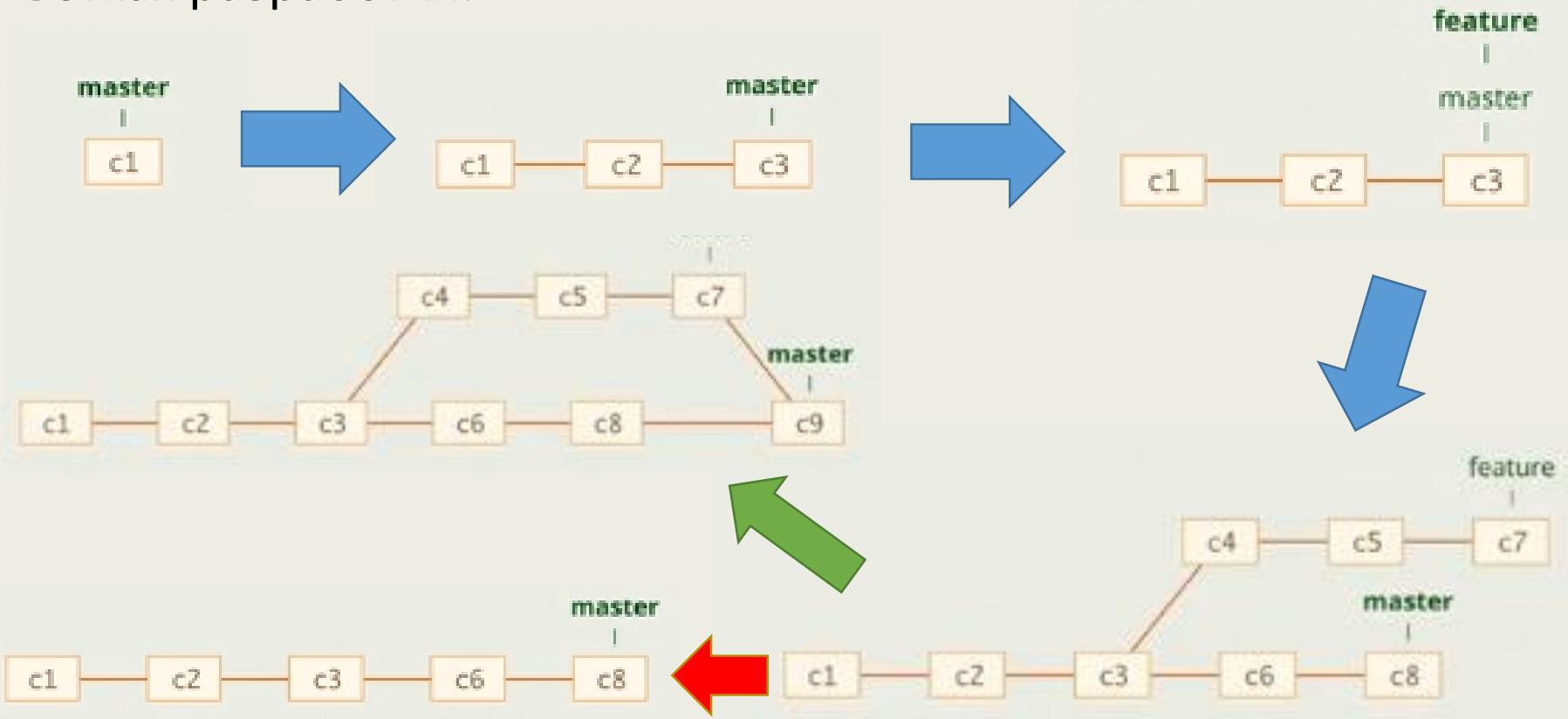
git (ветки)

ТЕМАТИЧЕСКИЕ ВЕТКИ

Ветки – это изолированной поток разработки по проекту, в который можно делать коммиты так, что их не видно в других ветках разработки.



ПРАКТИЧЕСКАЯ РАБОТА №1

1. Открываем консоль и переходим в созданную на прошлом занятии папку «cd C:\TestGit».

2. Выполняем команду чтобы посмотреть информацию о доступных ветках:

```
git branch -v
```

3. Создадим новый файл script.js в папке TestGit с репозиторием и наполним её:

```
console.log('Hello from git');
```

ПРАКТИЧЕСКАЯ РАБОТА №1

Подключим файл script.js к index.html путём добавления в заголовочный тег следующего тега:

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

4. Закоммитем эти изменения выполнив команды:

```
git add .
git commit –m "create hello"
git push -u origin master
git branch -v
```

ПРАКТИЧЕСКАЯ РАБОТА №1

5. Допустим у нас появилась новая идея и для её реализации создадим новую ветку с названием feature:

```
git branch feature
```

6. Переключаемся на созданную ветку. Выполнив команду:

```
git checkout feature
```

7. Добавим строку в конец файла script.js:

```
console.log('Feature work!!!');
```

ПРАКТИЧЕСКАЯ РАБОТА №1

8. Закоммитим изменения:

```
git add .
git commit –m "create feature"
git push -u origin feature
git branch –v
```

9. Просмотрим история ветки. Выполнив команду:

```
git log
```

10. Посмотрим содержимое коммита. Выполнив команду:

```
git show
```

ПРАКТИЧЕСКАЯ РАБОТА №1

11. Вернёмся к основной ветке master:

```
git checkout master
git log
```

12. Теперь вольём изменения из ветки feature в ветку master. Выполнив команду:

```
git merge feature
```

13. Так как ветка feature выполнила свою функцию её можно удалить командой:

```
git branch -d feature
git push origin --delete feature
```

ПРАКТИЧЕСКАЯ РАБОТА №2

1. Продолжаем предыдущий пример. Создадим новую ветку выполнив команд:

```
git branch fix
```

2. В ветке master изменим файл script.js на следующее:

```
console.log('Base work!!!');
```

3. Созраним изменения в коммит:

```
git commit –am "change script.js"
git push -u origin master
```

ПРАКТИЧЕСКАЯ РАБОТА №2

4. Переключаемся в ветку fix:

```
git checkout fix
```

5. В ветке fix изменим файл script.js на следующее:

```
console.log('Fix work');
```

6. Закоммитем изменения:

```
git commit -am "change script.js"
git push -u origin fix
```

ПРАКТИЧЕСКАЯ РАБОТА №2

7. Переключимся к ветке master:

```
git checkout master
```

8. Сольём ветку fix в master:

```
git merge fix
```

9. Возник конфликт. Изменим файл script.js на:

```
console.log('Base work!!!');
console.log('Fix work');
```

10. Сохраняем изменения:

```
git commit -am "fix conflict script.js"
git push -u origin master
```