Javascript

JavaScript (JS) es un lenguaje ligero e interpretado, orientado a objetos, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node JS.Es un lenguaje script multi-paradigma,basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

Funciones y variables

```
var  x = 5;

function myFunction ( x , y ){
    return x * y;
}

function toCelsius(fahrenheit){
    return (5/9) * (fahrenheit-32);
}
```

Patrones de Diseno (Module)

El patron Module fue originalmente creado para encapsulamiento de clases.

Sirve para mantener datos de manera privada y publica.

```
var myNamespace = (function () {

  var myPrivateVar, myPrivateMethod;

  // A private counter variable
  myPrivateVar = 0;

  // A private function which logs any arguments
  myPrivateMethod = function( foo ) {
      console.log( foo );
  };

  return {

    // A public variable
    myPublicVar: "foo",

    // A public function utilizing privates
    myPublicFunction: function( bar ) {

      // Increment our private counter
      myPrivateVar++;

      // Call our private method using bar
      myPrivateMethod( bar );

    }
  };

})();
```

Patrones de diseno (MVC)

Model= Representa el model o modelos a usar para el sitio Web

Controller= Representa a quien se encargue de controlar todos los eventos y funciones que ocurran en dicho sitio.

View= Representa la vista, lo que el usuario ve.

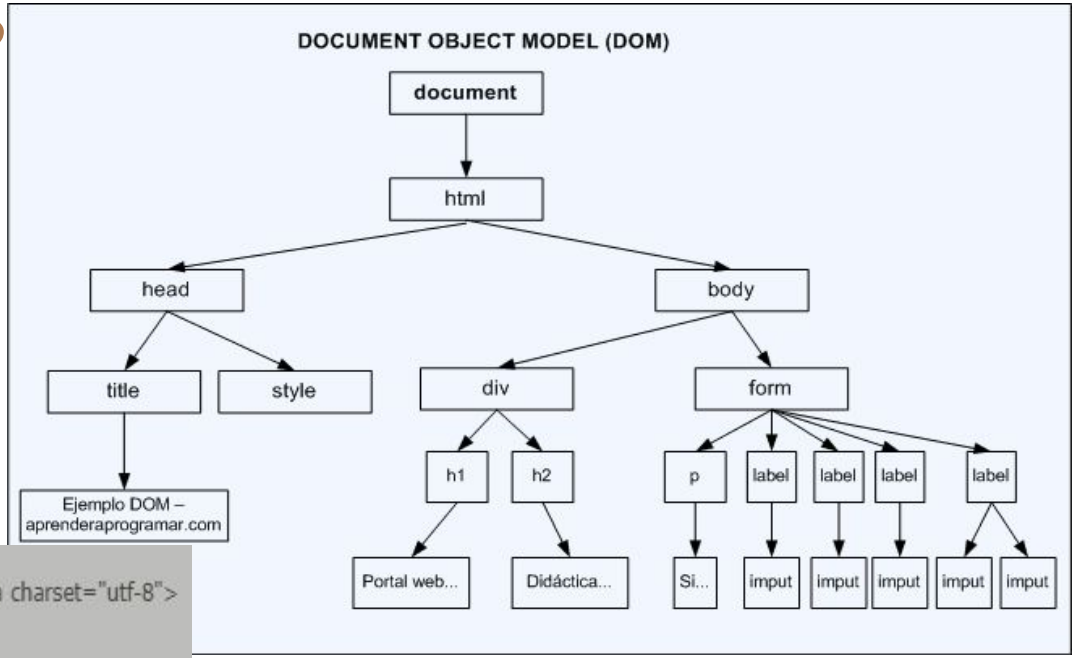
```
var Photo = Backbone.Model.extend({

    // Default attributes for the photo
    defaults: {
      src: "placeholder.jpg",
      caption: "A default image",
      viewed: false
    },

    // Ensure that each photo created has an `src`.
    initialize: function() {
       this.set( { "src": this.defaults.src} );
    }

});
```

```
var buildPhotoView = function ( photoModel, photoController ) {

  var base = document.createElement( "div" ),
      photoEl = document.createElement( "div" );

  base.appendChild(photoEl);

  var render = function () {
          // We use a templating library such as Underscore
          // templating which generates the HTML for our
          // photo entry
          photoEl.innerHTML = _.template( "#photoTemplate", {
              src: photoModel.getSrc()
          });
      };

  photoModel.addSubscriber( render );

  photoEl.addEventListener( "click", function () {
    photoController.handleEvent( "click", photoModel );
  });

  var show = function () {
    photoEl.style.display = "";
  };

  var hide = function () {
    photoEl.style.display = "none";
  };
```

```
// Controllers in Spine are created by inheriting from Spine

var PhotosController = Spine.Controller.sub({

  init: function () {
    this.item.bind( "update", this.proxy( this.render ));
    this.item.bind( "destroy", this.proxy( this.remove ));
  },

  render: function () {
    // Handle templating
    this.replace( $( "#photoTemplate" ).tmpl( this.item ) );
    return this;
  },

  remove: function () {
    this.el.remove();
    this.release();
  }
});
```

¿Que es el DOM?

Facilita una representación estructurada del documento y define de qué manera los programas pueden acceder, al fin de modificar, tanto su estructura, estilo y contenido.




```
<!DOCTYPE html>
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemplo() {
window.document.images[0].style.border = 'solid blue 10px';
document.images[1].style.border = 'solid red 10px';
document.links[0].style.color = 'grey';
}
</script>
```

Interfaces del DOM

Ejemplos DOM

- `document.getElementById(id)`
- `element.getElementsByTagName(name)`
- `document.createElement(name)`
- `parentNode.appendChild(node)`
- `element.innerHTML`
- `element.style.left`
- `element.setAttribute`
- `element.element.getAttribute`
- `element.addEventListener`
- `window.content`
- `window.onload`
- `window.dump`
- `window.scrollTo`

¿ Qué es un selector ?

Los **selectores** nos ayudan a seleccionar los diferentes elementos del DOM, bien sea por su nombre, tipo, valores o propiedades de los atributos.

¿Para qué sirve y cómo usar GetElementById?

Este método nos permite obtener un elemento del DOM según el valor del atributo id del mismo:**getElementsByTagName?**

```
<div id='div_id' > div con id='div_id' </div>
```

```
document.getElementsById("div_id")
```

¿Para qué sirve y cómo usar getElementsByTagName?

Este método nos permite obtener todos los elementos del DOM según el nombre de la etiqueta del elemento; la colección de elementos será devuelta en un array.

```
<p> etiqueta p 1 </p>
<h2> etiqueta h2 </h2>
<p> etiqueta p 2 </p>
```

```
document.getElementsByTagName("p")
/*
  nos trae un array cuyo tag name (nombre etiqueta) sea p
[
<p> etiqueta p 1 </p>,
<p> etiqueta p 2 </p>
]
*/
```

¿Para qué sirve y cómo usar getElementsByClassName?

Este método nos permite obtener todos los elementos del DOM segun el valor del atributo class del mismo; dicha colección será devuelta en un array.

```
<p class='miclase' > p con class='miclase' </p>
<h1 class='miclase' > h1 con class='miclase' </h1>
<p class='otraclase' > p con class='otraclase' </p>
<p class='miclase' > p con class='miclase' </p>
<p class='miclase' > p con class='miclase' </p>
```

```
document.getElementsByClassName("miclase")

/*
  nos trae un array con los elementos cuya clase sea 'miclase'
[
<p class='miclase' >p con class='miclase' </p>,
<h1 class='miclase' >h1 con class='miclase' >,
<p class='miclase' >p con class='miclase' </p>,
<p class='miclase' >p con class='miclase'</p>
]
*/
```

Jquery

jQuery es un framework de **JavaScript** para facilitar, entre otros, el acceso a los elementos del DOM, los efectos, interactuar con los documentos HTML, desarrollar animaciones y agregar interacción con la tecnología AJAX a páginas web.

La característica principal de la biblioteca es que permite cambiar el contenido de una página web sin necesidad de recargarla, mediante la manipulación del árbol DOM y peticiones AJAX. Para ello utiliza las funciones $() o jQuery().

Selectores en Jquery

```
/* Seleccionar etiquetas del DOM */
$("elemento")
/* Seleccionar id del DOM */
$("#idelemento")
/* Seleccionar clase del DOM */
$(".claseelemento")
```

Ejemplos

```
/* seleccionar las etiquetas "p" */
$("p")

/* seleccionar los elementos con id="inicio" */
$("#btn")

/* seleccionar los elementos con clase="principal" */
$(".menu")
```

Ejemplos

```
/* seleccionar los elementos con id="inicio" con clase="principal" y que son etiqueta "p" */
$("p#inicio.principal")

/* seleccionar las etiquetas "a" con clase="principal" dentro de una etiqueta "p" */
$("p a.principal")

/* seleccionar los elementos con id="inicio" o clase ="principal" */
$("#inicio,.principal")
```

¿ ECMAScript ?

Esto es lo que sucedió hace mucho, mucho tiempo:

1. JavaScript originalmente se llamaba JavaScript con la esperanza de capitalizar el éxito de Java.
2. Netscape luego envió JavaScript a ECMA International for Standardization. (ECMA es una organización que estandariza la información)
3. Esto da como resultado un nuevo estándar de lenguaje, conocido como ECMAScript.

En pocas palabras, ECMAScript es un estándar. Mientras que JavaScript es la implementación más popular de ese estándar. JavaScript implementa ECMAScript y se basa en él.

ECMAScript es un estándar. JavaScript es la implementación más popular de ese estándar. Otras implementaciones incluyen: SpiderMonkey, V8 y ActionScript.

¿ 'ES' ?

1. ES1: junio de 1997
2. ES2: junio de 1998
3. ES3: diciembre de 1999
4. ES4: abandonado
5. ES5: diciembre de 2009: casi 10 años más tarde, ES5 se lanzó en 2009. Pasarían casi seis años hasta que se lanzara la próxima versión de ECMAScript.
6. ES6 / ES2015 Junio de 2015: Quizás la causa de toda su confusión comienza aquí. Usted ve, ES6 y ES2015 son lo mismo.
7. ES2016 (ES7) Junio de 2016: Séptima edición de ECMAScript.
8. ES2017 (ES8) Junio de 2017: Octava edición de ECMAScript.
9. ES.Siguiente

ES6 Ventajas

- Syntactic sugar to make code cleaner
- ES6 is 100% Backward Compatible.
- New Features => Es6-features

- arrows
- classes
- template strings
- destructuring
- default + rest + spread
- let + const
- iterators + for..of
- modules
- module loaders
- map + set + weakmap + weakset
- math + number + string + array + object APIs

como..

- Babel transpiles ES6 a ES5

-Traspiler es un término específico para tomar el código fuente escrito en un idioma y transformarlo en otro idioma. Entonces, en este caso, Babel es una herramienta para transformar ES6 en ES5.

- JSBin

- JS Bin es una herramienta que le permite aprender, experimentar y enseñar utilizando tecnologías web.

- Tutorial NPM + BABEL Here