



Estructuras de Datos y Objetos

De Datos atómicos a Objetos

Números, Booleanos y Strings

Podemos construir objetos más
complejos a partir de ellos

- Objeto #1: Arrays

```
let listaDeNumeros = [2, 3, 5, 7, 11];  
console.log(listaDeNumeros[2]);  
console.log(listaDeNumeros[0]);  
console.log(listaDeNumeros[2 - 1]);
```

- Los elementos de un array se almacenan como propiedades del mismo, en este caso siendo números los nombres de las propiedades

01 Arrays.html

Métodos sobre Arrays

- Los valores string y arrays, además de la propiedad longitud, tienen una serie de propiedades que contienen valores:

```
let doh = "Doh";  
console.log(typeof  
doh.toUpperCase);  
console.log(doh.toUpperCase());
```

- Llamamos métodos a las propiedades que contienen funciones

Algunos métodos para manipular arrays – la pila

```
let secuencia = [1, 2, 3];  
secuencia.push(4);  
secuencia.push(5);  
console.log(secuencia);  
// → [1, 2, 3, 4, 5]  
console.log(secuencia.pop());  
// → 5  
console.log(secuencia);  
// → [1, 2, 3, 4]
```

Trabajando con objetos

```
let descriptions = {  
  trabajo: "Fue a trabajar",  
  "arbol tocado": "Tocó un árbol"  
};
```

Propiedad cuyo nombre
tiene espacios

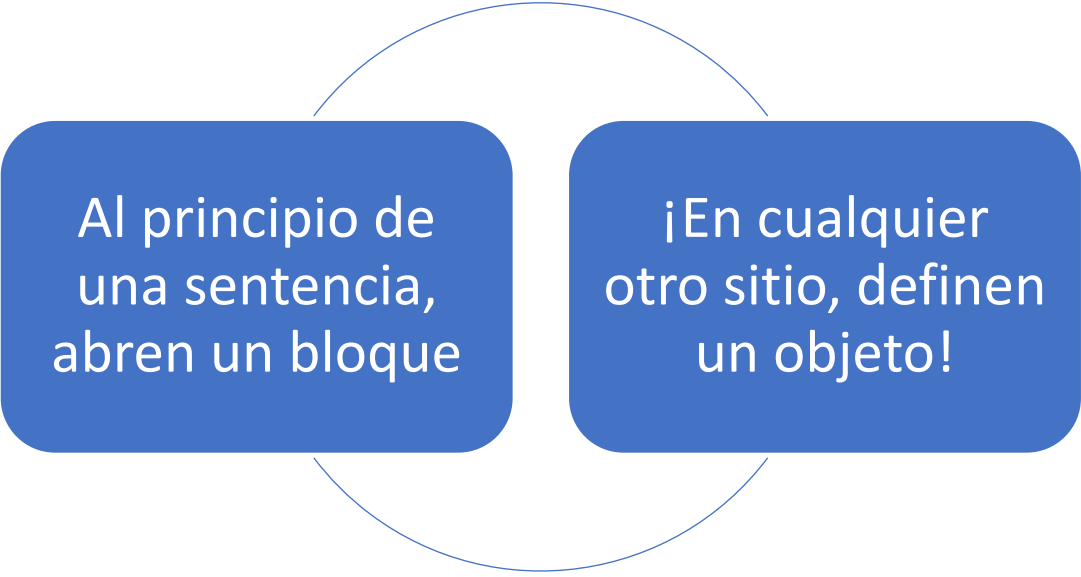
Propiedades

Valores de la
Propiedad
evento

- Los objetos son colecciones arbitrarias de propiedades
- Se pueden crear simplemente con {}

```
let dia = {  
  ardilla: false,  
  eventos: ["trabaja", "toca  
    arbol", "pizza", "corre"]  
};  
console.log(dia.ardilla);  
// → false  
console.log(dia.lobo);  
// → undefined  
dia.lobo = false;  
console.log(dia.lobo);  
// → false
```

Función de las { } en JS



Al principio de una sentencia, abren un bloque

¡En cualquier otro sitio, definen un objeto!

Borrando elementos de un objeto *(de uso inhabitual)*

```
let unObjeto = {izquierda: 1, derecha: 2};
console.log(unObjeto.izquierda);
// → 1
delete unObjeto.izquierda;
console.log(unObjeto.izquierda);
// → undefined
//¿Pertenece propiedad al objeto?
console.log("izquierda" in unObjeto);
// → false
console.log("derecha" in unObjeto);
// → true
```

Copiando propiedades de un objeto a otro

```
let objetoA = {a: 1, b: 2};  
Object.assign(objetoA, {b: 3, c: 4});  
console.log(objetoA);  
// → {a: 1, b: 3, c: 4}
```

diario, Array que contiene un objeto que tiene dos propiedades, ¡una de ellas es un array!

Los arrays: objetos especializados

```
let diario = [  
  {eventos: ["trabaja", "toca arbol", "pizza", "corre", "television"],  
    ardilla: false},  
  {eventos: [" trabaja ", "helado", "coliflor", "lasaña", "toca árbol", "lavar los diente"],  
    ardilla: false},  
  {eventos: ["finde", "bici", "descanso", "cacahuetes", "cerveza"],  
    ardilla: true},  
  /*... */  
];
```

```
typeof diario → "object"
```

05 Arrays de Objetos

Copiando objetos pasan cosas como en las funciones



1. `let objeto1 = { valor: 10 };`
2. `let objeto2 = { valor: "Me van a eliminar" };`
3. `objeto2 = objeto1;`
4. `let objeto3 = { valor: 10 };`
5. `console.log(objeto1 == objeto2);`
6. `console.log(objeto1 == objeto3);`
7. `objeto1.valor = 15;`
8. `console.log("Objeto 2 vale lo mismo que el 1! " + objeto2.valor);`
9. `console.log(objeto3.valor);`

Usando fors para navegar objetos

```
function tablaPara(eventos, diario) {  
    let tabla = [0, 0, 0, 0];  
    for (let i = 0; i < diario.length; i++) {  
        let puntero = diario[i], index = 0;  
        if  
(puntero.eventos.includes(evento)) index += 1;  
        if (puntero.ardilla) index += 2;  
        table[index] += 1;  
    }  
    return tabla;  
}  
console.log(tablaPara("pizza", diario));
```

```
function diadrioEventos(diario) {  
    let eventos = [];  
    for (let puntero of diario) {  
        for (let evento of puntero.eventos) {  
            if (!eventos.includes(evento)) {  
                eventos.push(event);  
            }  
        }  
    }  
    return eventos;  
}  
console.log(journalEvents(JOURNAL));
```



+ Arrays

Añadiendo/sacando elementos: la pila 2

```
let listaPdte = [];  
function recordar(tarea) {  
    listaPdte.push(tarea);  
}  
function getTarea() {  
    return listaPdte.shift();  
}  
function recordarUrgente(tarea) {  
    listaPdte.unshift(tarea);  
}
```

07 La pila

08 Búsqueda en Arrays

```
console.log([1, 2, 3, "CLARA", 2, 1].indexOf(88));  
console.log([1, 2, 3, "CLARA", 2, 1].indexOf(2));  
console.log([1, 2, 3, "CLARA", 2, 1].lastIndexOf(2));  
console.log([1, 2, 3, "CLARA", 2, 1].indexOf("CLARA",  
3));  
console.log([1, 2, 3, "CLARA", 2, 1].lastIndexOf(2, 0));  
  
console.log([0, 1, 2, 3, 4].slice(2, 4));  
console.log([0, 1, 2, 3, 4].slice(2));
```

Strings y propiedades

```
let kim = "Kim";  
kim.age = 88;  
console.log(kim.age);  
// → undefined
```

```
console.log("cocotera".slice(4, 7));  
// → ter  
console.log("cocotera".indexOf("e"));  
// → 5
```

09 Strings y propiedades

- Puedo "definir" propiedades a un string, JS no se queja, pero NO las guarda



Investiguemos cuántos métodos tiene un string!
¿Qué hacen trim, padStart, split, join, repeat?

Desestructurando los objetos:

```
function phi(table) {  
    return (table[3] * table[0] - table[2] * table[1])  
        / Math.sqrt((table[2] + table[3]) *  
            (table[0] + table[1]) *  
            (table[1] + table[3]) *  
            (table[0] + table[2]));  
}
```

```
function phi([n00, n01, n10, n11]) {  
    return (n11 * n00 - n10 * n01) /  
        Math.sqrt((n10 + n11) * (n00 + n01) *  
            (n01 + n11) * (n00 + n10));  
}
```

```
let nombre = { nombre: "Esteban", edad: 23 };  
console.log(nombre);
```

//Y si recibo un objeto pero sólo quiero trabajar con un elemento puedo hacer:

```
let { edad } = { nombre: "Esteban", edad: 23 };  
console.log(edad);
```

```
let { nombre2 } = { nombre2: "Esteban", edad: 23 };  
console.log(nombre2);
```

Y sin let, ¡peta!

```
{ nombre3 } = { nombre3: "Esteban", edad: 23 };
```

Usando JSON

```
let string = JSON.stringify({ardilla: false,  
eventos: ["finde"]});  
console.log(string);  
// → {"ardilla":false,"eventos":["finde"]}  
  
console.log(JSON.parse(string).eventos);  
// → ["finde"]
```

Entregable 3 – 60'



1. Escribe la función rango con dos argumentos, inicio y fin, que devuelve un array que contenga todos los números desde inicio hasta (incluyéndolo) fin.
 1. Entonces, escribe una función de suma que cuando le pasas un array de números, los suma y devuelve la suma
 2. Si lo pruebas con el rango 1 a 10, debiera devolver 55.
2. Modifica tu función para que le puedas pasar un tercer argumento que diga el valor de "paso incremental". Si no se especifica es 1.
 1. Si llamas a tu función con (1, 10, 2) debiera devolver [1, 3, 5, 7, 9].
 2. Funciona con pasos negativos? rango(5, 2, -1) debiera devolver [5, 4, 3, 2].