## Interviews

* Javascript is event driven (dirigido por eventos)
* Función IIFE

(function explainVar(){})()

IIFE (Inmediately Invoked Function Expression)

* ¿Cuál es la diferencia entre usar == y ===?

Compara valor, mientras que === compara también tipo

La diferencia es que mientras que con el == antes de hacer la comparación se convierten ambos datos a un tipo común. Con === ninguno de estos valores se convierte de manera implícita antes de ser comparado.

* ¿Cuál es la diferencia entre var, let y const?

ES5 var ES6 var, let y const.

Var: Las variables declaradas con var son procesadas antes de la ejecución del código. El scope de una variable declarada con var, es su contexto de ejecución. El scope de una variable declarada fuera de la función es global.

Ej. var i = 60;

(function explainVar(){ for( var i = 0; i < 5; i++){

console.log(i) //Output 0, 1, 2, 3, 4 }

})();

console.log("Despues del loop", i) // Output 60

En resumen, la declaración con var define una variable en el ámbito local actual (lease función) y se hereda a scopes descendientes por referencia. Si la variable es declarada fuera de una función, la variable será una variable global.

let y const son dos formas de declarar variables en JavaScript introducidas en ES6 que reducen el ámbito de la variable a bloques (con var el ámbito era la función actual) y no admiten hoisting. Además, las variables declaradas con const no pueden ser reasignadas (aunque no significa que su valor sea inmutable).

CONST: Es una constante la cual NO cambiara su valor en ningún momento en el futuro.

VAR: Es una variable que SI puede cambiar su valor y su scope es local.

LET: Es una variable que también podra cambiar su valor, pero solo vivirá(Funcionara) en el bloque donde fue declarada.

* ¿Qué es implicit coercion?

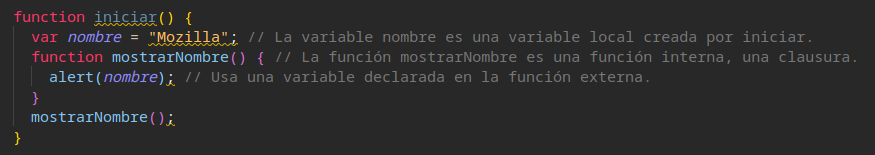
Coercion is the process of conversion of data from one type to another

* ¿En CSS qué es especificidad y cuáles son las reglas para determinarla?

Especificidad es la manera mediante la cual los navegadores deciden qué valores de una propiedad CSS son más relevantes para un elemento Selectores de tipo (p.e., h1) y pseudo-elementos (p.e., ::before). Selectores de clase (p.e., .example), selectores de atributos (p.e., [type="radio"]) y pseudo-clases (p.e., :hover). Selectores de ID (p.e., #example).

* Qué es un closure?

Una o closure es una función que guarda referencias del estado adyacente (ámbito léxico). En otras palabras, una clausura permite acceder al ámbito de una función exterior desde una función interior. En JavaScript, las clausuras se crean cada vez que una función es creada.



* ¿Qué es una pseudo-clase?

Una pseudoclase CSS es una palabra clave que se añade a los selectores y que especifica un estado especial del elemento seleccionado. Por ejemplo, :hover aplicará un estilo cuando el usuario haga hover sobre el elemento especificado por el selector.

* ¿En JS, qué es un efecto secundario o un side effect?

Los “side effects” son un término amplio, pero básicamente significa modificar cosas fuera del alcance de esa función inmediata. Algunos ejemplos de efectos secundarios …

Hacer llamadas a la API

console.log()

Math.random()

* ¿Qué es el HTML semántico y cuáles son algunos de sus beneficios?

HTML nos brinda una serie de etiquetas con mayor significado, para cada parte, sección, o elemento de nuestra página, y que, aunque en la práctica no generen un resultado distinto al de usar una etiqueta

como contenedor para todo, pueden darle mayor significado a nuestro código Código más claro y fácil de mantener Ayuda a tu sitio a ser accesible Mejora tu posicionamiento SEO

<article>

<aside>

<details>

<figcaption>

<figure>

<footer>

<header>

<main>

<mark>

<nav>

<section>

<summary>

<time>

* ¿En HTML, cuál es la diferencia entre un elemento en linea y un elemento en bloque?

Un elemento en línea ocupa sólo el espacio delimitado por las etiquetas que definen el elemento en línea. Un elemento en bloque ocupa todo el espacio de su elemento padre (contenedor)

* ¿Cómo funciona la herencia de Prototipos en JS?

<https://medium.com/@luismbcr/javascript-prototipos-y-herencia-ec4b0c695ef>

<https://developer.mozilla.org/es/docs/Web/JavaScript/Herencia_y_la_cadena_de_protipos>

Los prototipos son un mecanismo por el cual un objeto hereda propiedades y métodos de un padre, entonces en Javascript la herencia funciona por prototipos, acá no hay clases, todo se hereda por medio de prototipos.

* ¿Qué es un pseudo-elemento?

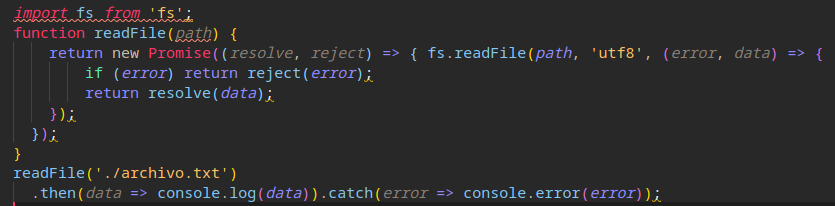
Al igual que las pseudo-clases, los pseudo-elementos se añaden a los selectores, pero en cambio, no describen un estado especial sino que, permiten añadir estilos a una parte concreta del documento. Por ejemplo, el pseudoelemento ::first-line selecciona solo la primera línea del elemento especificado por el selector.

* ¿Qué es hoisting?

una estricta definición de hoisting sugiere que las declaraciones de variables y funciones son físicamente movidas al comienzo del código

* ¿Cuál es la diferencia entre un callback y una promesa?

https://platzi.com/blog/que-es-y-como-funcionan-las-promesas-en-javascript/



Una Promise (promesa en castellano) es un objeto que representa la terminación o el fracaso de una operación asíncrona. Esencialmente, una promesa es un objeto devuelto al cuál se adjuntan funciones callback, en lugar de pasar callbacks a una función.

Un callback (llamada de vuelta) es una función que recibe como argumento otra función y la ejecuta.

* ¿Qué es un middleware?

Un middleware es un bloque de código que se ejecuta entre la petición que hace el usuario (request) hasta que la petición llega al servidor.

* ¿Qué es el session storage?

LocalStorage y sessionStorage, son propiedades que acceden al objeto Storage del navegador y tienen la función de almacenar datos de manera local,

* ¿Qué es el package.json?

De cierta forma, podemos considerar este package.json como un manifiesto de nuestro proyecto.

* ¿Qué es una función pura?

En programación, las Funciones Puras son aquellas que cumplen con dos requisitos básicos:

Dado unos parámetros de entrada de idéntico valor, la función siempre devolverá el mismo resultado.

El cómputo de la función, su lógica, no implica ningún efecto observable colateral fuera de ella.

* ¿Qué es el virtual DOM?

DOM (cuyas siglas son “Document Object Model") El Virtual DOM es una representación en memoria del DOM real que actúa de intermediario entre el estado de la aplicación y el DOM de la interfaz gráfica que está viendo el usuario.

* ¿Qué es un API?

Una API (siglas de ‘Application Programming Interface’) es un conjunto de reglas (código) y especificaciones que las aplicaciones pueden seguir para comunicarse entre ellas: sirviendo de interfaz entre programas diferentes de la misma manera en que la interfaz de usuario facilita la interacción humano-software.

* ¿Qué es prop drilling y cómo se puede evitar?
* ¿Qué es inmutabilidad y por qué es importante en React?

En primer lugar, lo inmutable es lo opuesto a mutable, y mutable significa cambiante, modificable … capaz de ser molestado.

Entonces, algo que es **in**mutable, entonces, es algo que no se puede cambiar.

Una princesa besa a una rana con la esperanza de que se convierta en un apuesto príncipe. El concepto de inmutabilidad establece que una rana siempre será una rana.

Array.sort() muta

<https://cacrmsoftware.com/por-que-es-tan-importante-la-inmutabilidad-en-javascript/>

* ¿Cuál es el ciclo de vida de un componente?

Una serie de funciones que se ejecutan en distintos momentos de la vida del componente y nos permiten realizar distintas acciones en estos momentos.

El ciclo de vida se puede dividir en 3 fases, el montado, actualización y desmontado del componente. Estas fases a su vez se dividen en varios métodos que puede tener el componente.

componentWillMount()

render()

componentDidMount()

componentWillReceiveProps(nextProps)

shouldComponentUpdate(nextProps, nextState)

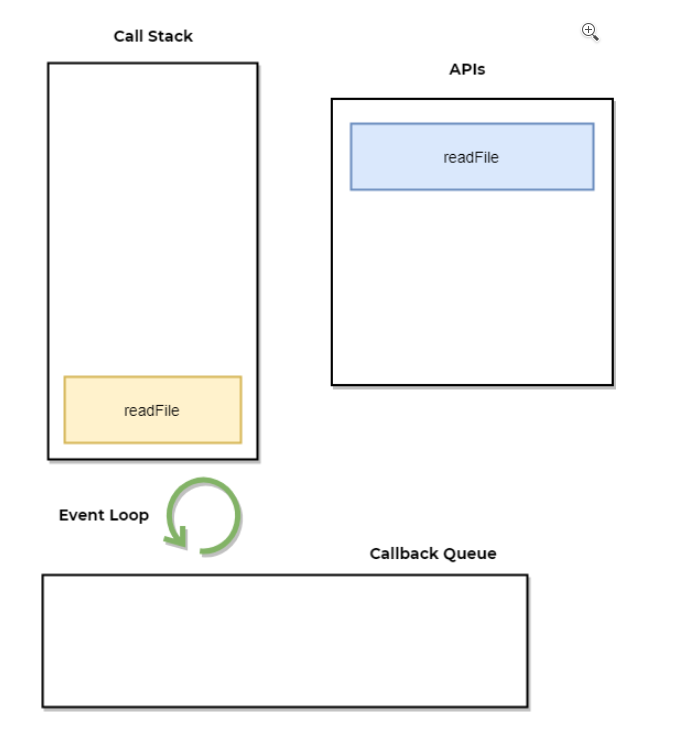
render()

componentDidUpdate(prevProps, prevState)

componentWillUnmount()

* Que es el Event Loop?

Cuando se ejecuta el código y en que orden



<https://medium.com/@_ferh97/nodejs-y-el-event-loop-21b33fea6b03>

* ApiRest JSON

<https://www.idento.es/blog/desarrollo-web/que-es-una-api-rest/>

* Diferencia npm npx

NPM es un gestor de paquetes de Node.js cuyo objetivo es gestionar los paquetes y las dependencias. Significa que podemos determinar en un fichero los paquetes que necesitamos (package.json) y luego instalar las dependencias que hay en él (npm install).

NPX es una herramienta para ejecutar paquetes de Node y viene incluido en la instalación de NPM desde la versión 5.2. NPX funciona de la siguiente manera: Comprueba si el paquete a ejecutar está instalado Si no está instalado lo instala Si está instalado lo ejecuta Con el parámetro --no-install podemos indicarle a NPX que si el paquete no existe no lo instale.

* Diferencias entre ES5 y ES6.

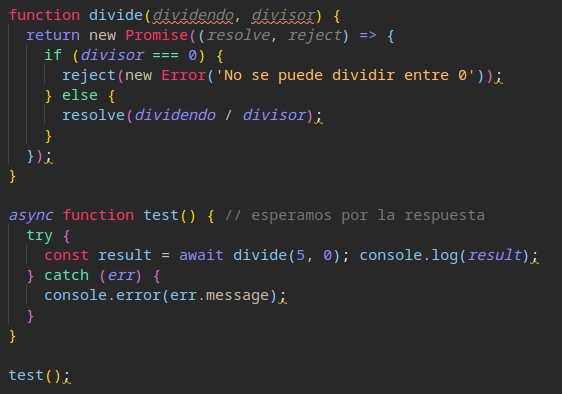
Arrow function

Asignación “clave:valor” en un objeto En los objetos, no siempre es necesario poner la clave:valor.

Valores por defecto en las funciones

Importación de JS require ES5 import ES6

<https://medium.com/@jagogutierrez/ecmascript-es6-diferencias-notables-al-es5-83d3e33ae201>

* Promises
* Diferencias function y arrow function <https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Funciones/Arrow_functions> Una de las razones por las que se introdujeron las funciones flecha fue para eliminar complejidades del ámbito (this) y hacer que la ejecución de funciones sea mucho más intuitiva.
* El modo estricto de JavaScript: qué es y para qué sirve

Una forma de trabajar con JavaScript bastante común en código profesional es utilizar siempre por defecto el Modo Estricto de este lenguaje. Se trata de una variante del lenguaje que es menos permisiva con ciertos tipos de comportamientos en el código y que hace que éste se comporte de un modo más estricto, como su propio nombre indica.