

Tema 2. Fonaments del llenguatge

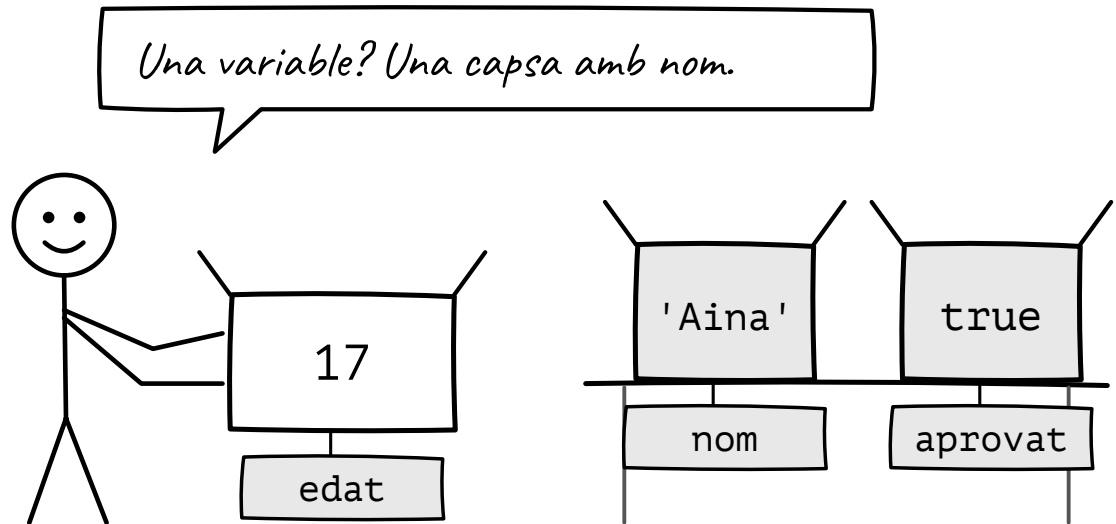


FIG 2.0 Una variable guarda un valor i té un nom per trobar-lo.

Al tema 1 vas muntar el servidor i vas executar dos programes de JavaScript. Aquí aprendràs a escriure'ls tu. Treballaràs sempre igual: **primer prediràs què farà el programa, després l'executaràs**. Si l'encertes, ho has entès. Si no, acabes de trobar el que has de mirar.

Què aprendràs

- Situar JavaScript: d'on surt, què és ECMAScript i on s'executa.
 - Escriure i executar programes amb Node, i llegir els missatges d'error.
 - Guardar valors en variables i constants amb `let` i `const`.
 - Reconèixer els tipus de dades i fer servir els operadors, també `===`.
 - Prendre decisions amb `if`, `else if`, `switch` i l'operador ternari.
 - Repetir amb `for` i `while`, i controlar-ho amb `break` i `continue`.
-

La teva xuleta per a tot el tema

Copia-la o deixa aquesta pàgina marcada. **Fer-la servir no és fer trampa**: els professionals també tenen les seves.

Vull...	Escriu...
Mostrar un valor	<code>console.log(valor)</code>
Un comentari	<code>// això no s'executa</code>
Un valor que canvia	<code>let punts = 0;</code>
Un valor que no canvia	<code>const nom = 'Aina';</code>
Saber el tipus	<code>typeof valor</code>
Comparar	<code>=== i !==</code> . Mai <code>==</code>
Residu d'una divisió	<code>7 % 2</code> dona 1
Posar valors dins d'un text	Accents oberts i <code>\${nom}</code>
Passar un text a número	<code>Number('42')</code>
Decidir	<code>if (...) { } else { }</code>
Repetir un nombre de vegades	<code>for (let i = 1; i <= 5; i++) { }</code>
Repetir mentre passi una cosa	<code>while (...) { }</code>
Sortir d'un bucle	<code>break</code>
Aturar un programa que no acaba	<code>Ctrl + C</code>

1. D'on surt JavaScript

El 1995, l'empresa Netscape tenia el navegador més usat del món i volia que les pàgines web fessin coses, no només mostrar text. Va encarregar un llenguatge a **Brendan Eich**, que en va fer la primera versió en uns deu dies. Es va dir Mocha, després LiveScript i, al final, **JavaScript**.

El nom va ser una decisió de màrqueting: Java estava de moda. **Java i JavaScript no tenen res a veure**, tret del nom.

ECMAScript: les regles del llenguatge

Si cada navegador fes JavaScript a la seva manera, una pàgina funcionaria en un i no en l'altre. Per això el llenguatge es va convertir en un **estàndard**: un document que diu exactament com ha de funcionar. L'estàndard es diu **ECMAScript** i el publica l'organització Ecma International.

Versió	Any	Què va portar
ECMAScript 1	1997	La primera versió de l'estàndard
ECMAScript 5	2009	Mode estricte, JSON
ECMAScript 2015 (ES6)	2015	El gran canvi: <code>let</code> , <code>const</code> , plantilles de text, funcions fletxa
ECMAScript 2016 i següents	Cada juny	Una versió nova cada any, amb canvis petits
ECMAScript 2026	2026	La 17a edició, la més recent

Quan busquis ajuda a internet, mira la data. **Un exemple d'abans de 2015 probablement fa servir** `var`, i en aquest llibre no ho farem.

Els motors

El programa que llegeix JavaScript i l'executa es diu **motor** (*engine*). Cada navegador en porta un.

Motor	On el trobes
V8	Chrome, Edge, Brave i Node.js
SpiderMonkey	Firefox
JavaScriptCore	Safari

Fixa't en l'última fila de V8. **Node.js és el motor de Chrome fora del navegador.** Per això el mateix JavaScript funciona als dos llocs.

Practica 2.1

Omple la taula. Fes servir les dues taules d'aquest apartat i el que vas fer al tema 1.

Pregunta	Resposta
Quin motor fa servir Firefox?	SpiderMonkey
Quin motor fa servir Node.js?	
Qui va crear JavaScript, i quin any?	
Com es diu l'estàndard del llenguatge?	
Quina versió va portar let i const?	
Al tema 1, on s'executava <code>hola.js</code> ?	

HTML, CSS i JavaScript

Una pàgina web es fa amb **tres llenguatges**, i cadascun té la seva feina.

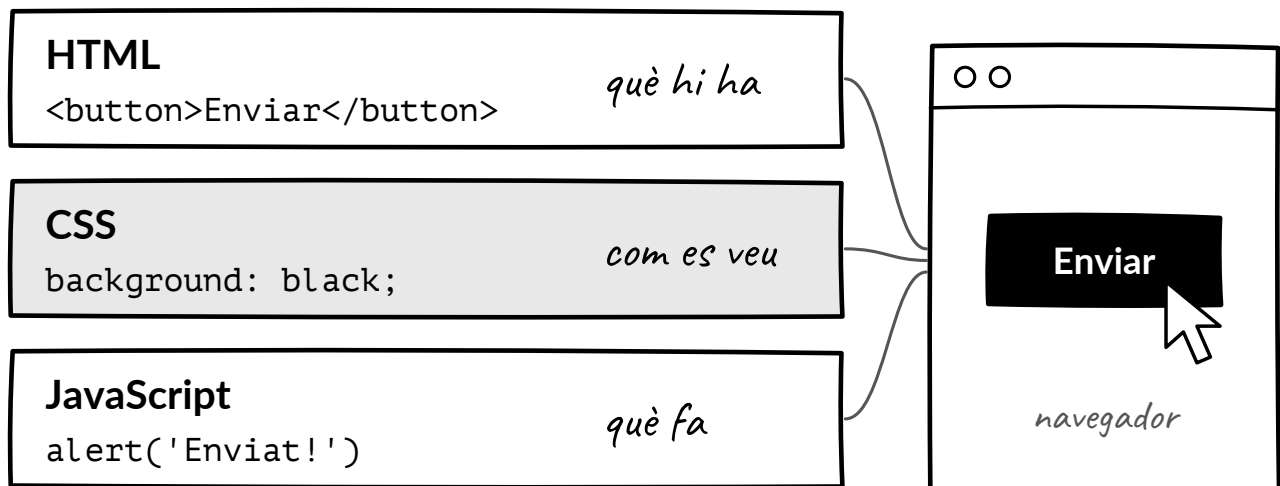


FIG 2.1 Cada llenguatge s'encarrega d'una cosa. Els tres junts fan el botó.

- **HTML** diu **què hi ha** a la pàgina: un títol, un paràgraf, un botó.
- **CSS** diu **com es veu**: colors, mides, posició.
- **JavaScript** diu **què fa**: què passa quan cliques el botó.

En aquest tema no toques HTML ni CSS. Aprens el JavaScript pur, amb Node. Al tema 4 el faràs servir dins les pàgines.

Practica 2.2

Escriu si cada cosa és feina de l'HTML, del CSS o del JavaScript.

Què vols	HTML, CSS o JavaScript?
Que hi hagi un formulari amb dos camps	HTML
Que el títol surti en vermell	
Que en clicar un botó surti un missatge	
Que hi hagi una imatge	
Que es comprovi si el correu té una @ abans d'enviar	
Que el menú quedi a l'esquerra	

La consola del navegador

Tots els navegadors porten unes **eines de desenvolupador** (*DevTools*). S'obren amb **F12**. A la pestanya **Console** pots escriure JavaScript i veure'n el resultat a l'instant. Serveix per provar una línia sense crear cap fitxer.

Guiat 2.3

Obre el navegador del portàtil, prem F12 i ves a la pestanya **Console**. Escriu cada línia i prem Retorn. Apunta què respon.

Escrius	Respon
2 + 3	5
7 * 24	
'Hola' + ' món'	
100 - 58	

2. El primer programa

A partir d'ara, tots els programes del tema van a una carpeta nova del servidor. Connecta VS Code a debian13, obre el terminal i crea-la:

```
mkdir -p ~/projectes/tema2
cd ~/projectes/tema2
```

Obre-la amb *File* → *Open Folder*. Crea aquest fitxer:

primer.js

```
1 // El meu primer programa del tema 2
2 console.log('Hola, 2n SMX');
3 console.log('Som', 24, 'alumnes');
4 console.log(7 * 24);
```

Executa'l:

```
node primer.js
```

```
Hola, 2n SMX
Som 24 alumnes
168
```

Fixa't en quatre coses:

- La línia 1 comença per `//`. És un **comentari** (*comment*): Node se la salta. Serveix per explicar el codi a qui el llegeix.
- Els textos van entre **cometes**: `'Hola, 2n SMX'`. Els números, no.
- `console.log` pot rebre diversos valors separats per comes. Els mostra en una línia, separats per un espai.
- Si li dones un càlcul, com `7 * 24`, mostra el resultat.

Cada línia acaba amb **punt i coma**. JavaScript sovint funciona sense, però en aquest llibre el posem sempre.

Practica 2.4

Aquest és el fitxer `presentacio.js`. **Abans d'executar-lo**, escriu què sortirà.

presentacio.js

```
1 // Mostra dades del grup
2 console.log('Grup:', '2n SMX');
3 console.log('Hores de programació:', 42);
4 // console.log('Aquesta línia no surt');
5 console.log(42 / 21, 'hores per sessió');
```

Quan hi ha un error

Tothom s'equivoca escrivint codi. El que diferencia un programador d'un altre és **com llegeix l'error**. Aquest programa té una cometa de menys a la línia 2:

```
1 console.log('Primera línia');
2 console.log('Hola');
```

sintaxi.js

```
/home/alumne/projectes/tema2/sintaxi.js:2
console.log('Hola');
      ^^^^^^^
```

SyntaxError: Invalid or unexpected token

Node.js v24.21.0

L'error es llegeix així:

1. **On és:** el fitxer i, després dels dos punts, la línia. Aquí, `sintaxi.js:2`.
2. **El tros:** Node copia la línia i marca amb ^ on s'ha perdut.
3. **El tipus i el missatge:** `SyntaxError` vol dir que el codi està mal escrit.

Mira una cosa: **no ha sortit ni** `Primera línia`. Amb un error de sintaxi, Node no executa res, perquè no entén el fitxer.

Aquest altre té el nom de `console` mal escrit:

```
1 console.log('Primera línia');
2 consol.log('Hola');
```

referencia.js

```
Primera línia
/home/alumne/projectes/tema2/referencia.js:2
consol.log('Hola');
^
```

ReferenceError: consol is not defined

Node.js v24.21.0

Ara sí que ha sortit la línia 1. `ReferenceError` vol dir que fas servir un nom que no existeix. El codi està ben escrit, però en arribar a la línia 2 Node no sap què és `consol`, i s'atura.

Si el missatge és molt llarg, sota el tipus d'error surten línies que comencen per `at`. Són el camí intern que ha fet Node. De moment no les cal llegir: amb el fitxer, la línia i el tipus en tens prou.

Practica 2.5

Per a cada missatge, digues quin tipus d'error és i què miraries primer.

Missatge	Tipus	Què miro primer
SyntaxError: Invalid or unexpected token	De sintaxi	Si falta una cometa o un parèntesi
ReferenceError: nmo is not defined		
SyntaxError: missing) after argument list		

3. Variables i constants

Una **variable** (*variable*) és un espai amb nom on guardes un valor per fer-lo servir després. Pensa-hi com en una capsa amb una etiqueta.

variables.js

```

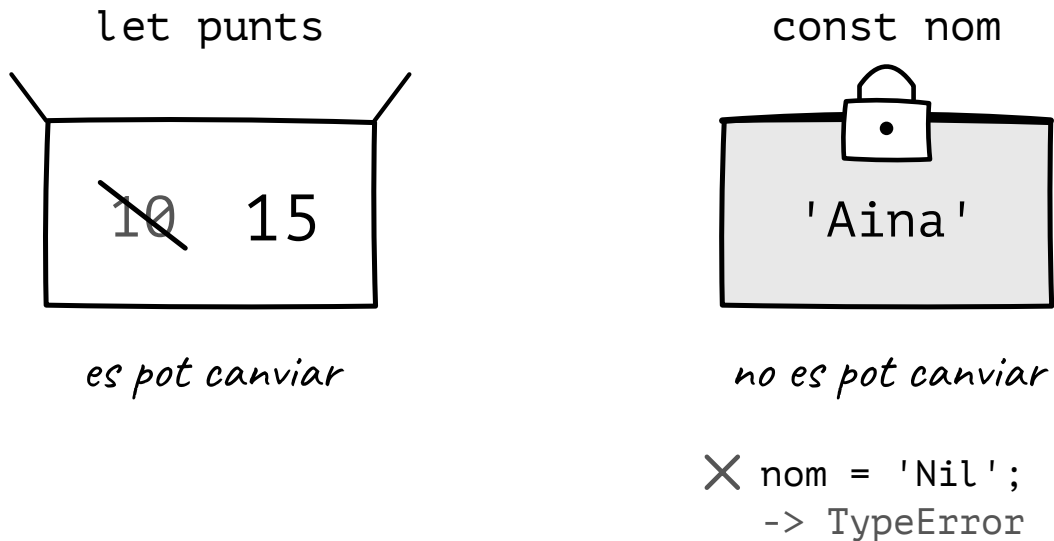
1 let punts = 10;
2 const nom = 'Aina';
3 console.log(nom, 'té', punts, 'punts');
4 punts = punts + 5;
5 console.log(nom, 'té', punts, 'punts');
```

```

Aina té 10 punts
Aina té 15 punts
```

- `let punts = 10;` crea la variable punts i hi guarda 10.

- `const nom = 'Aina'`; crea una **constant** (*constant*): un valor que no canviarà.
- `punts = punts + 5`; es llegeix de dreta a esquerra: **primer** calcula `punts + 5`, que dona 15, i **després** ho guarda a `punts`.

FIG 2.2 Amb `let` pots canviar el contingut. Amb `const`, no.

Si intentes canviar una constant, Node s'atura:

```
1 const iva = 21;
2 console.log('IVA:', iva);
3 iva = 10;
```

constant.js

```
IVA: 21
/home/alumne/projectes/tema2/constant.js:3
iva = 10;
  ^

TypeError: Assignment to constant variable.

Node.js v24.21.0
```

Quin fer servir? Comença sempre amb `const`. Si més endavant veus que el valor ha de canviar, passa-la a `let`.

I `var`? És la manera d'abans de 2015. Encara funciona, però té comportaments estranys que costen errors. La veuràs en exemples vells d'internet. **No la facis servir.**

Els noms

Regla	Bé	Malament
No pot començar per un número	alumne2	2alumne
Sense espais	nomComplet	nom complet
No pot ser una paraula del llenguatge	quantitat	let, const, if
Majúscules i minúscules són diferents	nom i Nom són dues variables	

Quan un nom té diverses paraules, s'ajunten i cada paraula nova comença en majúscula: `preuTotal`, `notaFinal`. Es diu **camelCase**, perquè les majúscules fan geys com un camell. En aquest llibre, a més, els noms van **sense accents**.

Exemple resolt. Taula de traça

Una **taula de traça** apunta què val cada variable després de cada línia. És la manera de seguir un programa amb bolígraf.

		traca.js
1	<code>let a = 3;</code>	
2	<code>let b = 4;</code>	
3	<code>a = a + b;</code>	
4	<code>b = a - b;</code>	
5	<code>console.log(a, b);</code>	

Línia	a	b	Explicació
1	3	—	Es crea a
2	3	4	Es crea b
3	7	4	3 + 4 dona 7, i va a a
4	7	3	7 - 4 dona 3, i va a b. Ja fa servir la a nova
5	7	3	Mostra 7 3

Comprovació: executa-ho. Surt 7 3.

Guiat 2.6

Completa la taula de traça. Les dues primeres files ja hi són.

		guiat.js
1	<code>let x = 2;</code>	
2	<code>let y = x * 5;</code>	
3	<code>x = y - 1;</code>	
4	<code>y = x + y;</code>	
5	<code>console.log(x, y);</code>	

Línia	x	y
1	2	—
2	2	10
3		
4		

Surt:

Practica 2.7

Decideix si cada dada va amb `let` o amb `const`.

Dada	let o const?
El nom d'un alumne	<code>const</code>
Els punts d'un jugador durant la partida	
El percentatge d'IVA	
El nombre de voltes que porta un bucle	
La data de naixement d'una persona	

4. Tipus de dades

Cada valor és d'un **tipus** (*data type*). El tipus diu què pots fer amb el valor: amb dos números pots multiplicar; amb dos textos, no.

Tipus	Què és	Exemples
<code>number</code>	Un número, enter o amb decimals	42, 3.5, -7
<code>string</code>	Un text, entre cometes	'Hola', "2n SMX"
<code>boolean</code>	Vertader o fals. Només té dos valors	<code>true</code> , <code>false</code>
<code>undefined</code>	Encara no té valor	Una variable creada sense valor
<code>null</code>	«Buit» a propòsit	<code>null</code>

Els decimals s'escriuen amb **punt**, no amb coma: 3.5.

L'operador `typeof` diu de quin tipus és un valor:

tipus.js

```
1 let buida;
2 console.log(typeof 42);
3 console.log(typeof 3.5);
4 console.log(typeof 'Hola');
5 console.log(typeof true);
6 console.log(typeof buida);
7 console.log(typeof null);
```

```
number
number
string
boolean
undefined
object
```

L'última línia no és una errada teva: és un error de JavaScript del 1995 que no s'ha corregit mai. `null` **és del tipus** `null`, encara que `typeof` digui `object`.

Practica 2.8

Escriu què mostrarà `console.log(typeof ...)` per a cada valor. Compte amb les cometes.

Valor	typeof
42	number
'42'	
false	
'false'	
0.5	
undefined	

Regla ràpida: **si porta cometes, és un** string, encara que a dins hi hagi un número.

5. Operadors

Aritmètics

Operador	Què fa	Exemple	Resultat
+ - *	Suma, resta, multiplicació	7 * 3	21
/	Divisió	10 / 4	2.5
%	Residu de la divisió	7 % 2	1
**	Potència	2 ** 3	8

El % és el que es fa servir més del que sembla. Si $n \% 2$ dona 0, n és parell. Si $n \% 5$ dona 0, n és múltiple de 5.

Igual que a matemàtiques, **primer** es fan **, **després** *, / i %, i **al final** + i -. Els parèntesis manen sobre tot: $2 + 3 * 4$ dona 14, i $(2 + 3) * 4$ dona 20.

D'assignació

Serveixen per canviar una variable a partir del valor que ja tenia. $n += 3$ és el mateix que $n = n + 3$, i igual amb $--$ i $*=$. $n++$ suma 1 a n , i $n--$ li'n resta 1.

De comparació

Comparen dos valors i sempre donen un boolean: true o false.

Operador	Vol dir	Exemple	Resultat
> <	Més gran, més petit	5 > 3	true
>= <=	Més gran o igual, més petit o igual	5 <= 5	true
===	Igual en valor i en tipus	5 === '5'	false
!==	Diferent en valor o en tipus	5 !== '5'	true
==	Igual després de convertir	5 == '5'	true

== o ===?

== intenta convertir els dos valors al mateix tipus abans de comparar. De vegades encerta i de vegades fa coses que ningú espera:

```
1 console.log(5 === '5');
2 console.log(5 == '5');
3 console.log(0 == '');
4 console.log(null == undefined);
```

comparar.js

```
false
true
true
true
```

Un número 0 i un text buit '' no són la mateixa cosa, però == diu que sí. **Fes servir sempre === i !==**. Així el resultat només depèn del que veus.

Lògics

Uneixen condicions.

- && es llegeix **i**. Dona true quan **les dues** condicions són true.
- || es llegeix **o**. Dona true quan **almenys una** és true.
- ! es llegeix **no**. Dona true quan la condició és false.

logics.js

```
1 const edat = 19;
2 const teCarnet = false;
3 console.log(edat >= 18 && teCarnet);
4 console.log(edat >= 18 || teCarnet);
5 console.log(!teCarnet);
```

```
false
true
true
```

Practica 2.9

Predir abans d'executar. Escriu el resultat de cada expressió. Després comprova-les a la consola del navegador.

Expressió	Resultat
17 % 5	2
3 + 2 * 2	
2 ** 4	
10 === '10'	
4 > 2 && 2 > 4	
12 % 4 === 0	

Practica 2.10

Escriu la condició en JavaScript. La variable ja existeix.

1. n és parell. (Comprovació: $n \% 2 === 0$.)

2. nota està entre 5 i 10, tots dos inclosos.

6. Textos

Un text (*string*) va entre cometes simples '...', dobles "...", o accents oberts. En aquest llibre fem servir les simples, **tret que el text porti un apòstrof**: "S'hi està bé". Si no, JavaScript creuria que el text acaba a l'apòstrof.

Hi ha dues maneres de muntar un text amb valors a dins.

```
1 const nom = 'Nil';
2 const edat = 17;
3 console.log('Em dic ' + nom + ' i tinc ' + edat + ' anys');
4 console.log(`Em dic ${nom} i tinc ${edat} anys`);
5 console.log(`L'any que ve en tindrè ${edat + 1}`);
```

textos.js

```
Em dic Nil i tinc 17 anys
Em dic Nil i tinc 17 anys
L'any que ve en tindrè 18
```

- La línia 3 **concatena** amb +: enganxa trossos. Has de vigilar els espais i les cometes.
- La línia 4 fa servir una **plantilla literal** (*template literal*): el text va entre accents oberts, i cada valor, dins de \${ }.
- A la línia 5, dins de \${ } hi ha un càlcul. També funciona, i l'apòstrof no molesta.

L'accent obert és a la tecla de la dreta de la P. Prem-la i, després, la barra espaiadora.

Practica 2.11

Reescriu cada línia amb una plantilla literal. nom val 'Marta' i preu val 12.

1. console.log('Hola, ' + nom + '!'); (Comprovació: tot el text va entre accents oberts, i \${nom} al lloc on anava el + nom +.)

```
2. console.log(nom + ' ha pagat ' + preu * 2 + ' euros');
```

7. Conversió de tipus

Què passa si sumes un text i un número? JavaScript no s'atura: **converteix** un dels dos. El problema és que no sempre ho fa com esperes.

conversio.js

```
1 console.log('2' + 3);
2 console.log('5' - 2);
3 console.log(Number('42') + 1);
4 console.log(Number('hola'));
```

```
23
3
43
NaN
```

- Amb +, si un dels dos és text, **tot passa a text** i s'enganxa: '2' + 3 dona '23'.
- Amb -, * i /, els textos passen a número: '5' - 2 dona 3.
- Number('42') converteix un text a número a propòsit. És el que has de fer. Si el text no és un número, dona NaN (*Not a Number*).

Tot el que escriu un usuari en un formulari arriba com a text, encara que siguin xifres. Ho veuràs al tema 4.

Practica 2.12

Caçar l'error. Aquest programa hauria de mostrar Total : 23 euros. Els preus arriben d'un formulari, per això són textos.

caixa.js

```
1 const preuCamiseta = '15';
2 const preuGorra = '8';
3 const total = preuCamiseta + preuGorra;
4 console.log('Total: ${total} euros');
```

- Què mostra de veritat?
- Per què? Corregeix la línia 3.

8. Decisions

Fins ara, els programes executaven totes les línies. Amb `if` (si) una part només s'executa quan es compleix una condició.

```

1 const nota = 6.5;
2 if (nota >= 5) {
3   console.log('Aprovat');
4 } else {
5   console.log('Suspès');
6 }

```

nota.js

Aprovat

- La condició va entre **parèntesis**: `(nota >= 5)`.
- El que s'executa si es compleix va entre **claus** `{ }`, i una mica cap a dins. Aquest marge es diu **sagnat** (*indentation*): no canvia res per a Node, però fa que tu vegis on comença i on acaba cada bloc.
- `else` (si no) és opcional. S'executa quan la condició dona fals.

Diversos camins: `else if`

```

1 const nota = 7.5;
2 if (nota < 5) {
3   console.log('Insuficient');
4 } else if (nota < 6) {
5   console.log('Suficient');
6 } else if (nota < 7) {
7   console.log('Bé');
8 } else if (nota < 9) {
9   console.log('Notable');
10 } else {
11   console.log('Excel·lent');
12 }

```

qualificacio.js

Notable

Les condicions es miren **de dalt a baix**, i **només s'executa el primer bloc que es compleix**. Quan arriba a `nota < 6`, ja sap que la nota no és més petita que 5: si ho fos, hauria entrat al primer bloc.

Guiat 2.13

Completar codi. El programa ha de dir si n és parell o senar. Omple els buits.

parell.js

```

1  const n = 7;
2  if (n _____ 2 _____ 0) {
3    console.log(`${n} és parell`);
4  } _____ {
5    console.log(`${n} és senar`);
6  }

```

Amb n igual a 7, surt:

Practica 2.14

Caçar l'error. Un company ha escrit això. Amb una nota de 8, diu Suficient.

mal.js

```

1  const nota = 8;
2  if (nota >= 5) {
3    console.log('Suficient');
4  } else if (nota >= 7) {
5    console.log('Notable');
6  } else {
7    console.log('Insuficient');
8  }

```

Per què no surt mai Notable? Com ho arreglaries?

Molts valors concrets: switch

Quan compares **una sola variable** amb molts valors concrets, `switch` queda més net que una cadena d'`else if`.

```

1  const dia = 'dimecres';
2  switch (dia) {
3    case 'dimecres':
4      console.log('Hi ha programació');
5      break;
6    case 'dissabte':
7    case 'diumenge':
8      console.log('Cap de setmana');
9      break;
10   default:
11     console.log('No hi ha programació');
12 }

```

dia.js

Hi ha programació

- Cada `case` és un valor possible. Dos `case` seguits comparteixen el mateix codi.
- `break` surt del `switch`. **Sense** `break`, **continua executant els** `case` **de sota**, encara que no coincideixin.
- `default` és el que passa si no coincideix cap `case`. És com l'`else`.

Per comparar, `switch` fa servir `===`.

Practica 2.15

Algú ha esborrat tots els `break` de `dia.js`. Què surt ara? Escriu-ho línia a línia.

L'operador ternari

Quan només has de triar entre **dos valors**, hi ha una forma curta:

```

1  const edat = 16;
2  const missatge = edat >= 18 ? 'Pots votar' : 'Encara no pots votar';
3  console.log(missatge);

```

ternari.js

Encara no pots votar

Es llegeix: *condició* ? valor si és `true` : valor si és `false`. Fes-lo servir només per a coses curtes. Si hi ha més d'una línia per camí, torna a `if`.

Practica 2.16

Escriure codi a mà. Tens `const temperatura = 18;`. Escriu un `if` que mostri `Fa fred` si és menor de 10, `S'hi està bé` si és menor de 25, i `Fa calor` en els altres casos. Compte amb l'apòstrof.

1	
2	
3	
4	
5	
6	
7	

Comprovació: amb 18 ha de sortir `S'hi està bé`. Prova també amb 5 i amb 25.

9. Bucles

Un **bucle** (*loop*) repeteix un bloc de codi. Imagina que has de mostrar els números de l'1 al 1000: no escriuràs 1000 línies.

for: un nombre de vegades

comptar.js

```
1 for (let i = 1; i <= 5; i++) {
2   console.log('Volta', i);
3 }
```

```
Volta 1
Volta 2
Volta 3
Volta 4
Volta 5
```

Dins dels parèntesis hi ha tres parts, separades per punt i coma:

Part	Què fa	Quan
<code>let i = 1</code>	Crea el comptador	Una sola vegada , al principi
<code>i <= 5</code>	La condició per fer una volta més	Abans de cada volta
<code>i++</code>	Suma 1 al comptador	Després de cada volta

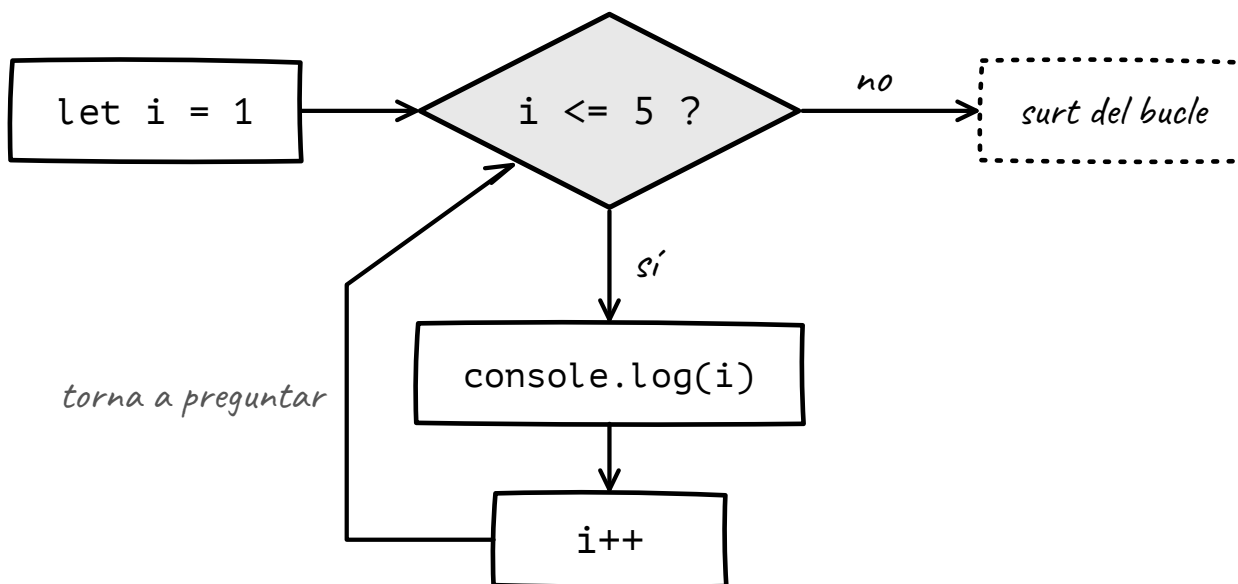


FIG 2.3 Com funciona un for. Quan la condició dona false, surt.

El comptador es diu `i` per costum. Pots posar-hi qualsevol nom.

Exemple resolt. Traça d'un bucle

```

1 let suma = 0;
2 for (let i = 1; i <= 4; i++) {
3   suma = suma + i;
4 }
5 console.log(suma);

```

suma.js

Volta	i	i <= 4?	suma després
1	1	true	0 + 1 = 1
2	2	true	1 + 2 = 3
3	3	true	3 + 3 = 6
4	4	true	6 + 4 = 10
—	5	false	Surt. Mostra 10

Fixa't en l'última fila: **la condició es comprova una vegada més que voltes fa el bucle**. Amb `i` igual a 5 ja no entra.

Practica 2.17

Escriure codi a mà. Escriu un `for` que mostri la taula del 7, amb aquest format:

```
7 x 1 = 7
7 x 2 = 14
...
7 x 10 = 70
```

1
2
3

while: mentre passi una cosa

Quan **no saps quantes voltes** caldran, fes servir `while` (*mentre*). Repeteix mentre la condició sigui `true`.

estalvi.js

```
1 let estalvi = 0;
2 let setmanes = 0;
3 while (estalvi < 100) {
4   estalvi = estalvi + 30;
5   setmanes++;
6 }
7 console.log(`${setmanes} setmanes, ${estalvi} euros`);
```

Aquí estalvies 30 euros cada setmana fins a arribar a 100. No saps d'entrada quantes setmanes seran: ho diu el bucle.

Practica 2.18

Predir abans d'executar. Què mostra `estalvi.js`? Fes les voltes a mà, en la taula.

Volta	estalvi després	setmanes després	estalvi < 100?
1	30	1	true
2			
3			
4			

Surt:

Un `while` mira la condició **abans** de cada volta. Si d'entrada ja és `false`, no en fa cap.

Els bucles infinits

Si la condició no arriba mai a ser false, el bucle no acaba. El terminal es queda escrivint sense parar, o penjat. **Atura'l amb Ctrl + C**, com el servidor del tema 1.

Practica 2.19

Caçar l'error. Aquest programa hauria de mostrar 1, 2 i 3. No l'executis sense haver-lo mirat abans.

```
1 let i = 1;
2 while (i <= 3) {
3   console.log(i);
4 }
```

infinif.js

- Què passa quan l'executes?
- Quina línia falta i on va?

10. Controlar el bucle: break i continue

- break **surt del bucle** de cop, encara que la condició encara es compleixi.
- continue **salta el que queda d'aquesta volta** i passa a la següent.

```
1 for (let i = 1; i <= 10; i++) {
2   if (i === 4) {
3     break;
4   }
5   console.log(i);
6 }
```

break.js

```
1
2
3
```

```
1 for (let i = 1; i <= 6; i++) {
2   if (i % 3 === 0) {
3     continue;
4   }
5   console.log(i);
6 }
```

continue.js

```
1
2
4
5
```

break va bé quan busques una cosa: quan la trobes, no cal continuar.

Practica 2.20

Predir abans d'executar. Escriu què surt de cada programa, en una línia i separat per comes.

a)

a.js

```
1 for (let i = 1; i <= 5; i++) {
2   if (i % 2 !== 0) {
3     continue;
4   }
5   console.log(i);
6 }
```

Surt:

b)

b.js

```
1 for (let i = 10; i >= 1; i--) {
2   if (i === 7) {
3     break;
4   }
5   console.log(i);
6 }
```

Surt:

El que has de saber sí o sí

- JavaScript el va crear Brendan Eich el 1995. L'estàndard es diu **ECMAScript**, i des del 2015 en surt una versió cada any.
- **V8** és el motor de Chrome i també el de **Node.js**.
- HTML diu **què hi ha**, CSS **com es veu** i JavaScript **què fa**.
- Un error es llegeix així: **fitxer i línia**, el tros marcat amb ^, i el **tipus**: SyntaxError, ReferenceError, TypeError.
- const **per defecte**, let si el valor canvia, var **mai**.
- Tipus: number, string, boolean, undefined, null. **Si porta cometes, és un string**.
- % dona el residu: $n \% 2 === 0$ vol dir parell.
- **Compara sempre amb === i !==**, mai amb ==.

- '2' + 3 dona '23'. Per sumar textos amb xifres, primer Number().
- En un if amb else if, **només s'executa el primer bloc que es compleix**.
- En un switch, **cada case porta el seu break**.
- for quan saps les voltes; while quan no.
- Un bucle que no acaba s'atura amb **Ctrl + C**.

Paraules noves

Paraula	Què és	Les meves notes
ECMAScript	L'estàndard de JavaScript: el document que diu com ha de funcionar.	
Motor (<i>engine</i>)	Programa que executa JavaScript. V8, SpiderMonkey, JavaScriptCore.	
Variable	Espai amb nom on es guarda un valor que pot canviar.	
Constant	Espai amb nom on es guarda un valor que no canvia.	
Tipus (<i>data type</i>)	La mena de valor: número, text, booleà...	
Booleà (<i>boolean</i>)	Valor que només pot ser true o false.	
Operador	Símbol que fa una operació: +, %, ==, &&...	
Plantilla literal (<i>template literal</i>)	Text entre accents oberts amb valors dins de \${ }.	
Bucle (<i>loop</i>)	Bloc de codi que es repeteix.	
Taula de traça	Taula amb el valor de cada variable pas a pas.	

Repàs del tema

A. Vertader o fals

Frase	V o F	Per què
Java i JavaScript són el mateix llenguatge		
Node.js fa servir el mateix motor que Chrome		
const es pot tornar a assignar si el valor és un número		
typeof '7' dona number		
5 === '5' dona true		
Un while fa sempre almenys una volta		

B. Preguntes curtes

1. Per què amb un `SyntaxError` no s'executa cap línia, i amb un `ReferenceError` sí que s'executen les d'abans?

2. Quan faries servir `while` en lloc de `for`?

C. Problemes

Problema 1. Escriu un programa que mostri els números de l'1 al 15. Però si el número és múltiple de 3, ha de mostrar Fizz; si és múltiple de 5, Buzz; i si ho és de 3 i de 5 alhora, FizzBuzz. Pensa en quin ordre han d'anar les condicions.

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

Problema 2. Aquest programa hauria de dir quantes entrades pots comprar amb 50 euros si cada una en val 12. Mostra Pots comprar 5 entrades, però amb 50 euros només en pots comprar 4. Troba l'error i corregeix-lo.

```

1 let diners = 50;
2 let entrades = 0;
3 while (diners >= 0) {
4   diners = diners - 12;
5   entrades++;
6 }
7 console.log(`Pots comprar ${entrades} entrades`);

```

entrades.js

