Funciones

miércoles, 16 de octubre de 2024

21:45

|  |  |  |
| --- | --- | --- |
| function crearUsuario(email, name){  return{  email,  name,  }    let user1 = crearUsuario ('Sandra', 'sandra@garcia')  let user1 = crearUsuario ('Lucas', 'lucas@garcia') | FUNCIÓN            VARIABLE | Esto nos crea dos objetos simples  1 -> Sandra, sandra@garcia    2 -> Sandra, lucas@garcia |
| function Usuario(email, name) {  this.email = email;  this.name = name; }  Usuario.prototype.saludar = function() {  console.log("Hola, soy " + this.name); };  let user = new Usuario('ana@example.com', 'Ana'); user.saludar(); // Hola, soy Ana | FUNCIONES CONSTRUCTORAS | Crea instancias de objetos con métodos |
| console.log (Usuario.name)  console.log (Usuario.lenght) | FUNCIONES PREDEFINIDAS | Son propiedades o métodos que están disponibles solo dentro de un objeto o tipo específico, como Function.name o Array.length. Son funciones o propiedades que JavaScript asigna por defecto a ciertos objetos o clases. |
|  | FUNCIONES INCORPORADAS | Son funciones que vienen incluidas en el lenguaje y normalmente pertenecen a objetos como Math, Array, Date, etc., y están diseñadas para realizar tareas específicas (como operaciones matemáticas, fechas, conversiones, etc.). |
|  | FUNCIONES GLOBALES | Son funciones que puedes utilizar en cualquier parte de tu código sin tener que invocar un objeto en particular. Se pueden llamar directamente porque pertenecen al ámbito global. |

|  |  |  |
| --- | --- | --- |
| // Función tradicional  function suma(a, b) {  return a + b;  }    // Función flecha  const suma = (a, b) => a + b; | FUNCIONES FLECHA | No se pueden usar como constructores  No tienen propiedades arguments  **Uso con Métodos de Array** |
| const personas = [  { nombre: "Alice", edad: 25 },  { nombre: "Bob", edad: 30 },  { nombre: "Charlie", edad: 35 }  ];    const mayoresDe30 = personas.filter(persona => persona.edad > 30);    const nombres = personas.map(persona => persona.nombre);    console.log(mayoresDe30);  console.log(nombres); | EJEMPLO            Usando una función flecha con filter    Usando una función flecha con map    // [{ nombre: "Charlie", edad: 35 }]  // ["Alice", "Bob", "Charlie"] |  |

|  |  |  |
| --- | --- | --- |
| **Característica** | **Funciones Incorporadas** | **Funciones Globales** |
| **Definición** | Funciones integradas en JavaScript que no requieren importación externa. | Funciones disponibles en todo el entorno de ejecución sin referencia a objetos. |
| **Ámbito** | Dependientes de un objeto o tipo, pero universales y accesibles. | Disponibles en todo el código, en cualquier lugar del entorno de ejecución. |
| **Acceso** | Usadas directamente, asociadas a objetos como Math, Array, etc. | Se pueden invocar directamente sin referencia a ningún objeto. |
| **Ejemplo de uso** | Math.max(1, 2, 3) - Devuelve el valor máximo entre los números dados. | isNaN("abc") - Devuelve true porque "abc" no es un número. |
|  | Math.random() - Devuelve un número aleatorio. | parseFloat("3.14") - Convierte una cadena en un número de punto flotante. |
|  | Date.now() - Devuelve el timestamp actual en milisegundos desde el 1 de enero de 1970. | setTimeout(() => {}, 1000) - Ejecuta una función después de un tiempo especificado. |
| **Ejemplos concretos** | parseInt("10") - Convierte una cadena en un número entero. | isFinite() - Verifica si un valor es un número finito. |
|  | JSON.parse() - Convierte un string en un objeto JSON. | eval() - Evalúa una cadena como código JavaScript. |
| **Uso Común** | Manipulación de números, fechas, arreglos y JSON. | Tareas comunes que no están vinculadas a un objeto, como conversión de tipos o control del tiempo. |
| **Pertenecen a** | Ligadas a objetos como Math, Date, JSON. | No pertenecen a ningún objeto, disponibles globalmente. |
| **Necesidad de Objeto** | Usualmente necesitan un objeto como Math, Date, etc., pero no siempre. | No necesitan ningún objeto para ser invocadas. |
| **Dependen de** | Clases u objetos de JavaScript (Math, Date, JSON). | Del entorno global del lenguaje JavaScript. |