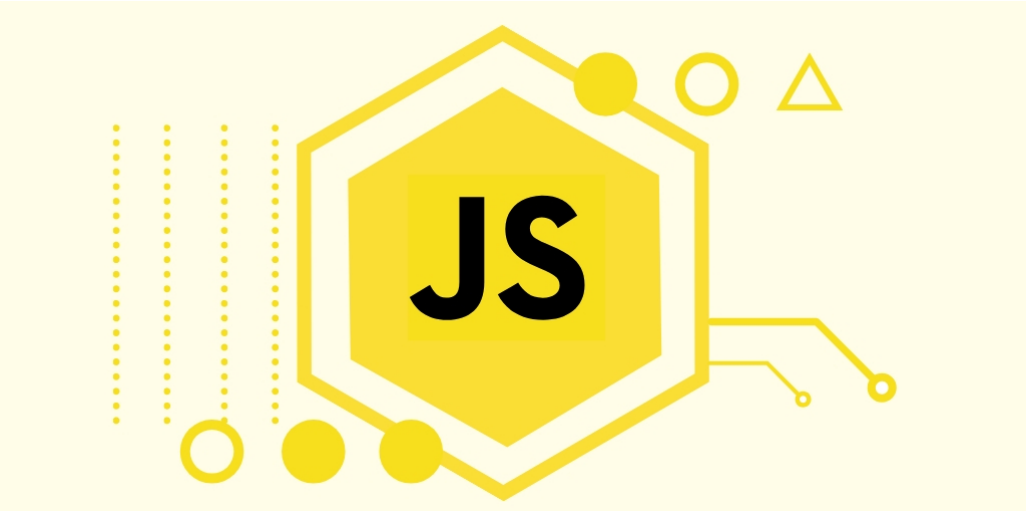
Universidad Autonoma de Aguascalientes

LICENCIATURA EN MATEMATICAS APLICADAS

MATERIA: Java Script

Docente: Bryan Ricardo Barbosa Olvera

FECHA DE CREACION: 18 de Junio de 2022



BRYAN RICARDO BARBOSA OLVERA

22 de junio de 2022

1. POO

1.1. Crear Objetos

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
```

1.2. Acceder a los valores

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
console.log(producto.nombre);
```

1.3. Agregar O Eliminar valores

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
//Agregar nuevas propiedades
producto.imagen = ïmagen.jpg";
//ELiminar propiedades del objeto
delete producto.precio
```

1.4. Destructuring

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
//Crea y asigna el valor a la variable
const nombre, precio = producto;
console.log(nombre,precio);
```

1.5. Destructuring de objetos anidados

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
informacion: {
fabricacion: {
pais: 'CHina'
}}}
const { nombre, informacion , informacion : { fabricacion: {pais }}} = producto;
console.log(pais);
```

1.6. Conjelar un objeto

```
”use strict ”
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
//Al conjelar el objeto no deja que se le agregen o eliminen valores del objeto
Object.freeze(producto);
//el siguiente comando indica con un true si esta conjelado el objeto
console.log(Object.isFrozen(producto));
```

1.7. Sellar un objeto

```
”use strict ”
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
//Al sellar un objeto es parecido a conjelarlo con la diferencia que le permite cambiar el valor de las llaves
Object.seal(producto);
producto.precio = 200;
//el siguiente comando indica con un true si esta sellado el objeto
console.log(Object.isSealed(producto));
```

1.8. Spread Operator o Rest Operator

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
const medidas = {
peso: ’1kg’,
medida: ’1m’
}
//Lo que realiza es unir dos objetos en uno solo
const resultado = { ...producto , ...medidas }
```

1.9. La palabra reservada this

```
//Te permite no perder la referencia de la varable que se esta llamando y no tomar variables fuera del objeto
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
,mostrarInfo: function () {
console.log(‘EL producto tiene como nombre: ${ this.nombre }‘) }
}
```

1.10. .keys .values .entries

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
}
//.keys te retorna las llaves del objeto en un objeto
console.log(Object.keys(producto));
//.values te retorna los valores del objeto en un objeto
console.log(Object.values(producto));
//.entries te retorna las llaves y los valores del objeto en pares en un objeto
console.log(Object.entries(producto));
```

2. ARRAYS

2.1. Crear un arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];

2.2. Acceder a los valores de un arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
console.log(meses[0]); //Lo siguiente es para acceder al valor de un arreglo dentro de otro arreglo
const todo = [1,[1,2]];
console.log(todo[1][1]);
//Observece que imprimira el valor 2

2.3. Longitud de un arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
console.log(meses.length);
//Observese que retornara como valor 7

2.4. Agregar un valor nuevo en un arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
//Observamos que el arreglo solo tiene 7 valores, en el cual se puede acceder en la posicion6
//Para agregar uno nuevo es de la siguiente manera:
meses[7] = 'Nuevo mes';

2.5. Agregar elementos con el spreed operator al inicio o final

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
let mesesNuevo = [...meses, 'Junio'];
mesesNuevo = ['Diciembre' , ...mesesNuevo];

2.6. Eliminar un valor al principio del arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.shift();

2.7. Eliminar un valor al final del arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.pop();

2.8. Eliminar un valor en cualquier posicion del arreglo

const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.splice(1,1);
//Elimina febrero
meses.splice(posicion donde empesara a borrar valores , cantidad de valores que se quieren eliminar);

2.9. destructuring con Arreglos

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
const [vEnero, vFebrero] = meses;
console.log(vFebrero);
//Observemos que crea la variable vFebrereo con el valor febrero no necesariamente con el mismo nombre
const [, vFebrero] = meses;
console.log(vFebrero);
//Observemos que crea la variable vFebrereo pero dejamos un espacio para indicar que no queremos crear una variable inecesaria
const [, vFebrero,...mesesFaltantes] = meses;
console.log(mesesFaltantes);
//meses Faltantes obtendra un arreglo con todos los valores que no se asignaron
```

2.10. Iterarcon Arreglos con .forEach

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.forEach((producto) {
console.log(producto);
} )
```

2.11. Iterarcon Arreglos con .map

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.map((producto) {
console.log(producto);
} )
//Realiza lo mismo que el forEach pero el map crea un nuevo arreglo
```

3. Iteradores

3.1. .forEach

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.forEach((pendiente, index)=>{
console.log('${ indice }$ : { mes }');
} )
```

3.2. .map

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
meses.map((pendiente, index)=>{
console.log('${ indice }$ : { mes }');
} )
//La diferencia principal al .forEach es que crea un nuevo arreglo
```

3.3. .forOf

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
for( let mes of meses) {
console.log(pendiente);
}
```

3.4. .forIn

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
informacion: {
fabricacion: {
pais: 'CHina'
}}}
for(let propiedad in producto ) {
console.log( `${[propiedad] }`);
}
```

3.5. ALternativa del .forIn con el .forOf

```
const producto = {
nombre: "Monitor 20 pulgadas",
precio: 300,
informacion: {
fabricacion: {
pais: 'CHina'
}}}
for(let [llave, valor] of Object.entries(producto) ) {
console.log(valor);
console.log(llave);
}
```

4. Array Methods

4.1. .includes

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
const resultado = meses.includes('Enero');
//te retorna un valor booleano si existe el valor en el arreglo
//solo funciona en arreglos que no tienen objetos dentro
```

4.2. .some

```
const carrito =[
{ nombre: 'Monitor 27 Pulgadas' , precio: 500 } ,
{ nombre: 'Television' , precio: 100 }
]
const existe = carrito.some(producto =>{
return producto.nombre === 'Television'
} ) //te retorna un valor booleano si existe el valor en el arreglo
```

4.3. .findIndex

```
const meses = ['Enero' , 'Febrero' , 'Marzo' , 'Abril' , 'Mayo' , 'Junio' , 'Julio'];
const indice = meses.findIndex( mes =>mes === 'Abril');
//Te retorna e indice donde encontro el resultado , si no existe te retorna el -1
```

4.4. .reduce

```
const carrito =[
{ nombre: 'Monitor 27 Pulgadas' , precio: 500 } ,
{ nombre: 'Television' , precio: 100 }
]
const existe = carrito.some(producto =>{
let resultado = carrito.reduce( (total,producto) =>total +producto.precio, 0 );
//El total guarda la instancia por eso no se asigna +=, te retornara la suma de todos los precios del arreglo
//El cero indica desde que numero empesara el total
```

4.5. .filter

```
const carrito =[
{ nombre: 'Monitor 27 Pulgadas' , precio: 500 } ,
{ nombre: 'Television' , precio: 400 }
]
let resultado;
resultado = carrito.filter(producto=>producto.precio >400);
//creara un arreglo con todos los valores que cumplen la condicion
```

4.6. .find

```
const carrito =[
{ nombre: 'Monitor 27 Pulgadas' , precio: 500 } ,
{ nombre: 'Television' , precio: 400 }
]
const resultado2 = carrito.find(producto =>producto.precio === 100 );
console.log(resultado2);
//resultado2 toma el valor del primer elemento que encuentre
```

4.7. .every

```
const carrito =[
{ nombre: 'Monitor 27 Pulgadas' , precio: 500 } ,
{ nombre: 'Television' , precio: 400 }
]
const resultado = carrito.every(producto =>producto.precio <1000);
//Te retornara un true si todos los valores del arreglo cumplen la condicion
```

5. DOM

5.1. document

let elemento = document;
//Te seleciona todo el html del documento

5.2. document.all

let elemento = document.all;
//Te seleciona todos los elementos que conforman el HTML

5.3. document.head

let elemento = document.head;
//Te seleciona todos los elementos que conforman head del HTML

5.4. document.body

let elemento = document.body;
//Te seleciona todos los elementos que conforman body del HTML

5.5. document.domain

let elemento = document.domain;
//Te seleciona el dominio

5.6. document.forms

let elemento = document.forms;
//Te seleciona todos los formularios y te da la cantidad

5.7. document.forms[index]

let elemento = document.forms[0];
//Te seleciona el formulario en la posicion 0

5.8. document.forms[index].id

let elemento = document.forms[0].id;
//Te seleciona el id del formulario

5.9. document.forms[index].method

let elemento = document.forms[0].method;
//Te seleciona el metodo del formulario

5.10. document.forms[index].classList

let elemento = document.forms[0].classList;
//Te seleciona todas las clases del formulario

5.11. document.links

let elemento = document.links;
//Te retorna todos los enlaces del documento

5.12. document.images

let elemento = document.images;
//Te retorna todas las imagenes del documento

5.13. document.scripts

let elemento = document.scripts;
//Te retorna todos los scrpts del documento

5.14. Seleccionar los elementos por su clase

const header = document.getElementsByClassName('header');
console.log(geader);
//Si existe mas de un elemento con la misma clase se trae todos los contenedores con la misma clase

5.15. Seleccionar los elementos por su id

const formulario = document.getElementById('formulario');

5.16. querySelector

const card = document.querySelector('.card');
console.log(card);
//Te retorna el primero que encuentre
const card = document.querySelector('.premium .info');
//Puedes tener selectores mas precisos como en css
const card = document.querySelector('.section.hospedaje .card:nth-child(2)');
//Te retornara el segundo card
const formulario = document.querySelector('#form');
//Tambien funciona con los id

5.17. querySelectorAll

const card = document.querySelectorAll('.card');
console.log(card);
//FUnciona igual que el anterior pero te retorna todos los que tengan la misma clase

5.18. Modificar textos o imagenes

const encabezado = document.querySelector('.contenido-hero h1');
console.log(encabezado);
console.log(encabezado.textContent);
//el textContent se trae el texto
console.log(encabezado.innerHTML);
//EL innerHTML se trae todo el HTML
const imagen = document.querySelector('.card img');
imagen.src = 'img/hacer2.jpg'

//Lo que realiza es cambiar la imagen

5.19. Cambiando el CSS

const encabezado = document.querySelector('h1');
console.log(encabezado.style);
//.style selecciona los estilos de la clase o id del html
encabezado.style.backgroundColor = red;
Puede modificar todos los estilos de css, solo que sin el guion y con mayuscula al inicio
const card = document.querySelector('.card');
card.classList.add('nueva-clase');
//con .add puedes agregar una nueva clase
card.classList.remove('nueva-clase');
//con .remove puedes quitar una nueva clase
//Es mas recomendado agregar o quitar las clases para dejar los estilos en el documento css

5.20. Traversing the DOM del padre al hijo

const navegacion = document.querySelector('.navegacion'); console.log(navegacion.children);
//Te trae todos los elementos hijso de la clase console.log(navegacion.children[0]);
//Puedes acceder a ellos como en un arreglo
console.log(navegacion.children[0].children[1]);
//Puedes selecccionar al hijo y despues listar los hijos y seleccionar un hijo

5.21. Traversing the DOM del hijo al padre

const card = document.querySelector('.card'); console.log(navegacion.parentElement);
//Te lleva hacia el padre

5.22. Traversing the DOM entre hermanos

const card = document.querySelector('.card'); console.log(navegacion.nextElementSibling);
//Te retorna el elemento hermano siguinte
console.log(navegacion.previousElementSibling);
//Te retorna el elemento hermano anterior

5.23. Traversing the DOM tarearte el ultimo o el final

const card = document.querySelector('.card'); console.log(navegacion.firstElementChild);
//Te selecciona el primer hijo
console.log(navegacion.lastElementChild);
//Te selecciona el ultimo hijo

5.24. Eliminar elementos en el DOM