Undefined Academy

Bootcamp de JavaScript Full-stack

Semana 2, Clase 1:

Introducción a la programación y JavaScript

Metodología de trabajo

Formato de las sesiones  (1h 30m)

- `10 min` Revisión de ejercicio anterior
- `60 min` Presentación teórica-práctica
- `10 min` Preguntas y discusión

Herramientas del Bootcamp 🔧

- **Twitch**: https://undf.sh/en-vivo
- **Discord**: https://undf.sh/discord
- **Notion**: https://undf.sh/base

Agenda

1. **Introducción a la programación**
2. **Introducción a JavaScript**

Introducción a la programación

- ¿Qué es el código y los algoritmos?
- Sentencias y Expresiones en los lenguajes
- ¿Qué son los operadores en programación?
- Valores y tipos en el código
- Lenguajes interpretados y Compilados

Introducción a JavaScript

- Tipos y valores en JavaScript
- JavaScript y sus variables var, let y const
- Igualdades en JavaScript
- Falsy, Truthy y Nullish en JavaScript

Revisión de ejercicio anterior

Introducción a la programación

¿Qué es el código y los algoritmos?

¿Qué es el código y los algoritmos?

El código es un conjunto de **instrucciones** que le dicen a la computadora qué tareas realizar.

Estas instrucciones se llaman **algoritmos**, por lo que un código podría tener uno o más **algoritmos**.

¿Cuál es el algoritmo para hacer un café? ☕

Algoritmo para preparar café ☕

1. Verificar si hay suficiente agua en la cafetera.
2. Si no hay suficiente agua, llenar la cafetera con agua fresca.
3. Verificar si hay suficiente café en el filtro.
4. Si no hay suficiente café, añadir la cantidad necesaria al filtro.
5. Encender la cafetera y esperar a que termine de hacer el café.
6. Mientras se hace el café, preparar la taza o el vaso donde se servirá.
7. Cuando la cafetera haya terminado, apagarla y retirar el filtro y los posos de café.
8. Servir el café en la taza o el vaso preparado previamente.
9. Añadir leche, azúcar u otros ingredientes adicionales según sea necesario.
10. Disfrutar del café recién hecho.

Otros ejemplos de algoritmos

- Las instrucciones de una receta de cocina.
- Las instrucciones para armar un mueble[1].

1. Si el mueble no es de IKEA puede que incluso al seguir las instrucciones todo salga mal.

Algoritmos en diferentes areas

1. En matemáticas, se utilizan algoritmos para realizar cálculos y resolver problemas.
2. En física, se utilizan algoritmos para modelar y simular sistemas físicos.
3. En química, se utilizan algoritmos para predecir y modelar la estructura y el comportamiento de moléculas.
4. En biología, se utilizan algoritmos para analizar y procesar grandes cantidades de datos genómicos y proteómicos.

Sentencias y Expresiones en los lenguajes

- Una **expresión**[1] es una combinación de valores, variables, operadores y funciones que se evalúa para producir un resultado.

- Una **sentencia**[2] es una instrucción que realiza una acción en el programa.

  Las sentencias pueden cambiar el estado de las variables, realizar operaciones de entrada/salida, controlar el flujo de ejecución del programa, entre otras acciones

1. Expression
2. Statement

Ejemplos de expresiones:

1. `5 + 3` — esta expresión suma los números `5` y `3`, y devuelve el resultado `8`.
2. `x * y` — esta expresión multiplica dos variables `x` e `y`, y devuelve el resultado.
3. `"Hola " + nombre` — esta expresión concatena la cadena `"Hola "` con el valor de la variable `nombre` y devuelve la cadena resultante.

Ejemplos de sentencias:

```
si (x > 5) { imprimir("x es mayor que 5"); }
```

☝ esta sentencia comprueba si el valor de la variable `x` es mayor que `5` y si lo es, muestra un mensaje.

```
edad = obtener("¿Cuál es tu edad?");
```

☝ esta sentencia utiliza la *función* `obtener` para pedir al usuario que ingrese su edad y guardar el valor en la variable `edad`.

Sentencias y Expresiones en los lenguajes

Una sentencia puede estar hecha de una o más expresiones: `a = b * 2`

1. `2` es una **expresión de valor literal**[1].
2. `b` es una **expresión de variable**[2].
3. `b * 2` es una **expresión aritmética**[3].
4. `a = b * 2` es una **expresión de asignación**[4].

1. literal value expression
2. variable expression
3. arithmetic expression
4. assignment expression

¿Es `a = b * 2` una sentencia y expresión?

- Es una **sentencia** porque es una instrucción que realiza una acción, que es asignar el valor de `b * 2` a la variable `a`.
- También es una **expresión** porque produce un valor como resultado de su evaluación. En este caso, la **expresión** `b * 2` se evalúa como un número, que luego se asigna a la variable `a`.

¿Qué son los operadores en programación?

¿Qué son los operadores en programación?

Los operadores son **símbolos** o **palabras reservadas** que se utilizan para realizar *operaciones* o *comparaciones* entre **valores** o **variables**.

1. Operadores aritméticos

- `+`: suma dos valores.

  Ejemplo: `2 + 3`.

- `*`: multiplica dos valores.

  Ejemplo: `4 * 6`.

2. Operadores de asignación

- `=`: asigna un valor a una variable.

  Ejemplo: `x = 5`.

3. Operadores de comparación

- `==`: comprueba si dos valores son iguales.

  Ejemplo: `2 == 2`.

- `>=`: comprueba si el valor de la izquierda es mayor o igual que el valor de la derecha.

  Ejemplo: `5 >= 5`.

4. Operadores lógicos

- `&&`: devuelve verdadero si ambas expresiones son verdaderas.

  Ejemplo: `x > 5 && y < 10`.

- `!`: devuelve el valor opuesto de la expresión booleana.

  Ejemplo: `!(x > 5)`.

Valores y tipos en el código

Valores y tipos en el código

- Cuando necesitamos hacer matemáticas, queremos un **número**[1].

  Ejemplo: `6, 10, 0, -20, 0.55`

- Cuando necesitamos imprimir un valor en la pantalla, necesitamos una **cadena de texto**[2].

  Ejemplo: `"Hola", "10", "🤯"`

- Cuando necesitamos tomar una decisión en un programa, necesitamos un valor **booleano**[3].

  Ejemplo: `verdadero, falso`

1. Number
2. String
3. Boolean

Lenguajes interpretados y Compilados

Lenguajes interpretados y Compilados

La diferencia es el proceso de traducción del código fuente a un programa ejecutable.

- **Compilado**: usa un *compilador* para traducir el código fuente en lenguaje de máquina.

  Ejemplo: `C++`, `Java`

- **Interpretado**: usa un *intérprete* para traducir el código fuente a lenguaje de máquina en tiempo de ejecución.

  Ejemplo: `PHP`, `Python`

🍎 ¿JavaScript es Compilado o interpretado?

JavaScript 101

Tipos y valores en JavaScript

Valores Primitivos

Valores Primitivos [1]

Booleans (`true` y `false`)

Numbers (`-100`, `3.14`)

Strings (`"hola"`, `'abracadabra'`)

Symbols (`Symbol()`)

BigInts[2] (`10n`)

Undefined (`undefined`)

Null (`null`)

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#primitive_values
2. Big Intenger: número entero grande.

typeof

```
typeof true
typeof "hello"
typeof Symbol()
typeof 10n
typeof undefined
typeof null
```

typeof null [1]

```
typeof null === "object";
```

En un inicio la representación interna del valor nulo se definió como un puntero nulo cero (0x00) en la mayoría de los sistemas.

La etiqueta de tipo para los objetos también se estableció cómo cero en dicha versión.[2]

Se propuso una solución para ECMAScript pero fue rechazada.[3]

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/typeof#typeof_null
2. https://2ality.com/2013/10/typeof-null.html
3. https://web.archive.org/web/20160331031419/http://wiki.ecmascript.org:80/doku.php?id=harmony:typeof_null

Objetos & Funciones

Objetos & Funciones

- **Objects** (`{}`)
- **Functions** (`x => x * 2`)

typeof

```
typeof {}
typeof (function() {})
```

*Todo en JavaScript que no sea un **valor primitivo** es un **objeto**.* [1]

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Data_structures#objects

JavaScript y sus variables var, let y const

JavaScript y sus variables var, let y const [1]

Propiedad	var	let	const
scope (alcance)	función	bloque	bloque
re-asignación	✅	✅	❌
re-declaración	✅	❌	❌

1. https://www.youtube.com/watch?v=ojrvxYcKeYg

Igualdades en JavaScript

Igualdades en JavaScript

- Loose-equality `==`
- Strict-equality `===`

Falsy, Truthy y Nullish en JavaScript

Los valores "falsy" en JavaScript:

1. `false`: el valor booleano falso.
2. `0`: el número cero.
3. `-0`: el número cero negativo.
4. `0n`: el BigInt cero.
5. `""`: la cadena de texto vacía.
6. `null`: un valor nulo.
7. `undefined`: un valor no definido.
8. `NaN`: un valor que representa "Not a Number".

Cualquier otra cosa que no sea "falsy" es "truthy":

1. `true`: el valor booleano verdadero.
2. `1`: cualquier número diferente de cero se considera verdadero.
3. `"false"`: cualquier cadena de texto no vacía se considera verdadera.
4. `[]`: un arreglo vacío se considera verdadera.
5. `{}`: cualquier objeto vacío se considera verdadero.
6. `function() {}`: cualquier función definida se considera verdadera.
7. `new Date()`: cualquier objeto de fecha se considera verdadero.
8. `42n`: cualquier BigInt diferente de cero se considera verdadero.

Los valores "Nullish" son `null` y `undefined`

Recursos 📚

- https://learnprogramming.online/
- https://justjavascript.com/
- https://javascript.info/
- https://jgthms.com/javascript-in-14-minutes/
- https://www.youtube.com/watch?v=ojrvxYcKeYg

Calentamiento

Hugo, Paco y Luis tienen una cantidad desconocida de monedas cada uno.

Sabemos que Paco tiene el doble de monedas que Hugo y que Luis tiene 10 monedas más que Paco.

Si los tres juntos tienen un total de 85 monedas.

¿Cuántas monedas tiene cada uno?

```
// Asignamos la cantidad de
monedas de Hugo, este valor es el
que tienes que resolver.
let hugo = 0;

// Calculamos la cantidad de
monedas de Paco y Luis en función
de Hugo
let paco = 2 * hugo;
let luis = paco + 10;

// Sumamos las cantidades de
monedas de los tres amigos
let total = hugo + paco + luis;

if (total === 85) {
    console.log("Hugo: " + hugo)
    console.log("Paco: " + paco)
    console.log("Luis: " + luis)
}
```

Ejercicio

¿Cómo puedo implementar una expresión para verificar si un valor es un objeto?

Ejemplo: `typeof obj === "object"`

¿Preguntas?