Javascript

CONSTRUCCIÓN DE INTERFACES DE USUARIO

1er Cuatrimestre de 2019

Universidad Nacional de Quilmes

Consideraciones

- Esta clase es una **presentación** de Javascript.
- **No es** una guía exhaustiva.
- Todo nuevo lenguaje se aprende codeando.
- Cuando surjan dudas primero vayan a https://developer.mozilla.org/es/docs/Web/JavaScript
- Y tienen las Prácticas y los TPs para codear.

Acerca de Javascript

- Lenguaje de Propósito General
- Multi Plataforma
- Tipado dinámico

- JavaScript fue desarrollado originalmente por Brendan Eich de Netscape con el nombre de Mocha, el cual fue renombrado posteriormente a LiveScript, para finalmente quedar como JavaScript.

Un poco de sintaxis

- Variables
  - Mutables

```
var a = "Se está dejando de usar"
```

```
let a = "para usar let"
```

  - Inmutables

```
const b = 1
```

Un poco de sintaxis

- Objeto: Es un conjunto de clave valor encerrado entre llaves
  - { x: 1, y: 2 }

- Lista: Es un conjunto de valores encerrados entre corchetes
  - [1, 2]
  - [1, ‘2’, ‘tres’, { x: 1, y: 2 }]

Un poco de sintaxis

- Strings
  - `“comillas dobles”`
  - `‘Comillas simples’`
  - `` `Por último las comillas francesas` ``
- La última forma se utiliza para hacer String interpolation
  - `` `Hello ${name}` ``

Un poco de sintaxis

- Funciones

```
function sayHello(name) {
   return `Hello ${name}`
}
const sayHello = name => `Hello ${name}`
```

- Tiene default parameters

```
function sayHello(name = 'World') {
   return `Hello ${name}`
}
```

Un poco de sintaxis

```
class Gretter {
    constructor(name) {
        this.name = name
    }

    sayHello() {
        return `Hello ${this.name}`
    }
}
```

Un poco de sintaxis

- Igualdades
  - Débil o relajada

```
const one = 1
const anotherOne = "1"
console.log(blue == sameBlue) // true
```

  - Estricta

```
const one = 1
const anotherOne = "1"
console.log(blue === sameBlue) // false
```

Un poco de sintaxis

- If clasico

```
let max
if (a > b) {
  max = a
} else {
  max = b
}
```

- If ternario

```
const max = (a > b) ? a : b
```

Operaciones con Colecciones

```
const list =[1, 2, 3]
```

Filter
```
list.filter( v => v > 2 ) // [3]
```

Map
```
list.map(v => v * 2 ) // [2, 4, 6]
```

Every
```
list.every(v => v > 2 ) // false
```

Find
```
list.find(v => v > 3) // undefined
```

First, Last
```
list[0] // 1
```

Sorted
```
list.sort() // [1, 2, 3]
```

Libreria para agregar mas funciones a las listas
https://lodash.com/docs/4.17.11


LIVE DEMO

Request

Para hacer request vamos a utilizar una librería llamada Axios

```
const axios = require('axios');
const ___ = axios.get('https://randomuser.me/api/')
console.log(___)
```

Promises

Una Promesa es un proxy para un valor no necesariamente conocido en el momento que es creada la promesa.

Las Promesas tienen tres estados:

- Pendiente (pending)
- Cumplida (fulfilled)
- Rechazada (rejected)

Promises

```
const axios = require('axios');
const ___ = axios.get('https://randomuser.me/api/')
  .then(data => console.log(data))
  .catch(error => console.log(error))
```

Spread

Permite a un elemento iterable tal como un arreglo ser expandido en lugares donde cero o más argumentos (para llamadas de función) ó elementos (para Array) son esperados, o a un objeto ser expandido en lugares donde cero o más pares de valores clave son esperados.

```
function sum(x, y, z) {
 return x + y + z;
}
const numbers = [1, 2, 3];
console.log(sum(...numbers));
```


LIVE DEMO