



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

Desarrollo en React JS

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148

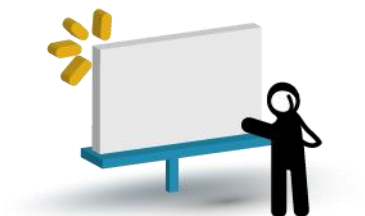
www.sceu.frba.utn.edu.ar/e-learning



Módulo I

Nivelación, Javascript y configuración del entorno

Unidad 2: Javascript ES 6



Presentación:

En esta unidad veremos la sintaxis de javascript, como trabaja con el navegador web y haremos nuestro primer pequeño código.

Qué son los eventos, como capturar y trabajar con estos y como validar formularios.

Algunas particularidades de ECMAScript 6



Objetivos:

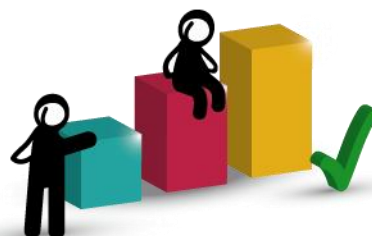
Que los participantes*:

- Conozcan la sintaxis de javascript
- Aprendan cómo vincular el lenguaje con ciertas etiquetas de HTML
- Comprendan nuevas funcionalidades brindadas por ECMAScript 6



Bloques temáticos*:

- Sintaxis
- Variables
- Operadores
- Estructuras
- Funciones – Arrays
- Formularios
- Campos de texto
- Checkbox
- Select
- Eventos
- Seleccionar elemento e imprimir en html
- Definir una constante
- Let
- Función Arrow
- Clases
- Template Strings
- Valores por defecto
- Módulos



Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.



Tomen nota*

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.



Javascript – Sintaxis

Comentarios

```
<script>
//Este es un comentario de una línea
/*Este comentario se puede extender
por varias líneas.
Las que quieras*/
</script>
```

En caso de utilizar archivos JS minificados, se deben utilizar comentarios extendidos y NO de una línea.

* Archivo minificado -> un js minificado es “comprimir” todo el contenido de nuestro js en un archivo de una única línea.

Separación de instrucciones

Javascript tiene dos maneras de separar instrucciones:

- La primera es a través del carácter punto y coma (;)

```
llamadoFuncion();
```

- La segunda es a través de un salto de línea.

```
llamadoFuncion()
```




Javascript – Variables

Variables

Una variable es un espacio en memoria donde se almacena un dato, un espacio donde podemos guardar cualquier tipo de información que necesitemos para realizar las acciones de nuestros programas.

Los nombres de las variables han de construirse con caracteres alfanuméricos y el carácter subrayado (_).

Los nombres tienen que comenzar por un carácter alfabético o el subrayado. **No podemos utilizar caracteres raros como el signo +, un espacio.**

Las variables no pueden utilizar nombres reservados, por ejemplo if, for, while, etc



Declaración de Variables

Declarar variables consiste en definir y de paso informar al sistema de qué vas a utilizar una variable.

Javascript cuenta con la palabra "**var**" que utilizaremos cuando queramos declarar una o varias variables. Como es lógico, se utiliza esa palabra para definir la variable antes de utilizarla.

Ejemplo:

```
var operando1;  
var operando2;  
  
var operando1 = 23;  
var operando2 = 33;  
  
var operando1,operando2;
```

Ejemplo:

```
-----  
<script language="javascript">  
  
    //creo una variable  
    var x;  
    //x actualmente no tiene ningún valor  
    x = 25;  
    //ahora x tiene el valor numérico 25  
  
    //creo una segunda variable  
    var y = 230;  
    //he creado una variable y asignado un valor en un solo paso!!  
  
    //las variables guardan datos con los que puedo realizar operaciones  
    var suma;  
    suma = x + y;  
    //he creado la variable suma y he asignado la suma de x e y  
    alert(suma);  
    //he mostrado el valor de la variable suma  
  
</script>
```



Nota: Si nosotros no declaramos el **modo estricto**, podremos utilizar variables sin necesidad de declararlas con la palabra reservada **var**.

Para declarar el **strict mode** debemos anteponer al código “**strict mode**” por ejemplo:

```
v = 15
function f1() {
  "use strict";
  var v = "Hi! I'm a strict mode script!";
}
```

En este caso la primer asignación se hace sin modo estricto por lo cual como vemos no tiene la palabra var adelante.

En cambio dentro de la función f1 se hace en modo estricto por lo cual debe tener la palabra var delante.



Ámbito de las Variables

Se le llama **ámbito de las variables** al lugar donde estas están disponibles. Por lo general, cuando declaramos una variable hacemos que esté disponible en el lugar donde se ha declarado.

- Variables globales: son las que están declaradas en el ámbito más amplio posible.

```
<script>  
var variableGlobal  
</script>
```

- Variables locales: sólo podremos acceder a ellas dentro del lugar donde se ha declarado

```
<script>  
function miFuncion () {  
    var variableLocal  
}  
</script>
```



Javascript – Operadores

Operadores de cadenas

Las cadenas de caracteres, o variables de texto, también tienen sus propios operadores para realizar acciones típicas sobre cadenas. Aunque javascript sólo tiene un operador para cadenas se pueden realizar otras acciones con una serie de funciones predefinidas en el lenguaje que veremos más adelante.

Para concatenar 2 cadenas de texto debemos usar el operador “+”

```
<script>
cadenaConcatenada = cadena1 + cadena2 /
</script>
```

Operadores lógicos

Estos operadores sirven para realizar operaciones lógicas, que son aquellas que dan como resultado un verdadero o un falso, y se utilizan para tomar decisiones en nuestros scripts.

```
<script>

prueba && prueba2 //Operador and

prueba || prueba2 //Operador or

</script>
```

Typeof

Para comprobar el tipo de un dato se puede utilizar otro operador que está disponible a partir de javascript 1.1, el **operador typeof**, que devuelve una cadena de texto que describe el tipo del operador que estamos comprobando.

```
<script>

document.write("<br>El tipo de booleano es: " + typeof booleano)

</script>
El tipo de booleano es: boolean
```



Operadores de asignación

Un operador de asignación asigna un valor al operando de la izquierda en función del valor del operando de la derecha. El operador básico de asignación es el de igual (=), que asigna el valor del operando de la derecha al operando de la izquierda. Por ejemplo, $x = y$, está asignando el valor de y a x .

Nombre	Operador abreviado	Significado
Operadores de asignación	$x = y$	$x = y$
Asignación de adición	$x += y$	$x = x + y$
Asignación de sustracción	$x -= y$	$x = x - y$
Asignación de multiplicación	$x *= y$	$x = x * y$
Asignación de división	$x /= y$	$x = x / y$
Asignación de resto	$x \% = y$	$x = x \% y$
Asignación de exponenciación	$x ** = y$	$x = x ** y$
Asignación de desplazamiento a la izquierda	$x << = y$	$x = x << y$
Asignación de desplazamiento a la derecha	$x >> = y$	$x = x >> y$
Asignación de desplazamiento a la derecha sin signo	$x >>> = y$	$x = x >>> y$
Asignación AND binaria	$x \& = y$	$x = x \& y$
Asignación XOR binaria	$x \wedge = y$	$x = x \wedge y$
Asignación OR binaria	$x = y$	$x = x y$



Operadores de comparación

Un operador de comparación compara sus operandos y devuelve un valor lógico en función de si la comparación es verdadera (true) o falsa (false). Los operadores pueden ser numéricos, de cadena de caracteres (Strings), lógicos o de objetos.


Operador	Descripción	Ejemplos devolviendo true
Igualdad (==)	Devuelve true si ambos operandos son iguales.	<code>3 == var1</code> <code>"3" == var1</code> <code>3 == "3"</code>
Desigualdad (!=)	Devuelve true si ambos operandos no son iguales.	<code>var1 != 4</code> <code>var2 != "3"</code>
Estrictamente iguales (===)	Devuelve true si los operandos son igual y tienen el mismo tipo. Mira también <code>Object.is</code> y <code>sameness in JS</code> .	<code>3 === var1</code>
Estrictamente desiguales (!==)	Devuelve true si los operandos no son iguales y/o no son del mismo tipo.	<code>var1 !== "3"</code> <code>3 !== "3"</code>
Mayor que (>)	Devuelve true si el operando de la izquierda es mayor que el operando de la derecha.	<code>var2 > var1</code> <code>"12" > 2</code>
Mayor o igual que (>=)	Devuelve true si el operando de la izquierda es mayor o igual que el operando de la derecha.	<code>var2 >= var1</code> <code>var1 >= 3</code>
Menor que (<)	Devuelve true si el operando de la izquierda es menor que el operando de la derecha.	<code>var1 < var2</code> <code>"2" < 12</code>
Menor o igual que (<=)	Devuelve true si el operando de la izquierda es menor o igual que el operando de la derecha.	<code>var1 <= var2</code> <code>var2 <= 5</code>



Operadores aritméticos

Los operadores aritméticos toman los valores numéricos (tanto literales como variables) de sus operandos y devuelven un único resultado numérico. Los operadores aritméticos estándar son la suma (+), la resta (-), la multiplicación (*) y la división (/).

Tabla 3.3 Operadores aritméticos

Operador	Descripción	Ejemplo
Resto (%)	Operador binario correspondiente al módulo de una operación. Devuelve el resto de la división de dos operandos.	12 % 5 devuelve 2.
Incremento (++)	Operador unario. Incrementa en una unidad al operando. Si es usado antes del operando (++x) devuelve el valor del operando después de añadirle 1 y si se usa después del operando (x++) devuelve el valor de este antes de añadirle 1.	Si x es 3, entonces ++x establece x a 4 y devuelve 4, mientras que x++ devuelve 3 y, solo después de devolver el valor, establece x a 4.
Decremento (--)	Operador unario. Resta una unidad al operando. Dependiendo de la posición con respecto al operando tiene el mismo comportamiento que el operador de incremento.	Si x es 3, entonces --x establece x a 2 y devuelve 2, mientras que x-- devuelve 3 y, solo después de devolver el valor, establece x a 2.
Negación Unaria (-)	Operación unaria. Intenta convertir a número al operando y devuelve su forma negativa.	- "3" devuelve -3. - true devuelve -1.
Unario positivo (+)	Operación unaria. Intenta convertir a número al operando.	+ "3" devuelve 3. + true devuelve 1.
Exponenciación (**) 	Calcula la potencia de la base al valor del exponente. Es equivalente a $\text{base}^{\text{exponente}}$	2 ** 3 devuelve 8. 10 ** -1 devuelve 0.1.



Javascript – Estructuras

IF

La estructura if nos permitirá mediante el uso de operadores de comparación tomar una decisión a partir de si dicha comparación o valor de una expresión es verdadera o falsa (en este caso entra por el else)

```
<script>

if (expresión) { |
    //acciones a realizar en caso positivo
    //...
} else {
    //acciones a realizar en caso negativo
    //...
}

</script>
```

For

La estructura for nos permitirá recorrer un array en javascript.

Cada elemento recorrido se reconoce como una iteración, se “cortara” el for cuando la condición sea falsa.

```
<script>
for (inicialización; condición; actualización) {
    //sentencias a ejecutar en cada iteración
}
</script>
```

- Inicialización: Aquí inicializamos las variables. Por ejemplo: **i=0**
- Condición: Mientras la condición sea verdadera se seguirán produciendo iteraciones. Ejemplo **i<5**
- Actualización: se actualiza la variable de la condición. Ejemplo: **i++**



While

La estructura while es similar a la for, nos permite realizar iteraciones sobre un array

```
<script>
while (condición){
    //sentencias a ejecutar
}
</script>
```

Como vemos no cuenta con el elemento de inicialización ni el de actualización.



Javascript – Funciones - Arrays

Definición

Primero se escribe la palabra **function**, reservada para este uso. Seguidamente se escribe el nombre de la función, que como los nombres de variables puede tener números, letras y algún carácter adicional como en guión bajo.

```
<script>
function nombrefuncion () {
    instrucciones de la función
    ... |
}
</script>
```

Entre paréntesis, luego del nombre de la función, se definirán los parámetros que recibirá la misma. En el caso de js no hace falta el tipo de dato en cada parámetro recibido.

Arrays

El primer paso para utilizar un array es crearlo. Para ello utilizamos un objeto Javascript ya implementado en el navegador.

```
<script>
//Array vacio
var miArray = new Array()
//Creacion de array con numero de compartimentos
var miArray = new Array(10)
//Setear datos
miArray[0] = 290
miArray[1] = 97
miArray[2] = 127
//Declaracion e inicializacion
var arrayRapido = [12,45,"array inicializado en su declaración"]
</script>
```

A diferencia de PHP no podemos tener vectores asociativos, es decir arrays cuya clave sea un string.



Longitud de un array

Para obtener la longitud de un array utilizaremos el método **length**

```
<script>  
document.write("Longitud del array: " + miArray.length)  
</script>
```



Javascript – Formularios

Acceder a formularios con document

```
> document.form_registrar  
< ▶ <form method="POST" name="form_registrar" class="regForm">...</form>
```

Acceder al formulario a través del objeto document y el nombre del formulario.

```
> document.forms[0]  
< ▶ <form method="POST" name="form_registrar" class="regForm">...</form>
```

Acceder al formulario a través del objeto document y el índice del formulario.

Acceder a campos dentro del formulario

```
> document.form_registrar.nombre  
< ▶ <input name="nombre" type="text" class="form-control" aria-required="true">
```

Acceder al valor del campo

```
document.form_registrar.nombre.value  
" "
```



Evento onclick

Con este evento definimos una acción (función) a ejecutar cuando se hace click en un elemento (un botón)

```
<input type="Button" name="enviar" value=" enviar " onclick="enviar()" >
```

Propiedades

- **Action:** Es la acción que queremos realizar cuando se submite un formulario.
- **Encoding:** El tipo de codificación del formulario
- **Length:** El número de campos del formulario.
- **Method:** El método por el que mandamos la información. Corresponde con el atributo METHOD del formulario.
- **Name:** El nombre del formulario, que corresponde con el atributo NAME del formulario.
- **Target:** La ventana o frame en la que está dirigido el formulario. Cuando se submite se actualizará la ventana o frame indicado. Corresponde con el atributo target del formulario.

Ejemplo de propiedad method:

```
document.form_registrar.method  
"post"
```

Métodos

- **submit():** Para hacer que el formulario se submite, aunque no se haya pulsado el botón de submit.
- **reset():** Para reiniciar todos los campos del formulario, como si se hubiese pulsado el botón de reset.



Javascript – Campos de texto

Propiedades

- **defaultValue:** Es el valor por defecto que tiene un campo. Lo que contiene el atributo VALUE de la etiqueta <INPUT>.
- **Form:** Hace referencia al formulario.
- **Name:** Contiene el nombre de este campo de formulario
- **Type:** Contiene el tipo de campo de formulario que es.
- **Value:** El texto que hay escrito en el campo.

Métodos

- **blur():** Retira el foco de la aplicación del campo de texto.
- **focus():** Pone el foco de la aplicación en el campo de texto.
- **select():** Selecciona el texto del campo.

Javascript – Checkbox

Propiedades

- **checked:** Informa sobre el estado del checkbox. Puede ser true o false.
- **defaultChecked:** Si está chequeada por defecto o no.
- **Value:** El valor actual del checkbox.

Métodos

- **click():** Es como si hiciésemos un click sobre el checkbox, es decir, cambia el estado del checkbox.
- **blur():** Retira el foco de la aplicación del checkbox.
- **focus():** Coloca el foco de la aplicación en el checkbox.



Javascript – Select

Propiedades

- **length**: Guarda la cantidad de opciones del campo select. Cantidad de etiquetas <OPTION>
- **Option**: Hace referencia a cada una de sus opciones. Son por sí mismas objetos.
- **Options**: Un array con cada una de las opciones del select.
- **selectedIndex**: Es el índice de la opción que se encuentra seleccionada.

Métodos

- **blur()**: Para retirar el foco de la aplicación de ese elemento de formulario.
- **focus()**: Para poner el foco de la aplicación.
- **Objeto option**:
 - **defaultSelected**
Indica con un true o un false si esa opción es la opción por defecto.
 - **index**
El índice de esa opción dentro del select.
 - **selected**
Indica si esa opción se encuentra seleccionada o no.
 - **text**
Es el texto de la opción. Lo que puede ver el usuario en el select, que se escribe después de la etiqueta <OPTION>.
 - **value**
Indica el valor de la opción, que se introduce con el atributo VALUE de la etiqueta <OPTION>.



Javascript – Eventos

Manejadores

- **onblur**

Se desata un evento onblur cuando un elemento pierde el foco de la aplicación. El foco de la aplicación es el lugar donde está situado el cursor, por ejemplo puede estar situado sobre un campo de texto, una página, un botón o cualquier otro elemento.

- **onchange**

Se desata este evento cuando cambia el estado de un elemento de formulario, en ocasiones no se produce hasta que el usuario retira el foco de la aplicación del elemento.

- **onclick**

Se produce cuando se da una pulsación o clic al botón del ratón sobre un elemento de la página, generalmente un botón o un enlace.

- **onload**

Este evento se desata cuando la página, o en Javascript 1.1 las imágenes, ha terminado de cargarse.

- **onsubmit**

Ocurre cuando el visitante apreta sobre el botón de enviar el formulario. Se ejecuta antes del envío propiamente dicho.

Para ver más manejadores: <http://www.desarrolloweb.com/articulos/1236.php>

Ejemplo de on blur:

```
</head>
<body>
<form name=f1>
Escriba un número entero: <input type=text name=numero size=8 value="" onblur="compruebaValidoEntero()" >
</form>

</body>
</html>
```



Javascript – Seleccionar elemento e imprimir en html

getElementById

Permite, como su nombre indica, seleccionar un elemento del documento por medio del valor del atributo id que se le haya asignado

```
2
3 <script>
4     document.getElementById('id_del_elemento');
5 </script>
```

InnerHTML

Retorna o setea el contenido HTML de un elemento

- Set

```
document.getElementById("alertas").innerHTML += "El campo nombre es obligatorio<br/>";
```

- Get

```
document.getElementById("alertas").innerHTML;
```



Javascript – Definir una constante

En javascript ES 6 podemos definir una constante con la palabra reservada **const**

```
const PI = 3.141593;  
alert(PI > 3.0);
```

Javascript – Let

Podemos utilizar **let** en lugar de **var** a la hora de declarar una variable, cuando queremos que esta solo sea accedida de manera local en determinado ámbito. Por ejemplo:

```
//ES5  
(function() {  
  console.log(x); // x no está definida aún.  
  if(true) {  
    var x = "hola mundo";  
  }  
  console.log(x);  
  // Imprime "hola mundo", porque "var" hace que sea global  
  // a la función;  
})();  
  
//ES6  
(function() {  
  if(true) {  
    let x = "hola mundo";  
  }  
  console.log(x);  
  //Da error, porque "x" ha sido definida dentro del "if"  
})();
```



Javascript – Función Arrow

```
var miFuncion = function(num) {  
    return num + num;  
}  
//ES6  
var miFuncion = (num) => num + num;
```

```
//ES6  
var data = [{...}, {...}, {...}, ...];  
data.forEach(elem => {  
    console.log(elem);  
});
```

En ambos ejemplos se puede que se sustituye el uso de la palabra reservada **function** dando simplicidad en el código.

(parametro1,parametro2,...,parámetro n) =>{Definición de la función}



Javascript – Clases

Ahora JavaScript tendrá clases, muy parecidas las funciones constructoras de objetos que realizamos en el estándar anterior, pero ahora bajo el paradigma de clases, con todo lo que eso conlleva, como por ejemplo, herencia.

```
class LibroTecnico extends Libro {  
  constructor(tematica, paginas) {  
    super(tematica, paginas);  
    this.capitulos = [];  
    this.precio = "";  
    // ...  
  }  
  metodo() {  
    // ...  
  }  
}
```



This

La variable `this` muchas veces se vuelve un dolor de cabeza. antiguamente teníamos que cachearlo en otra variable ya que solo hace referencia al contexto en el que nos encontremos. Por ejemplo, en el siguiente código si no hacemos `var that = this` dentro de la función `document.addEventListener`, `this` haría referencia a la función que pasamos por Callback y no podríamos llamar a `foo()`

```
//ES3
var obj = {
  foo : function() {...},
  bar : function() {
    var that = this;
    document.addEventListener("click", function(e) {
      that.foo();
    });
  }
}
```

Con ECMAScript5 la cosa cambió un poco, y gracias al método `bind` podíamos indicarle que `this` hace referencia a un contexto y no a otro.

```
//ES5
var obj = {
  foo : function() {...},
  bar : function() {
    document.addEventListener("click", function(e) {
      this.foo();
    }).bind(this);
  }
}
```



Ahora con ES6 y la función Arrow => la cosa es todavía más visual y sencilla.

```
//ES6
var obj = {
  foo : function() {...},
  bar : function() {
    document.addEventListener("click", (e) => this.foo());
  }
}
```

Javascript – Template Strings

Con ES6 podemos interpolar Strings de una forma más sencilla que como estábamos haciendo hasta ahora. Fíjate en este ejemplo:

```
//ES6
let nombre1 = "JavaScript";
let nombre2 = "awesome";
console.log(`Sólo quiero decir que ${nombre1} is ${nombre2}`);
// Solo quiero decir que JavaScript is awesome
```

También podemos tener String multilínea sin necesidad de concatenarlos con +

```
//ES5
var saludo = "ola " +
"que " +
"ase ";

//ES6
var saludo = "ola
que
ase";

console.log("hola
que
ase");
```



Javascript – Valores por defecto

Otra novedad es asignar valores por defecto a las variables que se pasan por parámetros en las funciones. Antes teníamos que comprobar si la variable ya tenía un valor. Ahora con ES6 se la podemos asignar según creemos la función.

```
//ES5
function(valor) {
  valor = valor || "foo";
}

//ES6
function(valor = "foo") {...};
```

Javascript – Módulos

Ahora JavaScript se empieza a parecer a lenguajes como Python o Ruby. Llamamos a las funciones desde los propios Scripts, sin tener que importarlos en el HTML, si usamos JavaScript en el navegador.

```
//File: lib/person.js
module "person" {
  export function hello(nombre) {
    return nombre;
  }
}
```

Y para importar en otro fichero:

```
//File: app.js
import { hello } from "person";
var app = {
  foo: function() {
    hello("Carlos");
  }
}
export app;
```




UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 33

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning



Bibliografía y Webgrafía utilizada y sugerida

Fedosejev, A. (2015). React.js Essentials (1 ed.). EEUU, Packt.

Amler, . (2016). ReactJS by Example (1 ed.). EEUU, Packt.

Stein, J. (2016). ReactJS Cookbook (1 ed.). EEUU, Packt.

https://www.tutorialspoint.com/es6/es6_syntax.htm

<http://www.desarrolloweb.com/manuales/20/>

<http://www.desarrolloweb.com/articulos/826.php>

<http://www.desarrolloweb.com/articulos/827.php>

<http://www.desarrolloweb.com/articulos/846.php>

<http://www.desarrolloweb.com/articulos/861.php>

<https://carlosazaustre.es/ecmascript-6-el-nuevo-estandar-de-javascript/>



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

**Centro de
e-Learning**

p. 35

Centro de e-Learning SCEU UTN - BA.

Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning



Lo que vimos:

En esta unidad aprendimos la sintaxis básica de javascript y las particularidades a la hora de trabajar con ECMAScript 6



Lo que viene:

En la próxima unidad aprendemos más sobre react js, cómo crear una aplicación con react, que debemos instalar en nuestro entorno de trabajo y así poder comenzar a desarrollar en dicho framework.

