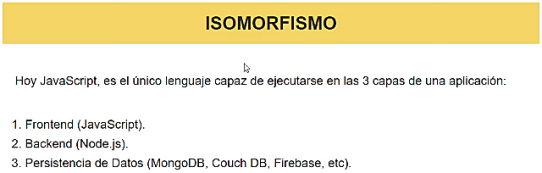
**Javascript**

**Introducción**

**Isomorfismo**

JS es el único lenguaje de programación donde puedes hacer *isomorfismo*; es decir, puedes hacer una aplicación de principio a fin con una sola tecnología. También es el único lenguaje del lado del *front,* nace en los navegadores, los demás tienen que compilar todo su código a código nativo JS para que los navegadores lo interprenten. Ahora, JS también trabaja o puede programar al lado del ambiente del *servidor,* eso lo hace por medio de Node.js



Cuando se habla de *3 capas* se refiere al ***Modelo-vista-controlador,* ¿y qué es?**

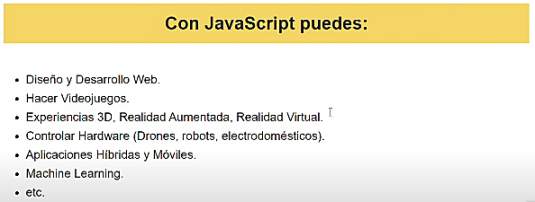
**Modelo Vista Controlador (MVC)** es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

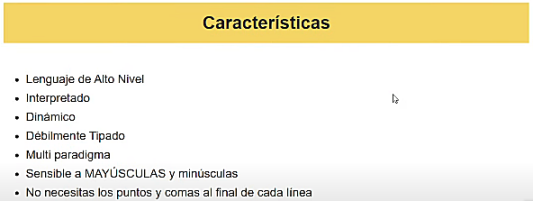
Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo.

El ***Modelo*** que contiene una representación de los datos que maneja el sistema, su lógica de negocio (está en el servidor), y sus mecanismos de persistencia. *(SQL + NO SQL)*

La ***Vista,*** o interfaz de usuario (la *UX,* la *interactividad* o el *front*), que compone la información que se envía al cliente y los mecanismos interacción con éste *(Frontend)*

El ***Controlador,*** que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno (Backend).





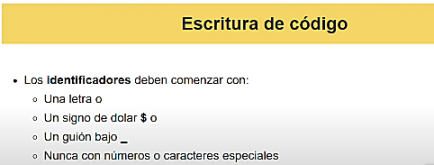
**Lenguaje de Alto Nivel**: Cuando estamos frente a un *lenguaje de Alto Nivel* no nos tenemos que preocupar por, por ejemplo: gestionar la memoria, la recoleción de basura para mejorar el rendimiento de los recursos que estamos utilizando de la computadora *“Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas. Estos lenguajes permiten una máxima flexibilidad al programador a la hora de abstraerse o de ser literal.”*

**Interpretado**: Cuando estamos frente a un lenguaje *interpretado* no se necesita una fase de compilación para, posteriormente, pasar a una fase ejecución; es decir, JS se compila al tiempo que se ejecuta, todo en un sólo paso; y, justamente eso, es lo que lo vuelve **Dinámico.**

**Debilmente tipado:** Cuando estamos frente a un lenguaje *debilmente tipado* se refiere a que las variables, para ser declaradas, no necesariamente se casan o pertenecen a un sólo tipo de dato; de hecho, a lo largo del código, el tipo de dato de una misma variable puede variar; ser originalmente, por ejemplo, una variable de tipo númerica y, luego, ser una variable de tipo *string* o viceversa.

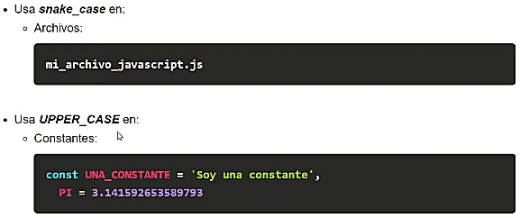
**Multiparadigma:** Que puede dedicarse a varios *tipos* de programación; por ejemplo, JS se puede emplear para hacer programación estructurada, programación orientada a objetos, programación funcional, etc.

Js es ***Keysensitive;*** es decir, es sensible a mayúsculas y minúsculas.Y por último, en JS **no necesitas los puntos y comas al final de cada línea**; sin embargo, *es una buena práctica.* Quizás, en el mejor de los escenarios, sí será necesario hacer uso de puntos y comas **cuando pongas varias declaraciones en una misma línea de código**, entre otros casos.



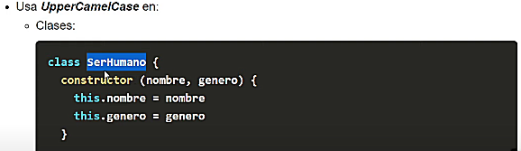
Los *identificadores* son los nombres que les pasas a una variable, función, clase, etc... Adicionalmente, los *identificadores* no pueden repetirse en ningún escenario; es decir, dos entidades no pueden tener el mismo nombre. Tampoco puedes definir un *identificador* con espacios en blancos.

***Buenas prácticas a la hora de definir tus identificadores***

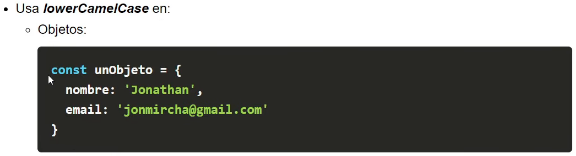


***Snake\_case*** consiste en usar *guiones bajo* para la separación de cada palabra, al ser pasada, en la definición de un identificador.

***Upper\_case*** consiste en definir todo el identificador con letras Mayusculas.

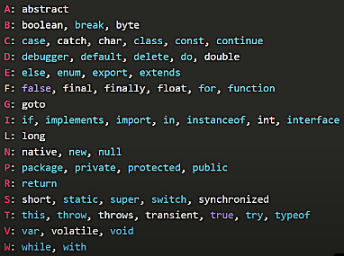


***UpperCamelCase*** consiste en defnir la letra inicial de cada palabra, del identificador,en mayúscula.

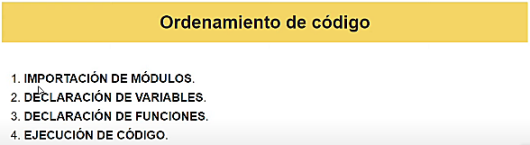


***lowerCamelCase*** consiste en definir la letra inicial de la 1ra palabra, del identificador, en minúscula; del resto, todas las iniciales de las palabras subsiguientes irán en mayúscula. Esta misma *buena práctica* se aplica no sólo en ***Objetos;*** sino, también, en valores ***Primitivos, Instancias*** y en ***Funciones.***

*Palabras reservadas*

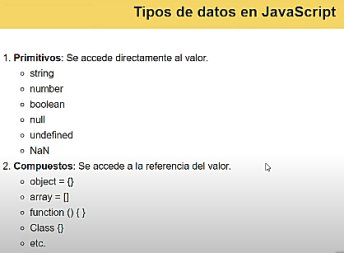


***Buenas prácticas a la hora de ordenar tu código***



Se le llama ***Modulos*** a los archivos JS; más allá de eso, cuando hablamos de ***Importación de módulos,*** nos referimos a, desde el navegador, mandar a llamar archivos JS dentro de otros.

Cuando hablamos de ***Ejecución de código*** nos podemos referir a, por ejemplo, crear una instancia de una clase, mandar a llamar métodos de un evento, asignar eventos, etc.



En ***Js*** no hablamos de números *flotantes,* *enteros,* etc... solamente de un mismo tipo de dato que los *encierra* a todos: ***number***