Index.js es el default para correr en node.js

./ me permite correr un archivo dentro de la misma raíz cuando no tengo un servidor, ejemplo, si tengo el index.js en la misma ubicación de index.html en el src ./indx.js para que lo corra.

En el navegador nunca podrán implementarse dos bloques de código javascript al mismo tiempo

Poner las etiquetas script al final me permite que los script se carguen luego del html y el css, es recomendable ponerlos como ultima línea del body.

Con typeof y algo me dice de que tipo de dato es ese algo, ej typeof “hola” me devolverá que es un string.

Typeof puedo utilizarlo también para corroborar si una variable existe, por ejempl:

If (typeof myVar === “undefined”) {

Alert(‘la variable no existe’);

}

OJO, DESPUES DE LAS LLAVES NO SE USA ;

Para meter comiilas dobles dentro de comillas dobles uso “la palabra \”hola\” se usa para saludar”, mas practico es usar siempre comillas simples, entonces dentro puedo usar las dobles, ejemp´lo ‘la palabra “hola” se usa para saludar’

Cada statement que devuelve un valor es un expression. Ej:

>‘ la palabra “hola” se usa para saludar’

<0 “la palabra ‘hola’ se usa para saludar” el signo <0 en la consola me indica que es una expression devuelta por lo ejecutado anteriormente.

Tipos de datos primitivos:

String – ‘ hola’

Number – 33

Boolean – true or false

Null – representa la ausencia de un objeto, en javascript todo es objeto, por ejemplo los definidos anteriormente. Solamente de un solo valor posible, null.

Undefined, tiene un solo valor posible, undefined.

Todo lo que no es dato primitivo es object.

Es recomendable siempre mantener el mismo tipo de dato en una variable.

Las variables se inician siempre automáticamente en undefined, hasta tanto yo les declare algo.

Para declarar muchas variables de una puedo separlas con coma, ej var nombreUsuario, myVar, nombre.

Los paréntesis es una expresión que devuelve lo que tenga adentro.

Los tipos de datos primitivos son inmutables, esto quiere decir que no puedo reemplazar una parte de el.

Ej: var myVar=’abc’

myVar[1]=’-‘ esto no devuelve myVar=’a-c’ sino que devuelve ‘abc’ como era originalmente, esto es muy importante de entender y aplica a todos los primitivos.

Las funciones de tipo de dato con la primera en mayúscula devuelve lo que le metamos adentro a ese tipo de dato (casting) ej String(3) devuelve ‘3’.

Las comparaciones booleanas con and (&&) , si la primer expresión es false devuelve false, sino la ultima, ejemplo

0 && 1

0

True && 1

1

Las Las comparaciones booleanas con or(||) corta en el primer true, si la segunda es true devuelve la segunda y si ambas son false devuelve la ultima.

1 || 2

1

0 || 2

2

0 || false

false

Fulsy cualquier expresión que aplicándole boolean da false

Trully cualquier expresión que aplicándole boolean da true

Operadores

== convierte los dos al mismo tipo y evalua(osea que no tienen que tener el mismo tipo, convierte para comparar), con el === evalua sin convertir tipos, por lo que deben ser igual estrictos, por eso siempre se usa siempre ===.

Modulo, en los numero pares su modulo por 2 es cero, por ende es una manera de comprobar pares e impares, ejemplo 10 % 2= 0 por ende par.

Operador ternario:

Se usa para cuando el if o else lo único que hace es asignar a una variable, de esta manera es mas corto y semántico.

Var numero = Number(prompt(“ingresa un numero”));

Var parImparMensaje = (numero % 2 === 0)? “Par” : “Impar”;

Si lo de entre paréntesis es true devuelve Par, sino Impar.

Con if seria

If(numero % 2 ===0){

parImarMensaje=’Par’;

}

Else {

parImparMensaje= ‘Impar’;

}