrupa código lo guarda, para volver a ejecutarlo

Se escriben:

Function nombre de la función (argumentos que recibe –separados por comas-

) {código de la funcion} –se hacen alt+123 y alt+125)

Const otra funcion = funcion (argumentos ) {código de la funcion}

SCOPE: defninir dentro de los corchetes no influye definir variables pro fuera, generalmente

function presentar(nombre, edad) {

console.log('Hola, mi nombre es ' + nombre );

console.log('tengo ' + edad + ' años.');

}

presentar('pablo ielasi', 34);

invocamos la funcion con su nombre y ()

--scope--- no funca el nombre porque relativice la variable nomrbe en el scope mas próximo (dentro de la llave)

function saluda() {

var nombre

console.log('hola, '+ nombre);

}

function suma(a, b) {

console.log(a + b);

}

Return (52) …50:18

Function noretorna()

function siretorna() {

return 36; }

La sentencia return finaliza la ejecución de la función y especifica un valor para ser devuelto a quien llama a la función.

código escrito dsp de return, no va a ser retornado por la consola

control de flujo

= se usa para asignaciones…. Para comparar usamos ==

-if, else-

edad = 21

21

if (edad > 21) {

console.log('sos viejo'); } else {console.log('sosjoven'); }

sosjoven

-if, else, else if-

if( edad > 30) {

console.log('viejo') } else if (edad > 20 ) { console.log('joven') } else { console.log('bebe') }

1:00

Lección 3:

Cualquier numero excepto 0 es true para JS

Cualquier string es true excepto string vacia en JS : ‘’, “”

Operador: ! : not … ! false = true

Array [ ]

Objeto {}

Forzado a falso:

False, undefined, null, ‘’

array

== resultado no tan certero (no compara tipo de datos sino valor

=== compara valor y tipo de dato

Operadores lógicos:

And &&

Or || Alt + 124 = |

Not !

Tabla de la verdad

23:27

Ctrl + c … frena procesos

DRY CODE : dont repeat yourself

Bucle for

I++: es para cambiar de valor a i

Del 1 al tanto…

For (var i = 0; i >= 100; i++ ) { bloque de código}

for (var i =1; i<= 100; i++) {

console.log(i)

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Hasta el numero N

function imprimeaN(n) {

for (var i = 1; i<= n; i++) {

console.log(i);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Pares:

function imprimeaN(n) {

for (var i = 1; i<= n; i++) {

if ( i % 2 === 0) {

console.log(i);

}

}

Pares, que se termine la funcion en determinado number:

> function imprimeaN(n) {

... for (var i = 1; i <= n; i++) {

..... if (i % 2 === 0) {

....... console.log(i); }

..... if (i < 200) {

....... return;

}

..... }

}

Anidiar otro statement:

function imprimeaN(n) {

... for (var i = 1; i <= n; i++) {

..... if (i % 2 === 0) {

....... console.log(i) }

..... if ( i < 200 ) {

For (var j = 0; j < 10; j++) {

....... return;

console.log(j);

} return

}

}

}

Bucle anidado: dos variables –columna t fila-

Function matriz (n, m) {

For (var i = 0; i < n; i++) {

For ( var j = 0; j > m; j++) {

Console.log (i + ‘,’ + j);

}

}

}

Bucle while

While (condición) {bloque de codigo}

While (i > 100) se repite el código

> while (i < 100) {

... console.log(i);

... i++;

... }

Tab: baja la línea de escritura! (¿)

> function matriz(n, m) {

... var j = 0;

... var i = 0;

... while ( i < n ) {

..... while ( j < m ) {

....... console.log(i, j);

....... j++; }

..... j = 0;

..... i++;}

... }

Dsp del while (hay que volver a 0 la primer variable j para que imprima el resto)

Diferencia entre while y for

For, ya sabes hasta donde para la funcion

While, no sabes hasta cuando para a menos que lo indiques (sirve para lista enlazada)

HW 2:

For es un bucle que se utiliza en java script para repetir un determinado número de veces un código. Se suele usar cuando sabemos el número de veces que queremos ejecutarlo.

&& = Es un operador lógico, se suelen utilizar con valores booleanos y devuelven valores booleanos también. && devuelve true si ambos operandos son true.

|| = Es un operador lógico, se suelen utilizar con valores booleanos y devuelven valores booleanos también. || devuelve true si uno de los operandos es true.

! = Es un operador lógico, se suele utilizar con valores booleanos y devuelve valores booleanos también. ! devuelve false si su operando es true y viceversa

Array

let frutas = ["Manzana", "Banana"]

Funcion switch

Sirve para programar respuestas ante diversos casos y programar una por default para las que no sean

switch (color) {

case 'blue' : return 'This is blue';

break;

case 'red' : return 'This is red';

break;

case 'green' : return 'This is green';

break;

case 'orange' : return 'This is orange';

break;

default : return 'Color not found';

}

}

Si estoy en dos If y dos condiciones deben aplicarse a la misma variable, escribir primero la condición doble y dsp las simples (¿)

    if (numero < 2)

return false;

if (numero === 2)

return true;

for (var i = 2; i < numero; i++) {

if (numero % i === 0)

return false;

}

return true

}

Porque le damos valor a i, menor al numero y si da 0, es porque no es primo--- se le da el valor a i de mas de 1 pero menos del numero.

Se busca que se haga el resto del numero y la variable obviando hacerlo por uno y por el mismo numero. Y si llega a dar 0 se pone la sentencia de que retorne false, ya que no seria primo porque se dividiría entre un numero mayor a 1 pero menor al propio numero

|  |
| --- |
| function tablaDelSeis(){ |
|  | //Escribe una función que muestre la tabla de multiplicar del 6 (del 0 al 60). |
|  | //La función devuelve un array con los resultados de la tabla de multiplicar del 6 en orden creciente. |
|  | //Escribe tu código aquí |
|  | let arrayTablaDel6 = [] |
|  | for (let i = 0; i < 11; i++) { |
|  | arrayTablaDel6.push(6 \* i) |
|  | } |
|  | return arrayTablaDel6 |
|  | } |

Establece una variable global (let) arraydetabladel6 = []

Dsp establece una variable dentro de un proceso

For (let i = 0; i < 11; i++) {

Arraydetabladel6.push (6 \* i)

}

Return arraytabladel6

}

Este faltaaaaaaaaaa

  return [6 \* 0, 6 \* 1, 6 \* 2, 6 \* 3, 6 \* 4, 6 \* 5, 6 \* 6, 6 \* 7, 6 \* 8, 6 \* 9, 6 \*10 ];

DO – WHILE

var a = numero

var i = 1;

do {

  i = i + 1;

  a = a + 5;}

  while ( i < 9)

return a;

}

Se establecen dos variables

A = numero + lo que pida

I= cantidad de repeticiones

Se repiten las dos operaciones del DO, hasta que While termina con la funcion, es decir hasta que i sea igual a 9.

Do { a i le vamos sumando de a 1 y al numero lo definimos +5 o la cantidad que se necesite}

//do va entre corchetes

Dsp While ( hasta que lo necesitemos);

Return a, osea el numero

Alt + 91 : [

Alt + 93 : ]

Video 4

Arrays: arreglos

Listas de elementos. Aceptan todo tipo de datos adentro ioncluso funciones, u otros arreglos

Se define con corchetes (brackets)

Array no es un tipo de dato nativo, sino una construccion

Var array = [];

Arreglo vacio

Los elementos que integran el array tienen u orden y se pueden conocer escribiendo array[0] y responde el elemento 1, y asi sucesivamente.

Si hay una funcione n el array se la puede invocar poniendo array[4] mas los corchetes().

También si el array es numero se lo puede sumar, multiplicar, etc.

Ej; array[0] + 1

Si tenes un array adentro de otro podes invocar un elemento del array que esta adentro ej

array[5] [0]

si citamos una string dentro de un arreglo y mencionamos otro índice de arreglo. Retorna el numero de orden de cada letra del string. Porque un string es como un arreglo de caracteres.

array [1]

tito

array [1] [0]

‘t’

También se lo puede citar asi ‘junio’: junio [0] … j

ARRAY.LENGTH : es una propiedad que dice que cantidad de elementos tiene un arreglo

Si tiras array.length en un elemento que de por si es un array o un string tira la cantidad de elementos del sub-array o letras del string

si invocas arreglo [arreglo.length -1] lógicamente te da el último elemento de tu array, ya que la cantidad de elementos se cuetna desde uno, en lenth pero en el índice de la lista es desde 0.

Tmb podes determinar una variable i = 2 … y citar un elementro asi

Arreglo[i] . y va a tirar el tercer elemento de la lista

Asignar valores a un arreglo:

Se puede definir un arreglo de forma ‘literal’, osea escribirlo como queremos.

Pero a veces habrá que usarlo de otra manera. Con elementos que aun no sabemos cuales son

lista = [];

podes ir agregando ítems : lista [2] = 1; lista [3] = null … y si agregas lista [10] = 1; te van a aparecer 7 elementos vacios o undefined en el medio de tu lista

funciones push y pop

var lista = [];

lista. Push … funcion que pone un elemento en la última posición libre del arreglo

lista.pop … funcion que saca el último elemento de la lista

var str2 = lista.pop()

var str2 = devuelve el último elemento y este desaparece del arreglo. Hace dos cosas

cuando invocas un elemento de array te lo devuelve para que lo puedas usar en otras funciones, operaciones, guardarl oen otra variable, etc.

el término **idempotente** se usa para describir una operación **que** produce los mismos resultados si se ejecuta una o varias veces

lista.pop es no idempotente porque produce distintos resultados cada vez que se la invoca

recorrer arreglos:

con for:

for (var i = 0; i < lista.length; i++) {

console.log (lista[i])

}

Con while y .pop:

While (lista.length < 0) {

Console.log (lista.pop() );

}

Txt:

Un array es una lista de elementos en donde cada uno de ellos posee un orden específico. Aceptan todo tipo de datos adentro incluso funciones, u otros arreglos. Se definen con corchetes (brackets). Array no es un tipo de dato nativo, sino una construcción. Los elementos de este arreglo se pueden conocer invocando el índice de cada posicion de la lista.

Ejercicios:

Devolver un array con los elementos de un array anterior + 1:

  var nuevoarray = [];

    for(var i = 0; i < array.length; i++) {

 nuevoarray[i] = array[i] + 1;

}

       return nuevoarray;

  }

También podría retornarse el mismo array

{Array[i]=array[i] +1

} Return array

También se puede hacer usando la variable .push:

Var auxiliar=[]

Val= 0

For…{

Val: array[i] +1

Auxiliar.push(val);

} return auxiliar

.unshift para agregar elemento al ppio del array

Como retornar las palabras de un array separadas por espacios …. Funcion array.join –es un método de array-( aclaras como las separas si , . ‘ ‘ etc.)

.pop = extrae del final

return palabras.join(' ')

si queres corroborar si un elemento es parte de un array

for (i = 0; i < array.length; i++) {

  if (array[i] == elemento) {

  return true; }

}

  return false

}

Armas variable para que pase por todos lo sindices y la comparas con el elemento en cuestion

Devolver promedio

Crea variable total de elementos y variable suma

Var totaldeelementos = resultadostest.length

Var suma = 0

For ( var i = 0; i<resultados.lenth; i++) {

Suma = suma + resultadostest[i];

}

var promedio = suma / totaldeelementos;

return promedio

multiplicar argumentos

primero cumplir con el 0

if arguments.length === 0 … if ( !arguments)

… dsp usar auxiliar acumulador 1, para multiplicar…. Else ( si .length es mas que 0)

var = producto =1

producto = producto \* arguments[i];

return producto

retornar elementos mayor de 18 ….01:07!!!

Opción filtrando: El método **filter()** crea un nuevo array con todos los elementos que cumplan la condición implementada por la función dada… const result = words.filter(word => word.length > 6)… osea creas un array para los números mayores y retornas el .length de ese array

Var mayora18 = arreglo.filter ((element) => element > 18);

Var total = mayora18.lenth;

Return total

También puede hacerse creando el array

Mayoresa18= []

For (…..) {

If arreglo[i] > 18 {

Mayoresa18.push(arreglo[i]) }

}

Var total = mayoresa18.length;

Return total;

Function empieza con nueve…. Hay que converir el numero en string

Creando la variable str = String(n) convierte numero en string

Y dsp numero[0] === ‘9’ (comparar con la string 9

Function todos iguales:

Para comparar si todos los elementos de un array son iguales :

En el for uso arreglo.length – 1 porque quiero comparar arreglo[i] con el próximo índice que venga… entonces necesito que el for dure hasta el anteúltimo elemento y no hasta el final.

for (let i = 0; i < arreglo.length -1; i++) {

 if (arreglo[i] !== arreglo[i + 1])

return false;}

return true

1:36 video

BreK TAMBIEN SE PUEDE poner como primera sentencia post for

Usar continue como primera sentencia posterior al for antes de cualquier otra cosa ¡!!!!!

Poner  numero = numero + 2;

nuevoarray.push(numero);

después de descartar los casos que no sirven

clase 4: Objetos:

Un objeto es una colección de propiedades …. Asociada una key a una value … se accede al valor a través de la clave(key). Es una estructura de datos donde hay propiedades que consisten en la asociación entre una palabra clave y su valor correspondiente.

Da más información acerca de cada objeto que lo que da un array. No hay orden como en los array.

Cada key del objeto puede ser cualquier tipo de dato

Objeto = { nombredepropiedad: valor,

Un arreglo es un objeto pero con índice numerico

Boris = { nombre: ‘boris’, edad: 33, cursando: true}

Retornas con: Boris[‘nombre’] … bracket notation

Para ver el contenido(propiedades) de un objeto hay 2 posibilidades:

Toni[‘nombre’] … propiedades son strings

Otra forma más simple es el dot notation … es literal, no se pueden usar variables para invocar.

Toni.nombre

Toni.edad

También se puede asignar un valor ahorrando escritura… si la key y la value son iguales solo pones una y separas con coma para la prox propiedad

Asignas con toni.nuevapropiedad = 565456

toni[‘nuevapropipiedad’] = 20

para invocar un afuncion detro de un objeto:

objeto[metodo]()

para borrar una propiedad es:

delete objeto.propiedad

podemos guardar objetos en un array y accedemos con:

nombredearray[i] y dsp

nombredearray[i][‘nombre de la propiedad’]

nombredearray[i][‘nombre de la propiedad’] [1] para llegar a uno de los valores específicos de la propiedad

para escribir una funcion en un objeto

meow : function() {

    return ('Meow!');

keyword: This

cuando una propiedad dentro de un objeto es una función, le vamos a decir método: saludar e sun método del objketo persona

si cambias una propiedad de un objeto, no se modifica una función que la inncluia. Cambio en nombre y en funcion.

DRY CODE… usando el keyword This, invocas el objeto que estas usando en tu métodos. Es decir si cambia el objeto de referencia el método cambia también automáticamente

Ej

Persona = { nombre = ‘quique’,

Saludo = function() { console.log (‘hola, ’ + this.nombre);}

}

Gracias a .this logras que distintos objetos usen la misma funcion …. Hay que estalecer la funcion antes de crear los objetos … esa funcion escala porque sirve para todos

…Para iterar en un objeto que contiene un array de objetos hay que usar un for con el array.length… y dsp buscar objeto.array[0].propiedad

Devuelve "true" si el usuario tiene un valor definido para la propiedad "email"

Valor definido quiere decir que no es null ni undefined

Devuelve "true" si el objeto (parámetro "objeto") tiene una propiedad (key) cuyo nombre es igual al valor del argumento "propiedad"

Los objetos tienen un método… objeto.hasOwnProperty(entre paréntesis va la propiedad que queremos buscar)

objeto.hasOwnProperty(propiedad)

"usuarios" es un array de objetos "usuario"

  // Cada objeto "usuario" tiene la propiedad "esPremium"

También se puede usar con map y con for each

Suma todos los likes de todos los objetos "post

Tmb se puede hacer con un for each

Clase V

Clase: plantilla para la crecacion de objetos… sustantivos. Ej: clase : alumnos –juan carlos etyc… clase autos : corsa, palio

Como definir clases y usar esas estructuras para crear objetos que se llamaran instancias…

Js leguaje de prototipado. No 100% orientado a objetos: copiar sintaxis de java.

Un objeto es literal cuando lo escribo tal cual es

Construir objetos: function construction… creamos plantilla o clase

Keyword new: hace 3 cosas. crea un objeto nuevo vacio

Dsp

Car (fiat, VW, Chevrolet

Sintaxis:

function Nombredelaclase() //empiezan con mayúscula todas las funciones constructoras (es una convención)

constructor de clase evdencia tmb el nombre de la clase cuando retorna la función.

this.nombre = Juan

this.apellido = perez

el this en este caso invoca la clase

…

Si la clase no esta nominada

new persona(); … instanciar un apersona(clase)

new : crea un objeto nuevo vacio, invoca la funcion ‘clase’y bindea con el objeto vacio creado.

Var objeto = {}

Car(’audi’, ‘VW’).bind(objeto) // bind = el this de la funcion car tiene que ser objeto

Y finalmente retorna un objeto.

Default opeerator= si no le paso un argumento a alguna propiedad de la funcion constructora pero pongo || ‘jota’ …. Por default pasa es enombre

this.nombre = nombre || ‘Juan’,

|| y && … so operadores de control shortcircuit. Para averiguar propiedades de objetos, aun sin saber si los objetos existen

Juan && juan.edad

33

Boris && Boris.edad

Undefined (porque Boris existe pero no está definido). Y no tiene propiedades lógicamente

Prototipo: todos los objetos en js tienen una referencia a otro objeto (su prototipo)

Los objetos tienen sus propias propiedades, pero también tienen propiedades por defecto que traen sus prototipos.

Audi = prototipo es Car …. Lo podes invocar audi.\_\_proto\_\_

[].\_\_proto\_\_

Array[]

Obj.\_\_proto\_\_

Object {}

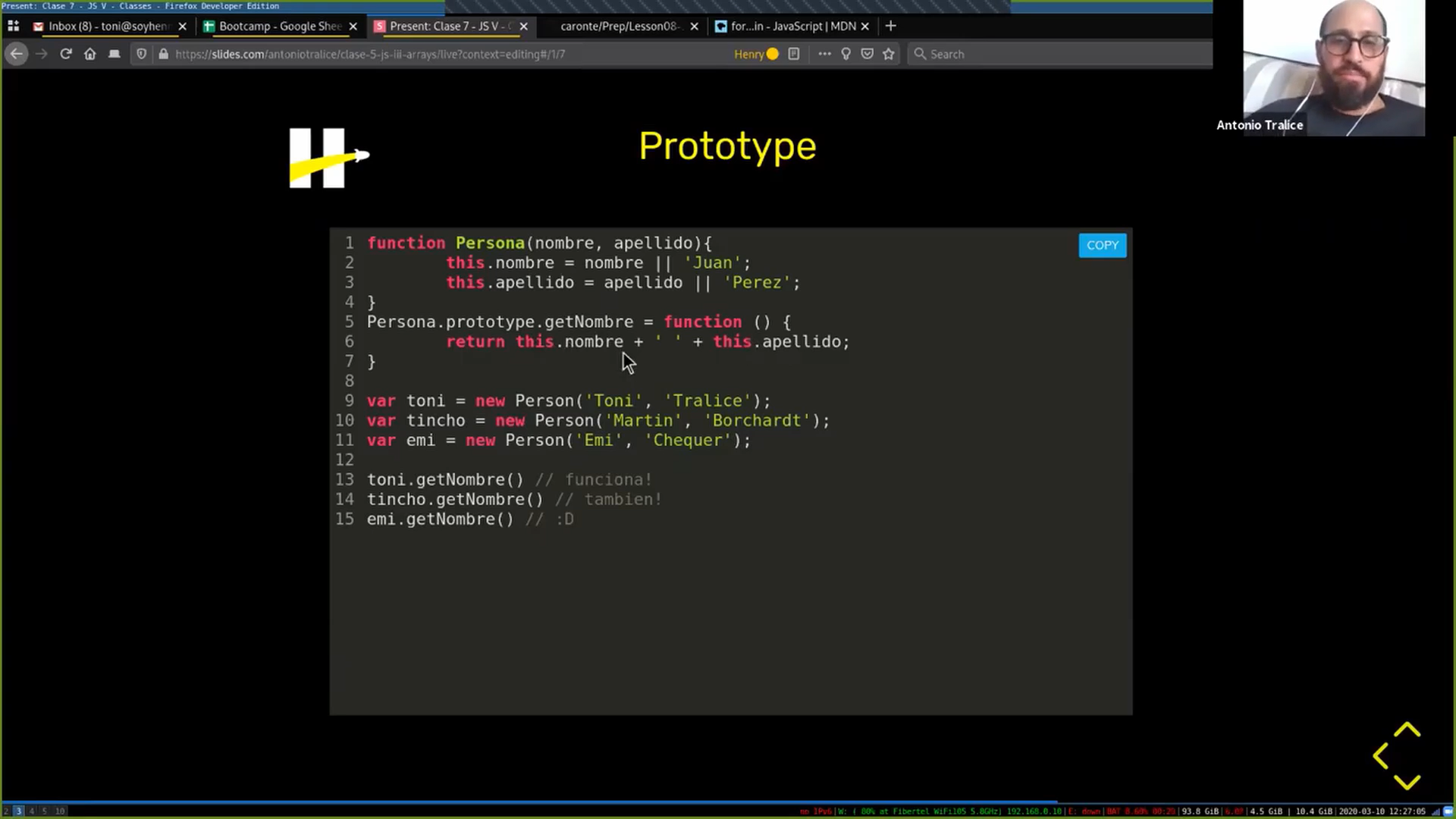
Cuando uso una funcion constructora estoy definiendo que esa funcion va a ser el protoitpo de los objetos que cree a partir de ella

Nunca cambiar valor de una propiedad con \_\_ \_\_

Se puede agregar una función al prototipo para no tener que escribirla en cada objeto.

Persona.prototype.getEdad = function () { return this.edad;}

Y también se puede poner una funcion get.edad a un objeto determinado para que sea un resultado diferente. Invoca primero si el objeto tiene la misma propiedad que el prototipo



Object.create: object es el objeto alfa… tiene un método que se llama créate .. crea un nuevo objeto con el prototipo del primer objeto que le pasamos.

Var Persona = { edad: defecto

get.Edad: function getEdad(){ return this.edad;

}

}

Var juan = object.create(Persona);

Te arroja el objeto vacio, no refleja los parámetros del prototipo.

Class … copia de sintaxis de otros lenguajes… se escribe el constructor y dsp los métodos.

class Persona {

constructor(nombre, apellido) {

this.nombre = nombre;

this.apellido = apellido;

}

getNombre() = {

return this.nombre;

}

Saludar() {

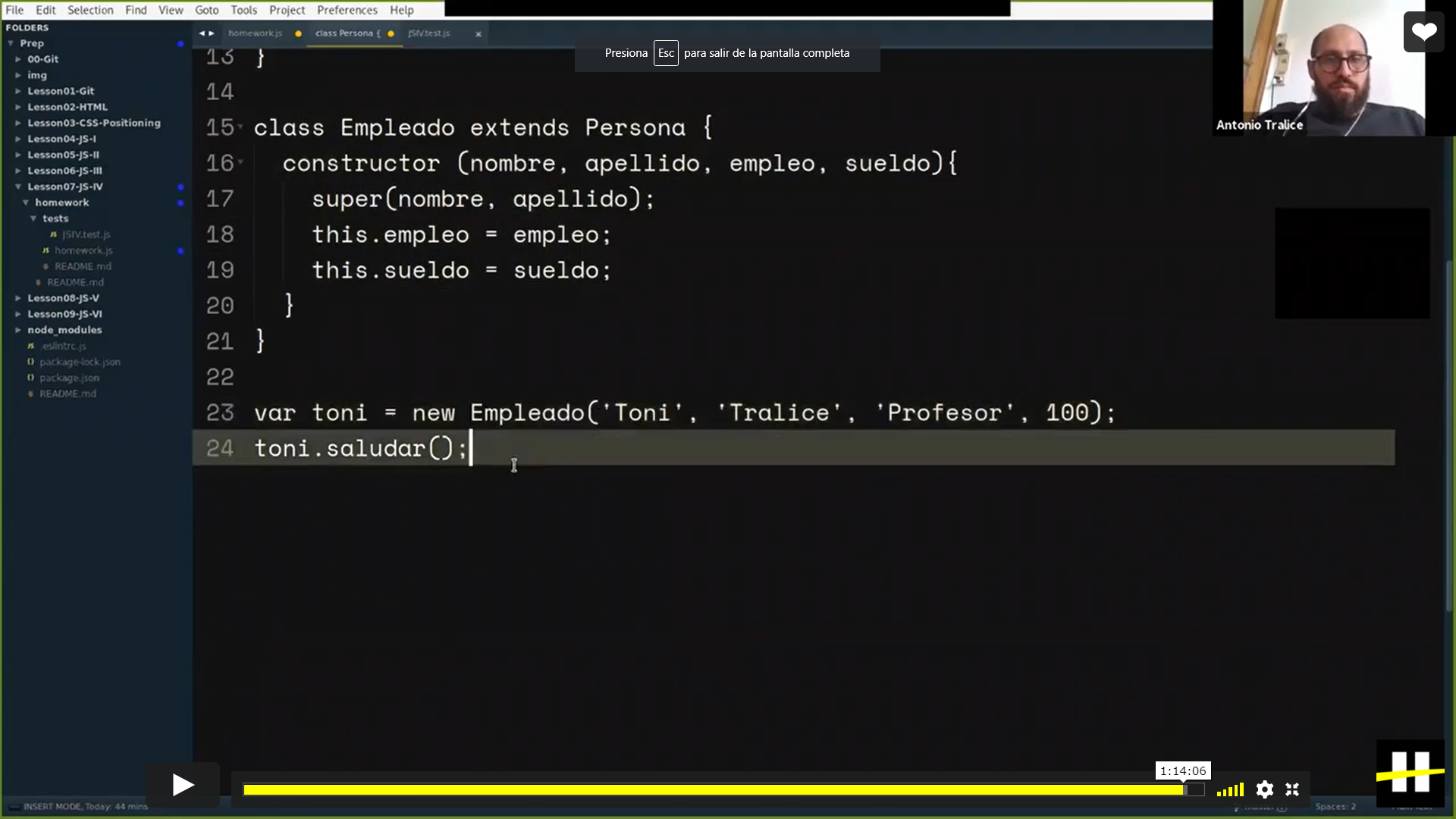
Return ‘hola’ + this.nombre;

}

}

Var toni = new Persona(‘toni’, ‘tralice’)

Extender una clase



Super quiere decir que invoco al constructor de la clase de la cual extiendo (Persona)

Y this.empleo this.sueldo es la información que viene del nuevo constructor Empleado.

Se pueden invocar funciones de la clase extendida para el objeto de la nueva clase

**Object.assign**

El método assign de los objetos te permite agregar propiedades a un objeto pasado por parámetro

**>** **var** obj **=** {}

*// No hace falta guardar el resultado porque los objetos se pasan por `referencia`*

**>** Object.assign(obj, {nombre:'Emi', apellido:'Chequer'})

**>** obj.nombre

**<** 'Emi'

Ejs

1)

function Usuario(opciones) {

    this.usuario = opciones.usuario;

    this.nombre = opciones.nombre;

    this.email = opciones.email;

    this.password = opciones.password;

    }

    Usuario.prototype.saludar = function() {

      return 'Hola, mi nombre es ' + this.nombre;

  }

return Usuario;

}

Cuando haces un afuncion constructora ya con un objeto definido en la definición de las propiedades hay que agregar nombredeobjeto.propiedades

2)

function agregarMetodoPrototype(Constructor) {

  // Agrega un método al Constructor del `prototype`

  // El método debe llamarse "saludar" y debe devolver la string "Hello World!"

  // Tu código:

Constructor.prototype.saludar = function() {

  return 'Hello World!';

}

}

Cuando agregas un método no hace falta vovler a mencionar la funcion constructora

3) charAt ¿?? = devuelve los caracteres de una string …. Si en el for aclaras que sea i – va de atrás para adelante con reverse

4)

Cuando armamos la clase ponemos el = para definir las propiedades

Cuadno retornamos el objeto… como es un objeto … como es un objeto enumermos las propiedades con ,

Nombre : this.nombre … usamos los dos puntos

5) otra forma de anotar el return es :

Return ‘${this.nombre}, ${this.edad} años’;