Profundizando en JavaScript

Víctor Custodio

Profundizando en JavaScript

- Una vez vista la introducción a JavaScript, ya tenemos una pinzalada de todo lo que es JS.
- Esto es necesario ya que los distintos conceptos de JS están muy interrelacionados, y es difícil profundizar en cada concepto sin al menos tener conocimiento del resto de conceptos con los que esta relacionado.
- Durante este powerpoint profundizaremos más en los conceptos estudiados en la introducción así como veremos nueves elementos de JavaScript

Strings

Strings

- Podemos usar Comillas dobles o simples, para definirlos.
- También podemos usar comillas en el interior siempre y cuando sean distintas a las comillas exteriores:
- ```
  var resp = “A él le llaman ‘ el Dandy'";
  var resp = ‘A él le llaman “el dandy"';
  ```
  ```

Strings - Longitud

- Podemos acceder a la longitud de una cadena accediendo a su propiedad length:

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var long= txt.length;
```

Strings – Pueden ser objetos

- Hemos visto los strings como valores primitivos, pero tambien pueden definirse como objetos:

```
var x = "John";
var y = new String("John");
```

Crear Strigns como objetos es eficientemente peor que como valores primitivos y su uso es distinto:

Strings – Pueden ser objetos

- ```
  var x = "John";
  var y = new String("John");
  ```

- ```
  // (x == y) es Verdad porque x e y tienen el mismo valor
  ```
- ```
  // (x === y) es falso porque x e y son de distintos tipos
  ```
- ```
  var x = new String("John");
  var y = new String("John");

  // (x == y) es falso porque los objectos no pueden ser comparados con ==(recuerda un objeto es una referencia, y objetos distintos referencian a una posición distinta, aunque el contenido sea el mismo)
  ```

Strings – Métodos y propiedades

- Valores primitivos , como " John Doe " , no pueden tener propiedades o métodos (porque no son objetos) .
- Pero con JavaScript métodos y propiedades también están disponibles para valores primitivos , debido a que JavaScript trata valores primitivos como objetos al ejecutar los métodos y propiedades.
- A continuación veremos los métodos más importantes en los elementos Strings:

Strings - Métodos

- **charAt(indice)**
  Devuelve el carácter que hay en la posición indicada como índice. Las posiciones de un string empiezan en 0.
- **indexOf(carácter,desde)**
  Devuelve la posición de la primera vez que aparece el carácter indicado por parámetro en un string. Si no encuentra el carácter en el string devuelve -1. El segundo parámetro es opcional y sirve para indicar a partir de que posición se desea que empiece la búsqueda.
- **lastIndexOf(carácter,desde)**
  Busca la posición de un carácter exáctamente igual a como lo hace la función indexOf pero desde el final en lugar del principio. El segundo parámetro indica el número de caracteres desde donde se busca, igual que en indexOf.
- **replace(substring_a_buscar,nuevoStr)**
  Implementado en Javascript 1.2, sirve para reemplazar porciones del texto de un string por otro texto, por ejemplo, podríamos uilizarlo para reemplazar todas las apariciones del substring "xxx" por "yyy". El método no reemplaza en el string, sino que devuelve un resultante de hacer ese reemplazo. Acepta expresiones regulares como substring a buscar.

Strings - Métodos

- **split(separador**)
  Sirve para crear un vector a partir de un String en el que cada elemento es la parte del String que está separada por el separador indicado por parámetro.
- **substring(inicio,fin)**
  Devuelve el substring que empieza en el carácter de inicio y termina en el carácter de fin. Si intercambiamos los parámetros de inicio y fin también funciona. Simplemente nos da el substring que hay entre el carácter menor y el mayor.
- **toLowerCase()**
  Pone todas los caracteres de un string en minúsculas.
- **toUpperCase()**
  Pone todas los caracteres de un string en mayúsculas.
- **toString()**
  Este método lo tienen todos los objetos y se usa para convertirlos en cadenas.

Strings - Métodos

Método	Descripción
anchor (Método)	Coloca un delimitador HTML que tiene un atributo NAME a ambos lados del texto.
big (Método)	Coloca etiquetas HTML <BIG> a ambos lados del texto.
blink (Método)	Coloca etiquetas HTML <BLINK> a ambos lados del texto.
bold (Método)	Coloca etiquetas HTML <B> a ambos lados del texto.
charAt (Método)	Devuelve el carácter que se encuentra en el índice especificado.
charCodeAt (Método)	Devuelve la codificación Unicode del carácter que se especifique.
concat (Método, String)	Devuelve una cadena que contiene la concatenación de las dos cadenas proporcionadas.
fixed (Método)	Coloca etiquetas HTML <TT> a ambos lados del texto.
fontcolor (Método)	Coloca etiquetas HTML <FONT> con un atributo COLOR a ambos lados del texto.
fontsize (Método)	Coloca etiquetas HTML <FONT> con un atributo SIZE alrededor del texto.
hasOwnProperty (Método)	Devuelve un valor booleano que indica si un objeto tiene una propiedad con el nombre especificado.
indexOf (Método, String)	Devuelve la posición del carácter donde tiene lugar la primera repetición de una subcadena dentro de una cadena.
isPrototypeOf (Método)	Devuelve un valor booleano que indica si un objeto existe en la cadena de prototipo de otro objeto.
italics (Método)	Coloca etiquetas HTML <I> a ambos lados del texto.
lastIndexOf (Método, String)	Devuelve la última repetición de una subcadena dentro de una cadena.
link (Método)	Coloca un delimitador HTML que tiene un atributo HREF a ambos lados del texto.
localeCompare (Método)	Devuelve un valor que indica si dos cadenas son equivalentes en la configuración regional actual.
match (Método)	Busca una cadena mediante un objeto **Regular Expression** proporcionado y devuelve los resultados como una matriz.

propertyIsEnumerable (Método)	Devuelve un valor booleano que indica si una propiedad especificada forma parte de un objeto y si se puede enumerar.
replace (Método)	Usa una expresión regular para reemplazar texto en una cadena y devuelve el resultado.
search (Método)	Devuelve la posición de la primera coincidencia de subcadena en una búsqueda de expresión regular.
slice (Método, String)	Devuelve una sección de una cadena.
small (Método)	Coloca etiquetas HTML <SMALL> a ambos lados del texto.
split (Método)	Devuelve la matriz de cadenas resultante de la separación de una cadena en subcadenas.
strike (Método)	Coloca etiquetas HTML <STRIKE> a ambos lados del texto.
sub (Método)	Coloca etiquetas HTML <SUB> a ambos lados del texto.
substr (Método)	Devuelve una subcadena que comienza en una posición especificada y tiene una longitud especificada.
substring (Método)	Devuelve la subcadena en la ubicación especificada dentro de un objeto **String**.
sup (Método)	Coloca etiquetas HTML <SUP> a ambos lados del texto.
toLocaleLowerCase (Método)	Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a minúsculas, según la configuración regional actual del entorno de host.
toLocaleString (Método)	Devuelve un objeto convertido en cadena usando la configuración regional actual.
toLocaleUpperCase (Método)	Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a mayúsculas, según la configuración regional actual del entorno de host.
toLowerCase (Método)	Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a minúsculas.
toString (Método)	Devuelve la cadena.
toUpperCase (Método)	Devuelve una cadena en la que todos los caracteres alfabéticos se convierten a mayúsculas.
trim (Método)	Devuelve una cadena donde se han quitado los caracteres de espacio en blanco iniciales y finales y los caracteres de terminador de línea.
valueOf (Método)	Devuelve la cadena.

Strigns-métodos

- Aprende a utilizar los métodos de String aquí:
- https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String#M.C3.A9todos_no_relacionados_con_HTML
- Y realiza los siguientes ejercicios:
- Strings1,2,3,4,5,6,7,8

Numbers

Numbers

```
var x = 34.00;     // Un numero sin decimales
var y = 34;        // Numero con decimals
```

- Numeros largos en format cientifico:

```
var x = 123e5;     // 12300000
var y = 123e-5;    // 0.00123
```

- Numeros en Hexadecimal:

```
var x = 0xFF;              // x será 255
```

Numbers - Infinito

- Infinito: Un valor que JS asigna a los números mayores al más largo posible (15 dígitos)

```
var myNumber = 2;
while(myNumber != Infinity){// Ejecuta hasta el Infinito
     myNumber = myNumber * myNumber;
}
```

También dará Infinito: `var x =  2 / 0;`

```
Si hacemos typeof Infinity; // devolverá "number"
```

Numbers – NaN (Not a Number)

- Existe una palabra reservada para verificar que una variable no es un numero: NaN
- `var x = 100 /“pera";// x será NaN (Not a Number)`
- `var x = 100 / "10";// x será NaN (Not a Number)`

- Podemos utilizar una función global de JS para verificar si una variables es un número o no.

```
var x = 100 / "Apple";
isNaN(x);
```

```
Si hacemos typeof x;          // devolverá "number"
```

Numbers como Objetos

- También podemos crear números como objetos
- `var x = 123;  //typeof devolverá number`
  `var y = new Number(123); //typeof devolverá object`
- `// x == y será true`
- `// x===y será false`

```
var x = new Number(500);
var y = new Number(500);

// (x == y) es falso porque dos objetos no se comparan
con ==
```

Numbers – Propiedades y métodos

- Al igual que con los Strings, aunque no sea objetos, para las propiedades y métodos JS los trata como tales y nos ofrece algunas propiedades y métodos.
- **Propiedades**:
- Para acceder al valor de las propiedades usar Number.propiedad

MAX_VALUE	Devuelve el número más alto en JS
MIN_VALUE	Devuelve el número más bajo en JS
NEGATIVE_INFINITY	Representa infinito negativo
NaN	Representa un NaN
POSITIVE_INFINITY	Representa Infinito positivo

Numbers - Métodos

- A la hora de trabajar con números JS proporciona 2 tipos de métodos.
  - **Metodos de Number** : son métodos que pueden usarse siempre sobre números por ejemplo toString().
  - **Métodos globales** : que pueden usarse sobre otros objetos de JS pero que se usan para trabajar con números. Por ejemplo partseInt().

Numbers- Métodos de numbers

Metodos	Descripción
toString()	Devuelve un número como un String
toExponential()	Devuelve un string, a partir de redondear un número y usando la notación exponencial (cientifica)
toFixed()	Devuelve un string, a partir de redondear un número y usando una cantidad de decimals especificada por parametros
toPrecision()	Devuelve un string de un número, con un determinado número de digitos
valueOf()	Devuelve un número (objeto o number) como un number.

Number- toString()

```
var x = 123;
x.toString();
(123).toString();
(100 + 23).toString();
// Siempre devolverá el String “123”
```

Numbers- Otros Métodos de los objetos numbers

- ```
  var x = 9.656;
  x.toExponential(2);     // devuelve 9.66e+0
  x.toExponential(4);     // devuelve 9.6560e+0
  ```

- ```
  x.toFixed(0);           // devuelve 10
  x.toFixed(2);           // devuelve 9.66
  ```

- ```
  x.toPrecision(2);       // devuelve 9.7
  x.toPrecision(4);       // devuelve 9.656
  ```
  ```

Numbers- Métodos globales para trabajar con números

Metodo	Descripción
Number()	Devuelve un number, convertido a partir del argumento
parseFloat()	Pasa un argumento a un numero con decimals
parseInt()	Pasa un argumento a un entero

Numbers- Number()

```
x = true;
Number(x);          // devuelve 1
x = false;
Number(x);          // devuelve 0
x = new Date();
Number(x);          // devuelve 1404568027739
x = "10"
Number(x);          // devuelve 10
x = "10 20"
Number(x);          // devuelve NaN
```

Numbers – Otros métodos globales

- parseInt -> Devuelve un número entero, permite espacios pero solo coge el primer número.

- ```
  parseInt("10");          // devuelve 10
  parseInt("10.33");       // devuelve 10
  parseInt("10 20 30");    // devuelve 10
  parseInt("10 years");    // devuelve 10
  parseInt("years 10");    // devuelve NaN
  ```
  ```

Numbers – Otros métodos globales

- parseFloat -> Devuelve un número con decimales, permite espacios pero solo coge el primer número.

```
parseFloat("10");          // devuelve 10
parseFloat("10.33");       // devuelve 10.33
parseFloat("10 20 30");    // devuelve 10
parseFloat("10 years");    // devuelve 10
parseFloat("years 10");    // devuelve NaN
```

Math

Math

- El objeto global Math nos permite realizar operaciones matemáticas y otras tareas más complejas que los operadores sobre números no nos permiten.

Math - Métodos

- abs(x) Devuelve el valor absolute de x
- acos(x) devuelve el arcocoseno de x en radianes
- asin(x) devuelve el arcoseno de x en radianes
- atan(x) devuelve el arcotangente de x en radianes
- cos(x) devuelve el coseno de x (x en radianes)
- Ceil(x) devuelve el entero redondeado más cercano por arriba

Math- Métodos

- exp(x) devuevle el valor de e elevado a x.
- floor(x) devuelve el valor de x redondeado hacia abajo.
- log(x) devuelve el logaritmo en base de E de x.
- max(x,y,z,...,n) devuelve el valor máximo de una lista de números.
- min(x,y,z,...,n) devuelve el valor mínimo de una lista de números.
- pow(x,y) devuelve la potencia y de x.
- random() devuelve un número aleatorio entre 0 y 1.
- round(x) redondea x al valor entero más cercano.
- sin(x) devuelve el seno de x.
- sqrt(x) devuelve la raiz cuadrada de x.
- tan(x) devuelve la tangente de x.

Algunos ejemplos de uso.

- `Math.random();` `// devuelve un numero aleatorio`
- Math.max(0, 150, 30, 20, -8); `// devuelve 150`
- Math.ceil(4.7); `// devuelve 5`
- Math.floor(4.7); `// devuelve 4`

Math - Constantes

- Las constantes son propiedades(como variables) que no cambiarán su valor nunca. Se suelen escribir en mayúsculas para diferenciarlas de las variables.
- El objeto Math tiene algunas constantes interensantes como:
  - Math.PI `// devuelve el número Pi`

- Realiza los ejercicios Math1,2,3,4,5

Fechas (Date)

Mostrando fechas

- Una fecha en JS puede ser escrito como un String
  - **Tue May 05 2015 19:08:53 GMT+0200 (Hora de verano romance)**
  - O como un numero
    - **1430845733639**

- Las fechas escritas como numero especifican el número e milisegundos que han pasado desde el 1 de Enero de 1970.

- ```

  <script>
  document.getElementById("demo").innerHTML = new Date();
  </script>
  ```
  ```

Creando fechas

- Podemos crear fechas de 4 formas distintas:

```
new Date()  // fecha actual como hemos visto
new Date(milliseconds)//Nuevo date con la fecha que le pasemos en milisegundos.
new Date(dateString)//Nuevo date con la fecha que le pasemos como texto

new Date(year, month, day, hours, minutes, seconds, milliseconds) )//Nuevo date con
la fecha que le pasemos con los valores de año, mes etc….(numeros)
```

Creando fechas.

- Ejemplos:
- `var d = new Date("October 13, 2016 10:13:00");`
- var d = new Date(86400000);
- var d = new Date(99,5,24,11,33,30,0);
- var d = new Date(99,5,24); `)//no es necesario pasarle todos los parametros`

Formato de Fecha

- Podemos mostrar las fechas con un formato más agradable:
- ```
  <script>
  var d = new Date();
  document.getElementById("demo").innerHTML = d.toDateString();
  </script>
  ```

- Realiza los ejercicios Date1,2,3
  ```

Formato de Fecha

- Existen generalmente 3 formatos de fecha validos en JS:
  - ISO Dates (formato estándar de fecha)
  - Long Dates(formato de fecha largo)
  - Short Dates (formato de fecha corto)

Formato de fecha ISO

- YYYY-MM-DDTHH:MM:SS
- Puede utilizarse solamente parte del formato, ejemplo:

```
Var d = new Date("2015-03-25");
var d = new Date("2015-03");
var d = new Date("2015");
var d = new Date("2015-03-25T12:00:00");
```

Formato Largo

MMM DD YYYY

- También podemos cambiar el orden de los valores. JS los diferencia por el número de caracteres.
- var d = new Date("Mar 25 2015");
- var d = new Date("25 Mar 2015");

Formato Corto

- MM/DD/YYYY
- Podemos usar tanto / como –
  - var d = new Date("03/25/2015");
  - var d = new Date("03-25-2015");
- Tambien acepta "YYYY/MM/DD“
  - var d = new Date("2015/03/25");

Date – Métodos

- Una vez que tenemos los objeto Date podemos utilizar sus métodos a nuestro antojo.
- Tenemos métodos get que nos devuelven información sobre la fecha:

Metodo	Descripción
getDate()	Nos devuelve un numero del 1 al 31
getDay()	Nos devuelve un numero del 0 al 6
getFullYear()	Nos devuelve el año (yyyy)
getHours()	Nos devuelve la hora (0-23)
getMilliseconds()	Nos devuelve los milisegundos(0-999)
getMinutes()	Nos devuelve los minutos (0-59)
getMonth()	Nos devuelve el numero de mes (0-11)
getSeconds()	Nos devuelve los segundos (0-59)
getTime()	Nos devuelve la hora (milisegundos desde 1, 1970)

Date – Métodos set

- También métodos para modificar las propiedades del objeto (métodos set)

setDate()	Modifica el día (1-31)
setFullYear()	Modifica el año y opcionalmente el mes y el día
setHours()	Modifica la hora (0-23)
setMilliseconds()	Modifica los milisegundos (0-999)
setMinutes()	Modifica los minutos(0-59)
setMonth()	Modifica el mes (0-11)
setSeconds()	Modifica los segundos (0-59)
setTime()	Modifica la fecha en milisegundos desde el 1 de enero 1970)

Comparar fechas

- Podemos usar el operador > y < para saber si una fecha es posterior a otra:

```
var hoy, otrodia,

hoy = new Date();
otrodia = new Date();
otrodia.setFullYear(2100, 0, 14);

if (otro > hoy) {

//hoy es antes que otro día

} else {
  //hoy es despues de otro día
}
```

Arrays

Arrays

- Una matriz (array) es una variable especial , con capacidad para más de un valor a la vez. Si tenemos una lista de elementos (una lista de nombres de coche , por ejemplo) , el almacenamiento de los coches en las variables individuales podría tener este aspecto:

```
var car1 = "Saab";
var car2 = "Volvo";
var car3 = "BMW";

…..

….
```

Para solucionar este problema existen los Arrays, que almacenan varios valores en una sola variable

Arrays

- La sintaxis de la definición de arrays es al siguiente:
  - var *nombreArray* = [*elemento1*, elemento2, ...];
- Ejemplo: `var cars= [Saab", "Volvo", "BMW“];`

- Aunque realmente un Array es un objeto predefinido en JavaScript, por lo que tambien podemos crear el array de la siguiente forma:

```
var cars = new Array("Saab", "Volvo", "BMW");
```

Arrays – Acceso a sus elementos

- Para acceder a los elementos de un array, debemos indicar entre corchetes el índice del elemento al que queremos accerder:

```
var cars= ["Saab", "Volvo", "BMW“];
var nombreCoche= cars[0];
```

- Los indices en los arrays comienzan por 0. En nuestra nueva variable almacenaremos el valor “Saab” .
- De igual forma también podemos modificar el contenido de los arrays:

```
cars[0] = “Seat”
```

Arrays – Elementos de distintos tipos

- Los elementos de un Array no tienen porque ser objetos del mismo tipo :

```
myArray[0] = Date.now;
myArray[1] = myFunction;
myArray[2] = myCars;
```

- Podemos tener cualquier tipo de objeto, incluso funciones y otros arrays

Arrays – Es un objeto

- Realmente un Array es un tipo especial de objetos, donde para acceder a sus propiedades no usamos el nombre de la propiedad sino el índice (su posición dentro del objeto):
- Ejemplo de acceso al nombre de una persona en un array.

```
var persona = ["John", "Doe", 46];
var nombre= persona[0];
```

- Ejemplo de acceso al nombre de una persona en un objeto.

```
var persona = {nombre:"John", apellido:"Doe", edad:46};
var nombre= persona[“nombre"];
```

Arrays- Propiedades y Métodos

- Como todo Objeto de JS tiene sus propiedades y métodos que nos ayudarán a trabajar con Arrays:
- Propiedades

Propiedad	Descripción
length	Es la longitud del array

Arrays- Propiedades y Métodos

Method	Description
concat()	Une dos o más Arrays y devuelve la unión
indexOf()	Devuelve la posición de un elemento del array
join()	Une todos los elementos de un array en un String
lastIndexOf()	Busca un elemento del array empezando por el final y devuelve su posición
pop()	Elimina el último elemento del array
push()	Añade un nuevo elemento, como último del array y devuelve la nueva longitud
reverse()	Reordena los elementos del array de forma inversa
shift()	Elimina el primer elemento del array y nos lo devuelve
slice()	Selecciona una parte del array y nos devuelve el nuevo array
sort()	Ordena los elementos de un array
splice()	Añade/elimina elementos de un array
toString()	Convierte un array en un String y devuelve el resultado
unshift()	Añade nuevos elementos al principio de un array y devuelve la nueva longitud
valueOf()	Devuelve el valor primitivo de un array

Trabajando con Arrays

- Tenemos el detalle de como se usan estos elementos en la siguiente dirección:
- https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Array
- Ejercicios: En un documento HTML, crea varios Arrays y prueba todos los métodos especificados en la diapositiva anterior