


Empezamos a las
19:00hs





Variables

Guardemos nuestros datos



TEMAS

- Repaso
- Variables
- Scope
- Primitivos y no primitivos
- Tipos de datos
- Hoisting

Repaso de la clase anterior

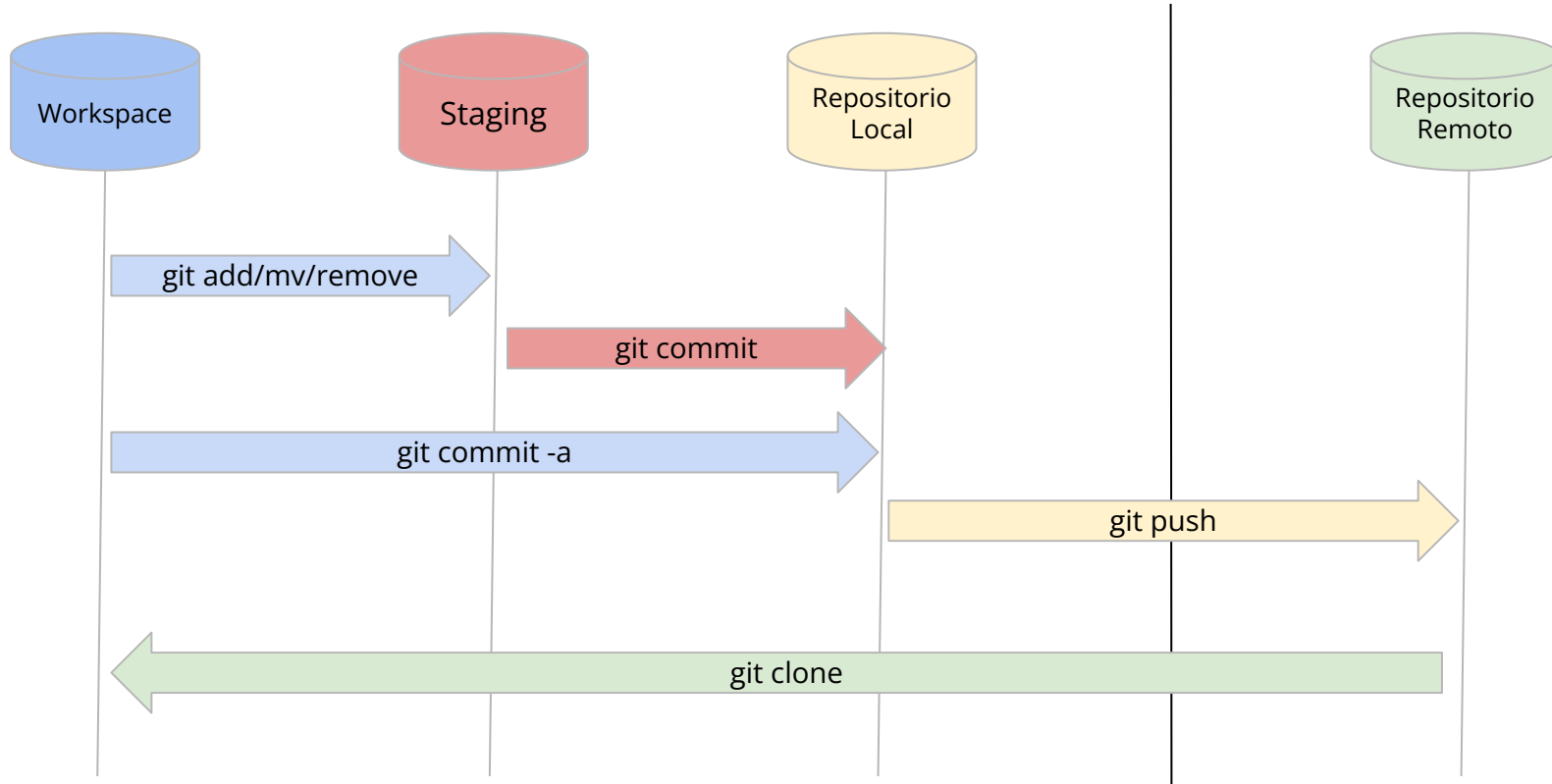
- Flujo de GIT
- Clone
- Commit
- Push
- Pull
- Merge
- Keys

GIT

Git es un sistema de control de versiones distribuido, diseñado para rastrear y gestionar cambios en archivos y proyectos de desarrollo de software. A diferencia de los sistemas de control de versiones centralizados, Git almacena la historia completa del proyecto en cada repositorio local, lo que permite un acceso completo y un seguimiento de cambios incluso en ausencia de una conexión de red.



Estadios



Comandos

clone -> clona un repositorio remoto a tu workspace

add / mv / rm -> aplica los cambios sobre staging

commit -> confirma los cambios en el repositorio local

pull -> descarga los cambios del repositorio remoto al local

push -> sube los cambios del repositorio local al remoto

merge -> fusiona 2 ramas

branch -> muestra el branch actual

checkout [rama] -> cambia de rama

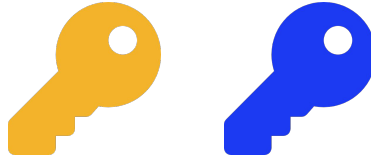
checkout -b [rama] -> crea una nueva rama

status -> muestra los cambios pendientes de subir a tu repositorio local



ssh key

Llaves diferentes



Cliente



llave privada



Internet

Servidor



llave pública



Variable

En programación, las variables son como cajas de almacenamiento donde podemos guardar diferentes cosas. Cada caja tiene un nombre y puede contener un valor específico, como números o palabras. Las variables nos permiten acceder y manipular esos valores a lo largo de nuestro programa.

Almacenamiento Volátil en la Memoria RAM

El almacenamiento en la memoria RAM es volátil. Esto significa que, al igual que cuando apagas una computadora, cuando finaliza un programa, toda la información almacenada en las variables desaparece, es como vaciar todas las cajas y perdiéramos su contenido.

Declaración de variables

En JavaScript, podemos declarar variables utilizando diferentes palabras clave: `var`, `let` y `const`. Cada una tiene características distintas que debemos tener en cuenta al programar.

JS index.js U ●

monedas > JS index.js > ...

```
1  var edad = 25;  
2  let nombre = "Juan";  
3  const PI = 3.1416;  
4
```

Primitivos

Los tipos de datos primitivos se llaman así debido a que son los bloques de construcción básicos y fundamentales en un lenguaje de programación. Son los tipos de datos más simples que no pueden descomponerse en partes más pequeñas

- `number`: representa números, tanto enteros como decimales.
- `string`: representa cadenas de texto.
- `boolean`: representa valores booleanos `true` (verdadero) o `false` (falso).
- `null`: representa la ausencia de un valor.
- `undefined`: representa el valor indefinido.
- `Symbol`: representa un valor único e inmutable que se puede utilizar como identificador de propiedad en objetos.

No Primitivos

Los tipos de datos no primitivos, que son estructuras de datos más complejas **que pueden contener múltiples valores y propiedades**, los tipos de datos primitivos representan valores individuales y no tienen métodos o funcionalidades adicionales asociadas.

- object: representa una estructura de datos compleja compuesta por pares clave-valor.
- array: representa una lista de elementos ordenados.
- function: representa un bloque de código reutilizable que realiza una tarea específica.
- date: representa una fecha y hora específicas.
- RegExp: representa una expresión regular, que se utiliza para buscar patrones en cadenas de texto.

Nombre de variables

Si



- camelCase
- comenzar con \$, _ o letra

No



- underscore
- palabras reservadas
var, let, const,
function, while, etc
- comenzar con
número

Scope

En JavaScript, el scope (alcance) se refiere a la visibilidad y accesibilidad de las variables, funciones y objetos en una parte específica del código durante la ejecución del programa. Define las reglas sobre cómo se accede y se maneja la información dentro de un bloque de código.

El scope en JavaScript se basa en la estructura de los bloques de código, como las funciones y los bloques de declaración (como los bloques if, for, while, etc.). Cada vez que se crea un nuevo bloque, se crea un nuevo scope.

Hoisting

El hoisting en JavaScript es un comportamiento que ocurre durante la fase de compilación o interpretación del código. Consiste en el movimiento de las declaraciones de variables y funciones hacia la parte superior de su scope actual, antes de que se ejecute el código.

<https://developer.mozilla.org/es/docs/Glossary/Hoisting>

“Conceptualmente, por ejemplo, una estricta definición de hoisting sugiere que las declaraciones de variables y funciones son físicamente movidas al comienzo del código, pero esto no es lo que ocurre en realidad. **Lo que sucede es que las declaraciones de variables y funciones son asignadas en memoria durante la fase de compilación**, pero quedan exactamente en dónde las has escrito en el código.”



Live code





¡Gracias!

Espero que este sea un largo
camino que transitemos juntos

