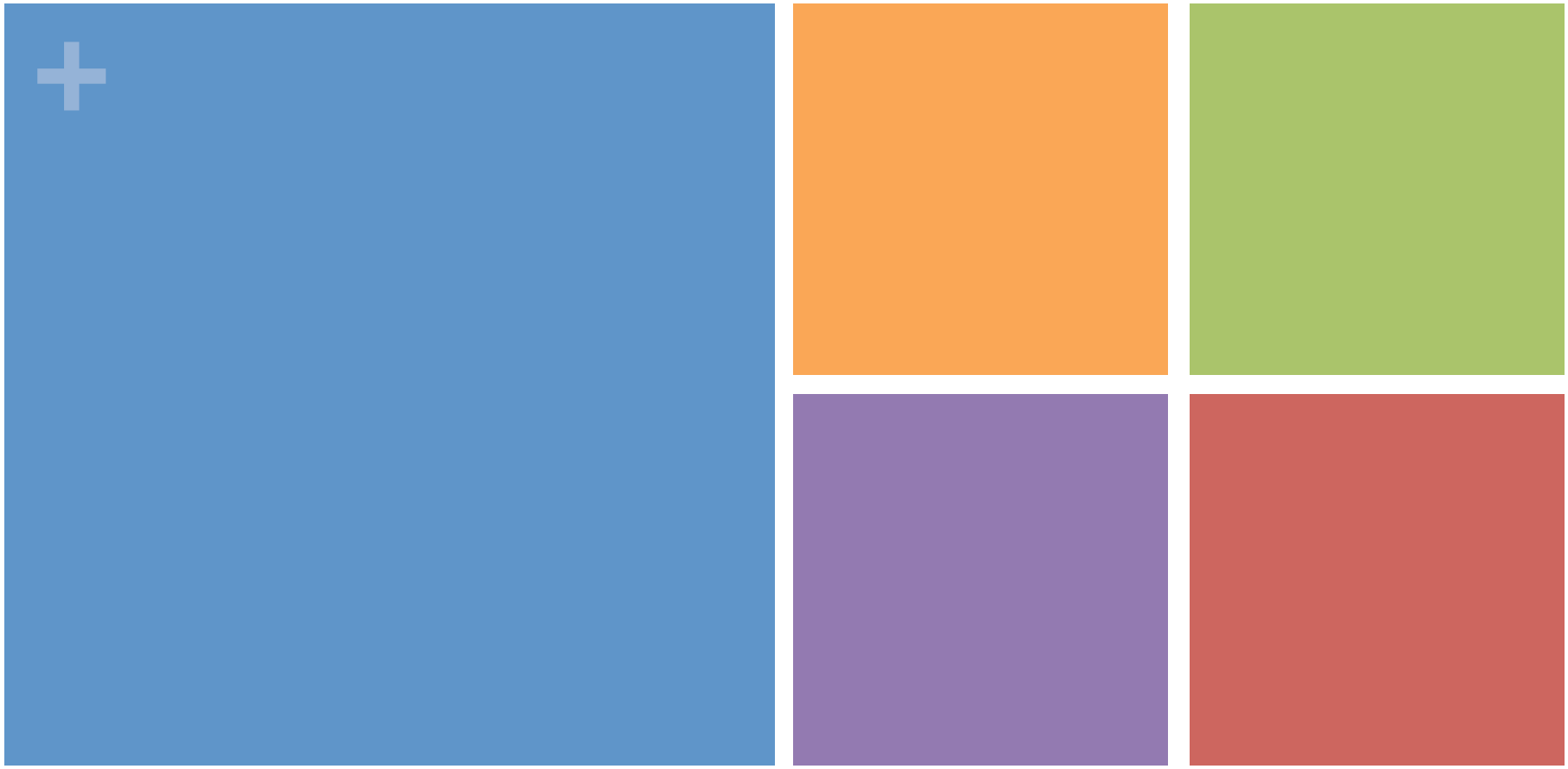

Fundamentos de Programación

+

JavaSript

Introducción

Es un lenguaje interpretado, orientado a objetos.
Inicialmente conocido como lenguaje de scripts para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js.

for

Sintaxis:

```
for([expresion-inicial]; [condicion]; [expresion-final])
        sentencia
```

Ejemplo:

```
for (var i = 0 ; i < 10;  i++) {
  n = i+1;
  funcion(n);
}
```

while

Sintaxis:

while (condición) sentencia;

Ejemplo:

```
var i = 0;
while ( i<9 ){
  i = i+1;
  funcion(i);
}
```

do…while

Sintaxis:

do sentencia while (condición);

Ejemplo:

```
do {
   i = i+1;
   funcion(i);
} while (i < 5);
```

if

Sintaxis:

if (condición) sentencia1 [else sentencia2]

Ejemplo:

```
i = i – n;
if (i <0)
   funcion(i);
else if (i>2){
   funcion2(i);
}
else
   funcion3(i)
```

switch

Sintaxis:

```
switch (expresion) {
  case valor1:
                    //Sentencia
                    break;
case valorn:
                    //SentenciaN
                    break;
  default:
                    //Sentencia Default
}
```

switch

Ejemplo:

```
switch (pedido) {
  case ”Combo":
    total = total + 180;
    break;
  case ”Hamburguesa
    total = total + 110;
    break;
}
```

function

Sintaxis:

```
function nombre([parametro1] [, parametroN])
{sentencias}
```

Ejemplo:

```
function reglade3(x,y,z) {
        return x*y/z;
}
```

throw

Sintaxis:

throw expresion;

try…catch

Sintaxis:

try {sentencia}
[catch (var_excepcion if condicion) {
Sentencia1}
catch (var_excepcion if condicion2) {
Sentencia2}]
[finally {sentencia2
}]

try…catch

Ejemplo:

```
try {
   funcion();
} catch (e if e instanceof RangeError) {
   salida = " Error tipo 1";
} catch (e if e instanceof EvalError) {
    salida = " Error tipo 2";
} catch (e) {
    salida = " Otro tipo de Error";
}
finally {
    console.log(salida);
}
```

var

Sintaxis:

var varnombre [= valor1 [, varnombre2 [= valor2 ...[, varnombreN [= valorN ']]]]];

Ejemplo:

```
var pedido;
var i=2, j=3;
```

Objetos

- Array
- Boolean
- Date
- JSON
- Map
- NaN
- Number
- Object
- Set
- String

Propiedades y funciones para texto

length

Devuelve el número de caracteres que la forma

```
var texto = "Hola Mundo";
var largo = texto.length; // largo = 10
```

+ o concat()

Concatenar varias cadenas de texto

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";
var mensaje = mensaje1 + mensaje2;
// mensaje = "Hola Mundo"

var mensaje2 = mensaje1.concat(" Mundo");
// mensaje2 = "Hola Mundo"
```

toUpperCase()

Transforma los caracteres en mayúsculas

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase();
// mensaje2 = "HOLA"
```

toLowerCase()

Transforma los caracteres de la cadena minúsculas:

```
var mensaje1 = "HolA";
var mensaje2 = mensaje1.toLowerCase();
// mensaje2 = "hola"
```

charAt(posicion)

Obtiene el carácter en la posición indicada:

```
var mensaje = "Hola";
var letra = mensaje.charAt(0);
// letra = H
letra = mensaje.charAt(2);
// letra = l
```

indexOf(caracter)

Calcula la posición del carácter indicado dentro de la cadena. Si se encuentra varias veces, se devuelve la primera posición desde la izquierda. Si no contiene el carácter, la función devuelve -1.

```
var mensaje = "Hola";
var posicion = mensaje.indexOf('a');
// posicion = 3
posicion = mensaje.indexOf('b');
// posicion = -1
```

lastIndexOf()

Calcula la última posición del carácter indicado. Si la cadena no contiene el carácter, la función devuelve el valor -1.

```
var mensaje = "Hola";
var posicion = mensaje.lastIndexOf('a');
// posicion = 3
posicion = mensaje.lastIndexOf('b');
// posicion = -1
```

substring(inicio, final)

Extrae una porción de una cadena de texto. El segundo parámetro es opcional. Si sólo se indica el parámetro inicio, la función devuelve la parte de la cadena original correspondiente desde esa posición hasta el final.

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2);
// porcion = "la Mundo"
porcion = mensaje.substring(5);
// porcion = "Mundo"
porcion = mensaje.substring(7);
// porcion = "ndo”
```

substring(inicio, final)

Si se indica un inicio negativo, se devuelve la misma cadena original.

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(-2);
// porcion = "Hola Mundo”
```

substring(inicio, final)

Cuando se indica el inicio y el final, se devuelve la parte de la cadena original comprendida entre la posición inicial y la inmediatamente anterior a la posición final

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(1, 8);
// porcion = "ola Mun"
porcion = mensaje.substring(3, 4);
// porcion = "a"
```

split

Convierte una cadena de texto en un array de cadenas de texto. La función parte la cadena de texto a partir del carácter separador indicado.

```
var mensaje = "Hola Mundo, soy una cadena de texto!";
var palabras = mensaje.split(" ");
// palabras = ["Hola", "Mundo,", "soy", "una", "cadena",
"de", "texto!"];
var palabra = "Hola";
var letras = palabra.split("");
// letras = ["H", "o", "l", "a"]
```

Propiedades y funciones para arreglos

length

Calcula la cantidad de elementos en un array.

```
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length;
// numeroVocales = 5
```

concat()

Concatena elementos de varios arrays

```
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6);
 // array2 = [1, 2, 3, 4, 5, 6]
array3 = array1.concat([4, 5, 6]);
 // array3 = [1, 2, 3, 4, 5, 6]
```

join(separador)

Une los elementos de un array en una cadena de texto. Para unir los elementos se utiliza el carácter separador.

```
var array = ["hola", "mundo"];
var mensaje = array.join("");
// mensaje = "holamundo"
mensaje = array.join(" ");
// mensaje = "hola mundo"
```

pop()

Elimina el último elemento del array y lo devuelve.
El array original se modifica.

```
var array = [1, 2, 3];
var ultimo = array.pop();
// ahora array = [1, 2], ultimo = 3
```

push()

Agrega un elemento al final del array. El array original se modifica.

```
var array = [1, 2, 3];
array.push(4);
// ahora array = [1, 2, 3, 4]
```

shift()

Elimina el primer elemento del array y lo devuelve.

```
var array = [1, 2, 3];
var primero = array.shift();
// ahora array = [2, 3], primero = 1
```

unshift()

Añade un elemento al principio del array.
El array original se modifica y aumenta su longitud en 1 elemento.

```
var array = [1, 2, 3];
array.unshift(0);
// ahora array = [0, 1, 2, 3]
```

reverse()

Modifica un array invirtiendo su posición original:

```
var array = [1, 2, 3];
array.reverse();
// ahora array = [3, 2, 1]
```

Funciones para números

NaN (Not a Number)

El valor NaN indica un valor numérico no definido (0/0).

```
var num1 = 0;
var num2 = 0;
alert(num1/num2);
 // se muestra el valor NaN
```

isNaN()

Dice si el valor es NaN.

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
  alert("No está definido el valor");
}
else {
  alert("Reusltado => " + numero1/numero2);
}
```

Infinity , –Infinity

Valor numérico +infinito y -infinito.

```
var numero1 = 10;
var numero2 = 0;
alert(numero1/numero2);
// se muestra el valor Infinity
```

toFixed(digitos)

Devuelve el número con los decimales indicados por el parámetro y redondea dn caso de ser necesario.

```
var numero1 = 4564.34567;
numero1.toFixed(2);
// 4564.35
numero1.toFixed(6);
// 4564.345670
numero1.toFixed();
// 4564
```