

Tema 1

Introducción al curso



Introducción a Javascript



Introducción a Javascript

- Para comenzar a trabajar con Node.js vamos a dar un repaso a los conceptos más importantes del lenguaje Javascript.
- **Javascript** se trata de un lenguaje de programación principalmente usado para la creación de páginas web dinámicas.
- Es un lenguaje de programación interpretado, esto es, puede ser probado en cualquier navegador sin necesidad de ser compilado.
- Podemos incluir código Javascript de tres formas distintas:
 - Directamente en el documento web (etiqueta **<script>** en la cabecera)
 - En los diferentes elementos HTML (mediante eventos como **onclick**)
 - En un archivo externo (apuntando a cualquier archivo .js)

Introducción a Javascript

- Para declarar variables utilizamos la palabra **var** delante del nombre que usaremos y podremos usar un **=** para darle valor.
- El nombre sólo puede estar formado por letras, números (el primer carácter no puede ser un número), símbolo dólar y el guión bajo.
- Las variables pueden contener varios tipos de valores: numéricos, cadenas de texto, booleanos y arrays (js no es tipado).

```
var numero = 1;
var texto = "Hola mundo";
var booleano = true;
var arrayDias = ["lunes", "martes", "miércoles", "jueves",
"viernes", "sábado", "domingo"];
console.log (numero + " - " + texto);
console.log (booleano);
console.log (arrayDias);
```

Introducción a Javascript

- Las variables de tipo **numéricas** pueden almacenar valores tanto enteros como decimales, utilizándose el punto (.) para estas últimas.
- Las cadenas de texto se representan con comillas simples o dobles y nos permiten incluir caracteres especiales (\n, \t, \`, \", ...)
- Tenemos funciones que nos permiten transformar cadenas de texto a su valor numérico. Estas son **parseInt()** y **parseFloat()**.
- De la misma manera, para transformar una variable de tipo numérico a su representación como cadena de texto contamos con el método **toString()**.

Introducción a Javascript

- Los **booleanos** pueden tomar valor **true** o **false** sin comillas y son de gran utilidad a la hora de evaluar condiciones.
- Los **array** son colecciones de elementos de cualquiera de los tipos anteriores.
- Se declaran entre corchetes [] y para acceder a cualquiera de sus componentes tendremos que utilizar la fórmula **nombreArray[índice]**.
- Hay que tener en cuenta que el índice de un array empieza siempre por cero.

Introducción a Javascript

- Podemos crear determinadas estructuras condicionales con **if** que nos permiten evaluar el resultado de ciertas sentencias.
- Con la palabra reservada **else** interactuamos con el caso contrario de la condición que estamos evaluando:

```
var numero = 5;
if (numero > 3){
    console.log ("El número es mayor que 3");
}else{
    console.log ("El número NO es mayor que 3")
}
```

Introducción a Javascript

- Mediante la sentencia **switch** podemos elegir entre diferentes condiciones:

```
var numero = 5;
switch (numero){
  case 1:
    console.log ("Has introducido el numero 1");
    break;
  case 2:
    console.log ("Has introducido el numero 2");
    break;
  case 3:
    console.log ("Has introducido el numero 3");
    break;
  case 4:
    console.log ("Has introducido el numero 4");
    break;
  default:
    console.log ("Has introducido otro numero");
}
```


Introducción a Javascript

- Además de las estructuras condicionales, contamos con 3 tipos de bucles diferentes que nos permiten repetir una o varias sentencias tantas veces como necesitemos.
- Los diferentes bucles son:

```
//Se ejecuta el código
//mientras que la condicion
//sea true
while (condicion){
    //Código
}
```

```
//Ejecuta las sentencias mientras que se cumpla
//la condición, modificando el contador con el
//incremento especificado
for (contador; condicion; incremento){
    //Código
}
```

```
//Se ejecuta el código
//Mientras se cumpla la condición.
//Mínimo se ejecuta una vez
do{
    //Código
}while(condicion);
```

Introducción a Javascript

```
var arrayDias = [  
    "lunes", "martes", "miércoles",  
    "jueves", "viernes", "sábado", "domingo"];  
console.log(arrayDias.toString());  
for(var i = 0; i < arrayDias.length; i++)  
    console.log("arrayDias["+i+"]: "+arrayDias[i]);
```

Introducción a Javascript

```
var arrayDias = [
    "lunes", "martes", "miércoles",
    "jueves", "viernes", "sábado", "domingo"];
console.log(arrayDias.toString());
for(var i = 0; i < arrayDias.length; i++)
    console.log("arrayDias["+i+"]: "+arrayDias[i]);
```

"lunes,martes,miércoles,jueves,viernes,sábado,domingo"

"arrayDias[0]: lunes"

"arrayDias[1]: martes"

"arrayDias[2]: miércoles"

"arrayDias[3]: jueves"

"arrayDias[4]: viernes"

"arrayDias[5]: sábado"

"arrayDias[6]: domingo"

Introducción a Javascript

- Para optimizar el código podemos utilizar funciones. Favorece la reusabilidad del código ya que podemos encapsular diferentes sentencias y lanzarlas donde necesitemos a partir del nombre de la función.

```
function nombreFuncion(param1, param2){
    //Código
    console.log(param1+" "+param2);
}
//Cuando queramos ejecutar la función
nombreFunción("Hola", "Mundo");
```

- Las funciones pueden contar con 1 o varios parámetros y pueden devolver un valor gracias a la ejecución de **return** al final de la misma.

Introducción a ES6



Introducción a ES6

- ECMAScript 2015 añade nueva sintaxis para escribir aplicaciones complejas, incluyendo clases y módulos. También incluye iteradores, generadores al estilo de Python, funciones por flecha, datos binarios, colecciones (maps, sets...) metaprogramming para objetos visuales.

Simplifica la escritura, mejora la comprensión y ofrece más posibilidades.

- En las siguientes diapositivas vamos a comparar algunos ejemplos escritos en Js 'plano', y en la nueva versión.

Constantes en JS y ES6

```
Object.defineProperty(typeof global === "object" ? global :
window, "PI", {
  value:      3.141593,
  enumerable: true,
  writable:   false,
  configurable: false
})
PI > 3.0;
```

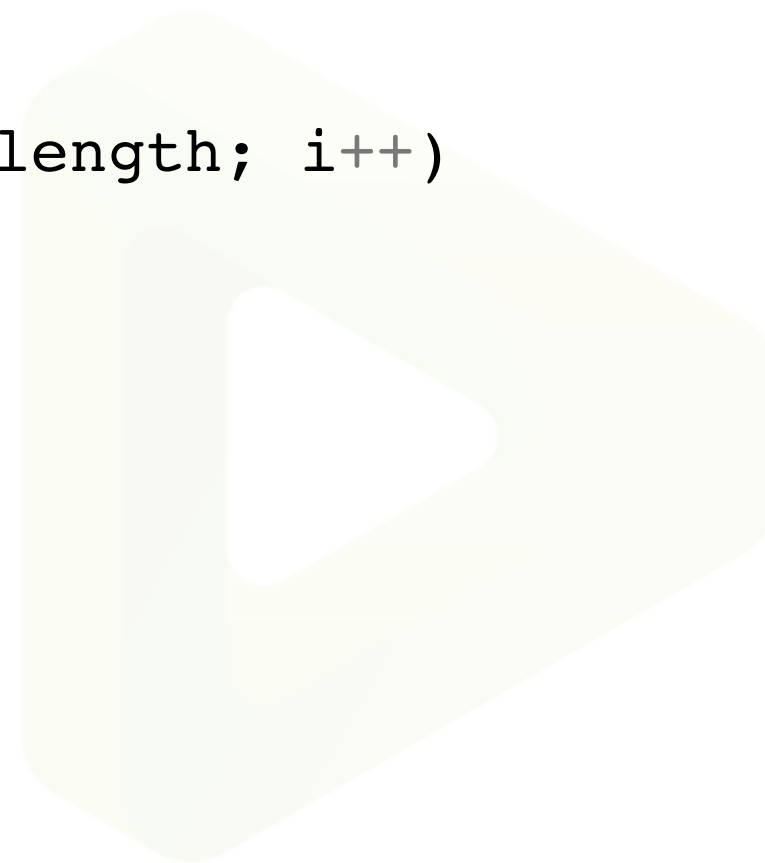
Constantes en JS y ES6

```
Object.defineProperty(typeof global === "object" ? global :
window, "PI", {
  value:      3.141593,
  enumerable: true,
  writable:   false,
  configurable: false
})
PI > 3.0;
```

```
const PI = 3.141593
PI > 3.0
```


For en JS y ES6

```
var i, x, y;
for (i = 0; i < a.length; i++) {
    x = a[i];
    ...
}
for (i = 0; i < b.length; i++)
    y = b[i];
    ...
}
```



For en JS y ES6

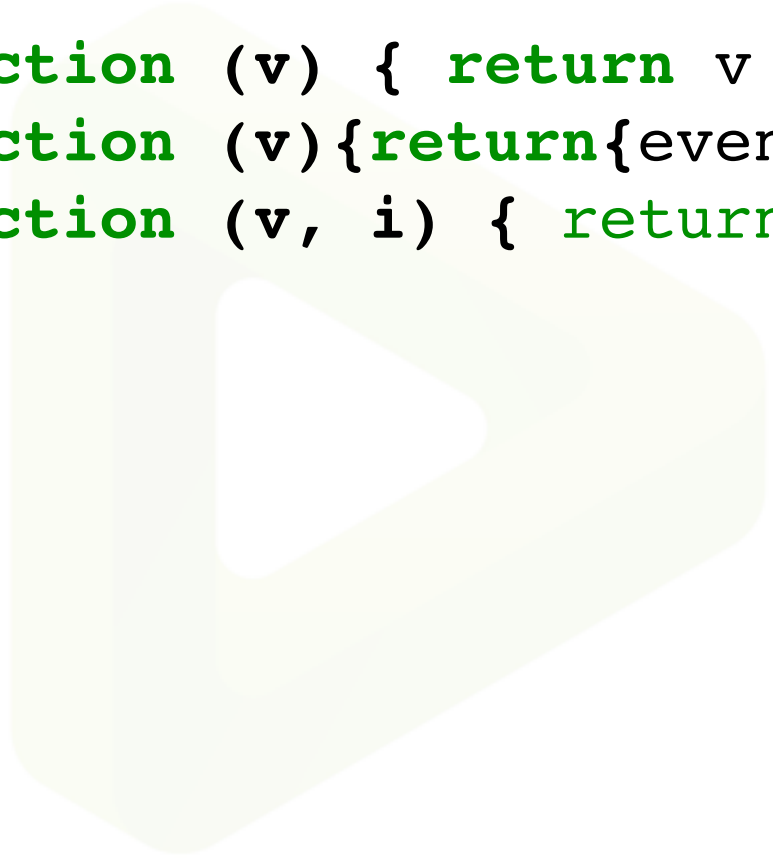
```
var i, x, y;
for (i = 0; i < a.length; i++) {
  x = a[i];
  ...
}
for (i = 0; i < b.length; i++)
  y = b[i];
  ...
}
```

```
for (let i = 0; i < a.length; i++) {
  let x = a[i]
  ...
}
for (let i = 0; i < b.length; i++) {
  let y = b[i]
  ...
}
```

//let permite asignar valores con validez únicamente dentro del contexto en la que se ha definido

Arrow Functions en JS y ES6

```
odds    = evens.map(function (v) { return v + 1; });  
pairs   = evens.map(function (v){return{even: v, odd: v+1};});  
nums    = evens.map(function (v, i) { return v + i; });
```



Arrow Functions en JS y ES6

```
odds    = evens.map(function (v) { return v + 1; });
pairs   = evens.map(function (v){return{even: v, odd: v+1};});
nums    = evens.map(function (v, i) { return v + i; });
```

```
odds    = evens.map(v => v + 1);
pairs   = evens.map(v => ({ even: v, odd: v + 1 }));
nums    = evens.map((v, i) => v + i);
//Menos código y más entendible con las arrow functions
```

Arrow Functions y map en ES6

```
var a = [
  "Hydrogen",
  "Helium",
  "Lithium",
  "Beryllium"
];
```

```
var a2 = a.map(function(s) { return s.length });
console.log(a2);
```

```
var a3 = a.map( s => s.length );
console.log(a3);
```

```
[8, 6, 7, 10]
[8, 6, 7, 10]
```

Objeto Set en ES6

El objeto Set nos permite almacenar valores únicos de cualquier tipo.

```
var mySet = new Set();

mySet.add(5);
mySet.add("algun texto");
var o = {a: 1, b: 2};
mySet.add(o);

mySet.has(1); // false, 1 no ha sido añadido
mySet.has(5); // true
mySet.has(Math.sqrt(25)); // true
mySet.has("Algun Texto".toLowerCase()); // true
mySet.has(o); // true

mySet.size; // 3

mySet.delete(5); // quitamos 5 del set
mySet.has(5); // false, 5 ha sido eliminado

mySet.size; // 2, ahora tenemos 2 valores
```

Valores por defecto en función con JS y ES6

```
function f (x, y, z) {
  if (y === undefined)
    y = 7;
  if (z === undefined)
    z = 42;
  return x + y + z;
};
f(1) === 50;
```

Valores por defecto en función con JS y ES6

```
function f (x, y, z) {
  if (y === undefined)
    y = 7;
  if (z === undefined)
    z = 42;
  return x + y + z;
};
f(1) === 50;
```

```
function f (x, y = 7, z = 42) {
  return x + y + z;
}
f(1) === 50;
```


Template en JS y ES6 y interpolación de strings

```
var customer = { name: "Foo" };
var card = { amount: 7, product: "Bar", unitprice: 42 };

message = "Hello " + customer.name + ",\n" +
"want to buy " + card.amount + " " + card.product + " for\n" +
"a total of " + (card.amount * card.unitprice) + " bucks?";
```

Template en JS y ES6 y interpolación de strings

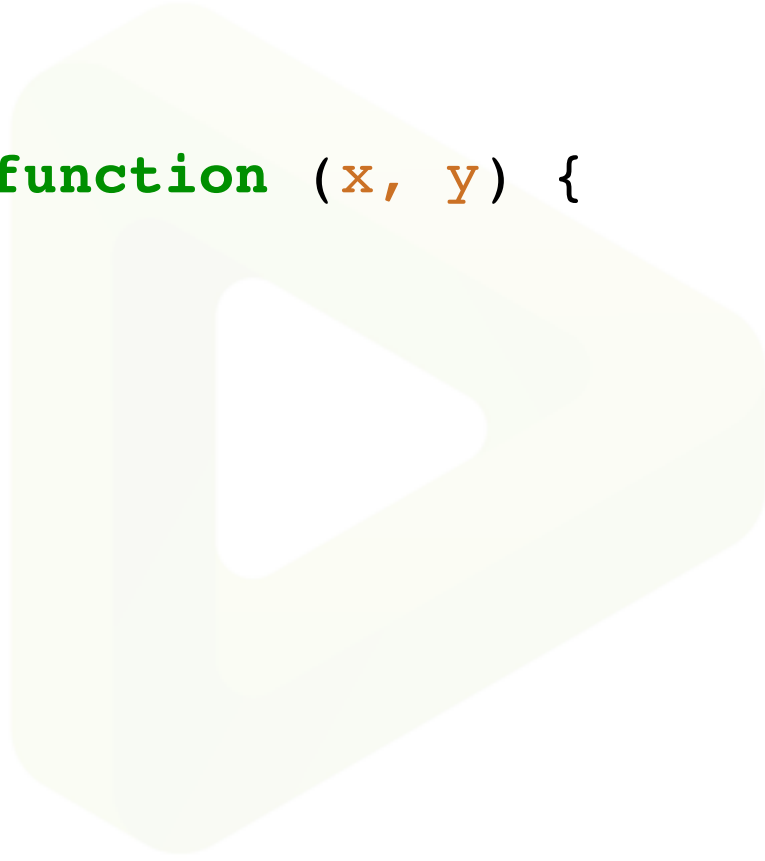
```
var customer = { name: "Foo" };
var card = { amount: 7, product: "Bar", unitprice: 42 };

message = "Hello " + customer.name + ",\n" +
"want to buy " + card.amount + " " + card.product + " for\n" +
"a total of " + (card.amount * card.unitprice) + " bucks?";
```

```
var customer = { name: "Foo" }
var card = { amount: 7, product: "Bar", unitprice: 42 }
message = `Hello ${customer.name},
want to buy ${card.amount} ${card.product} for
a total of ${card.amount * card.unitprice} bucks?`
```

Clases en ES6

```
var Shape = function (id, x, y) {
  this.id = id;
  this.move(x, y);
};
Shape.prototype.move = function (x, y) {
  this.x = x;
  this.y = y;
};
```



Clases en ES6

```
var Shape = function (id, x, y) {
  this.id = id;
  this.move(x, y);
};
Shape.prototype.move = function (x, y) {
  this.x = x;
  this.y = y;
};
```

```
'use strict';
class Shape {
  constructor (id, x, y) {
    this.id = id
    this.move(x, y)
  }
  move (x, y) {
    this.x = x
    this.y = y
  }
}
```

Clases estáticas en ES6

```
var Rectangle = function (id, x, y, width, height) {
    ...
};
Rectangle.defaultRectangle = function () {
    return new Rectangle("default", 0, 0, 100, 100);
};
var defRectangle = Rectangle.defaultRectangle();
```

Clases estáticas en ES6

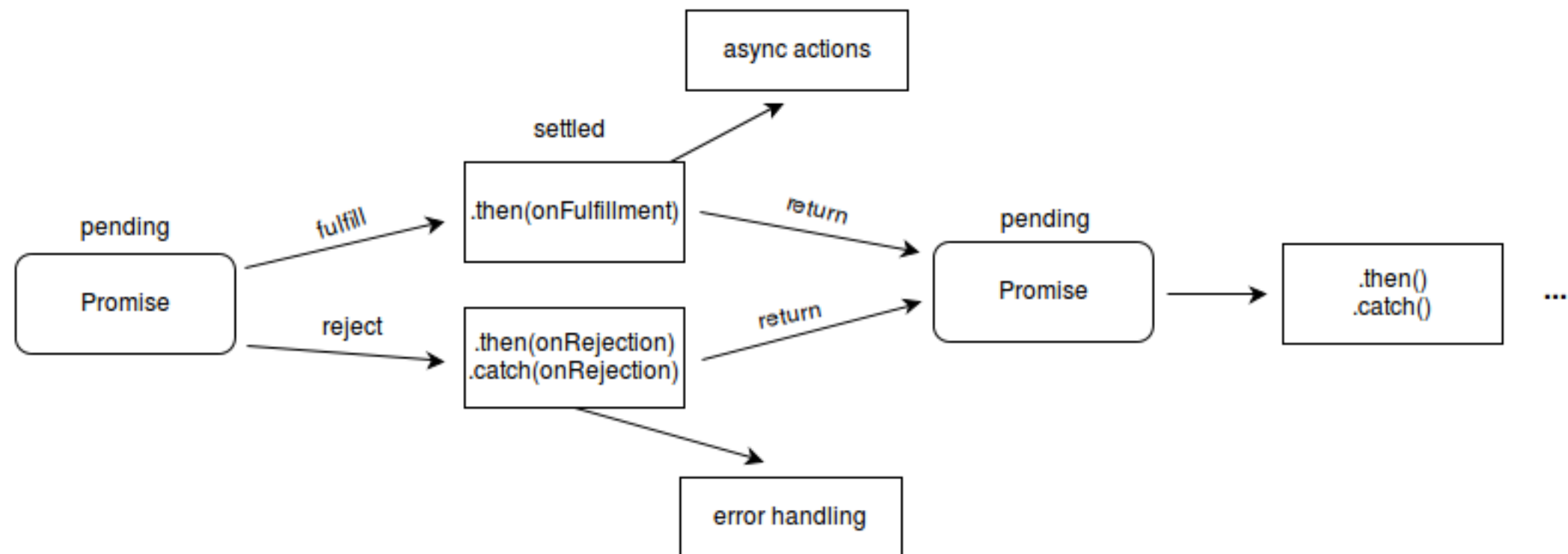
```
var Rectangle = function (id, x, y, width, height) {
    ...
};
Rectangle.defaultRectangle = function () {
    return new Rectangle("default", 0, 0, 100, 100);
};
var defRectangle = Rectangle.defaultRectangle();

class Rectangle extends Shape {
    ...
    static defaultRectangle () {
        return new Rectangle("default", 0, 0, 100, 100)
    }
}
var defRectangle = Rectangle.defaultRectangle()
```

Promesas en ES6

El objeto Promesa nos permite realizar funciones asíncornas. Representa una operación que todavía no ha terminado, pero en la que contamos con un resultado.

Por ejemplo, una petición RESTful o una consulta a una BD.



Introducción a Javascript

- ES6 es una gran ayuda para escribir nuestro código de forma más rápida y comprensible... pero no todas las funcionalidad están disponibles ahora mismo. En cada versión estable de node se añade un pequeño conjunto.
- Podemos ver con detalle cuales se pueden utilizar y un ejemplo práctico de cada una en la [documentación oficial](#)

