Introducción a lógica de programación

DEV.F::
DESARROLLAMOS(PERSONAS);

¿Qué es lógica?

Lógica

- Método o razonamiento en el que las ideas se manifiestan o desarrollan de forma coherente y sin que haya contradicciones entre ellas.

¿Y por qué tengo que saber lógica?

el programador cuenta con un conjunto de herramientas y lenguajes para construir la solución

PROGRAMADOR

HERRAMIENTAS Y LENGUAJES

PROBLEMA

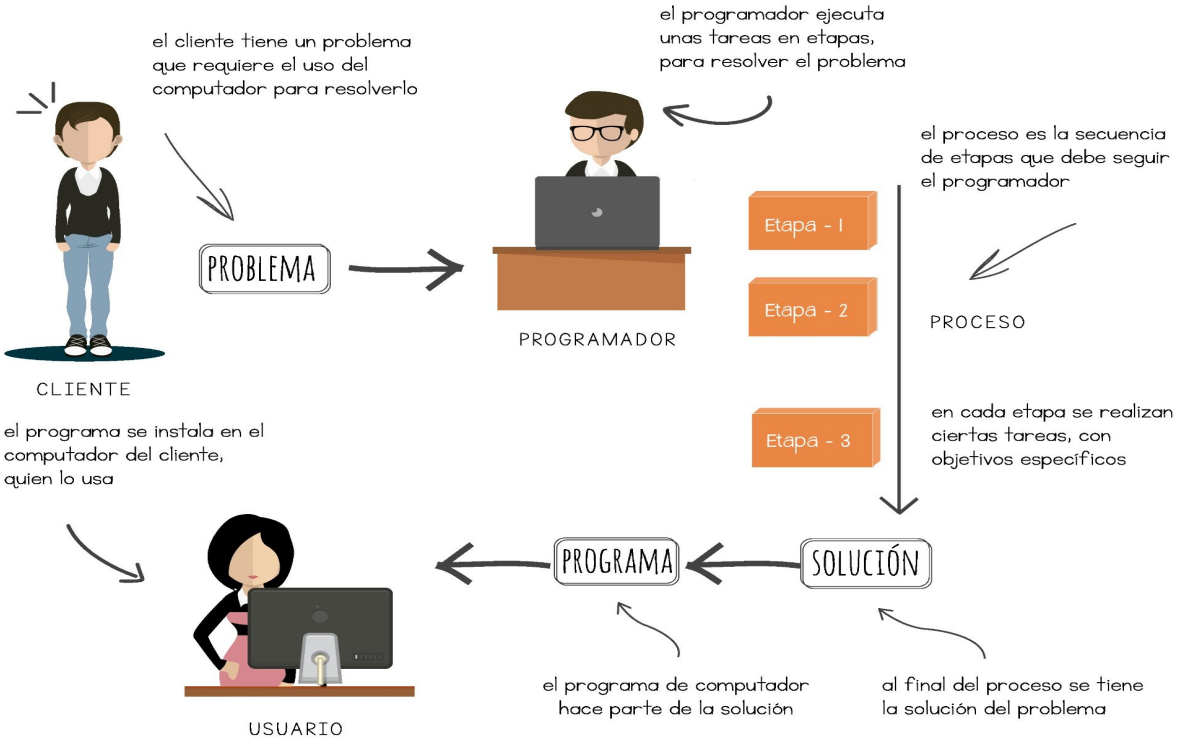
análisis del probema

diseño de la solución

construcción de la solución

proceso

SOLUCIÓN


el cliente tiene un problema
que requiere el uso del
computador para resolverlo
el programador ejecuta
unas tareas en etapas,
para resolver el problema
el proceso es la secuencia
de etapas que debe seguir
el programador
PROBLEMA
Etapa - 1
Etapa - 2
Etapa - 3
PROCESO
CLIENTE
PROGRAMADOR
en cada etapa se realizan
ciertas tareas, con
objetivos específicos
el programa se instala en el
computador del cliente,
quien lo usa
PROGRAMA
SOLUCIÓN
USUARIO
el programa de computador
hace parte de la solución
al final del proceso se tiene
la solución del problema

Fases del análisis de un problema

1. Definición.
2. Análisis (entradas, proceso, salidas).
3. Diseño de un algoritmo.
4. Transformación del algoritmo en código.
5. Ejecución y validación.
6. Pruebas de aceptación.

Fases del análisis de un problema

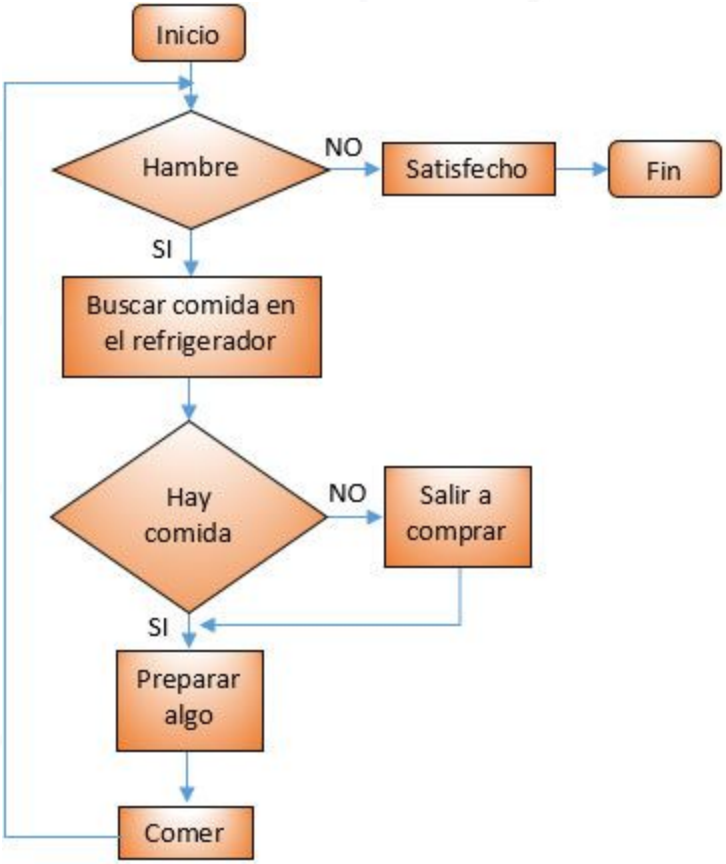
- Definición.
- Análisis (entradas, proceso, salidas).

Lenguaje natural

Fases del análisis de un problema

- Diseño de un algoritmo.

Gráficos (diagrama de flujo) y notas propias (pseudocódigo).



Fases del análisis de un problema

- Transformación del algoritmo en código.
- Ejecución y validación.

Lenguaje Natural

El lenguaje natural es la lengua que usan los individuos para interactuar a través de alguna forma de comunicación sea escrita, oral o no verbal.

Práctica 1

- Analizar y diseñar la solución para el siguiente problema:

  Definición: ¿Que tengo que hacer para ir al cine a ver Dr Strange?

Práctica 2

- Definir, analizar y diseñar la solución para el siguiente problema:

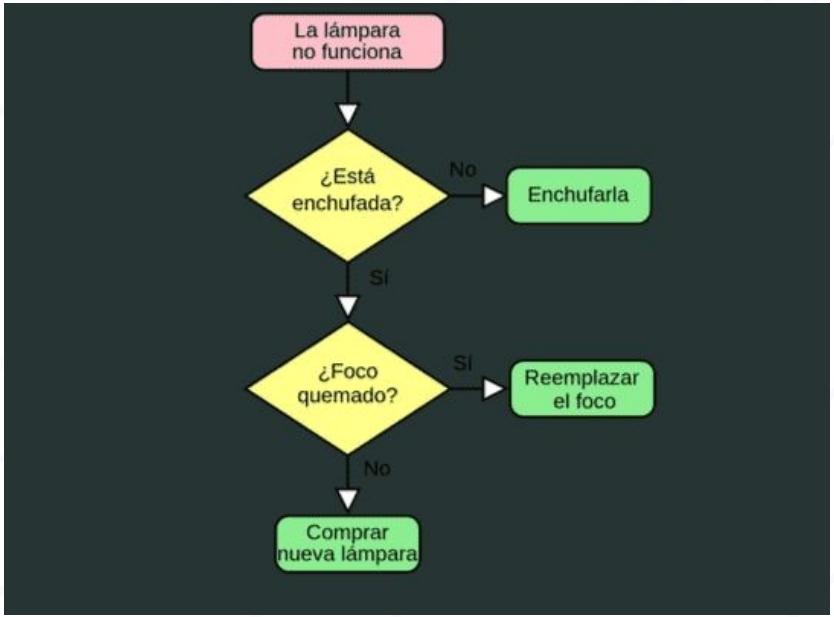
“Quiero una calculadora de áreas y perímetros para figuras”.

Algoritmos

DEV.F::
DESARROLLAMOS(PERSONAS);

Algoritmo

- Un algoritmo es un conjunto de acciones que especifican la secuencia de operaciones realizar, en orden, para resolver un problema.



Partes de un algoritmo

- **Entrada:** Se trata del conjunto de datos que el algoritmo necesita como insumo para procesar.
- **Proceso:** Son los pasos necesarios aplicados por el algoritmo a la entrada recibida para poder llegar a una salida o resolución del problema.
- **Salida:** Es el resultado producido por el algoritmo a partir del procesamiento de la entrada una vez terminada la ejecución del proceso.

Partes de un algoritmo

Entrada: Son los datos que se le dan al algoritmo

```
a = 1;
b = 2;
```

Proceso: Son las operaciones que se hacen con los datos

```
Suma = a + b;
```

Salida: Es el resultado final que se obtiene de las operaciones,

en este caso será 3

```
document.write(Suma)
console.log(Suma)
```

Tipos de salidas algoritmo

Todos los algoritmos tiene un fin, pero el resultado final de ese algoritmo

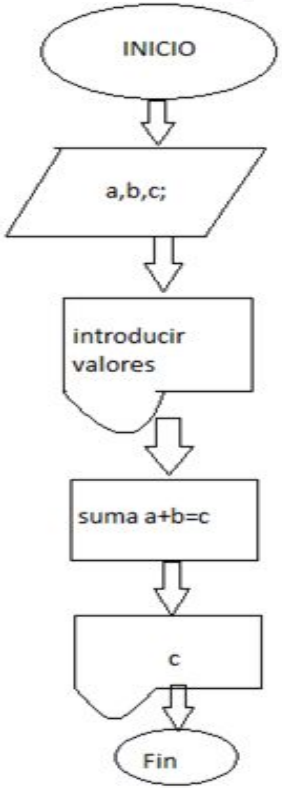
puede ser de tres tipos: algo que recibimos de retorno, algo que se muestra en pantalla o simplemente acción.

Características de un algoritmo

- **Exactitud:** Tiene que indicar un orden claro de la ejecución de cada paso, estos no pueden ser ambiguos.
- **Definido:** Si se realiza la ejecución de un mismo algoritmo en distintas instancias utilizando la misma entrada, debe resultar en la misma salida.
- **Completo:** En la solución se deben considerar todas las posibilidades del problema.
- **Finito:** Necesariamente un algoritmo debe tener un número finito de pasos.
- **Instrucciones entendibles:** Las instrucciones que lo describen deben ser claras y legibles.
- **General:** Debe poder abarcar problemas de un mismo tema soportando variantes del mismo.

Formas de representar un algoritmo

DDF




```
algoritmo Sumar

variables
  entero a, b, c

inicio
  escribir( "Introduzca el primer número (entero): " )
  leer( a )
  escribir( "Introduzca el segundo número (entero): " )
  leer( b )
  c ← a + b
  escribir( "La suma es: ", c )
fin
```

Pseudocódigo

Lenguajes
De
programación

DEV.F::

Práctica 3

Realizar la solución al problema de calcular el valor factorial de un determinado número. Utilice:

- Lenguaje natural.

Práctica 4

Realizar la solución al problema de calcular la serie de fibonacci de un determinado número Utilice:

- Lenguaje natural.

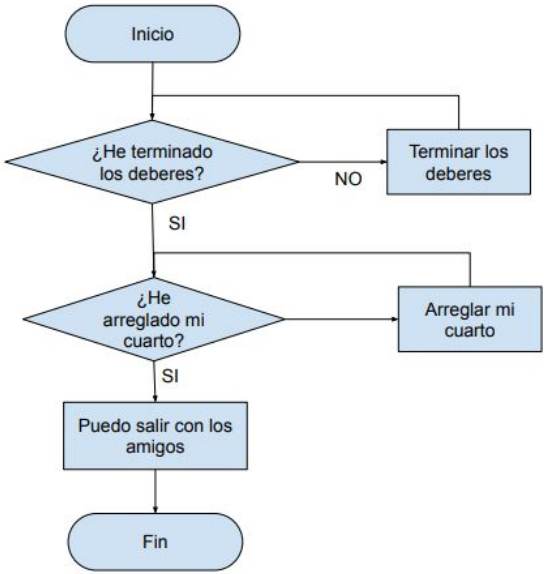
Diagramas de Flujo

DEV.F::
DESARROLLAMOS(PERSONAS);

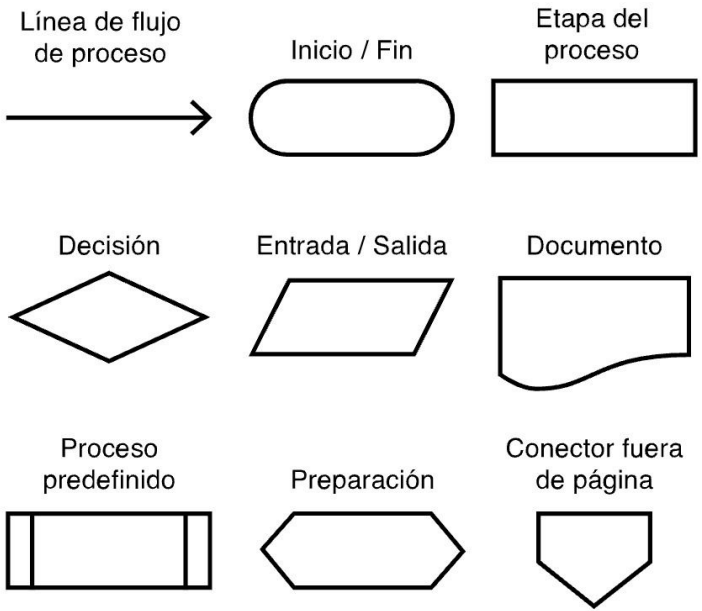
¿Qué es un diagrama de flujo?

Un diagrama de flujo es un esquema que describe un proceso, sistema o algoritmo.

Se usan ampliamente para documentar, planificar, mejorar y comunicar procesos complejos en una representación clara y fáciles de comprender.



Simbología



https://docs.google.com/document/d/1mdrgHLbndoxR9WHMv5dwEXOKud--_o1GJ4F4I8SkqTg/

https://drive.google.com/file/d/0B_mAPkMcdtUsMHgtUWZJMWZvYms/view?resourcekey=0-ITkYE1KVVbh_tCutVSL_ew

Símbolos de diagramas de flujo

Terminal/Terminador

Proceso

Documento

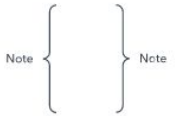
Decisión

Datos o entrada/salida

Datos almacenados

Flecha de flujo

Comentario o anotación



Proceso predefinido

Referencia/conector dentro de la página

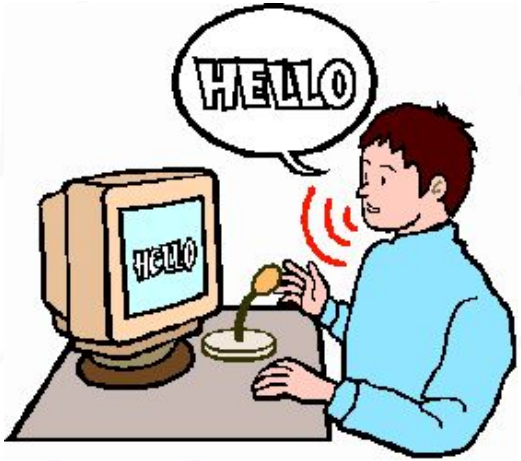
Pseudocódigo

DEV.F:
DESARROLLAMOS(PERSONAS);

¿Qué es un pseudocódigo?

El pseudocódigo es una forma de expresar los distintos pasos que va a realizar un programa, de la forma más parecida a un lenguaje de programación.

Su principal función es la de representar por pasos la solución a un problema combinando palabras entendibles por las personas que usualmente se usan en programación.



Convenciones de pseudocódigo

- “INICIO”
- “Leer”.
- “Si… entonces…”
- “Si no … entonces…”
- “Mientras…”
- “Si y sólo si --- entonces”
- “Imprimir”
- “FIN”

Ejemplo de pseudocódigo

INICIO

Solicita Tipo Figura

Guarda tipo en var1

Solicita Magnitud a calcular

Guarda magnitud en var2

Solicita dimensiones (L, l, a, b… etc…)

SI Área y Cuadrado ENTONCES formula = L * L

// Si y solo si magnitud es Área y la figura Cuadrado la formula = L * L

SI Perimetro y Cuadrado ENTONCES formula = 4 * L

…

Guardar resultado de la formula

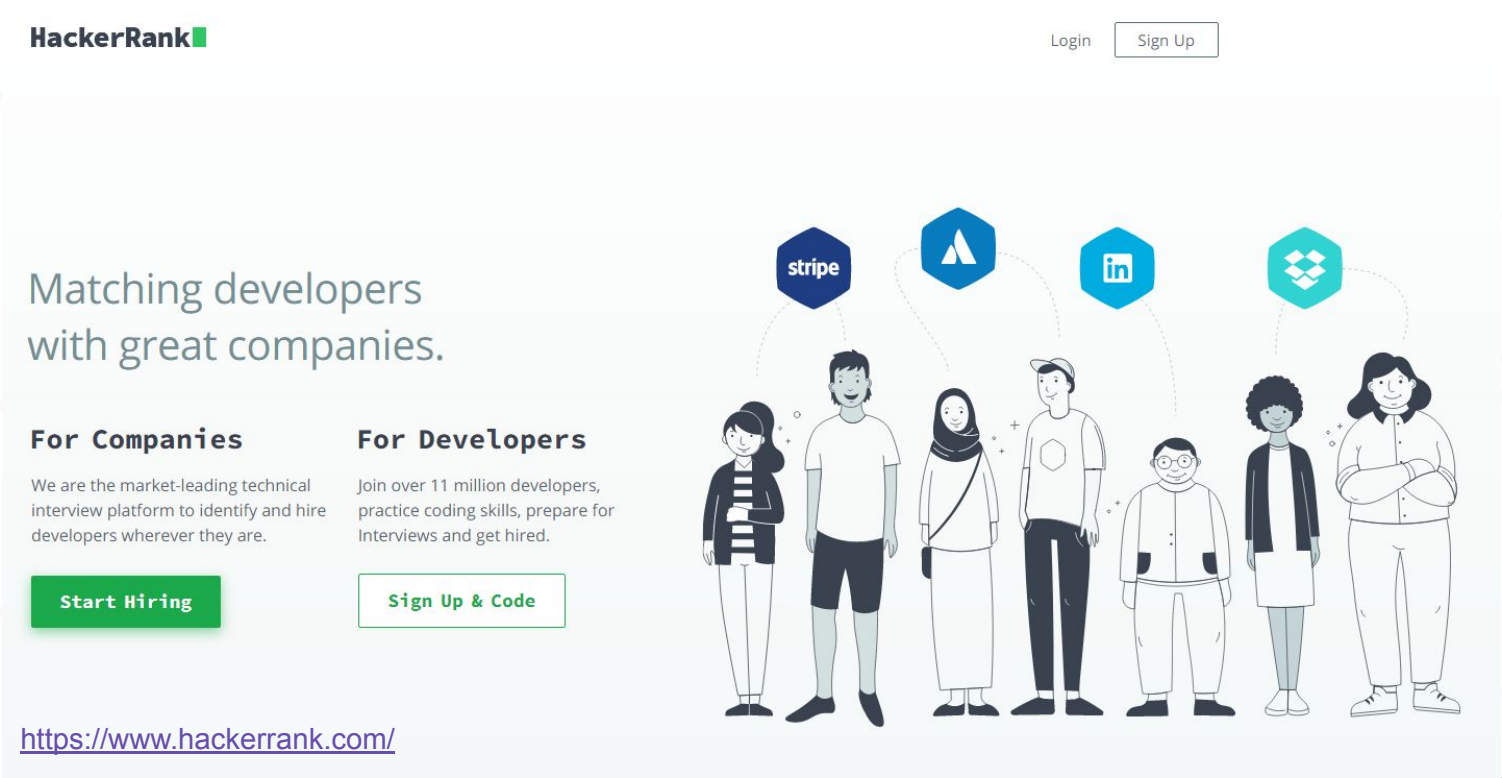
Imprime resultado

FIN

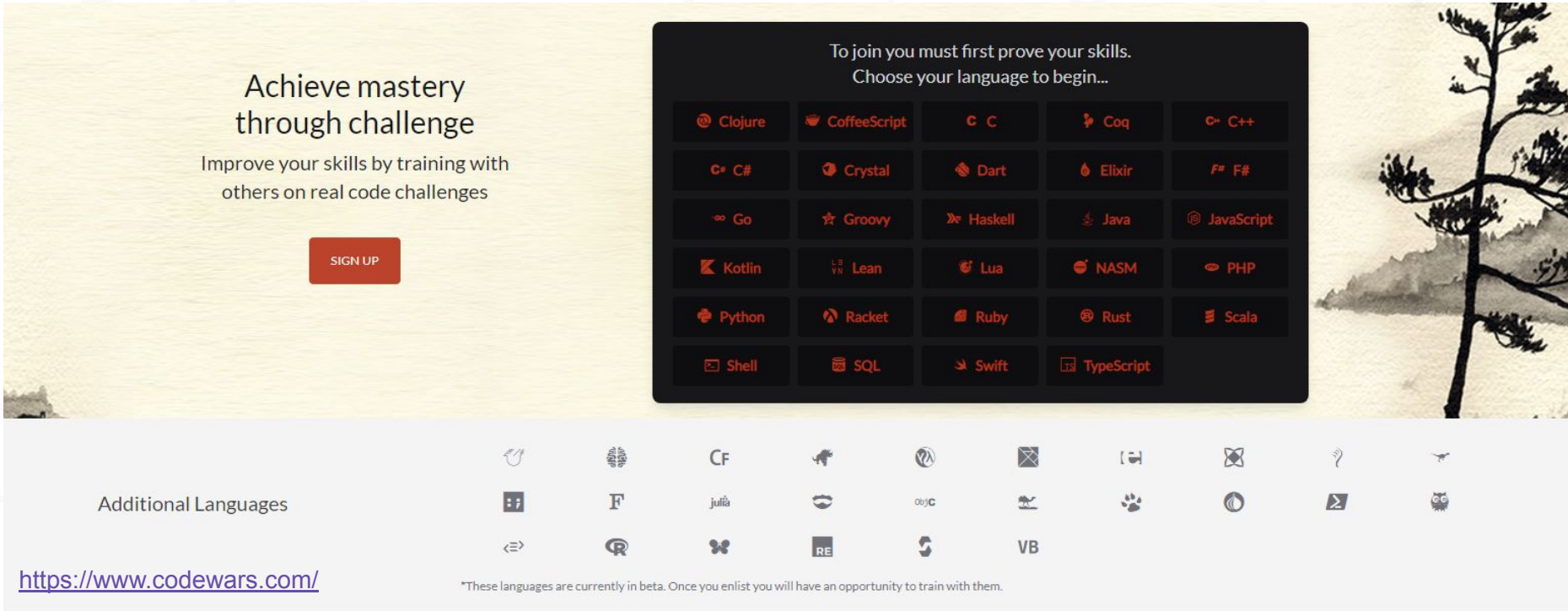
¿Cómo mejorar mi lógica?

DEV.F:
DESARROLLAMOS(PERSONAS);

Plataformas para estimular la lógica



https://www.hackerrank.com/



https://www.codewars.com/

Tips

¿Cómo puedo mejorar mi lógica de programador?

1. **Intenta entender el código de otros programadores.** Cuando trabajes en equipo o aprendas por tu cuenta. Al ver el código de otros programadores, descubres formas distintas de solucionar un mismo problema, formas que quizás a ti nunca se te hubiera ocurrido.

2. **Resuelve algoritmos.** Un algoritmo es la forma que tenemos para solucionar un problema sin la necesidad de programarlo. Soluciona tus ejercicios de programación en un papel, antes de intentar llevarlo a código.

3. **Aprende pseudocódigo.** El pseudocódigo es una forma de estructurar tu algoritmo en un lenguaje intermedio. No llega a ser un lenguaje de programación, pero tampoco es el lenguaje común con el que hablamos todos los días.

4. **Programa mucho.** No intentes desarrollar de frente programas grandes; sino, empieza desarrollando cientos de ejercicios pequeños.

Práctica 5

Realizar la solución al problema de calcular el valor factorial de un determinado número. Utilice:

- Lenguaje natural.
- Diagrama de flujo.
- Pseudocódigo.
- Código.
- Prueba de escritorio.

Práctica 6

Realizar la solución al problema de calcular la serie de fibonacci de un determinado número Utilice:

- Lenguaje natural.
- Diagrama de flujo.
- Pseudocódigo.
- Código.
- Prueba de escritorio.

Práctica 7

Realizar una calculadora que realice las 4 operaciones aritméticas con 2 números. Utilice:

- Lenguaje natural.
- Diagrama de flujo.
- Pseudocódigo.
- Código.
- Prueba de escritorio.

Práctica 8

Realizar una calculadora que suma, resta y multiplica dos matrices cuadradas. Utilice:

- Lenguaje natural.
- Diagrama de flujo.
- Pseudocódigo.
- Código.
- Prueba de escritorio.

ECMAScript

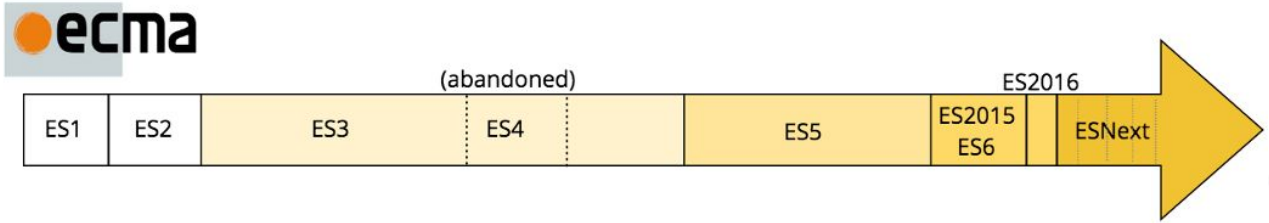
DEV.F:.
DESARROLLAMOS(PERSONAS);

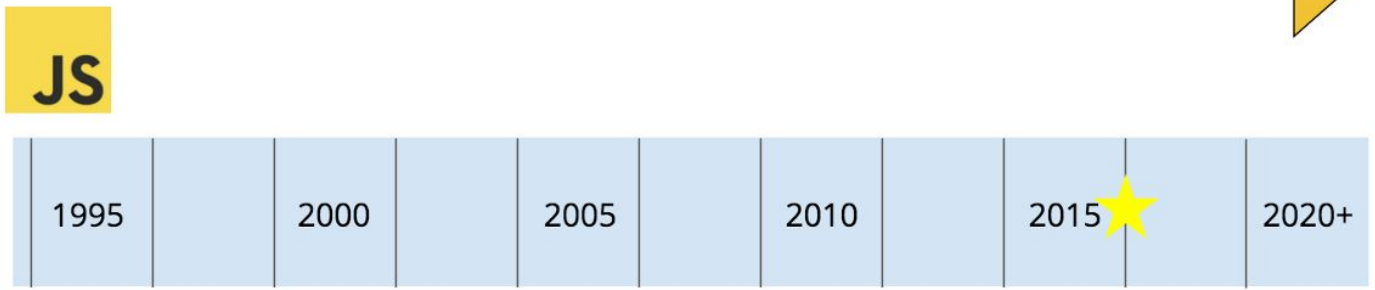
¿Qué es ECMA?

Ecma International es una organización internacional basada en membresías de estándares para la comunicación y la información. Adquirió el nombre Ecma International en 1994, cuando la European Computer Manufacturers Association (ECMA) cambió su nombre para expresar su alcance internacional.

Fue fundada en 1961 para estandarizar los sistemas informatizados en Europa.

ECMAScript


ecma
(abandoned)
ES2016
ES1
ES2
ES3
ES4
ES5
ES2015
ES6
ESNext


JS
1995
2000
2005
2010
2015
2020+


iOS

ECMAScript

Es el estándar que la empresa del mismo nombre definió para JavaScript en el año 2015 (ES6) y encarga de regir como debe ser interpretado y cómo debe funcionar el lenguaje JavaScript.

Versión	Nombre	Publicación	Guía
ES11	ES2020	Junio del 2020	-
ES10	ES2019	Junio del 2019	-
ES9	ES2018	Junio del 2018	-
ES8	ES2017	Junio del 2017	-
ES7	ES2016	Junio del 2016	-
ES6	ES2015	Junio del 2015	-
ES5.1	ES5.1	Junio del 2011	-

Características más utilizadas de ECMA

- Let y const.
- Arrow functions (short and long).
- For/of y for/in.
- Map vs forEach.
- Clases.
- Promises.
- Default parameters.
- Spread operator.
- Rest operator.

Características más utilizadas de ECMA

- Object entries y values.
- JS Modules.
- Literal templates / Template strings.
- Map y Set. => Estructuras de datos.
- Promises.

Características más utilizadas de ECMA

- Async / await.
- Asynchronous Iteration.
- Promise Finally.
- Object Rest Properties.
- New RegExp Features (https://regexr.com/).

Arrays

- Map y forEach(diferencias).
- Reduce.
- Push.
- Splice.
- Includes.
- Filter.
- Find.
- FindIndex.

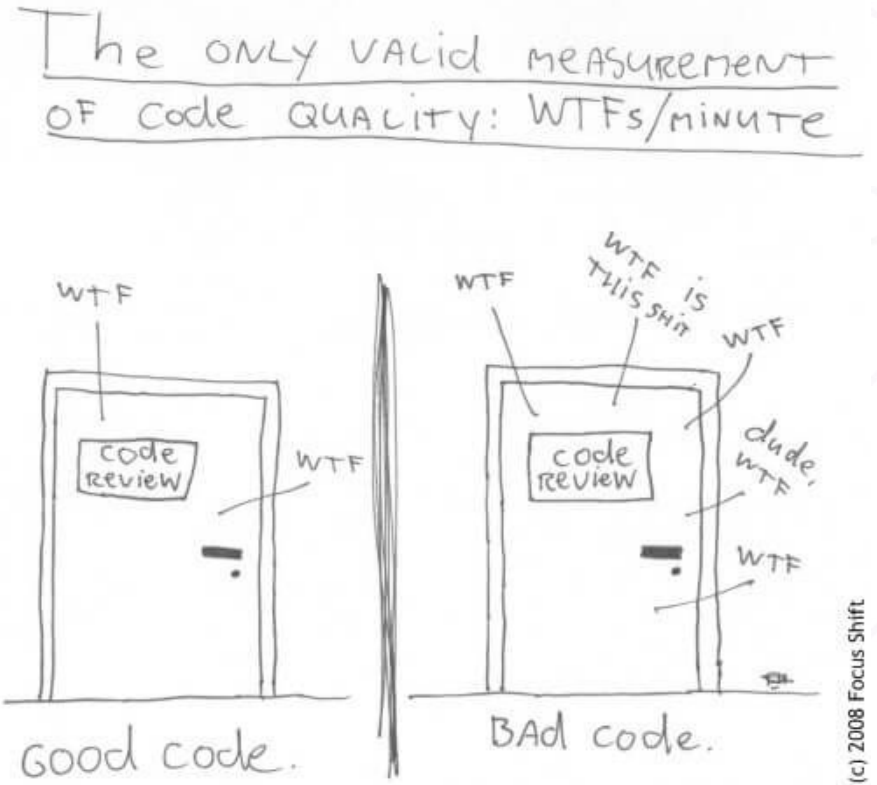
Arrays

```
[1, 2, 3].push(4) // [1,2,3,4]

[1, 2, 3].pop() // [1,2]

[1, 2, 3].shift() // [2,3]

[1, 2, 3].unshift(0) // [0,1,2,3]

['a', 'b'].concat('c') // ['a','b','c']

['a', 'b', 'c'].join('-') // a-b-c

['a', 'b', 'c'].slice(1) // ['a','b]

['a', 'b', 'c'].indexOf('b') // 1

;['a', 'b', 'c'].includes('c') // true

[3, 5, 6, 8].find((n) => n % 2 === 0) // 6

[2, 4, 3, 5].findIndex((n) => n % 2 !== 0) // 2

[3, 4, 8, 6].map((n) => n * 2) // [6,8,16,12]

[1, 4, 7, 8].filter((n) => n % 2 === 0) // [4,8]

[2, 4, 3, 7].reduce((acc, cur) => acc + cur) // 16

[2, 3, 4, 5].every((x) => x < 6) // true

[3, 5, 6, 8].some((n) => n > 6) // true

[1, 2, 3, 4].reverse() // [4,3,2,1]

[3, 5, 7, 8].at(-2) // 7
```

Bonus

- **Valor por defecto de una variable.**
- **Encadenamiento opcional (cortocircuito ?).**
- Hoisting.
- Tipos de salida de una función.
- Paso por referencia y valor.
- Tablas de verdad y tipos de operadores.
- Tipos de funciones en JS.
- **Delete, type of, instanceof.**
- Destructuring.

Clean code


The ONLY VALID MEASUREMENT OF CODE QUALITY: WTFs/MINUTE
WTF
code review
WTF
Good code.
WTF
WTF IS THIS SHIT
WTF
code review
dude, WTF
WTF
Bad code.
(c) 2008 Focus Shift

Buenas prácticas

- Tener un código (honor) para escribir mi código :v => Eslint, Prettier.
- Nombrado de variables y funciones.
- Reutilización y definición de funciones.
- Inmutabilidad.
- Longitud de línea e identación.
- Uso correcto de var, let y const.
- Documentación.
- Comparaciones de valores null, undefined, vacío (0 y emptyString), NaN, etc.
- Validación de JSON.
- Uso de dummyes y constantes.
- Normalización de datos (patrón selector).

Intro a API’s

DEV.F:
DESARROLLAMOS(PERSONAS);

Lista de API’s

- PokeApi.
- Swapi.
- Marvel Developer.
- Nasa.
- HP-API.
- OpenWeather.

Comunicación entre distintos sistemas

Arquitectura REST