Referencia Rápida de JavaScript

ES6+, DOM, eventos, Fetch API, async/await

Básico

Variables

```
let name = "Alice";    // reassignable
const PI = 3.14;       // constant
var old = "avoid";     // function-scoped (legacy)
```

Tipos de datos

string	Texto: **"hello"** o **'hello'**
number	Entero o decimal: **42**, **3.14**
boolean	**true** / **false**
null	Valor vacío intencional
undefined	Declarado pero sin asignar
object	Pares clave-valor: **{ a: 1 }**
array	Lista ordenada: **[1, 2, 3]**
symbol	Identificador único

Verificación y conversión de tipos

```
typeof "hello"     // "string"
typeof 42          // "number"
Number("42")       // 42
String(100)        // "100"
parseInt("3.9")    // 3
parseFloat("3.14") // 3.14
```

Strings

Template Literals

```
const name = "Alice";
`Hello, ${name}!`       // Hello, Alice!
`Total: ${2 + 3}`       // Total: 5
`Multi
 line string`
```

Métodos de string

s.length	Número de caracteres
s.toUpperCase()	Copia en MAYÚSCULAS
s.toLowerCase()	Copia en minúsculas
s.trim()	Eliminar espacios iniciales y finales
s.split(",")	Dividir en array
s.includes("x")	Verificar si contiene → bool
s.indexOf("x")	Primer índice (-1 si no existe)
s.slice(1, 4)	Subcadena por índice
s.replace(a, b)	Reemplazar primera coincidencia
s.replaceAll(a, b)	Reemplazar todas las coincidencias
s.startsWith(x)	Verificar prefijo → bool
s.endsWith(x)	Verificar sufijo → bool
s.padStart(n, c)	Rellenar al inicio hasta longitud n

Arrays

Crear y acceder

```
const fruits = ["apple", "banana", "cherry"];
fruits[0]          // "apple"
fruits.length      // 3
fruits.at(-1)      // "cherry"
```

Métodos que mutan

arr.push(x)	Agregar al final
arr.pop()	Eliminar y devolver el último
arr.unshift(x)	Agregar al inicio
arr.shift()	Eliminar y devolver el primero
arr.splice(i, n)	Eliminar n elementos en el índice i
arr.sort()	Ordenar en el lugar (lexicográfico)
arr.reverse()	Invertir en el lugar

Métodos que no mutan

arr.map(fn)	Transformar cada elemento
arr.filter(fn)	Mantener elementos donde fn sea verdadero
arr.reduce(fn, init)	Acumular en un único valor
arr.find(fn)	Primera coincidencia o undefined
arr.findIndex(fn)	Índice de la primera coincidencia (-1)
arr.includes(x)	Verificar si contiene → bool
arr.slice(a, b)	Copia superficial de una porción
arr.join(",")	Unir en string
arr.forEach(fn)	Iterar (sin valor de retorno)
[...a, ...b]	Concatenar arrays (spread)

Objetos

Crear y acceder

```
const user = { name: "Alice", age: 20 };
user.name          // "Alice"
user["age"]        // 20
user.gpa = 3.85;   // add/update
```

Destructuring y spread

```
const { name, age } = user;
const copy = { ...user, age: 21 };
```

Métodos de objeto

Object.keys(o)	Array de claves
Object.values(o)	Array de valores
Object.entries(o)	Array de pares [clave, valor]
Object.assign(t, s)	Copiar propiedades de s a t
"k" in obj	¿Existe la clave? → bool
delete obj.k	Eliminar propiedad
Object.freeze(o)	Hacer inmutable (superficial)

Flujo de control

if / else if / else

```
if (score >= 90) {
  grade = "A";
} else if (score >= 80) {
  grade = "B";
} else {
  grade = "C";
}
```

Ternario y coalescencia nula

```
const status = score >= 60 ? "pass" : "fail";
const name = user.name ?? "Anonymous";
```

switch

```
switch (color) {
  case "red":   stop();   break;
  case "green": go();     break;
  default:      wait();
}
```

Bucles

for / for...of / for...in

```
for (let i = 0; i < 5; i++) { }

for (const item of ["a", "b"]) { }  // arrays

for (const key in obj) { }  // object keys
```

while / do...while

```
while (count < 10) { count++; }

do { count++; } while (count < 10);
```

break y continue

```
for (let i = 0; i < 10; i++) {
  if (i === 5) break;       // stop loop
  if (i % 2 === 0) continue; // skip
}
```

Funciones

Declaración y arrow function

```
function greet(name) {
  return `Hello, ${name}!`;
}
const greet = (name) => `Hello, ${name}!`;
const square = x => x * x; // single param
```

Parámetros por defecto y rest

```
function greet(name = "World") { }

function sum(...nums) {
  return nums.reduce((a, b) => a + b, 0);
}
```

Callbacks

```
[1, 2, 3].map(x => x * 2);    // [2, 4, 6]
[1, 2, 3].filter(x => x > 1); // [2, 3]
setTimeout(() => console.log("done"), 1000);
```

Clases

```
class Dog {
  constructor(name, breed) {
    this.name = name;
    this.breed = breed;
  }
  bark() { return `${this.name} says Woof!`; }
}

class Puppy extends Dog {
  constructor(name, breed, toy) {
    super(name, breed);
    this.toy = toy;
  }
}
```

Manejo de errores

```
try {
  JSON.parse("bad json");
} catch (err) {
  console.error(err.message);
} finally {
  console.log("always runs");
}
```

Lanzar errores

```
throw new Error("Something went wrong");
```

DOM

Seleccionar elementos

```
document.querySelector(".cls")    // first match
document.querySelectorAll("li")   // all matches
document.getElementById("main")
```

Referencia Rápida de JavaScript

Modificar elementos

```
el.textContent = "new text";
el.innerHTML = "<b>bold</b>";
el.style.color = "red";
el.classList.add("active");
el.classList.toggle("hidden");
el.setAttribute("data-id", "42");
```

Eventos

```
btn.addEventListener("click", (e) => {
  console.log(e.target);
});
```

Crear elementos

```
const li = document.createElement("li");
li.textContent = "New item";
ul.appendChild(li);
el.remove(); // remove element
```

Fetch API

Solicitud GET

```
fetch("https://api.example.com/data")
  .then(res => res.json())
  .then(data => console.log(data))
  .catch(err => console.error(err));
```

Solicitud POST

```
fetch(url, {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({ key: "value" }),
});
```

Async / Await

```
async function loadData() {
  try {
    const res = await fetch(url);
    const data = await res.json();
    return data;
  } catch (err) {
    console.error(err);
  }
}
```

Solicitudes en paralelo

```
const [users, posts] = await Promise.all([
  fetch("/users").then(r => r.json()),
  fetch("/posts").then(r => r.json()),
]);
```

Módulos

Exportaciones nombradas

```
// math.js
export const PI = 3.14;
export function add(a, b) { return a + b; }

// main.js
import { PI, add } from "./math.js";
```

Exportación por defecto

```
// logger.js
export default function log(msg) { }

// main.js
import log from "./logger.js";
```