Objetos predefinidos - String

Lucas Márquez Muñoz

Objetos predefinidos - String

- Definición y constructor
- Cadenas primitivas y objeto String
- Accediendo a elementos de una cadena
- Métodos
- Cadenas de plantilla Multi-línea
- Cadenas y RegEx
- Métodos relacionados con HTML
- Bibliografía
- Dudas y preguntas

Lucas Márquez Muñoz

Objetos predefinidos - String

- El objeto String representa una serie de caracteres dentro de una cadena.
- Constructor: new String(cadena):

```
> cadena = new String("cadena");
< ▶ String {0: "c", 1: "a", 2: "d", 3: "e", 4: "n", 5: "a", length: 6,
  [[PrimitiveValue]]: "cadena"}
```

- Las cadenas literales también crean cadenas:

```
> primitiva = "cadena";
< "cadena"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

- JavaScript convierte automáticamente entre cadenas primitivas y objetos de cadena.
- Si llamamos a un método del objeto String en una cadena primitiva, JavaScript la convierte en un objeto String temporal de forma automática y luego lo descarta.

```
> primitiva = "cadena";
<- "cadena"
> primitiva.length
<- 6
> 'casa'.length
<- 4
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Las cadenas primitivas y los objetos String dan resultados diferentes al ser evaluados en JavaScript:

- Las cadenas primitivas son tratadas como código fuente.
- Los objetos String son tratados como un objeto de secuencia de caracteres.

Los objetos String pueden convertirse a cadenas primitivas con String.valueOf()

Lucas Márquez Muñoz

Objetos predefinidos - String

- Cada elemento en la cadena ocupa una posición, estando el primero en el índice 0.
- Dos formas de acceder a los caracteres individuales de una cadena (El primer método no forma parte de ECMAScript, es una característica de JavaScript):

```
> cadena = "cadena"
<- "cadena"
> cadena[2]
<- "d"
> cadena.charAt(2)
<- "d"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Algunos métodos:
charAt(): Devuelve el carácter en el índice especificado.
charCodeAt(): Devuelve el valor Unicode del carácter en el índice especificado.

```
> ejemplo = new String('ejemplo');
< ▶ String {0: "e", 1: "j", 2: "e", 3: "m", 4: "p", 5: "l", 6: "o", length: 7,
    [[PrimitiveValue]]: "ejemplo"}
> ejemplo.charCodeAt(0)
< 101
```

concat(): Combina el texto de dos cadenas y devuelve una nueva:

```
> hola = "hola"
< "hola"
> yAdios = " y adiós"
< " y adiós"
> hola.concat(yAdios)
< "hola y adiós"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

indexOf(): Devuelve el índice de la primera ocurrencia del valor especificado (-1 si no encuentra)

```
> ejemplo.indexOf('e')
<- 0
```

lastIndexOf(): Devuelve el índice de la última ocurrencia del valor especificado (-1 si no encuentra)

```
> ejemplo.lastIndexOf('e')
<- 2
```

slice(Inicio, [fin)): Extrae una sección de una cadena y devuelve una cadena nueva

```
> ejemplo.slice(2,5)
<- "emp"
```

```
> ejemplo.slice(-5,5)
<- "emp"
```

```
> ejemplo.slice(2,-2)
<- "emp"
```

```
> ejemploPartido = ejemplo.slice(2,5)
<- "emp"
> ejemploPartido
<- "emp"
```

substring(), substr(): Igual que slice, pero si alguno de los valores es menor de 0 es tratado como si fuera 0

```
> ejemplo.substring(-5,5)
<- "ejemp"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

toLowerCase(): Devuelve la cadena con todas las letras en minúsculas

```
> cadenaMixta = "    Espero que la presentación vaya bien    "
<- "    Espero que la presentación vaya bien    "
> cadenaMixta.toLowerCase()
<- "    espero que la presentación vaya bien    "
```

toUpperCase(): Devuelve la cadena con todas las letras en mayúsculas

```
> cadenaMixta.toUpperCase()
<- "    ESPERO QUE LA PRESENTACIÓN VAYA BIEN    "
```

trim(): Elimina los espacios en blanco que haya al comienzo y al final de la cadena

```
> cadenaMixta.trim()
<- "Espero que la presentación vaya bien"
```

(trimLeft() y trimRight() no estandarizados)

```
> cadenaMixta.trimLeft()
<- "Espero que la presentación vaya bien    "
```

```
> cadenaMixta.trimRight()
<- "    Espero que la presentación vaya bien"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Cadenas de plantilla multi-línea

- Las cadenas de plantilla multi-línea son literales de cadenas que permiten expresiones incrustadas y, además, pueden utilizar cadenas de varias líneas.
- Se crean con el carácter (`)(acento grave) en lugar de las comillas simples o dobles.

```
> console.log("Línea de texto 1 \n\Línea de texto 2");
  Línea de texto 1
  Línea de texto 2
```

```
> console.log(`Línea de texto 1
  Línea de texto 2`);
  Línea de texto 1
  Línea de texto 2
```

expresiones incrustadas:

```
> console.log("Cinco es " + (a + b) + " y no " + (2 * a + b) + ".");
  Cinco es 5 y no 7.
```

```
> console.log(`5 es ${a + b} y no ${2 * a + b}.`);
  5 es 5 y no 7.
```

```
> console.log(`5 es {a + b} y no {2 * a + b}.`);
  5 es {a + b} y no {2 * a + b}.
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Cadenas y expresiones regulares:
match(regexp): Se utiliza para obtener todas las ocurrencias de una expresión regular dentro de una cadena

```
> texto ="Hola mi nombre es Lucas y mi DNI es 30983933N y tengo, además, 29
  años";
<- "Hola mi nombre es Lucas y mi DNI es 30983933N y tengo, además, 29 años"
> expresion =/[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKET]{1}/;
<- /[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKET]{1}/
> dni = texto.match(expresion);
<- ▶ ["30983933N"]
> letraDNI = dni[0].charAt(8);
<- "N"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Cadenas y expresiones regulares:
replace(regexp, texto): Se utiliza para obtener todas las ocurrencias de una expresión regular dentro de una cadena

```
> nuevoTexto = texto.replace(expresion, "*********");
< "Hola mi nombre es Lucas y mi DNI es ********* y tengo, además, 29 años"
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Cadenas y expresiones regulares:
search(regexp): Ejecuta la búsqueda que encaje entre una expresión regular y el objeto String desde el que se lo llama

```
> confirmaDNI = function(textoIntroducido){
  dniIntroducido = textoIntroducido.search(expresion);
  if(dniIntroducido != -1)
  document.write("Se ha introducido un DNI");
  else
  document.write("No se ha introducido un DNI");
  }
< function (textoIntroducido){
  dniIntroducido = textoIntroducido.search(expresion);
  if(dniIntroducido != -1)
  document.write("Se ha introducido un DNI");
  else
  document.write("No se ha introducido un DNI"…
> confirmaDNI(texto);
< undefined
> confirmaDNI(hola);
< undefined
```

Lucas Márquez Muñoz

Objetos predefinidos - String

Métodos relacionados con HTML:
bold()
fontcolor(color)
fontsize()
link()

```
> texto = "Moodle";
<- "Moodle"
> urlMoodle = "https://moodle.iesgrancapitan.org";
<- "https://moodle.iesgrancapitan.org"
> console.log("Enlace a " + texto.link(urlMoodle));
  Enlace a <a href="https://moodle.iesgrancapitan.org">Moodle</a>
```

Lucas Márquez Muñoz

Bibliografía

https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/String

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Text_formatting

Dudas y preguntas