UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Introducción a la Programación Web - Maquetador Web

Clase 5

NEORIS

Repaso

¿Qué vimos en la clase anterior?

- Variables
- Tipos de Datos
- Funciones + Funciones Anónimas
- Estructuras de Control
- Arrays

Arrays

Variable especial que **permite almacenar** más de un **valor** al mismo tiempo.

En Javascript, tanto la longitud como el tipo de los elementos de un array son variables.

Podemos agregar o quitar elementos de diferentes tipos de datos.

```
let autos = ["Civic", "Etios", 308]
```

Operaciones con Arrays

Añadir un elemento al final del array:

```
array.push(elemento);
```

Añadir un elemento al comienzo del array:

```
array.unshift(elemento);
```

Eliminar el último elemento de un array:

```
array.pop();
```

Eliminar el primer elemento de un array:

```
array.shift();
```

Saber la cantidad de elementos:

```
array.length;
```

Como recorrer un array

Utilizando un bucle FOR:

```
var frutas = [“banana”, “manzana”, “tomate”, “naranja”];

for(i = 0; i < frutas.length; i++){
    console.log(frutas[i]);
}
```

¿Y usando un WHILE?

Como recorrer un array

Con un bucle WHILE:

```
let largo = frutas.length
let indice = 0

while (indice < largo) {
    console.log(frutas[indice]);
    indice++;
}
```

ForEach

```
const array1 = ['a', 'b', 'c'];

array1.forEach(function(value, index){
    console.log(value);
    console.log(index);
})
```

```
> var array1 = ['a', 'b', 'c'];

  array1.forEach(function(value, index){
      console.log(value + " " + index);
  })
  a 0
  b 1
  c 2
```

ForEach - Ejercicio

Dado el siguiente arreglo:

const numeros = [1, 2, 3, 4, 5, 10, 20];

Decir cuánto es la suma total de todos sus números.

ForEach - Ejercicio

Resolución:

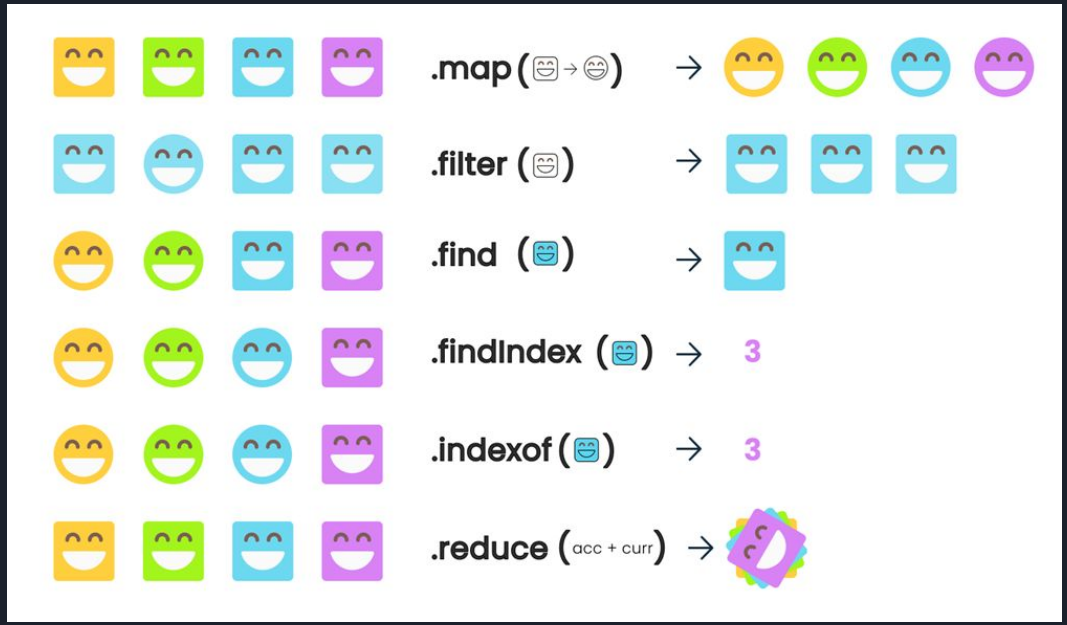
```
const numeros = [1, 2, 3, 4, 5, 10, 20];

let total = 0;

numeros.forEach(function (valor){
    total += valor;
  })

console.log("El valor total es: " + total);
```

Javascript - Operadores Funcionales



Javascript - Operadores Funcionales

Filter

Crea un array con todos los elementos que cumplan una condición implementada por la función dada.

No genera efecto colateral.

Ejemplo: Dado el array palabras, guardar en una variable las palabras que tengan más de 6 letras

```
const palabras = ['casa', 'perro', 'electrodoméstico', 'universidad',
'mueblería', 'gato'];
const resultado = palabras.filter(unaPalabra => unaPalabra.length > 6);
console.log(resultado);
// expected output: Array ["electrodoméstico", "universidad, "mueblería"]
```

Javascript - Operadores Funcionales

Map

Crea un array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.

No genera efecto colateral.

Ejemplo: Dado el array numeros, guardar en una variable el doble de los números dentro del array

```
var numeros = [1, 5, 10, 15];
var dobles = numeros.map(function(x) {
    return x * 2;
}); // dobles es [2, 10, 20, 30]
// numeros sigue siendo [1, 5, 10, 15]
```

Javascript - Operadores Funcionales

Some

Comprueba si al menos un elemento del array cumple con la condición implementada por la función implementada.

Ejemplo: Dado el array numeros, queremos saber si alguno de ellos es par

```
const numeros = [1, 2, 3, 4, 5];
// comprobamos si algún elemento es par

console.log(numeros.some(n => n % 2 === 0 ));
// expected output: true
```

Javascript - Operadores Funcionales

Every

Determina si todos los elementos del array satisfacen una condición

Ejemplo: Dado el array numeros, queremos saber si todos son mayores o iguales que 3

```
const numeros = [1, 2, 3, 4, 5];
console.log(numeros.every(n => n >= 3));
// expected output: false
```

Javascript - Operadores Funcionales

Reduce

Ejecuta una función reductora sobre cada elemento de un array, devolviendo como resultado un único valor.

No genera efecto colateral.

Ejemplo:

```
const array1 = [1, 2, 3, 4];
const reducer = (accumulator, currentValue) => accumulator + currentValue;
// 1 + 2 + 3 + 4
console.log(array1.reduce(reducer)); // expected output: 10 (1 + 2 + 3 + 4)
console.log(array1.reduce(reducer, 5)); // expected output: 15
```

¿Cómo definimos un objeto en JS?

```
var persona = {};
var persona = {
  nombres: ['Rodrigo', 'Juan'],
  edad: 32,
  intereses: ['música', 'esquí']
}
```

Para acceder a una propiedad específica lo hacemos utilizando el '.'
Por ejemplo: ***persona.edad*** nos devolverá la edad almacenada en dicha variable.

¿Cómo definimos un objeto en JS?

```
var persona = {
  nombres: ['Rodrigo', Juan],
  edad: 32,
  intereses: ['música', 'esquí'],

  saludo: function() {
    alert('Hola, Soy '+ this.nombre[0] + '. ');
  }
};
```

De la misma forma en que nosotros definimos un objeto con **propiedades** y **métodos**, Javascript entiende los elementos de nuestro .html de la misma manera.

Arrays de Objetos

Como en Javascript generalmente tenemos que manipular objetos, es necesario entender que muchas veces podemos estar trabajando con Arrays en cuyas posiciones tenemos un Objeto.

Arrays de Objetos

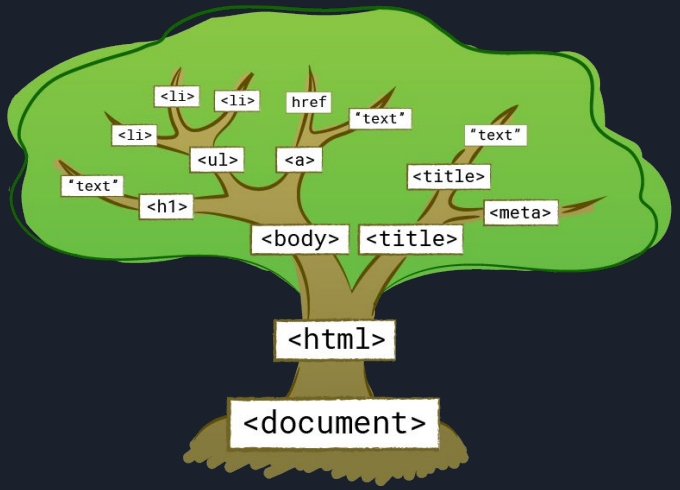
Se definen de la siguiente forma:

```
let autos = [
    {
    'marca': 'Honda',
    'modelo': 'Civic',
    'color': 'Negro',
    'anio': 2010
    },
    {
    'marca': 'VolksWagen',
    'modelo': 'Golf',
    'color': 'Blanco',
    'anio': 2015
    }
]
```

¿Cómo obtenemos la marca y el modelo del auto en la primera posición?

Javascript y HTML

Javascript entiende nuestra página web como una **estructura de árbol**, donde cada elemento de nuestro .html es un **objeto** o **nodo** con determinadas **características** y **funciones asociadas**.



DOM - Document Object Model

- El **DOM** es una **representación** orientada a **objetos** de la página web.
- Diagrama la página web como una **estructura de árbol**, donde cada elemento de nuestro .html es un **objeto** o **nodo** con determinadas **propiedades** y **funciones asociadas**.
- Javascript utiliza esta representación de la página de manera que podamos cambiar la estructura del documento, sus estilos y contenido.

Propiedades y Métodos

Una propiedad es un valor que podés **obtener** o **definir**, como por ejemplo obtener y redefinir el contenido de un elemento HTML, el color de letra aplicado a un texto, etc.

Un **método** es una acción (función) que podes realizar como *obtener*, *agregar* o *eliminar* un elemento HTML.

De esta forma, el DOM nos proporciona métodos y propiedades asociados a cada elemento HTML que nos permitirán manipular el documento.

En https://developer.mozilla.org/ podemos encontrar propiedades y métodos que nos sirven para manejar los elementos del documento.

Algunas propiedades y métodos

Propiedades

`element.innerHtml`

Obtener / cambiar el contenido del elemento

`element.classList`

Obtener / cambiar el contenido de la **lista de clases** del elemento

`element.children`

Accedemos a la lista de las etiquetas “hijas” del elemento

`element.id`

Accedemos al id del elemento

`element.style.estilo`

Accedemos a los estilos del elemento

`element.value`

Accedemos al valor del atributo de elemento

`. . .`

Algunas propiedades y métodos

Métodos

`document.getElementById("")`

Obtenemos un elemento por su ID

`document.getElementsByClassName("")`

Obtenemos una lista de elementos por su clase

`document.getElementsByTagName("")`

Obtenemos un elemento por su nombre de etiqueta

`element.insertCell("")`

Permite insertar una celda nueva en una tabla

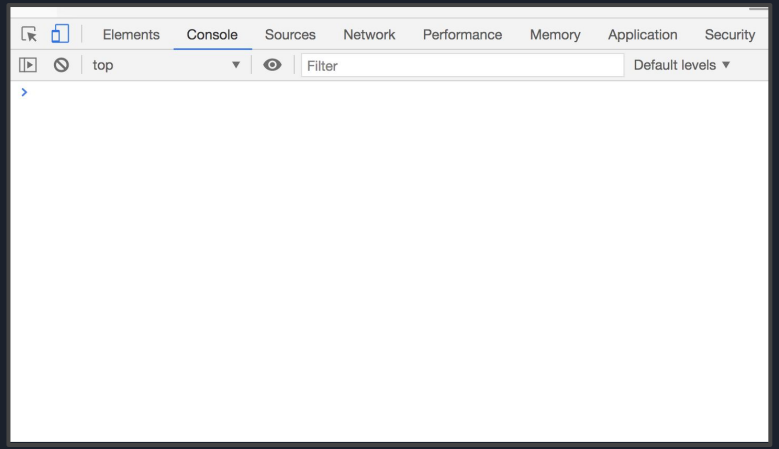
`element.appendChild()`

Permite agregar una etiqueta hija al elemento

Javascript en la consola del navegador

La consola permite interactuar con una página web mediante la realización de expresiones de Javascript dentro del contexto de la página.

Básicamente la consola brinda la capacidad de **escribir**, **administrar** y **monitorear** Javascript dentro del entorno de una página web.



Practiquemos en la consola

- Abrir la consola en una página web.
- Obtener todos los elementos “<p>”.
- Obtener todos los elementos “<h1>”
- Obtener un elemento por su Id. (Necesitamos conocer el id)
- Cambiarle el color de fondo a dicho elemento.
- *Obtener todos los elementos “<p>” y aplicarles un color de texto rojo.*

Practiquemos en la consola

Resolución

```
let todosLosP = document.getElementsByTagName("p");

let todosLosH1 = document.getElementsByTagName("h1");

let elementoPorId =
document.getElementById("firstHeading");

elementoPorId.style.backgroundColor = "red";

for(let i = 0; i < todosLosP.length; i++){
    todosLosP[i].style.color = "red";
}
```