INTRODUCCIÓN A JAVASCRIPT

Aplicaciones Web (OpenCourseWare, 2023)

Jesús Arias Fisteus

uc3m | Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática

VARIABLES

```
var n = 1;  // variable global
var sumDoubled = function(m) {
    var mDoubled = 2 * m;  // variable local
    n += mDoubled;
}

sumDoubled(2);
n;                  // 5
```

SENTENCIAS DE CONTROL

- `if`, `switch`
- `for`, `while`, `do while`
- `return`, `break`, `continue`

CONDICIONALES Y OPERADORES DE COMPARACIÓN

```
if (command === "create") {
    // ...
}

if (command !== "remove") {
    // ...
}

if (num >= 7) {
    // ...
}
```

OPERADORES LÓGICOS

```
if (command === "create" || command === "remove") {
    // ...
}

if (command === "remove" && num > 7) {
    // ...
}

var a = false;
while (!a) {
    // ...
}
```

BUCLES

```
var count = 0;
while (count < 10) {
    console.log(count);
    count++;
}

for(var count = 0; count < 10; count++) {
    console.log(count);
}

var person = {fname: "Lisa", lname: "Simpson", age: 8};
var x;
for (x in person) {
    console.log(person[x]);
}
```

TIPOS DE DATOS

- Tipos simples:
  - Números
  - Cadenas de texto
  - Booleanos
- Objetos:
  - Arrays
  - Funciones
  - Objetos

TIPOS DE DATOS

```
var count = 0;
var word = "University";
var active = true;
var square = function(n) {
    return n * n;
};
var color = {
    r: 30,
    g: 255,
    b: 128,
    luminosity: function() {
        return 0.21 * this.r + 0.72 * this.g + 0.07 * this.b;
    }
};
```

OBJETOS

```
var emptyObject = {};

var student = {
    "first-name": "Lisa",
    "last-name": "Simpson"
};
student["first-name"]     // "Lisa"
student["middle-name"]    // undefined
student["FIRST-NAME"]     // undefined
```

OBJETOS

```
var book = {
    title: "El ingenioso hidalgo Don Quijote de la Mancha",
    year: 1605,
    author: {
        name: "Miguel",
        surname: "de Cervantes Saavedra",
        birthDate: 1547
    }
};
book.year;                  // 1605
book.author.name            // "Miguel"
```

OBJETOS

Se pueden añadir nuevas propiedades a un objeto existente

```
book.author['birthPlace'] = 'Alcalá de Henares'
book.originalLanguage = 'Spanish';
```

OBJETOS

Los objetos se manipulan por referencia (como en Java)

```
var x = book;
x.genre = 'novel';
var genre = book.genre;
   // genre es 'novel' porque x y book
   // son referencias al mismo objeto

var a = {}, b = {}, c = {};
   // a, b y c hacen referencia cada una
   // a un objeto vacío diferente

a = b = c = {};
   // a, b y c hacen referencia, todas,
   // al mismo objeto vacío
```

FUNCIONES

```
var add = function(a, b) {
    return a + b;
};

var sum = add(3, 4);    // sum is 7
```

MÉTODOS

```
var color = {
    r: 30,
    g: 255,
    b: 128,
    luminosity: function() {
        return 0.21 * this.r + 0.72 * this.g + 0.07 * this.b;
    }
};
color.luminosity();  // 198.86
```

CONSTRUCTORES

```
// Declarar el constructor
var Color = function(r, g, b) {
    // Atributos
    this.r = r;
    this.g = g;
    this.b = b;

    // Métodos
    this.luminosity = function() {
        return Math.round(0.21 * this.r + 0.72 * this.g
                          + 0.07 * this.b);
    }

    this.toGrayScale = function() {
        var a = this.luminosity();
        return new Color(a, a, a);
    }
}
```

CONSTRUCTORES

```
// Crear un nuevo objeto con su constructor
var red = new Color(255, 0, 0);
red.luminosity();              // 54
var gray = red.toGrayScale(); // Color (r: 54, g: 54, b: 54)
```

CLASES (A PARTIR DE ECMASCRIPT 6)

```
class Color {
    constructor(r, g, b) {
        this.r = r;
        this.g = g;
        this.b = b;
    }

    get luminosity() {
        return Math.round(0.21 * this.r + 0.72 * this.g
                          + 0.07 * this.b);
    }

    toGrayScale() {
        var a = this.luminosity;
        return new Color(a, a, a);
    }
}
```

CONSTRUCTORES

```
// Crear un nuevo objeto con su constructor
var red = new Color(255, 0, 0);
red.luminosity;                  // 54
var gray = red.toGrayScale(); // Color (r: 54, g: 54, b: 54)
```

EXCEPCIONES

```
var add = function (a, b) {
    if (typeof a !== 'number' || typeof b !== 'number') {
        throw {
            name: 'TypeError',
            message: 'add necesita números'
        };
    }
    return a + b;
}

try {
    add("seven");
} catch (e) {
    console.log('¡Vaya!');
    console.log(e.name + ': ' + e.message);
}
```

ARRAYS

```
var empty = [];
var letters = ['A', 'B', 'C', 'D', 'E', 'F'];

empty[1]          // undefined
letters[1]        // 'B'

empty.length      // 0
numbers.length    // 6
```

ARRAYS

```
letters.length = 3;
// letters is ['A', 'B', 'C']

letters[letters.length] = 'Y';
// letters is ['A', 'B', 'C', 'Y']

letters.push('Z');
// letters is ['A', 'B', 'C', 'Y', 'Z']

delete letters[2];
// letters is ['A', 'B', undefined, 'Y', 'Z']

letters.splice(2, 1);
// letters is ['A', 'B', 'Y', 'Z']
```

CLIENT-SIDE JAVASCRIPT

CLIENT-SIDE JAVASCRIPT

- El término *client-side* JavaScript hace referencia al entorno de ejecución de código JavaScript proporcionado por los navegadores web.
- Este entorno lo conforman las APIs de JavaScript definidas por HTML5 y otros estándares relacionados, e implementadas por los navegadores.

CLIENT-SIDE JAVASCRIPT

- *Client-side* JavaScript hace interactivo el documento HTML mediante, principalmente:
  - Manejadores de eventos: código a ejecutar cuando se cargue o cierre el documento, el usuario interaccione con sus elementos, etc.
  - Modificación dinámica del documento: mediante el API DOM, principalmente, el programa JavaScript puede modificar el documento, y el navegador muestra inmediatamente los cambios.

INCLUSIÓN DE JAVASCRIPT EN HTML

```
<!-- directamente con el elemento script
     (en la cabecera o en el cuerpo del documento) -->
<script type="text/javascript">
var d = new Date();
document.write(d.toLocaleString());
</script>

<!-- desde un recurso externo -->
<script src="scripts/util.js" type="text/javascript"></script>

<!-- desde un manejador de eventos de HTML -->
<input type="button" value="Change" onclick="changeName()">
<p onmouseover="showHelp('p1')">...</p>
```

Cargar el código JavaScript al fondo del documento permite presentar más rápidamente la página (se cargan antes las imágenes, hojas CSS, etc.)

JQUERY

HOLA MUNDO CON JQUERY

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Mi primera página con jQuery</title>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script type="text/javascript">
      $(function() {
          console.log("ready!");
      });
    </script>
  </head>
  <body>
    <p>¡Hola Mundo!</p>
  </body>
</html>
```

CÓDIGO DE INICIALIZACIÓN

```
// Se ejecuta el código de inicialización cuando el árbol
// esté cargado
$(document).ready(function() {
    console.log("ready!");
});

// Forma compacta equivalente a lo anterior
$(function() {
    console.log("ready!");
});
```

ENCADENAMIENTO DE MÉTODOS

```
var title = $("<h1>¡Hola!</h1>");
title.css("font-family", "sans-serif");
title.css("color", "navy")
var body = $("body");
body.prepend(title);
title.fadeIn(5000);
title.fadeOut(5000);
```

```
var title = $("<h1>¡Hola!</h1>")
            .css("font-family", "sans-serif")
            .css("color", "navy");
$("body").prepend(title);
title.fadeIn(5000)
     .fadeOut(5000);
```

LECTURA Y ESCRITURA DE ATRIBUTOS

```
// Lee el atributo href del primer enlace:
$("a").first().attr("href");

// Escribe el atributo title de todos los nombres de escritor
$(".escritor").attr("title", "Esto es un nombre de escritor");

// Cambia varios atributos a la vez
$("#foto-cervantes").attr({src: "cervantes-2.jpg",
                           alt: "Cervantes por Luis de Madrazo"});

// Elimina un atributo
$("#foto-quevedo").removeAttr("height");
```

LECTURA Y ESCRITURA DE PROPIEDADES CSS

```
// Lee el valor de una propiedad CSS:
$(".info").first().css("font-size");

// Establece el valor de una propiedad CSS:
$(".info").css("font-variant", "small-caps");

// Establece una propiedad compuesta
$(".retrato").css("border", "solid red 2px");

// Establece varias propiedades a la vez:
$(".retrato").css({"padding": "5px",
                   "background-color": "gray",
                   "filter": "grayscale(50%)"});
```

LECTURA Y ESCRITURA DEL ATRIBUTO CLASS

```
// Elimina la pertenencia a una clase:
$(".escritor").last().removeClass("escritor");

// Añade la pertenencia a una clase:
$("a").last().addClass("encuadrado");

// Comprueba la pertenencia a una clase:
$("a").last().is(".encuadrado");

// Alterna la pertenencia a una clase:
$("a").last().toggleClass("encuadrado");
```

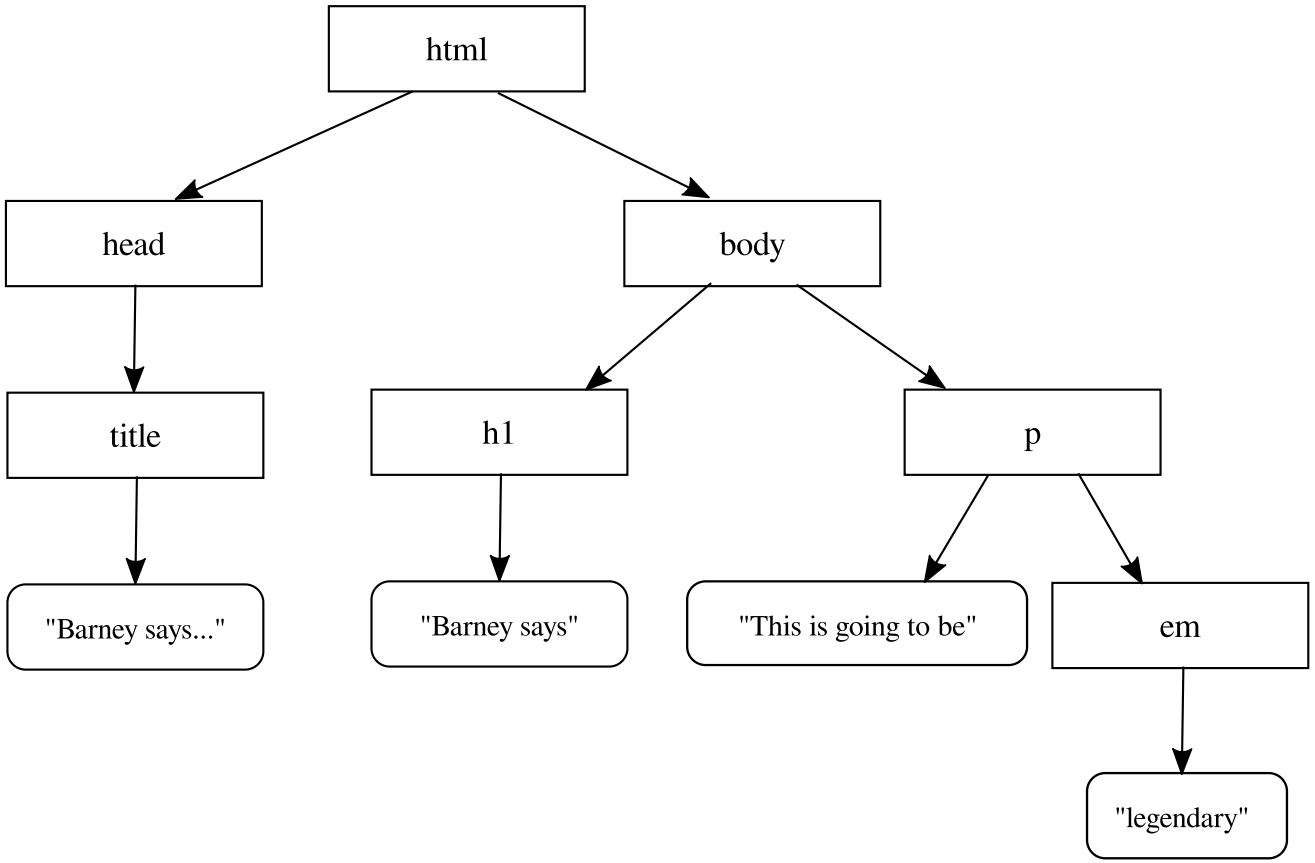
LECTURA Y ESCRITURA DEL CONTENIDO DE UN ELEMENTO

```
// Lee el contenido de un elemento como texto sin formato:
var t1 = $(".info").first().text();

// Lee el contenido de un elemento como texto HTML:
var t2 = $(".info").first().html();

// Establece el contenido de un elemento (texto plano):
$(".info").first().text("Texto borrado.");

// Establece el contenido de un elemento (HTML):
$(".info").first().html("Texto <strong>borrado</strong>.");
```

EL MODELO DOM

```
<!DOCTYPE html>
<html>
  <head>
    <title>Barney says...</title>
  </head>
  <body>
    <h1>Barney says</h1>
    <p>
      This is going to be <em>legendary</em>
    </p>
  </body>
</html>
```

EL MODELO DOM



AÑADIR CONTENIDO

```
// Añade un nuevo nodo hijo, al final:
$("ul").append("<li><div class='escritor'>José de Espronceda</div></li>");

// Crea y añade un nuevo nodo hijo, al principio:
var alfonso_div = $('<li>').append($('<div>')
                                   .addClass("escritor")
                                   .text("Alfonso X el Sabio"));
$("ul").prepend(alfonso_div);

// Añade dos nodos hermanos, uno anterior y otro posterior:
$("h1").before("<hr>");
$("h1").after("<hr>");
```

AÑADIR CONTENIDO

```
// Se pueden hacer las inserciones a la inversa,
// invocando los métodos sobre el nodo a insertar:
$(document.createTextNode(": escritores")).appendTo("h1");
$("<cite>[Wikipedia] </cite>").prependTo($(".info"));

$("<hr>").insertBefore("ul");
$("<hr>").insertAfter("ul");
```

AÑADIR CONTENIDO

```
// Envuelve elementos en otro elemento:
$(".retrato").wrap("<div>");

// Envuelve el contenido de un elemento dentro de otro:
$(".escritor").wrapInner("<strong>");
```

CLONAR CONTENIDO

```
$(".retratos").clone().insertBefore("ul");
```

ELIMINAR CONTENIDO

```
// Vacía el contenido de un nodo, pero no elimina el nodo:
$(".retratos").first().empty();

// Elimina un nodo con todo su contenido:
$("li").last().remove();

// Desenlaza un nodo del árbol, pero se conserva en una variable
var node = $("h1").detach();
$("body").append(node);
```

ACCESO A FORMULARIOS

```
// Lee el valor de un control:
$("input[name='pregunta1'").val();

// Cambia el valor de un control:
$("input[name='pregunta2'").val("Córdoba");

// Lee el botón tipo radio marcado:
$("input[name='pregunta3']:checked").val();

// Marca un botón de tipo radio:
$("input[name='pregunta3'][value='quevedo']").prop("checked", true);
```

ACCESO A FORMULARIOS

```
// Lee los controles marcados de tipo checkbox:
$("input[name='pregunta4']:checked").each(function(i, e) {
    console.log($(e).attr("value"));
});

// Marca controles de tipo checkbox:
$("input[name='pregunta4']").val(['gongora', 'cervantes']);
```

MANEJADORES DE EVENTOS

Permiten registrar acciones a llevar a cabo cuando ocurran determinados eventos sobre un elemento de la página. Por ejemplo:

- Se hace *click* sobre el elemento.
- El cursor pasa por encima del elemento.
- Cambia el valor de un control de formulario.

EVENTOS DE CARGA DEL DOCUMENTO

```
$(document).ready(function() {
  // Código a ejecutar cuando se haya cargado
  // el árbol DOM del documento.
});

$(window).on("load", function() {
  // Código a ejecutar cuando se haya representado la página
  // y cargado todos sus recursos adicionales.
});

$(window).on("unload", function() {
  // Código a ejecutar cuando se abandone la página
  // (se sigue un hipervínculo, se recarga el documento,
  // se va hacia atrás o adelante).
});
```

EVENTOS DEL NAVEGADOR

```
$("#img1").on("error", function() {
  // Código a ejecutar cuando se produzca un error
  // en la carga del elemento (p.e. una imagen).
});

$(window).resize(function() {
  // Código a ejecutar cuando se cambie el tamaño
  // de la ventana.
});

$(window).scroll(function() {
  // Código a ejecutar cuando se haga scroll en la página.
});

$("#e1").scroll(function() {
  // Código a ejecutar cuando se haga scroll
  // en el elemento con identificador #e1.
});
```

EVENTOS EN FORMULARIOS

```
$("input[name=email]").change(function() {
  // Código a ejecutar cuando cambie el valor del control.
});

$("form").submit(function(event) {
  // Código a ejecutar cuando el usuario presione en un botón
  // de envío del formulario.

  // Se puede también detener el envío:
  event.preventDefault();
});
```

EVENTOS EN FORMULARIOS (FOCO)

```
$("input[name=email]").focus(function() {
  // Código a ejecutar cuando el control consiga el foco.
});

$("input[name=email]").blur(function() {
  // Código a ejecutar cuando el control pierda el foco.
});

$("#e1").focusIn(function() {
  // Código a ejecutar cuando el elemento #e1
  // o algún descendiente suyo consiga el foco.
});

$("#e1").focusOut(function() {
  // Código a ejecutar cuando el elemento #e1
  // o algún descendiente suyo pierda el foco.
});
```

EVENTOS DEL CURSOR

```
$("#e1").click(function() {
  // Código a ejecutar cuando el usuario haga click
  // sobre el elemento #e1.
});

$("#e1").dblclick(function() {
  // Código a ejecutar cuando el usuario haga doble click
  // sobre el elemento #e1.
});

$("#e1").toggle(function() {
  // Código a ejecutar cuando el usuario haga click
  // por primera vez sobre el elemento #e1
  // (y en sucesivos ciclos).
}, function() {
  // Código a ejecutar cuando el usuario haga click
  // por segunda vez sobre el elemento #e1
  // (y en sucesivos ciclos).
});
```

EVENTOS DEL CURSOR

```
$("#e1").mouseenter(function() {
  // Código a ejecutar cuando el puntero entre en el área
  // del elemento #e1.
});

$("#e1").mouseleave(function() {
  // Código a ejecutar cuando el puntero abandone el área
  // del elemento #e1.
});

$("#e1").hover(function() {
  // Código a ejecutar cuando el puntero entre en el área
  // del elemento #e1.
}, function() {
  // Código a ejecutar cuando el puntero abandone el área
  // del elemento #e1.
});
```

EVENTOS DEL CURSOR

```
$("#e1").mousedown(function(event) {
  // Código a ejecutar cuando se presione un botón
  // dentro del área del elemento #e1.
  // Se puede saber qué botón se ha presionado:
  switch(event.button) {
    case 0:
      // botón izquierdo
      break;
    case 1:
      // botón central
      break;
    case 2:
      // botón derecho
      break;
  }
});
```

EVENTOS DEL CURSOR

```
$("#e1").mouseup(function(event) {
  // Código a ejecutar cuando se libere un botón
  // dentro del área del elemento #e1.
  // Se puede saber qué botón se ha liberado con event.button.
});

$("#e1").mousemove(function(event) {
  // Código a ejecutar cuando se mueva el cursor
  // dentro del área del elemento #e1.
});

// En todos estos eventos se pueden conocer las coordenadas
// del cursos con event.pageX, event.pageY, event.screenX, event.screenY,
// event.offsetX y event.offsetY.

// También se puede saber qué botones se encuentran presionados
// con event.buttons.

// https://developer.mozilla.org/en-US/docs/Web/API/MouseEvent
```

REFERENCIAS

- Douglas Crockford, *JavaScript: The Good Parts* O'Reilly (2008)
- David Flanagan, *JavaScript: The Definitive Guide* (7^{th} ed.) O'Reilly (2020)
- David Flanagan, *JavaScript: The Definitive Guide* (6^{a} ed.) O'Reilly (2010)

REFERENCIAS

- W3Schools
- MDN (Mozilla Developer Network)