HINTS

AXIOS

Axios es una librería de **JavaScript** que facilita todo tipo de operaciones con cliente HTTP. Está basada en promesas y, en comparación a **JQuery**, es mucho más liviana. Es compatible con **NodeJS** y con todos los navegadores en versiones actuales. Para seleccionar el uso de **JQuery**, **Fetch** o **Axios** habrá que analizar con detenimiento en qué situación nos encontramos, que deseamos hacer, si utilizamos algún *framework* de **JavaScript** o no, etc.

REST V/S RESTful

Aunque muchas veces se usan como sinónimos, lo cierto es que no lo son: **REST** (*Representational State Transfer*) es una arquitectura que se ejecuta sobre **HTTP**, mientras que **RESTful** hace referencia a un servicio web que implementa la arquitectura **REST**.

ASYNC/AWAIT

Sintaxis especial para trabajar con promesas de una forma más confortable. La palabra “async” indica que una función siempre devolverá una promesa, asegurándose de que la función devuelva la promesa o envolviendo las no promesas y transformándolas en una. “Await” por su parte, solo trabaja dentro de las funciones async, haciendo que **JavaScript** espere hasta que la promesa responda y devuelve su resultado.

Es importante destacar que si tratamos de usar await en funciones no async tendremos un error de sintaxis.

```
function resolveAfter2Seconds() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve('resolved');
    }, 2000);
  });
}

async function asyncCall() {
  console.log('calling');
  const result = await resolveAfter2Seconds();
  console.log(result);
  // expected output: "resolved"
}

asyncCall();
```

DIRECCIONES O MÉTODOS DE LA API

Cuando consumimos una **API** debemos procurar revisar la documentación que esta nos entrega, en la cual nos informarán la dirección a la cual debemos apuntar para acceder a ciertos servicios, esto quiere decir, nos mostrarán la **URL** exacta a la que debemos ingresar si queremos realizar un **POST** o un **DELETE**, entre otros.

COMILLAS ESPECIALES

También conocidas como comillas invertidas, se incorporaron con la especificación **ES2015**, dando la posibilidad de crear lo que se conoce como "*template literals*" o "*template strings*", que consiste en encapsular texto entre comillas invertidas. Uno de sus usos más extendidos y popularizados es el de concatenar String de modo que el código quede mucho más limpio.

La sintaxis antigua sería:

```
const datosUsuario = 'nombre:' + nombre + '' + apellido ;
```

La sintaxis usando las comillas especiales sería:

```
const datosUsuario = ` nombre : $ { nombre } $ { apellido } ` ;
```

Además, nos permite incorporar el código **HTML** de manera sencilla.

```
const html = `<article>
<h1>Titulo del Articulo</h1>
</article>`;
```

INTERPOLACIÓN

Continuando con el consejo anterior, el uso de estas comillas especiales y la sintaxis **${variable}** se conoce como interpolación. Para que este tipo de concatenación más legible se puede llevar a cabo siempre necesitará:

```
` $ { variable } `
```

Quedando de la siguiente forma:

```
alert ( ` Se Estima Que La Tierra Tiene $ { Edad } MIL Millones de años . ' );
```