Notas de JavaScript

Introducción

- **JavaScript (JS)**:
  - Lenguaje de scripting utilizado principalmente para el desarrollo web.
  - Basado en ECMAScript, con motores como V8 (Chrome) y SpiderMonkey (Firefox).
  - Multiplataforma, orientado a objetos, y tipado dinámico.
- **Usos**:
  - Desarrollo web, backend (Node.js), y aplicaciones especializadas como el telescopio James Webb.

Características clave

- **Interpretado**: Los navegadores modernos usan compilación Just-In-Time (JIT).
- **Limitaciones**:
  - No puede acceder directamente a funciones del sistema operativo.
  - Restringido por la política Same Origin.

Estructura y sintaxis

- **Incrustación**:
  - Código inline con `<script>...</script>` o externo con `<script src="file.js"></script>`.
  - Atributos como `defer` (ejecución después del DOM completo) y `async` (ejecución independiente).
- **Comentarios**:
  - Línea única: `//`.
  - Multilínea: `/* ... */`.

Variables y constantes

- Declaraciones:
  - `let`: Bloque local y no redeclarable.
  - `const`: Constantes que no cambian.
  - `var`: Alcance de función y permite redeclaración (desaconsejado).
- **Conversión de tipos**:
  - `String()`, `Number()`, `Boolean()`.

Funciones

Declaración:

```
function sayHello() {
  console.log("Hello!");
}
```

Expresión:

```
const sayHello = function() {
  console.log("Hello!");
};
```

Funciones flecha:

```
const sum = (a, b) => a + b;
```

Estructuras de control

- Condicionales: `if`, `else if`, `else`, operador ternario.
- Bucles: `for`, `while`, `do...while`.
- Operadores lógicos: `&&`, `||`, `!`, y `??` (nullish coalescing).

Tipos de datos y objetos

- Tipos básicos: `number`, `string`, `boolean`, `null`, `undefined`.
- Objetos:
    - Sintaxis literal: `{key: value}`.
    - Métodos: `.key` o `["key"]` para acceder a propiedades.

APIs del navegador

- `window`, `document`, eventos (`onclick`, `onload`).
- Manipulación del DOM:
    - Acceso a elementos: `getElementById`, `querySelector`.
    - Cambios en contenido: `.innerHTML`, `.textContent`.

Fetch y Promesas

- **Fetch API**:
    - Simplifica solicitudes HTTP asíncronas.

```
fetch('url')
  .then(response => response.json())
  .then(data => console.log(data));
```

- **Promises**:
    - Mecanismo para manejar operaciones asíncronas.
    - Estados: `pending`, `fulfilled`, `rejected`.

Encadenamiento:

```
promise
```

```
  .then(result => ...)
  .catch(error => ...);
```

Canvas

- Elemento para gráficos:

Configuración básica:

```
let canvas = document.getElementById("myCanvas");
let ctx = canvas.getContext("2d");
```

    - Dibujos: `ctx.fillRect`, `ctx.arc`, `ctx.stroke`.
    - Animaciones: `requestAnimationFrame`.

Recursos útiles

- **Tutoriales**:
    - JavaScript.info
    - W3Schools
- **Referencias**:
    - MDN Web Docs
    - API Fetch
    - Can I Use