Javascript

Programación Web I

Comisión Miércoles Noche

DIS. Alicia Rosenthal

Tec. Willian Dos Reis

Comisión Viernes Mañana

Ing. Gabriel Panik

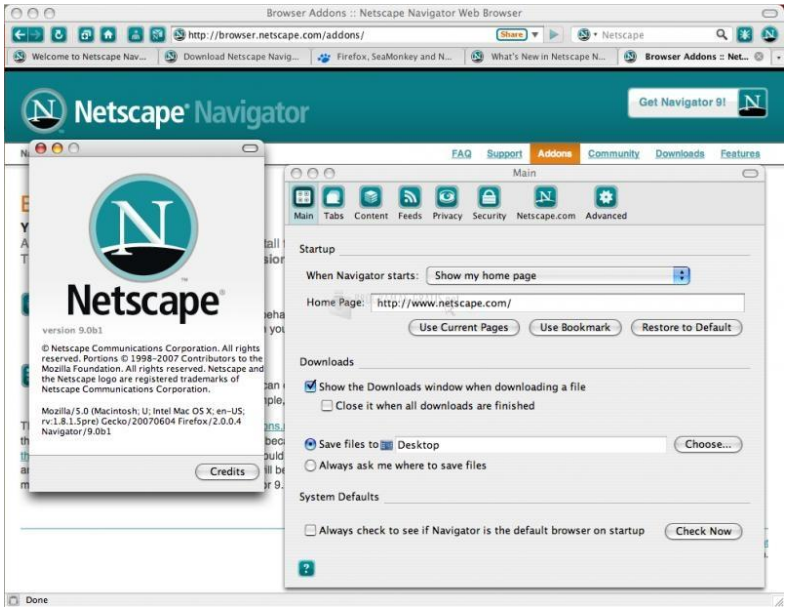
JAVASCRIPT

- Lenguaje Interpretado
- Multiplataforma
- Debilmente Tipado y Dinámico
- Orientado a Objetos y Eventos
- Imperativo
- Basado en prototipos

JAVASCRIPT

- Creado por Brendan Eich, programador de Netscape v2.0 en 1995. ->Livescript ->javascript

- Microsoft lanzó Jscript para Internet Explorer 3.0

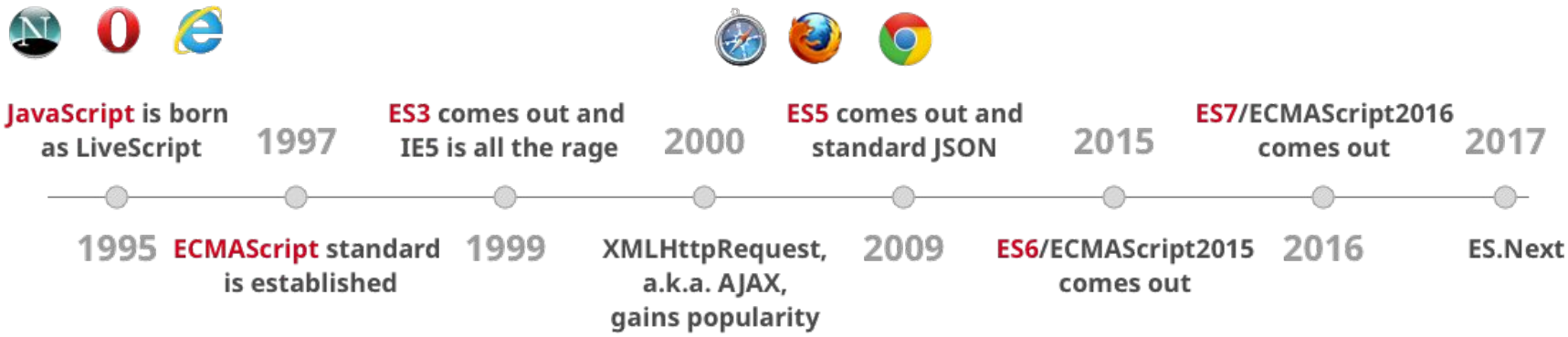


ECMAScript

ECMA

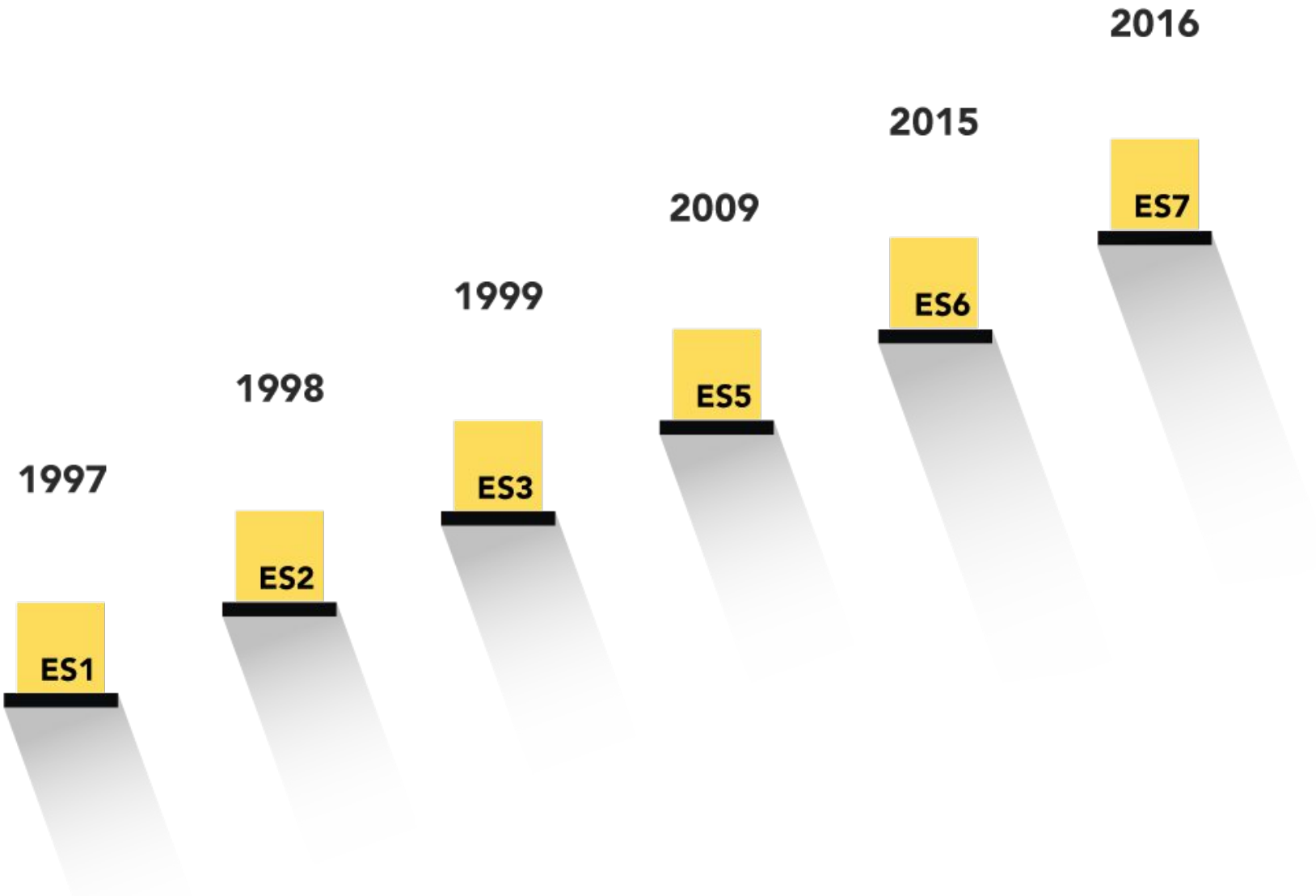
European Computer Manufacturers Association

https://www.w3schools.com/js/js_versions.asp



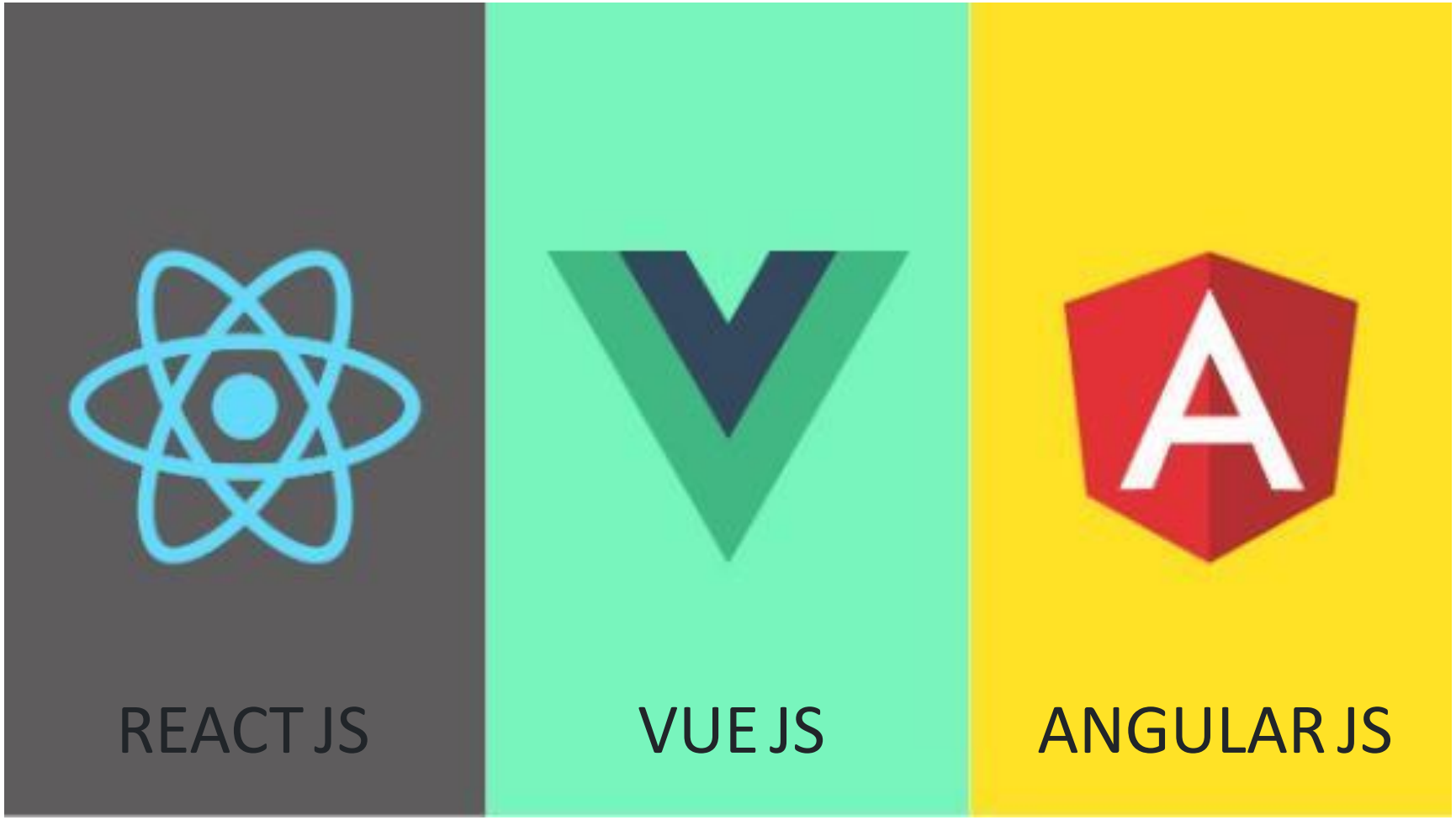
Estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa

ECMAScript

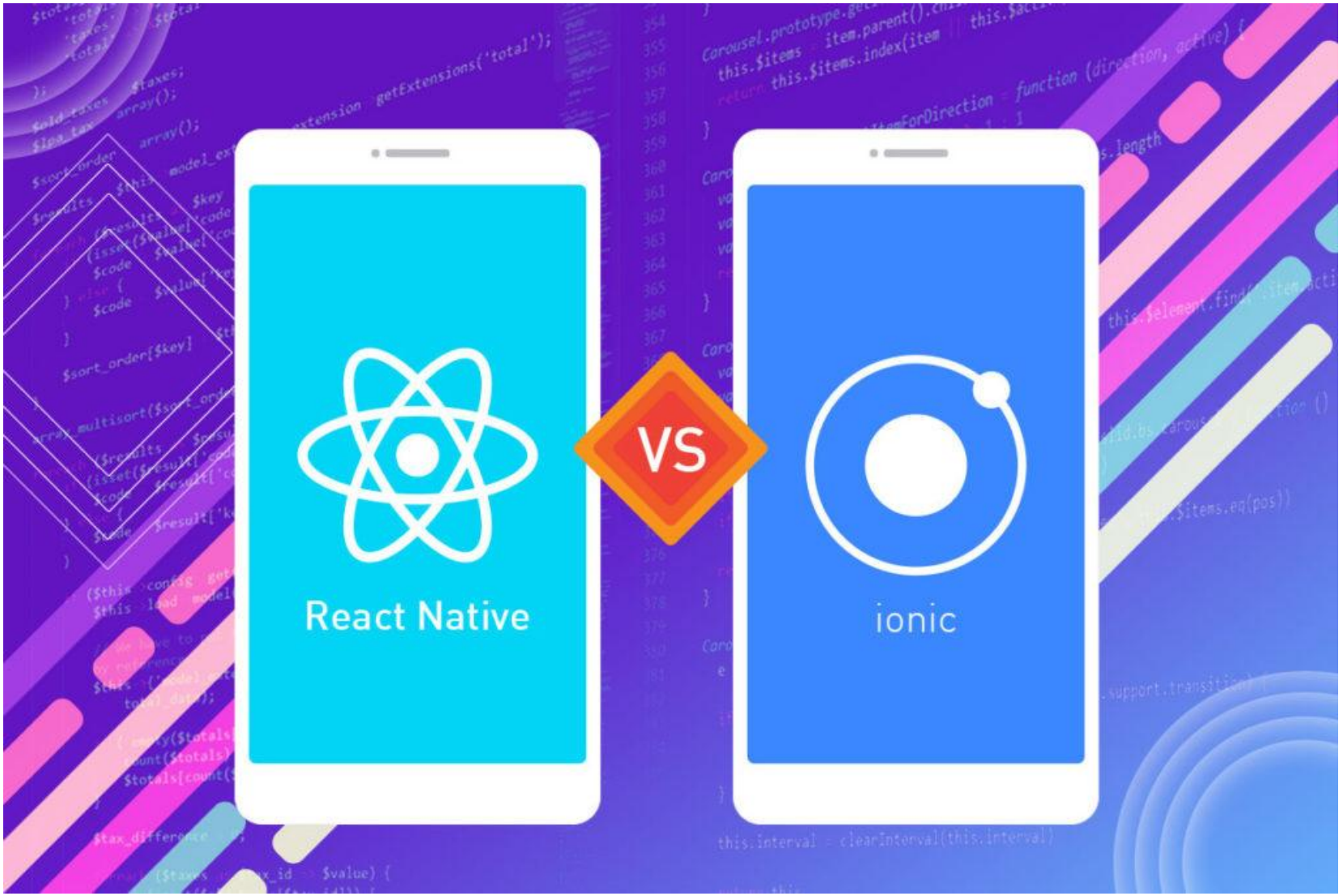


Librerías JAVASCRIPT

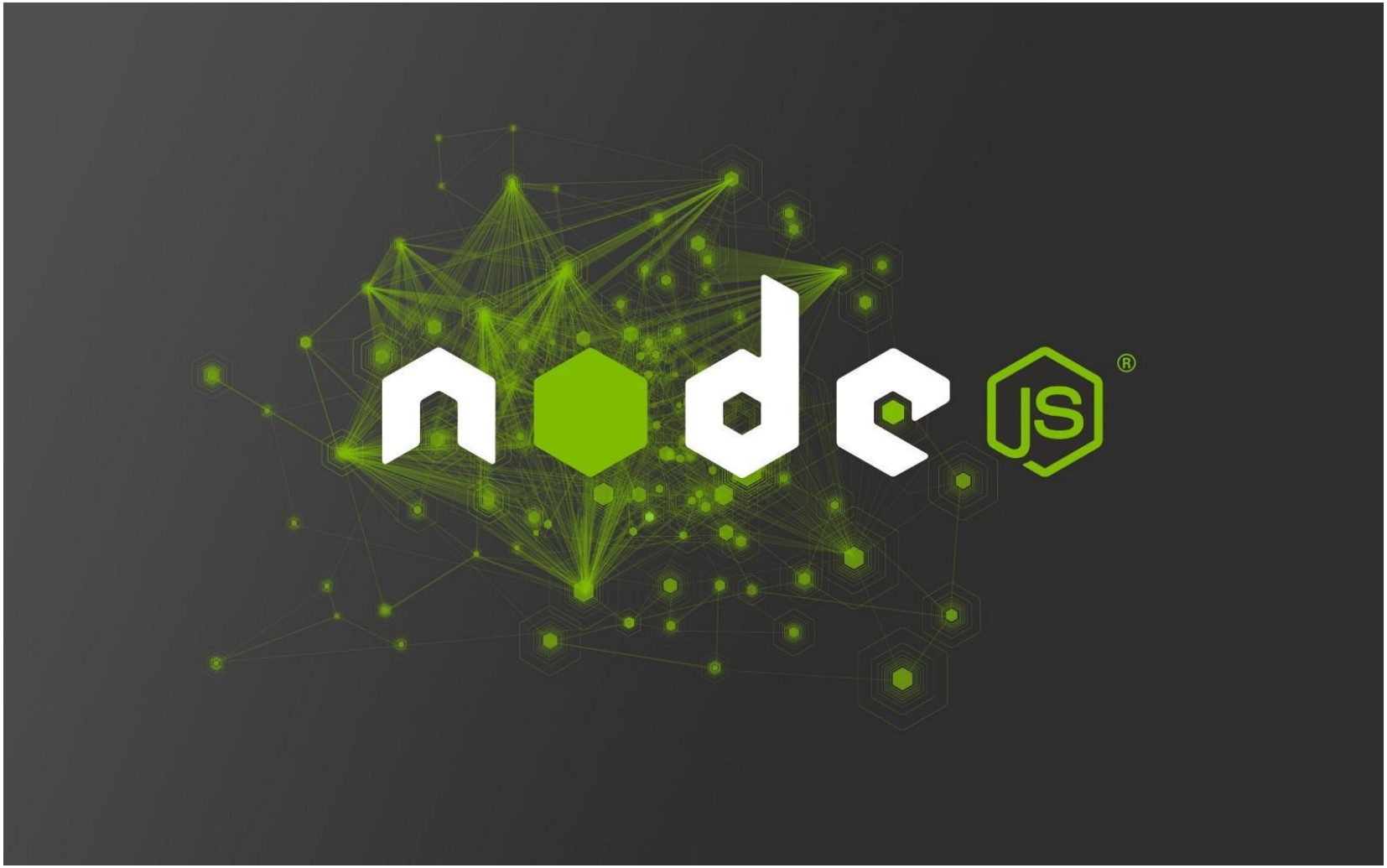
Frameworks JAVASCRIPT



Frameworks JAVASCRIPT



Frameworks JAVASCRIPT



INCLUIR JS en HTML

```
<head>
   <meta charset="UTF-8">
   <script>
      alert ("Primer Script");
   </script>
</head>
```

INCLUIR JS en Archivo Externo

1. Crear un archivo con la extensión .js dentro de una carpeta llamada js
2. Vincular el código javascript a nuestro html

```
<head>
    <meta charset="UTF-8">
    <script src="js/codigo.js">
    </script>
</head>
```

MOSTRAR DATOS

- //popup
```
alert ("");
```

- //en el documento HTML
```
document.write ("");
```

- //en la consola
```
console.log("");
```

SINTAXIS JAVASCRIPT

- **No** se tienen en cuenta los espacios en blanco y las nuevas líneas
- Se distinguen las mayúsculas y minúsculas
- **No** se define el tipo de las variables
- **No** es necesario terminar cada sentencia con el carácter de punto y coma
- Se pueden incluir comentarios
  - Una sola línea //
  - Varias líneas (entre /* y */)

Variables

- Almacenar un dato
- Se crean con la palabra reservada var

```
//declaración
var nombre;

//inicialización
nombre ="Pablo";
```

Variables

El nombre de una variable:

- **NO** puede empezar con un número
- **NO** puede usar carácteres especiales, ni espacios en blanco
- **Se puede** utilizar el signo $ y el guión bajo (_)
- Distingue Mayúsculas de Minúsculas

Variables

Convención para los nombres de Variables:

camelCase

Primera palabra toda en minúscula, primera letra de cada palabra siguiente Mayúsculas

Por ejemplo:

```
var nombreApellido;
var numeroDocumento;
```

Tipos de Datos Primitivos

TIPO	Valores
Number	Numéricos incluidos números negativos y decimales
String	Cadenas de Texto. Comillas simples o dobles.
Boolean	Valores posibles true o false
Null	Nulos o vacíos
Undefined	Sin definir, cuando una variable no fue inicializada.
Symbol	Nuevo tipo en JavaScript introducido en la versión ECMAScript Edition 6

PEDIR DATOS

Palabra reservada prompt

```
prompt("Mensaje al usuario");

var nombre;

nombre = prompt("Ingrese su nombre");
```

Constantes

- Almacenar un dato NO Variable
- Se crean con la palabra reservada const

```
//declaración
const PI = 3.14;
```

Constantes

Convención para los nombres de Constantes:

UPPERCASE

Toda la palabra en MAYÚSCULA

Por ejemplo:

```
const PI;
const IVA;
```

Operadores de Asignación

- =
- +=, -=, *=, /=

Ejemplos:

```
numeroUno = 8;
numeroDos = 3;
numeroUno += 3;
```

Operadores Aritméticos

- +
- -
- *
- /
- %
- ++ (Incremento)
- -- (decremento)

Operadores Lógicos

- Negación (!)
- AND (&&)
- OR (||)

Operadores Lógicos

Ejemplos:

```
a = 8;
b = 3;
c = 3;
document.write( (a == b) && (c > b) );
document.write( (a == b) && (b == c) );
document.write( (a == b) || (b == c) );
document.write( (b <= c) );
document.write( !(b <= c) );
```

Operadores Relacionales

- >, >= Mayor, mayor o igual
- <, <= Menor, menor o igual
- == Igual
- != Distinto

Operadores Relacionales

```
var numero1 = 3;
var numero2 = 5;
resultado = numero1 > numero2;
resultado = numero1 < numero2;
numero1 = 5; numero2 = 5;
resultado = numero1 >= numero2;
resultado = numero1 <= numero2;
resultado = numero1 == numero2;
resultado = numero1 != numero2;
numero1 = 5; numero2 = 4;
resultado = numero1 == numero2 ;
resultado = numero1 === numero2;
```

PARSEAR DATOS

- `isNaN(variable);`
  `//devuelve true si no lo es`
- `parseInt(variable);`
  `//convierte en número entero`
- `parseFloat(variable);`
  `//convierte en número flotante`
- `String(variable);`
  `//convierte en cadena de texto`

CONDICIONALES IF ELSE

```
if(condicion){

} else{

}
```

CONDICIONALES ELSE if

```
if(condicion){

} else if(condicion){

}
```

CONDICIONALES SWITCH

```
switch(nomVariable) {
case "opcion1" :
    instrucciones;
   break;
case "opcion2" :
    instrucciones;
    break;
.....................
case "opcionN" :
    instrucciones;
    break;
default :
    instrucciones;
}
```

CONDICIONALES SWITCH

```
switch(numero) {
    case 5:
    ...
    break;
    case 8:
     ...
     break;
    case 20:
    ...
    break;
    default:
    ...
    break;
}
```

BUCLE FOR

```
for(var i=0;i<=valor;i++){
    //instrucciones a repetir
}
```

BUCLE WHILE

```
while(condicion a evaluar){

}
```

CONDICIONALES ELSE if

```
do{

} while(condicion)
```

Funciones

¿Qué es una función?

Una función es un conjunto de instrucciones que se agrupan para realizar una tarea concreta y que se pueden reutilizar fácilmente.

```
function nombre_funcion(argumento1, argumento2, …., argumentoN) {

        ...

        return valor_a_retornar;
}
```

A continuación se presentarán las funciones y propiedades básicas de Javascript

FUNCIONES útiles Números

Error común para números

- **NaN** (del inglés, *"Not a Number"*. Indica un valor numérico no definido. Por ejemplo, la división 0/0)

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

Funciones útiles para números

- **isNaN()** (Permite proteger a la aplicación de posibles valores numéricos no definidos)

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
        alert("La división no está definida para los números indicados");
}
else {
        alert("La división es igual a => " + numero1/numero2);
}
```

FUNCIONES útiles Números

Funciones útiles para números

- **Infinity** (Hace referencia a un valor numérico infinito y positivo. También existe el valor –Infinity para los infinitos negativos)

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

- **toFixed(digitos)** (Devuelve el número original con tantos decimales como los indicados por el parámetro dígitos y realiza los redondeos necesarios. Se trata de una función muy útil por ejemplo para mostrar precios)

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

FUNCIONES útiles Cadenas

`length`

Calcula la longitud de una cadena de texto (el número de caracteres que la forman)

```
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length;
```

FUNCIONES útiles Cadenas

`+ o concat`

(Se emplean para concatenar varias cadenas de texto)

```
var mensaje1 = "Hola";
var mensaje2 = " Mundo";

var mensaje = mensaje1 + mensaje2;

var mensaje3 = mensaje1.concat(mensaje2);
```

FUNCIONES útiles Cadenas

toUpperCase()

Transforma el texto en mayúscula

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

toLowerCase()

Transforma el texto en minúscula

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toLowerCase();
```

FUNCIONES útiles Cadenas

charAt()

Obtiene el caracter que en la posición indicada

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

substring(inicio, final)

Toma una porción del texto

```
var mensaje = "Hola Mundo";
var porcion = mensaje.substring(2) //"la Mundo"
porcion = mensaje.substring(1, 8); //"ola Mun"
```

FUNCIONES útiles Cadenas

`indexOf(caracter)`

(Calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda. Si la cadena no contiene el carácter, la función devuelve el valor -1)

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

FUNCIONES útiles Cadenas

`lastIndexOf(caracter)`

(calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si la cadena no contiene el carácter, la función devuelve el valor -1. La posición devuelta se calcula empezando a contar desde el principio de la palabra)

```
var mensaje = "Hola Carola";
var posicion = mensaje.lastIndexOf('a');
//posicion =10
posicion = mensaje.lastIndexOf('b'); //
posicion=-1
```

Arrays

```
var variable1 = new Array();

var variable1 = new Array(10);

var variable1 = new Array(2, "hola",

true, 45.34);
```

Arrays

```
var variable1 = new Array();

variable1[0] = 2;

variable1[1] = "hola";

variable1[2] = true;

variable1[3] = 45.34;
```

Arrays

```
var dias = ["Lunes", "Martes", "Miércoles",

"Jueves", "Viernes", "Sábado", "Domingo"];



for(var i=0; i<7; i++) {

  alert(dias[i]);

}
```

Arrays

```
var dias = ["Lunes", "Martes", "Miércoles",

"Jueves", "Viernes", "Sábado", "Domingo"];



for(i in dias) {

  alert(dias[i]);

}
```

Funciones útiles para arrays

Funciones útiles para arrays

- **length** (Calcula el número de elementos de un array)

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

- **concat()** (Se emplea para concatenar los elementos de varios array)

```
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)** (Es la función contraria a split(). Une todos los elementos de un array para formar una cadena de texto. Para unir los elementos se utiliza el carácter separador indicado)

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

Funciones útiles para arrays

Funciones útiles para arrays

- **pop() (**Elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento)

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

- **push()** (Añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

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

- **shift()** (Elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento)

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

Funciones útiles para arrays

Funciones útiles para arrays

- **unshift()** (Añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento. También es posible añadir más de un elemento a la vez)

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

- **reverse()** (Modifica un array colocando sus elementos en el orden inverso a su posición original)

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

Ámbito de las variables

Variables locales

El ámbito de una variable (llamado "scope" en inglés) es la zona del programa en la que se define la variable. JavaScript define dos ámbitos para las variables: global y local.

```
function creaMensaje() {
        var mensaje = “Mensaje de prueba”;
}
creaMensaje();
alert(mensaje);
```

Al ejecutar el código anterior no se muestra ningún mensaje por pantalla. La razón es que la variable ”mensaje” se ha definido dentro de la función creaMensaje() y por tanto, es una variable local que solamente está definida dentro de la función.

Cualquier instrucción que se encuentre dentro de la función puede hacer uso de esa variable, pero todas las instrucciones que se encuentren en otras funciones o fuera de cualquier función no tendrán definida la variable “mensaje”.

Ámbito de las variables

Variables globales

Está definida en cualquier punto del programa (incluso dentro de cualquier función).

```
var mensaje = “Mensaje de prueba”;

function muestraMensaje() {
        alert(mensaje);
}
```

La variable ”mensaje” se ha definido fuera de cualquier función.

Este tipo de variables automáticamente se transforman en variables globales y están disponibles en cualquier punto del programa (incluso dentro de cualquier función)

Si en el interior de una función, las variables se declaran mediante var se consideran locales y las variables que no se han declarado mediante var, se transforman automáticamente en variables globales.

¿Qué sucede si una función define una variable local con el mismo nombre que una variable global que ya existe? En este caso, las variables locales prevalecen sobre las globales, pero sólo dentro de la función:

Ámbito de las variables

Variables globales

¿Qué sucede si dentro de una función se define una variable global con el mismo nombre que otra variable global que ya existe?
La variable global definida dentro de la función simplemente modifica el valor de la variable global definida anteriormente:

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
        mensaje = "gana la de dentro";
        alert(mensaje);
}

alert(mensaje);         // gana de la fuera
muestraMensaje();  // gana la de adentro
alert(mensaje);// gana la de adentro
```