UNIVERSIDAD
DE MÁLAGA

Iniciación a JavaScript

Objetos JS

Daniel Garrido

dgm@uma.es

UNIVERSIDAD
DE MÁLAGA

Objetos

Tipo de datos que permite almacenar propiedades y métodos (más adelante).

```
var objectName = {
  propertyName: propertyValue,  // propertyName con y sin " "
  ...
};
```

```
var aboutMe = {
  hometown: "Pasadena, CA",
  hair: "brown"
};
```

```
var lizzieTheCat = {
  age: 18,
  furColor: "grey",
  likes: ["catnip", "milk"],
  birthday: {"month": 7, "day": 17, year: 1994}
};
```

Acceso a Objetos

Accedemos a las propiedades con la notación ".":

```
var aboutMe = {
  hometown: "Pasadena, CA",
  hair: "brown"
};

var myHometown = aboutMe.hometown;
```

O con los [ ] (como arrays):

```
var myHair = aboutMe["hair"];
```

Las propiedades que no existan devolverán undefined:

```
var myGender = aboutMe["gender"];
```

Cambiando Objetos

Podemos usar "." o [ ] para cambiar los objetos.

Cambiar propiedades existentes:

```
var aboutMe = {
  hometown: "Pasadena, CA",
  hair: "brown"
};
aboutMe.hair = "blue";
```

O añadir propiedades:

```
aboutMe.gender = "female";
```

Es posible borrar propiedades:

```
delete aboutMe.gender;
```

Arrays de Objetos

Un array también puede almacenar objetos:

```
var myCats = [
  {name: "Lizzie",
   age: 18},
  {name: "Daemon",
   age: 1}
];

for (var i = 0; i < myCats.length; i++) {
  var myCat = myCats[i];
  console.log(myCat.name + ' is ' + myCat.age + ' years old.')
}
```

Objetos como argumentos

Los objetos pueden ser pasados a las funciones:

```
var lizzieTheCat = {
  age: 18,
  furColor: "grey",
  likes: ["catnip", "milk"],
  birthday: {"month": 7, "day": 17, year: 1994}
}

function describeCat(cat) {
  console.log("This cat is " + cat.age + " years old with " +
}

describeCat(lizzieTheCat);
```

¡TIEMPO DE EJERCICIOS!

Objetos

Métodos de Objetos

Las funciones de un objeto se denominan "métodos".

```
var lizzieTheCat = {
  age: 18,
  furColor: 'grey',
  meow: function() {
    console.log('meowww');
  },
  eat: function(food) {
    console.log('Yum, I love ' + food);
  },
  sleep: function(numMinutes) {
    for (var i = 0; i < numMinutes; i++) {
      console.log('z');
    }
  }
};
```

Métodos de Objetos

Desde los métodos podemos acceder a las propiedades

```
var lizzieTheCat = {  age: 18,
  furColor: 'grey',
  myColor:
     function() {
      console.log("My color "
          + this.furColor); }
}
```

Métodos de Objetos

Para llamar a los métodos usamos la notación ".":

```
lizzieTheCat.meow();
lizzieTheCat.eat('brown mushy stuff');
lizzieTheCat.sleep(10);
```

Desde ES6 no hace falta poner function para declarar métodos:

```
const dog = {
    name: 'Boo',
    bark() {
      console.log('yip!');
    }
  };
dog.bark(); // yip!
```

Claves de Objetos

`Object.keys()` retorna como array todos los nombres de propiedades y métodos de un objeto.

```
var lizzieTheCat = {
  age: 18,
  furColor: 'grey',
  meow: function() {
    console.log('meowww');
  },
  eat: function(food) {
    console.log('Yum, I love ' + food);
  },
  sleep: function(numMinutes) {
    for (var i = 0; i < numMinutes; i++) {
      console.log('z');
    }
  }
};
```

```
Object.keys(lizzieTheCat); // ["age", "furColor", "meow", "eat
```

Bucle for...in

Un bucle for...in permite iterar sobre todas las propiedades de un objeto:

```
var obj = {a: 1, b: 2, c: 3};

for (const prop in obj) {
  console.log(`obj.${prop} = ${obj[prop]}`);
}

// Output:
// "obj.a = 1"
// "obj.b = 2"
// "obj.c = 3"
```

Desmontando objetos (ES6)

Podemos desempaquetar las propiedades de un objeto en varias variables

```
const person = {
    name: 'Whitney',
    age: 38
};

const { name: localName, age } = person;
console.log(localName, age);     // Whitney 38
```

Esto mismo también puede ser usado con arrays

```
const colors = ['red', 'green', 'blue'];

const [ first, second, third ] = colors;

console.log(first, second, third); // red green blue
```

```
let a = 1;
let b = 2;

[ a, b ] = [ b, a ];
```

Objetos predefinidos

JS proporciona varios objetos predefinidos:

- Array
  Array.isArray()
- Number
  Number(), Number.parseInt(), Number.parseFloat()
- Date
  Date.UTC(), Date.now(), Date.parse()
- Math
  ¡Muuuchas cosas útiles!

Objetos predefinidos

Ejemplos

```
var x = Math.PI;            // Retorna PI
var y = Math.sqrt(16);      // Raíz cuadrada de 16

var d = new Date();
var n = d.getDay(); // Día de la semana

Math.floor((Math.random() * 10) + 1); // Número entre 1 y 10

var e = Math.sin(Math.PI / 2);

const fruits = ["Banana", "Orange", "Apple", "Mango"];
Array.isArray(fruits)   // Retorna true
```

Set y Map (ES6)

Set: Lista de valores únicos. Acceso rápido a sus datos.

```
const set = new Set([1, 2, 3, 4, 5, 5, 5, 5]);
console.log(set.size); // 5
```

```
set.add(6);
console.log(set.size); // 6
```

Map: Lista de parejas clave-valor, donde la clave y el valor pueden ser de cualquier tipo.

```
const map = new Map();
map.set('title', 'JS206: Intro to  ES6');
map.set('group', 'GDI');

console.log(map.get('title')); //  JS206: Intro to ES6
console.log(map.get('group')); //  GDI
```

Iteradores Predefinidos

- keys() retorna un iterador con las claves de la colección
- entries() retorna un iterador cuyos valores son parejas clave-valor
- values() retorna un iterador cuyos valores son los de la colección

```
for (let value of set.values()){
    console.log(value);
}
```

```
for (let value of map.values()){
    console.log(value);
}
```

Métodos para Arrays (nivel intermedio)

Desde ES6 muchos métodos nuevos

Útil para...

Método		Retorna
`arr.forEach((element, index, array) =>{ });`	itera a través de los elementos	`undefined`
`arr.every((element, index, array) =>{ });`	comprueba si todos los elementos pasan un test	`true||false`
`arr.some((element, index, array) =>{ });`	comprueba si algún (al menos 1) `arr` elemento pasa un test	`true||false`
`arr.filter((element, index, array) =>{ });`	crea un nuevo array filtrando los elementos originales del array	`Array`
`arr.map((element, index, array) =>{ });`	crea un nuevo array modificando los elementos del array original	`Array`
`arr.find((element, index, array) =>{ });`	encuentra el primer elemento que pasa un test	1 elemento
`arr.findIndex((element, index, array) =>{ });`	encuentra el índice del primer elemento que pasa un test	1 índice
`arr.reduce((calculatedValue, element, index, array) =>{ }, initialValue);`	pasa un `initialValue` y lo modifica de acuerdo con el valor del `elemento` actual	1 valor reducido

Métodos para Arrays

Alejándonos de los bucles for

- Más fácil para que otros vean y entiendan inmediatamente lo que queremos hacer
- Estilo de programación funcional
- Más compacto —¿por qué escribir más código?
- Más mantenible y escalable

foreach

ES5: BUCLE FOR

```
var names = ['Morgan', 'Taylor', 'Lesley'];

for (var i = 0; i < names.length; i++) {
  console.log(names[i]);
}
```

ES6: FOREACH

```
const names = ['Morgan', 'Taylor', 'Lesley'];

names.forEach((name, index, array) => console.log(name));
```

index y array son opcionales

every

ES5: FOR LOOP

```
var students = [
  { name: 'Morgan', present: true },
  { name: 'Sam', present: false },
  { name: 'Taylor', present: true }
  ];
var allPresent = true;
for (var i = 0; i < students.length; i++) {
  if (!students[i].present) {
    allPresent = false;
    break;
  }
}
console.log(allPresent); // false
```

ES6: EVERY

```
const students = [
  { name: 'Morgan', present: true },
  { name: 'Sam', present: false },
  { name: 'Taylor', present: true }
  ];
const allPresent = students.every(student => student.present);
console.log(allPresent); // false
```

El bucle se corta en cuanto algún elemento no cumple la condición

filter

Filtrar datos según alguna condición

```
const filteredArr = arr.filter(iteratorFunction);
```

```
const users = [
  { username: 'ryan10', active: true },
  { username: 'morgan', active: false }
  ];
const activeUsers = users.filter(user => user.active);
console.log(activeUsers);
```

map

Permite transformar un array asignando el resultado a otro array

```
const modifiedArr = arr.map(iteratorFunction);
```

```
const numbers = [2, 6, 10];

const halvedNumbers = numbers.map(number => number / 2);

console.log(halvedNumbers); // 1, 3, 5
```

find

Localizar elementos

```
const matchingObject = arr.find(iteratorFunction);
```

```
const jobs = [
  { title: 'Electrician' },
  { title: 'Developer' },
  { title: 'Barista' }
  ];
const devJob = jobs.find(job => job.title === 'Developer');
console.log(devJob); // { title: 'Developer' }
```

findindex

Posición dentro del array

```
const matchingIndex = arr.findIndex(iteratorFunction);
```

```
const numbers = [ 25, 30, 35, 40, 45 ];
  const firstIndex = numbers.findIndex(number => number > 33);
  console.log(firstIndex); // 2 (the location of 35)
```

reduce

Operación reduce

```
const reducedValue = arr.reduce(iteratorFunction, initialValue
```

```
const numbers = [2, 6, 10];

  const sum = numbers.reduce(function(currentSum, number) {
    return currentSum + number;
  }, 0);

  console.log(sum); // 18
```

reduce

reduce es un poco diferente porque toma 2 parámetros:

1. `iteratorFunction`
   - Utiliza un parámetro `calculatedValue`, que es modificado y retornado en cada iteración
2. `initialValue` —utilizado en la primera iteración para usarlo como`calculatedValue`

¡TIEMPO DE EJERCICIOS! Funciones para Arrays

EJERCICIOS ADICIONALES

Ejercicio adicional: CajaRegistradora
Validación de Tarjeta de Crédito

Acknowledgements / Reconocimientos:

Gran parte del material del curso procede originalmente de la web Teaching Materials Posteriormente, han sido enriquecidos con material adicional de diversas fuentes públicas así como adaptados a las necesidades específicas del curso a impartir.