Argentina programa 4.0

UNIVERSIDAD TECNOLOGICA NACIONAL

Desarrollo en React JS

JavaScript ES6

Desarrollador Web con React Js

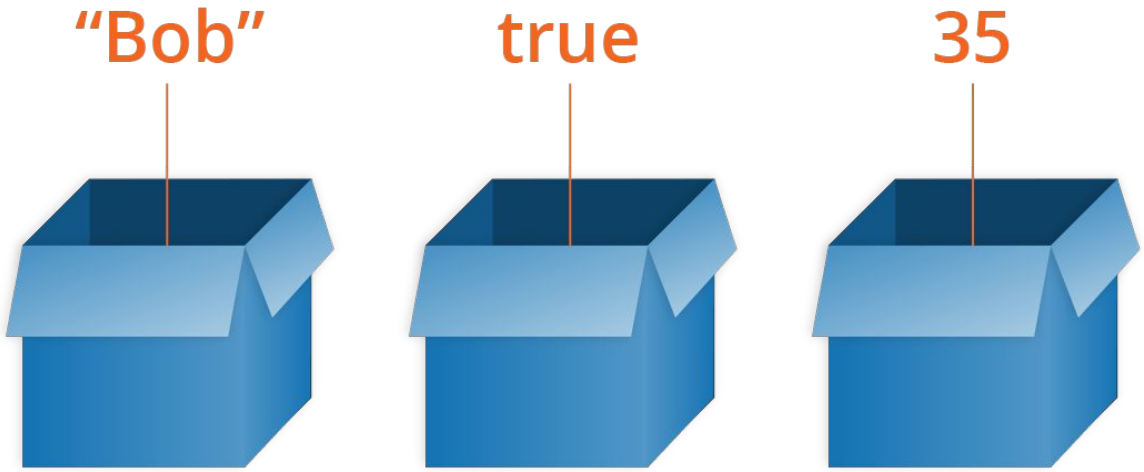
¿Qué es una variable?

Argentina programa 4.0

Una variable es un contenedor para un valor, como un número que podríamos usar en una suma, o una cadena que podríamos usar como parte de una oración.

- **Los valores que contienen pueden cambiar.**
- **Pueden contener casi cualquier cosa, no solo cadenas y números.** Las variables también pueden contener datos complejos e incluso funciones.

Las variables contienen valores, no son los valores en sí mismos; son contenedores de valores. **Puedes pensar en ellas como pequeñas cajas de cartón en las que puedes guardar cosas.**



Argentina programa 4.0

Declarar una variable:

Para usar una variable, primero debes crearla, escribimos la palabra clave **var** o **let** seguida del nombre con el que deseas llamar a tu variable.

```
let myName;
```

Iniciar una variable:

Una vez que hayas declarado una variable, la puedes iniciar con un valor. Para ello, escribe el nombre de la variable, seguido de un signo igual (**=**)**,** seguido del valor que deseas darle.

```
myName = 'Chris';
```

Puedes declarar e iniciar una variable al mismo tiempo, así:

```
let myDog = 'Rover';
```

Actualizar una variable:

Una vez que una variable se ha iniciado con un valor, puedes cambiar (o actualizar) ese valor simplemente dándole un valor diferente.

```
myName = 'Bob';
```

Tipo de las variables

- Números
- Cadenas de caracteres (Strings)
- Booleanos
- Arreglos (Arrays)
- Objetos

Tipado dinámico

JavaScript es un "lenguaje tipado dinámicamente", lo cual significa que, a diferencia de otros lenguajes, no es necesario especificar qué tipo de datos contendrá una variable (números, cadenas, arreglos, etc.).

¿Qué es una matriz?

Argentina programa 4.0

Las matrices se describen como "objetos tipo lista"; son objetos individuales que contienen múltiples valores almacenados en una lista. Pueden almacenarse en variables y tratarse de la misma manera que cualquier otro tipo de valor, la diferencia es que podemos acceder individualmente a cada valor dentro de la lista.

Creando una matriz

Las matrices se construyen con corchetes, que contiene una lista de elementos separdos por comas.

```
let shopping = ['bread', 'milk', 'cheese', 'hummus', 'noodles'];
let sequence = [1, 1, 2, 3, 5, 8, 13];
let random = ['tree', 795, [0, 1, 2]];
```

Accediendo y modificando elementos de la matriz

Argentina programa 4.0

Puedes acceder a elementos individuales en la matriz mediante la notación de corchetes: **[indice]**

```
shopping[0];
// devuelve "bread"
```

También puedes modificar un elemento en una matriz simplemente dando a un item de la matriz un nuevo valor.

```
shopping[0] = 'tahini';
shopping;
// shopping ahora devolverá [ "tahini", "milk", "cheese",
"hummus", "noodles" ]
```

Encontrar la longitud de una matriz

Argentina programa 4.0

Puedes averiguar la longitud de una matriz (cuántos elementos contiene) utilizando la propiedad **length**.

```
sequence.length;
// devuelve 7
```

Agregar y eliminar elementos de la matriz

Para añdir o eliminar un elemento al final de una matriz podemos usar **push()** y **pop()** respectivamente.

```
sequence.push('8');
sequence.pop;
```

Condicionales

Argentina programa 4.0

El código necesita realizar decisiones y llevar a cabo diferentes acciones acordes dependiendo de distintas entradas. Por ejemplo, en una aplicación del clima, si se observa en la mañana, se despliega una gráfica del amanecer; si es de noche, muestra estrellas y una luna.

Operadores de comparación

- **===** y **!==** prueba si un valor es exactamente igual a otro, o sino es indentico a otro valor.
- **<** y **>** prueba si un valor es menor o mayor que otro.
- **<=** y **>=** prueba si un valor es menor e igual o mayor e igual que otro.

Declaraciones if ... else

Argentina programa 4.0

```
if (condición) {
  código a ejecutar si la condición es verdadera
} else {
  ejecuta este otro código si la condición es falsa
}

let queso = 'Cheddar';
if (queso) {
  console.log('¡Siii! Hay queso para hacer tostadas con
queso.');
} else {
  console.log('No hay tostadas con queso para ti hoy :(.');
}
```

Operadores lógicos: AND, OR y NOT

Argentina programa 4.0

- **&& AND**: le permite encadenar dos o más expresiones para que **todas** ellas se tengan que evaluar individualmente **true** para que la expresión entera retorne true.

- **|| OR**: le permite encadenar dos o más expresiones para que **una o más** de ellas se tengan que evaluar individualmente **true** para que la expresión entera retorne true.

- El último tipo de operador lógico, **NOT,** es expresado con el operador **!**, puede ser usado para negar una expresión.

Declaraciones con switch

Argentina programa 4.0

Los switch statements toman una sola expresión/valor como una entrada, y luego pasan a través de una serie de opciones hasta que encuentran una que coincida con ese valor, ejecutando el código correspondiente que va junto con ella.

```
switch (expresion) {
    case choice1:
        ejecuta este código
    break;
    case choice2:
        ejecuta este código
    break;
    // Se pueden incluir todos los casos que quieras
    default:
        por si acaso, corre este código
}
```

¿Qué son los valores de retorno?

Argentina programa 4.0

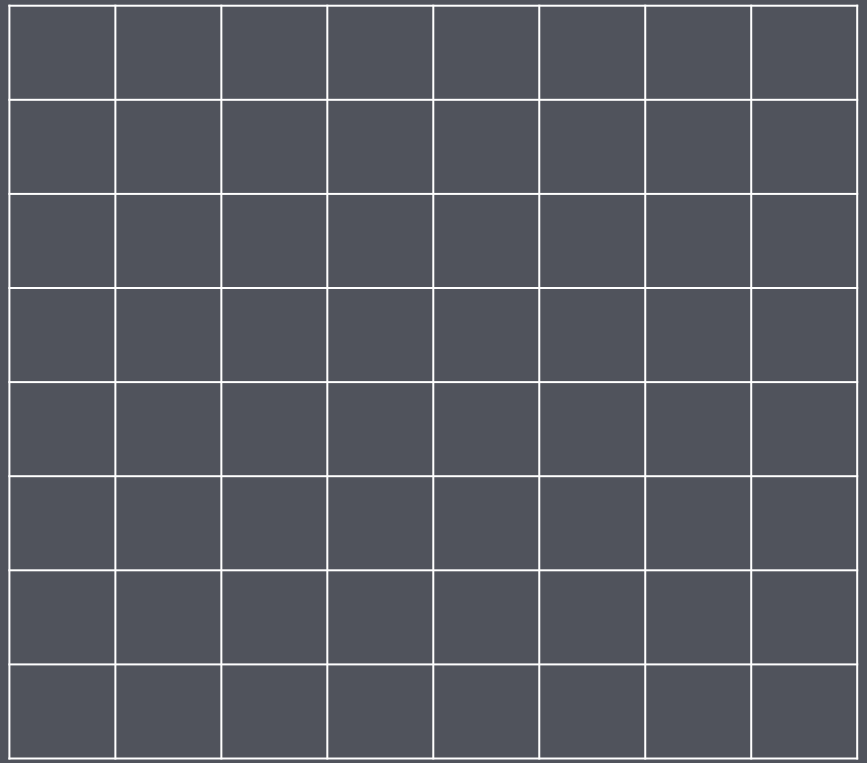
Los valores de retorno son exactamente como suenan: los valores devueltos por la función cuando se completa.

Algunas funciones no devuelven ningún valor específico como resultado de la función que se invoca (**void** o **undefined**).

```
function squared(num) {
    return num * num;
}

function cubed(num) {
    return num * num * num;
}
```

ECMAScript 6



ES6 – ECMAScript 6

Argentina programa 4.0

Es el estándar que define cómo debe de ser el lenguaje JavaScript.

JavaScript es interpretado y procesado por multitud de plataformas, entre las que se encuentran los navegadores web, así como NodeJS y otros ámbitos más específicos como el desarrollo de aplicaciones.

Todos los responsables de cada una de esas tecnologías se encargan de interpretar el lenguaje tal como dice el estándar ECMAScript.

Transpiladores

Argentina programa 4.0

Los transpiladores son programas capaces de traducir el código de un lenguaje para otro, o de una versión para otra. Dicho de otra manera, el código con posibles problemas de compatibilidad, hacerlo compatible con cualquier plataforma.

El programador escribe el código y el transpilador lo convierte en un proceso de **"tradución/compilación = transpilación"**.

Hoy tenemos transpiladores para traducir ES6 a ES5, pero también los hay para traducir de otros lenguajes a Javascript. TypeScript, CoffeeScript o Flow, son lenguajes que una vez transpilados se convierten en Javascript ES5, compatible con cualquier plataforma.

Template Strings

Argentina programa 4.0

Con ES6 podemos interpolar Strings de una forma más sencilla que como estábamos haciendo hasta ahora.

```
/*IN ES5*/
var userName = 'Hello World';
var message = 'Hey' + userName + ',';

/*IN ES6*/
let userName = 'Hello World';
let message = `Hey ${userName},`;
```

- Utilizamos el **backtick (`)** en lugar de las comillas.
- Luego colocamos el **placeholder ${...}** para identificar el contenido a ser interpretado.

De esta forma evitamos tener que concatenar las variables al string definido con comillas.

Let && Const

Argentina programa 4.0

Podemos definir una **constante** con la palabra reservada **const.**

```
const PI = 3.141593;
alert(PI > 3.0);
```

Las constantes no podrán ser redefinidas luego.

Var - Let

- El scope (alcance) de **var** abarca toda la función en la cual esta definida.
- Con **Let** el alcance de la variable pasa a ser las llaves en la cual está definida:

Tanto **let** como **const** no crean una propiedad global si se utilizan en el nivel superior

Función Arrow

Argentina programa 4.0

Es una simplificación sintáctica de la función en ES5

```
//ES5
const add = function(num) {
    return num + num;
}

//ES6
const add = (num) => {
    return num + num;
}
//en caso de ser una unica sentencia y sin {} aplica return
const add = (num) => num + num;
//Si recibe un unico parametro no hace falta los ()
const add = num => num + num;
//Si no recibe parametros se debe colocar ()
const add = () => num + num;
```

Deconstructor

Es una expresión que permite extraer propiedades de un objeto o ítems de un array

```
//Objeto
const address = {
    street: 'Pallimon',
    city: 'Kollam',
    state: 'Kerala'
};

//ES5
var street = address.street;
var city = address.city;
var state = address.state;
//ES6
const { street, city, state } = address;
```

En este caso vemos que podemos acceder y crear una constante “Street”, “city”, “state” relacionada con las propiedades que tienen el mismo nombre en el objeto address

```
//ES5
var values = ['Hello', 'World'];
var first = values[0];
var last = values[1];
//ES6
const values = ['Hello', 'World'];
const [first, last] = values;
```

En caso de ser un array, la constante “first” tendrá asignado el valor del array en el índice 0, mientras que “last” el valor del array en el índice 1

Valores por defecto

Argentina programa 4.0

Asignar valores por defecto a las variables que se pasan por parámetros en las funciones. Antes teníamos que comprobar si la variable ya tenía un valor, ahora con ES6 se la podemos asignar según creemos la función.

```
//ES5
function getUser (name, year) {
    year = (typeof year !== 'undefined') ? year : 2018;
    // remainder of the function…
}
//ES6
function getUser (name, year = 2018) {
    // function body here...
}
```

Como vemos el parámetro “year” cuando no reciba valor tomara por default el valor 2018

Import && Export

Argentina programa 4.0

Llamamos a las funciones desde los propios Scripts, sin tener que importarlos en el HTML, si usamos JavaScript en el navegador.

```
// MyClass.js
class MyClass{
  constructor() {}
}
export default MyClass;

// Main.js
import MyClass from 'MyClass';
```

- En el archivo **“MyClass.js”** tenemos la declaración de la clase y el export de la misma.
- En el archivo **“Main.js”** importamos MyClass (debe tener el export previamente)

En caso de no realizar **export default**
(es decir no declarar el artefacto como default)
debemos realizar el import con **{}:**

```
// MyClass.js
class MyClass{
  constructor() {}
}
export MyClass;

// Main.js
import {MyClass} from 'MyClass';
```

Solo se puede declarar un solo artefacto como default por modulo.

Rest y Spread

Argentina programa 4.0

Spread: Propaga los elementos de un array de forma individual

Podemos utilizarlo para concatenar 2 arrays:

```
function getSum(x, y, z){
    console.log(x+y+z);
}
let sumArray = [10,20,30];
getSum(...sumArray);
```

```
var a = [1, 2];
var b = [3, 4];
var c = [...a,...b]
console.log(c);
```

En este caso “c” tendrá los valores de a y b.

Este operador también puede ser utilizados para objetos:

```
let alumno = {
    nombre:"Leandro",
    apellido:"Gil"
}
let cursoAlumno = {...alumno,{curso:"php"}}
```

Promises

Una Promise (promesa en castellano) es un objeto que representa la terminación o el fracaso eventual de una operación asíncrona. Una promesa puede ser creada usando su constructor. Sin embargo, la mayoría de la gente son consumidores de promesas ya creadas devueltas desde funciones.

```
let p = new Promise(function(resolve, reject) {
        if (/* condition */) {
            resolve(/* value */);
            // fulfilled successfully
        }else {
            reject(/* reason */);
            // error, rejected
} });

p.then((val) => console.log("fulfilled:", val))  //10
  .catch((err) => console.log("rejected:", err));
```

La promesa es un objeto en este caso lo asignamos a la variable “p”.
Este objeto recibe como parámetro una función de **callback** la cual recibe el parámetro **resolve** y **reject**.

- En caso que se resuelva la promesa de forma **correcta** retornara:
  resolve(valor_retorno)
  Esto indicara que la ejecución fue correcta y retorna el valor deseado.
- En caso que ocurra un **error** o **excepción** se retornara:
  reject(valor_error)
  Se indica un error o excepción y la causa como parámetro.

Luego tratamos la promesa aplicando el método “then” y “catch”

Cuando se resuelva la misma de forma correcta (resolve) ejecuta el then y recibe el parámetro devuelto. En caso de error o excepción (reject) ejecuta el catch.

Async / Await

Argentina programa 4.0

- Una función **"async",** es una función que sabe cómo esperar la posibilidad de que la palabra clave "await" sea utilizada para invocar código asíncrono.
- Por otro lado**, await** solo trabaja dentro de las funciones async.

Esta puede ser puesta frente a cualquier función async basada en una promesa para pausar tu código en esa línea hasta que se cumpla la promesa, entonces retorna el valor resultante.

Mientras tanto, otro código que puede estar esperando una oportunidad para ejecutarse, puede hacerlo.

Clases

Argentina programa 4.0

Ahora JavaScript tendrá clases, muy parecidas las funciones constructoras de objetos que realizamos en el estándar anterior, pero ahora bajo el paradigma de clases, con todo lo que eso conlleva, como, por ejemplo, herencia.

```
class LibroTecnico extends Libro {
    constructor(tematica, paginas) {
        super(tematica, paginas);
        this.capitulos = [];
        this.precio = "";
        // ...
    }
    metodo() {
        // ...
    }
}
```

This

Argentina programa 4.0

La variable **this** muchas veces se vuelve un dolor de cabeza. antiguamente teníamos que cachearlo en otra variable ya que solo hace referencia al contexto en el que nos encontremos.

En el siguiente código si no hacemos **var that = this** dentro de la función **document.addEventListener**, **this** haría referencia a la función que pasamos por Callback y no podríamos llamar a foo()

```
//ES3
var obj = {
    foo : function() {...},
    bar : function() {
        var that = this;
        document.addEventListener("click", function(e) {
            that.foo();
        });
    }
}
```

Con ECMAScript5 la cosa cambió un poco, y gracias al método bind podíamos indicarle que this hace referencia a un contexto y no a otro.

Argentina programa 4.0

```
//ES5
var obj = {
    foo : function() {...},
    bar : function() {
        document.addEventListener("click", function(e) {
            this.foo();
        }.bind(this));
    }
}
```

Ahora con ES6 y la función **Arrow =>** la cosa es todavía más visual y sencilla.

```
//ES6
var obj = {
    foo : function() {...},
    bar : function() {
        document.addEventListener("click", (e) => this.foo());
    }
}
```

Gracias!