Sé Programar 2

Taller de Programación I (Universidad de Buenos Aires)

Apéndice

Referencia rápida del lenguaje JavaScript

El lenguaje JavaScript es utilizado ampliamente para construir software en todo el mundo, siendo una de las principales tecnologías de la Web. En este capítulo sólo usamos una muy pequeña parte del mismo, que listamos a continuación:

Declaración de Funciones

A partir de la Le

A partir de la Lección 1: Funciones y tipos de datos

Las funciones en JavaScript se declaran mediante la *palabra clave* `function`, y su cuerpo va entre llaves `{` y `}`:

```
function nombreDeLaFuncion(parametro1, parametro2, parametro3) {
  return ...;
}
```

Toda función debe tener al menos un retorno, que se expresa mediante `return`.

Operadores matemáticos

A partir de la Lección 1: Funciones y tipos de datos

```
4 + 5
10 - 5
8 * 9
10 / 5
```

Operadores lógicos

A partir de la Lección 1: Funciones y tipos de datos

```
true && false
true || false
! false
```

Comparaciones

A partir de la Lección 1: Funciones y tipos de datos

```
// para cualquier tipo de dato
"hola" === "hola"
"hola" !== "chau"

// para números
4 >= 5
```

```
4 > 5
4 <= 5
4 < 5
```

Alternativa Condicional

A partir de la Lección 1: Funciones y tipos de datos

Los `if` s en JavaScript encierran la condición entre paréntesis y su cuerpo entre llaves:

```
if (hayPersonasEnEspera()) {
  llamarSiguientePersona();
}
```

Además, los `if` s pueden opcionalmente tener un `else` :

```
if (hayPersonasEnEspera()) {
  llamarSiguientePersona();
} else {
  esperarSiguientePersona();
}
```

Por último, podemos combinar varios `if` s para tomar decisiones ante múltiples condiciones:

```
if (hayPersonasEnEspera()) {
  llamarSiguientePersona();
} else if (elPuestoDebeSeguirAbierto()) {
  esperarSiguientePersona();
} else {
  cerrarPuesto();
}
```

Variables

A partir de la Lección 3: Variables y procedimientos

Las variables nos permiten *recordar* valores y se declaran mediante la palabra reservada `let` y se les da un valor inicial usando `=` :

```
let pesosEnMiBilletera = 100;
let diasQueFaltan
```

La mismas se asignan mediante `=` :

```
pesosEnMiBilletera = 65;
diasQueFaltanParaElVerano = 7;
```

En ocasiones las asignaremos usando el valor anterior:

```
pesosEnMiBilletera = pesosEnMiBilletera * 2;
diasQueFaltanParaElVerano = diasQueFaltanParaElVerano - 1;
```

La asignación anterior se puede compactar combinando el signo `=` y la operación:

```
pesosEnMiBilletera *= 2;
diasQueFaltanParaElVerano -= 1;
```

Repetición indexada

A partir de la Lección 7: Recorridos

Las listas pueden ser *recorridas*, visitando y haciendo algo con cada uno de sus elementos. Para ello contamos con la estructura de control `for..of` , que encierra su generador entre paréntesis (`(` y `)`) y su cuerpo entre llaves (`{` y `}`):

```
let patrimoniosDeLaHumanidad = [
  {declarado: 1979, nombre: "Parque nacional Tikal", pais: "Guatemala"},
  {declarado: 1983, nombre: "Santuario histórico de Machu Picchu", pais: "Perú"}
  {declarado: 1986, nombre: "Parque nacional do Iguaçu", pais: "Brasil"},
  {declarado: 1995, nombre: "Parque nacional de Rapa Nui", pais: "Chile"},
  {declarado: 2003, nombre: "Quebrada de Humahuaca", pais: "Argentina"}
]


let cantidadPatrimoniosDeclaradosEnEsteSiglo = 0;
for (let patrimonio of patrimoniosDeLaHumanidad) {
  if (patrimonio.declarado >= 2000) {
    cantidadPatrimoniosDeclaradosEnEsteSiglo += 1;
  }
}
```

Biblioteca simplificada

Biblioteca simplificada

Utilizamos una biblioteca de funciones inspirada en la que ya viene con JavaScript, pero simplifiacada para que sea más sencilla y segura de usar. A continuación listamos las

principales funciones que se pueden usar, indicando el equivalente *real* en JavaScript cuando corresponda.

`longitud(unString)`

> A partir de la Lección 1: Funciones y tipos de datos
> Versión simplificada de `length`

Uso:

```
> longitud("hola")
4
```

`convertirEnMayuscula(unString)`

> A partir de la Lección 1: Funciones y tipos de datos
> Versión simplificada de `toUpperCase`

Convierte un `unString` en mayúsculas:

```
> convertirEnMayuscula("hola")
"HOLA"
```

`comienzaCon(unString, otroString)`

> A partir de la Lección 1: Funciones y tipos de datos
> Versión simplificada de `startsWith`

Dice si `unString` empieza con `otroString`:

```
> comienzaCon("aprendiendo a programar", "aprendiendo")
true

> comienzaCon("aprendiendo a programar", "aprend")
true
```

```
> comienzaCon("aprendiendo a programar", "programar")
false


> comienzaCon("aprendiendo a programar", "tomar el té")
false
```

imprimir(unString)

A partir de la Lección 3: Variables y procedimientos

Versión simplificada de `console.log`

Imprime por pantalla `unString` :

```
> imprimir("¡estoy imprimiendo!")
¡estoy imprimiendo!
```

tirarDado()

A partir de la Lección 3: Variables y procedimientos

Devuelve al azar un número entre 1 y 6:

```
> tirarDado()
5
> tirarDado()
1
> tirarDado()
2
```

listasIguales(unaLista, otraLista)

A partir de la Lección 5: Listas

```
> listasIguales([1,4,7], [1,4,7])
true

> listasIguales([1,4,7], [1,4,8])
false
```

`longitud(unaLista)`

A partir de la Lección 5: Listas

- `length` de listas

Nos dice cuan largo es `unaLista` :

```
> longitud([true, false, false, true])
4
> longitud([5, 6, 3])
3
```

`agregar(unaLista, unElemento)`

A partir de la Lección 5: Listas

Versión simplificada de `push`

Inserta `unElemento` al final de `unaLista` . Este es un procedimiento que no devuelve nada pero modifica a `unaLista` :

```
> let cancionesFavoritas = ["La colina de la vida", "Zamba por vos"]
// agrega el elemento "Seminare" a la lista  cancionesFavoritas
> agregar(cancionesFavoritas, "Seminare")
// ahora la lista tiene un elemento más:
> cancionesFavoritas
["La colina de la vida", "Zamba por vos", "Seminare"]
```

`remover(unaLista, unElemento)`

A partir de la Lección 5: Listas

Quita `unElemento` de `unaLista` . Este es un procedimiento que no devuelve nada pero modifica a unaLista:

```
> let listaDeCompras = ["leche", "pan", "arroz", "aceite", "yerba"]
// removemos "pan"
> remove(listaDeCompras, "pan")
// "pan" ya no está en lista de compras
> listaDeCompras
```

```
["leche", "arroz", "aceite", "yerba"]
```

`posicion(unaLista, unElemento)`

A partir de la Lección 5: Listas

Versión simplificada de `indexOf`

Nos dice en qué posición se encuentra `unElemento` dentro de `unaLista`. Si el elemento no está en la lista, devuelve `-1`

```
> let premios = ["dani", "agus", "juli", "fran"]
> posicion(premios, "dani")
0
> posicion(premios, "juli")
2
> posicion(premios, "feli")
-1
```

Bibliografía complementaria

- https://developer.mozilla.org/es/docs/Web/JavaScript
- https://es.javascript.info/