IIC2026 - Visualización de Información:

Javascript

Una introducción

Eugenio Herrera eiherrera@uc.cl

Repaso HTML & CSS

```html
<!doctype html>
<html>
  <head>
    <title>Título del documento</title>
  </head>
  <body>
    <h1 id="light-blue">Lorem ipsum</h1>
    <p class="orange">
      "Neque porro quisquam est
      qui dolorem ipsum quia dolor sit amet,
      consectetur, adipisci velit..."
    </p>
  </body>
</html>
```

Repaso HTML & CSS

```css
h1 {
    font-size: 28px;
}
p {
    font-size: 20px;
}
.orange {
    color: rgb(245, 152, 66);
}
#light-blue {
    color: rgb(32, 188, 250);
}
```

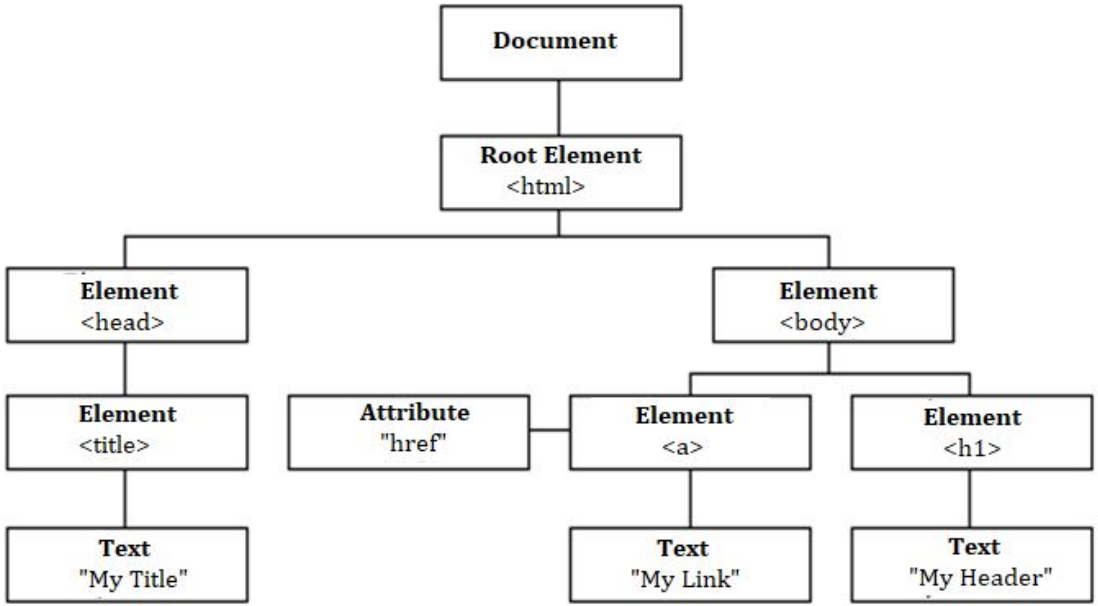
Repaso HTML & CSS

Lorem ipsum

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

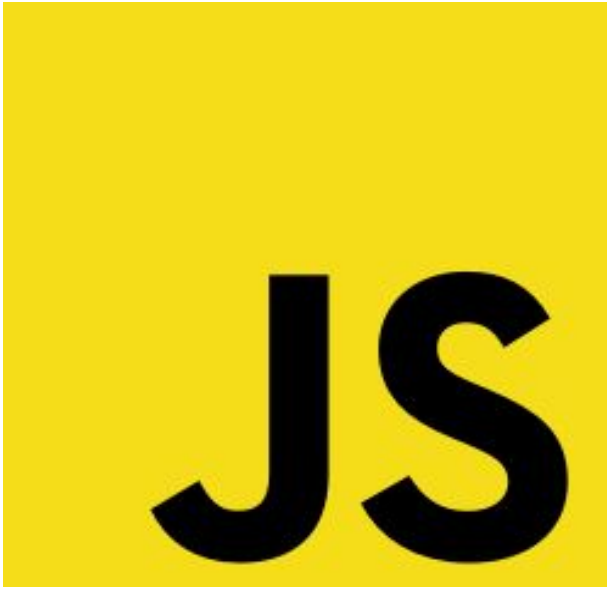
¿DOM?

Document Object Model



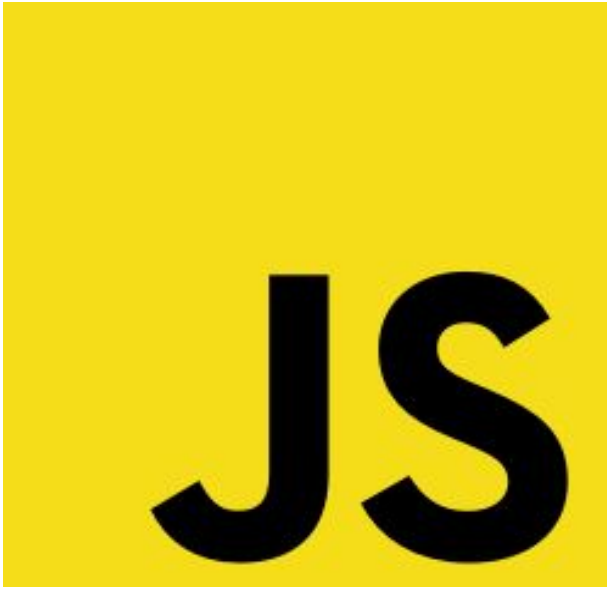
¿Qué es Javascript?

- **Lenguaje de programación** multiparadigma de alto nivel.
- Es interpretado, al igual que Python y Ruby.
- Sirve para **manipular el DOM**.
- Es **interpretado** por el **navegador**.



¿Qué es Javascript?

- **Lenguaje de programación** multiparadigma de alto nivel.
- Es interpretado, al igual que Python y Ruby.
- Sirve para **manipular el DOM**.
- Es **interpretado** por el **navegador**.



¿Cómo parto usándolo?

```
<!doctype html>
<html>
  <head>
    <script src="script.js"></script>
  </head>
  <body>
  </body>
</html>
```

Sintaxis

O cómo hacer en Javascript lo que hacía en Python

Variables

```python
numero = 14
flotante = 3.14
string = "Akira"
lista = [1, 2, 3, 4]
```

.py

```javascript
numero = 14;
var flotante = 3.14;
let string = "Akira";
const lista = [1, 2, 3, 4];
```

.js

let

```
let x;

x = 1;
x = "hola";
x = [1, 2, 3];

// No hay errores!
```

- Su valor puede variar.
- Tiene scope de bloque.

const

```
const x;
// SyntaxError: missing = in const declaration

const x = 1;
x = "hola";
// TypeError: invalid assignment to const `x'
```

- Su valor no puede variar.
- Tiene scope de bloque.

¿Y var?

Weak typing

```python
"Este es un string normal en python."
'Este es un string normal en python.'
"Esto no se puede hacer en python." + 2
# La última línea causará un error
```

.py

```javascript
const x = "Esto no se puede hacer en python." + 2;
console.log(x);
// 'Esto no se puede hacer en python.2'
```

.js

Cuidado con el weak typing...

```
> '5' - 3
2         // weak typing + implicit conversions * headaches
> '5' + 3
'53'      // Because we all love consistency
> '5' - '4'
1         // string - string * integer. What?
> '5' + + '5'
'55'
> 'foo' + + 'foo'
'fooNaN' // Marvelous.
> '5' + - '2'
'5-2'
> '5' + - + - - + - - + + - + - + - + - - - '-2'
'52'      // Apparently it's ok

> var x * 3;
> '5' + x - x
50
> '5' - x + x
5       // Because fuck math
```

for loops

```python
lista = [1, 2, 3, 4]
for element in lista:
    print(element)
```

.py

```javascript
const arreglo = [1, 2, 3, 4];
for (let index = 0; index < arreglo.length; index++){
    console.log(arreglo[index]);
}
```

.js

for loops (ES6)

```python
lista = [1, 2, 3, 4]
for element in lista:
    print(element)
```

.py

```javascript
const arreglo = [1, 2, 3, 4];
for (const element of arreglo){
    console.log(element);
}
```

.js

forEach

```
const arreglo = [1, 2, 3, 4];
for (const element of arreglo){
    console.log(element);
}
```

.js

```
const arreglo = [1, 2, 3, 4];
arreglo.forEach(function(element){
    console.log(element);
})
```

.js

ifs

```python
if verdad and falso:
    print('No se imprimirá esto.')
elif verdad or falso:
    print('Esto se imprime.)
else:
    print('No se llega a esta línea.')
```

.py

```javascript
if (verdad && falso){
  console.log('No se imprimirá esto.');
} else if (verdad || falso){
  console.log('Esto se imprime.);
} else {
  console.log('No se llega a esta línea.');
}
```

.js

functions

```
def funcion(x, y):
    return x + y
```

.py

```
function funcion(x, y){
    return x + y;
}
```

.js

arrow functions

```
function funcion(x, y){
    return x + y;
}
```

.js

```
const funcion = (x, y) => {
    return x + y;
}
```

.js

var

```
var x = 12;
function myFunc(){
    x = "doce";
}
myFunc();
// ahora x vale 'doce'
```

- Su valor puede variar.
- Tiene scope global.

var

```
var callbacks = [];
(function(){
  var i;
  for (i = 0; i < 5; i++) {
    callbacks.push( function() { return i; } );
  }
});
callbacks.forEach(func => {
    console.log(func());
})
```

- Su valor puede variar.
- Tiene scope global.

var

```
var callbacks = [];
(function(){
  var i;
  for (i = 0; i < 5; i++) {
    callbacks.push( function() { return i; } );
  }
});
callbacks.forEach(func => {
    console.log(func());
})
```

Imprime:
5, 5, 5, 5, 5

¿y si no usamos var, const o let?

```
variable1 = "h";
var variable2 = "o";
let variable3 = "l";
const variable4 = "a";
```

variable1 es **implícitamente** declarada como **var**

¡Veamos unos ejemplos!