**## Respuestas al Test de JavaScript**

**Recuerda que estas respuestas (o al menos la mayoría) NO SON ABSOLUTAS. Es completamente válido (en la mayoría de los casos) si tienes otras respuestas. Recuerda que podemos discutirlo en la sección de comentarios del curso. :D**

**## Variables y operaciones**

**### 1️⃣ Responde las siguientes preguntas en la sección de comentarios:**

**- ¿Qué es una variable y para qué sirve?**

Cajitas (espacios en memoria) donde podemos guardar información (dependiendo de los tipos y estructuras de datos que soporte nuestro lenguaje).

**- ¿Cuál es la diferencia entre declarar e inicializar una variable?**

Declarar es cuando le decimos a JavaScript que vamos a crear una variable con el nombre tal. Mientras que inicializar (o reinicializar) es asignarle un valor a esa variable. Recordemos que si hacemos una variable con “const” no podremos cambiar su memoria.

**- ¿Cuál es la diferencia entre sumar números y concatenar strings?**

**- ¿Cuál operador me permite sumar o concatenar?**

EL operador que nos permite sumar o concatenar es + Este operador nos permite sumar números cuando lo usamos con números. Pero cuando los strings, lo que hace es unir (concatenar, así se dice) ambos strings.

**### 2️⃣ Determina el nombre y tipo de dato para almacenar en variables la siguiente información:**

**-** Nombre: string

- Apellido: string

- Nombre de usuario en Platzi: string ejemplo = (@fulanito)

- Edad: number

- Correo electrónico: string ejemplo = (lala@gmail.com)

- Mayor de edad: boolean

- Dinero ahorrado: number

- Deudas: number

**### 3️⃣ Traduce a código JavaScript las variables del ejemplo anterior y deja tu código en los comentarios.**

**```**

let nombre = 'Sergio David';

let apellido = 'Hidalgo Granados';

let username = 'shidalgra';

let edad = 37;

let mail = 'shidalgra@gmail.com';

let esMayorDeEdad = true;

let dineroAhorrado = 5;

let deudas = 100;

**```**

**### 4️⃣ Calcula e imprime las siguientes variables a partir de las variables del ejemplo anterior:**

**- Nombre completo (nombre y apellido)**

**- Dinero real (dinero ahorrado menos deudas)**

**```**

let nombreCompleto = nombre + ' ' + apellido;

let dineroReal = dineroAhorrado - deudas;

**```**

**## Funciones**

**### 1️⃣ Responde las siguientes preguntas en la sección de comentarios:**

**- ¿Qué es una función?**

Las funciones nos permiten encapsular (guardar) bloques de código para reutilizarlos y ejecutarlos en el futuro.

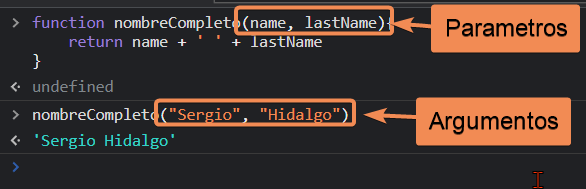
**- ¿Cuándo me sirve usar una función en mi código?**

* Nos sirve cuando tenemos código repetido, cuando tenemos bloques de código que se repiten una y otra vez y lo que cambia son cosas mínimas.
* Nos sirve cuando tenemos variables o bloques de código muy parecidos (con cambios que podrían ser parámetros y argumentos) que podemos encapsular para reutilizar más de una vez en el futuro.
* También nos sirve para ordenar y mejorar la legibilidad de nuestro código.

**- ¿Cuál es la diferencia entre parámetros y argumentos de una función?**

Las funciones reciben parámetros cuando las creamos. Y les enviamos argumentos cuando las ejecutamos.

**Ejemplo:**



**### 2️⃣ Convierte el siguiente código en una función, pero cambiando cuando sea necesario las variables constantes por parámetros y argumentos en una función:**

**Ejemplo código en consola de un navegador**

***```***

*const name = "Sergio David";*

*const lastname = "Hidalgo Granados";*

*const completeName = name +’ ‘+ lastname;*

*const nickname = "shidalgra";*

*console.log ("Mi nombre es " + completeName + ", pero prefiero que me digas " + nickname + ".");*

***```***

*const name = "Sergio";*

*const lastName = "Hidalgo";*

*const completeName = name + lastName;*

*const nickname = "shidalgo";*

*console.log ("Mi nombre es " + completeName + ", pero prefiero que me digas " + nickname + ".");*

***```***

*function nombreCompleto(name, lastName){*

*return name + ' ' + lastName*

*}*

***```***

*function saludo(name, lastname) {*

*const completeName = nombreCompleto(name, lastname);*

*const nickname = "shidalgo";*

*console.log("Mi nombre es "+ completeName + ", pero prefiero que me digas " + nickname + ".");*

*}*

***```***

*function saludo(name, lastname, username) {*

*const completeName = nombreCompleto(name, lastname);*

*console.log("Mi nombre es "+ completeName + ", pero prefiero que me digas " + username + ".");*

*}*

***```***

*saludo('Juanita', 'Cascante', 'JuanitaCas')*

**## Condicionales**

**### 1️⃣ Responde las siguientes preguntas en la sección de comentarios:**

**- ¿Qué es un condicional?**

Son la forma en que ejecutamos un bloque de código u otro dependiendo de alguna condición o validación.

**- ¿Qué tipos de condicionales existen en JavaScript y cuáles son sus diferencias?**

If(else y else if), switch, Operador ternario

El condicional if (con else y else if) nos permite hacer validaciones completamente distintas (si así lo queremos) en cada validación o condición. En cambio, en el switch todos los cases se comparan con la misma variable o condición que definimos en el switch

**- ¿Puedo combinar funciones y condicionales?**

Si. Las funciones pueden encapsular cualquier bloque de código, incluyendo condicionales.

**### 2️⃣ Replica el comportamiento del siguiente código que usa la sentencia switch utilizando if, else y else if:**

**```**

*const tipoDeSuscripcion = "Basic";*

*switch (tipoDeSuscripcion) {*

*case "Free":*

*console.log("Solo puedes tomar los cursos gratis");*

*break;*

*case "Basic":*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un mes");*

*break;*

*case "Expert":*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un año");*

*break;*

*case "ExpertPlus":*

*console.log("Tú y alguien más pueden tomar TODOS los cursos de Platzi durante un año");*

*break;*

*}*

**```**

*const tipoDeSuscripcion = "Expert";*

*if(tipoDeSuscripcion == "Free"){*

*console.log("Solo puedes tomar los cursos gratis");*

*// return; //este return solo sirve si estamos en funciones y es como el break de un switch*

*}else if(tipoDeSuscripcion == "Basic"){*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un mes");*

*}else if(tipoDeSuscripcion == "Expert"){*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un año");*

*}else if(tipoDeSuscripcion == "ExpertPlus"){*

*console.log("Tú y alguien más pueden tomar TODOS los cursos de Platzi durante un año");*

*}*

**Operador Ternario funciona así**

Si es verdadero

nombreVariable **=** Condición **?** acción1 **:** accion2;

Si es falso

**### 3️⃣ Replica el comportamiento de tu condicional anterior con if, else y else if, pero ahora solo con if (sin else ni else if).**

**Ejercicio # 2️⃣ pasado utilizando solo “if”**

*function conseguirTipoSuscripcion(suscripcion){*

*if(suscripcion == "Free"){*

*console.log("Solo puedes tomar los cursos gratis");*

*return; //este return solo sirve si estamos en funciones y es como el break de un switch*

*}*

*if(suscripcion == "Basic"){*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un mes");*

*return;*

*}*

*if(suscripcion == "Expert"){*

*console.log("Puedes tomar casi todos los cursos de Platzi durante un año");*

*return;*

*}*

*if(suscripcion == "ExpertPlus"){*

*console.log("Tú y alguien más pueden tomar TODOS los cursos de Platzi durante un año");*

*return;*

*}*

*console.warn('Este tipo de suscripcion no existe');*

*}*

**> 💡 Bonus: si ya eres una experta o experto en el lenguaje, te desafío a comentar cómo replicar este comportamiento con arrays o objetos y un solo condicional. 😏**

*const tiposDeSuscripciones = {*

*free: 'solo puedes tomar los cursos gratis',*

*basic: 'Puedes tomar casi todos los cursos platzi durante un mes',*

*expert: 'Puedes tomar casi todos los cursos platzi durante un año',*

*expertduo: 'Tú y alguien mas pueden tomar Todos los cursos platzi durante un año' ,*

*};*

*//--*

*function conseguirTipoDeSuscripcion(suscripcion){*

*if(tiposDeSuscripciones[suscripcion]){*

*console.log(tiposDeSuscripciones[suscripcion]);*

*return;*

*}*

*console.warn('Esa suscripción no existe');*

*}*

*conseguirTipoDeSuscripcion('free')*

**## Ciclos**

**### 1️⃣ Responde las siguientes preguntas en la sección de comentarios:**

**- ¿Qué es un ciclo?**

La forma de ejecutar un bloque de código hasta que se cumpla cierta condición. O que la condición sea verdadera

**- ¿Qué tipos de ciclos existen en JavaScript?**

While, do while, for

**- ¿Qué es un ciclo infinito y por qué es un problema?**

Es cuando la validación de nuestros condicionales nunca se cumple y termina toteando o colgando (dañando) la aplicación(e.j. cuando el navegador ya no puede más de tanta ejecución de ese bloque de código).

**- ¿Puedo mezclar ciclos y condicionales?**

Sí, aunque los ciclos son una especie de condicionales, nada nos impide agregar mas condicionales dentro de l ciclo.

**### 2️⃣ Replica el comportamiento de los siguientes ciclos for utilizando ciclos while:**

**```**

for (let i = 0; i < 5; i++) { //si es menor que

console.log("El valor de i es: " + i);

}

//Aquí el while

*while (i < 5) {*

*console.log("El valor de i es: "+i);*

*i++;*

*}*

---------------------------------------------------------------------------------------------------------

for (let i = 10; i >= 2; i--) {

console.log("El valor de i es: " + i); //si es mayor o igual que

}

//Aquí el while

*while(i >= 2){*

*console.log("El valor es: "+i);*

*i--*

*}*

**```**

**### 3️⃣ Escribe un código en JavaScript que le pregunte a los usuarios cuánto es `2 + 2`. Si responden bien, mostramos un mensaje de felicitaciones, pero si responden mal, volvemos a empezar.**

**> 💡 Pista: puedes usar la función prompt de JavaScript.**

*Let respuesta;*

*while (respuesta != '4'){*

*let pregunta = prompt('¿Cuanto es 2 + 2?')*

*respuesta = pregunta;*

*}*

**## Listas**

**### 1️⃣ Responde las siguientes preguntas en la sección de comentarios:**

**- ¿Qué es un array?**

Es una lista de elementos

```

cosnt array = [1, ‘jaja’, true, false, {nombre: ‘fulanito’, edad: 3 } ]; //puede ser cualquier tipo de variable

```

**- ¿Qué es un objeto?**

Es una lista de elementos, PERO cada elemento tiene un nombre clave

```

*const nombreDeObjeto = {*

*nombre: "Fulanito",*

*edad: 3,*

*email: "fulanito@fulanos.com",*

*comidasFavoritas: ["Pollo frito", "vegetales"],*

*};*

```

**- ¿Cuándo es mejor usar objetos o arrays?**

Arrays cuando lo que haremos en un elemento es lo mismo que en todos los demás (la regla se puede incumplir). Mientras que un objeto cuando los nombres de cada elemento son importantes para nuestro algoritmo.

**- ¿Puedo mezclar arrays con objetos o incluso objetos con arrays?**

Sí, los arrays pueden guardar objetos. Y los objetos pueden guardar arrays entre sus propiedades

**### 2️⃣ Crea una función que pueda recibir cualquier array como parámetro e imprima su primer elemento.**

**```**

*function imprimirPrimerElementoArray(arr){*

*console.log(arr[0])*

*}*

**```**

**### 3️⃣ Crea una función que pueda recibir cualquier array como parámetro e imprima todos sus elementos uno por uno (no se vale imprimir el array completo).**

*function imprimirElementoPorElemento(arr){*

*for(let i = 0; i < arr.length; i++){ //tenemos que tener cuidado con el “< que” o “> que”*

*console.log(arr[i]);*

*}*

*}*

**```**

**### 4️⃣ Crea una función que pueda recibir cualquier objeto como parámetro e imprima todos sus elementos uno por uno (no se vale imprimir el objeto completo).**

**```**

*const nombreDeObjeto = {*

*nombre: "Fulanito",*

*edad: 3,*

*email: "fulanito@fulanos.com",*

*comidasFavoritas: ["Pollo frito", "vegetales"],*

*};*

*function imprimirElementoPorElementoObjeto(obj){*

*const arr = Object.values(obj);*

*for(let i = 0; i < arr.length; i++){*

*console.log(arr[i])*

*}*

*}*

*imprimirElementoPorElementoObjeto(nombreDeObjeto)*

**```**

**Icono

Descripción generada automáticamente**

**Hay varias formas de ejecutar JS,**

*// h1 { color: red }        //en css*

*//.parrafito { ... }*

*//#pid { ... }*

*document. getElementById(‘xxx’)*

*document. getElementsByClassName(‘xxx’)*

*const h1 = document.querySelector('h1');*

*const p = document.querySelector('p');*

*const parrafito = document.querySelector('.parrafito');*

*const pid = document.getElementById('#pid');*

*const input = document.querySelector('input');*

*console.log(input.value);*

*console.log({*

*h1,*

*p,*

*parrafito,*

*pid,*

*input,*

*});*

**

**HTML**

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta http-equiv="X-UA-Compatible" content="IE=edge">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Manipulación del DOM básica - Curso Práctico de JavaScript</title>*

*</head>*

*<body>*

*<h1 class="Verde">Manipulación del DOM básica</h1>*

*<h1 pantalla="PlatziShidalgo">Manipulación del DOM básica</h1>*

*<p>Esto es un párrafo</p>*

*<p class="parrafito">Esto es un párrafo con clase</p>*

*<p id="pid">Esto es un párrafo con ID</p>*

*<input placeholder="Escribe algo aquí">*

*<script src="./script.js"></script>*

*</body>*

*</html>*

**JS**

*const h1 = document.querySelector('h1');*

*const p = document.querySelector('p');*

*const parrafito = document.querySelector('.parrafito');*

*const pid = document.getElementById('pid');*

*const input = document.querySelector('input');*

*console.log(input.value);*

*console.log({*

*h1,*

*p,*

*parrafito,*

*pid,*

*input,*

*});*

*//Manipulacion del DOM desde JavaScript*

*h1.innerHTML = 'Patito <br> Feo';   //aca podriamos sobreescribir el h1 que tenemos en html*

*h1.innerText = 'Patito <br> Feo';   //aca podriamos sobreescribir el h1 que tenemos en html*

*// console.log(h1.getAttribute('class'));*

*// h1.setAttribute('class', 'rojo') //Aca le podemos cambiar un atributo a la clase EJemplo si es Verde cambiara a rojo*

*h1.classList.add('rojo');           // para agregar un atributo a una clase*

*h1.classList.remove('Verde');       // para remover un atributo a una clase*

*// h1.classList.toggle('Verde');    //     para funciones*

*// h1.classList.contains('Verde');*

*input.value = "456"; //para cambiar atributos desde cero*

*//------------------------------------------------------------------------------------*

*const img = document.createElement('img');// para crear un atributo desde cero*

*img.setAttribute('src', 'https://blob.sololearn.com/avatars/71480c9e-c973-4be7-bd98-d317d18b65f6.jpg')*

*console.log(img);*

*//pid.appendChild(img);   //apend y appendChild son lo mismo... agregaran a lo que este HTML*

*pid.innerHTML = "";*

*pid.append(img);*

*//------------------------------------------------------------------------------------*

**

*Hacer que las personas que visiten la Web, vean que la página esta viva*

**HTML**

*<!-- Ir al archivo indexPrimero darle click derecho y a "copy path" pegar la URL en el navegador -->*

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta http-equiv="X-UA-Compatible" content="IE=edge">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Manipulación del DOM básica - Curso Práctico de JavaScript</title>*

*</head>*

*<body>*

*<h1>Manipulación del DOM básica</h1>*

*<input id="calculo1" placeholder="Escribe algo aquí">*

*<input id="calculo2" placeholder="Escribe algo aquí"> <!-- acá se le puede póner en vez de onclick sería el onchange="" -->*

*<button id="btnCalcular" onclick="btnOnclick()">Calcular</button> <!-- acá utilizamos el onclick -->*

*<p id="result"></p>*

*<script src="./script.js"></script>*

*</body>*

*</html>*

**JS**

*const h1 = document.querySelector('h1');*

*const input1 = document.querySelector('#calculo1');*

*const input2 = document.querySelector('#calculo2');*

*const btn = document.querySelector('#btnCalcular');*

*const pResult = document.querySelector('#result');*

*function btnOnclick(){*

*const sumaInputs = input1.value + ' ' + input2.value;*

*pResult.innerText = "Resultado: " + sumaInputs;*

*}*

*Logotipo

Descripción generada automáticamente*

***HTML***

*<!-- Ir al archivo indexPrimero darle click derecho y a "copy path" pegar la URL en el navegador -->*

*<!DOCTYPE html>*

*<html lang="en">*

*<head>*

*<meta charset="UTF-8">*

*<meta http-equiv="X-UA-Compatible" content="IE=edge">*

*<meta name="viewport" content="width=device-width, initial-scale=1.0">*

*<title>Manipulación del DOM básica - Curso Práctico de JavaScript</title>*

*</head>*

*<body>*

*<h1>Manipulación del DOM básica</h1>*

*<form id="form">*

*<input id="calculo1" placeholder="Escribe algo aquí">*

*<input id="calculo2" placeholder="Escribe algo aquí">*

*<button type="button" id="btnCalcular">Calcular</button>   <!-- acá quitamos el onclick -->*

*<!-- Un boton el ultimo de un formulario lo interpretan como tipo submit por eso con un form se recarga la pagina*

*y si tuvieramos algo en los inputs nos enviaria por la nomenclatura llamada query parameters-->*

*<!-- entonces dentro de un form no nos serviria el JavaScript addEventListener -->*

*<p id="result"></p>*

*</form>*

*<script src="./script.js"></script>*

*</body>*

*</html>*

***JS***

*//Ir al archivo indexPrimero darle click derecho y a "copy path" pegar la URL en el navegador*

*const h1 = document.querySelector('h1');*

*const form = document.querySelector('#form');*

*const input1 = document.querySelector('#calculo1');*

*const input2 = document.querySelector('#calculo2');*

*const btn = document.querySelector('#btnCalcular');*

*const pResult = document.querySelector('#result');*

*//Para hacer el evento desde acá y quitar el onclick de html entonces:*

*//cual selector html estamos hablando.addEventListener( 'evento' , funcion a ejecutar );*

*// btn.addEventListener('click', btnOnclick );*

*// function btnOnclick(){*

*//     const sumaInputs = input1.value + ' ' + input2.value;*

*//     pResult.innerText = "Resultado: " + sumaInputs;*

*// }*

*// si tenemos este btn dentro de un form entonces no podemos utilizar este addEventListener*

*// para poder utilizarlo tenermos que hacer lo const form*

*// //--------------------------------------------------------------------------------------------------------*

*// // si ponemos todo dentro de un form*

*// // al poner el form entonces esta linea 12 cambia a:*

*// form.addEventListener('submit', sumarInputValues);*

*// //y la linea 14 tambien cambia*

*// function sumarInputValues(event){*

*//     console.log({event});*

*//     event.preventDefault();*

*//     const sumaInputs = input1.value + ' ' + input2.value;*

*//     pResult.innerText = "Resultado: " + sumaInputs;*

*// }*

*//--------------------------------------------------------------------------------------------------------*

*// Otra forma de resolver lo de un form*

*// al poner el form entonces esta linea 12 cambia a:*

*btn.addEventListener('click', sumarInputValues);*

*//y la linea 14 tambien cambia*

*function sumarInputValues(event){*

*console.log({event});   // no hay que hacer un console.log*

*// event.preventDefault(); no hay que hacer un preventDefault*

*// tenemos que irnos al html y decirle al boton que es de type="button"*

*const sumaInputs = input1.value + ' ' + input2.value;*

*pResult.innerText = "Resultado: " + sumaInputs;*

*}*

**

*Git bash se utiliza desde la terminal se escoge desde el signo de mas donde dice PowerShell escogemos la carpeta donde vamos a trabajar y damos el comando*

*“code ./ -r” sin comillas*

***Instalar GIT***

***GitHub para Windows***

[*http://windows.github.com*](http://windows.github.com)

***GitHub para mac***

[*http://mac.github.com*](http://mac.github.com)

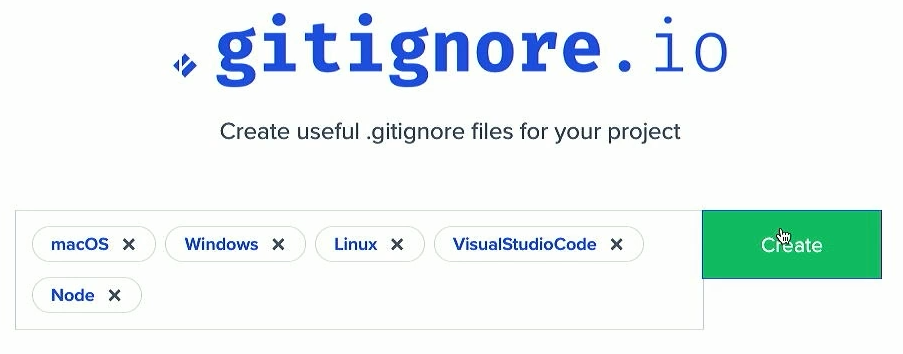
*Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.*

***Git para toda plataforma***

[*http://git-scm.com*](http://git-scm.com)

***Pagina para hacer .gitignore***

[*http://gitignore.io*](http://gitignore.io)

**

***Configurar herramientas***

*Configura la información del usuario para todos los repositorios locales*

***$ git config --global user.name “[name]”*** *Establece el nombre que desea que esté anexado a sus transacciones de commit*

***$ git config --global user.email “[email address]”*** *Establece el e-mail que desea que esté anexado a sus transacciones de commit*

***$ git config --global color.ui auto*** *Habilita el uso útil de color del producto de la línea de comando*

***Crear Repositorios***

*Inicia un nuevo repositorio u obtiene uno de una URL existente*

***$ git init [Project-name]*** *Crea un nuevo repositorio local con el nombre especificado*

***$ git clone [url]*** *Descarga un proyecto y toda su historia de versión*

***Efectuar cambios***

*Revisa las ediciones y elabora una transacción de commit*

***$ git status*** *Enumera todos los archivos nuevos o modificados que se deben confirmar*

***$ git diff*** *Muestra las diferencias de archivos que no se han enviado aún al área de espera*

***$ git add [file****] Toma una instantánea del archivo para preparar la versión*

***$ git add -A*** *Agrega todos los archivos en el status para preparar la versión*

***$ git diff –staged*** *Muestra las diferencias del archivo entre el área de espera y la última versión del archivo*

***$ git reset [file****] Mueve el archivo del área de espera, pero preserva el su contenido*

***$ git commit -m “[mensaje descripción]”*** *Registra las instantáneas del archivo permanente en el historial de versión*

***$ git push*** *Envía al repositorio los archivos agregados con git add o git add -A (hay que estar seguros)*

***Cambios grupales***

*Nombra una serie de commits y combina esfuerzos ya culminados*

***$ git branch*** *Enumera todas las ramas en el repositorio actual*

***$ git branch [branch-name****] Crea una nueva rama*

***$ git checkout [Branch-name]*** *Cambia a la rama especificada y actualiza el directorio activo*

***$ git merge [branch]*** *Combina el historial de la rama especificada con la rama actual*

***$ git Branch -d [Branch-name]*** *Borra la rama especificada*

***Nombres del archivo de refactorización***

*Reubica y retira los archivos con versión*

***$ git rm [file]*** *Borra el archivo del directorio activo y pone en el área de espera el archivo borrado*

***$ git rm --cached [file]*** *Retira el archivo del control de versiones, pero preserva el archivo a nivel local*

***$ git mv [file-original] [file-renamed]*** *Cambia el nombre del archivo y lo prepara para el commit*

***Suprimir tracking (Quita archivos cuando se le hace el git add)***

*Excluye los archivos temporales y las rutas*

***\*.log*** *Un archivo de texto llamado* **.gitignore***suprime la creación accidental de versiones de archivos y*

***Build/*** *rutas que concuerdan con los patrones especificados.*

***Temp-\****

***$ git ls-files --other --ignored --exclude-standard*** *Enumera todos los archivos ignorados en este proyecto*

***Guardar fragmentos***

*Almacena y restaura cambios incompletos*

***$ git stash*** *Almacena temporalmente todos los archivos tracked modificados*

***$ git stash pop*** *Restaura los archivos guardados más recientemente*

***$ git stash list*** *Enumera todos los sets de cambios en guardado rápido*

***$ git stash drop*** *Elimina el set de cambios en guardado rápido más reciente*

***Repasar historial***

***$ git log*** *Enumera el historial de la versión para la rama actual*

***$ git log --follow [file]*** *Enumera el historial de versión para el archivo, incluidos los cambios de nombre*

***$ git diff [first-branch]…[second-branch]*** *Muestra las diferencias de contenido entre dos ramas*

***$ git show [commit]*** *Produce metadatos y cambios de contenido del commit especificado*

***Rehacer Commits***

*Borra errores y elabora historial de reemplazo*

***$ git reset [commit]*** *Deshace todos los commits después de* ***[commit]****, preservado los cambios localmente*

***$ git reset --hard [commit]*** *Desecha todo el historial y regresa el commit especificado*

***Sincronizar cambios***

*Registrar un marcador de repositorio e intercambiar historial de versión*

***$ git fetch [bookmark]*** *Descarga todo el historial del marcador del repositorio*

***$ git merge [bookmark]/[branch]*** *Combina la rama del marcador con la rama local actual*

***$ git push [alias] [branch]*** *Carga todos los commits de la rama local a GitHub*

***$ git pull*** *Descarga el historial del marcador e incorpora cambios*