DESARROLLO WEB EN ENTORNO CLIENTE

CAPÍTULO 4:

Programación con funciones, arrays y objetos definidos por el usuario

Juan Manuel Vara Mesa
Marcos López Sanz
David Granada
Emanuel Irrazábal
Jesús Javier Jiménez Hernández
Jenifer Verde Marín

Ra-Ma®

Funciones predefinidas del lenguaje

- JavaScript cuenta con una serie de funciones integradas en el lenguaje.
- Dichas funciones se pueden utilizar sin conocer todas las instrucciones que ejecuta.
- Simplemente se debe conocer el nombre de la función y el resultado que se obtiene al utilizarla.

Funciones predefinidas del lenguaje

- Las siguientes son algunas de las principales funciones predefinidas de JavaScript:

Funciones Predefinidas	
`escape()`	`Number()`
`eval()`	`String()`
`isFinite()`	`parseInt()`
`isNaN()`	`parseFloat()`

Funciones predefinidas del lenguaje

- `escape()`: recibe como argumento una cadena de caracteres y devuelve esa misma cadena sustituida con su codificación en ASCII.

```
<script type="text/javascript">
  var input = prompt("Introduce una cadena");
  var inputCodificado = escape(input);
  alert("Cadena codificada: " + inputCodificado);
</script>
```

Actividad 4-1:

Cree un script que utilice el método document.write() para mostrar por pantalla la codificación de todas las vocales con tilde (á,é,í,ó,ú). Para realizar dicha codificación debe de usara la función escape().

Funciones predefinidas del lenguaje

- `eval()`: convierte una cadena que pasamos como argumento en código JavaScript ejecutable.

```
<script type="text/javascript">
  var input = prompt("Introduce una operación numérica");
  var resultado = eval(input);
  alert ("El resultado de la operación es: " + resultado);
</script>
```

Funciones predefinidas del lenguaje

- `isFinite()`: verifica si el número que pasamos como argumento es o no un número finito.

```
if(isFinite(argumento)) {
  //instrucciones si el argumento es un número finito
}else{
  //instrucciones si el argumento no es un número finito
}
```

Funciones predefinidas del lenguaje

- `isNan()`: comprueba si el valor que pasamos como argumento es un de tipo numérico.

```
<script type="text/javascript">
  var input = prompt("Introduce un valor numérico: ");
  if (isNaN(input)){
    alert("El dato ingresado no es numérico.");
  }else{
    alert("El dato ingresado es numérico.");
  }
</script>
```

Funciones predefinidas del lenguaje

- `String()`: convierte el objeto pasado como argumento en una cadena que represente el valor de dicho objeto.

```
<script type="text/javascript">
  var fecha = new Date()
  var fechaString = String(fecha)
  alert("La fecha actual es: "+fechaString);
</script>
```

Funciones predefinidas del lenguaje

- `Number()`: convierte el objeto pasado como argumento en un número que represente el valor de dicho objeto.

Funciones predefinidas del lenguaje

- `parseInt()`: convierte la cadena que pasamos como argumento en un valor numérico de tipo entero.

```
<script type="text/javascript">
  var input = prompt("Introduce un valor: ");
  var inputParsed = parseInt(input);
  alert("parseInt("+input+"): "+inputParsed);
</script>
```

Funciones predefinidas del lenguaje

- `parseFloat()`: convierte la cadena que pasamos como argumento en un valor numérico de tipo flotante. 

```
<script type="text/javascript">
  var input = prompt("Introduce un valor: ");
  var inputParsed = parseFloat(input);
  alert("parseFloat("+input+"): " + inputParsed);
</script>
```

Funciones del usuario

- Es posible crear funciones personalizadas diferentes a las funciones predefinidas por el lenguaje.
- Con estas funciones se pueden realizar las tareas que queramos.
- Una tarea se realiza mediante un grupo de instrucciones relacionadas a las cuales debemos dar un nombre.

Funciones del usuario

- Definición de funciones:
  - El mejor lugar para definir las funciones es dentro de las etiquetas HTML ~~`<body>` y `</body>`~~. <head> y </head>
  - El motivo es que el navegador carga siempre todo lo que se encuentra entre estas etiquetas.
  - La definición de una función consta de cinco partes:
    - La palabra clave `function`.
    - El nombre de la función.
    - Los argumentos utilizados.
    - El grupo de instrucciones.
    - La palabra clave `return`.

Funciones del usuario

- Definición de funciones – Sintaxis:

```
function nombre_función ([argumentos]){
   grupo_de_instrucciones;
   [return valor;]
}
```

Funciones del usuario

- Definición de funciones – `Function`:
  - Es la palabra clave que se debe utilizar antes de definir cualquier función.

Funciones del usuario

- Definición de funciones – Nombre:
  - El nombre de la función se sitúa al inicio de la definición y antes del paréntesis que contiene los posibles argumentos.
    - Deben usarse sólo letras, números o el carácter de subrayado.
    - Debe ser único en el código JavaScript de la página web.
    - No pueden empezar por un número.
    - No puede ser una de las palabras clave del lenguaje.
    - No puede ser una de las palabras reservadas del lenguaje.

Funciones del usuario

- Definición de funciones – Argumento:
  - Los argumentos se definen dentro del paréntesis situado después del nombre de la función.
  - No todas las funciones requieren argumentos, con lo cual el paréntesis se deja vacío.

Funciones del usuario

- Definición de funciones – Grupo de instrucciones:
  - El grupo de instrucciones es el bloque de código JavaScript que se ejecuta cuando invocamos a la función desde otra parte de la aplicación.
  - Las llaves ({ }) delimitan el inicio y el fin de las instrucciones.

Funciones del usuario

- Definición de funciones – `Return`:
  - La palabra clave `return` es opcional en la definición de una función.
  - Indica al navegador que devuelva un valor a la sentencia que haya invocado a la función.

Funciones del usuario

- Ejemplo – Función que calcula el importe de un producto después de haberle aplicado el IVA:

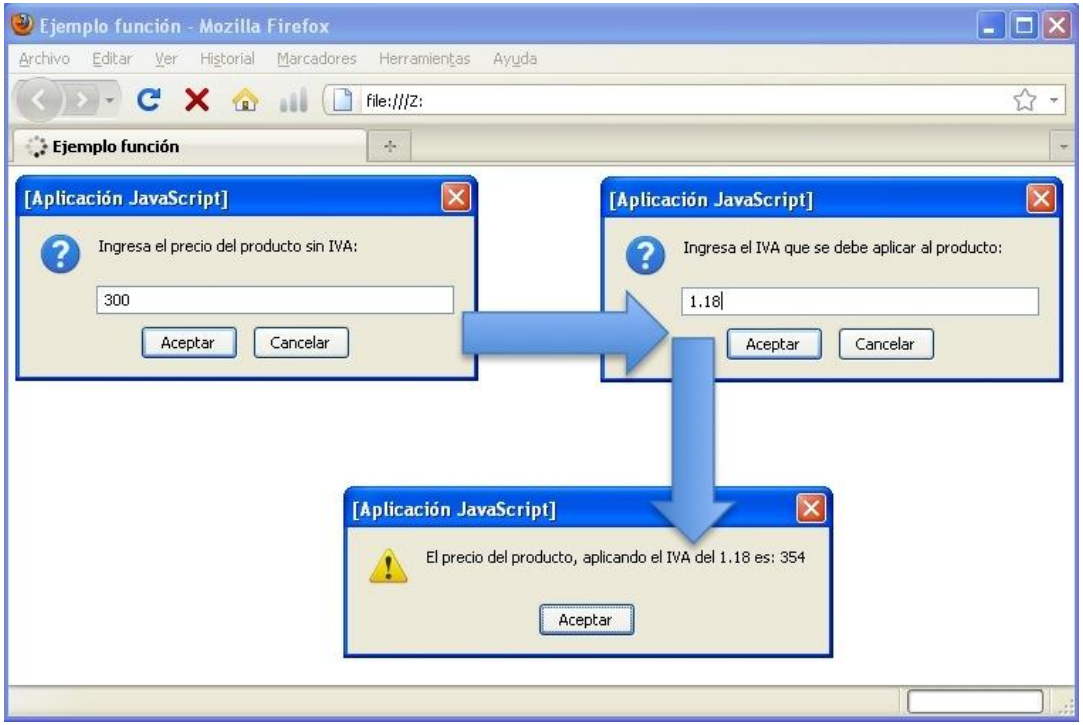
```
function aplicar_IVA(valorProducto, IVA){
  var productoConIVA = valorProducto * IVA;
  alert("El precio del producto, apicando el
IVA
  del  " + IVA + " es: " + productoConIVA);
}
```

Funciones del usuario

- Invocación de funciones:
  - Una vez definida la función es necesaria llamarla para que el navegador ejecute el grupo de instrucciones.
  - Se invoca usando su nombre seguido del paréntesis.
  - Si tiene argumentos, se deben especificar en el mismo orden en el que se han definido en la función.

Funciones del usuario

- Ejemplo: `aplicar_IVA(300, 1.18).`



Funciones del usuario

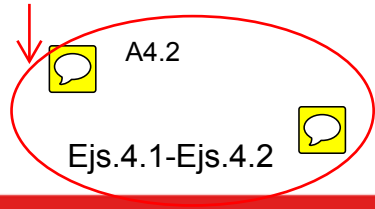
- Invocar una función desde JavaScript:

```
<html><head>
    <title>Invocar función desde JavaScript</title>
    <script type="text/javascript">
      function mi_funcion([args]){
        //instrucciones
      }
    </script></head>
  <body>
    <script type="text/javascript">
      mi_funcion([args]);
    </script>
  </body></html>
```

Funciones del usuario

- Invocar una función desde HTML:

```
<html>
  <head>
    <title>Invocar función desde JavaScript</title>
    <script type="text/javascript">
      function mi_funcion([args]){
        //instrucciones
      }
    </script>
  </head>
  <body onload="mi_funcion([args])"></body>
</html>
```



Arrays

- La mayor parte de las aplicaciones web gestionan un número elevado de datos.
- Por ejemplos si se quisiera definir el nombre de 180 productos alimenticios:

```
var producto1 = "Pan";
var producto2 = "Agua";
var producto3 = "Lentejas";
var producto4 = "Naranjas";
var producto5 = "Cereales";
...
var producto180 = "Salsa agridulce";
```

Arrays

- Si posteriormente se quisiera mostrar el nombre de estos productos:

```
document.write(producto1);
document.write(producto2);
document.write(producto3);
document.write(producto4);
document.write(producto5);
...
document.write(producto180);
```

Arrays

- El anterior ejemplo es correcto, pero sería una tarea compleja, repetitiva y propensa a errores.
- Para gestionar este tipo de escenarios se pueden utilizar los arrays.
- Un array es un conjunto ordenado de valores relacionados.
- Cada uno de estos valores se denomina elemento y cada elemento tiene un índice que indica su posición numérica en el array.

Arrays

- Declaración de arrays:
  - Al igual que ocurre con las variables, es necesario declarar un array antes de poder usarlo.
  - La declaración de un array consta de seis partes:
    - La palabra clave `var`.
    - El nombre del array.
    - El operador de asignación.
    - La palabra clave para la creación de objetos `new`.
    - El constructor `Array`.
    - El paréntesis final.

Arrays

- Declaración de arrays – Sintaxis:
  - *(1)*:

```
var nombre_del_array = new Array();
```

  - *(2)*:

```
var nombre_del_array = new
Array(número_de_elementos);
```

Arrays

- Inicialización de arrays:
  - Una vez declarado el array se puede comenzar el proceso de inicializar o popular el arraya con los elementos que contendrá.
  - La sintaxis es la siguiente:

```
nombre_del_array[índice] = valor_del_elemento;
```

Arrays

- Es posible declarar e inicializar simultáneamente mediante la escritura de los elementos dentro del paréntesis del constructor.

```
var productos_alimenticios = new
Array('Pan', 'Agua', 'Lentejas');
```

Arrays

- Uso de arrays mediante bucles:
  - Si se mezclan las características de los bucles unto a las de los arrays se pueden apreciar las ventajas que proporciona este objeto.
  - Por ejemplo:

```
var codigos_productos = new Array();
for (var i=0; i<10;i++){
 codigos_productos[i] = "Codigo_producto_" + i;
}
```

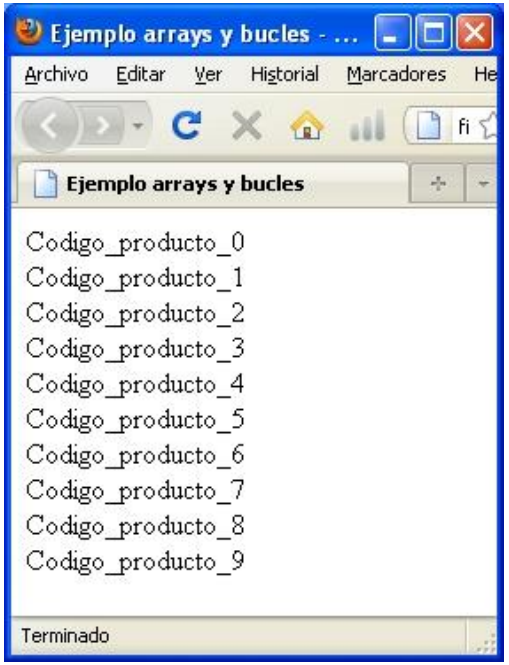
Arrays

- Uso de arrays mediante bucles:
  - La inicialización de un array con un bucle funciona mejor en dos casos:
    - Cuando los valores de los elementos se pueden generar usando una expresión que cambia en cada iteración del bucle.
    - Cuando se necesita asignar el mismo valor a todos los elementos del array.

Arrays

- Mediante el uso de un bucle se pueden escribir instrucciones mucho más limpias y eficientes:

```
for (var i=0; i<10; i++){
 document.write
 (codigos_productos[i] + "<br>");
}
```



Arrays

- Propiedades de los arrays:
  - El objeto array tiene dos propiedades:
    1. lenght:

```
for (var i=0; i<codigos_productos.length; i++){
  document.write(codigos_productos[i] + "<br>");
}
```

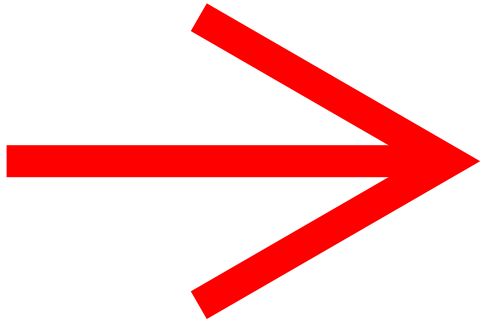
    2. prototype:

```
Array.prototype.nueva_propiedad = valor;
Array.prototype.nuevo_metodo = nombre_de_la_funcion;
```

Arrays

- Métodos de los arrays:

Métodos	
`push()`	`shift()`
`concat()`	`pop()`
`join()`	`slice()`
`reverse()`	`sort()`
`unshift()`	`splice()`



Arrays

- Métodos de los arrays – `push()`:
  - Añade nuevos elementos al array y devuelve la nueva longitud del array.

```
<script type="text/javascript">
  var pizzas = new Array("Carbonara", "Quattro_Stagioni",
    "Diavola");
  var nuevo_numero_de_pizzas = pizzas.push("Margherita",
    "Boscaiola");
  document.write("Número de pizzas disponibles: " +
    nuevo_numero_de_pizzas + "<br />");
  document.write(pizzas);
</script>
```

Arrays

- Métodos de los arrays – `concat()`:
  - Selecciona un array y lo concatena con otros elementos en un nuevo array.

```
<script type="text/javascript">
  var equipos_a = new Array("Real Madrid", "Barcelona",
    "Valencia");
  var equipos_b = new Array("Hércules", "Elche",
    "Valladolid");
  var equipos_copa_del_rey = equipos_a.concat(equipos_b);
  document.write("Equipos que juegan la copa: " +
    equipos_copa_del_rey);
</script>
```

Arrays

- Métodos de los arrays – `join()`:
  - Concatena los elementos de un array en una sola cadena separada por un carácter opcional.

```
<script type="text/javascript">
  var pizzas = new Array("Carbonara", "Quattro_Stagioni",
    "Diavola");
  document.write(pizzas.join(" - "));
</script>
```

Arrays

- Métodos de los arrays – `reverse()`:
  - Invierte el orden de los elementos de un array.

```
<script type="text/javascript">
  var numeros = new Array(1,2,3,4,5,6,7,8,9,10);
  numeros.reverse();
  document.write(numeros);
</script>
```

Arrays

- Métodos de los arrays – `unshift()`:
  - Añade nuevos elementos al inicio de un array y devuelve el número de elementos del nuevo array modificado.

```
<script type="text/javascript">
  var sedes_JJOO = new Array("Atenas", "Sydney",
    "Atlanta");
  var numero_sedes = sedes_JJOO.unshift("Pekín");
  document.write("Últimas " + numero_sedes + " sedes
    olímpicas: " + sedes_JJOO);
</script>
```

Arrays

- Métodos de los arrays – `shift()`:
  - Elimina el primer elemento de un array.

```
<script type="text/javascript">
  var pizzas = new Array("Carbonara", "Quattro_Stagioni",
    "Diavola");
  var pizza_removida = pizzas.shift();
  document.write("Pizza eliminada de la lista: " +
    pizza_removida + "<br />");
  document.write("Nueva lista de pizzas: " + pizzas);
</script>
```

Arrays

- Métodos de los arrays – `pop()`:
  - Elimina el último elemento de un array.

```
<script type="text/javascript">
  var premios = new Array("Coche", "1000 Euros", "Manual de
    JavaScript");
  var tercer_premio = premios.pop();
  document.write("El tercer premio es: " + tercer_premio +
    "<br />");
  document.write("Quedan los siguientes premios: " +
    premios);
</script>
```

Arrays

- Métodos de los arrays – `slice()`:
  - Devuelve un nuevo array con un subconjunto de los elementos del array que ha usado el método.

```
<script type="text/javascript">
  var numeros = new Array(1,2,3,4,5,6,7,8,9,10);
  var primeros_cinco = numeros.slice(0,5);
  var ultimos_cuatro = numeros.slice(-4);
  document.write(primeros_cinco + "<br>");
  document.write(ultimos_cuatro);
</script>
```

Arrays

- Métodos de los arrays – `sort()`:
  - Ordena alfabéticamente los elementos de un array. Podemos definir una nueva función para ordenarlos con otro criterio.

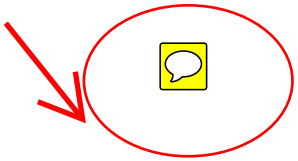
```
<script type="text/javascript">
  var apellidos = new Array("Pérez", "Guijarro", "Arias",
    "González");
  apellidos.sort();
  document.write(apellidos);
</script>
```

Arrays

- Métodos de los arrays – `splice()`:
  - Elimina, sustituye o añade elementos del array dependiendo de los argumentos del método.

```
<script type="text/javascript">
  var coches = new Array("Ferrari", "BMW", "Fiat");
  coches.splice(2,0,"Seat");
  document.write(coches);
</script>
```

Ver texto:
ArraysMultidimensionales.docx



Objetos definidos por el usuario

- JavaScript proporciona una serie objetos predefinidos, sin embargo es posible crear nuevos objetos definidos por el usuario.
- Cada uno de estos objetos puede tener sus propios métodos y propiedades.
- La creación de nuevos objetos resulta útil en el desarrollo de aplicaciones avanzadas.

Diferencias de OOP entre Java y JavaScript:

La más importante es que esta basada en prototipos

–Usa prototipos en vez de clases para la herencia

–Las funciones cumplen el doble propósito de actuar como constructores y métodos

•Si usamos un new delante de una llamada a una función, entonces creamos un nuevo objeto que podemos manipular a través de la palabra clave this

•A diferencia de muchos lenguajes orientados a objetos, no hay distinción entre una definición de función y de método

–Cuando una función es llamada como método de un objeto, la palabra clave this permite manipular los atributos del objeto cuyo método ha sido invocado

OOP en JavaScript

•Los objetos en JavaScript toman la forma internamente de un array asociativo, mapean nombres de propiedades a valores
–Son contenedores para valores y funciones
–Tiene varios tipos de objetos ya definidos:
•Array, Boolean, Date, Function, Math, Number, Object, RegExp y String
•También existen objetos soportados por el entorno de ejecución como window, form, links en el DOM.
•JavaScript es un lenguaje orientado a objetos basado en prototipos.
–La herencia tiene lugar entre objetos en vez de clases
–Se pueden añadir nuevas propiedades y métodos a un objeto a través de la palabra clave prototype
•No es obligatorio borrar objetos ya que tiene garbage collector
•JavaScript tiene los siguientes tipos de datos primitivos:
–Números como 42 or 3.14159
–Valores lógicos true or false
–Strings, como "Howdy!"
–null, palabra clave denotando un valor nulo
–undefined, indica que el valor no ha sido definido

Objetos definidos por el usuario

- Declaración e inicialización de los objetos:
  - Un objeto es una entidad que posee unas propiedades que lo caracterizan y unos métodos que actúan sobre estas propiedades.
  - Su sintaxis es la siguiente:

```
function mi_objeto (valor_1, valor_2, valor_x){
  this.propiedad_1 = valor_1;
  this.propiedad_2 = valor_2;
  this.propiedad_x = valor_x;
}
```

Objetos definidos por el usuario

- Ejemplo:

Primero declaramos la función que será el nuevo tipo de objeto

```
<script type="text/javascript">
  function Coche(marca_in, modelo_in, anyo_in){
    this.marca = marca_in;
    this.modelo = modelo_in;
    this.anyo = anyo_in;
  }
</script>
```

Objetos definidos por el usuario

- Una vez declarado el nuevo tipo de objeto se pueden crear instancias mediante la palabra clave `new`:

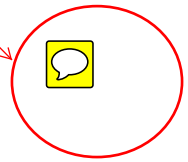
```
<script type="text/javascript">
  var coches = new Array(4);
  coches[0] = new Coche("Ferrari", "Scaglietti", "2010");
  coches[1] = new Coche("BMW", "Z4", "2010");
  coches[2] = new Coche("Seat", "Toledo", "1999");
  coches[3] = new Coche("Fiat", "500", "1995");
  for(i=0; i<coches.length; i++){
    document.write("Marca: "  + coches[i].marca +
    " - Modelo: " + coches[i].modelo + " - A&ntilde;o
    de fabricaci&oacute;n: " + coches[i].anyo + "<br>");
  }
</script>
```

Objetos definidos por el usuario

- Es posible añadir otras propiedades a cada instancia del objeto, por ejemplo:

```
function Coche (marca_in, modelo_in, anyo_in){
  this.marca = marca_in;
  this.modelo = modelo_in;
  this.anyo = anyo_in;
}
var mi_coche = new coche("Pegeout", "206cc", "2003");
mi_coche.color = "azul";
```

Realizar actividad:



Ejemplo ver MetodosObjetos.html

```
<html>
<head>
 <meta http-equiv="content-type"
   content="text/html;charset=utf-8">
 <title>Funciones de objetos</title>
 <script type="text/javascript">
  function imprimeDatos(){
      document.write("Marca: " + this.marca);
      document.write("<br>Modelo: " + this.modelo);
      document.write("<br>A&ntilde;o: " + this.anyo);
  }

  function Coche(nombre_in, modelo_in, anyo_in){
      this.marca = nombre_in;
      this.modelo = modelo_in;
    this.anyo = anyo_in;
      this.imprimeDatos = imprimeDatos;
  }
 </script>
</head>
<body>
 <script type="text/javascript">
  var mi_coche = new Coche("Seat","Toledo", "1999");
  mi_coche.imprimeDatos();
 </script>
</body>
</html>
```

Ver documento ArraysAsociativos.docx

Este documento contiene la explicación de arrays asociativos en JavaScript, ejemplo resuelto y ejercicio a realizar