UT3. JAVASCRIPT. INTRODUCCIÓN

LENGUAJES DE MARCAS 1ºDAW

VIRGINIA ZORNOZA

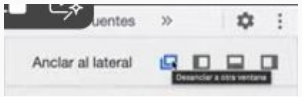
ÍNDICE

1. Navegador, about:blank, inspeccionar, tema oscuro, consola, desanclar

En el navegador que utilices, abres una pestaña y escribes “about:blank”

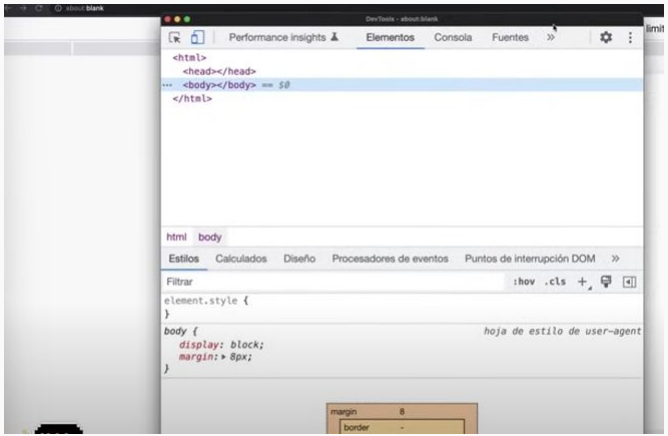
Botón derecho > inspeccionar

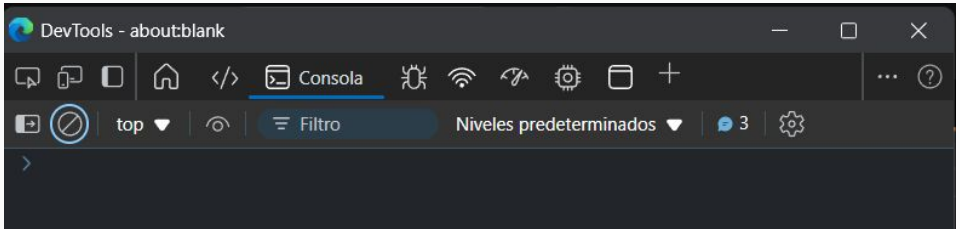
3 puntitos > desanclar



Consola

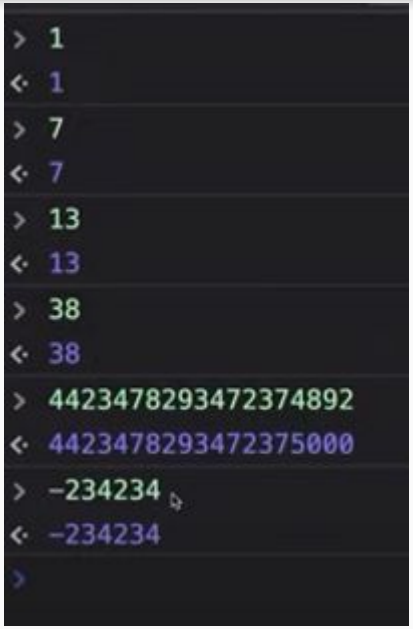
Aspecto oscuro (opcional)

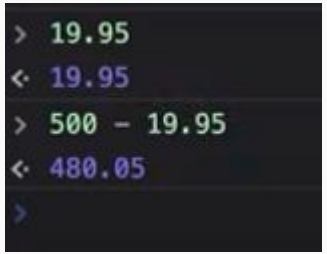




2. Tipos de datos: números, texto comillas, comilla invertida, booleanos

- Números y operadores





2. Tipos de datos: números, texto comillas, comilla invertida, booleanos

- Cadenas de texto

  De 3 maneras distintas

```
> "Miguel Ángel"
< 'Miguel Ángel'
> 'Miguel Ángel'
< 'Miguel Ángel'
> `Peluche de midudev`
< 'Peluche de midudev'
```

- Concatenar:

```
> 'Virginia' + ' ' + 'Zornoza'
< 'Virginia Zornoza'
```

- Operaciones acento invertido:

```
> `Peluche de midudev cuesta ${100 * 1.20}€`
< 'Peluche de midudev cuesta 120€'
```

2. Tipos de datos: números, texto comillas, comilla invertida, booleanos

- Booleano: true o false
- AND &&
- OR ||
- NOT !

```
> false || true
< true
> 2 > 3 || 3 > 2
< true
> "hola" === "adios" || "hasta luego" === "adios"
< false
```

```
> true && true && true
< true
> true && true && false
< false
> (2 === 2) && (4 === 4)
< true
> (2 === 2) && (4 === 4)
< false
>
```

```
> 3 > 2
< true
> 2 > 3
< false
> "Miguel" === "Migueeeel"
< false
> "Pera" !== "Manzana"
< true
> 2 >= 3
< false
> 2 >= 2
< true
```

3. Null y undefined

4. Variables (let) y constantes (const)

- **let** asigna un valor a un espacio de memoria, y se le puede asignar cualquier valor de los vistos antes.
- Expresiones: devuelven un valor
- Declaraciones: le digo que haga algo pero no me devuelve nada.

```
> 500 - 20
<· 480
```

```
> let finanzas = 500
<· undefined
```

- La variable guarda el valor,

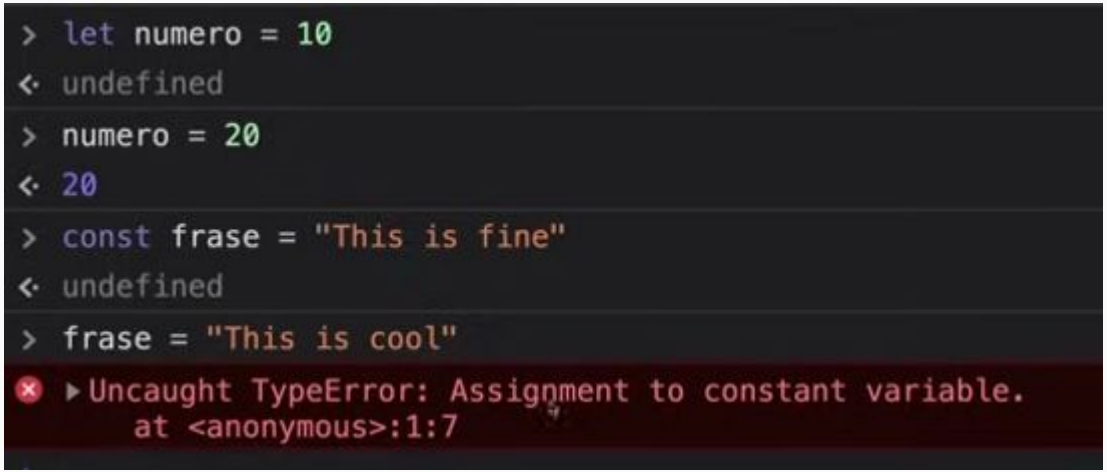
  se puede recuperar y cambiar

  (actualizar) su valor.

```
> finanzas
<· 300
> finanzas + 20 + 100
<· 420
> finanzas
<· 300
> finanzas = finanzas + 20 + 100
<· 420
```

4. Variables (let) y constantes (const)

- **Const**: no puedes reasignarle valor a esa variable.



5. Condiciones: if else, while y for

- **IF ELSE**: tome una decisión según una condición booleana.

```
> tenemosComidaDentro = false
<· false
> if (tenemosComidaDentro) {
    "cocinar"
  } else {
    "pitar con un sonido MUY FUERTE"
  }
<· 'pitar con un sonido MUY FUERTE'
```

- **IF ELSE IF**

```
> const cantidadDeDetergente = 100
<· undefined
> if (cantidadDeDetergente < 10) {
    "Detergente insuficiente"
  } else if (cantidadDeDetergente < 20)  {
    "Detergente esta casi casi"
  } else if (cantidadDeDetergente < 25) {
    "Te falta una chispita"
  } else {
    "Todo esta perfecto"
  }
<· 'Todo esta perfecto'
```

5. Condiciones: if else, while y for

- **WHILE:**

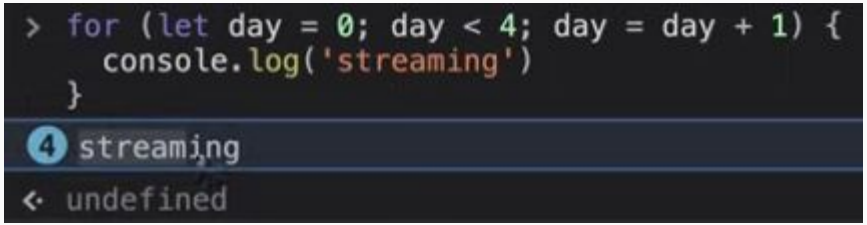
```
> let day = 0
<· undefined
> while (day < 4) {
    "streaming"
    day = day + 1
  }
<· 4
```

- FUNCIÓN **CONSOLE.LOG()**

```
> day = 0
  while (day < 4) {
    console.log("streaming")
    day = day + 1
  }
4 streaming
<· 4
```

5. Condiciones: if else, while y for

- **FOR (inicialización de la variable; condición; incremento variable)**



6. Incremento y decremento

- Incremento sufijo: lo incrementa después.
- Incremento prefijo: lo incrementa antes.
- Igual con decremento.

```
> personasEnBici = personasEnBici + 1
< 9
> ++personasEnBici
< 10
> personasEnBici++
< 10
> personasEnBici
< 11
> for (let day = 0; day < 4; day++) {
    console.log('streaming')
  }
```

```
> personasEnBici = personasEnBici + 1
< 14
> personasEnBici++
< 14
> personasEnBici
< 15
> ++personasEnBici
< 16
```

7. Comentarios // y /* */ multilínea

- Comentar el código (no todo. Es mejor usar variables explícitas)
- En una línea //
- En varias líneas /* */
- Enter sin ejecutar en consola: MAY + ENTER

```
> // Esta variable es
  // necesaria para...
  // seguir programando...
  //
```

```
> /* Este comentario
  es multilinea




  TAN LARGO COMO QUIERAS
  */
<- undefined
```

7. Comentarios // y /* */ multilínea

- **Ejercicio:**

Consejo: usar console.log y bucle

8. Function nombre(){ return} y ejecutarla nombre()

- **Función que cada vez que toque el peluche se descuentan 100€**

```
> let finanzas = 500
<- undefined
> finanzas = finanzas - 100
<- 400
> function tocarPeluche() {
    finanzas = finanzas - 100
  }
<- undefined
```

8. Function nombre(){ return} y ejecutarla nombre()

- **Función que cada vez que toque el peluche se descuentan 100€**
- **Ejecutar la función: tocarPeluche()**

Al poner () lo diferenciamos de una variable

```
> let finanzas = 1000

  function tocarPeluche() {
    console.log('tocarPeluche')
    finanzas = finanzas - 100
  }

  tocarPeluche()
  tocarPeluche
<· undefined
> finanzas
<· 900
```

```
> function tocarPeluche() {
    console.log('- tocar el peluche cuesta 100€')
    finanzas = finanzas - 100
    return finanzas
  }
<· undefined
> tocarPeluche()
  - tocar el peluche cuesta 100€
<· 800
```

8. Function nombre(){ return} y ejecutarla nombre()

- **Parametrizar la función**

```
> function tocarPeluche(coste) {
    console.log(`- tocar el peluche cuesta ${coste}€`)
    finanzas = finanzas - coste
    return finanzas
  }
```

- Al llamar a la función se le pasan **argumentos**

```
> tocarPeluche(20)
  - tocar el peluche cuesta 20€
< 480
> tocarPeluche(50)
  - tocar el peluche cuesta 50€
< 430
```

- Si no le paso valor, me devuelve NaN (Not a Number)

```
> tocarPeluche()
  - tocar el peluche cuesta undefined€
< NaN
```

9. Valores por defecto en funciones

- A los parámetros se les puede asignar un valor por defecto, para cuando no se les pasa valor

```
> function tocarPeluche(coste = 100) {
    console.log(`- tocar el peluche cuesta ${coste}€`)
    finanzas = finanzas - coste
    return finanzas
  }
< undefined
> tocarPeluche()
  - tocar el peluche cuesta 100€
< 900
```

10. BIBLIOGRAFÍA

VÍDEO DE MIDULIVE