Programa o módulo de formación:	**Programación de software**	
Competencia	Interpretar la información técnica de diseño para la codificación del software.	
RAP	Codificar El Software, Utilizando El Lenguaje De Programación Y La Plataforma Seleccionada	
Fase proyecto	Fase III	
Actividad proyecto	Desarrollar solución de software	
Actividad Técnica de Aprendizaje:	• Conociendo generalidades de JavaScript • Funciones, DOM, Acceso al DOM	
Nombre del taller	Conociendo y aplicando JavaScript	Guía JavaScript
Objetivo del taller:	**Conocer acerca de JavaScript, el DOM, las funciones, tipos de funciones, métodos del DOM**	

Actividad.

1) Realizar las siguientes consultas acerca de JavaScript

- ¿Qué es el Scope y el alcance de las variables (globales, locales, en bloque)
- ¿Qué es una función en JavaScript?
- Tipos de funciones y maneras de declararlas, de un ejemplo de cada una
- ¿Qué es el DOM?

- Completar la tabla con las definiciones de cada tipo de dato del DOM

Tipo de datos DOM	**Definición**
Document	El nodo document es el nodo raíz, a partir del cual derivan el resto de los nodos.
Element	Son los nodos definidos por etiquetas HTML. Por ejemplo una etiqueta div genera un nodo. Si dentro de ese div tenemos tres etiquetas p, dichas etiquetas definen nodos hijos de la etiqueta div.
Nodelist	Los objetos NodeList son colecciones de nodos como los devueltos por propiedades como Node. childNodes y el método document. querySelectorAll ()
Attribute	Los atributos de las etiquetas definen nodos, aunque trabajando con JavaScript no los veremos cómo nodos, sino que lo consideramos información asociada al nodo de tipo element.

- ¿Cuáles son los métodos a través de los cuales puedo acceder al DOM?, indique 2 ejemplos
- ¿Cuáles con los métodos a través de los cuales puedo crear elementos en el DOM?, indique 2 ejemplos
- ¿Cuáles son los métodos para modificar atributos en HTML?, indique 2 ejemplos

2) Realice una infografía con la información obtenida hasta el momento de JavaScript y socializa con tu grupo.

Material de apoyo:
DOM:
https://www.youtube.com/watch?v=1YzjLETzm4M&ab_channel=Granosdecaf%C3%A9
Funciones:
https://www.youtube.com/watch?v=H6U1Pm7x60E&t=172s&ab_channel=jonmircha
Infografía: https://app.genial.ly/templates/infographics

Ambiente de Formación: virtual
Materiales o recursos: - equipo con conexión a internet
Duración: 4 horas

SOLUCIÓN.

1.¿Qué es el Scope?

El Scope de una variable hace referencia al lugar donde esta va a vivir, o podrá ser accesible.

Podríamos decir también que Scope es el alcance que determina la accesibilidad de las variables en cada parte de nuestro código.

Ejemplos de Scope en variables.

Global Scope.

Se dice que una variable se encuentra en el scope global cuando está declarada fuera de una función o de un bloque. Vamos a poder acceder a este tipo de variables desde cualquier parte de nuestro código, ya sea dentro o fuera de una función.

Local Scope.

Las variables que definimos dentro de una función son variables locales, es decir se encuentran en el Scope local. Esto significa que este tipo de variables van a vivir únicamente dentro de la función en donde las hayamos declarado y si intentamos accederlas fuera de ella, dichas variables no van a estar definidas.

Block Scope.

A diferencia del scope local este scope está limitado al bloque de código donde fue definida la variable. Desde ECMAScript 6 contamos con los keyword let y const los cuales nos permiten tener un scope de bloque, esto quiere decir que las variables solo van a vivir dentro del bloque de código correspondiente.

2.¿Qué es una función en JavaScript?

Una función es una porción de código reusable que puede ser llamada en cualquier momento desde nuestro programa. Una función en JavaScript es similar a un procedimiento — un conjunto de instrucciones que realiza una tarea o calcula un valor, pero para que un procedimiento califique como función, debe tomar alguna entrada y devolver una salida donde hay alguna relación obvia entre la entrada y la salida.

3.Tipos de funciones y maneras de declararlas.

Funciones por declaración: Este tipo de función se creará con la palabra reservada function, seguido obligatoriamente por un nombre, que identificará a nuestra función, una lista de parámetros entre paréntesis, y el símbolo de las llaves {}. Qué será el que delimite el contenido de nuestro conjunto de sentencias.

Ejemplo:

function hola(nombre) { console.log (`Hola ${nombre}.`) }

hola('Victor'); Response => Hola Victor

Funciones por expresión: Es similar a las de declaración, La única diferencia es que la definición de nuestra nueva función no comienza por instrucción function, no es compatible con hoisting, y el nombre de la función es opcional.

Ejemplo:

const SUMADOS = function sumaDos(valor) { return valor + 2; }

console.log(SumaDos(5)); Response => 7

Funciones flecha o Arrow Functions: Es una forma más corta que se creó con el ES6 para escribir funciones, Primero definiremos la lista de parámetros, en caso de ser necesario, entre paréntesis seguido del símbolo => y las {} para indicar las instrucciones que se van a realizar.

Ejemplo: hello = () => { return "Hello World!"; }

Response => Hello World!

4.¿Qué es el DOM?

DOM es la abreviatura de Document Objetc Module y a través del DOM se puede acceder, por medio de JavaScript, a cualquiera de estos elementos, es decir a sus correspondientes objetos para alterar sus propiedades o invocar a sus métodos. Con todo, a través del DOM, queda disponible para los programadores de JavaScript, cualquier elemento de la página, para modificarlos, suprimirlos, crear nuevos elementos y colocarlos en la página, etc.

5.Métodos para acceder al DOM

Existen varios métodos para acceder al DOM desde JavaScript ya que todo el HTML en si está lleno de objetos los cuales, son llamados nodos, también existen nodos padres y nodos hijos, a los que se puede acceder con una serie de códigos, sin embargo los métodos más usados para entrar al DOM son:

Ejemplo 1. document.getElementById (): Con esta línea de código lo que le estamos diciendo al navegador es que en document que es el documento HTML traiga el elemento por el identificador.

```
<html>

<head>
<title>ejemplo de getElementsByTagName</title>

<script type="text/javascript">

function getAllParaElems()
{
  var allParas = document.getElementsByTagName("p");

  var num = allParas.length;

  alert("Hay " + num + " <p> elementos en este documento");
}


</script>
</head>

<body style="border: solid green 3px">
<p>Algo de texto</p>
<p>Algo de texto</p>
<p>Algo de texto</p>
<p>Algo de texto</p>

<button onclick="getAllParaElems();">
 muestra todos los elementos p en el documento</button><br />


</body>
</html>
```

Ejemplo 2. document.getElementByTagName(): Con este método le estamos diciendo al navegador que busque en el documento HTML la etiqueta que está dentro del paréntesis y que la traiga en forma de lista.

```
<html lang="es">

<head>
<title>Modificación del borde de una imagen</title>
<script type="text/javascript">
function setBorderWidth(width) {
  document.getElementById("img1").style.borderWidth = width + "px";
}
</script>
</head>

<body>
<p>
  <img id="img1" src="https://pbs.twimg.com/media/E5tvmRJX0AEc4Bs.png" style
="border: 5px solid green;" width="100"
height="100" alt="test de borde">
</p>

<form name="Formulario">
  <p><input type="button" value="Definir un borde de 20px"
onclick="setBorderWidth(20);"> <input type="button" value="Definir un borde
de 5px"
onclick="setBorderWidth(5);"></p>
</form>

</body>
</html>
```

6.Métodos para crear elementos en el DOM

Podemos crear elementos en el DOM con el siguiente código document.createElement(); para entender mejor esto veamos los siguientes ejemplos:

Ejemplo 1. Crear un párrafo: para crear un párrafo debemos crear un elemento “p” el cual irá dentro del paréntesis a forma de etiqueta.

```
<script>
    // Crear nodo de tipo Element
    var parrafo = document.createElement("p");


    // Crear nodo de tipo Text
    var contenido = document.createTextNode("Hola Mundo!");

    // Añadir el nodo Text como hijo del nodo tipo Element
    parrafo.appendChild(contenido);

    // Añadir el nodo Element como hijo de la pagina
    document.body.appendChild(parrafo);
</script>
```

Aquí lo que hacemos es utilizar la jerarquía de los nodos hijos y los nodos padres para que la cadena de texto entre dentro de la etiqueta “p” y después añadimos la etiqueta “p” al documento HTML

Ejemplo 2. Crear un título: En este caso dentro de los paréntesis pondremos “h1” que es la etiqueta que se utiliza por lo general para los títulos. Como en el caso anterior utilizaremos la jerarquía de los nodos padres e hijos para que así esto se muestre en el documento HTML.

```
<script>
    // Crear nodo de tipo Element
    var parrafo = document.createElement("h1");


    // Crear nodo de tipo Text
    var contenido = document.createTextNode("titulo de la pagina");

    // Añadir el nodo Text como hijo del nodo tipo Element
    parrafo.appendChild(contenido);

    // Añadir el nodo Element como hijo de la pagina
    document.body.appendChild(parrafo);
</script>
```

7. Elementos para editar atributos

Existen las siguientes maneras de editar atributos según el caso:

hasAttribute(); : Muestra un booleano true o false.

getAttribute(); Muestra el valor de un atributo especificado o null.

setAttribute(); : Agrega o actualiza el valor de un atributo especificado.

removeAttribute(); : Elimina un atributo de un elemento.

Ejemplo 1. Remover atributo de una imagen:

```
const img = document.querySelector('img');      // Utilizamos esta línea para buscar la imagen el documento HTML

img.removeAttribute('src');          // con esta línea se elimina el atributo src que es el que busca el recurso y por lo tanto no se podría buscar la imagen.
```

Ejemplo 2. Llamar una imagen:

```
const img = document.querySelector('img');      // Utilizamos esta línea para buscar la imagen el documento HTML

img.setAttribute('src', 'https://js-tutorials.nyc3.digitaloceanspaces.com/octopus.png');  //Con esta línea le estamos agregando el atributo src junto con la dirección URL donde está la imagen y por lo tanto el atributo src puede buscar el elemento y mostrarlo en el HTML.
```