# Scripting en el cliente: Javascript

Tecnologías Web

# La clase `String`

- Se pueden manejar cadenas mediante objetos de la clase `String`
- En JavaScript las cadenas se pueden representar entre comillas simples (') o dobles (") indistintamente
  - ¡Cuidado! es necesario que se cierren y abran con el mismo carácter
  - De esta manera se puede incluir una cadena dentro de otra, por ejemplo, al declarar un manejador de evento
- Las cadenas se manejan de manera similar a las de Java
  - Se concatenan con +
  - Se puede acceder a la longitud mediante la propiedad *length*

# Métodos de la clase `String`

`charAt(`*`posición`*`);`

- Devuelve el carácter de la cadena que se encuentra en *posición*.

`slice(`*`inicio, fin`*`) / substring(`*`inicio, fin`*`)`

Devuelve la subcadena que comienza en la posición *inicio* y termina en *fin*-1

- **Diferencia:** en `slice(...)` los índices pueden ser negativos, en ese caso se cuenta desde el final de la cadena

`substr(`*`inicio, longitud`*`)`

- Devuelve la subcadena que comienza en *inicio* y tiene *longitud* caracteres

`indexOf(`*`subcadena`*`, [`*`inicio`*`])`

- Busca la primera aparición de *subcadena* desde en la cadena desde la posición *inicio;* si no se especifica *inicio*, será desde el principio de la cadena.

`lastIndexOf(`*`subcadena`*`, [`*`inicio`*`])`

- Igual que el anterior, pero busca la última aparición

`toUpperCase() / toLowerCase()`

- transforman la cadena a mayúsculas y a minúsculas, respectivamente

# Objeto `navigator`

- No es un objeto integrado en el lenguaje como `Date`, `Math` y `String`
- Tampoco forma parte de la jerarquía del objeto `Window`.
- Pertenece al DOM 0
- Nos proporciona información relacionada con el navegador.
  - Nos permiten adaptar nuestro JavaScript al navegador y a la versión particular del mismo.

Browser sniffers

# El objeto window

- ¿Cómo usar el objeto window?
  - Como cualquier objeto
  - Como objeto implícito
  - Mediante una variable
  - Como una ventana particular

Tipos de referencias al objeto window

```
// Refencias como cualquier objeto normal
window.nombre_propiedad;
window.nombre_metodo();
// Referencias como objeto implicito
nombre_propiedad;
nombre_metodo();
// Referencia mediante variable
var ventana = window;
ventana.nombre_propiedad;
ventana.nombre_metodo();
// Referencia a la ventana actual (útil para el caso de marcos y pop-ups
window.self.nombre_propiedad;
window.self.nombre_metodo();
self.nombre_propiedad;
self.nombre_metodo();
```

# Usando el objeto window

- Visualizar mensajes en la barra de estado

Personalizando vínculos: Mensaje por defecto y personalizados

```
<html>
  <head><title>Personalizando vínculos</title></head>
  <script language="JavaScript" type="text/javascript"><!--
    // Cambia el mensaje que aparece en la barra de estado
    function cambiaMensaje(mensaje){
      window.status = mensaje;
      return true;
    }
    // Establece el mensaje por defecto del navegador
    window.defaultStatus="Este es el mensaje por defecto";
  --></script>
  </head>
  <body>
    <p>Mueve el puntero del ratón sobre los vínculos</p>
     <a href="v1" onmouseover="return cambiaMensaje('Mensaje1');"
                  onmouseout="return cambiaMensaje('');">Vinculo 1</a>
     <a href="v2" onmouseover="return cambiaMensaje('Mensaje2');"
                  onmouseout="return cambiaMensaje('');">Vinculo 2</a>
  </body>
</html>
```

**Indica que no hay que seguir procesando el evento**

**Habilita el mensaje por defecto**

# Usando el objeto `window`

- Ejecución de funciones en otro momento
  - Además de los métodos habituales de ejecución de funciones, podemos diferir la ejecución de una función durante un tiempo o podemos hacer que una función se ejecute cada cierto intervalo de tiempo.
- Diferir la ejecución de una función

```
window.setTimeout(expresion, milisegs, arg1, arg2, ...);
```

  - `expresion` → Nombre de la función a ejecutar.
  - `milisegs` → Número de milisegundos que se debe esperar.
  - `arg1, 1rg2` → Argumentos opcionales que se le pasan a la función.
  - Devuelve un valor para identificar la operación por si se desea cancelar.

```
window.clearTimeout(ident);
```

  - `ident` → Variable que alamacena el valor devuleto por el método `setTimeout()`

# Usando el objeto `window`

- Ejecutando una función cada cierto tiempo

  `window.setInterval(exp, milisegs, arg1, arg2, ...);`

  - `exp` → Nombre de la función que se desea ejecutar
  - `milisegs` → Número de milisegundos del intervalo
  - `arg1, arg2` → argumentos opcionales para la función.
  - Devuelve un valor que identifica la operación `setInterval()`

  `window.clearInterval(ident)`

  - `ident` → Identificador devuelto por la operación `setInterval()`

- Podemos utilizar estas funciones para crear cuentas atrás, animaciones, etc.

# Usando el objeto window

Desplazar un mensaje por la barra de estado

```
<script language="JavaScript" type="text/javascript"><!--
  var msg = "Creación de Sitios Web (CSW)"; // Mensaje a mostrar
  var msg_delay = 500; // Demora de desplazamiento
  var max_desp = 2; // Número máximo de desplazamientos
  var pos_actual = -1; // Posición a partir de la cual se mostrará la cadena
  var total_desp = 0; // Almacena cuantas veces se ha mostrado el mensaje
  var id_intervalo = setInterval("scroll_msg()", msg_delay);
  function scroll_msg(){
    // Aumentar la posición actual dentro de la cadena de mensaje
    pos_actual++;
    // Seguimos ?
    if ( pos_actual < msg.length ){
      status = msg.substring(pos_actual);
    }else{
      pos_actual = 0;
      total_desp++;

      // Hemos alcanzado el máximo de desplazamientos ?
      if ( total_desp == max_desp ){
        clearInterval(id_intervalo);
        status="";
        return;
      }
    }
  }
--></script>
```

# Usando el objeto `window`

- Podemos controlar la navegación de las páginas desde un guión JavaScript usando el objeto `location`

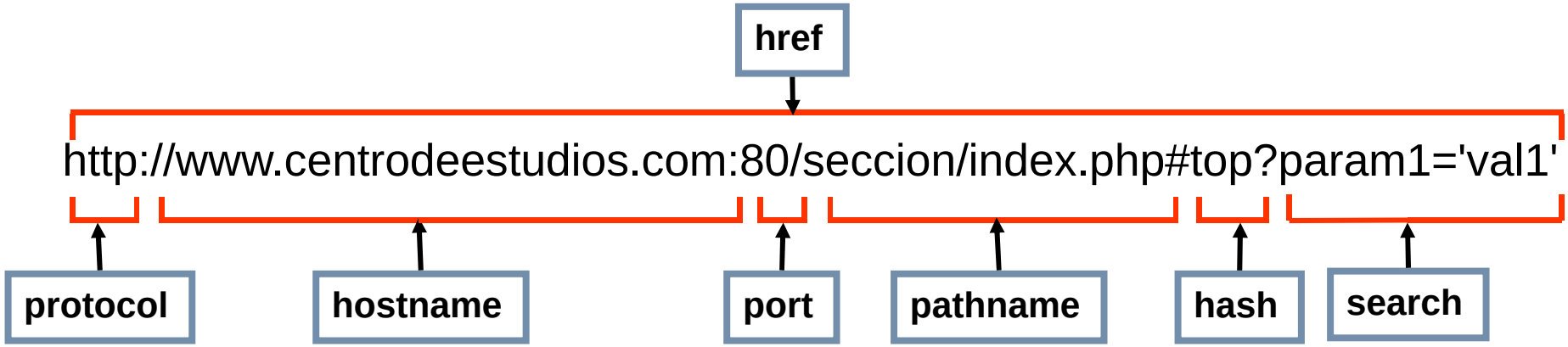
  `window.location=URL`

  - URL → Dirección de la página a cargar

  `location.`*`nombrePropiedad`*`=`*`valorPropiedad`*

  `location[`*`nombrePropiedad`*`]=`*`valorPropiedad`*

  - Propiedades: `protocol, hostname, port, host, pathname, hash, search, href`



# Usando el objeto `window`

- Métodos del objeto location

  `reload(`*`desdeCache`*`);`

  - *`desdeCache`* → (Opcionalmente) Si es `false` (valor por defecto) la página se recargará desde la cache del navegador o `true` y obligamos a que se recargue desde el servidor.

# ¿Validación en cliente o en servidor?

- Ventajas de la validación en el lado del cliente:
  - Eficiencia
    - No es necesario realizar una petición HTTP al servidor
  - Comodidad para el cliente
- Desventaja:
  - No garantiza la seguridad
    - El navegador del cliente podría no entender JavaScript
    - El código HTML y JavaScript es accesible al cliente
      - Un cliente malintencionado podría modificarlo para evitar la validación
- Solución:
  - Realizar la validación en **ambos** lados
    - Comodidad para el cliente
    - Se evitan peticiones innecesarias → menos carga para el servidor
    - Seguridad garantizada
    - Más trabajo para el programador

# Trabajando con formularios

- Acceso a los formularios de una página

```
window.document.forms[indice]
window.document.forms["nombre"]
window.document.forms.nombre
window.document.nombre
```

- Propiedades de un objeto `form`
  - `action` → Devuelve o asigna la URL a la que será enviado el formulario
  - `elements` → Devuelve un array con todos los campos del formulario
  - `encoding` → Devuelve o asigna el tipo MIME del formulario
  - `length` → Devuelve el número de campos definidos en el formulario
  - `method` → Devuelve o angina le método por el que será enviados los datos al servidor
  - `name` → Devuelve o asigna el nombre del formulario
  - `target` → Devuelve o angina el nombre del marco en el que se deberá mostrar la respuesta del servidor

# Trabajando con formularios (cont.)

- Acceso a los elementos de un formulario

```
formulario.elements[indice]
formulario.elements["nombre"]
formulario.nombre
```

- Propiedades habituales de los campos
  - `form` → Devuelve una referencia al objeto Form que contiene el campo
  - `name` → Devuelve el nombre del campo
  - `value` → Devuelve o asigna el valor del campo
  - `type` → Devuelve el tipo del campo (`<input type="tipo" ...>`)
    - Submit → <input type="submit"
    - Reset → <input type="reset"
    - Button → <input type="button"
    - Text → <input type="text"
    - Password → <input type="password"
    - FileUpload → <input type="file"
    - Hidden → <input type="hidden"
    - Checkbox → <input type="checkbox"
    - Radio → <input type="radio"

# Trabajando con formularios (cont.)

- Podemos tener un manejador de eventos para varios formularios, para distinguir cada formulario podemos usar el siguiente código

Paso de formularios a una función

```
<form action="servidor/pagina.php"
      method="post"
      name="prueba"
      onsubmit="manejador(this)">

<input type="text"
       name="campo"
       onchange="manejador(this.form)">
```

- Evento `onSubmit` de un objeto `Form`

  `<form onsubmit="return submit_hldr()" ...>`

  - Nos permite controlar si se enviará realmente o no un formulario
  - Si se devuelve `true`, se realiza el envío y si es `false` se rechaza.

- Evento `onChange` de un objeto `Input`
  - Nos permite controlar el valor del campo una vez ha sido elegido por el usuario.

# Ejemplo de validación de formulario

Página de validación

```
<html>
  <head><title>Ejemplo de validación</title>
    <script type="text/javascript" language="JavaScript" src="form.js"></script>
  </head>
  <body>
    <form action="..." onsubmit="return validar(this)">
      Nombre: <input type="text" name="nombre"/><br/>
      Edad: <input type="text" name="edad"/><br/>
      <input type="submit" value="enviar"/>
    </form>
  </body>
</html>
```

form.js

```
function validar(formulario) {
  if (formulario.nombre.value=="") {
    window.alert("Debe introducir su nombre");
    return false;
  }
  edad=parseInt(formulario.edad.value);
  if (isNaN(edad) || edad<0 || edad>100) {
    window.alert("La edad debe ser un número entre 0 y 100");
    return false;
  }
  return true;
}
```

# Referencias

- http://www.howtocreate.co.uk/
  - Contiene muchísima información sobre DHTML y JavaScript
- Browser sniffers
  - http://jsbrwsniff.sourceforge.net/ http://www.mozilla.org/docs/web-developer/sniffer/browser_type_oo.html
  - http://www.mozilla.org/docs/web-developer/sniffer/browser_type.html
  - http://www.dynamicdrive.com/dynamicindex9/browsersniffer.htm

# Críticas, dudas y sugerencias...

Federico Peinado
www.federicopeinado.es