DESARROLLO WEB EN ENTORNO CLIENTE

UD3 – OBJETOS PREDEFINIDOS DEL LENGUAJE (Parte 2)

CICLO FORMATIVO DE GRADO SUPERIOR EN DESARROLLO DE APLICACIONES WEB

I.E.S. HERMENEGILDO LANZ – 2022/2023

PROFESORA: VANESA ESPÍN

vespin@ieshlanz.es

Índice

PARTE 1

1. Introducción
2. Objetos predefinidos de más alto nivel en JavaScript
   a. windows
   b. document
   c. Form
   d. Expresiones regulares

PARTE 2

3. Objetos Nativos en JavaScript
   a. String
   b. Math
   c. Number
   d. Boolean
   e. Date
4. Las cookies

El Objeto String

El objeto String se utiliza para representar y manipular una cadena de caracteres.

```
var  miCadena = new  String("texto de la cadena");

var miCadena = "texto de la cadena";

cadena.propiedad;
cadena.metodo( [parámetros] );
```

Métodos y propiedades del objeto String para practicar en:
https://www.w3schools.com/js/js_string_methods.asp

PROPIEDADES Y MÉTODOS DEL OBJETO STRING

Métodos	Descripción
charAt()	Devuelve el carácter especificado por la posición que se indica entre paréntesis.
charCodeAt()	Devuelve el **Unicode** del carácter especificado por la posición que se indica entre paréntesis.
concat()	Une una o más cadenas y devuelve el resultado de esa unión.
fromCharCode()	Convierte valores Unicode a caracteres.
indexOf()	Devuelve la posición de la primera ocurrencia del carácter buscado en la cadena.
lastIndexOf()	Devuelve la posición de la última ocurrencia del carácter buscado en la cadena.
match()	Busca una coincidencia entre una expresión regular y una cadena y devuelve las coincidencias o null si no ha encontrado nada.
replace()	Busca una subcadena en la cadena y la reemplaza por la nueva cadena especificada.
search()	Busca una subcadena en la cadena y devuelve la posición dónde se encontró.
slice()	Extrae una parte de la cadena y devuelve una nueva cadena.
split()	Divide una cadena en un array de subcadenas.
substr()	Extrae los caracteres de una cadena, comenzando en una determinada posición y con el número de caracteres indicado.
substring()	Extrae los caracteres de una cadena entre dos índices especificados.
toLowerCase()	Convierte una cadena en minúsculas.
toUpperCase()	Convierte una cadena en mayúsculas.

Propiedad	Descripción
length	Contiene la longitud de una cadena.

Ejercicio 11 – Uso de objeto String

Dentro de un elemento <section> realiza un formulario con los siguientes campos:

Cadena: de texto. Será nuestro objeto String para trabajar.

Vocal: lista desplegable donde el usuario escogerá una vocal (hecha con bucle for)

Subcadena: de texto.

Al enviar el formulario, debajo añadiremos, en un <p> “resultado”, lo siguiente (recuerda ir actualizando el objeto cadena, no ir creando variables nuevas a no ser que sea necesario):

a. Ocurrencias de *vocal* en *cadena*. Escribiremos por ejemplo: “a aparece en ‘casa’ 2 veces”

b. *cadena* pero agregando después de cada letra su índice. Solo imprimir, no guardar cadena. Ejemplo: dada *casa* : "*c0a1s2a3*"

c. Si aparece la subcadena en la cadena (ignorando mayúsculas y minúsculas) y si es así, cuantas veces y en qué posición.

d. Reemplazamos todas las ocurrencias de la subcadena con la nueva subcadena “dwec”.

e. Eliminamos ahora todas las ocurrencias de “dwec” de la variable cadena.

f. Resultado de unir cadena (actual), vocal y subcadena en una nueva variable *nuevaCadena*. Deben ir serparadas por espacios.

Ejercicio 12

Cifrado estilo César con el objeto String

Consiste en tomar cada letra de un mensaje y desplazarla en el alfabeto el número que diga una clave. Por ejemplo, A desplazada con clave 2 es C.

- Crea una página que pida al usuario un texto y una clave y traduzca el texto desplazando los caracteres usando los caracteres de la tabla Unicode (no hace falta que la Y sea la A y la Z sea B, pero se puede hacer esta versión también)

El Objeto Math

- El objeto Math permite realizar tareas matemáticas en números.
- El objeto:
  - no tiene constructor.
  - es estático.
  - Los métodos y propiedades se pueden usar sin crear primero un objeto matemático.
- Conoceremos:
  - Las 8 constantes de Math
  - Los métodos. Sintaxis: `Math.method(number);`
    - ✓ Matemáticos
    - ✓ Random
    - ✓ Logaritmos
    - ✓ Redondeos
    - ✓ Trigonométricos

ENLACES PARA ESTUDIAR EL OBJETO MATH:
https://www.w3schools.com/js/js_math.asp
https://lenguajejs.com/javascript/number/objeto-math/

El Objeto Number

- Number es un objeto primitivo que permite representar y manipular valores numéricos.
- El constructor Number contiene constantes y métodos para trabajar con números.
- Valores de otro tipo pueden ser convertidos a números usando la función Number(). No confundir constructor con función.

```
new Number(value);
var a = new Number('123'); // a === 123 es false ya que a es una instancia de objeto
var b = Number('123'); // b === 123 es true ya que b es un valor
a instanceof Number; // es true
b instanceof Number; // es false
```

El Objeto Number. Propiedades

The following table lists the standard properties of the Number object.

Property	Description
MIN_SAFE_INTEGER	Represents the maximum safe integer in JavaScript (2^{53} - 1).
MAX_VALUE	Returns the largest numeric value representable in JavaScript, approximately 1.79E+308. Values larger than MAX_VALUE are represented as Infinity.
MIN_SAFE_INTEGER	Represents the minimum safe integer in JavaScript (-(2^{53} - 1)).
MIN_VALUE	Returns the smallest positive numeric value representable in JavaScript, approximately 5e-324. It is closest to 0, not the most negative number. Values smaller than MIN_VALUE are converted to 0.
NEGATIVE_INFINITY	Represents the negative infinity value.
NaN	Represents "Not-A-Number" value.
POSITIVE_INFINITY	Represents the infinity value.
prototype	Allows you to add new properties and methods to a Number object.

El Objeto Number. Métodos

The following table lists the standard methods of the Number object.

Method	Description
isFinite()	Checks whether the passed value is a finite number.
isInteger()	Checks whether the passed value is an integer.
isNaN()	Checks whether the passed value is NaN and its type is Number.
isSafeInteger()	Checks whether a value is a safe integer.
toExponential()	Converts a number to exponential notation.
toFixed()	Formats a number using fixed-point notation.
toPrecision()	Returns a string representing the number to the specified precision.
toString()	Converts a number to a string.
valueOf()	Returns the primitive value of a Number object.

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

Ejercicio 13

https://www.w3schools.com/jsref/jsref_parseint.asp

https://www.w3schools.com/jsref/jsref_toprecision.asp

a) Crea un botón “Cambio de base” para convertir un número dado en una base (pedidos al usuario) a otra base random (máximo base 36). Usa el método parseInt.

b) Crea un botón “Binario a Decimal” para convertir un número binario (pedido al usuario) a número decimal. Usa el método parseInt.

c) Crea un botón “ Significativos” para pedir al usuario un número:
   - Si es superior a 10000000, se mostrará por pantalla ese número con 3 dígitos significativos.
   - Si está entre 0 y 1 (con decimales), se mostrará por pantalla ese número con 2 dígitos significativos
   - Si el número introducido no cumple ninguna condición, se volverá a solicitar que se introduzca.

d) Crea un botón “ Decimales” para pedir al usuario un número con 4 decimales y se muestre por pantalla ese número redondeado a dos decimales. Si el número introducido no cumple la condición, se volverá a solicitar que se introduzca (para ello habrá que analizar que la cadena introducida cuente con cuatro dígitos después del carácter de punto decimal).

El Objeto Boolean.

- Aunque podemos declarar objetos Boolean usando el constructor new, no se recomienda.
- Cualquier objeto, incluido un objeto booleano cuyo valor sea falso, se evalúa como verdadero cuando se pasa a una declaración condicional.

```
const x = new Boolean(false);
if (x) {
  // este código se ejecuta
}
```

```
const x = false;
if (x) {
  // este código no se ejecuta
}
```

- Tampoco debemos usar el constructor new Boolean() para convertir un valor no booleano en uno booleano; utilizamos la función Boolean o un NO doble

```
const good = Boolean(expression);     // use this
const good2 = !!(expression);         // or this
const bad = new Boolean(expression); // don't use this!
```

El Objeto Date

https://www.w3schools.com/jsref/jsref_obj_date.asp

En JavaScript el manejo de fechas es algo complejo y poco intuitivo, lo que ha dado pie, a que existan muchas librerías, que nos facilitan mucho el día a día. Entre ellas:

- *Dates.js* que nos permite una manipulación rápida y precisa con una sintaxis muy intuitiva.
- *Momments.js* que nos aporta mejoras en la sintaxis, además de un complemento genial para gestionar zonas horarias.
- *Timeago* que nos permite convertir fechas a "hace xxx minutos, segundos, días…" de forma dinámica.

Si no queremos usar librerías podemos usar el **Objeto Date**:

```
var ahora = new Date(); //Mon Oct 17 2022 09:19:26 GMT+0200 (hora de verano de Europa central)
```

Podemos pasarle parámetros al constructor para los diferentes formatos de fecha…

Formatos de definición de fechas

Usando milisegundos (desde el 1/1/1970 00:00)

```
new Date(milliseconds);
```

```
var diaDespues = new Date (3600*24*1000); //Fri Jan 02 1970 01:00:00 GMT+0100 (hora estándar de Europa central)
```

Usando cadenas de texto

```
new Date(dateString); //dateString puede estar en formato ISO
var newYear = new Date("January 1, 2022 00:00:00");
var today = new Date(("2022-10-19"); //ISO: YYYY-MM-DD
```

Usando números tiene cierta complejidad oculta a simple vista.

```
new Date(year, month, day, hours, minutes, seconds);
```

```
var newYear = new Date(2022, 0, 1, 0, 0, 0,); Podemos ir quitando parametros del final, pero no se puede llegar a omitir el mes
```

Usando UTC (Hora universal)

```
var newYear = new Date(Date.UTC(2022, 0, 1));
```

Los meses empiezan por cero. Los días de la semana empiezan en domingo siendo este el día 0.

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

Métodos de Date. Getters

getFullYear()	Get the **year** as a four digit number (yyyy)
getMonth()	Get the **month** as a number (0-11)
getDate()	Get the **day** as a number (1-31)
getDay()	Get the **weekday** as a number (0-6)
getHours()	Get the **hour** (0-23)
getMinutes()	Get the **minute** (0-59)
getSeconds()	Get the **second** (0-59)
getMilliseconds()	Get the **millisecond** (0-999)
getTime()	Get the **time** (milliseconds since January 1, 1970)

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

Métodos de Date. Getters. Ejemplo

```
var ahora = new Date();
console.log("El año: " + ahora.getFullYear()); // 4 dígitos
console.log("El mes: " + ahora.getMonth()); // 0 - 11
console.log("El día de la semana: " + ahora.getDay()); // 0 - 6 (D - S)
console.log("El día del mes: " + ahora.getDate()); // 1-31
console.log("==== HORA ====");
console.log("Horas: " + ahora.getHours());
console.log("Minutos: " + ahora.getMinutes());
console.log("Segundos: " + ahora.getSeconds());
console.log("Milisegundos desde 1/1/1970: "+ ahora.getTime());
console.log("milisegundos: " + ahora.getMilliseconds());
console.log("==== OTROS ====");
console.log("Diferencia horaria respecto a UTC: " +
ahora.getTimezoneOffset());
```

Métodos de fechas más usados. Setters

setDate()	Set the day as a number (1-31)
setFullYear()	Set the year (optionally month and day)
setHours()	Set the hour (0-23)
setMilliseconds()	Set the milliseconds (0-999)
setMinutes()	Set the minutes (0-59)
setMonth()	Set the month (0-11)
setSeconds()	Set the seconds (0-59)
setTime()	Set the time (milliseconds since January 1, 1970)

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

Ejercicio 14

1. Realiza un script que muestre la fecha y hora actual. La salida deberá tener el siguiente formato:
   Hoy es martes, 19 de Octubre de 2022 y son las 19:20 horas.
2. Realiza un script que pida un nombre y dos apellidos e indique el tiempo que se tardó en introducir los datos. La salida sería algo así como:
   En introducir Pepito de los Palotes has tardado 25 segundos.
3. Hacer un script con una función en JavaScript que reciba un número de minutos (puede ser positivo o negativo) y devuelva el día de la semana y la hora (en formato de 24 horas) que sería si al domingo a la medianoche se le sumara o restara ese número de minutos.
   Por ejemplo, al llamar a la función con el número 3 debería devolver “Domingo 00:03” y si se llama con el número -3 debería devolver “Sábado 23:57”, al igual que si se llama con 0 debería devolver “Domingo 00:00”.

Cookies en JavaScript

Las cookies permiten almacenar información del usuario en las páginas web.

- Las cookies son datos, almacenados en pequeños archivos de texto, en el ordenador.
- Cuando un servidor web ha enviado una página web a un navegador, la conexión se cierra y el servidor se olvida de todo sobre el usuario.
- Las cookies se inventaron para resolver el problema "cómo recordar información sobre el usuario"
  - Cuando un usuario visita una página web, su nombre puede almacenarse en una cookie.
  - La próxima vez que el usuario visite la página, la cookie "recordará" su nombre.
- Las cookies se guardan en pares de nombre-valor como `username = John Doe`
- Cuando un navegador solicita una página web de un servidor, las cookies que pertenecen a la página se agregan a la solicitud. De esta forma, el servidor obtiene los datos necesarios para "recordar" información sobre los usuarios.

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

Ninguno de los ejemplos siguientes funcionará si tu navegador tiene desactivado el soporte de cookies locales.

Crear una cookie

- JavaScript puede crear, leer y eliminar cookies con la propiedad `document.cookie`.
- Se puede crear una cookie como esta:
```
document.cookie = "username=John Doe";
```
- También puede agregar una fecha de caducidad (en hora UTC). Por defecto, la cookie se elimina cuando se cierra el navegador:
```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC ";
```
- Con un parámetro de ruta, puede decirle al navegador a qué ruta pertenece la cookie. Por defecto, la cookie pertenece a la página actual.
```
document.cookie = "username=John Doe; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
```

Al crear una nueva cookie, no se sobreescriben las cookies anteriores sino que se añade a la cadena de cookies

Leer, modificar y eliminar cookies

- Se puede leer una cookie así:
  ```
  let x = document.cookie;
  ```

`document.cookie` devolverá todas las cookies en una cadena como:
`cookie1=valor; galleta2=valor; cookie3=valor;`

- Puedes modificar una cookie de la misma manera que la creas:
  ```
  document.cookie = "username=John Sith; expires=Thu, 18 Dec 2013 12:00:00 UTC; path=/";
  ```

- Eliminar una cookie es muy sencillo. No hay que especificar un valor de cookie, simplemente establecer el parámetro de expiración en una fecha pasada:
  ```
  document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/";
  ```

Debes definir la ruta de la cookie para asegurarte de eliminar la cookie correcta. Algunos navegadores no te permitirán eliminar una cookie si no especifica la ruta.

EJERCICIO 15 Implementar el siguiente ejemplo de cookies

- En el ejemplo a seguir, crearemos una cookie que almacena el nombre de un visitante.
- La primera vez que un visitante llega a la página web, se le pedirá que introduzca su nombre. A continuación, el nombre se almacena en una cookie.
- La próxima vez que el visitante llegue a la misma página, recibirá un mensaje de bienvenida.
- Para el ejemplo crearemos 3 funciones de JavaScript:
  1. Una función para establecer un valor de cookie
  2. Una función para obtener un valor de cookie.
  3. Una función para comprobar el valor de una cookie.

```
If (navigator.cookieEnabled == true ) {
        alert("El uso de cookies está activado");
}
else {
        alert("El uso de cookies está desactivado");
}
```

Función para establecer valor de cookie

Recuerda que las cookies se envían en las cabeceras HTTP y, por tanto, deben estar correctamente codificadas. Puedes utilizar *encodeURIComponent*(), acostúmbrate a utilizarlo siempre para evitarte sorpresas.

```
function setCookie(cname, cvalue, exdays) {
  const d = new Date();
  d.setTime(d.getTime() + (exdays*24*60*60*1000));
  let expires = "expires="+ d.toUTCString();
  document.cookie = cname + "=" + encodeURIComponent(cvalue)
+ ";" + expires + ";path=/";
}
```

- La función almacena el nombre del visitante en una cookie:
- Los parámetros de la función anterior son el nombre de la cookie (cname), el valor de la cookie (cvalue) y el número de días hasta que caduque la cookie (exdays).
- La función establece una cookie sumando el nombre de la cookie, el valor de la cookie y la cadena de expiración.

Función para obtener una cookie

```
function getCookie(cname) {
  let name = cname + "=";
  let decodedCookie = decodeURIComponent(document.cookie);
  let ca = decodedCookie.split(';');
  for(let i = 0; i <ca.length; i++) {
    let c = ca[i];
    while (c.charAt(0) == ' ') {
      c = c.substring(1);
    }
    if (c.indexOf(name) == 0) {
      return c.substring(name.length, c.length);
    }
  }
  return "";
}
```

- La función devuelve el valor de una cookie especificada con cname

Función para chequear una cookie

```
function checkCookie() {
  let username = getCookie("username");
  if (username != "") {
   alert("Welcome again " + username);
  } else {
    username = prompt("Please enter your name:", "");
    if (username != "" && username != null) {
      setCookie("username", username, 365);
    }
  }
}
```

- Por último, creamos la función que comprueba si se ha establecido una cookie.
- Si la cookie está configurada, mostrará un saludo.
- Si la cookie no está configurada, mostrará un cuadro de aviso, solicitando el nombre del usuario, y almacena la cookie de nombre de usuario durante 365 días, llamando a la función setCookie.


That's all Folks!