GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

ies SERPIS | departament informàtica

Unidad 2 – Objetos predefinidos del lenguaje

DWEC - 2º DAW

Rafael Torres

Unidad 2 – Índice

1. Objetos predefinidos del lenguaje Javascript

Objetos predefinidos del lenguaje: son propios del lenguaje y existen independientemente del contexto.

Como objetos que son, todos disponen de propiedades y métodos.

- Objeto String: para representar y operar con cadenas de texto.
- Objeto Number: para representar números.
- Objeto Math: para representar constantes y funciones matemáticas.
- Objeto Date: para representar fechas.
- Objeto RegExp: para representar expresiones regulares.

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

El objeto String es un constructor de cadenas. De hecho, no es exactamente lo mismo el tipo primitivo string que un objeto String.

```
var cad1 = 'Esto es una cadena de texto';
var cad2 = new String('Esto es una cadena de texto');
console.log(typeof(cad1)); // string
console.log(typeof(cad2)); // object
console.log(cad1 == cad2); //true
console.log(cad1 === cad2); //false
```

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

En JavaScript, aunque las cadenas de texto (strings) son **tipos primitivos**, el lenguaje permite que puedas acceder a métodos y propiedades en ellas como si fueran objetos. Esto sucede gracias a un mecanismo que se conoce como **autoencapsulamiento** o **autoboxing**.

- **Autoboxing**: Cada vez que llamas a un método o propiedad de un primitivo, JavaScript crea una instancia temporal de la clase String. Es como si internamente JavaScript hiciera algo similar a new String("Hola"), pero solo mientras se ejecuta el método.

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

Ejemplo del autoboxing:

```
let texto = "Hola";

// Accedemos a una propiedad del objeto String
console.log(texto.length); // 4

// Llamamos a un método del objeto String
console.log(texto.toUpperCase()); // "HOLA"

// El valor sigue siendo primitivo
console.log(typeof texto); // "string"
```

Esto pasa con otros tipos primitivos como Number y Boleean.

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

Propiedades
Prototype: permite añadir propiedades y métodos al objeto
Constructor: devuelve la función constructora de la cadena
Length: devuelve la longitud de la cadena

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

Métodos
charAt(x): devuelve el carácter de la posición x (comenzando por el 0)
concat(str): concatena la cadena str en la cadena original
startsWith(str): comprueba si la cadena comienza con los caracteres o cadena especificados.
endsWith(str): comprueba si la cadena termina con los caracteres o cadena especificados.
includes(str): comprueba si la cadena contiene los caracteres o cadena especificados.
indexOf(str): devuelve la posición de la primera ocurrencia de la cadena que pasamos como parámetro
lastIndexOf(str): devuelve la posición de la última ocurrencia de la cadena que pasamos como parámetro
match(regexp): busca en una cadena comparándola con la expresión regular regexp, y devuelve las coincidencias en un array
repeat(x): devuelve una nueva cadena que es la repetición de la cadena tantas veces como el valor del parámetro.

1.1. Objetos predefinidos del lenguaje Javascript: El objeto String

Métodos
xreplace(str1, str2): busca dentro de la cadena los caracteres (o expresión regular) str1, y devuelve una cadena con estos valores reemplazados por el segundo parámetro str2
search(str): busca dentro de la cadena los caracteres (o expresión regular) str, y devuelve la posición de la primera ocurrencia.
slice(x,y): extrae una parte de una cadena entre los caracteres x e y, y devuelve una nueva cadena.
split(car): separa una cadena en un array de subcadenas, cogiendo como separador el carácter car.
substr(x, y): extrae caracteres de una cadena, empezando en la posición x, y el número de caracteres especificado por y
substring(x, y): extrae caracteres de una cadena entre los dos índices especificados, x e y.
toLowerCase(): convierte una cadena a minúsculas
toUpperCase(): convierte una cadena a mayúsculas
trim(): elimina los espacios en blanco de ambos extremos de la cadena
toString(): devuelve el valor de un objeto
StringvalueOf(): devuelve el tipo primitivo de un objeto String

```
const minusculas = texto.toLowerCase();
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

En JavaScript, a diferencia de otros lenguajes de programación, sólo existe un tipo de datos para los números. Todos los números se guardan en memoria con una resolución de 64 bits (doble precisión) y coma flotante.

```
var a = 20;
var b = -2.718;
var c = 1e4;
var d = 2.23e-3
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Además de la notación decimal, podemos representar a los números en binario, octal o hexadecimal.

```
console.log(0b1010)
console.log(0xFF)
```

```
PS C:\Users\bea_v\Documents\SERPIS_24_25\dwec\ejercicios\2.objetos_predefinidos> node .\2.Number.js
10
255
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Y con el método .toString() podemos representar los números en distintas notaciones:

```
var num = 76;
console.log(num.toString(16) + " en hexadecimal");
console.log(num.toString(8) + " en octal");
console.log(num.toString(2) + " en binari");
console.log(typeof(num.toString(2))); //string
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Infinity es el valor para representar que hemos sobrepasado la precisión de los números (que es de hasta 15 dígitos en la parte entera):

```
console.log(5/0); // devuelve infinity
```

Not a Number (NaN) es una palabra reservada para indicar que el valor no representa un número. Podemos utilizar la función global isNaN() para saber si el argumento es un número:

```
console.log(3*"a"); //NaN. Recordad que 3+"a" devuelve "3a"
console.log(isNaN(3*4)); //false
console.log(isNaN(3*"a")); //true
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Al igual que ocurre con las cadenas, en JavaScript normalmente los números son tipos primitivos, pero también los podemos crear como objetos:

```
var x = 25; //tipo primitivo
var y = new Number(25); //object
console.log(typeof x); //number
console.log(typeof y); //object
console.log(x == y); //true
console.log(x === y); //false
```

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Propiedades
Prototype: permite añadir propiedades y métodos al objeto
NaN: representa un valor no numérico

1.2. Objetos predefinidos del lenguaje Javascript: El objeto Number

Métodos
isFinite(): comprueba si un valor es un número finito
isInteger(): comprueba si un valor es un entero
isNaN(): comprueba si un valor es un NaN
toExponential(x): representa un numero con su notación exponencial. Por ejemplo, 20,31 a 2.031e+1, que significa 2.031 * 10^1.
toFixed(n): formatea un numero con una precisión decimal de n dígitos. Devuelve una cadena que es la representación del número
toPrecision(n): formatea un numero a una precisión de n digitos (número total de dígitos, incluyendo la parte entera y la decimal).
toString(): convierte un número en una cadena
parseInt(): convierte un número entero.
parseFloat(): convierte un número a decimal
valueOf(): devuelve el valor del tipo primitivo del número

1.3. Objetos predefinidos del lenguaje Javascript: El objeto Math

El objeto Math es especial en el sentido de que no tiene constructor. No existe ningún método para crear un objeto Math. Podemos utilizarlo sin tener que crearlo.

Propiedades
JavaScript tiene almacenadas las principales constantes matemáticas (número pi, número e, etc.), y para acceder a él lo hacemos a través de las propiedades correspondientes del objeto Math: Math.E, Math.LN2, Math.LN10,Math .SQRT2...

1.3. Objetos predefinidos del lenguaje Javascript: El objeto Math

Métodos
abs(x): valor absoluto
sin(x), cos(x), tan(x): funciones trigonométricas asin(x), acos(x), atan(x): devuelve en radianes el arcsinus, arccosinus, arctangent
round(x): redondea x al valor entero más cercano
ceil(x), floor(x): devuelve el mismo número pero redondeado al entero más cercano hacia arriba (ceil) o hacia abajo (floor)
truncx(x): elimina la parte fraccionaria de un número (y queda sólo la parte entera).
exp(x): devuelve el valor $eˆx$.
max(a,b,...), min(a,b,...): devuelve el mayor valor (o más pequeño)de la lista de números
pow(x,y): devuelve $xˆy$.
log(x): devuelve el logaritmo natural (base e) de x
random(): devuelve un número aleatorio entre 0 (incluido) y 1 (excluido)
sqrt(x): devuelve la raíz cuadrada de x

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

El objeto Date se utiliza para trabajar con fechas y tiempo

```
var data = new Date(); //fecha del sistema
console.log(data);
var data = new Date(34885453664); //genera una fecha
que representa los milisegundos que han pasado desde
el 1 de enero de 1970
console.log(data);
data = new Date('2016/05/23');
console.log(data);
data = new Date(2016,5,23,12,15,24,220); //año, mes,
dia, hora, minutos, segundos, milisegundos
console.log(data);
```

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

PROPIEDADES
constructor: devuelve la función que ha creado el prototipo del objeto Date
prototype: permite añadir propiedades y métodos al objeto.

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

MÉTODOS
getDay(): devuelve el día de la semana (0-6, empezando por domingo).
getFullYear(): devuelve el año (4 dígitos)
getMonth(): devuelve el mes (0-11).
getDate(): devuelve el día del mes (1-31)
getHours(): devuelve la hora (0-23)
getMinutes(): devuelve los minutos (0-59).
getSeconds(): devuelve los segundos (0-59).
getMilliseconds(): devuelve los milisegundos (0-999).
getUTCDate(): devuelve el día del mes (1-31), de acuerdo con el horario UTC universal.
getUTCDay(): devuelve el día de la semana (0-6, empezando por domingo), de acuerdo con el horario UTC universal.
getUTCFullYear(): devuelve el año (4 dígitos), de acuerdo con el horario UTC universal.
getUTCMonth(): devuelve el mes (0-11), de acuerdo con el horario UTC universal.
getUTCHours(): devuelve la hora (0-23), de acuerdo con el horario UTC universal.
getUTCMinutes(): devuelve los minutos (0-59), de acuerdo con el horario UTC universal.

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

MÉTODOS
getUTCSeconds(): devuelve los segundos (0-59), de acuerdo con el horario UTC universal.
getUTCMilliseconds(): devuelve los milisegundos (0-999), de acuerdo con el horario UTC universal.
getTime(): devuelve el número de milisegundos que han transcurrido desde la fecha hasta el 1 de enero de 1970.
now(): devuelve el número de milisegundos que han transcurrido desde la fecha del sistema hasta el 1 de enero de 1970.
parse(): transforma una cadena de texto (representando una fecha), y devuelve el número de milisegundos transcurridos desde la fecha hasta el 1 de enero de 1970.
setFullYear(): especifica el día del año.
setMonth(): especifica el mes.
setDate(): especifica el día del mes.
setHours(): especifica la hora.
setMinutes(): especifica los minutos.
setSeconds(): especifica los segundos.
setMilliseconds(): especifica los milisegundos.
setTime(): especifica una fecha a partir del número de milisegundos antes o después del 1 de enero de 1970.

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

MÉTODOS
setUTCFullYear(): especifica el año, de acuerdo con el horario UTC universal.
setUTCMonth(): especifica el mes, de acuerdo con el horario UTC universal
setUTCDate(): especifica el día del mes, de acuerdo con el horario UTC universal.
setUTCHours(): especifica la hora, de acuerdo con el horario UTC universal
setUTCMinutes(): especifica los minutos, de acuerdo con el horario UTC universal.
setUTCSeconds(): especifica los segundos, de acuerdo con el horario UTC universal.
setUTCMilliseconds(): especifica los milisegundos, de acuerdo con el horario UTC universal.
toDateString(): convierte la parte de la fecha en una cadena legible.
toLocaleDateString(): convierte la parte de la fecha en una cadena legible, utilizando las convenciones locales.
toISOString(): convierte la fecha a cadena, utilizando la convención ISO
toJSON(): convierte la fecha a cadena con formato JSON.
toTimeString(): convierte la parte del timestamp del objeto Date a cadena.
toLocaleTimeString(): convierte la parte del timestamp del objeto Date, utilizando las convenciones locales.
toString(): convierte el objeto Date a cadena.
toLocaleString(): convierte el objeto Date a cadena, utilizando las convenciones locales

1.4. Objetos predefinidos del lenguaje Javascript: El objeto Date

```
var d= new Date(); //Dia del Trabajo
d.setDate(1);
d.setMonth(4); //Representa el mes de mayo
d.setFullYear (2016);
console.log(d);
console.log(d.toDateString()); //Sun May 01 2016
console.log(d.toLocaleDateString()); //1/5/2016
//Como toLocaleDateString() devuelve una cadena, ya
puedo utilizar las funciones de cadena:
var array_data = d.toLocaleDateString().split('/');
console.log(array_data[0]); //dia
console.log(array_data[1]); //mes
console.log(array_data[2]); //año
```

1.5. Objetos predefinidos del lenguaje Javascript: El objeto RegExp

Las expresiones regulares son objetos de JavaScript que se utilizan para describir un patrón de caracteres.

Aplicadas a las cadenas de texto, con las expresiones regulares podemos saber si una cadena de texto cumple un patrón, y realizar funciones de búsqueda y reemplazo.

En una expresión regular distinguimos entre el patrón y los modificadores:

– /pattern/modificadores;

i→ case insensitive

```
var patro = /IOC/i
```

1.5. Objetos predefinidos del lenguaje Javascript: El objeto RegExp

Para ver el funcionamiento básico de las expresiones regulares veremos los métodos test y match:

- test: método de las expresiones regulares que devuelve true o false si la cadena de texto cumple el patrón.
- match: método del objeto String que realiza una búsqueda de la cadena contra un patrón, y devuelve un array con los resultados encontrados, o el valor primitivo null en caso de que no encuentre ninguno.

1.5. Objetos predefinidos del lenguaje Javascript: El objeto RegExp

```
var patrol = /serpi/i
var patro2 = /serpi/
var cad="Curso de JavaScript en el Serpis";
console.log(patrol.test(cad)); //true
console.log(patro2.test(cad)); //false
var res = cad.match(patrol);
console.log(res.length); //1, una sola ocurrencia
console.log(res[0]); //Serpi
```

1.5. Objetos predefinidos del lenguaje Javascript: El objeto RegExp

Dentro de una expresión regular podemos utilizar:

- brackets para expresar un rango de caracteres, y metacaracteres, que tienen un significado especial.
- Con el circunflejo (ˆ) estamos indicando cómo debe empezar la expresión. Con el dólar ($) estamos indicando cómo debe terminar la expresión.
- Utilizamos la contrabarra (\) para indicar que la barra (/) no es un metacarácter, sino un carácter a tener en cuenta.
- El * después de un carácter, significa "0 o más apariciones del elemento anterior". Ej: /ab*c/
- x{n} concuerda con exactamente n apariciones del elemento x
- [xyz] representa cualquier carácter de entre los que se muestran en los brackets [0-9] representa cualquier carácter comprendido en el rango expresado
- Hoja de referencia de expresiones regulares

Conclusiones

Los **objetos predefinidos** en JavaScript sirven como herramientas fundamentales para realizar operaciones comunes y específicas dentro del lenguaje, sin necesidad de que los programadores tengan que escribir su propia lógica desde cero.

Cada uno de estos objetos tiene un propósito particular y **facilita tareas** que van desde la manipulación de datos básicos hasta operaciones avanzadas en estructuras de datos o patrones de texto.