¡Javascript!

guayerd beta hub

guayerd

Agenda del día

01

Introducción

Presentación Unidad 2

02

Funciones

Definición conceptual.

Declaración.

Invocación.

Return.

Parámetros.

Argumentos.

Funciones anónimas.

Funciones de flecha.

03

Ejercitación

Crear una función que nos diga en qué estación del año estamos.

Envolver el código de la entrega del Sprint 1 en una función.

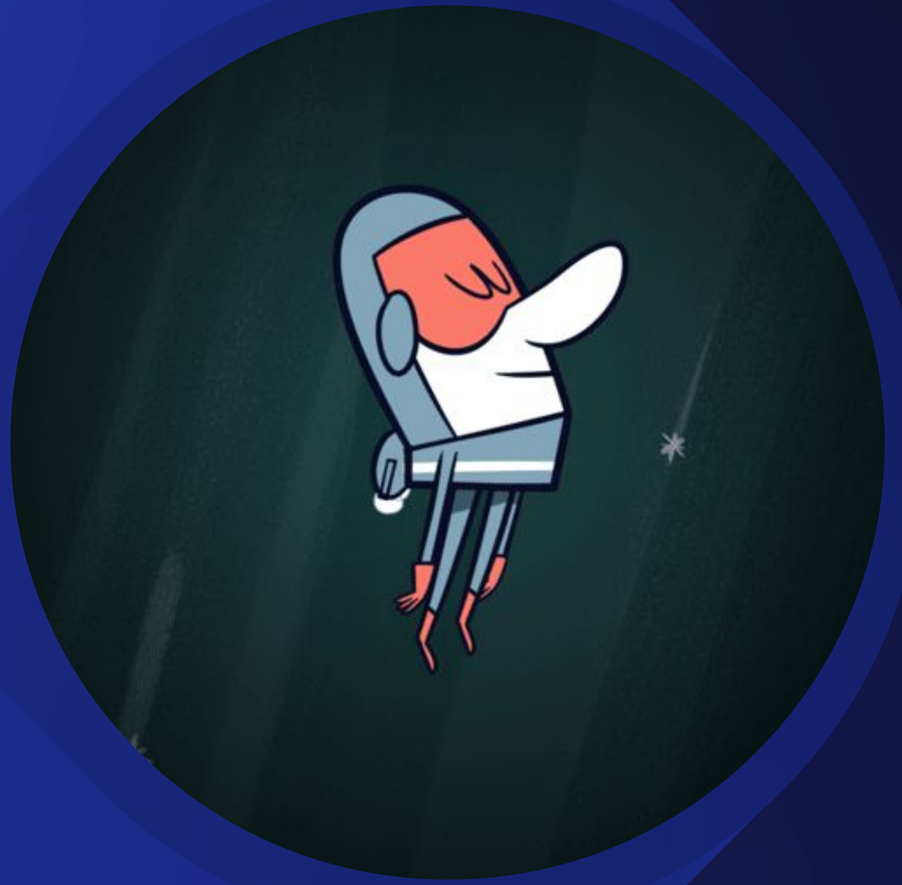
guayerd

daily

¿Cómo venimos?

¿Algo nos bloquea?

¿Cómo seguimos?



guayerd

Funciones

Javascript

guayerd

SPRINT 2

Funciones

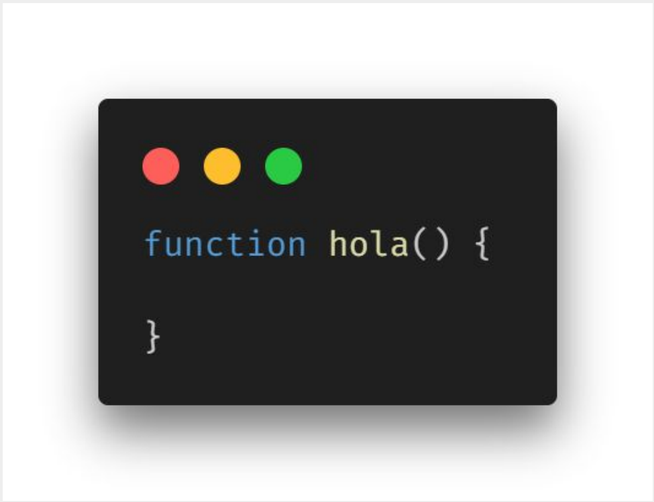
Las funciones permiten encapsular tareas y poder ejecutarlas cuando se desee.

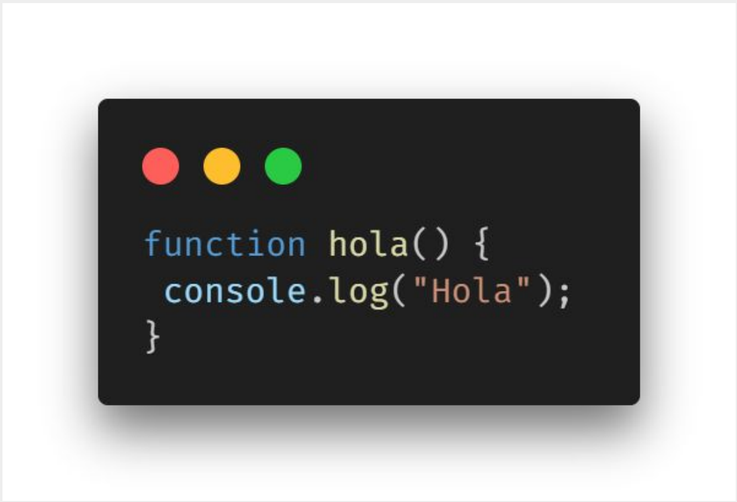
Su gran utilidad radica en que permiten reutilizar el código de manera dinámica y separan preocupaciones en la aplicación.

guayerd

Declaración de función

Una declaración de función normal comienza con la palabra **function**, seguida del nombre de la función. A continuación escribimos unos paréntesis y, por último, el bloque delimitado por llaves { }. El bloque contendrá las instrucciones a realizar.





guayerd

Invocación de función

Las funciones se ejecutan cuando se necesita.

Simplemente escribimos su nombre seguido de los paréntesis.

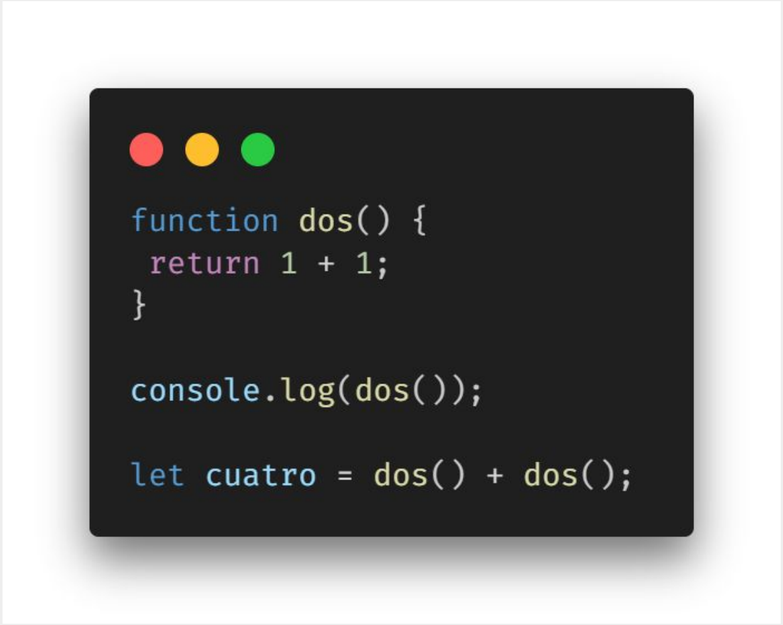
```
function hola() {
 console.log("Hola");
}

hola();
```

```
Hola
```

guayerd

Return

En general, es muy práctico que una función **retorne** un valor y nosotros podamos hacer lo que queramos con ese valor. Esto lo logramos con la palabra return precediendo el valor que queremos devolver.

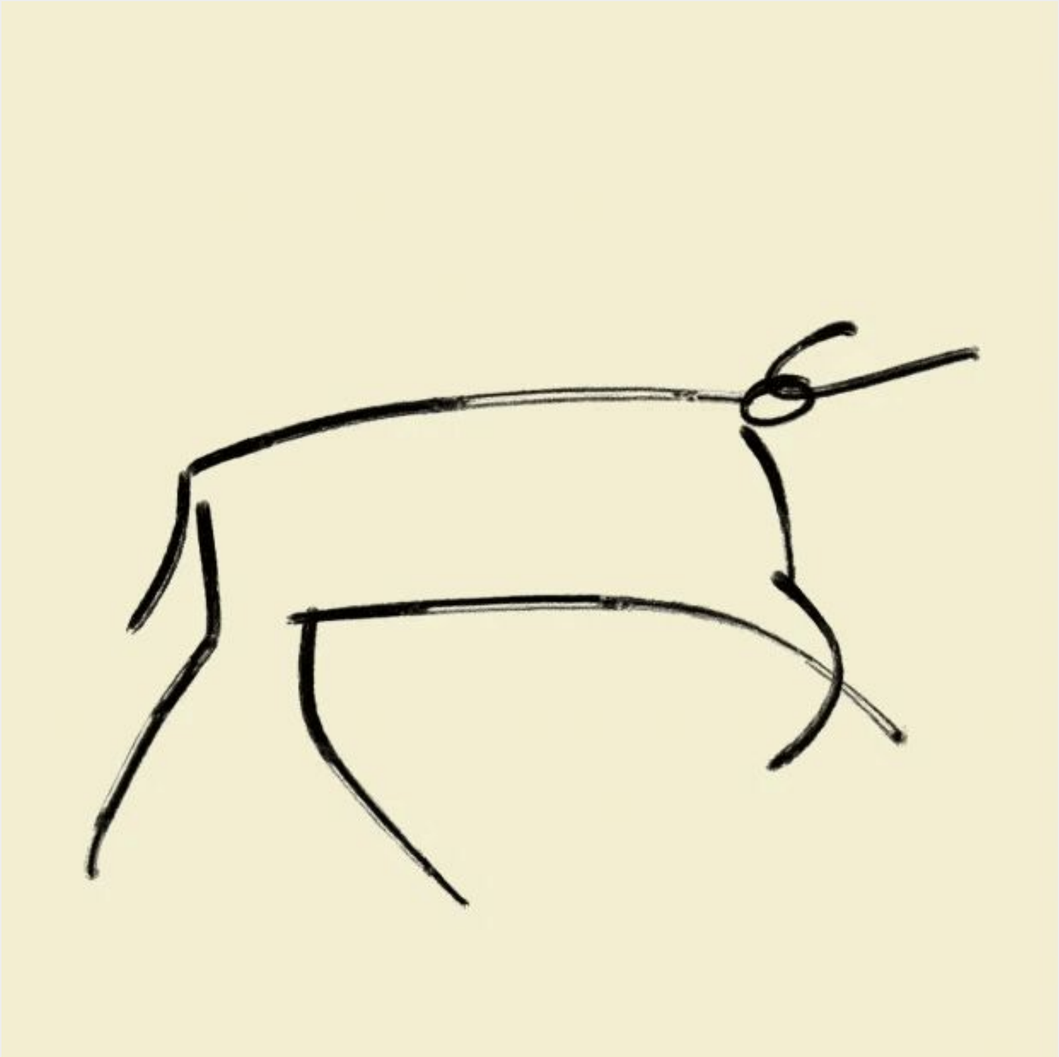


```
function dos() {
 return 1 + 1;
}

console.log(dos());

let cuatro = dos() + dos();
```

guayerd

Parámetros y argumentos

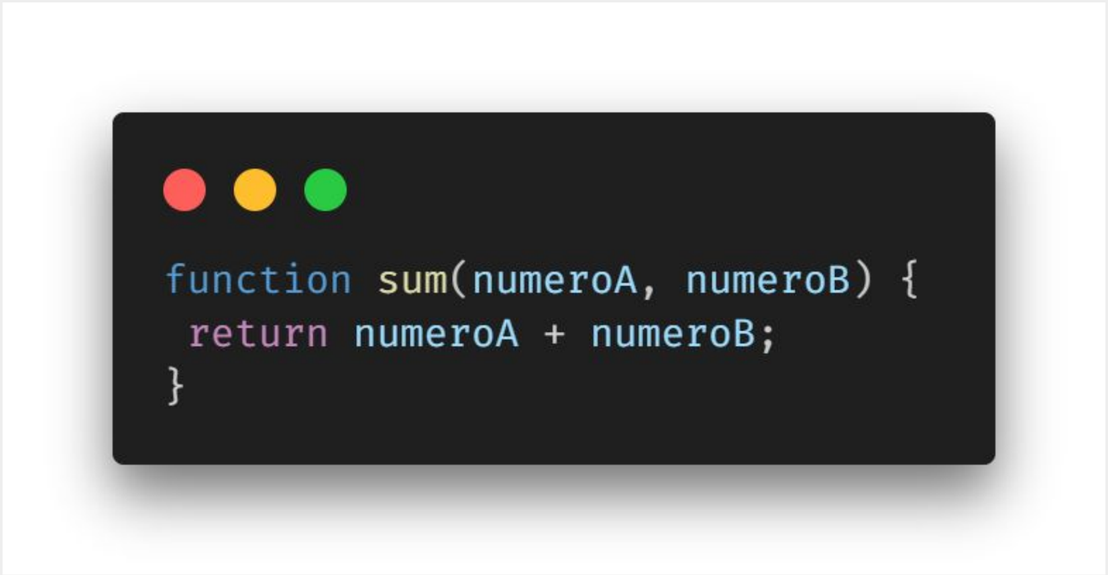
Funciones



guayerd

Parámetros

Podemos hacer que una función sea dinámica y responda valores distintos usando parámetros separados por comas en su **definición**. Los parámetros son variables internas de una función que toman sus valores de los argumentos pasados en la invocación de la función.



guayerd

Argumentos

Son simplemente valores separados por coma, pasados en la invocación de la función. Los parámetros toman estos valores, los cuáles son aplicados en la lógica interna de la función.

```
function sum(numeroA, numeroB) {
 return numeroA + numeroB;
}

console.log(sum(2, 2));
console.log(sum(5, 5));
```

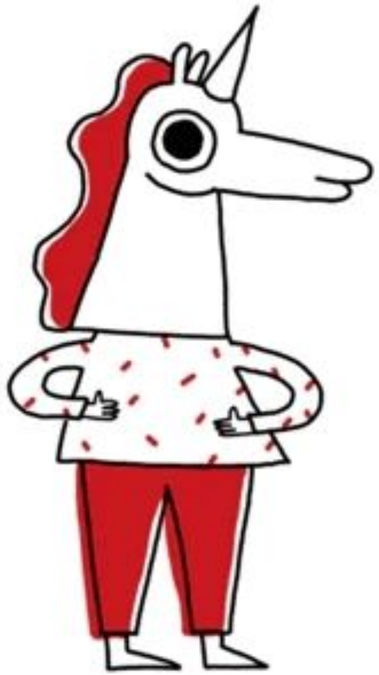
```
4
```

```
10
```

guayerd

Funciones anónimas

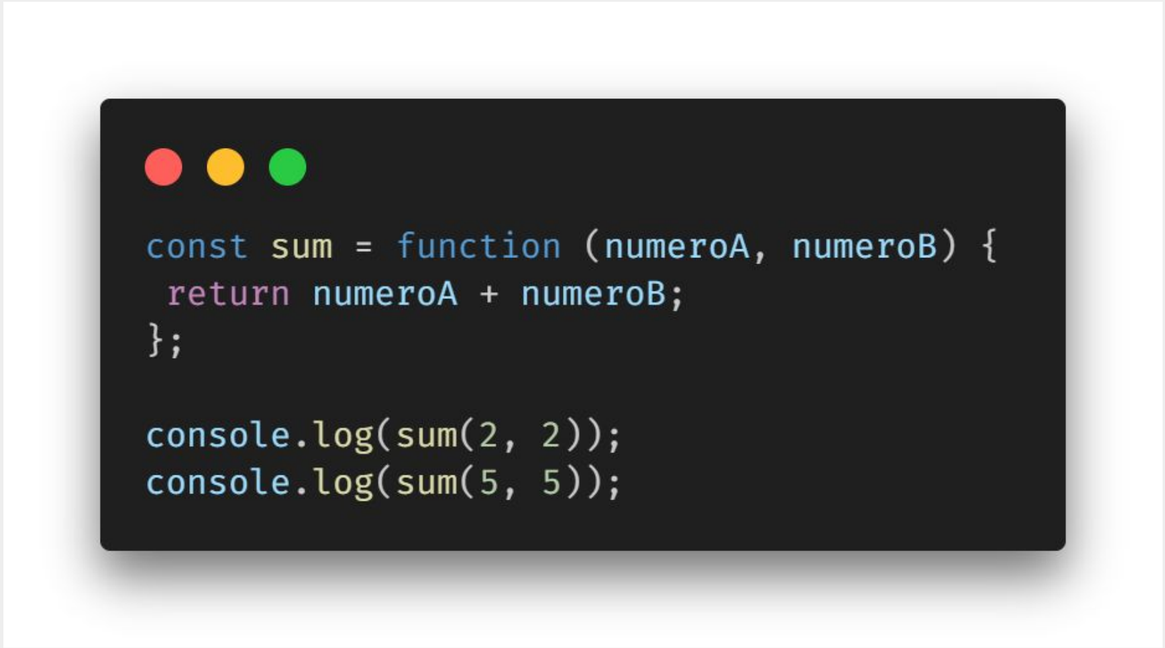
Funciones



guayerd

Expresiones de función

Asignamos una función anónima a una variable. Esto quiere decir que para Javascript una función es un valor.

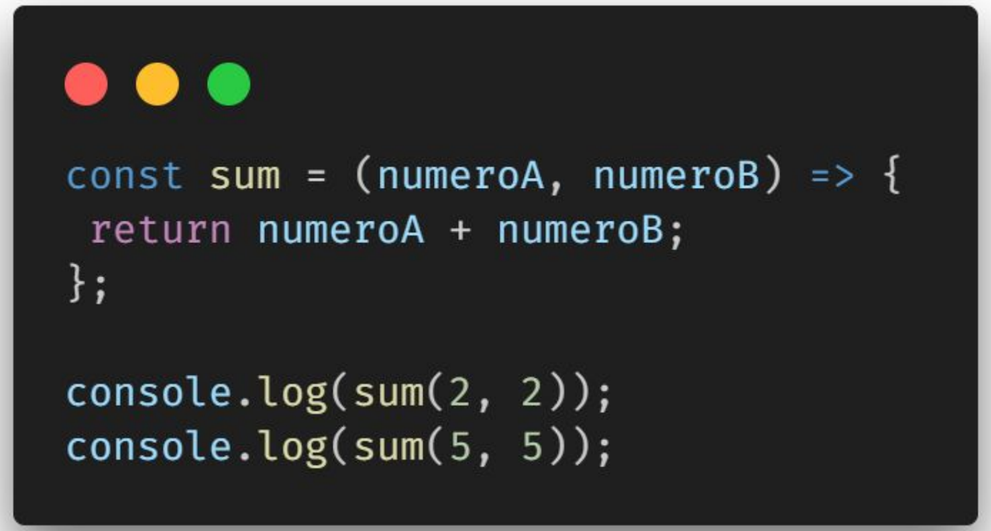


```
const sum = function (numeroA, numeroB) {
 return numeroA + numeroB;
};

console.log(sum(2, 2));
console.log(sum(5, 5));
```

guayerd

Expresiones de función de flecha

Sintaxis alternativa sin la palabra function y con una flecha luego del paréntesis.

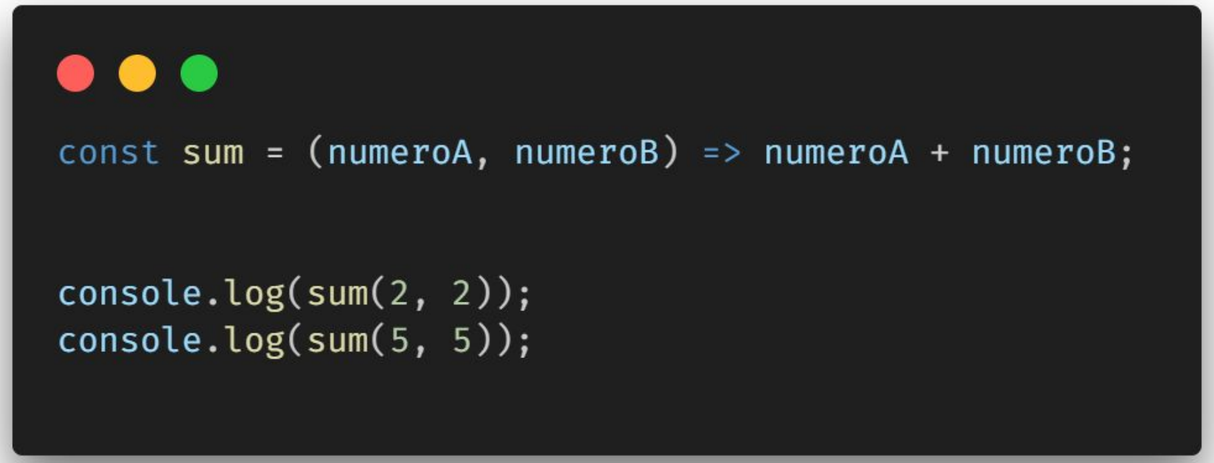


```
const sum = (numeroA, numeroB) => {
 return numeroA + numeroB;
};

console.log(sum(2, 2));
console.log(sum(5, 5));
```

guayerd

Ventaja de la función de flecha

Si en el cuerpo de la función sólo realizamos una tarea podemos omitir las llaves y el return. El retorno de la función, en este caso, es implícito.



```
const sum = (numeroA, numeroB) => numeroA + numeroB;


console.log(sum(2, 2));
console.log(sum(5, 5));
```

guayerd

¡Manos a la obra!

guayerd

SPRINT 2

Ejercicio 0010

CL10

Crear en un script una función que pregunte día, mes y año y devuelva en que estación climática del año te encuentras.

El resultado de la invocación a la función asignarlo a una variable y luego hacer un alert de la variable.

SPRINT 2

Ejercicio XOX

CL10

Cálculo de beca estudiantil: Crear en un script una función que reciba al menos el promedio escolar e ingresos familiares.

Te de una beca completa si tienes un promedio de al menos 9, tienes ingresos de no más de US$1000 o eres sobrino de los directivos.

Si tienes 7 o más e ingresos no mayores a US$500.

Caso contrario, no se otorga la beca.

guayerd

retro

¿Cómo nos fué?

¿Qué cosas no quedaron claras y necesitamos repasar la próxima?