Introducción a la *World Wide Web* (parte 2)

Nebil Kawas García
nebil@uc.cl
@nebil

repositorio: @nebil/intro-a-la-web

¿Dudas sobre la clase anterior?

Temario de hoy

- JavaScript & DOM

Temario de hoy

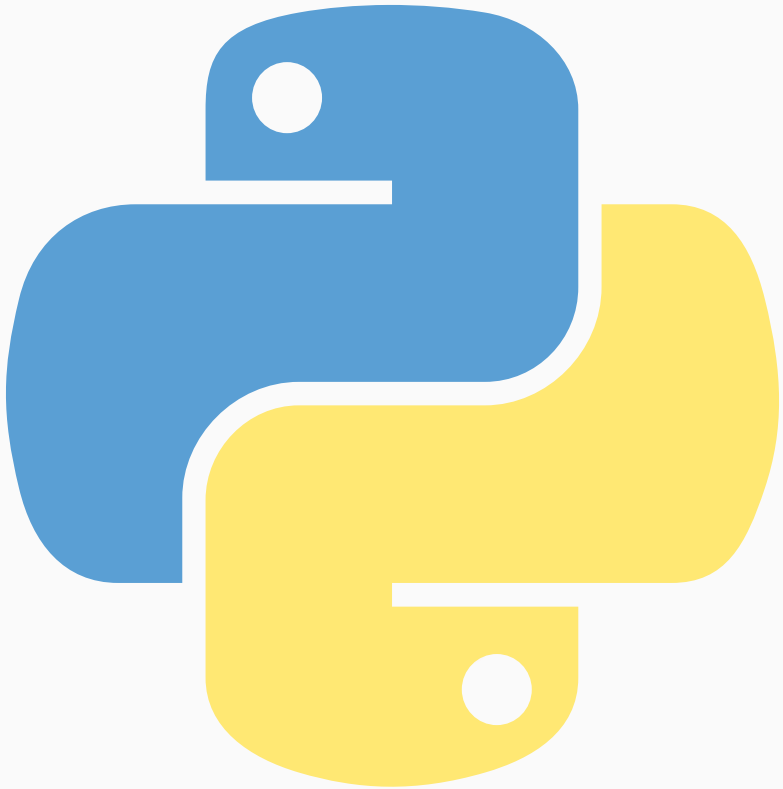
- JavaScript & DOM
- Librerías de JavaScript

JavaScript: la tercera pieza

¿Qué es JavaScript?

JavaScript: la tercera pieza

¿Qué es JavaScript... con respecto a Python?



JavaScript: la tercera pieza

¿Qué es JavaScript... con respecto a Python?

- Al igual que Python, JavaScript es un lenguaje de programación multiparadigma de alto nivel.

JavaScript: la tercera pieza

¿Qué es JavaScript... con respecto a Python?

- Al igual que Python, JavaScript es un lenguaje de programación multiparadigma de alto nivel.
- A diferencia de Python, JavaScript es el lenguaje que es interpretado por los navegadores.

JavaScript: la tercera pieza

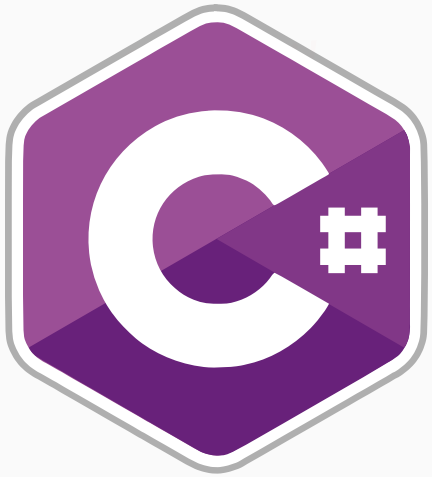
¿Qué es JavaScript... con respecto a Python?

- Al igual que Python, JavaScript es un lenguaje de programación multiparadigma de alto nivel.
- A diferencia de Python, JavaScript es el lenguaje que es interpretado por los navegadores.
- Tener un lenguaje de programación en el navegador permite darle dinamismo a la web.

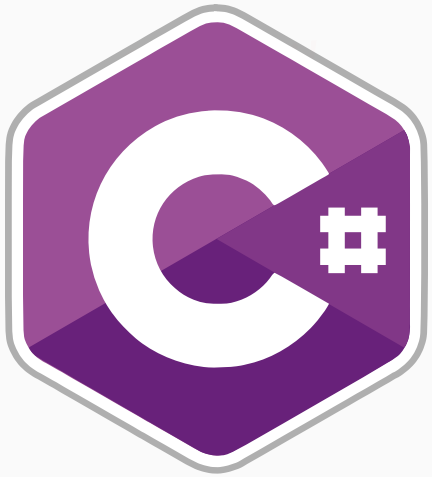
Duda: ¿por qué hay tantos lenguajes?

Duda: ¿por qué hay tantos lenguajes?

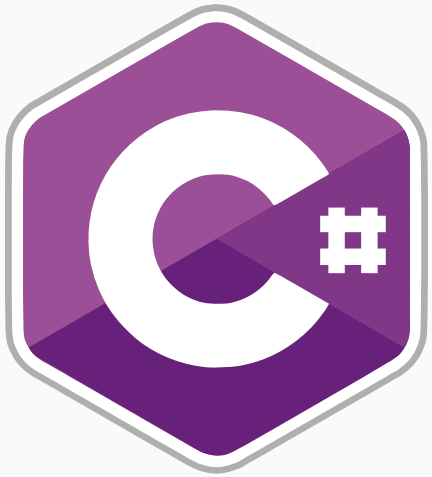
Duda: ¿por qué hay tantos lenguajes?



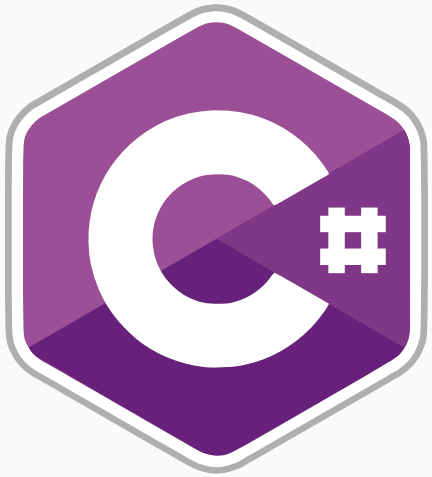
Duda: ¿por qué hay tantos lenguajes?

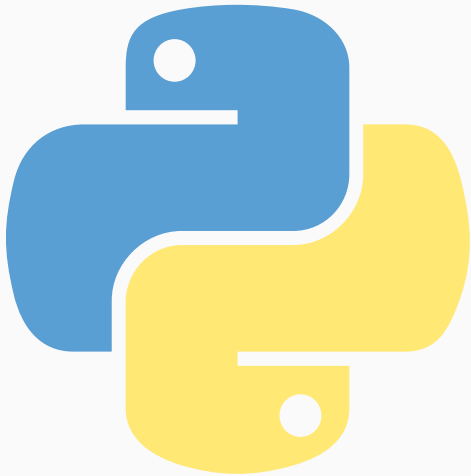


Duda: ¿por qué hay tantos lenguajes?



Duda: ¿por qué hay tantos lenguajes?

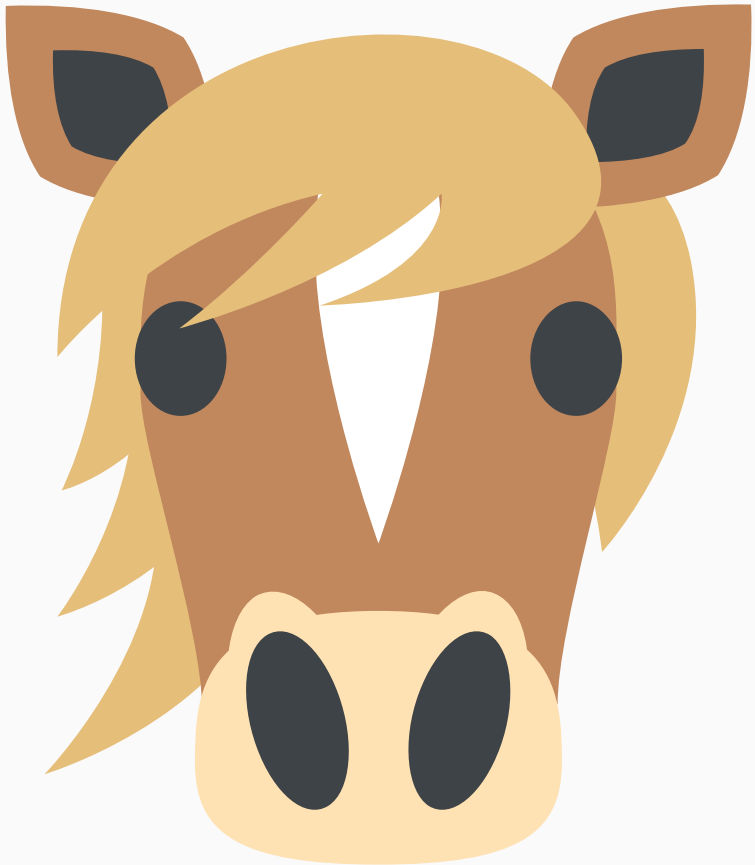




Duda: ¿por qué hay tantos lenguajes?

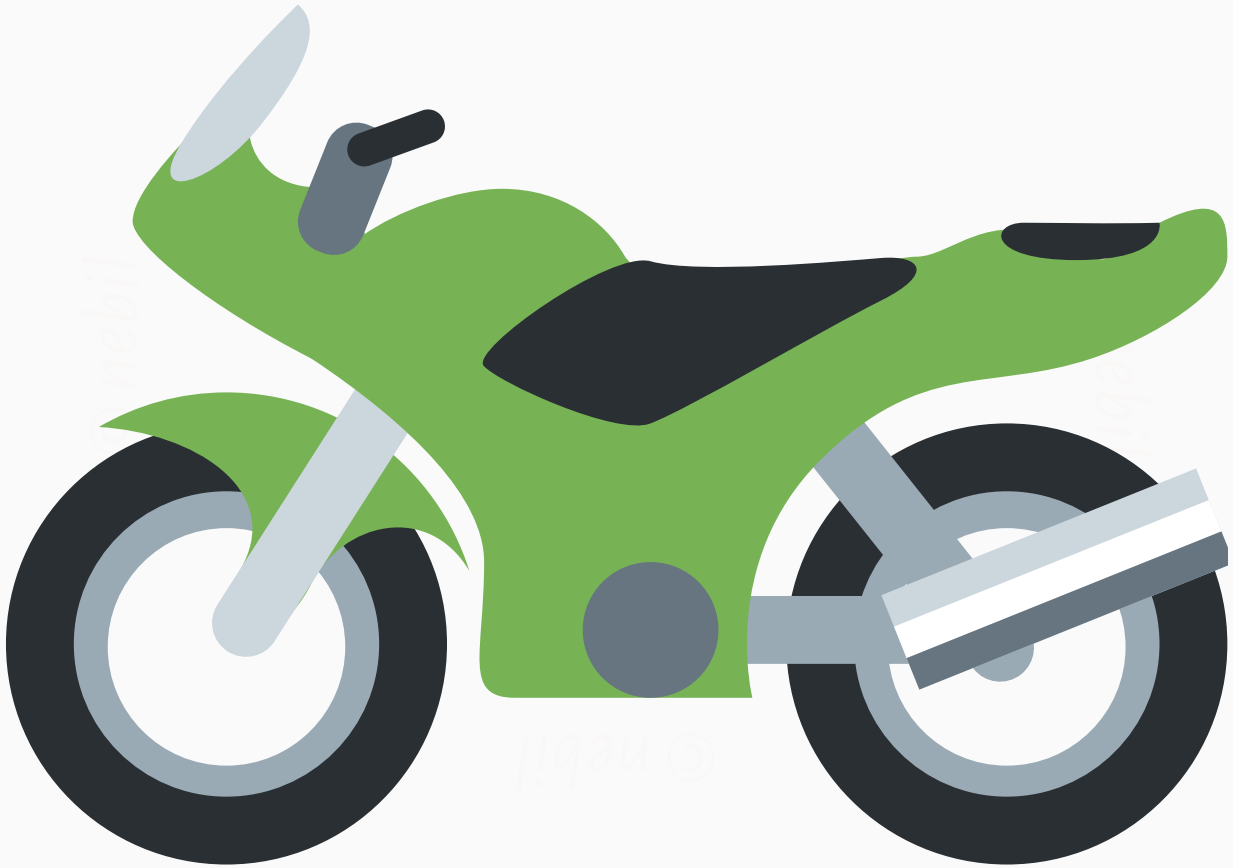
- Los lenguajes deben ser vistos como herramientas.
  Y tenemos diferentes herramientas...
  para diferentes trabajos.

Duda: ¿por qué hay tantos lenguajes?



Duda: ¿por qué hay tantos lenguajes?

Duda: ¿por qué hay tantos lenguajes?



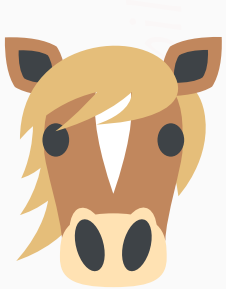
Duda: ¿por qué hay tantos lenguajes?

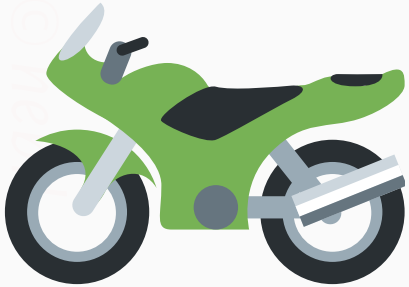
Duda: ¿por qué hay tantos lenguajes?

Duda: ¿por qué hay tantos lenguajes?

Duda: ¿por qué hay tantos lenguajes?

- Los lenguajes deben ser vistos como herramientas.
  Y tenemos diferentes herramientas...
  para diferentes trabajos.





Duda: ¿por qué hay tantos lenguajes?

- Los lenguajes deben ser vistos como herramientas.
  Y tenemos diferentes herramientas...
  para diferentes trabajos.
- Entre los lenguajes, (casi) siempre hay *trade-offs*:
  velocidad, seguridad, concurrencia, flexibilidad
  rapidez de desarrollo, etcétera.

Duda: ¿por qué hay tantos lenguajes?

- Los lenguajes deben ser vistos como herramientas.
  Y tenemos diferentes herramientas...
  para diferentes trabajos.
- Entre los lenguajes, (casi) siempre hay *trade-offs*:
  velocidad, seguridad, concurrencia, flexibilidad
  rapidez de desarrollo, etcétera.
- Y bueno, la elección también puede ser por gusto:
  existen lenguajes que son más estrictos (e.g. Java)
  y otros más flexibles (e.g. Ruby).

Un poco de historia

- JavaScript fue diseñado en sólo diez días, en mayo de 1995, por Brendan Eich, cuando trabajaba en Netscape.

Brendan Eich

Un poco de historia

- JavaScript fue diseñado en sólo diez días, en mayo de 1995, por Brendan Eich, cuando trabajaba en Netscape.
- Inicialmente, se llamó Mocha, luego LiveScript y finalmente, por asuntos de *marketing*... JavaScript.

Un poco de historia

- JavaScript fue diseñado en sólo diez días, en mayo de 1995, por Brendan Eich, cuando trabajaba en Netscape.
- Inicialmente, se llamó Mocha, luego LiveScript y finalmente, por asuntos de *marketing*... JavaScript.
- Fue estandarizado por Ecma International con el propósito de que todos los navegadores implementaran el mismo lenguaje.

Un poco de historia

- JavaScript fue diseñado en sólo diez días, en mayo de 1995, por Brendan Eich, cuando trabajaba en Netscape.
- Inicialmente, se llamó Mocha, luego LiveScript y finalmente, por asuntos de *marketing*... JavaScript.
- Fue estandarizado por Ecma International con el propósito de que todos los navegadores implementaran el mismo lenguaje.
- Este estándar se llama EcmaScript (abreviado ES). Hoy aprenderemos la sexta edición: ES6.

Variables

```python
un_entero_indudablemente_aleatorio = 42
mejor_lenguaje = "python"
mejor_navegador = "firefox"
```

```javascript
let unEnteroIndudablementeAleatorio = 42;
const mejorLenguaje = "python";
const mejorNavegador = "firefox"; // una constante.
```

El doble filo de *weak typing*

```
>>> "El mejor navegador es Firefox"
"El mejor navegador es Firefox"
>>> "El mejor lenguaje es Python" + 3
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

```
>>> "El mejor navegador es Firefox";
"El mejor navegador es Firefox"
>>> "El mejor lenguaje es Python" + 3;
"El mejor lenguaje es Python3"
```

length parte con *len*

```
>>> len("ciclopentanoperhidrofenantreno") # es un hidrocarburo
30
>>> len("¡Los espacios también cuentan!")
30
```

```
>>> "ciclopentanoperhidrofenantreno".length;
30
>>> "¡Los espacios también cuentan!".length;
30
```

for loop

```
>>> lista_de_pi = [3, 1, 4, 1, 5, 9, 2, 6, 5]
>>> for numero in lista_de_pi:
>>>     print(numero)
3
1
[...]
```

```
>>> const listaDePi = [3, 1, 4, 1, 5, 9, 2, 6, 5];
>>> for (var index = 0; index <= listaDePi.length; index++) {
>>>     console.log(listaDePi[index]);
>>> }
3
1
[...]
```

for loop (ES6)

```
>>> lista_de_pi = [3, 1, 4, 1, 5, 9, 2, 6, 5]
>>> for numero in lista_de_pi:
>>>     print(numero)
3
1
[...]
```

```
>>> const listaDePi = [3, 1, 4, 1, 5, 9, 2, 6, 5];
>>> for (const numero of listaDePi) {
>>>     console.log(numero);
>>> }
3
1
[...]
```

if, else if, else

```
>>> if algo_cierto and algo_falso:
>>>     print("Esto no se imprimirá.")
>>> elif algo_falso or algo_cierto:
>>>     print("Esto sí se imprimirá.")
>>> else:
>>>     print("Tengo sueño y se me acabó la imaginación.")
Esto sí se imprimirá.
```

```
>>> if (algoCierto && algoFalso) {
>>>     console.log("Esto no se imprimirá.");
>>> } else if (algoCierto || algoFalso) {
>>>     console.log("Esto sí se imprimirá.");
>>> } else {
        console.log("Y ahora tengo más sueño todavía.");
>>> }
Esto sí se imprimirá.
```

typeof

```
>>> type("Esta oración tiene cinco palabras.")
# <class 'str'>
>>> type(42)
# <class 'int'>
>>> type(3.14)
# <class 'float'>
>>> type(False)
# <class 'bool'>
```

```
>>> typeof("Pero esta tiene cuatro.");
"string"
>>> typeof(42);
"number"
>>> typeof(3.14);
"number"
>>> typeof(true);
"boolean"
```

arrays

```
>>> planetas = ["Mercurio", "Venus", "Tierra", "Marte"]
>>> planetas.append("Plutón")
>>> planetas
["Mercurio", "Venus", "Tierra", "Marte", "Plutón"]
>>> planetas.pop() # buen intento, Plutón
"Plutón"
>>> planetas
["Mercurio", "Venus", "Tierra", "Marte"]
```

```
>>> let planetas = ["Mercurio", "Venus", "Tierra", "Marte"];
>>> planetas.push("Plutón");
>>> planetas
["Mercurio", "Venus", "Tierra", "Marte", "Plutón"]
>>> planetas.pop();
"Plutón"
>>> planetas
["Mercurio", "Venus", "Tierra", "Marte"]
```

indexing & slicing

```
>>> otros_planetas = ["Júpiter", "Saturno", "Urano", "Neptuno"]
>>> otros_planetas[0]
["Júpiter"]
>>> otros_planetas[-1]
["Neptuno"]
>>> otros_planetas[1:3]
["Saturno", "Urano"]
>>> otros_planetas[2:]
["Urano", "Neptuno"]
>>> otros_planetas[-3:]
["Saturno", "Urano", "Neptuno"]
```

```
>>> let otrosPlanetas = ["Júpiter", "Saturno", "Urano", "Neptuno"];
>>> otrosPlanetas[0];
["Júpiter"]
>>> otrosPlanetas[otrosPlanetas.length - 1];
["Neptuno"]
>>> otrosPlanetas.slice(1, 3);
["Saturno", "Urano"]
>>> otrosPlanetas.slice(2);
["Urano", "Neptuno"]
>>> otrosPlanetas.slice(-3);
["Saturno", "Urano", "Neptuno"]
```

objects

```
>>> ether = {
>>>     'symbol': 'ETH',
>>>     'name': 'Ether',
>>>     'usd_value': 380
>>> }
>>> ether['usd_value']
380
```

```
>>> ether = {
>>>     symbol: 'ETH',
>>>     name: 'Ether',
>>>     usdValue: 380,
>>> }
>>> ether.usdValue
380
```

objects

```
>>> for key in ether:
>>>     print(key, ether[key])
symbol ETH
name Ether
usd_value 380
```

```
>>> for (let property in ether) {
>>>    console.log(property, ether[property])
>>> }
symbol ETH
name Ether
usd_value 380
```

objects

```
>>> nebcoin = {
>>>     'symbol': 'NEC',
>>>     'name': 'Nebcoin',
>>>     'usd_value': 42
>>> }
>>> currencies = [ether, nebcoin]
```

```
>>> nebcoin = {
>>>     symbol: 'NEC',
>>>     name: 'Nebcoin',
>>>     usdValue: 42,
>>> }
>>> const currencies = [ether, nebcoin];
```

objects

```
>>> nebcoin = {
>>>     'symbol': 'NEC',
>>>     'name': 'Nebcoin',
>>>     'usd_value': 42
>>> }
>>> currencies = [ether, nebcoin]
```

```
>>> nebcoin = {
>>>     symbol: 'NEC',
>>>     name: 'Nebcoin',
>>>     usdValue: 42,
>>> }
>>> const currencies = [ether, nebcoin];
```

destructuring (ES6)

```
>>> mejor_equipo = ('Palestino', 2, 'Municipal de La Cisterna')
>>> nombre, títulos, estadio = mejor_equipo
>>> estadio
"Municipal de La Cisterna"

>>> nombre, *el_resto = mejor_equipo
>>> el_resto
[2, "Municipal de La Cisterna"]
```

```
>>> const mejorEquipo = ['Palestino', 2, 'Mun. de La Cisterna'];
>>> let [nombre, títulos, estadio] = mejorEquipo;
>>> títulos;
2

>>> [nombre, ...elResto] = mejorEquipo;
>>> elResto
[2, 'Mun. de La Cisterna']
```

functions

```
>>> def suma_cinco(numero):
>>>     resultado = numero + 5
>>>     return resultado
>>> suma_cinco(5)
10
```

```
>>> function sumaCinco(numero) {
>>>    const resultado = numero + 5;
>>>    return resultado;
>>> }
>>> sumaCinco(5);
10
```

arrow functions

```
>>> def suma_cinco(numero):
>>>     resultado = numero + 5
>>>     return resultado
>>> suma_cinco(5)
10
```

```
>>> const sumaCinco = (numero) => {
>>>    const resultado = numero + 5;
>>>    return resultado;
>>> };
>>> sumaCinco(5);
10
```

arrow functions

```
>>> def suma_cinco(numero):
>>>     resultado = numero + 5
>>>     return resultado
>>> suma_cinco(5)
10
```

```
>>> const sumaCinco = (numero) => numero + 5;
>>> sumaCinco(5);
10
```

functional programming & callbacks

```
>>> const pokedex = ["Bulbasaur", "Ivysaur", "Venasaur",
                     "Charmander", "Charmeleon", "Charizard"];

// Llamo a una función por cada elemento del arreglo.
>>> pokedex.forEach(name => {
>>>     console.log(name);
>>> });
"Bulbasaur"
"Ivysaur"
...

// Y también, si quiero, puedo recibir cada índice.
>>> pokedex.forEach((name, index) => {
>>>     console.log(`${index + 1}. ${name}`); // template literals
>>> });
"1. Bulbasaur"
"2. Ivysaur"
...
```

functional programming & callbacks

```
>>> pokedex.forEach((name, index) => {
>>>     console.log(`${index + 1}. ${name}`);
>>> });
"1. Bulbasaur"
"2. Ivysaur"
...

// Además, es posible definir la función directamente.
>>> const printName = (name, index) => {
>>>     console.log(`${index + 1}. ${name}`);
>>> };
>>> pokedex.forEach(printName);
"1. Bulbasaur"
"2. Ivysaur"
...
```

functional programming & callbacks

```
// Definimos un arrow function.
>>> const shinyName = name => {
>>>     return `Shiny ${name}`;
>>> };
// Podría ser definido en una misma línea.
// >>> const shinyName = name => `Shiny ${name}`;

// La diferencia es que 'map' devuelve un nuevo arreglo.
>>> const shinies = pokedex.map(shinyName);
>>> shinies;
["Shiny Bulbasaur", "Shiny Ivysaur", "Shiny Venasaur", ...]

// O podríamos escribirlo directamente como función anónima.
// >>> const shinies = pokedex.map(name => `Shiny ${name}`);
```

functional programming & callbacks

```
// Definimos otro arrow function.
>>> const startsWithCharm = name => {
>>>     return name.startsWith('Charm');
>>> };
// Podría ser definido en una misma línea.
// >>> const startsWithChar = name => name.startsWith('Charm');

>>> startsWithChar('Charmander');
true
>>> startsWithChar('Charizard');
false

// 'filter' devuelve un nuevo arreglo con
// los elementos que cumplen la condición.
>>> const charming = pokedex.filter(startsWithChar);
>>> charming;
["Charmander", "Charmeleon"]
```

¿Qué es el DOM?

Veamos de qué se trata el Document Object Model.

- El DOM es una interfaz que, a través de una representación estructurada, permite acceder y manipular un documento HTML.
- Esta representación se modela como un árbol, en donde cada nodo es un objeto del documento.

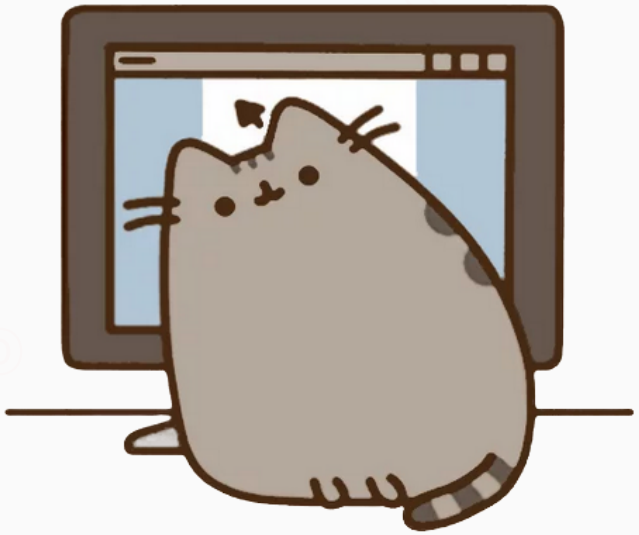
Ahora, a practicar...

Para seguir aprendiendo...

- ”Eloquent JavaScript” por Marijn Haverbeke
- ”You don’t know JS” por Kyle Simpson

Algunas herramientas útiles

- Los *developer tools* de cada navegador
- Referencia de JavaScript del MDN
- Tabla de compatibilidad



Muchas gracias

Licencia

El contenido de estos *slides* está bajo una licencia Creative Commons 4.0 – *attribution, no derivatives*.