JavaScript

JavaScript es un lenguaje de programación utilizado en desarrollos Web. Junto con **HTML** y **CSS**, se utiliza para el despliegue de soluciones **Frontend**. Sin embargo, el lenguaje ha adquirido otra dimensión en los últimos años, siendo actualmente empleado también en soluciones **Backend**, gracias a **Node.js**. Con esto, JavaScript se ha convertido en un lenguaje de propósito general.

Comenzamos nuestra aventura de aprendizaje de este lenguaje de programación, explorando las **nociones básicas** del mismo, que incluye los detalles relacionados a las definiciones de variables, tipos de datos simples y complejos, estructuras de control y declaración de funciones.

Los desarrollos basados en JavaScript usualmente envuelven el uso de múltiples herramientas, librerías y lenguajes de programación, tales como **HTML, Ajax, JQuery** y **Canvas**. Todos ellos deben ser usados de una forma integrada para lograr la solución deseada. Esta segunda mitad del camino de aprendizaje, se concentra en proporcionar el abanico de opciones que existen para trabajar en los desarrollos Web, con especial énfasis en el rol de JavaScript en dicho desarrollos, que se materializa en la forma de cómo integramos este código en nuestra solución.

La última parte de nuestra experiencia de aprendizaje es destinada a estudiar ciertos componentes más complejos del lenguaje de programación, tal como las **funciones asíncronas**, las **peticiones** a los servidores usando **Ajax**, la incorporación de **interactividad** a nuestras páginas usando **JQuery** y la elaboración de dibujos usando el elemento html, **Canvas**.

Finalizamos, revisando ciertas técnicas y mecanismos que podemos utilizar una vez que hayamos concluido nuestro programa y estemos listos para enviarlo al ambiente de producción. Es así como, exploramos las técnicas de **ofuscamiento** y **comprensión** y las **Redes de Distribución de Contenido**.

Luego de haber culminado exitosamente el curso de JavaScript has desarrollado las capacidades siguientes:

- Reconocer los fundamentos de JavaScript para la construcción de aplicaciones Web.
- Identificar características extendidas de JavaScript para el desarrollo de soluciones Web orientadas al Front-end.
- Aplicar técnicas de comunicación asíncrona para la interacción del usuario con la lógica de negocio programada en el Front-end.

- Desarrollar aplicaciones que implementan estructuras de datos y manejo avanzado de funciones JavaScript para la construcción de aplicaciones Web, que soporten el transporte de datos estructurados.
- Resumir los mecanismos y técnicas para la puesta en producción de los desarrollos Web Front-end con la finalidad de obtener un mejor rendimiento de nuestros sitios Web.

Ahora estás listo para desarrollar tus propias soluciones JavaScript en ambientes Front-end. Pero primero, te dejamos un resumen de lo que hemos aprendido sobre este lenguaje de programación.

UNIDAD 1: JavaScript-Conceptos básicos

JavaScript es un lenguaje de programación ampliamente utilizado para proporcionar **dinamismo e interactividad** a las páginas Web. Es un lenguaje de **alto nivel interpretado dinámico**, que significa que no requiere de un compilador para traducir sus instrucciones, que a su vez se generan en tiempo de ejecución. JavaScript es además un lenguaje que se considera **débilmente tipado** ya que sus variables se asignan a medida que se ejecuta el programa. Adicionalmente, el lenguaje incorpora aspectos de múltiples paradigmas de programación, tales como **orientación a objetos** y **lenguajes funcionales**, considerándose así **multiparadigma**.

JavaScript se usa como un lenguaje de programación para las aplicaciones corriendo en el navegador Web, por lo tanto, es una **tecnología Frontend**. Sin embargo, su uso se ha extendido a las aplicaciones que corren del lado del servidor o **Backend**, a través del uso de **Node.js**, que es una extensión del lenguaje para convertirlo en un **lenguaje de propósito general**, cuyos programas pueden ejecutarse en cualquier ambiente.

JavaScript, similarmente a otros lenguajes de programación, soporta **tipos de datos** simples, tales como, los strings y enteros, así como también **tipos más complejos**, tales como, los arreglos y el tipo de dato objeto. Los tipos de datos definen la clase de contenido que tienen las variables y por ende la clase de operaciones que pueden realizarse con ellas. Igualmente encontramos una serie de **estructuras de control** que incorporadas en nuestro código permiten definir lo que queremos que nuestro código realice. Ellas se clasifican en condicionales, tales como, **if/else** y ciclos, tales como, **for y while**.

Por otro lado, tenemos las **funciones** que nos permiten la construcción modular de nuestro programa y la reutilización de parte del código que se encuentra definido dentro de dichas funciones. Relacionado con las funciones está el **alcance de las variables** en el ámbito de un programa, el cual se determina desde las definiciones más internas hacia las más externas. Así, una variable puede tener un **ámbito global**, es decir, todo el programa o **local**, tal como en determinada función.

UNIDAD 2: Explorando JavaScript

Esta unidad, “Explorando JavaScript”, nos lleva a examinar estructuras y funciones más complejas en el lenguaje. Comenzamos estudiando un tipo de datos bastante particular denominado **objeto**, que es similar al tipo de dato registro en otros lenguajes de programación y que no debe confundirse con los objetos en el paradigma orientado a objetos. El tipo de dato objeto proporciona **gran flexibilidad** para definir estructuras con los campos necesarios, lo cual subsana el problema de la rigidez de otras estructuras como los arreglos.

Seguidamente, en esta unidad, analizamos el **paradigma orientado a objetos**, donde el sistema que estamos implementando se representa como un conjunto de objetos interactuando entre sí. Cuando usamos este paradigma, debemos aplicar el concepto de **abstracción**, para tratar de identificar en un problema dado aquellos elementos que son relevantes para su solución. Los conceptos fundamentales en la orientación a objetos son: las **clases**, que se representan como los tipos; los **objetos**, que son instancias de las clases; los **atributos**, que son propiedades de los objetos; y los **métodos**, que son funciones que determinan el comportamiento los objetos.

En JavaScript, una clase se define usando la palabra reservada **class**. Dentro de la clase el método **constructor** es de particular importancia porque es el que inicializa el objeto. Dentro de la clase también se encapsulan las **propiedades** del objeto. A través del uso de la palabra reservada **new**, se puede crear una instancia de la clase, es decir un **objeto**.

Seguimos profundizando en JavaScript, explorando las **expresiones regulares,** que son una forma de definir un patrón para ver si coincide con una cadena de caracteres dada y se utiliza comúnmente para procesar strings. También, revisamos cómo trabajar con la clase **Fecha**, que nos proporciona una amplio abanico de métodos para operar sobre las mismas.

Para finalizar, estudiamos el paso de funciones como argumentos en otras funciones. que es particularmente útil cuando trabajamos con el procesamiento de los elementos de un arreglo. Tenemos tres funciones útiles cuando trabajamos con arreglos, una de **filtrado** (filter), que permite seleccionar los elementos de un arreglo basados en lo que definamos en una función determinada; la otra es el **mapa** (map), que transforma la estructura del arreglo de acuerdo a lo que le especifiquemos; y el **encontrar** (find), que permite buscar un elemento determinado en un arreglo.

UNIDAD 3: JavaScript en el Navegador

Una vez que hemos examinado los componentes estructurales de JavaScript, avanzamos hacia la integración del código escrito en este lenguaje, también conocido como **javascript**, en nuestro navegador, de forma tal de incorporar dinamismo e interactividad a nuestras páginas.

Comenzamos esta unidad, explicando cómo se integran los javascripts usando las etiquetas html **script**. Dichos scripts pueden estar localizados en cualquier parte de nuestro código html, pero debemos tomar en cuenta que es importante que se carguen los elementos que dibujan una página antes del código que define su comportamiento, que es lo que se hace con el código JavaScript. Por otra parte, hoy en día los navegadores incorporan características para **depurar** un programa, es decir, estas facilidades permiten que los programadores vean los valores de variables o el estado de su aplicación en determinado momento.

Tradicionalmente **JavaScript** ha sido utilizado para **validar formularios**. Las validaciones se establecen con la ayuda de lo siguiente: las **etiquetas html** de **form**, el uso de **objetos,** tales como, los **botone**s, la definición de **eventos** asociados a estos objetos, tales como **onclick**, y los **javascripts** que implementan las reglas de validación.

El **Modelo de Objeto del Documento o DOM** (Document Object Model) es una **representación jerárquica** (o en forma de árbol) de los elementos de un **documento html**. JavaScript incorpora instrucciones para que obtengamos o quitemos elementos al DOM actual, haciendo nuestras páginas más dinámicas. En este mismo orden de ideas, tenemos el **objeto global window** soportado por todos los navegadores, el cual nos permite acceder sus diversos elementos, tales como la pantalla, la localización, el portapapeles y el documento. Este último incluye a su vez otros métodos y atributos, como por ejemplo, imágenes y enlaces.

Las **Web APIs** son un conjunto de objetos y funciones que se pueden llamar desde el navegador con JavaScript. Ellas nos permiten manejar los **objetos en el navegador**, tales como, la **localización** y la **pantalla**. Existen muchas APIs, pero una de ellas es la de **Web Storage** utilizada para guardar información del usuario, en la forma de **Local Storage** donde podemos guardar información local y **Session Storage** para guardar información durante la sesión del usuario.

Un **evento** se define como algo que el usuario o el navegador pueden hacer, y ante los cuales podemos hacer que un código JavaScript reaccione. Un ejemplo de evento es el **onclick**, que sucede cuando el usuario cliquea un botón. Adicionalmente, tenemos los métodos **settimeout** y **setinterval** que se pueden utilizar para planificar eventos a futuro dado un intervalo de tiempo específico.

UNIDAD 4: Avanzando con JavaScript

En esta unidad comenzamos destacando la importancia de **ECMAScript 6 (ES6)** como estándar para JavaScript. Sin embargo, este estándar se publicó mucho después que JavaScript surgiera, por lo cual ya muchas soluciones utilizaban versiones no estandarizadas del lenguaje o versiones previas a **ECMAScript 6**. Es por tal razón que surgen los **transpiladores**, que convierte código ES6 a código JavaScript compatible. Una de estas herramientas es **Babel**. Esto significa que si estamos siguiendo ES6, debemos usar una librería, tal como Babel, para que traduzca nuestro código, permitiendo que nuestras páginas se comporten apropiadamente en cualquier navegador.

Iniciamos esta última parte de nuestra travesía por JavaScript estudiando las formas de implementar **funciones asíncronas** usando **promesas**. Recordemos que en el caso de las funciones asíncronas, el intérprete no necesita esperar porque la función retorne un valor, entonces la promesa se puede cumplir (es decir estar disponible) ahora, en el futuro, o nunca. La función tiene dos funciones argumentos: **resolve** (resolver) y **reject** (rechazar), que resuelven o rechazan la promesa, respectivamente. Cuando tenemos muchas funciones asíncronas anidadas usando las promesas, esto puede resultar engorroso a la vista del desarrollador. Para simplificar este proceso, tenemos el **Await** y el **Async**, que simplifican el proceso de trabajar con promesas que están encadenadas.

Ajax es una técnica que nos permite realizar peticiones al servidor para que nos devuelva ciertos datos, que emplearemos para dibujar algunas porciones de nuestra **página html**. Junto con las peticiones Ajax, podemos usar **JQuery** que es una librería de JavaScript que simplifica la forma de sumar interactividad y dinamismo a las páginas.

Canvas es un elemento HTML que se emplea para dibujar gráficos usando JavaScript. Ella permite expandir el uso de este lenguaje para incorporar otras funcionalidades a nuestra página Web, basadas en el uso de animaciones, videos e imágenes.

Finalizamos esta parte de nuestro aprendizaje estudiando ciertas técnicas y mecanismos para poner nuestro código en ambiente de producción. Con las técnica de **compresión** es posible **minimizar** el código removiendo data innecesaria, tal como los espacios. Otra técnica es el **ofuscamiento** que convierte nuestro código en algo que es difícil de leer por alguien que acceda el mismo y consiste en cambiar el nombre de las variables y funciones. Una vez que el código esté en producción, se puede elegir usar las **Redes de Distribución de Contenidos o CDNs**, que permiten crear réplicas de nuestros archivos en varios servidores ubicados en diferentes localizaciones, ofreciendo grandes ventajas tales como el incremento de la velocidad de acceso al sitio, incremento de la tolerancia a fallas, incremento de la robustez e incremento de la seguridad.