**Esquema Cliente-Servidor**

Cliente: html, css, js (javascript)

Javascript es un lenguaje interpretado. El cliente, en este caso el chrome tiene un motor js incluido que va interpretando las líneas de código y las ejecuta. Es orientado a objetos (javascript)

En servidor también vamos a tener js pero node js

Entre cliente y servidor tenemos http que lo que haces es comunicar mediante peticiones de acuerdo a un protocolo

Ajax es un método de comunicación asincrónico entre el cliente y el servidor

Comunicación HTTP con métodos get, post

Nosotros vamos a tener un archvo node server.js que va a estar corriendo en el servidor.

JS es orientado a objetos, también es debilemente tipado. Ej. String cadena=”hola”

Var es la palabra reservada para declarar variables. Como la declaramos? Asi:

Var cadena=1;

Es dinamico (js). Si hago tipeof(cadena) me va a dar tipo numérico pero si en otra línea de código hago

Cadena = “hola”; el tipeof va a dar un string;

Si utilizo el “strict mode” me va a ir avisando si por error voy cambiado el tipo a la variable

Las variables son case sensitive

Tipos de variables primitivos:

number

string

boolean

undefined

Tipos de variables complejos:

object

function

null (es un objeto) el tipo primitivo de null es undefined

date

math

custom

Los ; en el código no es obligatorio(necesario) pero es altamente recomendable usarlos

Para los nombres de las variables en javascript se usa el lower camel-case (se empieza con minúscula)

Las variables no pueden empezar con números. Si puede por ejemplo empezar con \_ o con $

El guion bajo los vamos a usar para las variables privadas

Puedo declarar varias variables en una línea separándolas con coma. Ejemplo: var a,b,c;

Operadores:

Asignación: =

Increm/decremento ++ --

Lógicos:

Not ¡

And &&

Or ||

%- modulo

Relacionales: < > <= >= != == ===

// otra manera de escribir el if es con operador ternario:

var pais ="Argentina";

alert( (pais==="Argentina")? "local" : "Extranjero");

**Call-back**

Tiene que ver con hilos y llamadas sincronicas

Dentro de una función se van ejecutando sentencias de js hasta que aparece un sentencia de Ajax que le pide al servidor haga algo

Vamos a querer que alguien atienda el feedback del servidor. Ahí es donde entra a jugar el callback que atiende esa respuesta del servidor.

Callback no es ni mas ni menos que una función

Con el callback a la función que se la paso le estoy pasando código extra (que es el que tiene la la función callback)