Tutorial de YUI

La librería WireIt trabaja con YUI(Yahoo User Interfaz). Esta es una librería desarrollada por Yahoo y es un FrameWork de JavaScript.

Nos podemos descargar YUI de **http://developer.yahoo.com/yui/** . Existen varias versiones de YUI, la versión 2.8.2 es la versión actual. Es bueno saber que Wareit trabaja con la versión 2.7 .

En http://developer.yahoo.com/yui/2/ existe un apartado que se llama Getting Started, Aquí se nos explica de forma general los pasos a seguir para poner en marcha y aprender a utilizar la librería.

Para utilizar YUI es necesario incorporar en nuestra página HTML las siguientes librerías:

build/yuiloader/yuiloader-min.js
build/event/event-min.js

Esas son las mínimas para poder hacer uso del objeto YAHOO que es el que permite crear las aplicaciones e inicializar el FrameWork.

El api de del framework YUI lo podemos ver online en http://developer.yahoo.com/yui/docs/ . También cuando descargamos la librería y descomprimimos podremos visualizar una carpeta llamado doc, esta tiene adentro un archivo index.html y esa es nuestra api.

La idea es comenzar a aprender YUI para luego irnos incorporando en el uso de Wire IT, recordemos que Wireit está basado en YUI.

Una de las ventajas de YUI es el soporte de diferentes navegadores web. Estos los podemos ver en la lista del Graded Browser Support (Calificado compatibilidad de exploradores) http://developer.yahoo.com/yui/articles/gbs/#gbschart . Recordemos que una de las principales funciones de un FrameWork es hacerse responsable de las cosas que no corresponden al desarrollo como tal, es decir, validar si un determinado script de JavaScript funciona o no en un determinado Navegador Web no es algo que pertenezca al desarrollo de la aplicación, es algo que debe hacerse en todos los desarrollos que realicemos con JavaScript, y es aquí donde YUI se encarga de colocar el código correcto de acuerdo al navegador que está utilizando el usuario.

YUI tiene componentes CSS y componentes JavaScript(Núcleo de JavaScript que normaliza el Dom, Widgets y herramientas de desarrollo) que están allí para facilitar nuestro trabajo, lógicamente debemos primero aprender a como usarlos, lo que significa que al principio nos nadrá trabajo.

Vamos a desarrollar nuestra primera aplicación utilizando el FrameWork YUI. Lo primero será realizar el acostumbrado “HOLA MUNO “, algo muy simple en JavaScript, pero veamos como se hace utilizando el FrameWork YUI:

Lo primero será crear nuestra página html con las respectivas etiquetas mínimas necesarias:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title></title>
  <meta name="GENERATOR" content="Quanta Plus" />
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
   <div id="contenedor">
     <p>Haga click para ver el mensaje de Hola Mundo.</p>
   </div>
</body>
</html>
```

Como se puede observar en al código de arriba hemos creado 1 div que tiene como id la palabra “contenedor” . La idea es crear un avento asociado al div contenedor que permita sacar un aviso javascript que diga: “Hola Mundo”

Podemos darle estilo al div para mejorar su apariencia:

```
<style type="text/css">
    #contenedor {
      background-color:#00CCFF;
      border:1px dotted black;
      padding:1em; cursor:pointer;
    }
  </style>
```

Como vamos a utilizar el FrameWork lo segundo que haremos será llamar las librerías necesarias para utilizar el cargador de eventos(Esto se debe colocar dentro de la etiqueta <head>):

```
<script type="text/javascript" src="build/yahoo/yahoo-min.js"></script>
<script type="text/javascript" src="build/event/event-min.js"></script>
```

NOTA: Todo el código de YUI se encuentra dentro de la carpeta build

Una vez incluidas esta librerías procedemos a crear una función que imprima Hola mundo(Esto también va dentro del head)

```
var funcionSaludar = function(varaibleEvento)
{
    alert('¡Hola Mundo!');
}
```

Finalmente llamamos al cargador de eventos del YUI para proceder asociar la función a algún elemento de la página e indicarle bajo que Evento actuará:

YAHOO.util.Event.addListener("contenedor", "click", funcionSaludar);

El método addListener recibe tres parámetros:

YAHOO.util.Event.addListener(“Id de elemento HTML”, “Evento al que va actuar”, “Funcion que se llamará cuando ocurra el evento”);

Es por eso que pasamos los parámetros: contenedor, click y funcionSaludar

La página completa quedaría de esta forma:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title></title>
  <meta name="GENERATOR" content="Quanta Plus" />
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <!--   Estilos que le vamos a dar al div contenedor -->
  <style type="text/css">
    #contenedor {
      background-color:#00CCFF;
      border:1px dotted black;
      padding:1em; cursor:pointer;
    }
  </style>
  <!--   Carga de las librería de YUI para utilizar el manejador de eventos -->
   <!--Asumimos que la librería YUI está dentro de la carpeta build. Esta carpeta lo podemos observar cuando
  descargamos a YUI, y es la única necesaria para la utilización del FrameWork-->
  <script type="text/javascript" src="build/yahoo/yahoo-min.js"></script>
  <script type="text/javascript" src="build/event/event-min.js"></script>

  <script type="text/javascript">
  //Funcion que se ejecutará para saludar
  var funcionSaludar = function(e)
  {
    alert("¡Hola Mundo!");
  }

  //Procedemos a añadir la funcion al Listener
  YAHOO.util.Event.addListener("contenedor", "click", funcionSaludar);
  </script>
</head>
<body>
   <div id="contenedor">
     <p>Haga click para ver el mensaje de Hola Mundo.</p>
   </div>
```

</body>
</html>

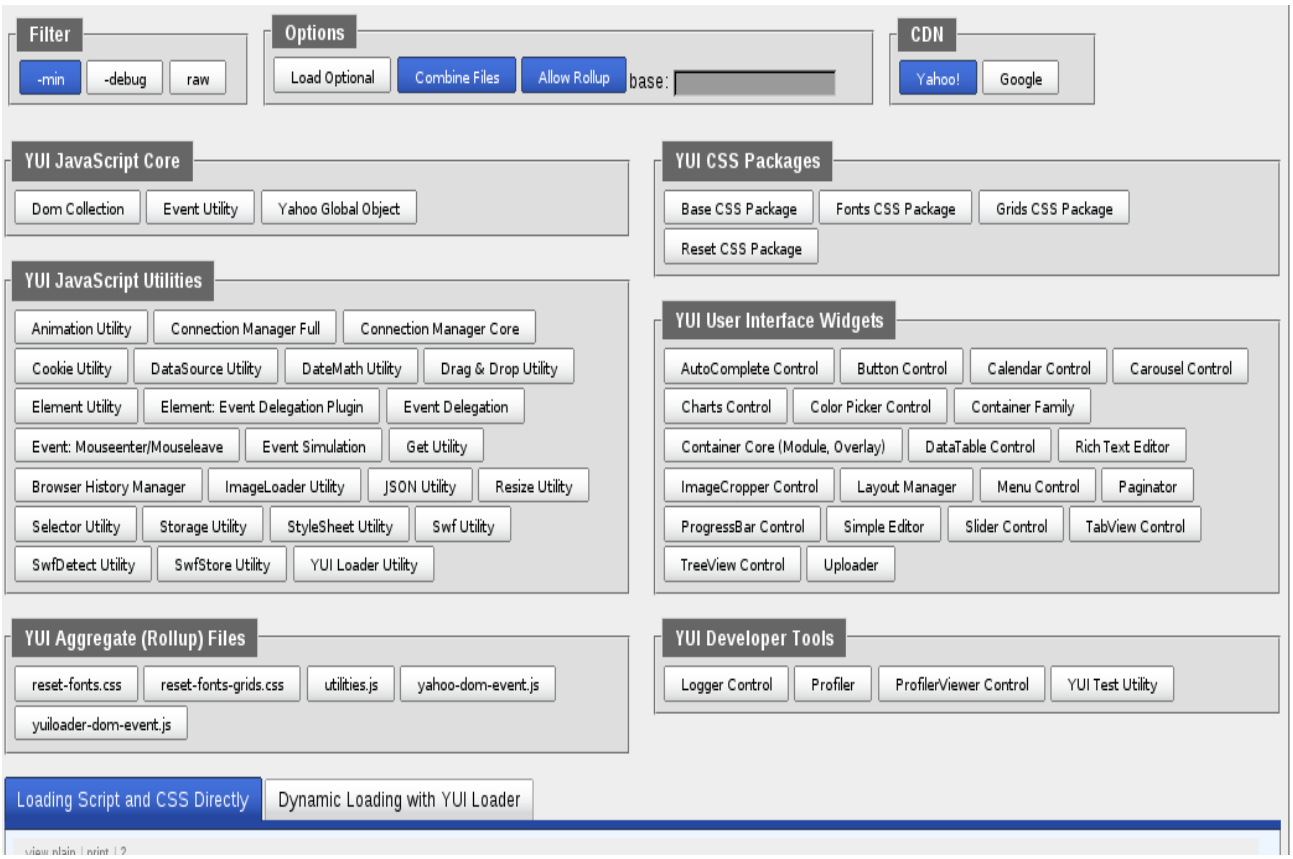
A este punto del tutorial es posible que se estén haciendo las siguientes preguntas:

1. ¿De que me sirve YUI si esto lo puedo hacer en JavaScript sin tener que importar los archivos .js del YUI?
2. Dentro de la carpeta build hay muchas subcarpetas con diferentes archivos .js ¿Cómo sé cuales son los archivos que debo usar?

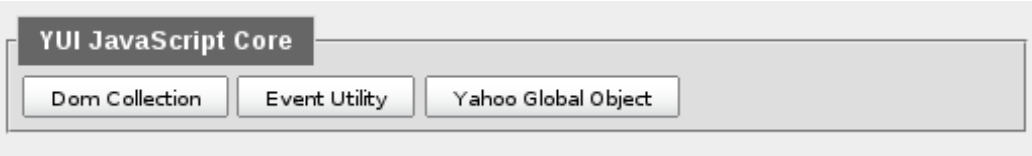
Bueno, respondamos las preguntas:

La respuesta a la primera pregunta es que para escribir un simple hola mundo no hace falta importar YUI, sin embargo, a medida que la aplicación comienza a hacerse compleja y necesitamos realizar interfaces que tienden a tener códigos JavaScript complejos, tales como cambiar la posición de un elemento, realizar un buscador con lista desplegable, entre otros, ahí si veremos la utilidad de utilizar un FrameWork que ya tenga interfaces que realicen ese trabajo.

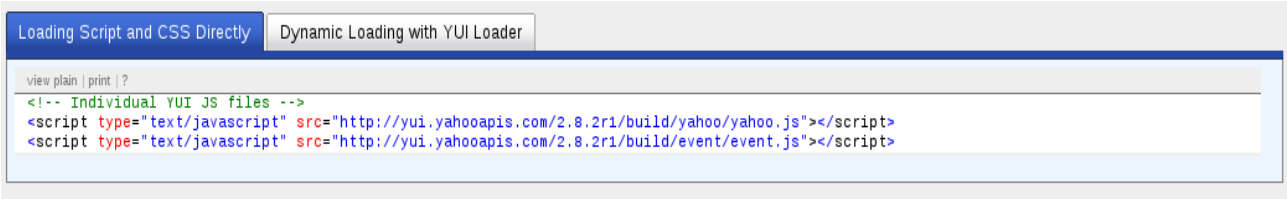
La respuesta a la segunda pregunta es que YUI tiene un API que permite visualizar sus diferentes componentes, el API la podemos ver en: http://developer.yahoo.com/yui/docs/ . Al principio saber cual es el módulo que uno desea usar del API es difícil, por eso es bueno hacer varios ejemplos para irse familiarizando con esto, sin embargo, una vez que sabemos cual es el módulo a utilizar entonces procedemos a irnos al cargador de dependencias que nos ofrece YUI. Para el caso del código de arriba vamos a utilizar el módulo Event, esto pertenece al Event Utility. Nos dirigimos entonces a http://developer.yahoo.com/yui/articles/hosting/ . Ahí vamos a ver una pantalla como esta:



Lo que se ve arriba es una interfaz que nos ofrece YUI para poder determinar cuales son los archivos que debe incluir para utilizar determinado componente. Está clasificado por la naturaleza de cada componente, por ejemplo podemos ver un tag que dice YUI JavaScript Core

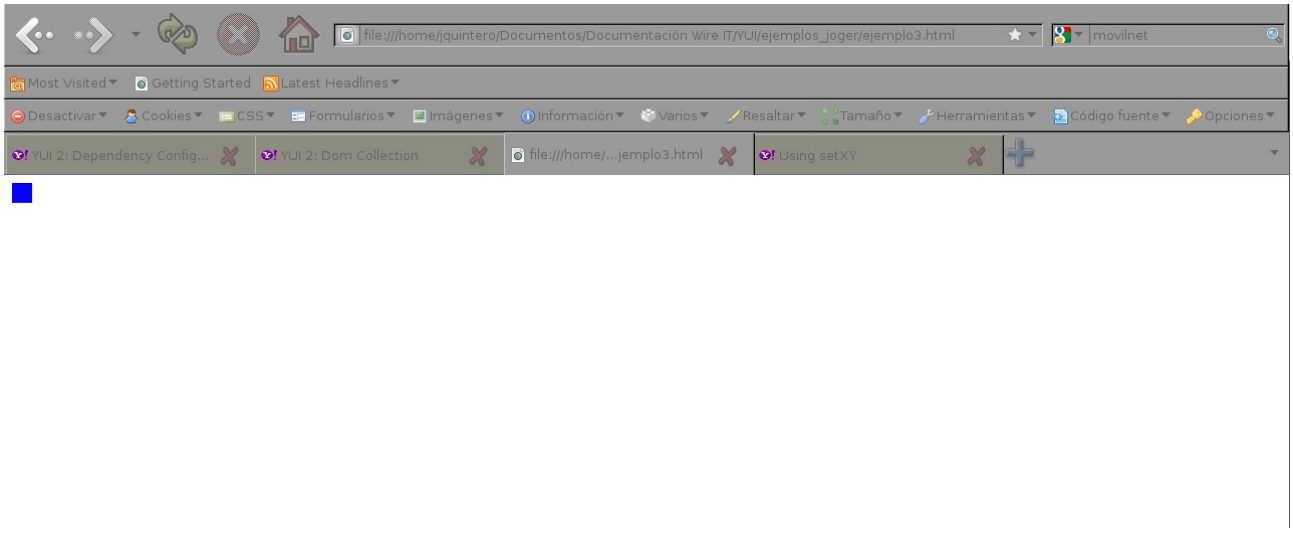


El YUI JavaScript Core es el núcleo de JavaScript de YUI. En esta sección vemos que hay un botón que dice: **Event Utility.** Si hacemos click en este botón en la parte de Loading Script and CSS Directly podremos observar los diferentes archivos que se cargan:

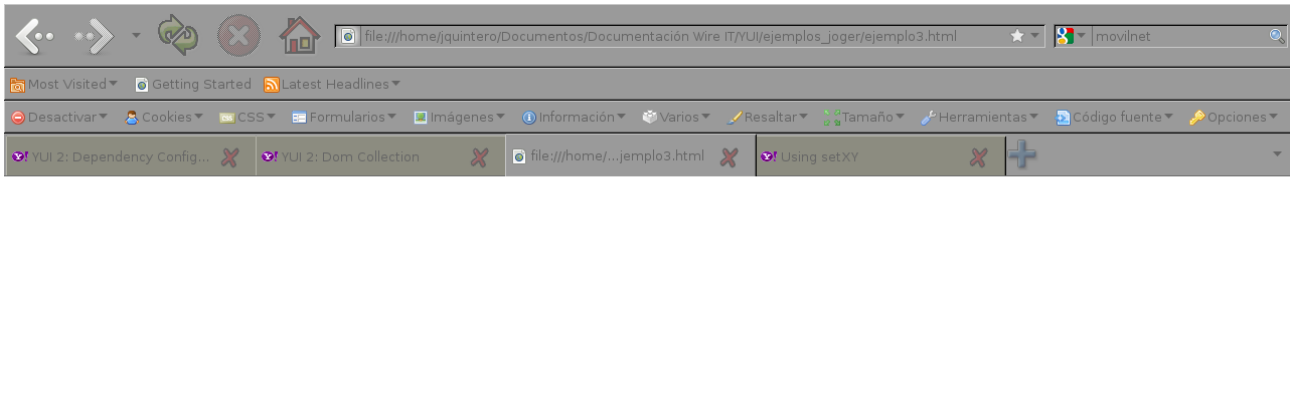


Como hemos visto arriba saber cuales son los archivos a cargar para utilizar determinado componente lo hace el Dependency Configurator.

Ahora vamos a hacer otro ejemplo un poco más complejo. Queremos realizar una página que tenga un pequeño cuadrado en cualquier posición del documento. Vamos a realizar una función que haga que ese cuadrado se desplace a un punto del documento donde se haga click . Nuestra pagina será algo como esto:



La idea es que cuando se haga click en cualquier otra parte del documento el cuadrado Azul se mueva al lugar donde se hizo click.



Pensemos un momento en JavaScript sin utilizar YUI. Para encontrar la posición actual de un elemento html lo podemos hacer a través de las propiedades offsetLeft y offsetTop, claro, si utilizamos un navegador Web basado en Netscape, porque si no es basado en Netscape entonces podrías tener que utilizar las propiedades pageX y pageY. Aquí comenzamos a ver la potencialidad de YUI, ya que él sería quién se encargaría de verificar que código usar para poder mover el elemento html. A demás, debemos saber todas las instrucciones JavaScript para hacer que un elemento se mueva, mientras que con YUI solo buscamos en el API y al encontrar los métodos hacemos uso de ellos.

El código de la imagen de arriba quedaría así:

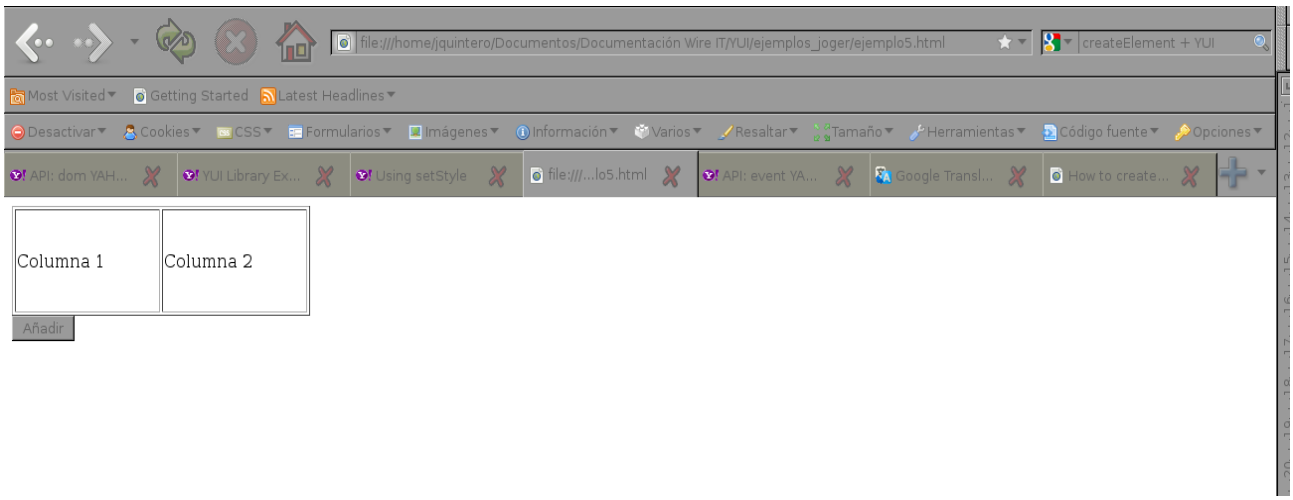
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
 <title></title>
 <meta name="GENERATOR" content="Quanta Plus" />
 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
 <!--  Damos estilo al div que contendrá el cuadrado rojo -->
 <style type="text/css">
  #elementoaMover {
   background-color:#00f;
   height:20px;
   width:20px;
  }
 </style>
 <!--  Incluimos la librería necesaria para trabajar con Event.on -->
 <script type="text/javascript" src="build/yahoo-dom-event/yahoo-dom-event.js"></script>
 <script type="text/javascript">
    //Creamos la función que movería el elemento a la posición del ratón
     var mover = function(e) {
        YAHOO.util.Dom.setXY('elementoaMover', YAHOO.util.Event.getXY(e));
     };
     //Registramos el evento para que al hacer click sobre el documento se llame la función mover
```

```
      YAHOO.util.Event.on(document, 'click', mover);
  </script>
</head>
<body>
  <div id="elementoaMover"></div>
</body>
</html>
```

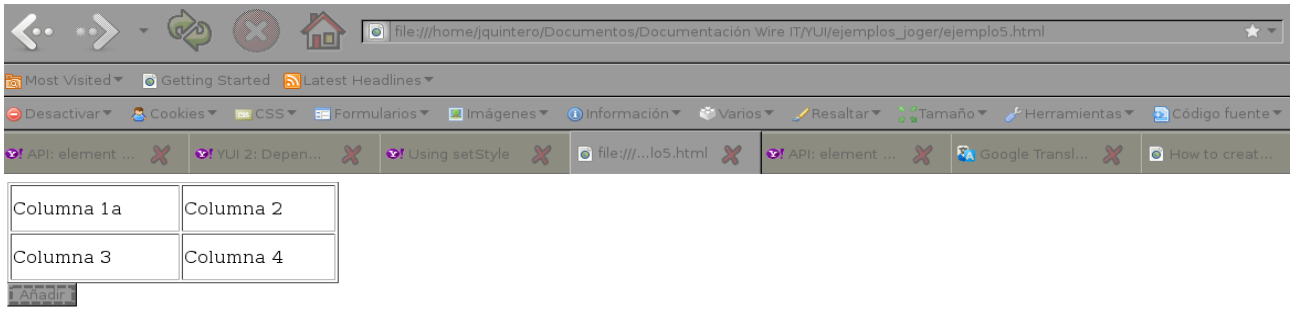
Hagamos otro ejemplo de manipulación del DOM a través de YUI. El código que verán a continuación permite afectar los estilos de un elemento HTML. La sintaxis es la siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
  <title></title>
  <meta name="GENERATOR" content="Quanta Plus" />
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  <!--   Damos estilo al div que contendrá el cuadrado rojo -->
  <style type="text/css">
   #elementoCambiar {
     background-color:#ccc;
     height:200px;
     width:200px;
   }
  </style>
  <!--   Incluimos la librería necesaria para trabajar con Event.addListener -->
  <script type="text/javascript" src="build/yahoo-dom-event/yahoo-dom-event.js"></script>
  <script type="text/javascript">
     //Creamos la función que movería el elemento a la posición del ratón
      var cambiarColor = function(e) {
         YAHOO.util.Dom.setStyle('elementoCambiar', 'background-color','#FF0004');
      };
      //Registramos el evento para que al hacer click sobre el documento se llame la función mover
      YAHOO.util.Event.addListener('ejecutar', 'click', cambiarColor);
  </script>
</head>
<body>
  <div id="elementoCambiar"></div>
  <input id="ejecutar" type="button" name="cambiar" />
</body>
</html>
```

Hagamos un ejemplo donde vamos a necesitar añadir más filas a una tabla de las que tiene cuando se carga la página HTML por primera vez. La página la primera vez que la cargamos nos mostrará algo como esto:



La idea aquí es que cuando se haga click sobre el botón añadir se proceda a crear una fila a la tabla. Algo como esto:



El código fuente para realizar eso sería:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <script type="text/javascript" src="build/yahoo-dom-event/yahoo-dom-event.js"></script>
  <script type="text/javascript" src="build/element/element-min.js"></script>
  <script type="text/javascript">
    var funcionAnhadir = function()
    {
          //OBTENEMOS LA TABLA A LA QUE LE VAMOS A AGREGAR LA NUEVA FILA Y LAS COLUMNAS
        var idTabla = YAHOO.util.Dom.get('idTabla');
        //CREAMOS LA FILA
        var eTr = new YAHOO.util.Element(document.createElement('tr'));
        //COLOCAMOS A LA FILA EL ID tr1
        eTr.set('id', 'tr1');
        //AÑADIMOS EL TR QUE CREAMOS A LA TABLA
        eTr.appendTo(idTabla);
        //CREAMOS EL PRIMER TD
        var eTd = new YAHOO.util.Element(document.createElement('td'));
        //COLOCAMS COMO ID td1
```

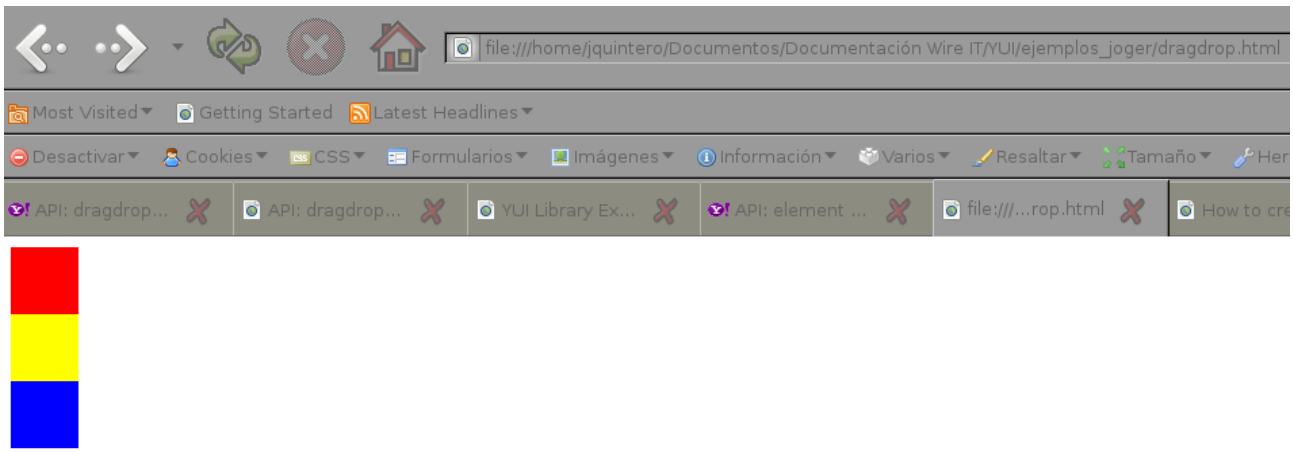
```
        eTd.set('id', 'td1');
        //AÑADIMOS TEXTO AL TD
        eTd.set('innerHTML', 'Columna 3');
        //INSERTAMOS EL TD DENTRO DEL TR QUE SE CREO
        eTd.appendTo(eTr);
        //REPETIMOS EL MISMO PROCESO PARA CREAR EL SEGUNDO TD
        var eTd = new YAHOO.util.Element(document.createElement('td'));
        eTd.set('id', 'td2');
        eTd.set('innerHTML', 'Columna 4');
        eTd.appendTo(eTr);
    };

    YAHOO.util.Event.addListener("boton", "click", funcionAnhadir);
  </script>
</head>
<body>
  <table id="idTabla" border="1" style="width:300px;height:100px;">
    <tr id="tr1">
      <td>Columna 1a</td>
      <td>Columna 2</td>
    </tr>
  </table>
  <input id="boton" type="button" name="Añadir" value="Añadir" />
</body>
</html>
```

Hasta ahora hemos hechos ejemplos sencillos. La idea ha sido familiarizarnos un poco con el uso de YUI. Sé que existen muchas dudas en relación a cómo supe los métodos que tenía que usar, recordemos que la respuesta a esto es el API y por supuesto hay que investigar y leer, en la medida que más usemos YUI notaremos que cada vez hacer algo será más fácil. Al principio dije que la idea de un FrameWork era aligerarte trabajo, sin embargo, mientras aprendemos a usarlo la verdad es que nos da trabajo.

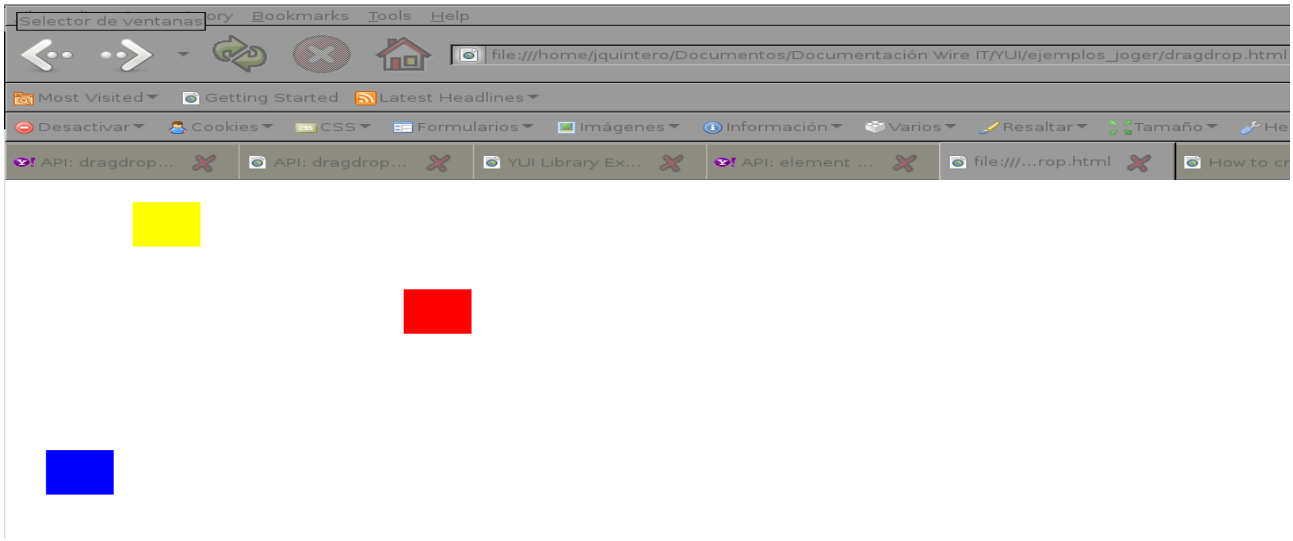
Hagamos ahora ejemplos un poco más complejos, esto con la idea de ver realmente la potencialidad de YUI.

Vamos a realizar una interfaz que permita mover elementos en un documento HTML. La interfaz consiste en tener tres cuadrados de colores diferentes, los cuales cuando el usuario mantenga el ratón presionado el botón izquierdo sobre uno de estos podrá desplazar a la posición que desee hasta que suelte el ratón.

La idea es que la página al principio aparezca como esto:



Y que al hacer click sobre uno de los cuadros, por ejemplo el rojo, lo podamos arrastras hasta donde queramos(Se debe poder mover cualquier cuadro que el usuario desee):



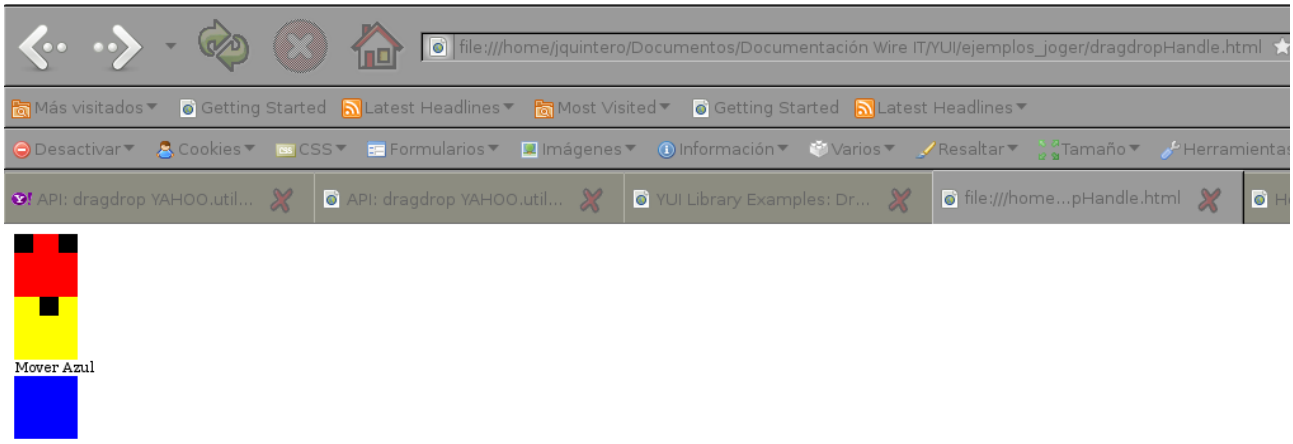
Para esto vamos a hacer uso del módulo de YUI llamado dragdrop. El código sería el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <style>
    #dd-demo-1 {
      width:50px;
      height:50px;
      background-color:red;
    }
    #dd-demo-2 {
      width:50px;
      height:50px;
      background-color:yellow;
```

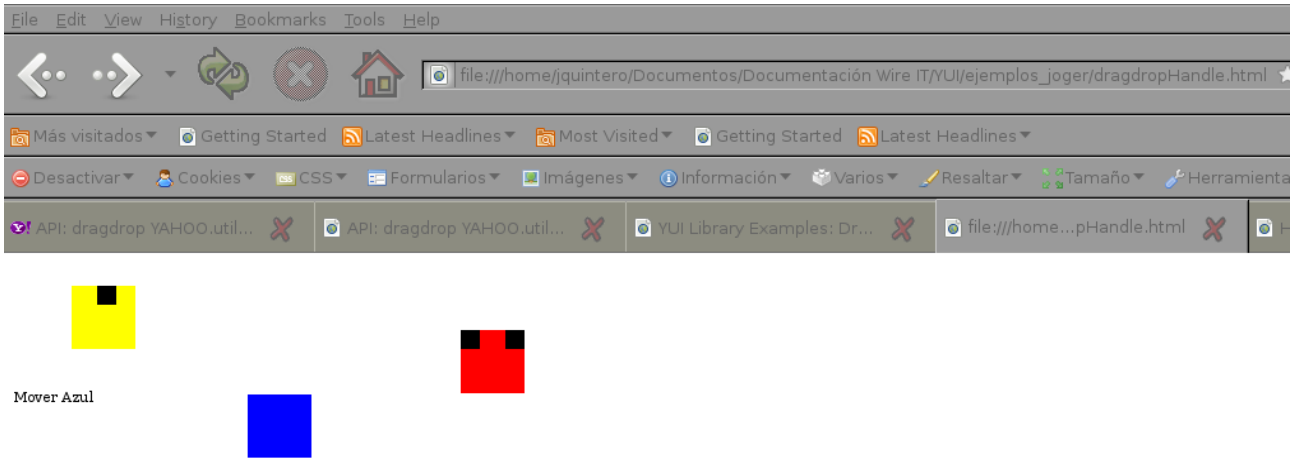
```
    }
    #dd-demo-3 {
      width:50px;
      height:50px;
      background-color:blue;
    }
  </style>
  <script type="text/javascript" src="build/yahoo-dom-event/yahoo-dom-event.js"></script>
  <script type="text/javascript" src="build/dragdrop/dragdrop-min.js"></script>
</head>
<body onload="moverCuadro()">
  <div id="dd-demo-1" class="dd-demo"></div>
  <div id="dd-demo-2" class="dd-demo"></div>
  <div id="dd-demo-3" class="dd-demo"></div>
  <script type="text/javascript">
       /*var dd, dd2, dd3;
       YAHOO.util.Event.onDOMReady(function() {
           dd = new YAHOO.util.DD("dd-demo-1");
           dd2 = new YAHOO.util.DD("dd-demo-2");
           dd3 = new YAHOO.util.DD("dd-demo-3");
       });*/
      var moverCuadro = function(){
         dd = new YAHOO.util.DD("dd-demo-1");
         dd2 = new YAHOO.util.DD("dd-demo-2");
         dd3 = new YAHOO.util.DD("dd-demo-3");
      };

      var quitarMovimiento= function(){
         dd = NULL;
         dd2 = Null;
         dd3 = new YAHOO.util.DD("dd-demo-3");
      };
</script>
</body>
</html>
```

Aquí es donde empezamos a ver las bondades de un FrameWork. Como observamos en el código de arriba son pocas las lineas de código empleadas para lograr este efecto,

Hagamos el mismo ejemplo de arriba con la diferencia de que el objeto sólo se podrá mover cuando se toque un punto del mismo. Es como que un objeto tiene un mango para moverlo.



En la imagen de arriba vemos como hay tres cuadrados, cada cuadrado tiene unos cuadritos pequeños de color negro, esto son los Handlers, es decir, el punto que va permitir mover el cuadrado grande. Existe una excepción con el cuadro Azul, ya que este tiene el Handler en la palabra que dice: “Mover Azul”. Si realizamos movimientos en los cuadros podríamos ver algo como esto:



El Handler del cuadro azul no se mueve porque está fuera del cuadro. Los demás Handler si se mueven porque están dentro de los cuadrados.

El código fuente de esta página quedaría así:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title></title>
  <script type="text/javascript" src="build/yahoo-dom-event/yahoo-dom-event.js"></script>
```

```
  <script type="text/javascript" src="build/dragdrop/dragdrop-min.js"></script>
  <script type="text/javascript">

    var dd, dd2, dd3;
    YAHOO.util.Event.onDOMReady(function() {
        dd = new YAHOO.util.DD("dd-demo-1");

        // Configure one or more child element as a drag handle
        dd.setHandleElId("dd-handle-1a");
        dd.setHandleElId("dd-handle-1b");

        dd2 = new YAHOO.util.DD("dd-demo-2");
        dd2.setHandleElId("dd-handle-2");

        dd3 = new YAHOO.util.DD("dd-demo-3");
        dd3.setHandleElId("dd-handle-3a");
        // A handle that is not child of the source element
        dd3.setOuterHandleElId("dd-handle-3b");
    });

</script>
<!--   Estilos para darle a los div apariencia de cuadrados -->
  <style type="text/css">
    body {
      font-size:8pt;
    }
    #dd-handle-1a {
      width:15px;
      height:15px;
      background-color:#000;
    }

    #dd-handle-1b {
      width:15px;
      height:15px;
      background-color:#000;
      position:relative;
      left:35px;
      top:-15px
    }

    #dd-handle-2 {
      width:15px;
      height:15px;
      background-color:#000;
      position:relative;
      left:20px;
      top:0px
    }

    #dd-demo-1, #dd-demo-2, #dd-demo-3 {
      width:50px;
```

```
      height:50px;
      position:relative;
    }
    #dd-demo-1 {
      background-color:red;
    }

    #dd-demo-2 {
      background-color:yellow;
    }

    #dd-demo-3 {
      background-color:blue;
    }
  </style>
</head>
<body>
  <div id="playground">
    <div id="dd-demo-1" class="dd-demo">
      <div id="dd-handle-1a" class="dd-multi-handle-1"></div>
      <div id="dd-handle-1b" class="dd-multi-handle-2"></div>
    </div>
    <div id="dd-demo-2" class="dd-demo">
      <div id="dd-handle-2" class="dd-handle"></div>
    </div>
    <div id="dd-handle-3b" class="dd-outer-handle">Mover Azul</div>
    <div id="dd-demo-3" class="dd-demo"></div>
  </div>
</body>
</html>
```

Lo importante en estos ejemplos es observar que YUI tiene una serie de métodos que nos permiten manipular el DOM de JavaScript. Recordemos que si queremos ver todos los métodos de esta clase lo podemos hacer desde el API:

http://developer.yahoo.com/yui/docs/YAHOO.util.Dom.html

Todos los ejemplos que hicimos arriba son sacados de la documentación de YUI. Si queremos ver más ejemplos para tener una mejor idea de lo que hace YUI podemos visitar:

http://developer.yahoo.com/yui/examples/

Lo que queda es ver ejemplos, realizarnos y comenzar a familiarizarnos con YUI, como ya dije antes, al principio será duro, una vez familiarizados con el FrameWork nuestros desarrollos serán más rápidos y óptimos.

Tutorial de Wire it

La librería Wireit la podemos encontrar en http://neyric.github.com/wireit/

Es una librería de código abierto creada en JavaScript que permite el desarrollo de aplicaciones con interfaz gráfica de flujo de datos, modelización gráfica y creación de editores gráficos.

Wireit está basado en YUI y en inputEx. Estos dos son FrameWorks. El api de inputEx lo podemos encontrar en:

http://neyric.github.com/inputex/api/index.html

El de YUI lo podemos ver en el tutorial de YUI.

En la medida en que trabajemos con Wireit iremos introduciéndonos en los conceptos que se manejan en esta librería. Lo primero que vamos a hacer es familiarizarnos con los conceptos.

El objetivo de Wireit es permitir la utilización de interfaces para la creación de:

1. Wires
2. Terminals
3. Containers
4. Layers

Wireit se apoya en la librerías:

1. YUI (FrameWork como Jquery, Motools, Prototipe, etc)
2. inputEX (Creación de campos de formularios)
3. Canvas (Permite la creación de los wire de forma dinámica Ver: http://billmill.org/static/canvastutorial/)

Así mismo Wireit permite la utilización de plugins(pequeñas aplicaciones más completas) como son:

1. Composable
    1. JBox
2. Editor
    1. WiringEditor
3. FormContainers

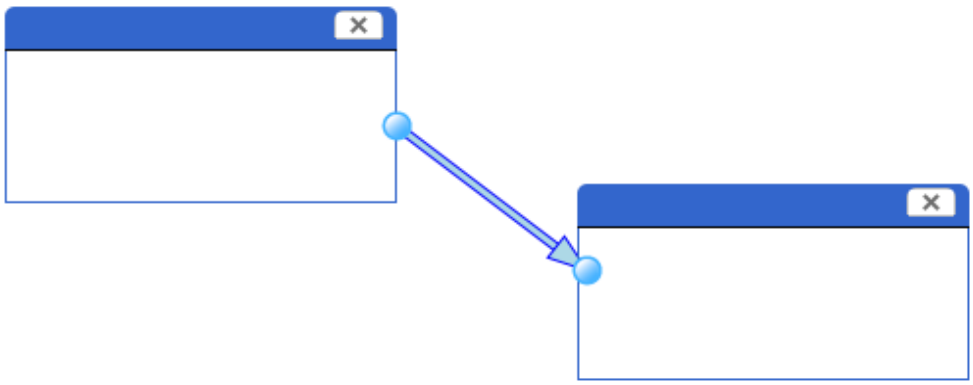
Veamos ahora que son casa una de esas cosas:

Terminals: Son los elementos que permiten el inicio o llegada de una Wire (La linea que se traza de un punto a otro) Los terminales se usan por lo general dos, ya que la idea de los terminales es permitir al usuario unidor dos elementos a través de estos.

Wires: Los terminales se pueden unir a través de una flecha(Wire) que puede ser de distintas formas, todo depende de la configuración que realicemos. La imagen de abajo muestra dos terminales unidos por una flecha:

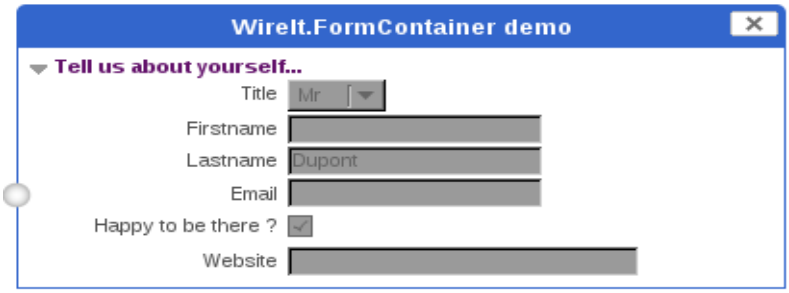
NOTA: Cuando los terminales están unidos por una flecha estos cambian de color

Containers: Son los elementos que contendrán a los terminales. Un Container puede tener tantos terminales como un usuario considere necesarios para el uso que se les esté dando. La idea de colocar un terminal en un container es poder unirlo a otro terminal de otro container. La siguiente imagen muestra dos Containers unidos por dos terminales y un Wire

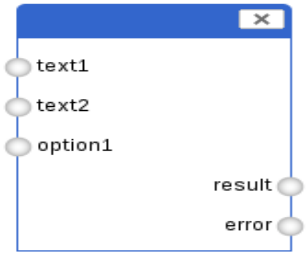


Como podemos observar en la imagen de arriba los contenedores tiene un Handler, que es la linea azul, y tienen una X que representa cerrar el contenedor. Así mismo le hemos añadido a cada contenedor un terminal y luego los unimos con una flecha.

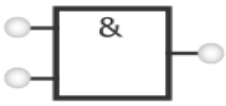
Existen varios tipos de contenedores, por ejemplo hay un contenedor que se llama FormContainer que nos permite tener campos de un formulario dentro de él:



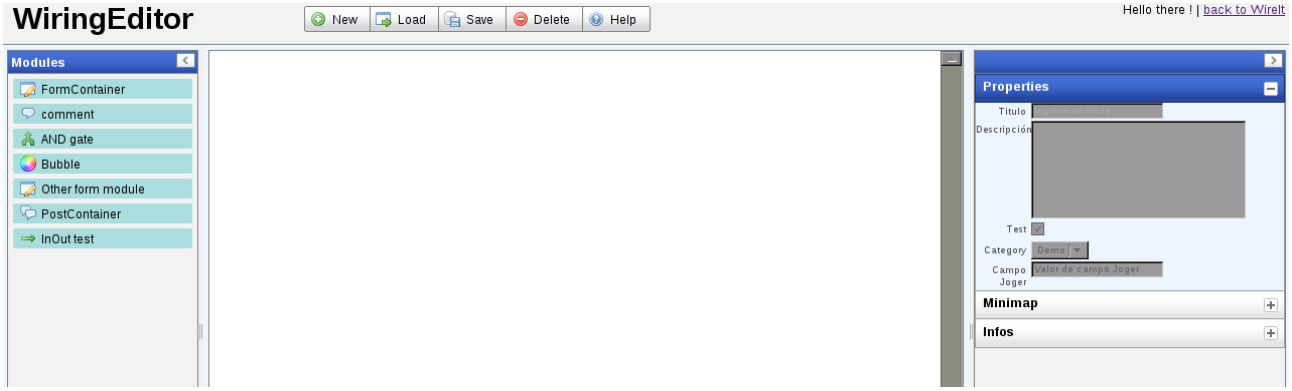
Existe el InOutContainer:



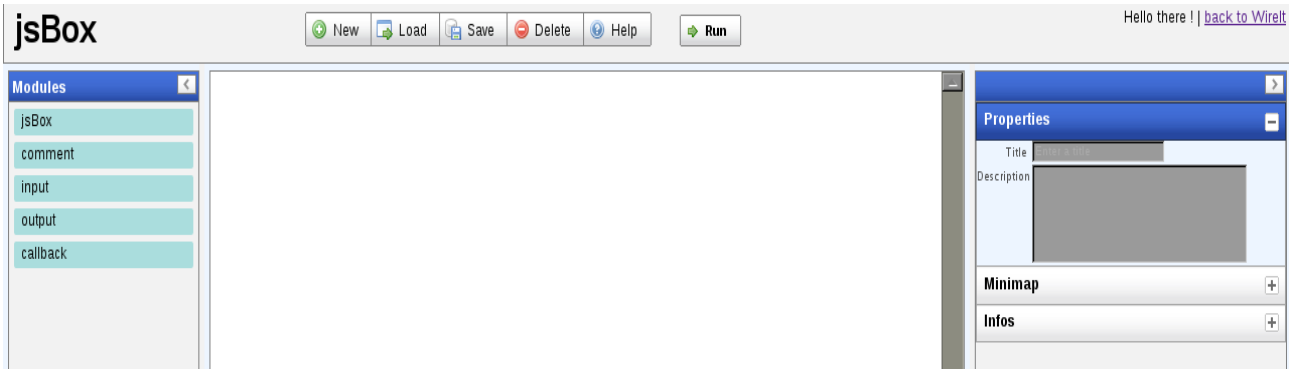
También podemos observar ImageContainer:



Wireit también interfaces más complejas, módulos que integran todos los elementos que vimos en interfaces altamente potentes. Un ejemplo de eso es el WiringEditor:



Un ejemplo del JsBox:



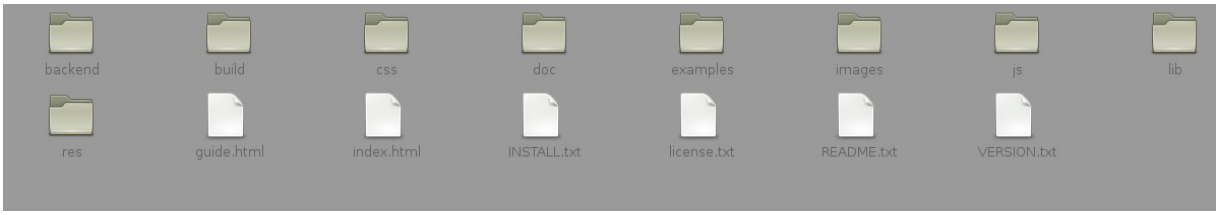
Hasta ahora sólo nos hemos familiarizado con las cosas que se pueden hacer con Wireit, es hora de ponernos a trabajar y aprender un poco de como desarrollar cosas con la librería.

Lo primero será entrar al portal de wireit que se encuentra en: http://neyric.github.com/wireit/

Allí vamos a encontrar información de como descargar la librería, vamos a ver ejemplos de como utilizarla, y vamos a ver una pequeña guía que te da un inicio a lo que es wireit

Para descargar la librería lo hacemos de: https://github.com/neyric/wireit/downloads . La versión estable para efectos de este tutorial es: WireIt-0.5.0.zip

Luego de descargarla procedemos a descomprimirla, allí vamos a encontrar varios archivos y carpetas:



Vamos a explicar un poco los archivos más importantes de la librería que acabamos de descargar. El archivo lib almacenas librería que Wireit necesita para realizar sus trabajos, entre ellas tenemos a YUI, a inputex y canvas.

Dentro de js están los archivos de javaScript que Wireit utiliza para sus ejemplos. También el de

imagenes y css son utilizados por los ejemplos de Wireit.

La carpeta doc tiene el Api de wireit. Si abrimos el archivo index.html de esa carpeta en un navegador web podemos observar el api de todas las clases de Wireit.

La carpeta examples tiene una serie de ejemplos que son muy útiles para aprender Wireit, de hecho, es el mejor amigo que tenemos a la hora de querer hacer algo, ya que el archivo guide.html no es muy conciso como tutorial y deja muchos vacíos en sus explicaciones.

El Api de Wireit lo podemos encontrar en:
http://neyric.github.com/wireit/api/

Ya familiarizados un poco con la estructura de Wireit procedamos a ver una serie de ejemplos que nos permitirán ingresar en el código fuente y cómo funciona esta librería.

Empecemos por la más simple, creemos dos terminales y probemos en el navegador web unirlos a través de un Wire, es decir, hacer algo como esto:

Las librerías a utilizar para realizar esto son:

```
<!-- LIBRERIAS DE YUI -->
<script type="text/javascript" src="../lib/yui/utilities/utilities.js"></script>


<!-- LIBRERIAS DE WIREIT -->
<script type="text/javascript" src="js/WireIt.js"></script>
<script type="text/javascript" src="js/CanvasElement.js"></script>
<script type="text/javascript" src="/js/Wire.js"></script>
<script type="text/javascript" src="js/Terminal.js"></script>
```

NOTA: No hay forma de saber cuales son los archivos que necesitamos utilizar, viendo el código de los ejemplos es que podemos deducir esto.

En el caso de la imagen de arriba podemos observar que hay dos terminales, para crear cada terminal es necesario hacer los div donde estos terminales van a estar, por lo general los terminales están en un contenedor, pero como todavía no tenemos contenedores es por eso que los vamos a posicionar en dos div que haremos nosotros mismos:

```
<div id="elemento1"></div>
<div id="elemento2"></div>
```

Con CSS posicionamos los div en sitios diferentes y que guarden cierta distancia entre ellos. Una vez incluidas las librerías necesarias y creados los div donde irán nuestros contenedores podemos entonces crear los scripts que crearan los contenedores:

```
//OBETENMOS EL DIV DEL TEMINAL1. ESTO LO HACEMOS UTILIZANDO YUI (Si
```

hay dudas con YUI ver el tutorial de YUI)

```
        var t1 = YAHOO.util.Dom.get('terminal1');
       //OBTENEMOS EL SEGUNDO DIV DEL TERMINAL2
        var t2 = YAHOO.util.Dom.get('terminal2');
        //CREAMOS EL PRIMER TERMINAL
        new WireIt.Terminal(t1, {direction: [1,1]});
        //CREAMOS EL SEGUNDO TERMINAL
        new WireIt.Terminal(t2, {direction: [0,1]});
```

El código completo quedaría así:

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
 <head>
  <title>EJEMPLOS DE TERMINAL UTILIZANDO WIREIT.</title>
  <!-- YUI -->
  <script type="text/javascript" src="../lib/yui/utilities/utilities.js"></script>
  <!-- WireIt -->
  <script type="text/javascript" src="../js/WireIt.js"></script>
  <script type="text/javascript" src="../js/CanvasElement.js"></script>
  <script type="text/javascript" src="../js/Wire.js"></script>
  <script type="text/javascript" src="../js/Terminal.js"></script>
  <link rel="stylesheet" type="text/css" href="../css/WireIt.css" />
  <style>
   /*CAMBIAMOS LA POSICIÓN DEL SEGUNDO TERMINAL*/
   #terminal2 {
      position:absolute;
      top:20;
      left:100px;
   }
  </style>
  <script>
//A LA CARGA DEL LA VENTANA LLAMAMOS A LA FUNCIÓN QUE SE HA CREADO ABAJO
   window.onload = function() {
       //OBETENMOS EL DIV DEL TEMINAL1. ESTO LO HACEMOS UTILIZANDO YUI
       var t1 = YAHOO.util.Dom.get('terminal1');
       //OBTENEMOS EL SEGUNDO DIV DEL TERMINAL2
       var t2 = YAHOO.util.Dom.get('terminal2');
       //CREAMOS EL PRIMER TERMINAL
       new WireIt.Terminal(t1, {});
       //CREAMOS EL SEGUNDO TERMINAL
       new WireIt.Terminal(t2, {});
   };
  </script>
 </head>
 <body>
      <div id="terminal1"></div>
      <div id="terminal2"></div>
 </body>
</html>
```

Ahora crearemos un Contenedor de tipo formulario. Los contenedores se encuentran dentro de

layers, los layers son como capas donde se almacenan los contenedores, ya que un layer puede tener uno o mas contenedores. Aquí vamos a hacer uso de los elementos de un formulario, estos elementos trabajan con al librería inputEX.

Para realizar un contenedor lo primero es crear un div dentro de nuestro documento body donde colocaremos nuestro contenedor. Veamos entonces el cuerpo del programa:

```
<body>
   <div id="contenedor"></div>
 </body>
```

Una vez realizado esto necesitamos crear en este orden:

1. Un layer: Este layer lo debemos asociar al div de id contenedor
2. Añadir contenedores al layer: Una vez creado el layer podemos asociarle contenedores
    1. Añadir a los contenedores los campos que van a ir en él. Esto lo hacemos apoyándonos en la librería inputEX

El layer lo creamos de la siguiente forma:

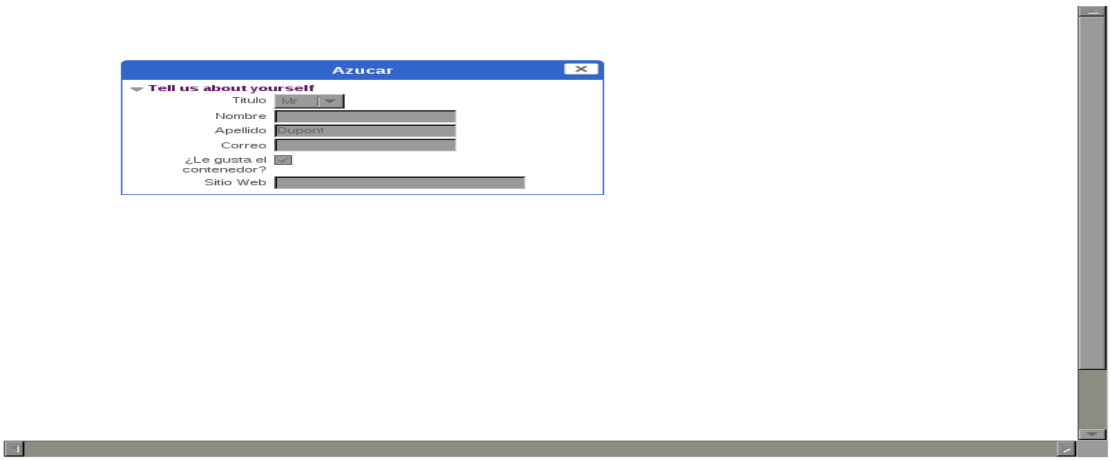
```
//Obtenemos el elemento con id contenedor
elemento = YAHOO.util.Dom.get('contenedor');
//Luego creamos el Layer y lo asociamos al elemento
//Ver Api para especificaciones de parámetros
var demoLayer = new WireIt.Layer({parentEl:elemento,layerMap: false});
```

Una vez creado el Layer procedemos a asociarle al layer los contenedores con sus respectivos terminales:

El método setWiring permite la asociación de contenedores:

```
demoLayer.setWiring({
            //Definimos el contenedor
            containers: [
                // FormContainer (usando inputEx)
                {
                    //Aquí podemos indicar una posición dentro del layer: Pop, Lento
                    position:[8,10],
                    //Indicamos el tipo del contenedor, recordemos que hay varios tipos de contenedores
                    style: "WireIt.FormContainer",
                    //Colocamos el título del contenedor
                    title: "Azúcar",
                    //Definimos los campos que estarán en el contenedor
                    fields: [
                        {type: 'select', inputParams: {label: 'Titulo', id: 'titlulo234',name: 'title', selectValues: ['Mr','Sra','Mme'] } },
                        {inputParams: {label: 'Nombre', name: 'firstname', required: true } },
                        {inputParams: {label: 'Apellido', name: 'lastname', value:'Dupont'} },
                        {type:'email', inputParams: {label: 'Correo', name: 'email', required: true}},
                        {type:'boolean', inputParams: {label: '¿Le gusta el contenedor?', name: 'pregunta'}},
                        {type:'url', inputParams: {label: 'Sitio Web', name:'website', size: 25}}
                     ],
                    //Creamos una leyenda
                    legend: "Tell us about yourself\n"
                }]
});
```

Con este código estaríamos creando algo como esto:



El scroling es el layer, el contenedor lo podemos mover en cualquier posición del layer.