Desarrollo Web: Javascript

Dr. Julio Mello
MSc. Marcos Benítez
Marzo/2025

Javascript

- Lenguaje de scripting.
- Javascript es un lenguaje interpretado.
- Javascript NO tiene nada que ver con el lenguaje Java, a pesar de tener nombres similares.
- Permite mejorar aplicaciones Web añadiendo contenido dinámico, personalizado, interactivo.

Javascript

- Principalmente, Javascript se utiliza en el contexto de páginas Web utilizando un navegador.
- Para crear código Javascript solamente necesitamos:
  - Un editor de texto: Para escribir el código JS
  - Un navegador Web: Con los que vemos las páginas Web que contienen el código JS. El navegador Web es el que provee el intérprete Javascript necesario para ejecutar nuestro código.

Javascript

- Javascript es de utilización masiva.
- Soportado por todos los principales navegadores.
- Tareas comunes realizadas con Javascript:
  - Interacción con el usuario.
  - Validación de las acciones de usuario.
  - Obtener y asociar información a un usuario.

Javascript

- Varios “trucos” utilizados en las páginas Web se realizan con Javascript:
  - Cambiar una imagen por otra.
  - Realizar una acción en base a un evento del usuario.
  - Mensajes deslizantes.
  - Menús expandibles.

Javascript

- El avance de los navegadores y su implementación de Javascript permite crear aplicaciones completas a través de él:
  - Google Docs
  - Google Maps
  - Google Calendar
- Javascript se ejecuta en el lado de Cliente, y permite aplicarse sobre las interfaces de usuario.

Javascript

- ¿Dónde ubicamos el código JS?
- Similar a CSS, el código JS asociado a una página Web puede agregarse de dos maneras:
  - En un archivo externo (al de la página Web)
  - En el mismo archivo (de la página Web), utilizando etiquetas HTML particulares.

Javascript

- La etiqueta HTML utilizada para indicar código Javascript es la etiqueta script.

```
<script>

</script>
```

Javascript

- El atributo script tiene un atributo obligatorio, por motivos de validación, que es el atributo **type**.
- El atributo type indica el tipo (lenguaje) de script utilizado.
- Para indicar código Javascript indicamos “**text/javascript**”.

Javascript

- Ejemplo

```
<script type="text/javascript">


</script>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
    <title>Insert title here</title>

    <script type="text/javascript">

    </script>
</head>
<body>
    <div>
        <form action="random.php" method="post">
```

Javascript

- Utilizamos un atributo de la etiqueta script para indicar si el código Javascript es interno (se encuentra dentro de la página Web) o externo (se encuentra definido en un archivo externo, con código Javascript únicamente).
- Este atributo es **src**.
- Indica la fuente (origen) de la ubicación del archivo Javascript externo a ser incluido dentro de una página.

Javascript

- Los archivos Javascript externos llevan la extensión .js
- Los archivos Javascript (externos) tienen como contenido solo código JS, es decir, no tienen etiquetas HTML (la etiqueta script ni ninguna otra).
- El atributo src permite indicar un archivo JS externo dentro de nuestro mismo sitio (utilizando enlaces relativos); o archivos JS externos ubicados en otro dominio.
- Una página Web puede utilizar cualquier combinación y número de código JS interno y externo en forma simultánea.

Javascript

- Al utilizar código Javascript interno, todo el código JS se escribe dentro de las etiquetas script.
- Al utilizar código Javascript externo, todo el código JS se escribe en un archivo externo.

Javascript

- Al igual que cuando utilizamos CSS en archivos externos, la principal ventaja de código JS en archivos externos es la reutilización de este código.
- Código que debe repetirse en múltiples páginas, debe incluirse en cada una de estas páginas mediante la etiqueta **script** y el atributo **src**.
- Si el código fuese interno, debería copiarse/pegarse todo el código JS en todas las páginas que utilicen dicho código.

Javascript

- Similarmente, la utilización de archivos JS externos permite que el navegador cachee estos archivos y solo se transfieran a través de la red una vez, utilizando en las páginas siguientes solamente la versión cacheada en forma local de archivo JS externo – similarmente a lo que ocurre con las imágenes y los archivos CSS externos.
- Esto es algo que no ocurre si el código JS es interno.

Javascript

- Al insertar código JS dentro de una página, mediante las etiquetas **script**, el navegador, al llegar a este bloque, no intenta desplegar el código que encuentra en este bloque sino que lo pasa al intérprete JS para ser ejecutado.
- Las etiquetas script pueden ser ubicadas dentro de las etiquetas:
  - <head></head>
  - <body></head>
- Recordemos que para probar nuestros ejemplos, solo tenemos que abrir la página Web en el navegador de nuestra preferencia.

Ejemplos

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo Básico con JavaScript</title>
</head>
<body>
    <h1>¡Bienvenido a mi página!</h1>
    <button onclick="mostrarMensaje()">Haz clic para ver el mensaje</button>

    <script>
        // Función en JavaScript para mostrar un mensaje
        function mostrarMensaje() {
            alert("¡Hola! Este es un mensaje desde JavaScript.");
        }
    </script>
</body>
</html>
```

Javascript

- El método **alert()** es una función JS que permite abrir una ventana de mensaje.
- Una función en JS (como en otros lenguajes de programación) es un bloque de código que es ejecutado cada vez que es llamado, y sirve para ejecutar una tarea. Recibe opcionalmente parámetros y un retorno.
- La función **alert()** recibe un parámetro que es una cadena de texto (entre comillas “ “), que es el texto que se desplegará en dicha ventana de alerta.

Javascript

- El método **alert()** sirve para indicar al usuario o darle un mensaje.
- El método **alert()** genera una ventana modal.
- Una ventana modal es aquella que no desaparecerá hasta que el usuario la cierre.
- Además, suspende la ejecución (interpretación) de la página hasta que el usuario la cierre.

Ejemplos

```
<body>
    <h1>Formulario de Nombre</h1>

    <form id="formulario">
        <label for="nombre">Ingresa tu nombre:</label><br>
        <input type="text" id="nombre" name="nombre"><br><br>
        <button type="button" onclick="validarFormulario()">Enviar</button>
    </form>

    <p id="mensajeError" style="color: red; display: none;">¡El campo nombre no puede estar vacío!</p>

    <script>
        // Función para validar si el campo nombre está vacío
        function validarFormulario() {
            var nombre = document.getElementById("nombre").value;
            var mensajeError = document.getElementById("mensajeError");

            if (nombre === "") {
                mensajeError.style.display = "block"; // Mostrar mensaje de error
            } else {
                mensajeError.style.display = "none"; // Ocultar mensaje de error
                alert("¡Formulario enviado correctamente!");
            }
        }
    </script>
</body>
```

Javascript

- El ejemplo anterior utiliza el elemento form de html para permitir la carga de datos.
- Así también se utiliza la palabra reservada **function**, el cual permite la escritura de bloque de código reutilizable dentro del contexto de su definición.
- En el ejemplo se puede visualizar el objeto del navegador llamado **document**
- El **document** es un objeto que tiene varias propiedades. Para acceder a una propiedad de dicho objeto, utilizamos el operador punto (.)

Ejemplos

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cambiar Contenido con JavaScript</title>
</head>
<body>
    <h1>Ejemplo de Cambio de Contenido</h1>

    <p id="contenido">Este es el contenido original.</p>

    <button onclick="cambiarContenido()">Haz clic para cambiar el contenido</button>

    <script>
        // Función para cambiar el contenido del párrafo
        function cambiarContenido() {
            document.getElementById("contenido").innerHTML = "¡El contenido ha cambiado!";
        }
    </script>
</body>
</html>

```

Javascript

- En el ejemplo anterior se sigue utilizando el objeto **document** con una pequeña variante, la utilización del **innerHTML** que nos permite el cambio del contenido de ese nodo dinámicamente.
- Ejemplo anterior utiliza el evento del onclick para realizar la operación de cambio de texto del elemento <p>

Ejemplos

```
<body>
    <h1>Hora Actual</h1>

    <button onclick="mostrarHora()">Mostrar hora actual</button>
    <p id="hora"></p>

    <script>
        // Función para mostrar la hora actual
        function mostrarHora() {
            var ahora = new Date();
            var hora = ahora.getHours();
            var minutos = ahora.getMinutes();
            var segundos = ahora.getSeconds();

            // Asegurarse de que los minutos y segundos siempre tengan dos dígitos
            minutos = minutos < 10 ? '0' + minutos : minutos;
            segundos = segundos < 10 ? '0' + segundos : segundos;

            var horaFormateada = hora + ":" + minutos + ":" + segundos;

            document.getElementById("hora").innerHTML = "La hora actual es: " + horaFormateada;
        }
    </script>
</body>
```

Javascript

- El ejemplo muestra la hora actual del sistema operativo
- En el ejemplo se utiliza la palabra reservada **var** para la definición de variables a utilizar
- Así también se utiliza la referencia del objeto **Date** para la obtención de la fecha/hora actual

Variables y Tipo de Datos

- Los datos pueden ser representados de muchas maneras, estos pueden ser de distintos tipos.
- Algunos lenguajes son fuertemente tipados, es decir, cuando usamos algún dato debemos indicar explícitamente el tipo de dato que estamos manejando. Normalmente estos lenguajes tienen reglas estrictas relacionadas a cómo se pueden manejar y combinar los datos de un determinado tipo respecto a otro.

Variables y Tipo de Datos

- Javascript, en cambio, NO es un lenguaje fuertemente tipado.
- En JS, cuando manejamos datos, digamos a través de una variable, no especificamos el tipo de dato que contiene.
- Además, cuando manejamos datos de distinto tipo en JS, por detrás, el intérprete se encargará de combinar estos datos de alguna manera para poder combinar datos de diferente tipo en forma implícita.

Variables y Tipo de Datos

Numéricos

- Los datos de tipo numérico existen en dos maneras:
  - Número enteros, positivos o negativos. Ej: 12
  - Números fraccionarios, positivos o negativos. Ej: 3.1415
- JS, a diferencia otros lenguajes, en el fondo trata todos los números como números fraccionarios (o de coma flotante) por detrás, sin que nosotros lo sepamos.

Variables y Tipo de Datos

Textos (Cadenas)

- Datos que representa uno o más caracteres de texto.
- Para indicar datos que representan texto simplemente definimos el valor (el dato) entre comillas. Ejemplo: “Hola Mundo”
- Podemos utilizar comillas dobles como comillas simples. Sin embargo, debemos recordar no mezclar las comillas (es decir, usar el mismo tipo de comilla al abrir que al cerrar el texto). Ejemplo inválido: “Error’ o ‘Error”

Variables y Tipo de Datos

Booleanos

- Son los datos que solo representan dos opciones: Sí o No
- Los valores booleanos en JS se representan como:
  - true
  - false

Variables y Tipo de Datos

Type	`typeof` return value	Object wrapper
Null	`"object"`	N/A
Undefined	`"undefined"`	N/A
Boolean	`"boolean"`	`Boolean`
Number	`"number"`	`Number`
BigInt	`"bigint"`	`BigInt`
String	`"string"`	`String`
Symbol	`"symbol"`	`Symbol`

Variables

- Las variables son contenedores que permiten almacenar datos en forma temporal, que puede ser modificada.
- Las variables JS se referencia mediante un nombre.
- Se almacenan en memoria.
- Las variables existen, como nuestro código JS, en el contexto de nuestra página.
- Cuando el usuario abre una página con código JS que declara (crea) una variable y le asigna un valor, esta existe en el contexto de esta página en particular.

Variables

- Cuando el usuario cierra el navegador, o cambia la página visualizada en el navegador a otra, el código JS y las variables se destruyen.
- Los nombres de las variables en JS son sensibles a las mayúsculas/minúsculas. Ejemplo: “miVariable” referencia a una variable distinta a “mivariable”.
- Las variables no pueden tener como nombre una “palabra reservada” JS. Una palabra reservada es aquella palabra que tiene un significado especial para el lenguaje, por tanto no pueden haber variables que las utilicen.

Variables

- Además, algunos caracteres no son permitidos de ser usados al nombrar variables. Ejemplo: % o &
- Se pueden utilizar números en el nombre de una variable, siempre y cuando el primer carácter del nombre de una variable NO sea un número.
- Para crear una variable en JS utilizamos la palabra reservada **var**
- Posterior a esta, indicamos el nombre de la variable, y cerramos la sentencia con “punto y coma”.

```
var miPrimeraVariable;
```

Variables

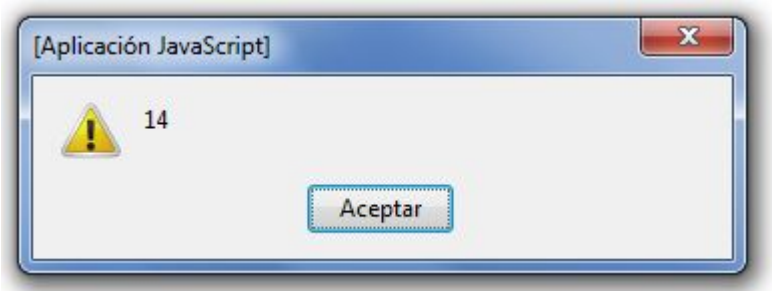
- Una vez declarada una variable, podemos darle un valor para que esta almacene.
- Para asignar un valor a una variable usamos el operador de asignación **“=“**.

```
var miPrimeraVariable;
miPrimeraVariable = 10;

var miSegundaVariable;
miSegundaVariable = "Soy una cadena";

var miTerceraVariable = true;
```

Operadores

- Operadores aritméticos básicos:
  - Multiplicación *
  - División /
  - Suma +
  - Resta –

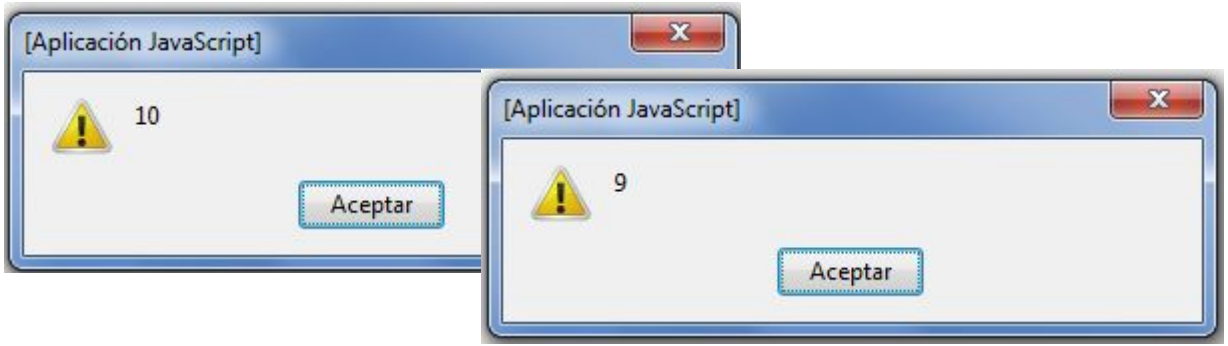
```
var total = 5 + 10 - 3 + 2;
alert(total);
```



Operadores

- Operadores de incremento/decremento:
  - Incremento en 1: ++
  - Decremento en 1: --

```
var miVariable = 10;

alert(miVariable--);

alert(miVariable++);
```



Operadores

- Operadores Abreviados:
  - Incremento: +=
  - Decremento: -=

```
var miVariable = 10;

miVariable += 6; //equivalente a: miVariable = miVariable + 6;
```

```
var miVariable = 10;

miVariable -= 3; //equivalente a: miVariable = miVariable - 3;
```

Operadores

- Operadores Abreviados:
  - Multiplicación: *=
  - División: /=

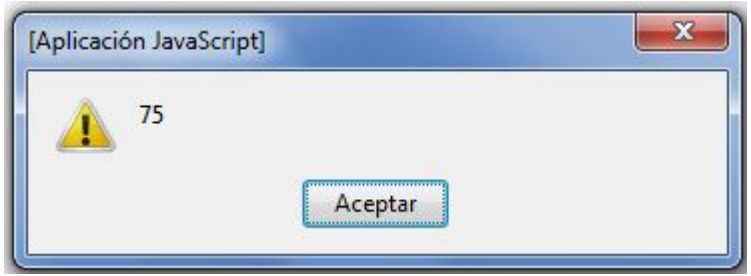
```
var miVariable = 10;

miVariable *= 50; //equivalente a: miVariable = miVariable * 50;
```

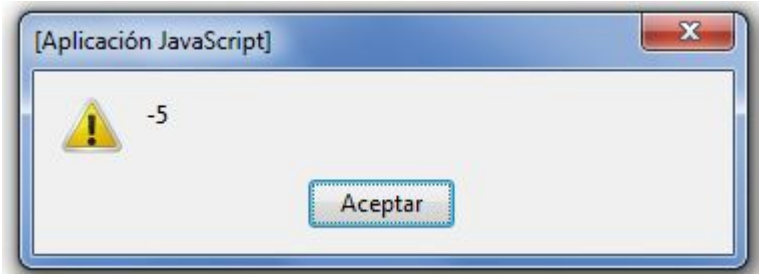
```
var miVariable = 10;

miVariable /= 5; //equivalente a: miVariable = miVariable / 5;
```

Precedencia de Operadores

- Algunos operadores tienen mayor precedencia que otros. Esto significa que se evalúan antes que los demás operadores que están en su mismo “nivel”.

```
var miVariable = 10;

var total = miVariable + 10 - 5 * 50 / miVariable;


alert(total);
```





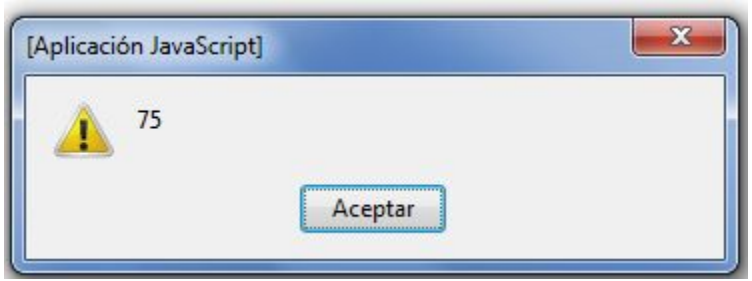
Precedencia de Operadores

- La precedencia de operadores indica que primero se efectúan las operaciones de multiplicación (*) y división (/), de izquierda a derecha.
- Posteriormente, se efectúan las operaciones de suma (+) y resta (-), de izquierda a derecha.
- Podemos darle mayor precedencia a un bloque de operadores utilizando los paréntesis “(“ y “)”.

Precedencia de Operadores

- Un o varios operadores agrupados dentro de paréntesis se ejecutan antes que los demás, igualmente que ocurre con lo que estudiamos en matemáticas.

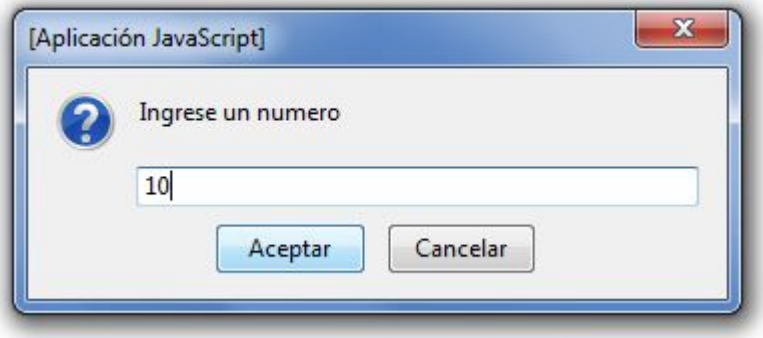
```
var miVariable = 10;

var total = (miVariable + 10 - 5) * 50 / miVariable;


alert(total);
```

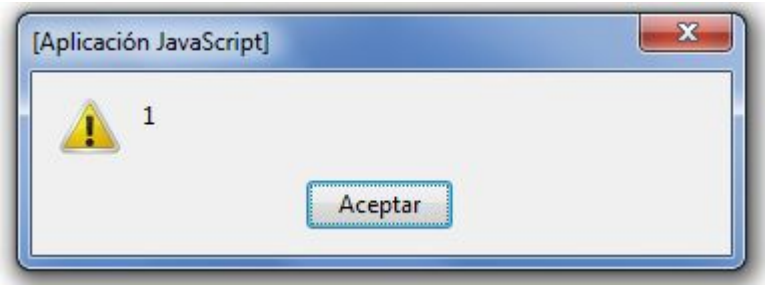


Precedencia de Operadores

- Ejemplo:

```
var entradaUsuario = prompt('Ingrese un numero');

alert((1 + entradaUsuario) / 10);
```





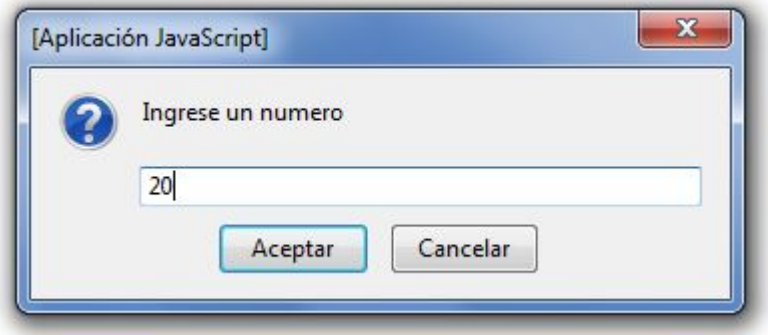
Precedencia de Operadores

- El método **prompt()**, similar al **alert()**, crea una ventana de mensaje para el usuario.
- Sin embargo, a diferencia del alert(), el prompt() provee un espacio para que el usuario ingrese un valor.
- Lo que el usuario ingrese en la ventana, al darle aceptar, se asignará a la variable como retorno de la función.
- Desde ahí, podemos procesar esta variable, con su valor, directamente.

Precedencia de Operadores

- El valor retornado desde el prompt() es un valor de tipo cadena (texto).
- Sin embargo, JS lo convierte implícitamente a un tipo numérico al ver que le aplicamos una operación aritmética al mismo.
- El método prompt() puede recibir un segundo parámetro, que es el valor inicial por defecto de la ventana de mensaje.

```
var entradaUsuario = prompt('Ingrese un numero',20);

alert((1 + entradaUsuario) / 10);
```



Errores

- Muchas veces, cuando nuestro programa escrito JS tiene un error, no vemos ningún mensaje en el navegador.
- Simplemente, ¡no funciona!
- ¿Qué podemos hacer para encontrar estos errores?

Errores

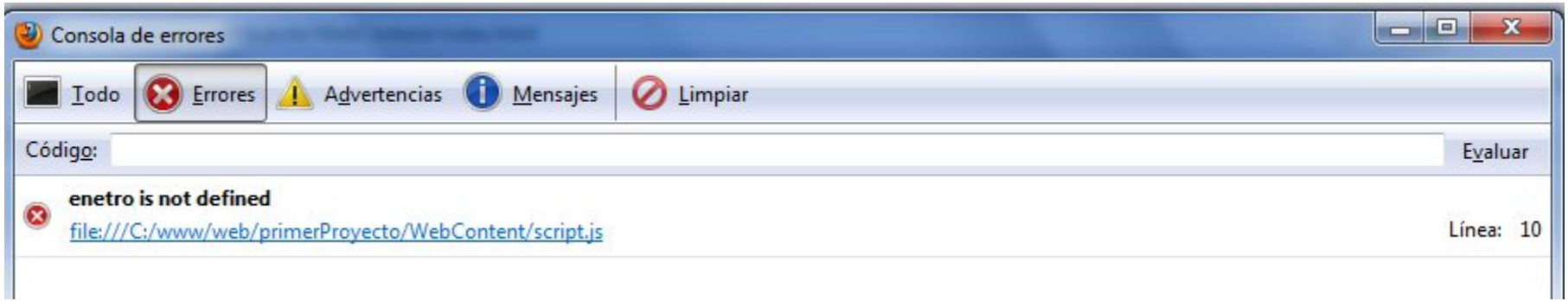
- Antes que nada, debemos conocer los errores más comunes:
  - Usar una variable que no fue declarada.
  - Nombres de variable mal escritos.
  - Olvidar que las variables son sensibles a mayúsculas y minúsculas.
  - No cerrar correctamente paréntesis o llaves.
- Pero, cuando ocurre un error, debemos encontrarlo.
- Utilizando Mozilla Firefox: “Herramientas” – “Consola de Errores”

Errores

- Por ejemplo, ejecutando este código, no obtenemos nada en pantalla:

```
var cadena = '10';

var entero = 5;

alert(cadena * enetro);
```

- Al abrir la “Consola de



Errores

- Nota: Cuando nuestro código tiene más de un error, la ejecución parará en el primer error encontrado, el cual obtendremos en los mensajes de error y ventanas de consola.

console.log()

- Un método de búsqueda de errores es el de la consola (console).
- En nuestro código de prueba podemos insertar una sentencia del tipo: **console.log(variable)**
- Esta sentencia imprime el valor de dicha variable. Si es un vector, nos permite visualizar sus elementos.
- Si es un objeto, nos permitirá ver sus propiedades y métodos.

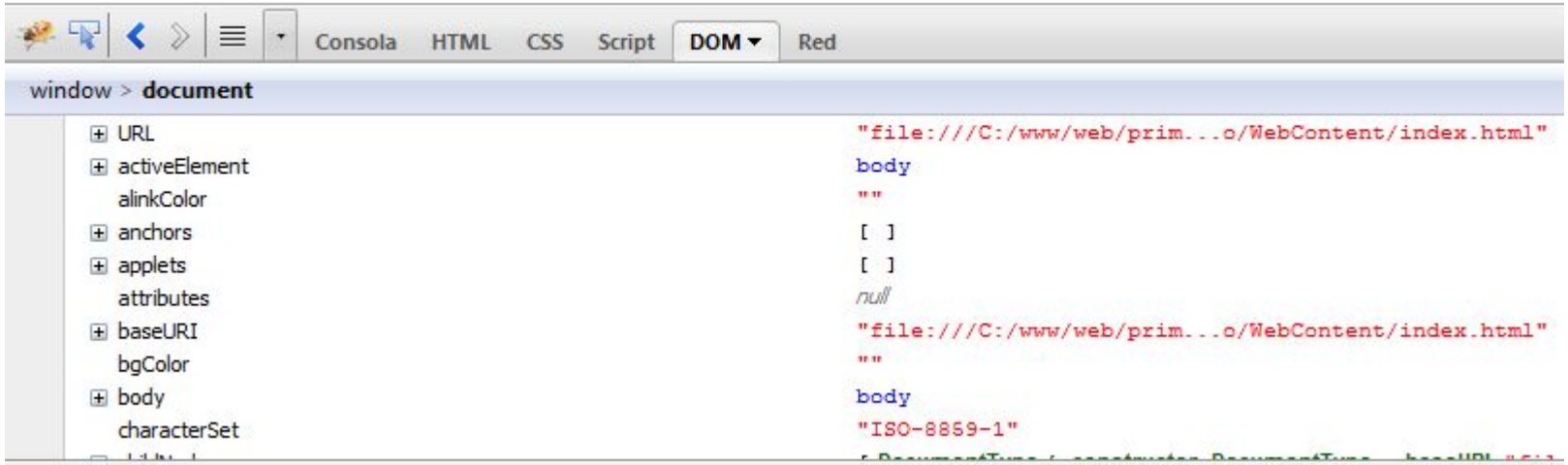
console.log()

- Ejemplo:

```
var cadena = '10';

var entero = 5;

console.log(cadena * entero);

console.log(document);
```

console.log()

- En Mozilla Firefox :



console.log()

- El método **console.log()** es de extremada utilidad cuando queremos encontrar errores, principalmente cuando tenemos que inspeccionar objetos.
- ***Solo debemos utilizar este método en código de prueba o en fase de testeo, nunca usarlo en código de producción.***

Ejercicios 1

- Desarrolla una página web que permita a los usuarios convertir un monto en Guaraníes (Gs) a su equivalente en Dólares (USD) usando un valor de cambio ingresado por el usuario.

Ejercicios 2

- Conversor de Temperatura: Celsius a Fahrenheit
  - Desarrolla una página web interactiva que permita a los usuarios convertir una temperatura proporcionada en grados Celsius a su equivalente en grados Fahrenheit.

Ejercicios 3

- Idem al ejercicio 2, pero de Fahrenheit a Celsius.

Ejercicios 4

- Desarrolla una página web interactiva que permita a los usuarios ingresar su nombre, apellido y edad. La página debe generar un mensaje de bienvenida personalizado que incluya la concatenación de los valores ingresados, mostrando un saludo al usuario de manera amigable.

Ejercicios 5

- Hacer una página Web que solicite al usuario su edad actual. Debe imprimir como resultado, en un único mensaje, el cálculo de la edad del usuario dentro de 5, 10 y 100 años.

Ejercicio 6

- Hacer una página que, ingresando un monto de dinero que represente el costo de un producto, imprima el valor del mismo + IVA (10%).

Ejercicio 7

- Hacer una página que, ingresando un monto de dinero que represente el costo de un producto incluyendo IVA, imprima el valor del mismo del producto (sin IVA) y el valor del IVA (10 %) en un mensaje.

Ejercicio 8

- Hacer una página que, ingresando un monto de dinero que represente el gasto total de la cuenta en un restaurante y posteriormente ingresando un número de porcentaje, imprima el valor de la propina.

Ejercicio 9

- Hacer una página que, ingresando un año pueda determinar si corresponde a un año bisiesto, alertar al usuario del resultado

Ejercicio 10

- Hacer una página Web que solicite al usuario dos fechas con formato (dd/mm/yyyy). Debe imprimir como resultado, en un único mensaje, el cálculo de la diferencia entre las fechas ingresadas.
  - Validar el formato antes de procesar los datos indicando al usuario que debe cumplir con dicho formato
  - Validar que la fecha no den resultados negativos