Imagina

Curso de Javascript/TS

Enrique de la Calle Santa Ana

7 Febrero 2,024

Índice

Introducción Javascript

Introducción a Javascript

- Algunos enlaces
  - https://developer.mozilla.org/es/docs/Web/JavaScript
  - https://es.javascript.info/
  - https://desarrolloweb.com/manuales/manual-javascript.html
  - https://javascript.espaciolatino.com/index.htm
  - https://www.arkaitzgarro.com/javascript/
  - Tipado Variables https://www.aluracursos.com/blog/tipado-dinamico-con-javascript

Breve Historia del lenguaje Javascript

Preparación del Taller

Preparamos nuestro entorno de trabajo / taller

Creación de proyecto

- Explicación general sobre nodejs
  - Nodejs es la versión javascript para servidor/ámbito local
  - Se aprovecha de la gestión no “bloqueante” de JS
  - NodeJs puede ejecutar código JS sin navegador ni ref html
- Explicación general sobre npm , nvm y npx
  - npm instalador de paquetes , alternativas yarn , etc
  - nvm múltiples versiones de nodejs
- Aprendemos a crear un core para nuestro proyecto
  - Decidimos la carpeta (c:\github\imaginajs)
  - Creo carpeta 01.-Proyecto_Base
  - npm init, y generara un package.json
  - Generamos un index tipo console.log(“hola mundo”);
  - En package.json , capitulo scripts “ejecuta”:”node test.js”
  - Modo terminal : npm run ejecuta
- Explicación general sobre nodemon
  - Es un bot que analiza los cambios de nuestro js y recompila en nodejs

Creación de proyecto

- Depuracion proyecto
  - Sintaxis en VSC
  - Use extrict;
  - Uso de console.log(…), console.warn(), console.error(), console.clear()
  - Parando la ejecución con VSC
- Pendientes
  - Nodemon: https://www.npmjs.com/package//nodemon
  - Babel: https://babeljs.io/docs/
  - Webpack https://webpack.js.org/

Especificaciones del lenguaje

- **Palabras reservadas**
  - Abstract boolean break byte,case catch char class const continue debugger default delete do double else enum export extends false final finally float for function Goto if implements import in instanceof int interface long native new null package private protected public return short static super switch synchronized this throw throws transient true try typeof var volatile void while with

Gramatica

- use strict // Nota: fuerza a que escribamos menos laxo (Lo aplicamos mas adelante)
- Comentarios
  - // Una sola linea
  - /* ….. */ Múltiples Lineas
  - //TODO y //FIXME (Indicaciones mias sobre rehacer o revisar)

- Tipo de sentencias
  - Asignacion: Donde se guarda el valor en una variable
  - Invocacion: Donde se llama a funciones / módulos / procedimientos
  - Control: donde se controla la ejecución o no de otras sentencias
  - La sentencia termina con un “;”
  - La invocación se realiza con el nombre de la función pasando parámetros entre paréntesis
  - JS admite Constantes , valores que solo pueden asignarse una vez
- Buenas practicas (Recomendaciones)
  - Elaborar el algoritmo antes
  - Divide y venceras , crea código lo mas pequeño posible

Gramatica

- Comenta
  - Comenta tu código para que otras personas puedan entenderlo
  - Comenta también el tipo de datos a utilizar y sus manipulaciones
- Codigo autodocumentado
  - Utiliza nombres de variables y funciones que sean relativas a lo que quieres hacer
- Identacion
  - Intenta que el código sea claro y pueda agruparse
- Variables
  - Intenta usar el modo estricto e inicializa las variables, no reuses nombres
- Variables globales
  - Minimiza
- Ley de Murphy
  - Si tienes dudas en el fallo de una parte , reescribe el código
- Refactoriza
  - Pasa por tu código varias veces para mejorarlo

Ámbito variables

- Var inicializa la variable
  - Se puede volver a usar (se debe evitar)
  - El ámbito es visible desde que se declara
- Let
  - Similar a var pero no permite redeclaracion
  - Su ámbito es la estructura de llaves (bloque)
- Const
  - Una vez definida , no se puede volver a asignar
- Tips
- Las var definidas en el html en la cabecera head son de variables globales
- Podemos crear una variable global si la asignamos a la variable “window”

  - P.e. window.miGlobal = “1.0”

Tipos de datos

- El tipado inferido quiere decir que no es necesario declarar el tipo de variable al inicio
  - En programación podemos usar typeof para saber el tipo de dato
- Var variable1 ; su valor original es “undefined”
- Var variable1 = 123 ; console.log(typeof variable1) = number
- Var variable2 = “123” ; console.log(typeof variable2) = string;
- Console.log(“la suma es “, variable1 + variable 2) = CONCATENACION
- Tips: console.log("suma ", 0.1 + 0.2) = 0.3000000000004
- console.log ((0.1 + 0.2).toFixed(2)) = 0.30

- Operadores Aritmeticos
  - Asignacion(=)
  - Suma(+) Resta(-) Producto(*) Cociente(/) Resto o Modulo (%) Incremento(++) Decremento(--)
  - Compuestos (+= , -= , *= , %=)
  - Delete (Operador unario , para eliminar elementos de un array o propiedades de un objeto
  - New (Instanciacion de objeto)
  - Typeof devuelve el tipo de dato

Operadores

- Operadores Binarios
  - Complemento(~) Negación Binaria
  - << y >> Desplazamiento bits a derecha e izquierda
  - AND (&&) OR (|) XOR(^)
  - New (Instanciacion de objeto)
- Operadores Logicos
  - NOT (!)
  - AND (&&)
  - OR (||)
- Operadores Relacionales
  - > , >= Mayor que o Mayor o Igual
  - <, <= Menor que o Menor o Igual
  - == , === , Igual o Idénticos (Idénticos == clase no solo su valor)
- Operadores Ternarios
  - ?? , asignación de valor aunque sea null (coalescencia)

Tipos de Variables

- Nativas o Primitivas
  - Undefined (Aun no se ha definido)
  - Boolean (true o false)
  - Number y BigInt (nº coma flotante) , Nan , +Infitity , -Infinity
  - String (Cadenas)
  - Symbol (átomo , valor primitivo e inmutable , es la clave de prop. Del objeto)
  - Null
- Object
  - Objetos, propiedades y métodos
- Date
  - Tratamiento de fechas
- JSON
  - Datos estructurados , JSON.parse() , JSON.stringify()
- Colecciones, Listas , Arrays , arreglos
  - Colección de datos de longitud variable , tratamiento

Flujos del programa

- Condicionales
  - If , else , end
  - Condicional simple (cond)?valor1:valor2
- Selección
  - switch
- Bucles
  - For: (for(var i = 0 ; i < 5 ; i++){})
  - While(condición){ ….}
  - Do{..}while(condicion)
  - for (item in ficha){} // propiedades objeto (object.keys)
  - for (item of ficha){} // iteracciones objeto (object.foreach)
  - == , === , Igual o Idénticos (Idénticos == clase no solo su valor)
- Rupturas
  - Break, continue

Especificaciones del lenguaje

- Paso por valor
  - // Asignación con variables
  - Let a= 20
  - Let b = a
  - b= 500
  - Console.log(“a=>“,a)
  - Console.log(“b=>“b)
- Paso por referencia
  - // Asignación con objetos
  - var person_1 = { name: "César", age: 25}
  - let person_2 = person_1
  - person_2.name = 'Tutancamon'
  - person_2.age = 69
  - console.log("person 1", JSON.stringify(person_1))
  - console.log("person 2", JSON.stringify(person_2))

Especificaciones del lenguaje

- Paso por referencia
  - // Asignación con array
  - let array_a = [10]
  - let array_b = array_a
  - array_b.push(999)
  - console.log("array_a => ", array_a)
  - console.log("array_b => ", array_b)// Asignación con objetos
- Iteradores
  - Iteradores y generadores - JavaScript | MDN (mozilla.org)
  - un **iterador** es un objeto que permite recorrer una colección y devolver un valor al terminar.
  - var it = crearIterador(["yo", "ya"]);
  - console.log(it.next().value); // 'yo'
  - console.log(it.next().value); // ‘ya’
  - console.log(it.next().done); // true
- Generadores
  - Similar a Iteradores pero pueden incluir en su constructor el estado completo de la funcion

Índice

Funciones

Funciones

- Bloque de código que puede devolver un valor
  - Function test(param1){return 123;}
  - Let var ret = test(“uno”);
  - Especial con el argumento spread
  - function sumar(…datos){};
  - // Datos implícitamente es un array
  - sumar(1,2) ; sumar(1,2,3)
  - Ámbito de las variables dentro de las funciones
- Funciones globales
  - Son funciones predefinidas en el sistema que no dependen de otros objetos
  - Eval(), isfinite(),isNaN(),parseFloat(),parseInt(),
  - encodeURI(),decodeURI(),encodeURIComponent(), decodeURIComponent()
- Funciones especiales
  - IIFE: son funciones autoejectuables: (function iniciando(params){console.log(“iniciando, params} (“args”)

Funciones

- Funciones especiales
  - Callbacks: funciones anónimas que se pueden usar para paso de parámetros
  - Ej. var test = function(num){ return num * 2}
  - function otra(num1, cb){ var oRet = cb(num1); return oRet }
  - Útiles en devoluciones de funciones y procesos finales de una función
- Funciones flecha
  - Simplificación de la nomenclatura , con la salvedad de this (no es el ámbito del creador)
  - Ej: const f1 = (param)=>{ return param * 2;}
  - Mas contraído const f1 = (param)=> param * 2;
- Sobrecarga de funciones (funciones con distintos parámetros)
  - Implicitamente no lo tiene , pero es tan abierto que no lo necesita
  - Parámetros variables (spread)
  - También hay una palabra clave “arguments” dentro de la función que es un array de parametros

Funciones Asincronas / Await / Async

- Funciones especiales
  - Programación asíncrona en JavaScript: Guía para Principiantes (freecodecamp.org)
  - Programacion sincrónica o paso a paso

Manejo Errores

Manejo Errores

- Bloque de try , catch , finally
    - Function test(param1){return 123;}
    - Let var ret = test(“uno”);
    - Especial con el argumento spread
    - function sumar(…datos){};
    - // Datos implícitamente es un array
    - sumar(1,2) ; sumar(1,2,3)
    - Ámbito de las variables dentro de las funciones

Modulos y Librerias

Módulos y librerias

- Require
  - Require es una función y como tal puede invocarse en cualquier sitio
  - La acción de require implica la carga del modulo
- Import
  - Pertenece a la nomenclatura del código (ES6) , por lo tanto su acción se produce en el transpilado
  - Es mas granular que require ya que solo hace uso de lo que necesita y no carga todo el modulo
- Export
  - Exportamos nuestras , clases , constantes, funciones mediante la directiva export
  - Si se marca una clase con el atributo default la importación es implícita

- Incorporacion de librerías de terceros
  - Incorporarmos librerías de terceros mediante npm en nodejs,
  - Podemos empaquetar nuestro código + lib terceros y usarlo en un <script src="bundle.js" />

Índice

POO Javascrip I , II

POO Javascript

- Links
  - https://www.freecodecamp.org/espanol/news/programacion-orientada-a-objectos-en-javascript-explicado-con-ejemplos/
  - https://developer.mozilla.org/es/docs/conflicting/Learn/JavaScript/Objects/Classes_in_JavaScript

- Prototype
  - Inicios era la manera de referenciar a un objeto
- ECMAScript / ES6
  - Class
  - Ubicación de las variables en las clases
  - Ubicación de los métodos y su ámbito

POO Javascript

- Instanciacion
  - Consegir una variable mediante el método new para poder acceder a las variables publicas y métodos de la clase
- Inicializacion
  - Los procesos que están dentro del método constructor, inicialización de variables llamasa a métodos iniciales , etc ECMAScript / ES6
- This
  - Modo de referencia a la clase dentro de ella misma
  - Analizar el ámbito tanto en los métodos como en los eventos
- Getter / setter
  - Uso y acceso a variables del objeto
  - Útil cuando el getter no es un dato puro (ej. Formato nombre y apellidos ,etc)

POO Javascript

- Herencia
  - Crear una jerarquía de clases con métodos comunes o abstraernos
  - Clave: inherits
  - Los procesos que están dentro del método constructor, inicialización de variables llamasa a métodos iniciales , etc ECMAScript / ES6
- Polimorfismo
  - La posibilidad de llamar al mismo nombre de método y ejecutar cosas diferentes según la clase
  - Ej: class user ; class userFile inherits user ; class userRest interits user
  - Existe el método login en las dos clases userFile y userRest pero cada uno ejecutara acciones distintas

Refactoring

Refactoring

- Como mejorar nuestro código
  - Consejos
  - https://dev.to/kelechikizito/javascript-refactoring-best-practices-1lld
  - https://www.youtube.com/watch?v=Wsiyad8L6zA

- Pautas
  - Una función , una tarea (atomizar el proceso)
  - Nombrado claro (nombres de función , nombres de variables etc)
  - Codigo legible y claro (si la simplicacion de sintaxis induce a no aclarar , no hacer)
  - Variables globales , las justas
  - Funciones puras (intentar que no dependan de variables exteriores)
  - Cuantas menos dependencias , mejor
  - Separar presentación / Dato (lógica del negocio)

Linting

Linting

- Links
  - https://www.freecodecamp.org/espanol/news/que-es-linting-y-eslint

- Install
  - Desde npm
  - Uso en VSC
- Aplicamos ESLINT a nuestro proyecto

Eventos, interactuando HTML

Eventos

- Elemento html Button
  - Binding enventos onclick
  - Entender Target
  - Cancelar eventos
- Elemento html Input
  - Binding enventos onchange
  - Revisar contenido cambiado

Persistencia

Persistencia

- Tratamiento de ficheros con NodeJS
  - Read , Write ,

- Entender el LocalStorage y SesiónStorage
  - Limitaciones
  - Debug

Imagina

Revisión de puntos

Revisión de puntos

- 1- INTRODUCCIÓN A JAVASCRIPT
  - ☐¿Qué es un lenguaje interpretado?
  - ☐¿Qué es JavaScript?
  - ☐¿Qué es ECMAScript y qué aporta?
  - ☐ Entornos de Desarrollo (IDEs) recomendados
- 2- LOS ARCHIVOS JAVASCRIPT
  - ☐ La extensión de los archivos JavaScript
  - ☐ Cómo ejecutar archivos JavaScript
  - ☐¿Qué es la documentación de un programa y buenas técnicas?
  - ☐ Tipos de comentarios
  - ☐ TODOs y FIXMEs
- 3- CREACIÓN DE PROYECTO BASE
  - ☐ Creación de proyecto Node
  - ☐ Introducción a Babel
  - ☐ Configuraciones necesarias para desarrollar con EcmaScript
  - ☐ Entendiendo la transpilación de código

Revisión de puntos

- 3- CREACIÓN DE PROYECTO BASE
  - ☐ Introducción a Nodemon
  - ☐ Configuraciones básicas de Nodemon
  - ☐ Ejecución de archivos con Nodemon
  - ☐ Introducción a WebPack
  - ☐ Configuraciones para builds con Webpack
  - ☐ Build del proyecto
  - ☐ Builds de desarrollo y producción
  - ☐ Analizando los bundles generados
  - ☐ Creación de scripts npm
  - ☐ Script de ejecución en desarrollo
- 4 SINTAXIS, VARIABLES Y PALABRAS RESERVADAS
  - ☐¿Qué son las Variables?
  - ☐¿Qué es el tipo de una variable?
  - ☐ El tipado inferido de JavaScript
  - ☐ Tipos Primitivos var, let y const

Revisión de puntos

- 4 SINTAXIS, VARIABLES Y PALABRAS RESERVADAS
  - ☐ Escritura dinámica
  - ☐ Haciendo uso del operador Typeof
  - ☐ Notación punto y coma, punto, corchetes y llaves
  - ☐ Diferencia entre null y undefined
- 5 CADENAS DE TEXTO, NÚMEROS Y LISTAS
  - ☐ String templates
  - ☐ Métodos para trabajar con cadenas de texto
  - ☐ Operadores principales
  - ☐ Métodos para trabajar con números
  - ☐ Particularidades de los números en JS
  - ☐ Tipos de datos en listas
  - ☐ Obtener elementos de una lista
  - ☐ Editar elementos de una lista
  - ☐ Eliminar elementos de una lista
  - ☐ Añadir elementos de una lista
  - ☐ Factor de propagación en listas
  - ☐ Métodos para trabajar con listas

Revisión de puntos

- 6 ESTRUCTURAS DE CONTROL
  - ☐¿Qué son las estructuras de control?
  - ☐ Las sentencias If, else
  - ☐ Sentencia Switch
  - ☐¿Qué es un bucle?
  - ☐¿Qué tipos de bucles existen y cómo se declaran?
  - ☐ El bucle For, Foreach y For ... in
  - ☐ El bucle while
  - ☐ El bucle Do While
  - ☐ Uso de continue y break
  - ☐¿Qué es el ámbito de un bucle?
  - ☐ Rotulando los bucles
- 7 DEBUGGING
  - ☐ ¿En qué consiste el debugging?
  - ☐ Beneficios del debugging
  - ☐¿Cómo depurar código desde el Navegador?

Revisión de puntos

- 7 DEBUGGING
  - ☐¿Cómo depurar código desde el IDE?
  - ☐ Buenas prácticas para debuging
  - ☐ Buenas prácticas con la función console.log y otras


Imagina