JS SEMINARIO DE LENGUAJES

EXPLICACIÓN PRÁCTICA 1

CONFIGURACIÓN BÁSICA Y PRIMEROS PASOS EN JAVASCRIPT

AGENDA

1. Introducción A Git
2. Introducción A Javascript
3. Variables En Javascript

INTRODUCCIÓN A GIT

En esta materia, vamos a utilizar Git como nuestro sistema de control de versiones para manejar el código fuente de nuestros proyectos.

¿QUÉ ES GIT?

Git es un sistema de control de versiones distribuido, diseñado para manejar todo tipo de proyectos con velocidad y eficiencia.

CONTROL DE VERSIONES

Git no solo almacena los cambios en los archivos de nuestro proyecto, sino que también permite navegar por todo el historial de cambios. Esto nos da la capacidad de comparar versiones, entender quién hizo qué y cuándo, y revertir a cualquier estado anterior del proyecto si es necesario.

IMPORTANCIA DE GIT EN DESARROLLO

Git nos permite colaborar eficientemente, mantener un historial completo de cambios, y revertir a versiones anteriores si es necesario. Será una herramienta esencial en nuestra materia.

PRÁCTICA CON GIT

A lo largo de la materia, realizaremos ejercicios prácticos para familiarizarnos con Git, desde configurar nuestro entorno hasta colaborar en proyectos grupales.

INSTALACIÓN DE GIT

- **Windows:** Instalarlo por medio de su sitio oficial.
- **GNU/Linux:** Utilizar el comando `sudo apt-get install git` en la terminal.
- **macOs:** Utilizar el comando `brew install git` en la terminal.

CLONAR EL REPOSITORIO DEL GRUPO

Para clonarse el repositorio de cada grupo de trabajo hay que hacer el siguiente comando.

```
git clone git@gitlab.catedras.linti.unlp.edu.ar:js/grupos-2024/grupo
```

A partir de ése momento, el repositorio se clonará en la carpeta que se encuentre en la terminal. Vamos a poder realizar modificaciones y subirlas al repositorio.

CONFIGURACIÓN DE GIT

Antes de comenzar a utilizar Git, es importante configurar nuestro nombre y dirección de correo electrónico. Esto nos permitirá identificar quién realizó cada cambio en el proyecto.

```
git config user.name "Tu Nombre"
git config user.email "tu@email.com"
```

VERIFICACIÓN

Para asegurarnos de que todo está correctamente instalado, podemos verificar las versiones de Git y de configuración

- `git --version`
- `git config --list`

FLUJO DE TRABAJO COMPLETO CON GIT

1. Realiza cambios en tu código o añade nuevos archivos a tu proyecto.
2. Abre la terminal y navega hasta la carpeta de tu proyecto.
3. Agrega los cambios al área de preparación.

```
git add .
```

4. Confirma los cambios en tu repositorio local.

```
git commit -m "Descripción de los cambios realizados"
```

5. Sube tus cambios al repositorio remoto.

```
git push origin main
```

OPERACIONES BÁSICAS DE GIT [1]

- **Clone:** Copiar un repositorio existente en una nueva ubicación local.
- **Add:** Agregar cambios en archivos a la área de preparación.
- **Commit:** Registrar los cambios preparados en el historial del repositorio.
- **Push:** Enviar cambios locales a un repositorio remoto.
- **Pull:** Actualizar el repositorio local con cambios desde un repositorio remoto.

OPERACIONES BÁSICAS DE GIT [2]

No es necesario memorizar estas operaciones por ahora; lo importante es comprender los conceptos subyacentes. Profundizaremos en estos comandos en la práctica.

COMANDOS AVANZADOS DE GIT

Para trabajar más efectivamente con Git, es importante conocer algunos comandos avanzados:

- **Status:** Muestra el estado actual del repositorio, indicando cambios sin agregar o sin comprometer.
- **Branch:** Permite crear, listar y eliminar ramas.
- **Merge:** Fusiona cambios de una rama a otra.
- **Rebase:** Integra cambios de una rama a otra reorganizando la historia de commits.

CONCEPTOS DE RAMAS

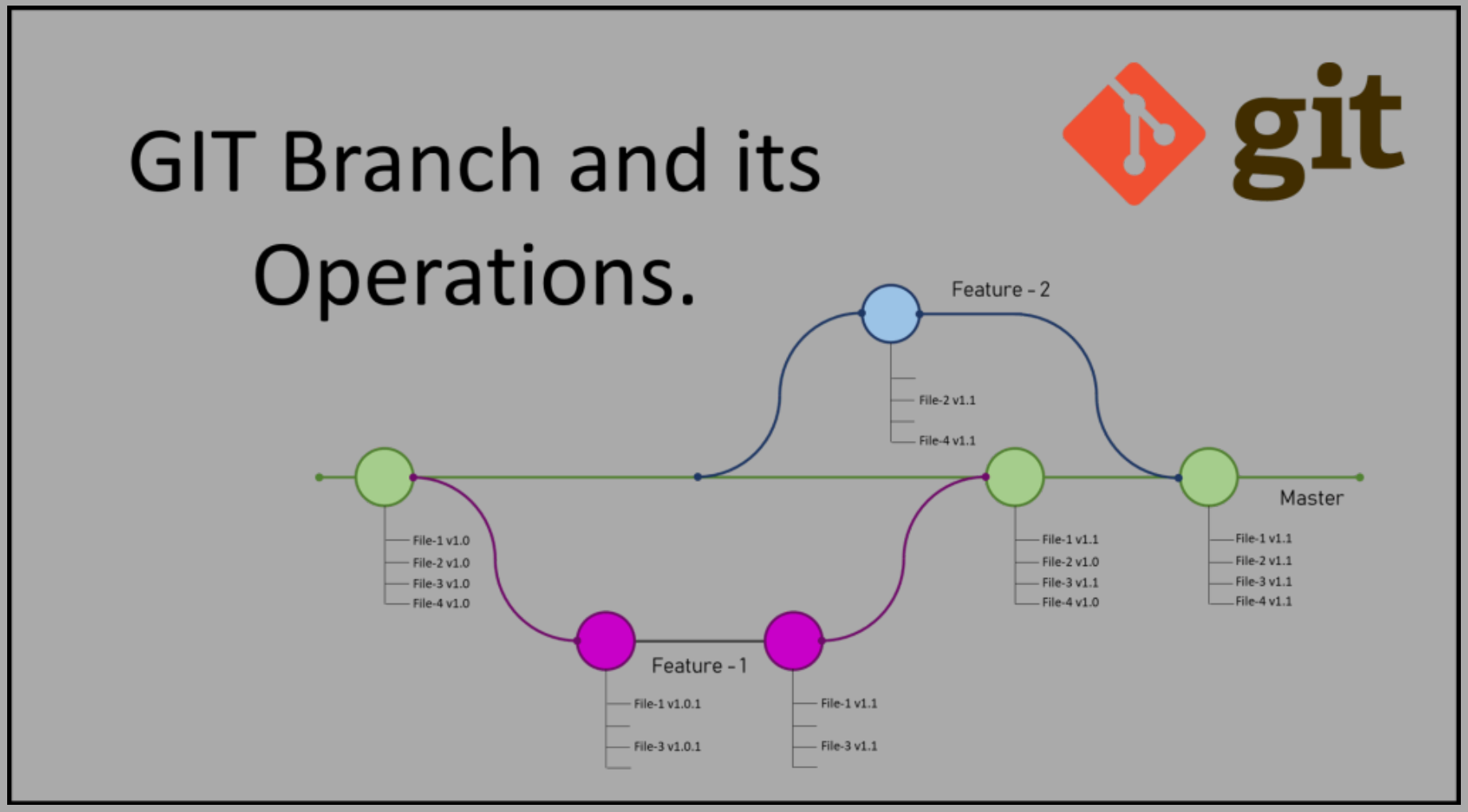
Las ramas en Git son fundamentales para manejar diferentes versiones de tu proyecto de manera simultánea.

Una rama en Git es un apuntador a uno o varios de los commits del proyecto. La rama principal suele llamarse `main`.

BENEFICIOS DE USAR RAMAS

- Desarrollo paralelo de diferentes características.
- Organización clara del trabajo.
- Pruebas y experimentación sin afectar el código principal.

FLUJO DE COMMITS Y RAMAS



INTRODUCCIÓN A JAVASCRIPT

JavaScript es un lenguaje de programación que se ejecuta (originalmente) en el navegador del cliente y permite crear páginas web interactivas.

A diferencia de HTML y CSS, que proporcionan la estructura y el estilo de una página, JavaScript proporciona funcionalidad interactiva.

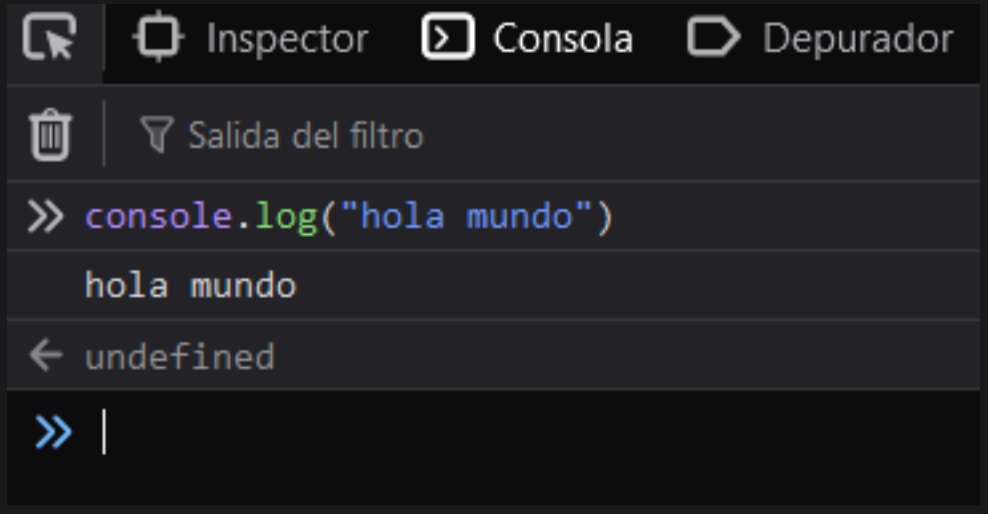
Es un lenguaje de programación interpretado, lo que significa que el código se ejecuta directamente sin necesidad de compilación previa.

EJEMPLO DE UN SCRIPT EN JAVASCRIPT

```
// Esto es un comentario
let contador = 1
while (contador <= 5) {
  console.log(contador)
  contador++
}
```

EJECUTAR JAVASCRIPT DESDE LA TERMINAL DEL NAVEGADOR

La consola del navegador es una herramienta poderosa que te permite interactuar con la página web mediante código JavaScript.



EJECUTAR JAVASCRIPT EN UN ARCHIVO HTML [1]

JavaScript se puede incorporar en documentos HTML de dos maneras:

- Internamente, mediante el uso de la etiqueta `<script>` dentro del documento HTML
- Externamente, vinculando un archivo JavaScript externo

EJECUTAR JAVASCRIPT EN UN ARCHIVO HTML [2]

MEDIANTE EL USO DE LA ETIQUETA `<script>`

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi Primera Página con JavaScript</title>
  </head>
  <body>
    <h1>¡Hola Mundo!</h1>
    <script>
      console.log('Hola Mundo desde el script interno')
    </script>
  </body>
</html>
```

EJECUTAR JAVASCRIPT EN UN ARCHIVO HTML [3]

VINCULANDO UN ARCHIVO JAVASCRIPT EXTERNO

```
<!DOCTYPE html>
<html>
  <head>
    <title>Mi Primera Página con JavaScript</title>
  </head>
  <body>
    <h1>¡Hola Mundo!</h1>
    <script src="script.js"></script>
  </body>
</html>
```

EJEMPLOS PRÁCTICOS PARA EJECUTAR [1]

Manipulación del DOM

```
document.body.style.backgroundColor = "lightblue"
```

Este código cambia el color de fondo de la página a celeste.

Alertas en el Navegador

```
alert('¡Bienvenido a JavaScript!')
```

Muestra una ventana de alerta con un mensaje.

Variables y Operaciones Básicas

```
let numero = 10
console.log(numero * 2)
```

Imprime 20 en la consola.

EJEMPLOS PRÁCTICOS PARA EJECUTAR [2]

Interacción con el Usuario

```
let nombre = prompt("¿Cuál es tu nombre?")
console.log('Hola, ' + nombre)
```

Solicita al usuario su nombre y luego lo saluda.

VARIABLES EN JAVASCRIPT

VARIABLES EN JAVASCRIPT: LET Y CONST

JavaScript ofrece diferentes formas de declarar variables, cada una con sus propias características:

- **let**: Permite declarar variables con un alcance limitado al bloque, declaración o expresión donde se usa.
- **const**: Similar a `let`, pero para declarar variables cuyo valor no se pretende cambiar (constantes).

Utilizar `let` y `const` te ayuda a escribir código más seguro y predecible.

VARIABLES LET

```
let saludo = 'Hola, Mundo!'

function saludar() {
  let saludo = '¡Hola desde la función!'
  console.log(saludo) // '¡Hola desde la función!'
}

saludar()

if (saludo === 'Hola, Mundo!') {
  let saludo2 = '¡Hola desde el bloque!'
}
console.log(saludo) // 'Hola, Mundo!'
consola.log(saludo2) // Error: saludo2 no está definido
```

VARIABLES CONST

```
const mensaje = 'Hola, Mundo!'
mensaje = 'Cambio de valor' // TypeError: Assignment to constant

if (true) {
  const mensaje2 = 'Hola, Mundo!'
  console.log(mensaje2) // 'Hola, Mundo!'
}
console.log(mensaje2) // Error: mensaje2 no está definido
```

TIPOS DE DATOS EN JAVASCRIPT [1]

JavaScript es un lenguaje de programación de tipado dinámico, lo que significa que no necesitas declarar el tipo de variable de antemano. Los principales tipos de datos son:

TIPOS DE DATOS EN JAVASCRIPT [2]

- **Números**: Para representar tanto enteros como flotantes.
- **Cadenas de Texto (Strings)**: Para representar texto.
- **Booleanos**: `true` o `false`.
- **Objetos**: Para agrupar datos y funciones relacionadas.
- **Undefined** y **null**: Para variables sin valor o con un valor nulo.

FIN DE LA CLASE