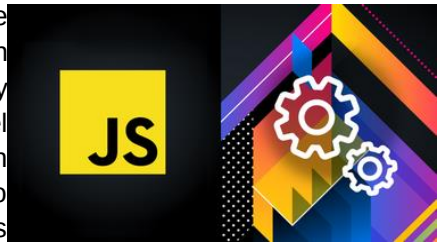
Carlos Roberto Loza Castillo 00002522

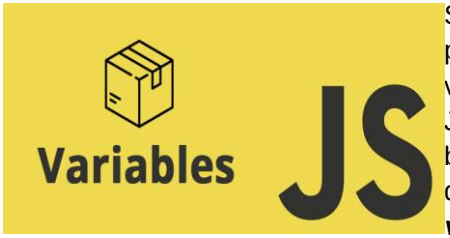
JSON: es un formato que almacena información estructurada y se utiliza principalmente para transferir datos entre un servidor y un cliente. El archivo es básicamente una alternativa más simple y liviana al XML (Lenguaje de marcado extenso, por sus siglas en inglés) que cuenta con funciones similares. Los desarrolladores usan json para trabajar con ajax (JavaScript asíncrono y XML). Estos formatos funcionan bien juntos para lograr la carga asincrónica de los datos almacenados, lo que significa que un sitio web puede actualizar su información sin actualizar la página. Ejemplo:


```
{
"_id": {"$oid": "63e2d62c2ea3af942f2db767"},
"title": "CARRITO DE COMPRAS(PROYECTO INNOVADOR)",
"project_type": "App Web",
"first_image": {"url":"/y07d8jl7hqzpbzgyajha.png","public_id": "y07d8jl7hqzpbzgyajha" }
}
```

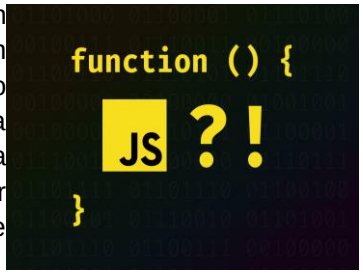



Estructuras de control: La parte más inmediatamente visible de un lenguaje de programación es su sintaxis o notación. Un analizador es un programa que lee un fragmento de texto y produce una estructura de datos que refleja la estructura del programa contenido en ese texto. Si el texto no forma un programa válido, el analizador debe señalar el error. Nuestro lenguaje tendrá una sintaxis simple y uniforme. Todo en Egg es una expresión. Una expresión puede ser el nombre de un enlace, un número, una cadena o una aplicación. Las aplicaciones se utilizan para llamadas a funciones pero también para construcciones como if o while. Para mantener el analizador simple, las cadenas en Egg no admiten nada como escapes de barra invertida. Una cadena es simplemente una secuencia de caracteres que no son comillas dobles, entre comillas dobles. Un número es una secuencia de dígitos. Los nombres vinculantes pueden consistir en cualquier carácter que no sea un espacio en blanco y que no tenga un significado especial en la sintaxis un ejemplo puede ser:

```
getProduct(productkey).then(product => {
const { stock } = product
if (cant <= 1) {
dispatch({ type: Actions.QUANTITY_PRODUCT, payload: { cant: 1, productkey, stock } });
} else {
dispatch({ type: Actions.QUANTITY_PRODUCT, payload: { cant, productkey, stock } });
}
}).catch((err) => {
SystemError(err)
});
```



Son la manera como los programadores le dan nombre a un valor para poder re usarlo, actualizarlo o simplemente registrarlo. Las variables se pueden usar para guardar cualquier tipo de dato en JavaScript. Existen varios tipos de variables, los let que son como bloques, lo var que son globales y las constantes que solo tienen datos y no se pueden modificar ejemplo:
var nombre = 'Carlos Roberto';
let apellidos = 'Loza Castillo';
const edad = 20;



Las funciones son uno de los bloques de construcción fundamentales en JavaScript. Una función en JavaScript es similar a un procedimiento, un conjunto de instrucciones que realiza una tarea o calcula un valor, pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida donde hay alguna relación obvia entre la entrada y la salida. Para usar una función, debes definir en algún lugar del ámbito desde el que deseas llamarla. Entre estas funciones se encuentran: normales, anónimas y las funciones flecha.

Funciones normales:

```
function SystemError(err) {
Swal.fire({
title: 'Connection server error',
text: err + ', we will solve this problem as soon as possible.',
icon: 'error',
button: "Aceptar",
footer: '<a href="mailto:ufostartservices@gmail.com">Report problem</a>'
}).then(() => {
setSystem(false);
})
}
```

Funciones flecha:

```
const getProducts = async () => {
GetProducts().then(product => {
setProducts(product.data);
}).catch((err) => {
SystemError(err)
});
};
```

Funciones anonimas:

```
(function(){
  const message = 'Selim Bradley is homunculus';
  console.log(message);
})();

// Selim Bradley is homunculus
```