3. JS

# 2.1. Intro

Con el conocimiento adquirido hasta ahora, somos capaces de transformar diseños en código HTML. Con javascript, vamos a poder darle movimiento y comportamiento a esos diseños.

El javascript es un lenguaje de programación que conocen los browsers y por lo tanto funciona Client Side.

Para programar en Javascript utilizamos una librería llamada JQuery.

Hay que referenciarla directamente o bajarla del sitio de JQuery y referenciarla localmente. En Moon la bajamos y referenciamos.

<script type="text/javascript" src="/js/jquery/jquery-1.9.1.min.js"></script>

<script type="text/javascript" src="/js/jquery/jquery-ui-1.10.2.custom.min.js"></script>

# **2.2. Funciones y objetos**

* En Javascript podemos definir funciones de la siguiente manera:

function nameOfFunction(var1, var2, var3){

….

};

* Y objetos de la siguiente manera:

var person = {};

person.name = ‘Luis’;

person.age = 16;

alert(person.age);

* Al ser un lenguaje sin tipos, podemos definir funciones dentro de objetos:

var Person = {};

Person.name = ‘Luis’;

Person.age = 16;

Person.setVariables = function (name, age){

this.name = name;

this.age = age;

};

* En Moon Ideas acostumbramos hacer un objeto con el nombre de la página que estoy trabajando y adentro todos sus métodos. A esto lo llamamos **namespace**.

var Login = {};

Login.bindFunctions = function (){

$(“#loginButton”).on({

‘click’:function(){

Login.login();

},

‘mouseover:function(){

Login.lightUpLoginButton();

},

‘mouseout:function(){

Login.lightDownLoginButton();

}

});

};

Login.login = function(){

var username = $(“#username”).val();

var password = $(“# password”).val();

…

}

# **2.3. #**

* En JQuery, se pueden ir a buscar objetos existentes en el HTML de la siguiente manera:

var loBuscoPorId = $(“#loginButton”);

var loBuscoPorClase = $(“.loginButton”);

# **2.4. Para ejecutar un código cuando el sitio se terminó de descargar:**

$(document).ready(

function() {

…

}

);

# 2.5. Eventos

* En JQuery, uno se suscribe a eventos y actúa sobre los mismos.

$('#mainMenuSpecials').on({

'mouseover':function(){

Main.displayFullMenu();

},

'mouseout':function(){

Main.hideFullMenu();

},

click:function(){

Main.doSomething();

}

});

# 2.6. Animaciones

* En JQuery, podemos animar objetos, pasarlos de una posición a otra, cambiarles la opacidad y mucho más.
* En la primer variable definimos las transformaciones que queremos hacer sobre el elemento, en la segunda definimos el tiempo de la animación y en la última definimos la función que queremos que se ejecute una vez finalizada.

$('body’).animate({opacity:1;},1000,

function(){

alert(“Done”);

}

);

* Si bien las animaciones de javascript se usan y mucho, se recomienda y en moon se estila a hacer las animaciones utilizando clases de css.  
    
  Por ejemplo, si yo quiero que con el click de un botón se abra un menú desplegable, podría hacerlo de dos formas.  
    
  Con js:  
    
  $(“#button”).click(function(){  
   $(“#menu”).animate({“height”:”100px”}, 300);  
  };  
    
  O apoyándome en una clase de CSS.  
    
  CSS  
  #menu{  
   height:0px;  
   transition: height .3s ease-out;  
  }  
  #menu.open{ → Solo aplica al div #menu cuando tiene también la clase open  
   height:100px;  
  }  
    
  JS  
  $(“#button”).click(function(){  
   $(“#menu”).addClass(“open”);  
  };  
    
  El beneficio es que queda el código más prolijo, podemos reutilizar animaciones fácilmente, y generalmente es más eficiente.

# 2.7. Ejercicio fácil

Al iniciar, el sitio no muestra la imagen.

Cuando está todo cargado, la imagen debe mostrarse gradualmente utilizando opacity.

1 - utilizando animaciones de javascript

2 - utilizando una clase de CSS y transitions

# 2.8. Agregar eventos a elementos que no existen…todavía

Puede suceder que haya elementos de HTML a los que me gustaría agregarles comportamientos pero todavía no existen en mi sitio.

Por ejemplo, en una galería de imágenes, en donde las imágenes pasan una después de la otra, hay imágenes que todavía no existen.

Puede pasar en juegos y en muchas otras situaciones.

Para estas situaciones, agregamos el evento a un elemento padre

$('#galery).on({

'mouseover':function(){

Gallery.pause = true;

},

'mouseout':function(){

Gallery.pause = false;

}

}, “#mainImage”,

);

En este ejemplo estamos agregando el comportamiento a “#gallery”, que si existe en el HTML, pero al final especificamos que el comportamiento es para cualquier objeto “#mainImage” dentro de él.

# 2.9. Ejercicio muy difícil

El ejercicio consta en hacer una galería de imágenes.

Las imágenes van pasando hasta que frena porque mi mouse pasa por encima de la imagen, luego continúa.

Hacerlo en partes.

1. Hacer función que carga las imágenes existentes de HTML a javascript y las guarda en un array.
2. Hacer función que selecciona la próxima imagen a mostrar.
3. Hacer función que recibe una imagen y la pone en el lugar de la vieja.
4. Mejorar la función anterior, hacer una animación que juega con los opacities de las imágenes.
5. Agregarle comportamientos a la imagen principal, cuando pasa el mouse seteamos la variable pause en true, cuando se va, en false.



# 2.10. Ajax

Hasta ahora sólo podemos enviar información al servidor a través de Forms de HTML.

Ajax nos permite consultar otras páginas sin necesidad de redirigir al usuario.

Lo vemos en un ejemplo, en donde voy a mostrar en un div un texto que tengo escrito en otro documento:

$.ajax({

type: "POST",

url: "/crearUsuario.php",

data:{

“nombre”: “Sebastian”,  
 “apellido”: “del campo”,

},

success: function(response){

$(#resultDiv).val(response);

},

error: function(){

alert('Error obteniendo datos del sitio”);

}

});

# 2.11. Ejercicio fácil

Experimentar con Ajax haciendo el punto 2.10.

# 2.12. Ejercicio de investigación

¿Qué es JQuery UI?

¿Qué es Sortable?  
Desarrollar un ejemplo de una lista que puede ordenar arrastrando elementos solamente en el eje Y.