Introducción a JavaScript

Luis Valencia Cabrera (lvalencia@us.es)

Research Group on Natural Computing
Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

10-02-2020, Bases de Datos

Introducción

- JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.
- Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.
- A pesar de su nombre, JavaScript no guarda ninguna relación directa con el lenguaje de programación Java. Legalmente, JavaScript es una marca registrada de la empresa Sun Microsystems.

Introducción

- JavaScript es una de las múltiples maneras que han surgido para extender las capacidades del lenguaje HTML.
- JavaScript no es un lenguaje de programación propiamente dicho como C, Scheme, Python, etc. Es un lenguaje *script* u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto y hojas de cálculo.
- JavaScript es un lenguaje interpretado que se embebe en una página web HTML. Un lenguaje interpretado significa que a las instrucciones las analiza y procesa el navegador en el momento que deben ser ejecutadas.

Glosario

- Script: cada uno de los programas, aplicaciones o trozos de código creados con el lenguaje de programación JavaScript. A veces se traduce al español directamente como *guión*, aunque script es una palabra comúnmente aceptada.
- Sentencia: cada una de las instrucciones que forman un script.
- Palabras reservadas: son las palabras (en inglés) que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente. Por ejemplo, `if`, `else`, `while`, `...`

Introducción

Un primer ejemplo

```
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<script type="text/javascript">
  document.write('Hola Mundo');
</script>
</body>
</html>
```

Introducción

- El programa JavaScript debe ir en la marca script, *inicializada la propiedad type con la cadena* `text/javascript`:

```
<script type="text/javascript">
    /*  Aquí vendría el código JavaScript  */
</script>
```

- Para imprimir caracteres sobre la página debemos llamar al comando `write` del objeto `document`, aunque veremos otros usos más habituales actuando sobre elementos específicos.
- La información a imprimirse debe ir entre comillas y encerrada entre paréntesis. Todo lo que indicamos entre comillas aparecerá tal cual dentro de la página HTML.
- Es decir, si pedimos al navegador que ejecute esta página mostrará el texto 'Hola Mundo'. Cada vez que escribimos una instrucción finalizamos con el carácter punto y coma.
- JavaScript es sensible a mayúsculas y minúsculas.

Introducción

- Si queremos que cada dato quede en una fila distinta de la página debemos insertar la marca
 (salto de linea en HTML), es decir debemos disponer:
  `document.write('<br>')`.
- Esto es, la ejecución del script *creará* el código HTML.
- Esta idea es fundamental: En lugar de crear nosotros el código HTML, escribimos sentencias JavaScript que escriban el código HTML. De este modo, la página creada cambia de una vez para otra según se ejecuten las instrucciones JavaScript.

Introducción

```
<html>
<head>
</head>
<body>

<script type="text/javascript">

  document.write('Esto va arriba');
  document.write('<br>');
  document.write('y esto va debajo');

</script>

</body>
</html>
```

Formas de dar un mensaje de texto

- `document.write("texto")` Al ejecutarse se escribe el texto.
- `window.alert("texto")` Hay un ejemplo de `alert` a continuación.
- Podemos usar la propiedad `innerHTML` del objeto devuelto por el método `getElementById(Id)` (un nodo HTML).
- Podemos imprimir a la consola de depuración mediante `console.log("texto")`.

Mensaje de texto

- El siguiente ejemplo muestra el carácter dinámico de JavaScript.
- Es un primer ejemplo de evento: *onclick*.
- Al pulsar sobre el botón se ejecuta la acción, en este caso *document.write*
- Nótese que toda la página se reescribe.

Introducción

```
<!DOCTYPE html>
<html>

<body>
    <h1>Otra página sencilla</h1>
    <p>Un párrafo</p>
    <button type="button"
            onclick="document.write('Sorpresa')">
        Haz la prueba
    </button>
</body>

</html>
```

Mensaje de texto

- El siguiente es un ejemplo de `window.alert`.
- Nótese que en este caso no se reescribe la página, sino que el texto sale en una ventana emergente.
- Puesto que no está asociada a ningún evento, la ventana auxiliar con el mensaje sale al cargar la página.

Introducción

```
<!DOCTYPE html>
<html>

<body>

    <h1>Una página sencilla</h1>
    <p>Un ejemplo de window.alert</p>

    <script>
        window.alert('Hola, mundo');
    </script>

</body>

</html>
```

Mensaje de texto

- También podemos asociar el `window.alert` a un evento.
- Por ejemplo, el evento `onmouseenter` tiene efecto cuando el ratón entra dentro del objeto en la pantalla.

Introducción

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
</head>

<body>
    <img onmouseenter="window.alert('Soy Mickey')"
        src="MickeyMouse.png" alt="Ratón Mickey"
        width="50">
    <p>Pasa el ratón por la imagen</p>
</body>

</html>
```

Mensaje de texto

- La forma usual de definir la acción asociada a un evento es mediante una función que definimos nosotros.
- Veamos un ejemplo sencillo definiendo una función y asociándole un `alert`.
- Como veremos, no es necesario anteponer *window* a `alert`.

Ejemplo

Un primer ejemplo

```
<html>
<head></head>
<body>
    <p>Pulsa para obtener un mensaje de alerta.</p>
    <button onclick="myFunction()">Pulsa</button>
    <script>
        function myFunction() {
            alert("Hola");
        }
    </script>
</body>
</html>
```

Mensaje de texto

- Otra forma más general de introducir un texto es localizar un objeto mediante su identificador dentro del documento.
- El objeto donde queremos hacer la modificación debe estar identificado.
- En este caso, la modificación la hacemos con *inner HTML*.

Ejemplo

```
<!DOCTYPE html>
<html>
<body>
<h1>Otra página más</h1>
<p id="demo">Hola</p>
<button onclick="myFunction()">Pulsa</button>
<script>
    function myFunction() {
        document.getElementById("demo")
                .innerHTML = 'Sorpresa';
    }
</script>
</body>
</html>
```

Comentarios

Se emplean para proporcionar explicaciones del código que vamos desarrollando. Hay varios tipos de comentarios aceptados en el lenguaje JavaScript:

- Si ocupan una línea, lo ponemos detrás de las barras dobles `// Comentario de una línea`.
- Si son más largos usamos los símbolos `/* */`.

```
/* Comentario
que ocupa
varias líneas */
```

Variables

- Las variables deben comenzar por letra o subrayado (_).
- No puede tener el nombre de una palabra clave del lenguaje.
- Hasta 2015, una variable se definía con la palabra `var`:
  ```
  var dia;
  ```
- Se pueden declarar varias variables en una misma línea:
  ```
  var dia, mes, calle;
  ```
- Una variable se puede definir e inicializar en un paso:
  ```
  var edad=20;
  ```
- O en dos pasos:
  ```
  var edad;
  edad=20;
  ```

Introducción

```
<html>
  <head>
  </head>
  <body>
  <script type="text/javascript">
    var nombre='Juan';
    var edad=63;
    document.write(nombre);
    document.write('<br>');
    document.write(edad);
  </script>
  </body>
  </html>
```

Variables y constantes en ES6/2015

En 2015 hay una evolución significativa de la especificación de JavaScript, conocida como ES6 o ES2015 (ES: ECMAScript, estandarizada por ECMA Internacional). Esta evolución afecta al uso de variables:

- Desaconseja el uso de `var` en la mayoría de situaciones.
- Promueve el uso de `let` para variables.
- Define además `const` para constantes.
- Puede ver una descripción detallada en este enlace:

`https://desarrolloweb.com/articulos/conociendo-variables-ecmascript.html`

Introducir datos

- Para la entrada de datos por teclado tenemos la función `prompt`. Cada vez que necesitamos introducir un dato con esta función, aparece una ventana donde cargamos el valor.
- La sintaxis de la función prompt es:

  ```
  var_objetivo=prompt(mensaje,valor_defecto);
  ```
- Como vemos, `prompt` tiene dos parámetros: uno es el mensaje a mostrar en la ventana y el otro el valor inicial que aparece en la misma.

Introducción

```
<html>
  <head>
  </head>
  <body>
  <script type="text/javascript">
    var nombre;
    nombre=prompt('Dime tu nombre:','');
    document.write('Hola ');
    document.write(nombre);
  </script>
  </body>
  </html>
```

Estructuras secuenciales

- Cuando en un programa sólo participan operaciones, entradas y salidas se la denomina estructura secuencial.
- El problema anterior, donde se introduce el nombre de una persona se trata de una estructura secuencial.
- Por ejemplo, podemos pedir dos números por teclado e imprimir su suma.
- Sólo debemos tener en cuenta que si queremos que el operador $+$ sume los contenidos de los valores numéricos ingresados por teclado, debemos llamar a la función `parseInt` y pasar como parámetro las variables `valor1` y `valor2` sucesivamente.

Introducción

```
<body>
<script type="text/javascript">
  var valor1;
  var valor2;
  valor1=prompt('Dame el primer número:','');
  valor2=prompt('Dame el segundo número','');
  var suma=parseInt(valor1)+parseInt(valor2);
  document.write('La suma es ');
  document.write(suma);
</script>
</body>
```

Introducción

- Cuando se presenta una elección, necesitamos una estructura condicional.
- En una estructura condicional simple, si se verifica la condición se realiza una acción y si no, no ocurre nada.
- Usaremos la instrucción `if` en el lenguaje JavaScript. La condición debe ir entre paréntesis. Si la condición se verifica verdadera se ejecuta todas las instrucciones que se encuentran encerradas entre las llaves de apertura y cerrado seguidas al if.
- Podemos utilizar alguno de los siguientes operadores relacionales: $>$, $>=$, $<$, $<=$, $!=$ (distinto), $==$ (igual), $===$ (igual en valor y tipo).
- En la condición del `if` deben intervenir una variable un operador relacional y otra variable o valor fijo.
- Ojo al uso del operador $+$ para cadenas de caracteres.

Introducción

```
<script type="text/javascript">
  var nombre, nota1, nota2, pro;
  nombre = prompt('Nombre:', 'Escribe aquí');
  nota1 = prompt('Primera nota:', 'Escribe aquí');
  nota2 = prompt('Segunda nota:', 'Escribe aquí');
  nota1 = parseInt(nota1);
  nota2 = parseInt(nota2);
  pro = (nota1 + nota2) / 2;
  if (pro >= 5) {
    document.write(nombre + ' aprueba con ' + pro);
  }
</script>
```

Condicional compuesta

- En general, queremos que si se verifica la condición se realice una acción y que se realice otra si la condición no se satisface.
- La estructura condicional compuesta se construye como:

```
if (cond) {
  //Instruccion(es) si se cumple cond
} else {
  //Instruccion(es) en otro caso
}
```

Introducción

```
<script type="text/javascript">
  var num1, num2;
  num1=prompt('Dame un número:','');
  num2=prompt('Dame otro número:','');
  num1=parseInt(num1);
  num2=parseInt(num2);
  if (num1>num2) {
    document.write('El mayor es '+num1);
  } else {
    document.write('El mayor es '+num2);
  }
</script>
```

Operadores lógicos

- Conjunción (y): &&
- Disyunción (o): ||

Introducción

```
<script type="text/javascript">
  var num1,num2;
  num1=prompt('Dame un número:','');
  num2=prompt('Dame otro número:','');
  num1=parseInt(num1);
  num2=parseInt(num2);
  if ((num1>num2) && (num1> 2 || num2 > 3)) {
    document.write('Sí se cumple');
  } else {
    document.write('No se cumple');
  }
</script>
```

Estructuras de control: while

```
while (cond) { /* Instrucciones */ }
```

- En primer lugar se verifica la condición; si esta resulta verdadera, se ejecutan las operaciones indicadas entre llaves.
- En caso que la condición sea falsa, la ejecución continúa con la instrucción siguiente al bloque encerrado entre llaves.
- El bloque se repite MIENTRAS la condición sea verdadera.
- Si la condición siempre se cumple, estamos en presencia de un ciclo repetitivo infinito.

Introducción

```
<script type="text/javascript">
  var x;
  x=1;
  while (x<=100)
  {
    document.write(x);
    document.write('<br>');
    x=x+1;
  }
</script>
```

Introducción

```
<script  type="text/javascript">
  var x=1;
  var suma=0;
  var valor;
  while (x<=5) {
    valor=prompt(`Dame un valor ${x}:`,'Suma 5 valores');
    valor=parseInt(valor);
    suma=suma+valor;
    x=x+1;
  }
  document.write(`La suma es ${suma}<br>`);
</script>
```

Estructuras de control: for

- Sintaxis de las estructuras `for`:

```
for (inic; cond; inc) {
  /* Conjunto de instrucciones */
}
```

- Como vemos, esta estructura repetitiva tiene tres argumentos: variable de inicialización, condición y operación de incremento o decremento.

Ejemplo

- Para escribir los números del 1 al 10, iniciamos la variable `f` como 1.
- A continuación escribimos la condición que debe verificarse `f<=10`. Como la condición se verifica como verdadera se ejecuta el bloque del `for` (en este caso mostramos el contenido de la variable `f` y un espacio en blanco).
- Tras ejecutar el bloque, pasa al tercer argumento del `for` (en este caso con el operador ++ se incrementa en uno el contenido de la variable `f`.

Introducción

```
<script type="text/javascript">
  var f;
  for(f=1;f<=10;f++) {
    document.write(f+" ");
  }
</script>
```

Funciones

- Tienen la siguiente estructura:

```
function nombre_func(arg_1, ..., arg_n) {
  /* Código de
  la función */
}
```

- También pueden ser definidas con expresiones de función:

```
var nombre_func = function (arg_1, ..., arg_n) {
  /* Código de
  la función */
};
```

Puede ver una explicación sobre definición y uso de funciones en:

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

Funciones

```
<script type="text/javascript">
  function mostrarComprendidos(x1,x2) {
    for (var inicio=x1; inicio<=x2; inicio++) {
      document.write(inicio+' ');
    }
  }
  var valor1, valor2;
  valor1 = prompt('Dame el valor inferior:','');
  valor1 = parseInt(valor1);
  valor2 = prompt('Dame el valor superior:','');
  valor2 = parseInt(valor2);
  mostrarComprendidos(valor1,valor2);
</script>
```

Otros ejemplos

- Javascript puede cambiar el contenido HTML.
- Usaremos el método `getElementById` para acceder a los nodos elemento de html por su identificador.
- Usaremos su propiedad `innerHTML` para modificar su contenido.
- Así, el siguiente ejemplo usa el método para encontrar el elemento con identificador `id = cambia` y cambia su contenido actual por `Esto es una sorpresa`.

Introducción

```
<body>
<h1>¿Qué puede hacer Javascript?</h1>
<p id="cambia">Puede cambiar el texto HTML</p>
<button type="button"
onclick="document.getElementById('cambia').innerHTML='¡SORPRESA!'">
    Haz la prueba
</button>
</body>
```

Otros ejemplos

- Javascript puede cambiar una imagen por otra
- En el siguiente ejemplo se combina una función con un condicional `if - else`.

Introducción

```
<body>
<h1>Este es el escudo de mi equipo</h1>
<img id="miescudo" onclick="changeImage()"
    src="escudo_1.jpg" width="100">
<p>Haz click sobre él.</p>
<script>
    function changeImage() {
        var image = document.getElementById('miescudo');
        if (image.src.match("escudo_1.jpg")) {
            image.src = "escudo_2.png";
        } else {
            image.src = "escudo_1.jpg";
        }
    }
</script>
```

Introducción

```
<h1>Este es el escudo de mi equipo</h1>
<img id="miescudo" onclick="changeImage()"
    src="escudo_1.jpg" width="100">
<p>Haz click sobre él.</p>
<script>
    function changeImage() {
        var image = document.getElementById('miescudo');
        if (image.width == "100") {
            image.width = "300";
        } else {
            image.width = "100";
        }
    }
</script>
```

Podemos hacer muchas más cosas ...

```
<p id="id1" title="Hola"
  style="background-color:red">Ejemplo</p>
<button type="button" onclick="funcionA()">
  Quiero cambiar
</button>
<script>
  function funcionA() {
    if (document.getElementById("id1")
                .innerHTML.match('Ejemplo')) {
      document.getElementById("id1").style =
        'background-color:yellow';
      document.getElementById("id1").innerHTML =
        'Ha cambiado';
    }
  }
</script>
```

Otro ejemplo

```
<h1>Elige un botón</h1>
<button type="button" onclick="funcionA()">
    Botón A
</button>
<button type="button" onclick="funcionB()">
    Botón B
</button>
<script>
    function funcionA() {
        window.alert('Has pulsado el botón A');
    }
    function funcionB() {
        window.alert('Has pulsado el botón B');
    }
</script>
```

. . . mucho más

- Podemos usar la palabra restringida `this` para hacer referencia al objeto sobre el que se define la función.
- El evento `onmouseover` tiene efecto cuando pasamos el ratón sobre el objeto.
- El evento `onmouseout` tiene efecto cuando dejamos de pasar el ratón sobre el objeto.

Un último ejemplo

```
<head>
    <style type="text/css">
        #miescudo {
            position: absolute;
            top: 100px;
            left: 20px;
        }
    </style>
</head>
```

```
<body>
    <h1>Este es el escudo de mi equipo</h1>
    <img id="miescudo" onmouseover="mueve(this)"
        src="escudo_1.jpg" width="200">
    <p>Si quieres cambiarlo, haz click sobre él.</p>
    <script>
        function mueve(x) {
            if (x.style.left == "20px") {
                x.style.left = "250px";
            } else {
                x.style.left = "20px";
            }
        }
    </script>
</body>
```

Para saber más

- http://www.javascriptya.com.ar/
- http://librosweb.es/libro/javascript/
- http://www.w3schools.com/js/default.asp
- ...

Estos son únicamente algunos enlaces útiles en relación con el material de la presentación, pero hay mucho más material en la sección de ***JavaScript*** *del sitio web de la asignatura.*