TEKNOTURF

GIT

Objectives

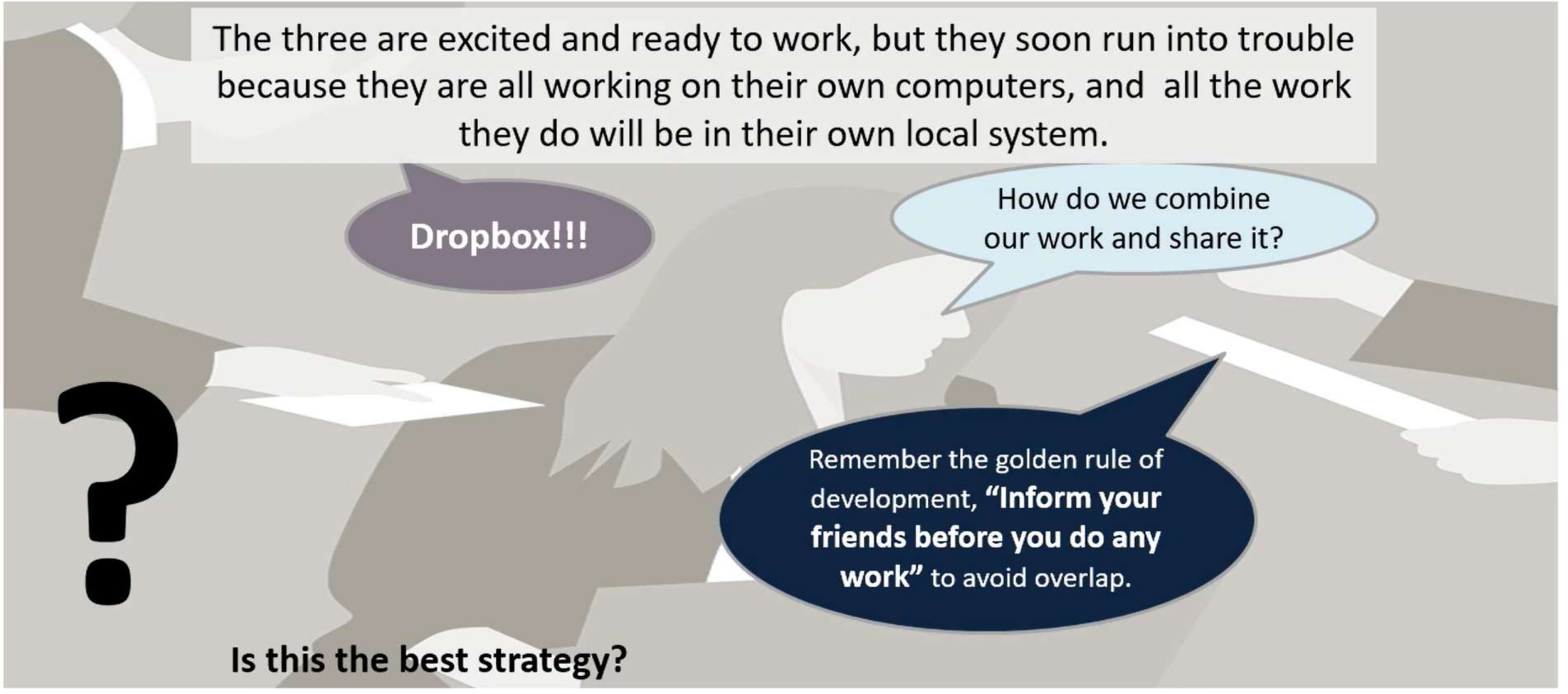
TEKNOTURF

In this module you will learn

- What is version control system?
- why VCS is used?
- GIT Introduction
- GIT Installation and Setup
- GIT Basics
- Creation of GIT Repository
- Security of artifacts
- Overview of GIT GUI

Suddenly one has become three, Lewis now has all the help he needs to really get started!!

TEKNOTURF

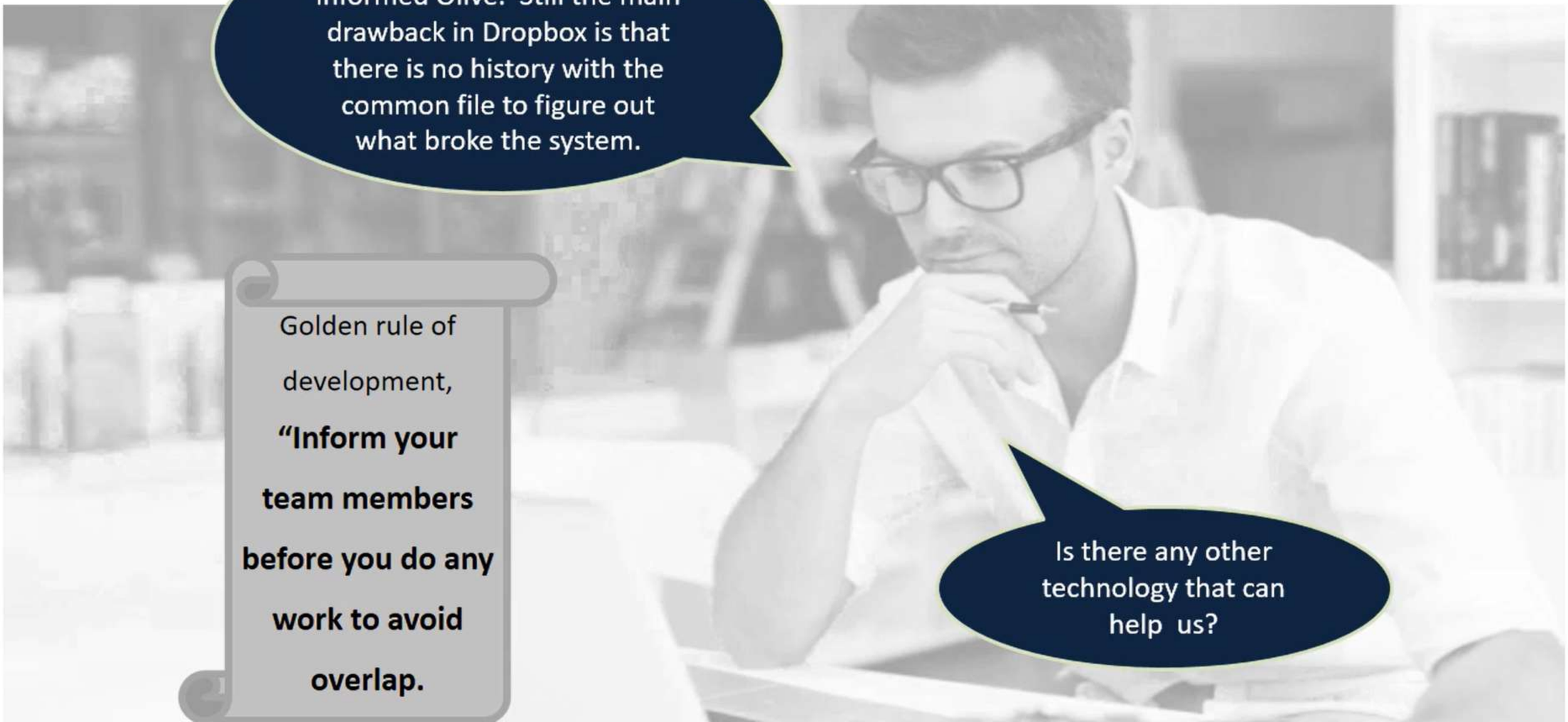

The three are excited and ready to work, but they soon run into trouble because they are all working on their own computers, and all the work they do will be in their own local system.
Dropbox!!!
How do we combine our work and share it?
?
Remember the golden rule of development, **"Inform your friends before you do any work"** to avoid overlap.
Is this the best strategy?

TEKNOTURF


Today I am going to create some new content for the module A.
I am happy with my work today. Let me update it in the shared folder in Dropbox.
I am done for the day and there is still no net connection. Let me update it in Dropbox and leave.
There is no internet connection here. Anyway, I will develop some content for module A.

TEKNOTURF

After Harry leaves, the internet connection starts working and it starts syncing the files to Dropbox. Olive's file is now overwritten with Harry's file.

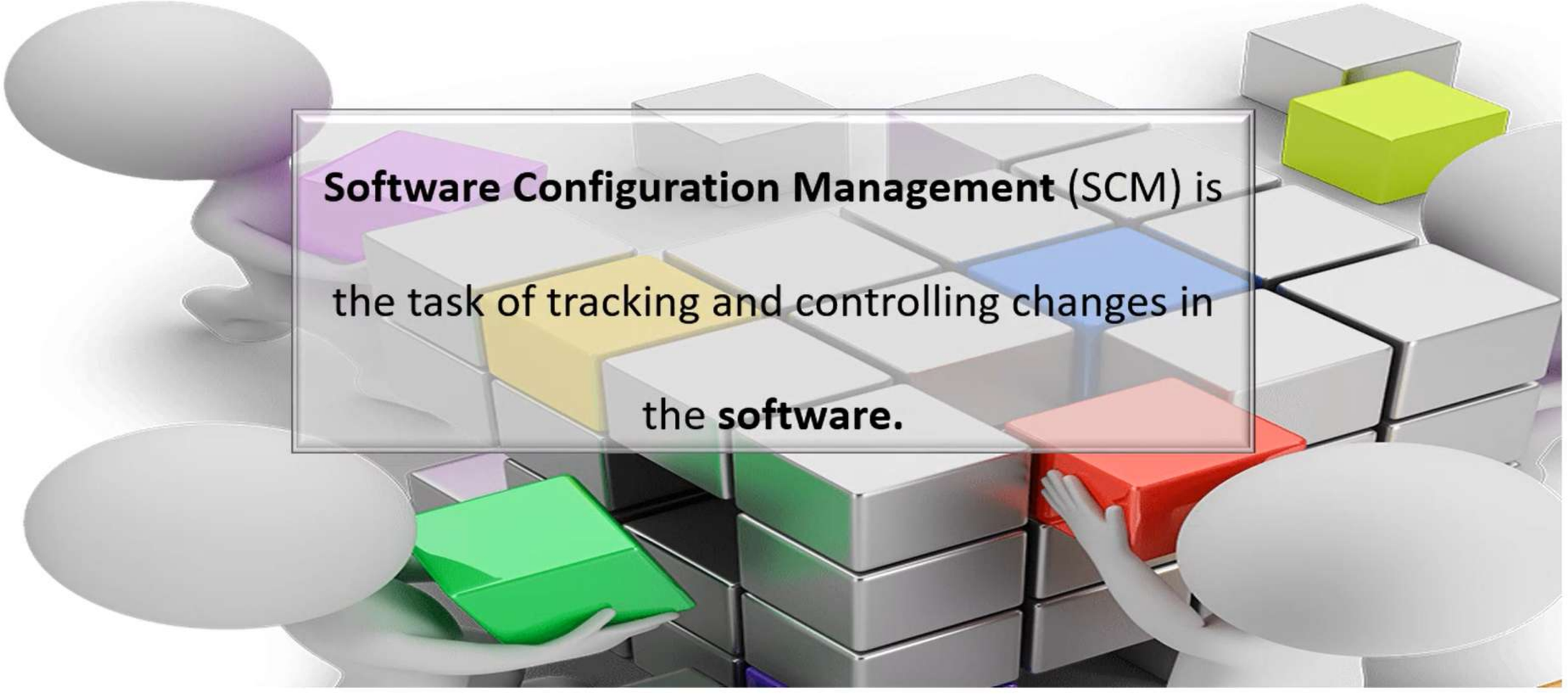

Hmm.. Harry should have informed Olive. Still the main drawback in Dropbox is that there is no history with the common file to figure out what broke the system.
Golden rule of development, **"Inform your team members before you do any work to avoid overlap.**
Is there any other technology that can help us?

TEKNOTURF

Lewis consults an expert on how to avoid 'conflicting content from collaboration'.

Configuration Management

TEKNOTURF



Software Configuration Management (SCM) is the task of tracking and controlling changes in the **software.**

Why do you need SCM?

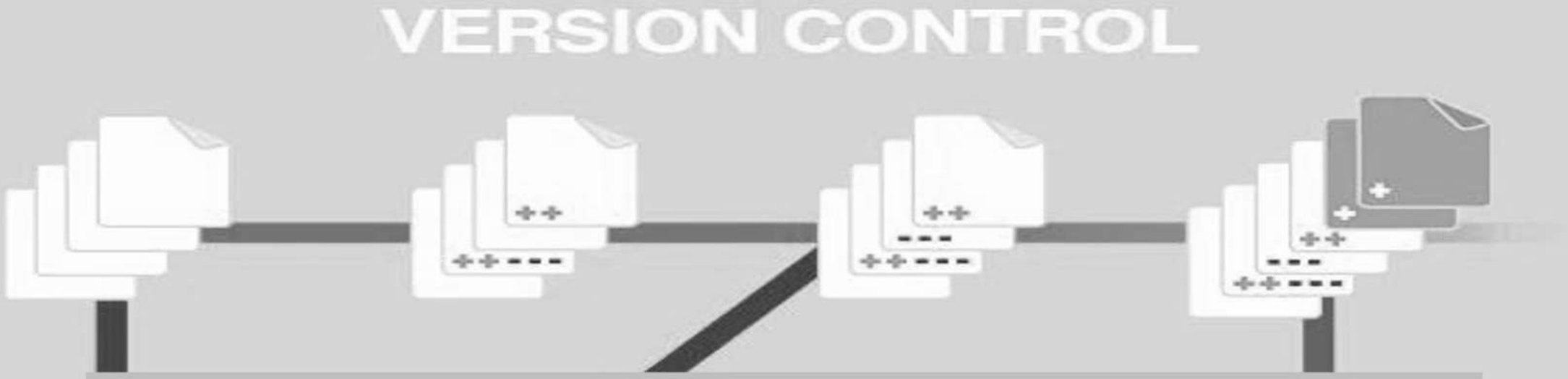
TEKNOTURF

- A developed and tested feature could go missing
- A fully tested program does not work
- Wrong version of the code is tested
- No traceability
- Programmer works on wrong versions
- The latest version of source code cannot be found
- These are systems used for maintaining different versions of the software systems and components.

Version Control System (VCS)

Version Control

TEKNOTURF



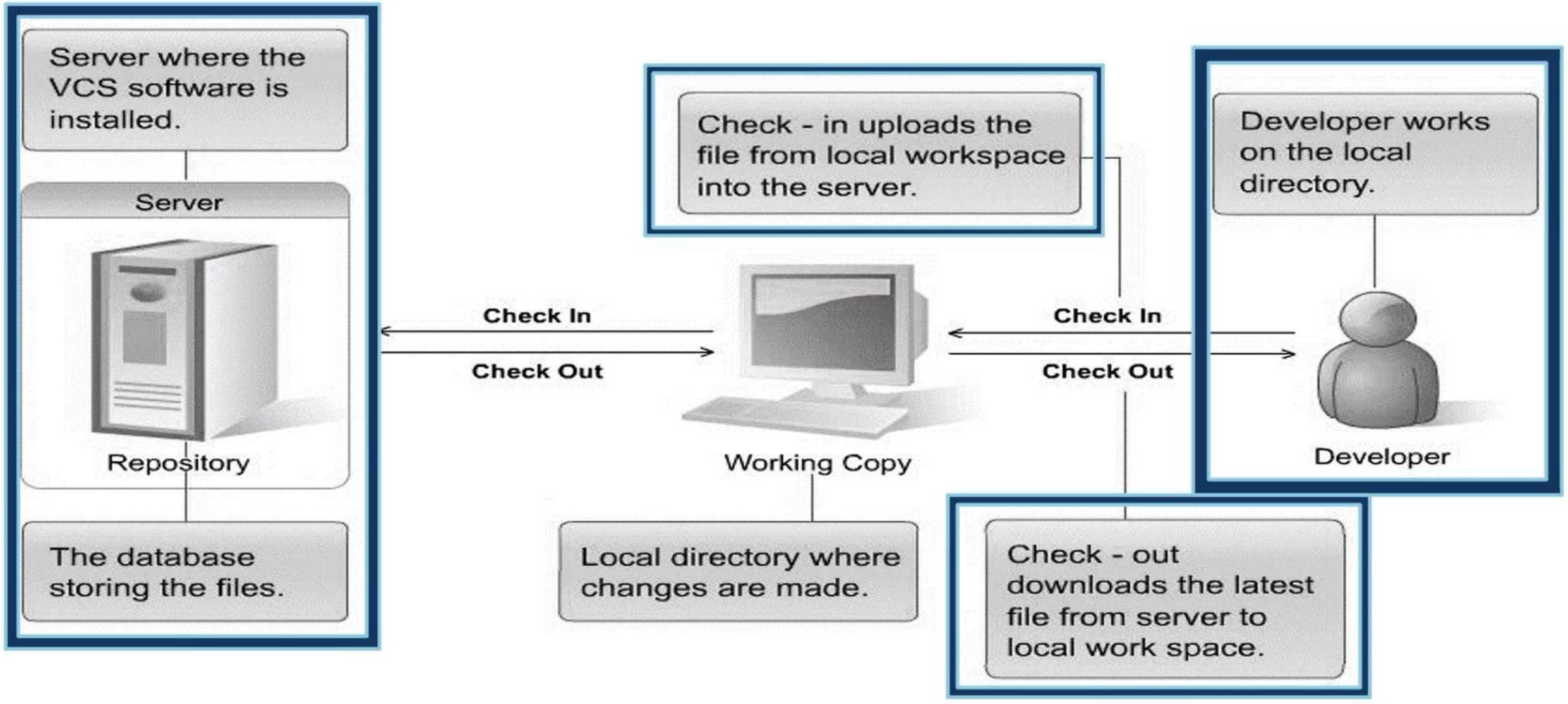
Version Control is a system that manages changes to a file or set of files over time so that it can be recalled to specific versions later. The files can be a document, images, code or anything.

Why VCS?

TEKNOTURF

- Backup and Restore
- Synchronization
- Undo Changes
- Track Changes
- Track Ownership
- Branching and Merging

Concept of VCS



Types of Version Control

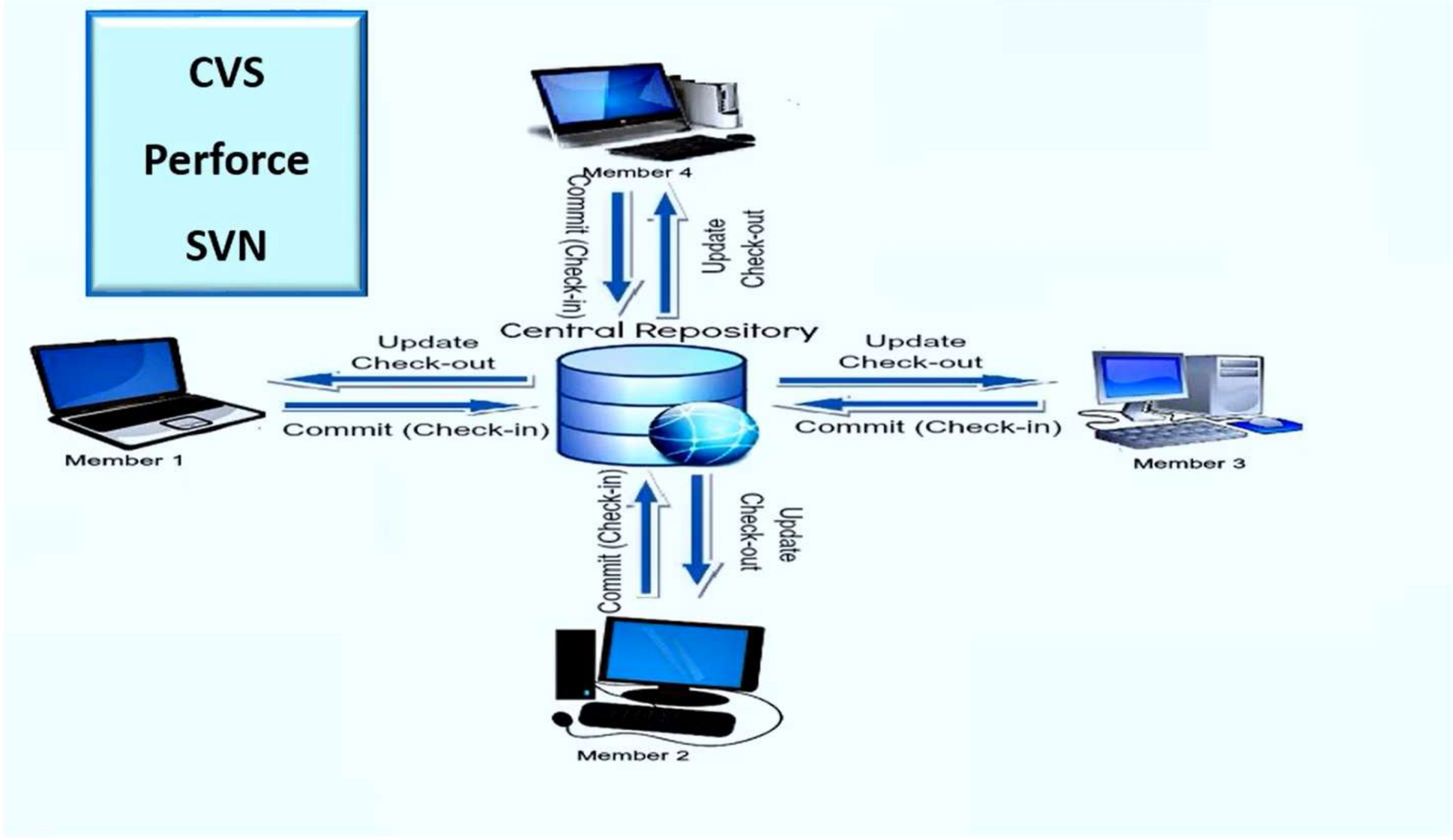
TEKNOTURF

Centralized Version Control

Distributed Version Control

Centralized Version Control

TEKNOTURF



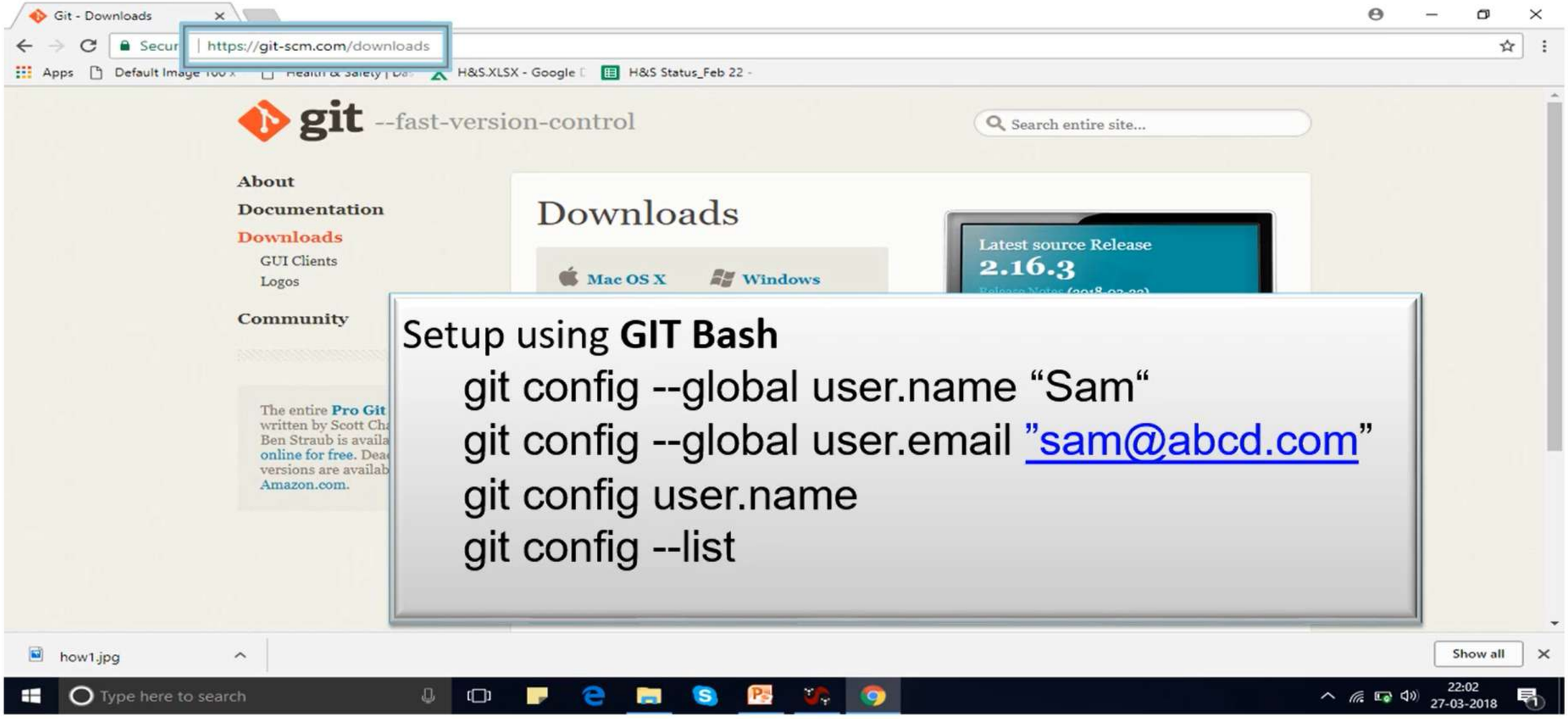
Introduction to GIT

TEKNOTURF

- **Git** is a distributed version control system for tracking changes in files and coordinating work on those files among multiple people.
- Open Source
- Developed by Linus Torvalds
- GIT provides both,
  - GIT GUI (Graphical User Interface)
  - GIT BASH (Command Line)
- Share code easily
- Keep track of any changes we make
- Maintain multiple versions of the same project/code base
- Clearly communicate what changes have been made

GIT Installation and Setup

TEKNOTURF



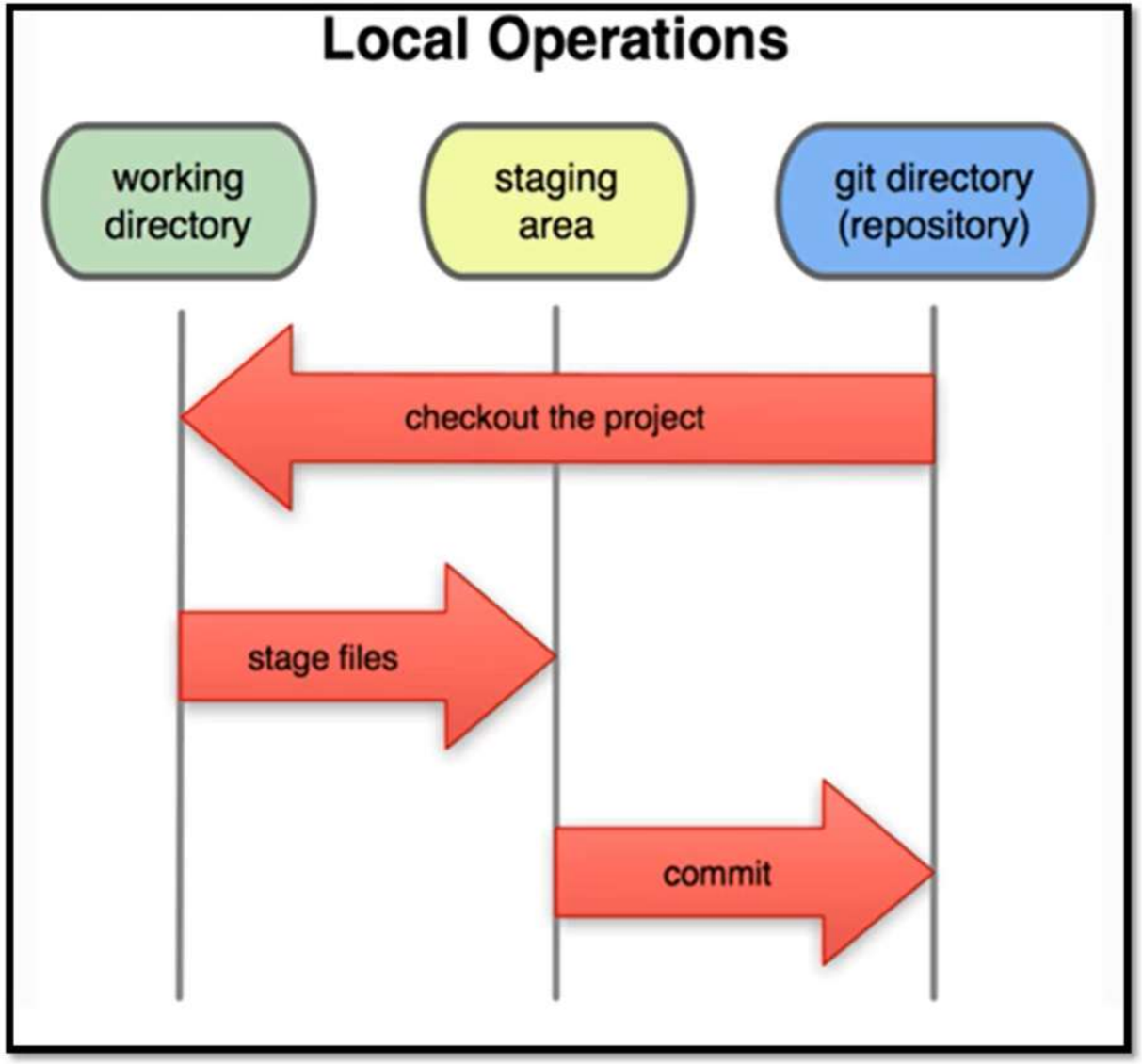
GIT Basics

TEKNOTURF

Git has three main states

that your files can reside in:

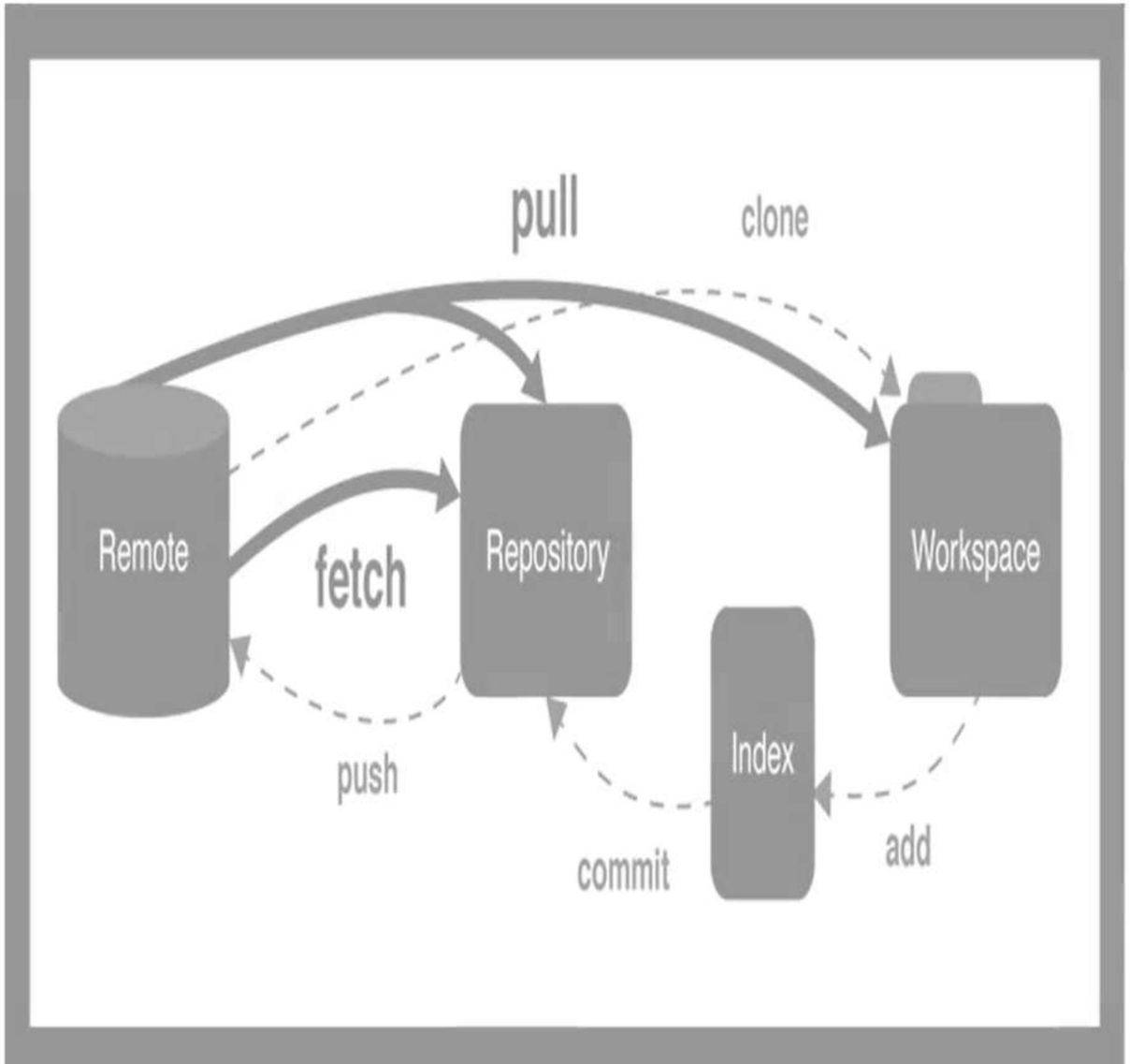
- commit
- modify
- Stage



Git Terms

TEKNOTURF

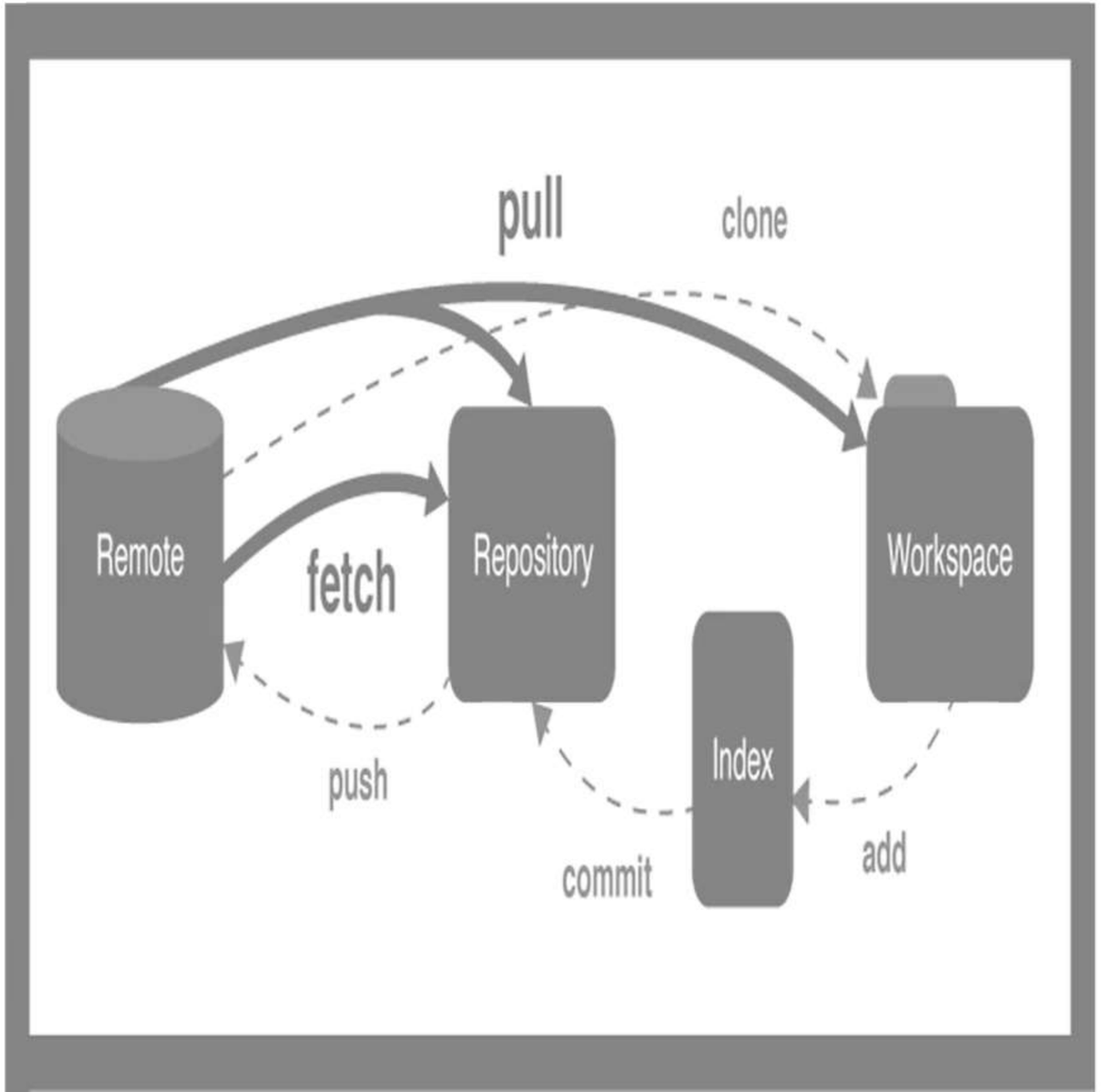
- **Repository:**
  - Git repository is like a directory that stores all the files, folders, and content needed for the project.
- **Fork:**
  - Creates a copy of a repository.
- **Branches:**
  - It can be a new version of a repository for users to alter and test changes.
- **HEAD:**
  - HEAD is a reference variable used to denote the most current commit of the repository in which you are working.



Git Terms contd..

TEKNOTURF

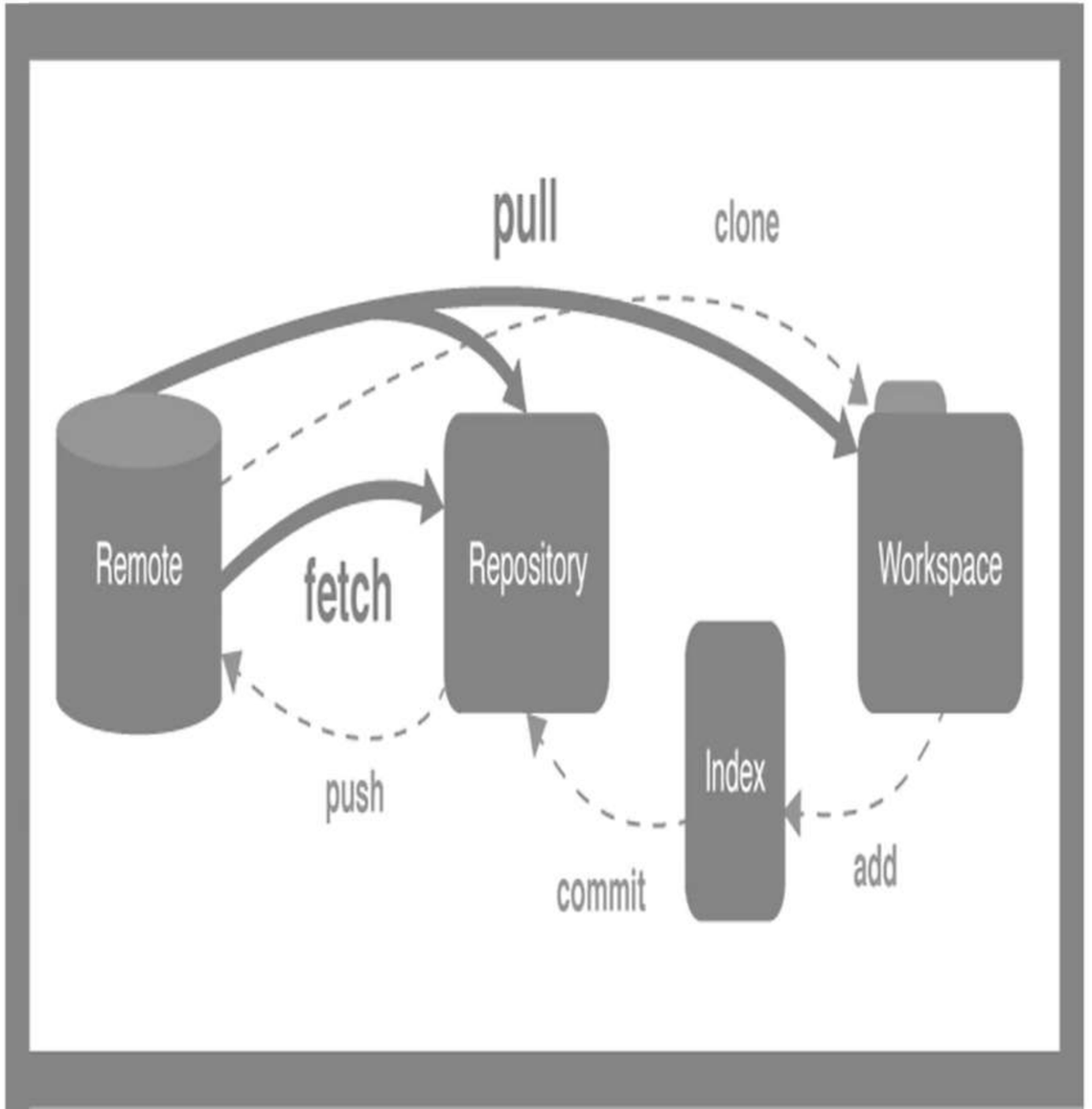
- **Checkout:**
  - The git checkout command is used to switch branches in a repository.
- **Fetch:**
  - downloading and copying that branch's files to your workstation.
- **Index:**
  - Files that have been modified will be staged within the index until you are ready to commit the files.
- **Master:**
  - The primary branch of all repositories.
- **Merge:**
  - Taking the changes from one branch and adding them into another branch.



Git Terms contd..

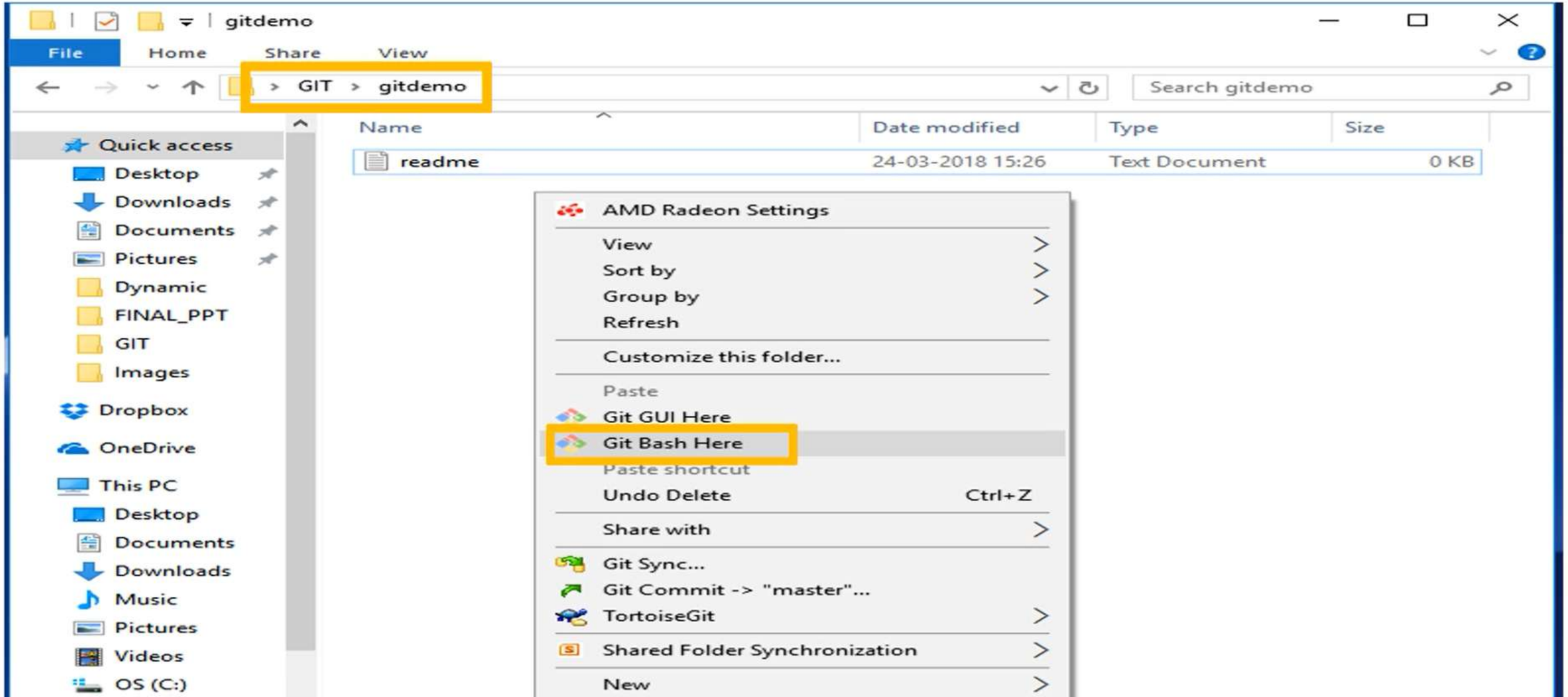
TEKNOTURF

- **Pull/Pull Request:**
  - A pull happens when adding the changes to the master branch.
- **Push:**
  - Pushing the changes on to the remote.
- **Tag:**
  - The most important portions of project history is marked using tags.
- **Upstream:**
  - This can be considered where you *push* your Git changes
- **Origin:**
  - The conventional name for the primary version of a repository.



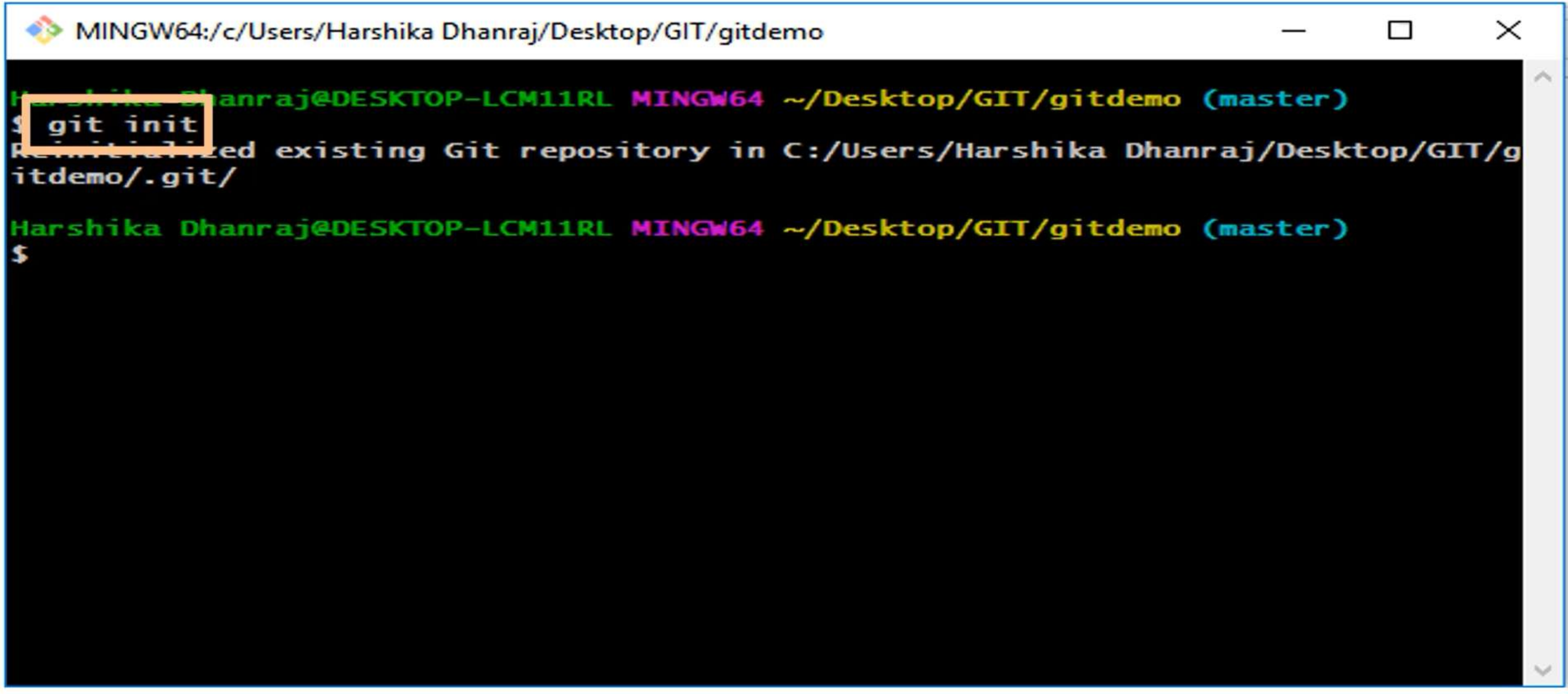
Creation of Git Repository

TEKNOTURF



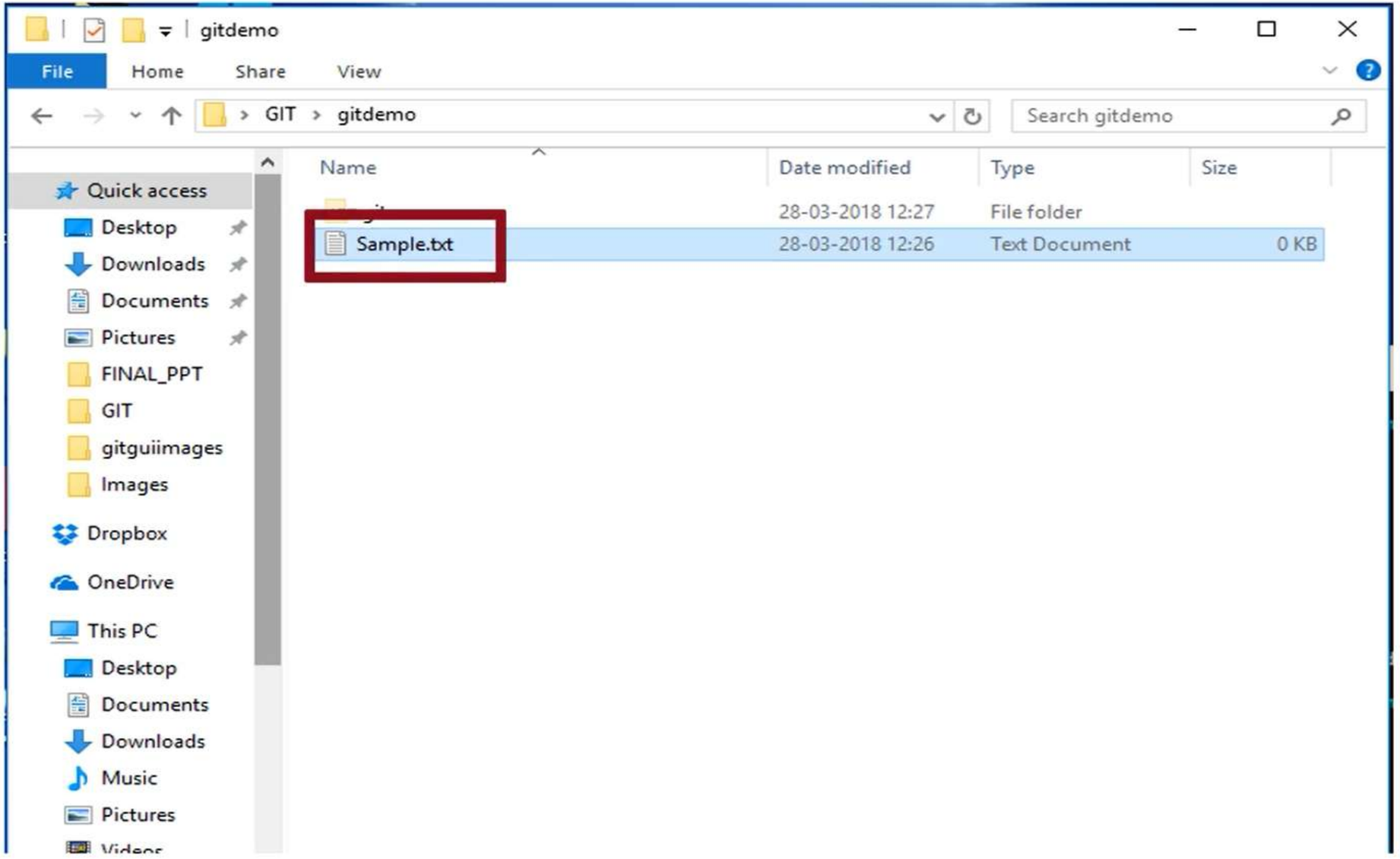
git init

TEKNOTURF



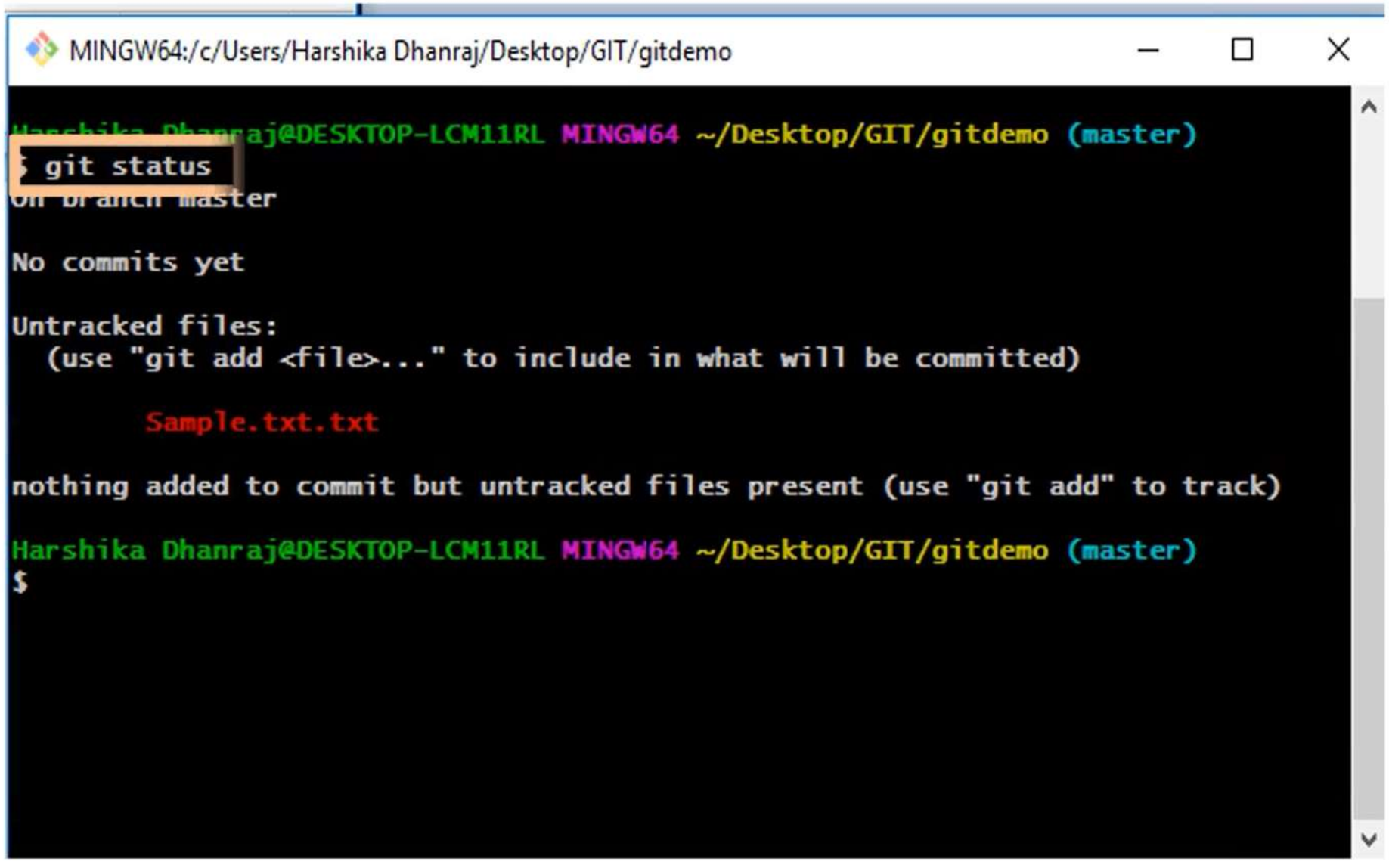
Adding Files to Local Repository

TEKNOTURF



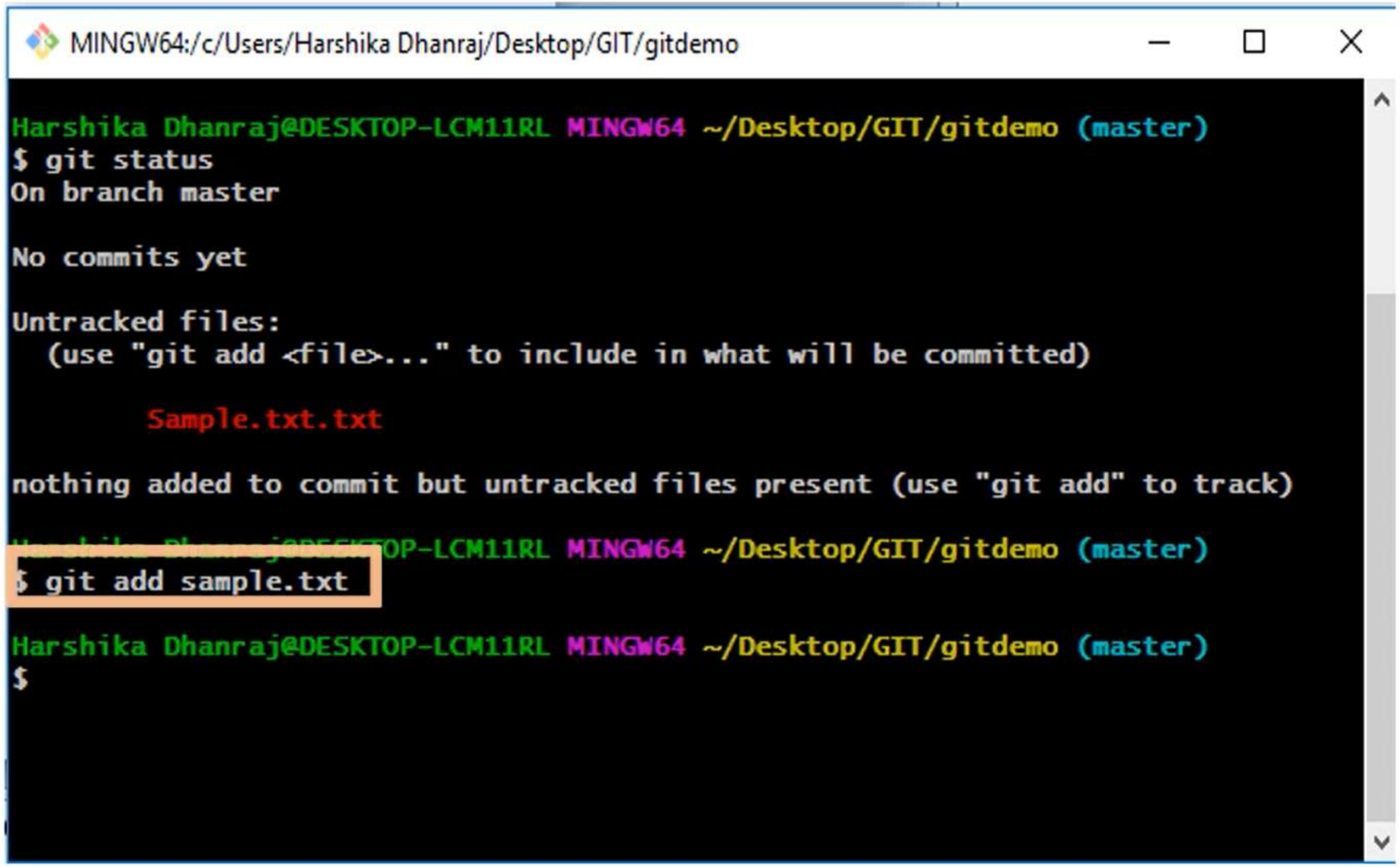
Adding Files to Local Repository

TEKNOTURF



Adding Files to Local Repository

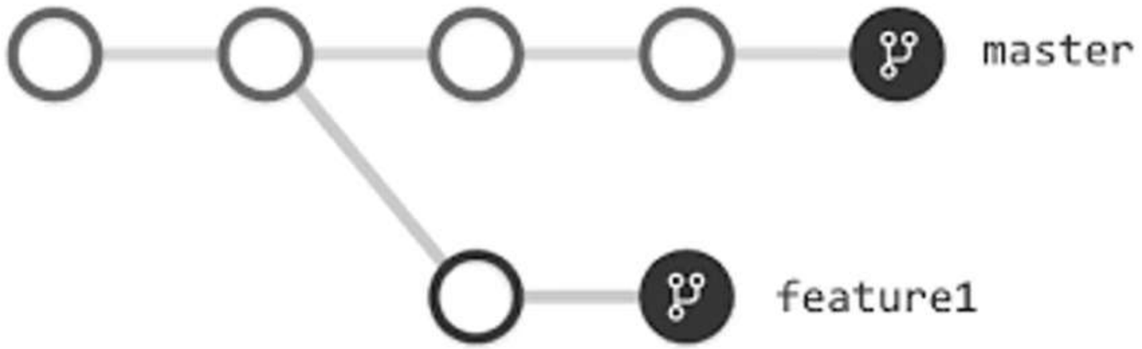
TEKNOTURF



Branching

TEKNOTURF

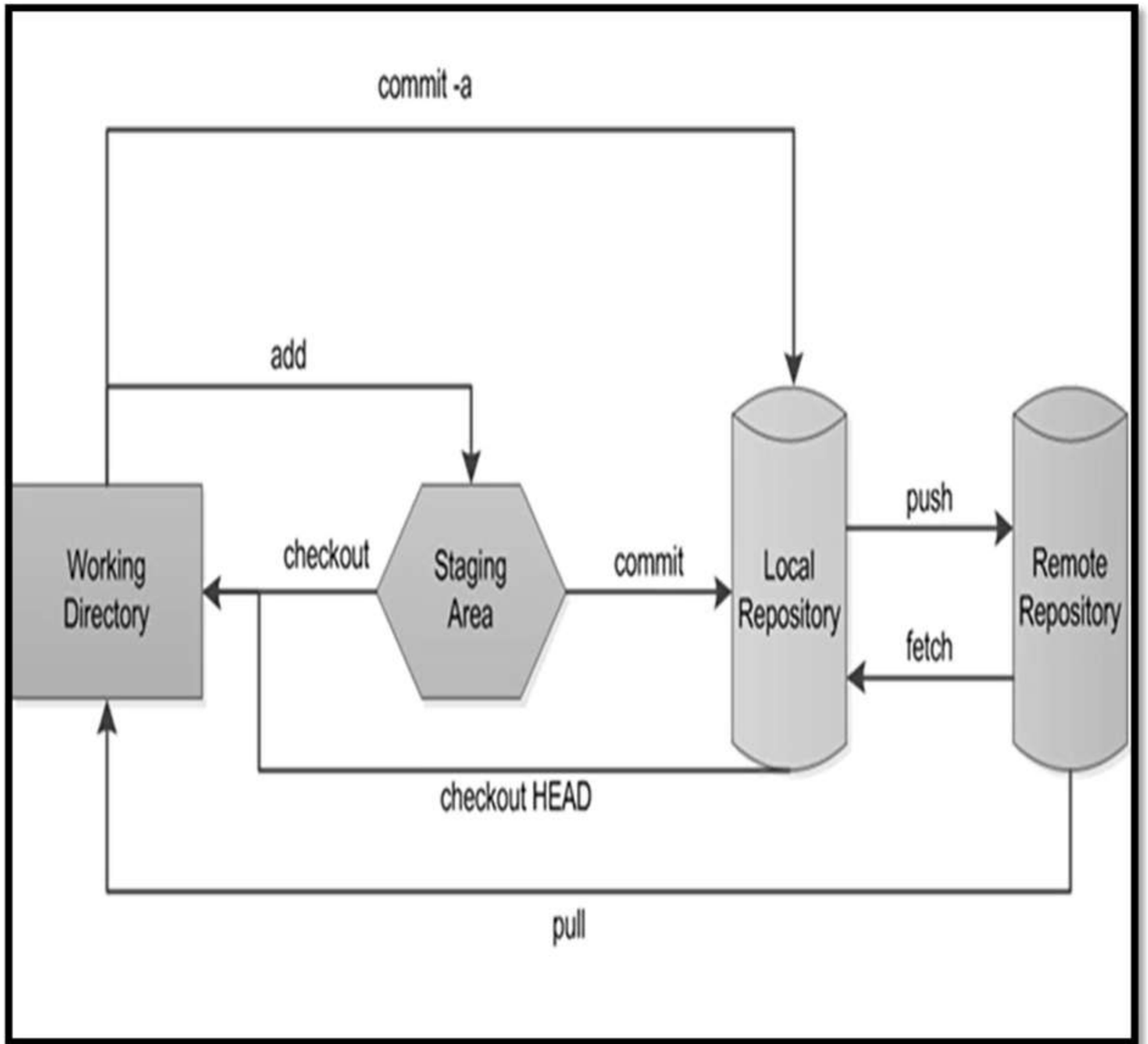
- A **branch in Git** is simply a lightweight movable pointer to one of these commits.
- The default **branch** name in **Git** is *master*.
- To create a new branch *git branch* command is used.



Checkout

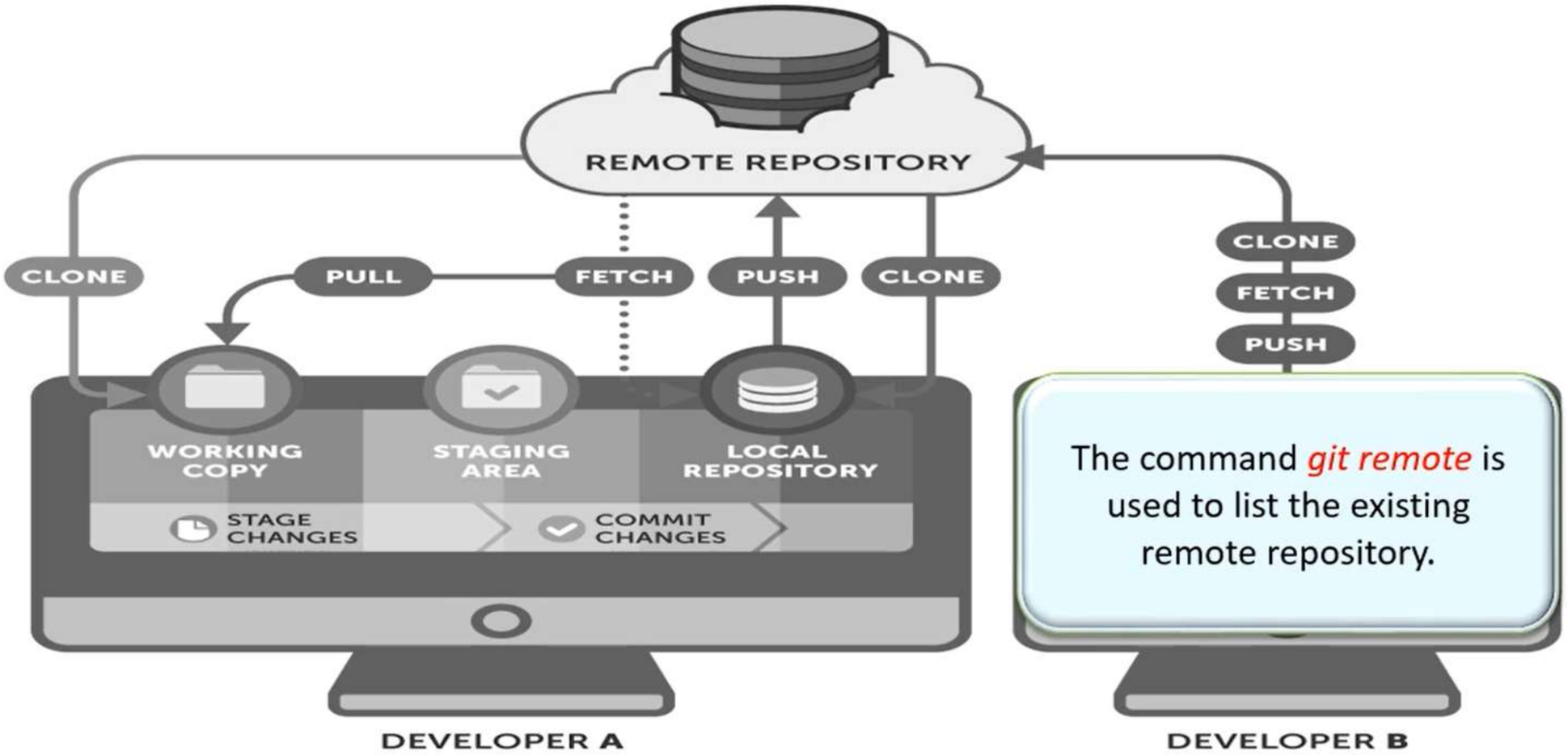
TEKNOTURF

- To switch between branches, **git checkout** command is used.
- Changes done in one branch do not get reflected in the master branch.
- To merge two branches, **git merge** command is used.



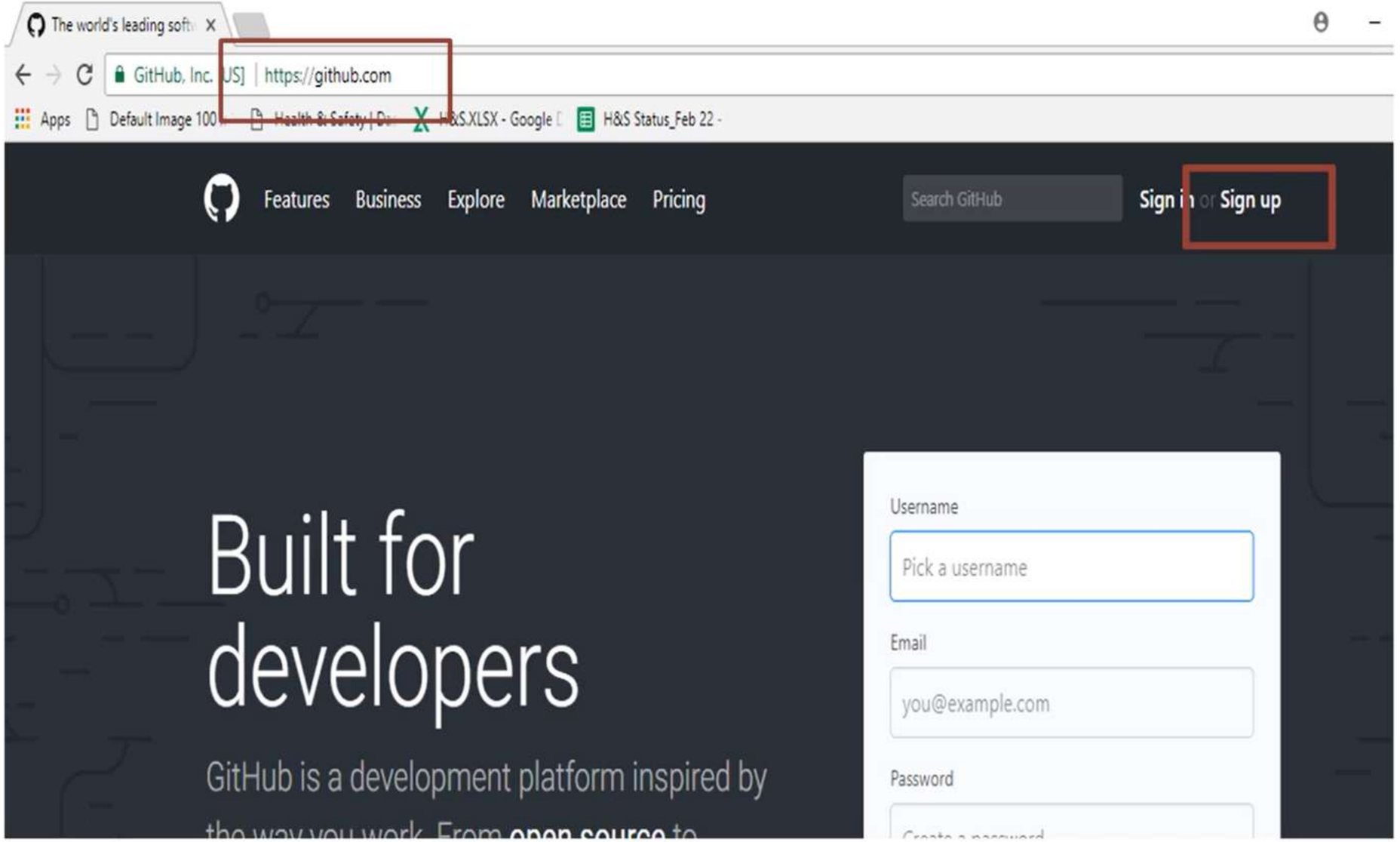
Remote Repository

In GIT, retrieving of files is done manually and the changes should be pushed to the remote repository.



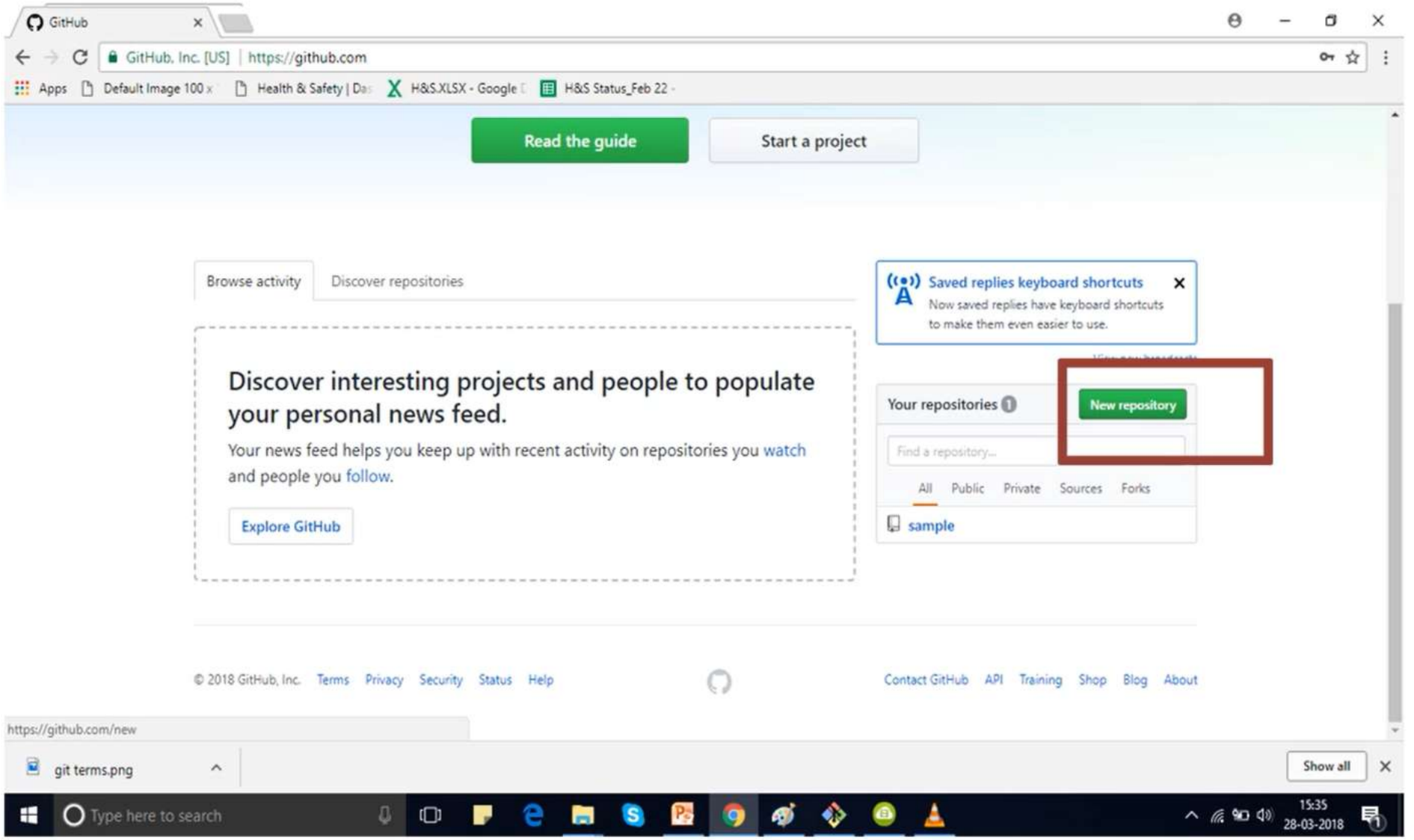
GitHub

TEKNOTURF



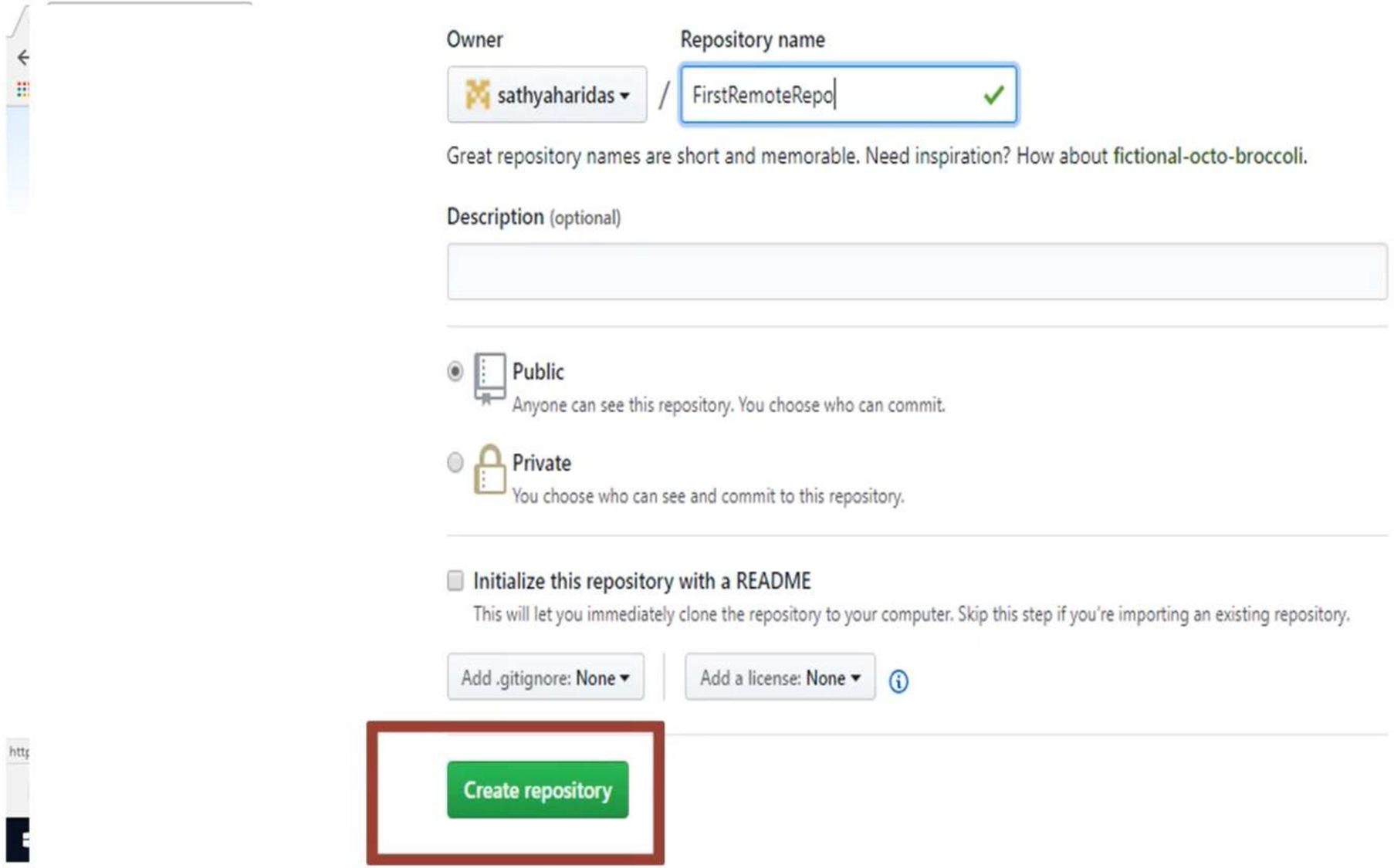
GitHub

TEKNOTURF

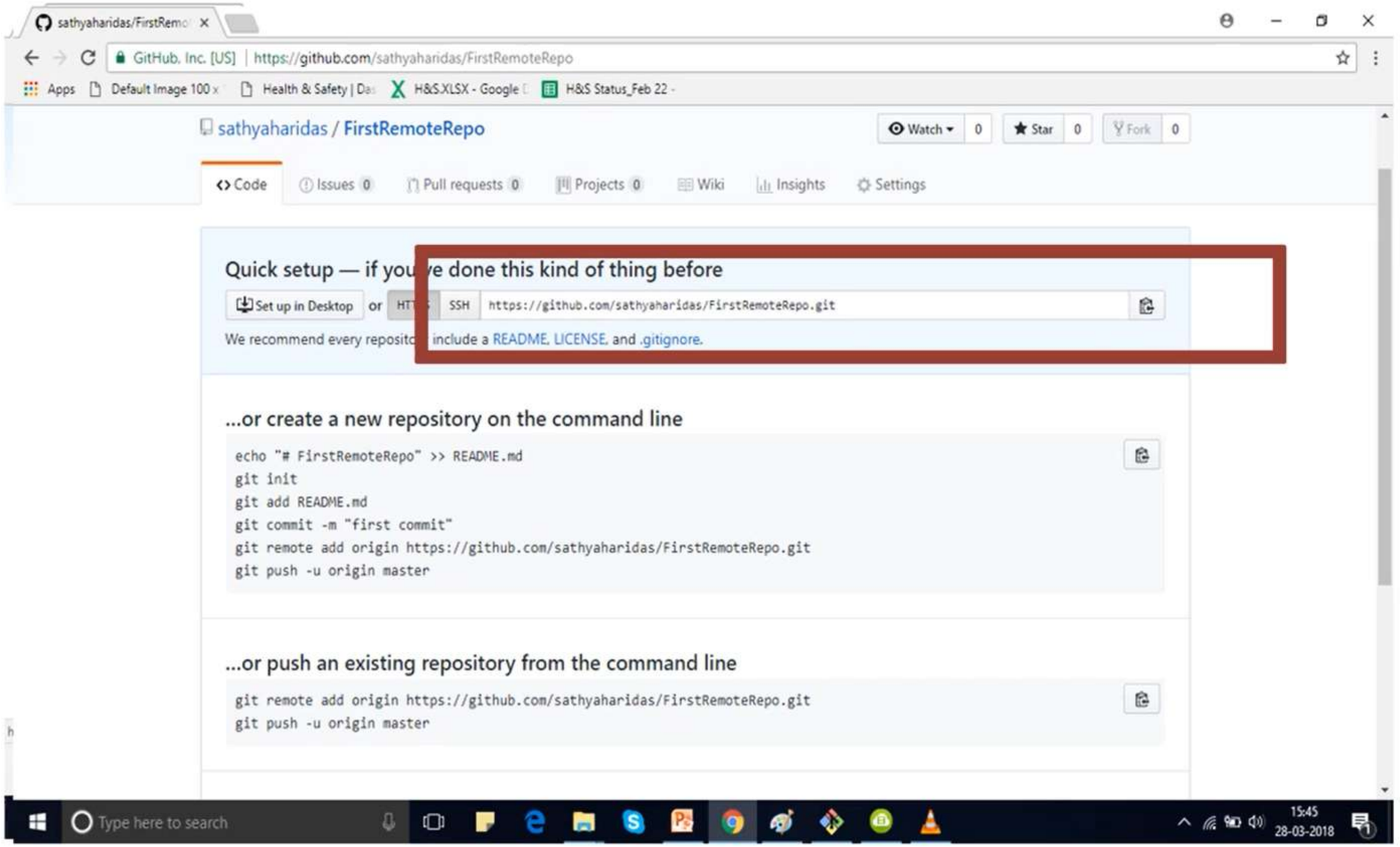


GitHub

TEKNOTURF



GitHub


sathyaharidas/FirstRemo
GitHub, Inc. [US] | https://github.com/sathyaharidas/FirstRemoteRepo
Apps
Default Image 100
Health & Safety | Da
H&S.XLSX - Google
H&S Status_Feb 22
sathyaharidas / FirstRemoteRepo
Watch 0
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/sathyaharidas/FirstRemoteRepo.git
We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# FirstRemoteRepo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/sathyaharidas/FirstRemoteRepo.git
git push -u origin master
...or push an existing repository from the command line
git remote add origin https://github.com/sathyaharidas/FirstRemoteRepo.git
git push -u origin master
Type here to search
15:45
28-03-2018

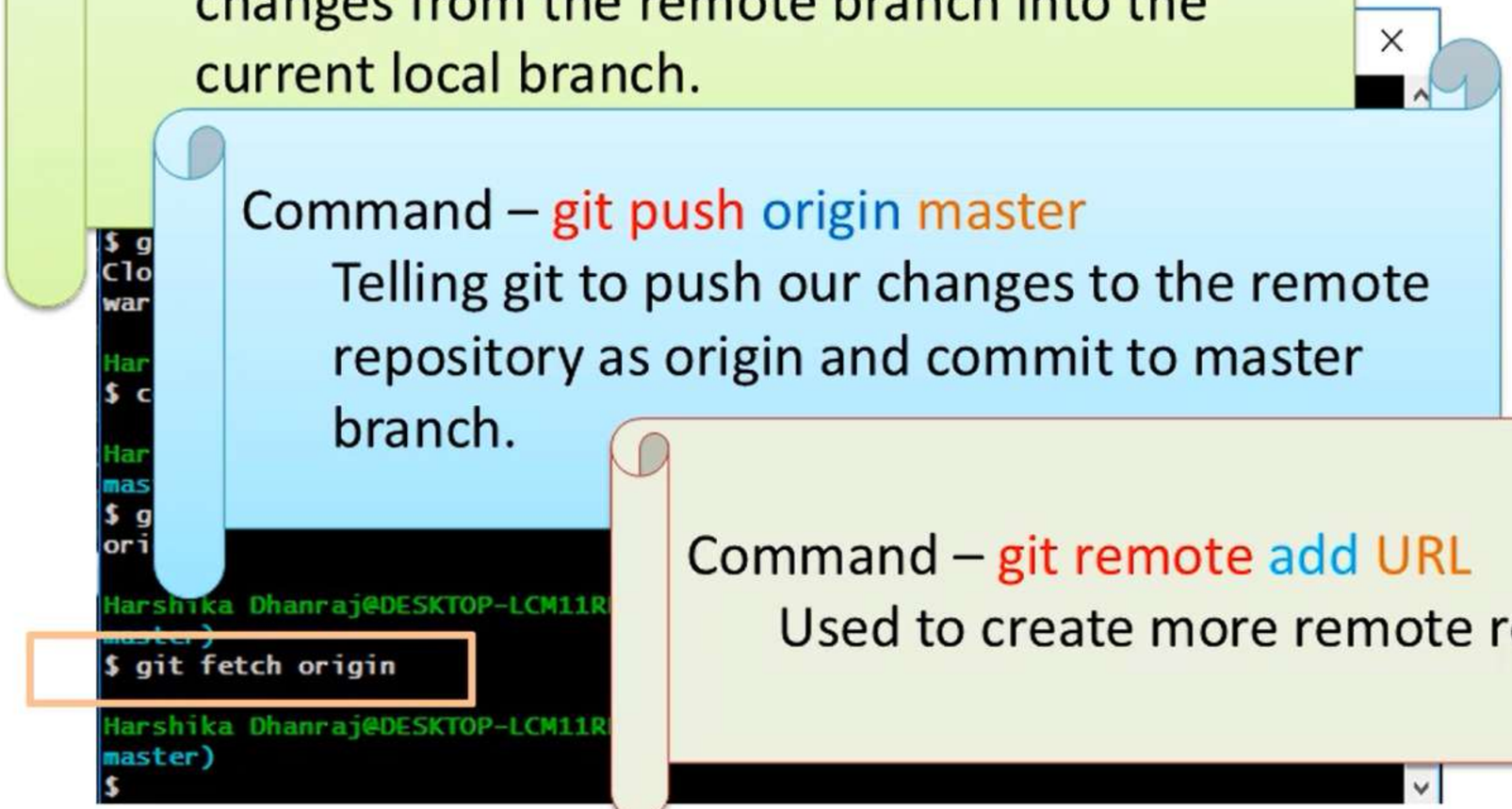
GitHub Cont...

Command – git pull origin

Will automatically fetch and merge the changes from the remote branch into the current local branch.

Command – git push origin master

Telling git to push our changes to the remote repository as origin and commit to master branch.

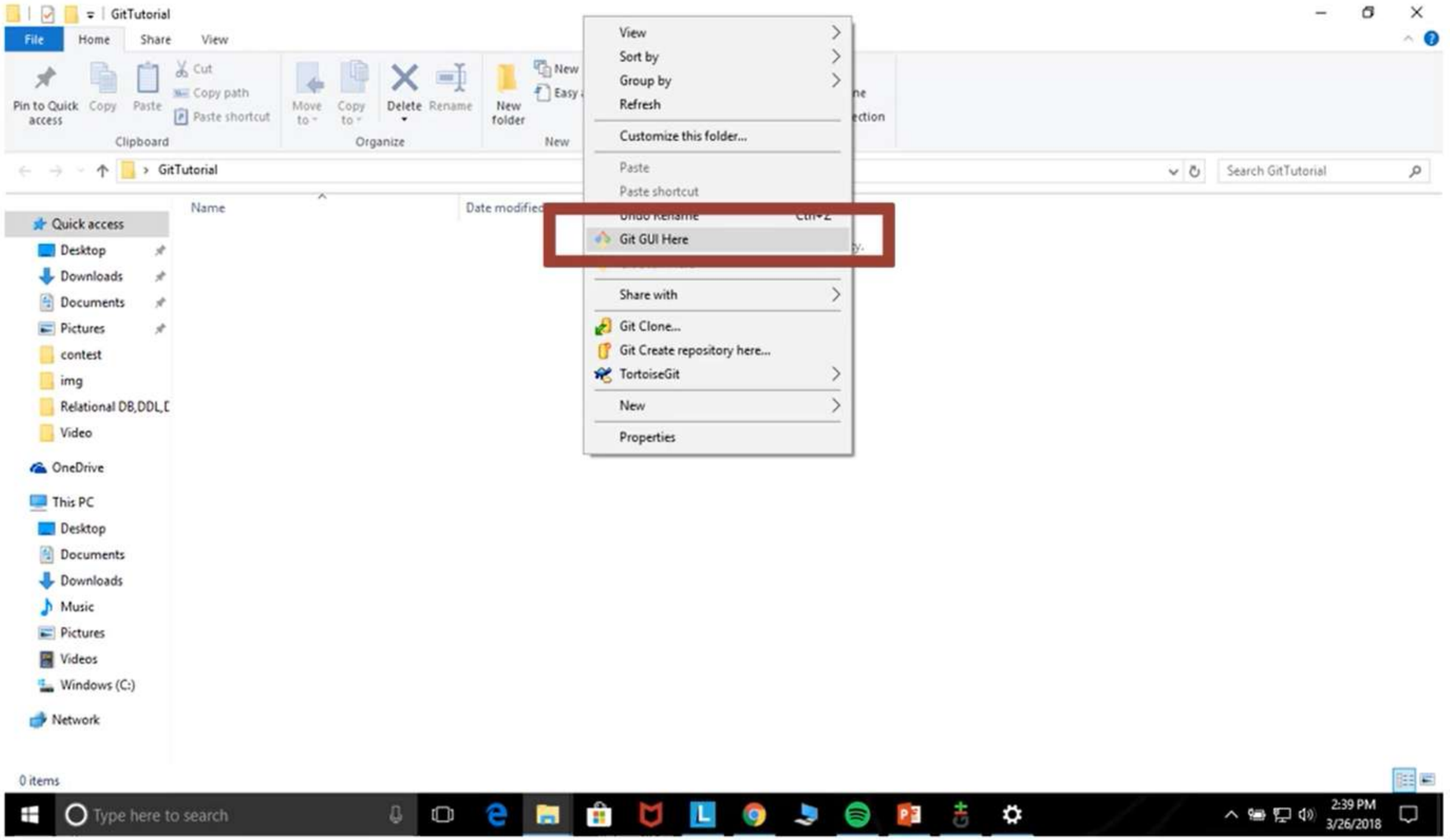


Command – git remote add URL

Used to create more remote repository.

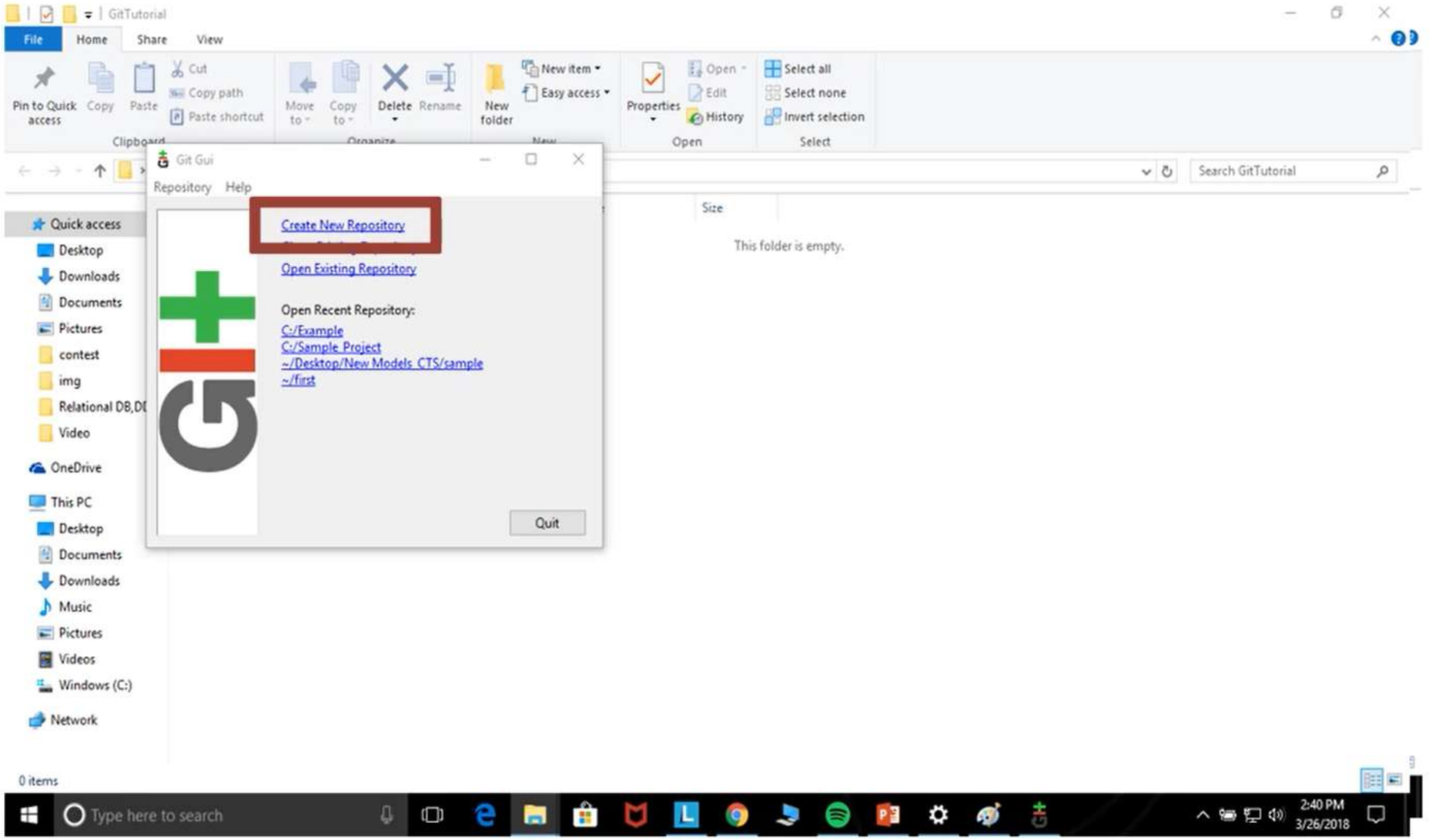
GIT GUI

TEKNOTURF



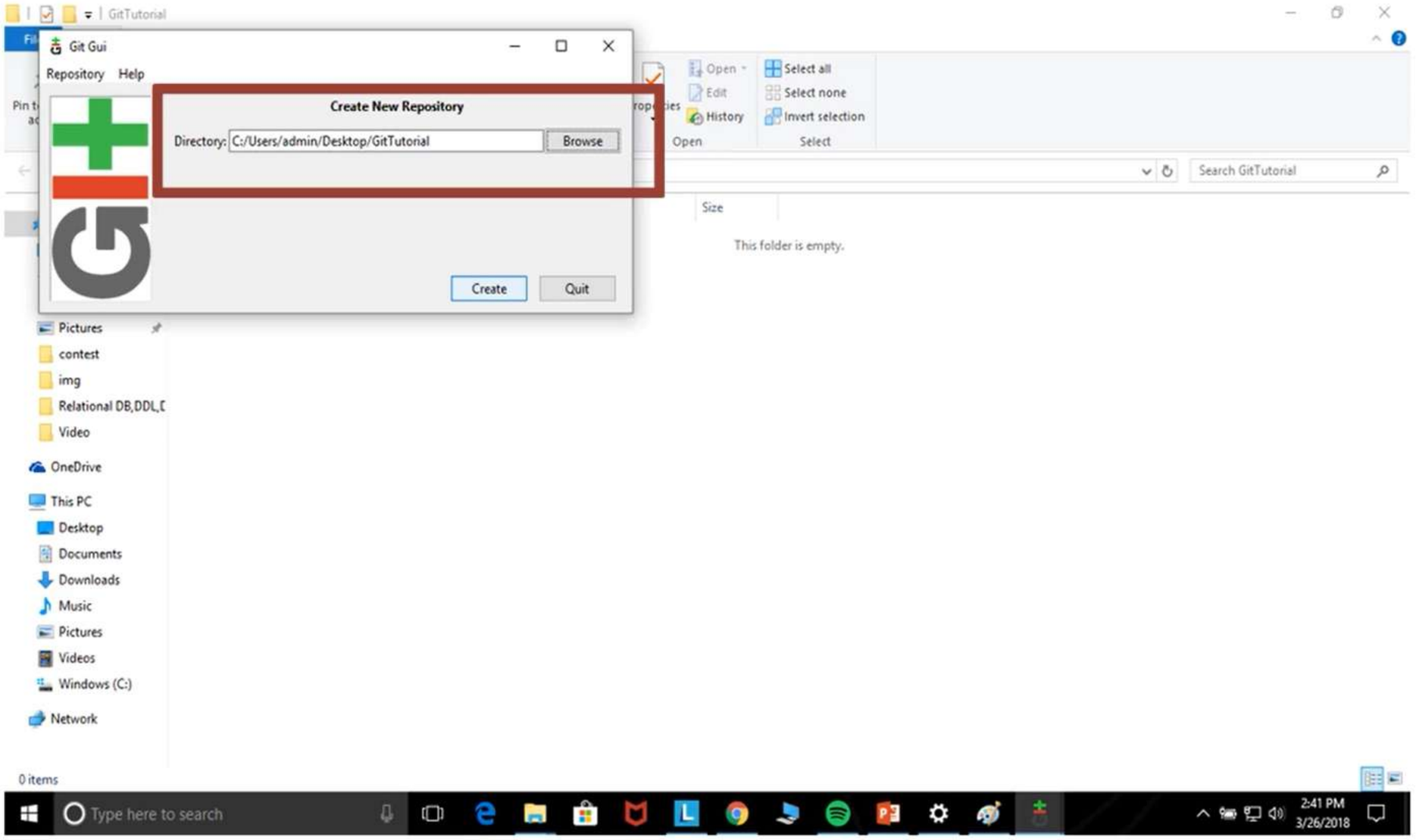
GIT GUI

TEKNOTURF



GIT GUI

TEKNOTURF



Security of artifacts

TEKNOTURF

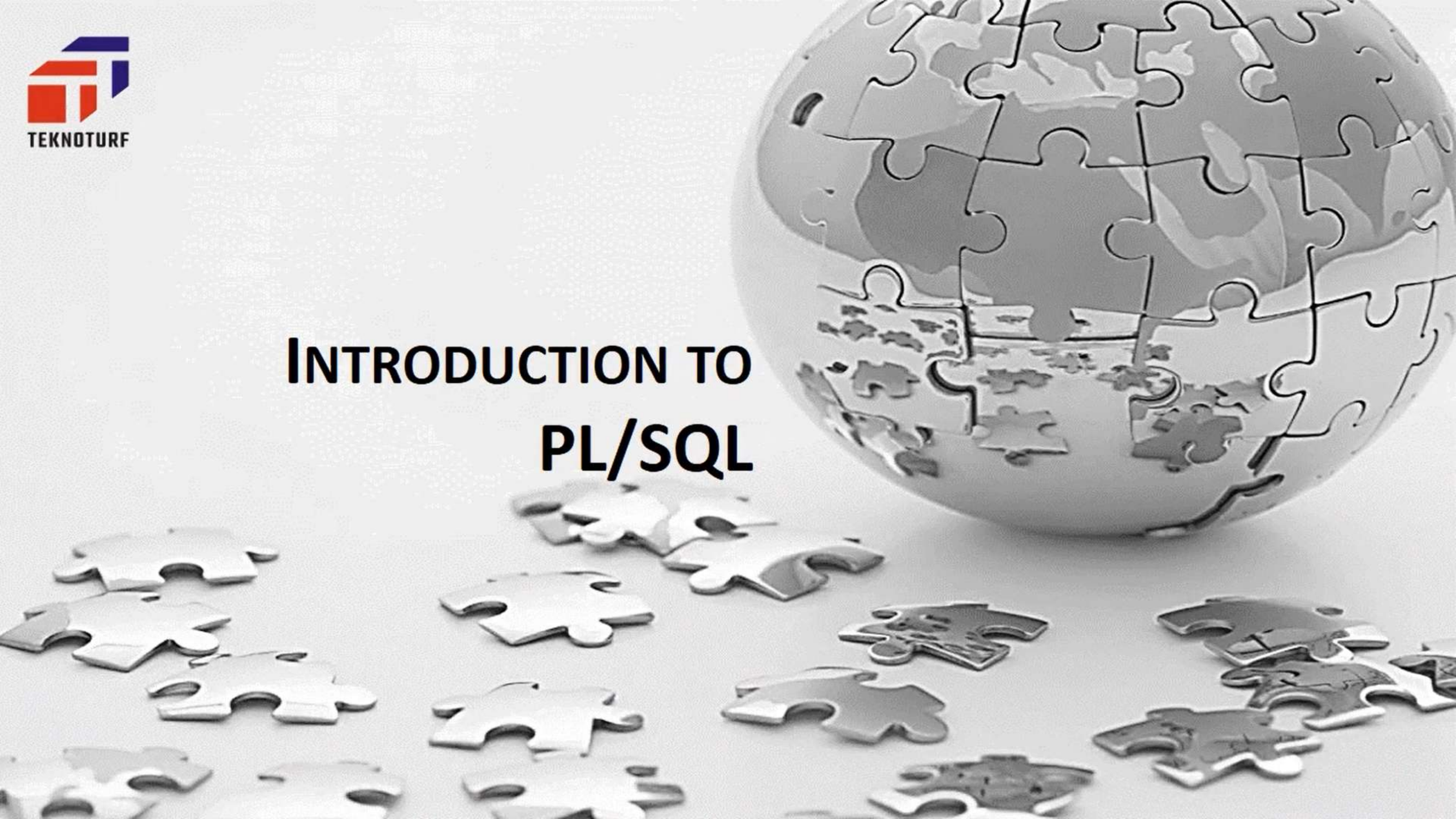
- **git-secret**
- It encrypts files and stores them inside the git repository, and it will have all the changes for every commit.
- It doesn’t require any other deploy operations other than git secret reveal, so it will automatically decrypt all the required files.

Summary

- Version control system
- Usage of VCS
- Introduction to GIT
- Installation of GIT and Setup
- Basics of GIT
- Creation of GIT Repository
- Overview of GIT GUI
- Security of artifacts

TEKNOTURF

INTRODUCTION TO PL/SQL



In this Module you will learn

TEKNOTURF

- PL/SQL Basics
- PL/SQL Block Structure
- Handling Variables in PL/SQL
- Scope and Visibility of Variables
- SQL in PL/SQL
- Programmatic Constructs

Scenario

TEKNOTURF

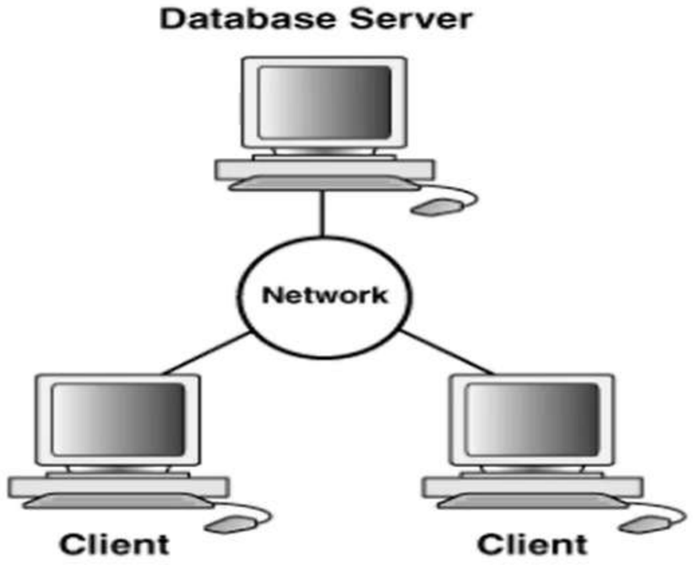
Teena is preparing for her competitive exam. She needs some clarifications in her lesson, so she plans to contact her teacher Mr. Johnson. Before making the call she notes down her queries in a paper, so that in one call she can get clarification for all her queries instead of making multiple calls.

Scenario

TEKNOTURF

As seen in Teena's scenario, in PL/SQL also the client requests are bundled in a block and sent to the server and the server sends the response back to the client.

I.e. instead of sending each query as a different request, one block will fetch all the queries to the server as a single request, which will reduce the network traffic between Client and Server.



PL /SQL

TEKNOTURF

Procedural Language supported by Oracle.

Used to automate business criteria

- by creating the stored procedure and functions
- trigger database events for certain business conditions

Offers modern software engineering features such as

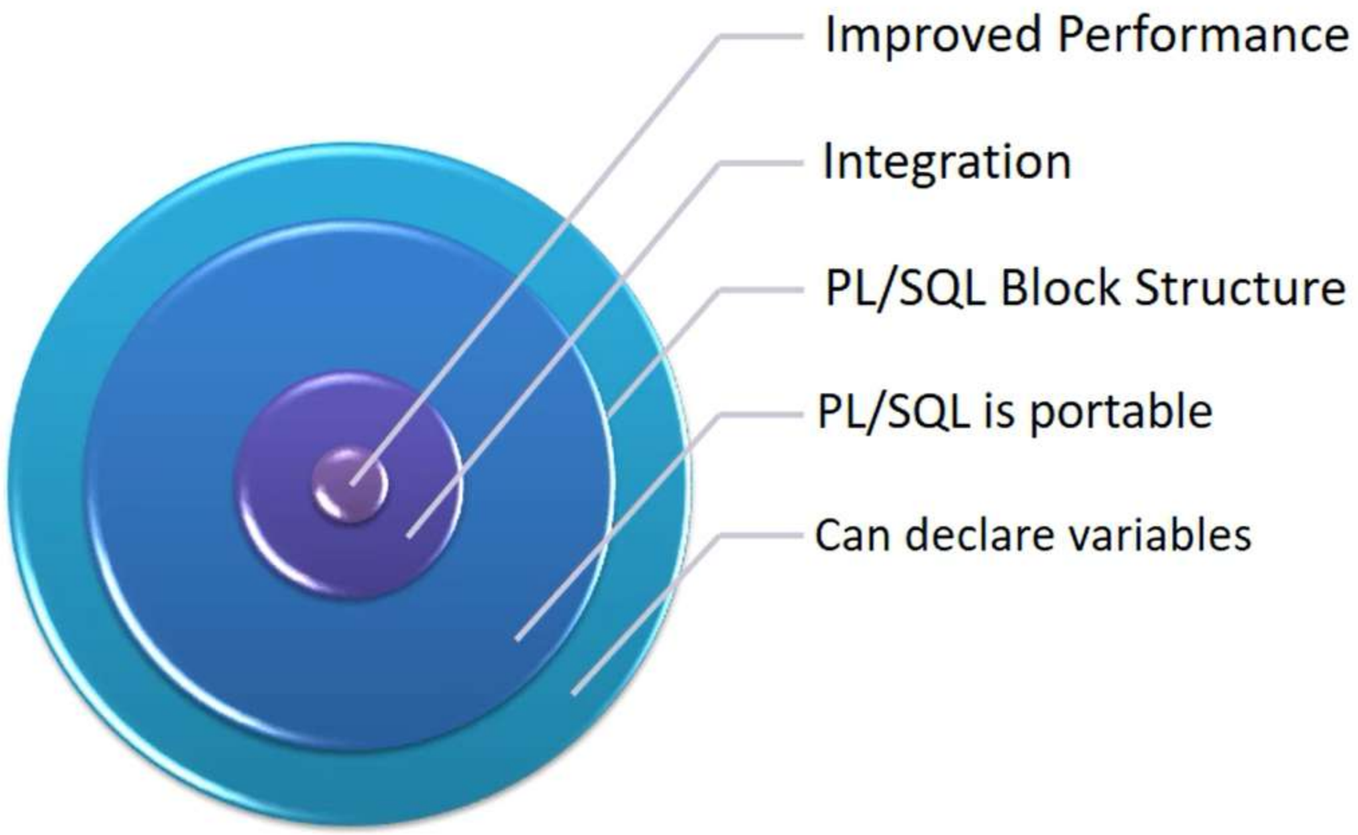
- data encapsulation
- exception handling
- information hiding
- object orientation

Allows to

- declare variables / constants,
- define programs / procedures and
- capture and deal with run-time errors.

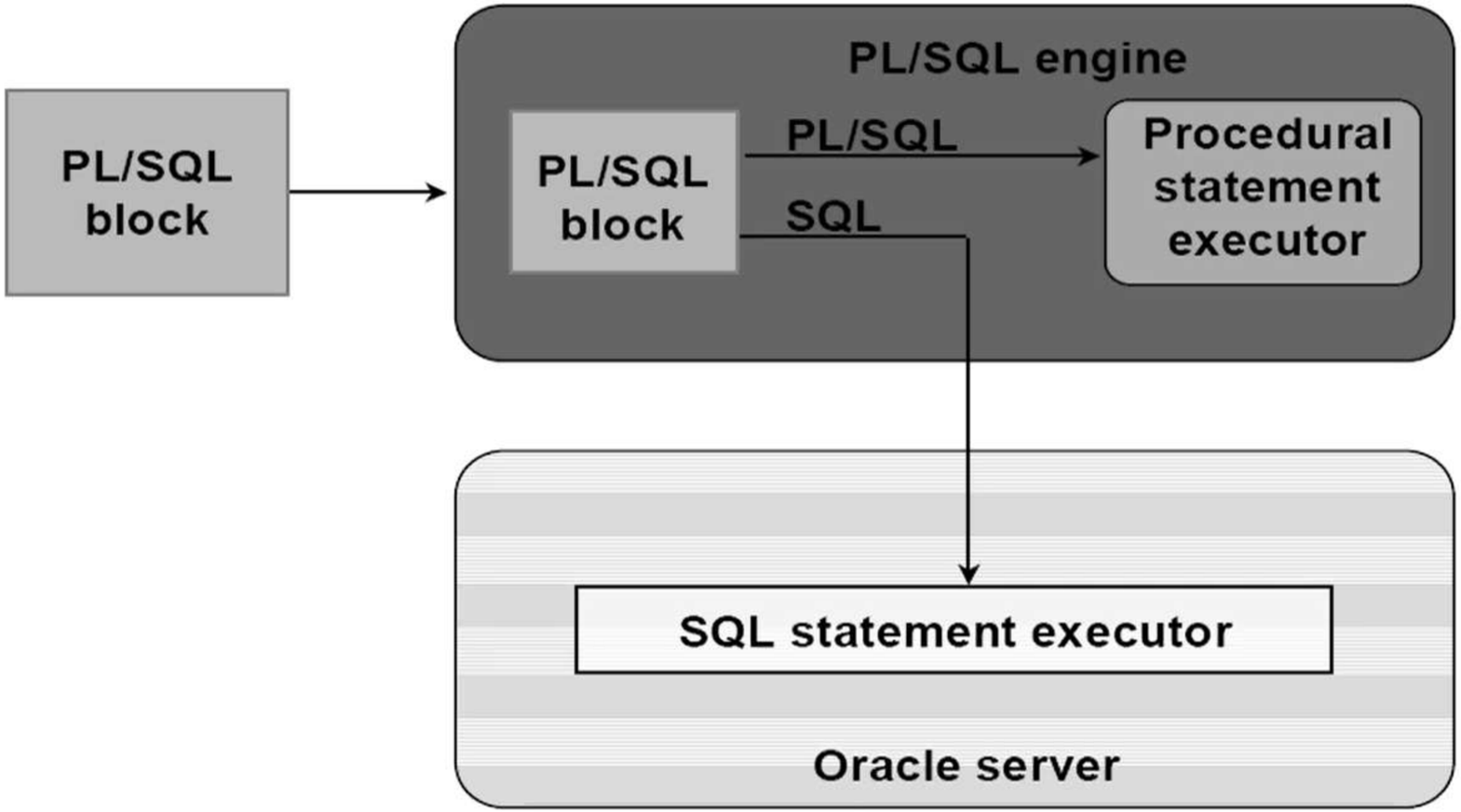
Benefits of PL /SQL

TEKNOTURF



PL/SQL Environment

TEKNOTURF



PL/SQL – Block Structure

TEKNOTURF

PL/ SQL statements are written and grouped into structures called as blocks

Each block is given name

A PL / SQL block without a name is called as an anonymous block

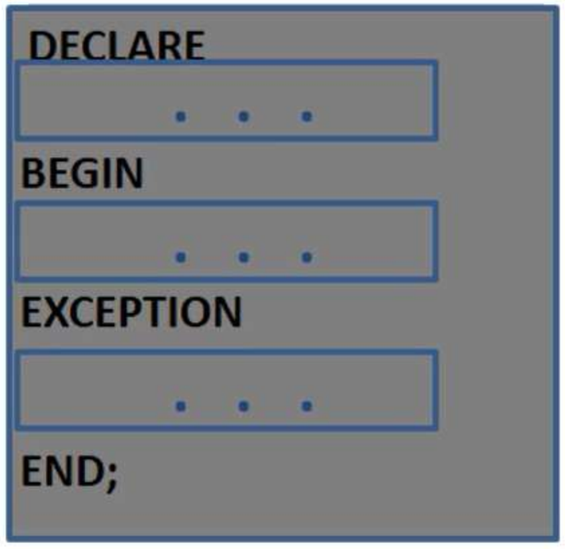
A PL / SQL block is organized into following sections

- An optional Declaration section
- A mandatory Execution section
- An optional Error Handling section

PL/SQL – Block Structure

TEKNOTURF

DECLARE (Optional)
Variables, cursors, User Defined Exceptions
BEGIN (Mandatory)
- SQL Statements
- PL / SQL Statements
EXCEPTION (Optional)
Actions to perform when error occurs
END; (Mandatory)



PL/SQL – Block Structure

TEKNOTURF

Section	Description	Inclusion
Declarative	Contains all variables, constants, cursors and user defined exceptions that are referenced in the executable and declarative sections	Optional
Executable	Contains SQL statements to manipulate data in the database and PL/SQL statements to manipulate data in the block.	Mandatory
Exception Handling	Specifies the action to perform when errors or abnormal conditions arise in the executable section	Optional

PL / SQL Block

TEKNOTURF

```
DECLARE
        v_variable varchar2(30);
BEGIN
        SELECT column_name INTO  v_variable  from
                                        table_name ;
EXCEPTION
        WHEN exception_name THEN
                ..............
END;
```

- Place a semicolon (;) at the end of a SQL statement or PL/SQL control statement.
- Section keywords DECLARE, BEGIN, and EXCEPTION are not followed by semicolons.
- END and all other PL/SQL statements require a semicolon to terminate the statement.

PL / SQL – Block Types

TEKNOTURF

Anonymous

```
[DECLARE]

BEGIN
  --statements
[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS
BEGIN
  --statements
[EXCEPTION]
END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
  --statements
RETURN value;
[EXCEPTION]
END;
```

Variables in PL/SQL

TEKNOTURF

Declaring PL/SQL Variable

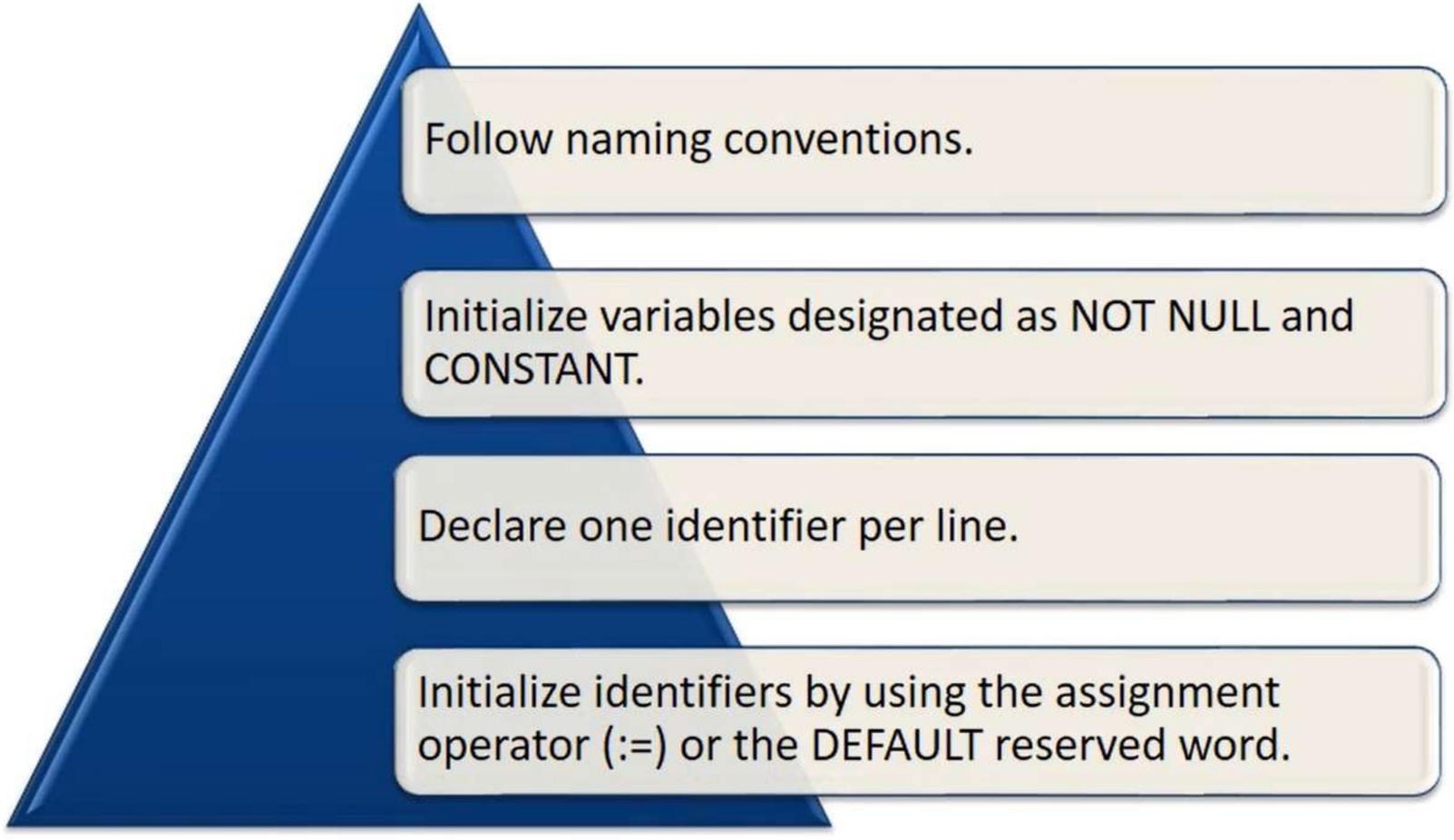
identifier [CONSTANT] *datatype* [NOT NULL]
[:= | DEFAULT *expr*];

Examples :

```
DECLARE
v_hiredate DATE;
v_deptno NUMBER(2) NOT NULL := 10;
v_location VARCHAR2(13) := 'Atlanta';
c_comm CONSTANT NUMBER := 1400;
```

Variables in PL/SQL

TEKNOTURF



Variables Initialization

TEKNOTURF

Syntax

```
identifier := expr;
```

Example

```
v_ename := 'Maduro'
v_mgr NUMBER(6) DEFAULT 100;
v_city VARCHAR2(30)NOT NULL :='Oxford'
```

Variables in PL/SQL

TEKNOTURF

Types of Variables

PL/SQL variables

- Scalar
- Composite
- Reference
- LOB (large objects)

Non PL/SQL variables

- Bind variable
- Host variable

Scalar Data Types

TEKNOTURF

Hold a single value

Have no internal components

Scalar Data Types

- CHAR [(*maximum_length*)]
- VARCHAR2 (*maximum_length*)
- LONG
- NUMBER [(*precision, scale*)]
- BOOLEAN
- DATE
- TIMESTAMP

Scalar Data Types

TEKNOTURF

Data Type	Description
CHAR[(*maximum_length*)]	Base type for fixed-length character data up to 32,767 bytes. If you do not specify a maximum length, the default length is set to 1
VARCHAR2(*maximum_length*)	Base type for variable-length character data up to 32,767 bytes. There is no default size for VARCHAR2 variables and constants.
LONG	Character data of variable length (a bigger version of the VARCHAR2 datatype)
NUMBER[(*precision, scale*)]	Number having precision *p* and scale *s*. The precision *p* can range from 1 to 38. The scale *s* can range from -84 to 127.

Scalar Data Types

TEKNOTURF

Data Type	Description
BOOLEAN	Base type that stores one of three possible values used for logical calculations: TRUE, FALSE, or NULL.
DATE	Base type for dates and times. DATE values include the time of day in seconds since midnight. The range for dates is between 4712 B.C. and 9999 A.D.
TIMESTAMP	The TIMESTAMP data type, which extends the DATE data type, stores the year, month, day, hour, minute and second

Scalar Data Types

TEKNOTURF

Examples :

```
v_sname         VARCHAR2(9);
v_salary        NUMBER(9,2) := 0;
v_dateofjoin    DATE := SYSDATE ;
c_servicetax    CONSTANT NUMBER(3,2) := 14;
v_status        BOOLEAN NOT NULL := TRUE;
```

Composite Data Types

TEKNOTURF

A composite type has internal components that can be manipulated individually.

Composite Data Types

- RECORD: contains multiple scalar values, similar to a table record
- TABLE: tabular structure with multiple columns and rows
- VARRAY: variable-sized array

Reference Data Types

TEKNOTURF

Directly reference a specific database field or record and assume the data type of the associated field or record

- %TYPE : same data type as a database field
- %ROWTYPE : same data type as a database record

The %TYPE Attribute

TEKNOTURF

Declare a variable according to

- A database column definition
- Another previously declared variable

Prefix %TYPE with

- The database table and column
- The previously declared variable name

Syntax

identifier **Table.column_name%TYPE;**

The %TYPE Attribute

TEKNOTURF

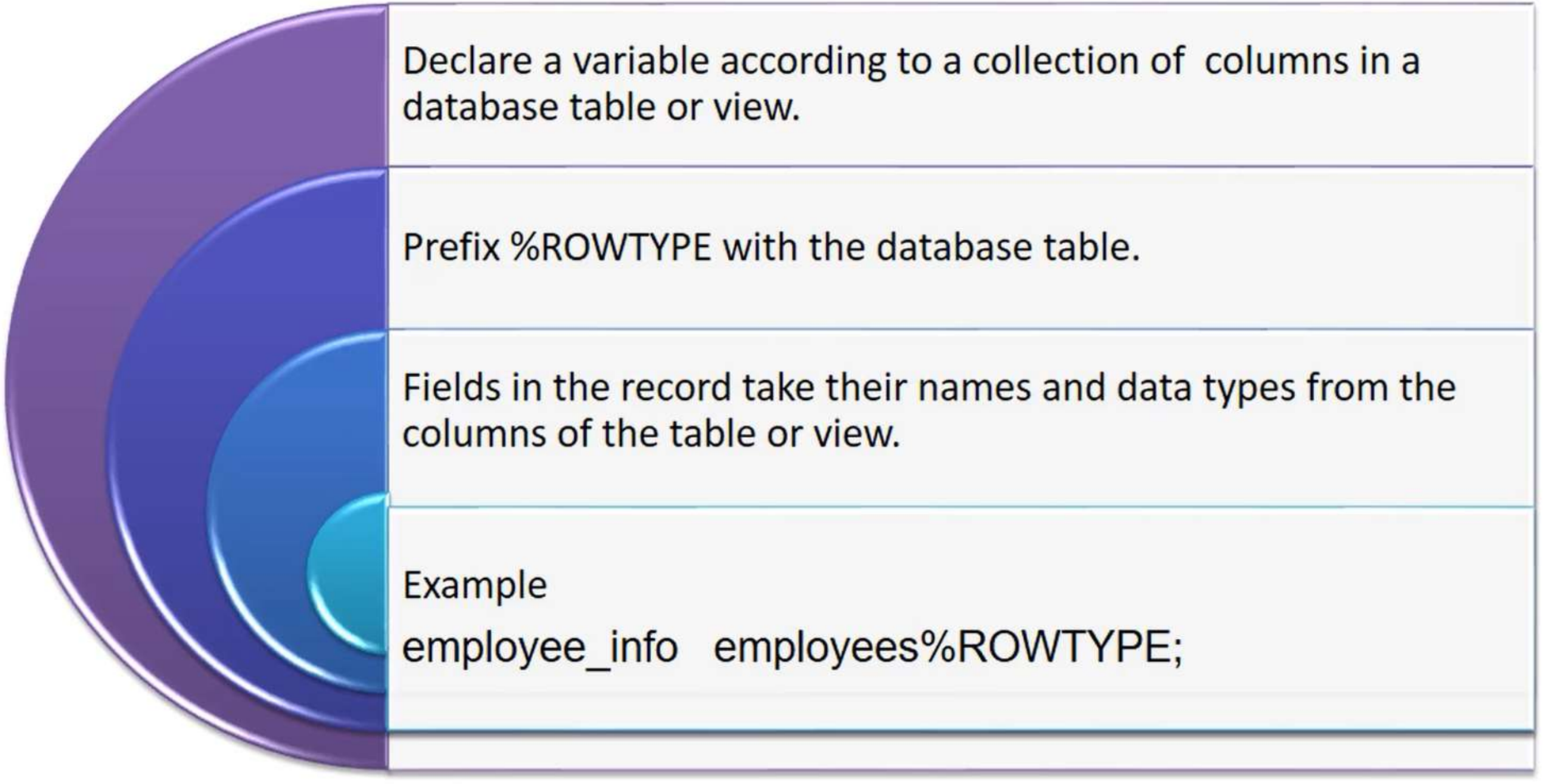
Examples

```
v_department_name
DEPARTMENT.dept_name%TYPE;
v_balanace_amt   NUMBER(7,2);
 v_withdraw_amt  balance_amt%TYPE;
```

```
DECLARE
        v_salary employee.salary%TYPE;
         v_salary employee.address%TYPE;
        v_emp_id number(6) := 101;
        v_emp_name employee.name%TYPE;
BEGIN
        SELECT * INTO v_emp_id, v_emp_name, v_salary ,var_address FROM          employee
                                        WHERE emp_id = v_emp_id;
        dbms_output.put_line('Employee Name:' || v_emp_name || 'Employee Salary:' || v_salary ||
                                        'Employee Address:' || v_address);
END;
```

The %ROWTYPE Attribute

TEKNOTURF



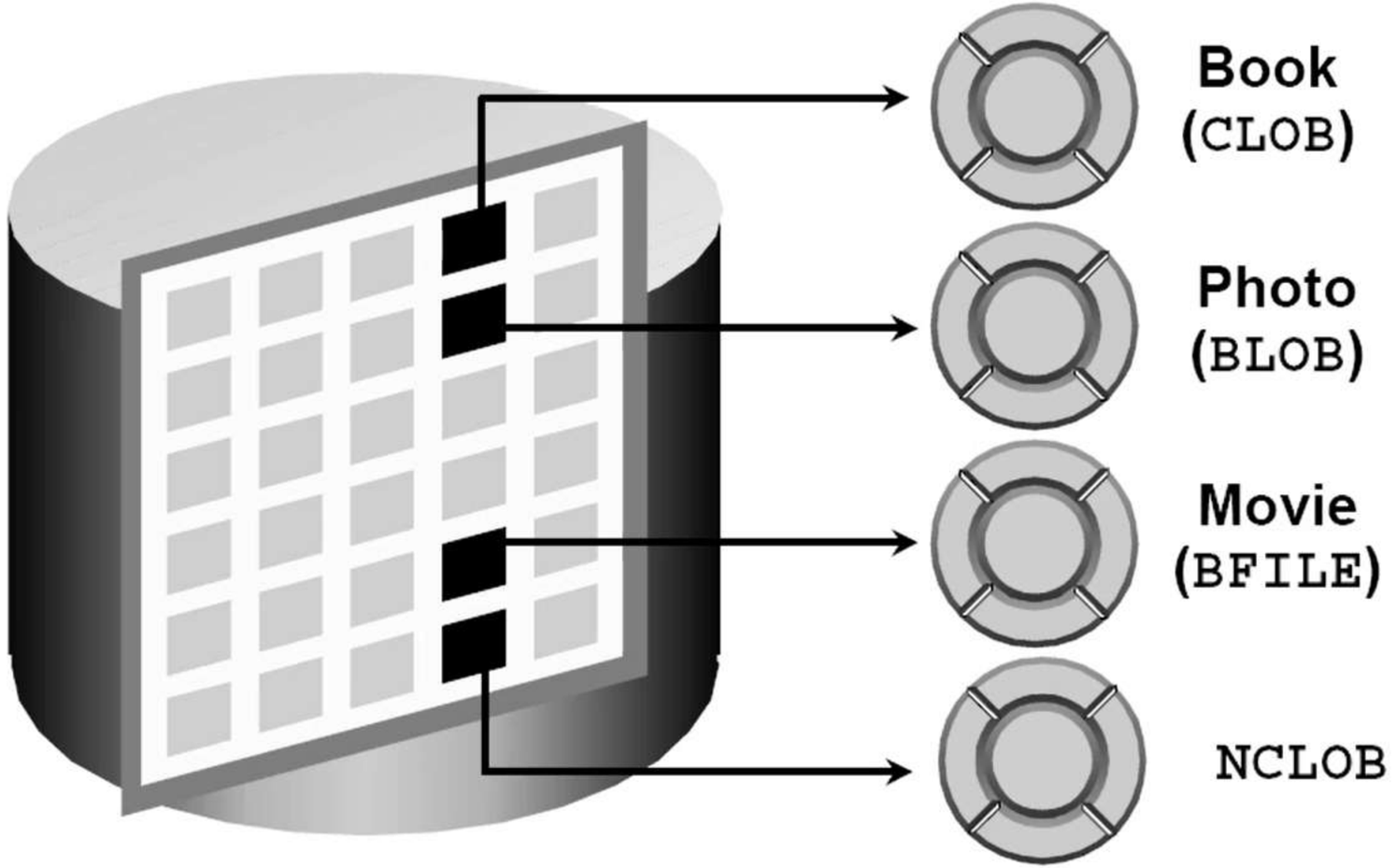
The %ROWTYPE Attribute

TEKNOTURF

```
DECLARE
        var_emp_id number(6) := 101;
        employee_info employee%ROWTYPE;
BEGIN
        SELECT * INTO employee_info FROM employee WHERE
                                        employee_id = var_emp_id;
        dbms_output.put_line(' EmployeeName :' ||
employee_info.emp_name);
```

LOB Data Types

TEKNOTURF



DBMS_OUTPUT.PUT_LINE

TEKNOTURF

An option for displaying information from a PL/SQL block is

DBMS_OUTPUT.PUT_LINE

DBMS_OUTPUT is an Oracle-supplied package

PUT_LINE is a procedure within that package

Must be enabled in iSQL*Plus with

SET SERVEROUTPUT ON

DBMS_OUTPUT.PUT_LINE

TEKNOTURF

```
SET SERVEROUTPUT ON
DEFINE  p_annual_sal = 60000
```

```
DECLARE
   v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
   v_sal := v_sal/12;
   DBMS_OUTPUT.PUT_LINE ('The monthly salary is '
                         || TO_CHAR(v_sal));
END;
/
```

SQL Functions in PL/SQL

TEKNOTURF

Available in procedural statements :

- Single-row functions

Not available in procedural statements

- DECODE
- Group functions

SQL Functions in PL/SQL

TEKNOTURF

Get the length of the String

```
v_desc_size INTEGER (5) ;

v_prod_description VARCHAR2(70) := 'You can use this product with your radios for
                                    higher frequency' ;

-- get the length of the string in prod_description
v_desc_size := LENGTH(v_prod_description) ;
```

Get the number of months an employee has worked:

```
v_tenure := MONTHS_BETWEEN (CURRENT_DATE, v_hiredate) ;
```

Data Type Conversion

TEKNOTURF

Converts data to compatible data types

Is of two types:

- Implicit conversion
- Explicit conversion

Functions:

- TO_CHAR
- TO_DATE
- TO_NUMBER
- TO_TIMESTAMP

Operators in PL/SQL

TEKNOTURF

- Logical
- Arithmetic
- Concatenation

same as in SQL

- Parenthesis to control order of operations
- Exponential Operator (**)

SQL Statements in PL/SQL

TEKNOTURF

- Retrieve a row from the database by using the *SELECT* command.
- Make changes to rows in the database by using DML commands.
- Control a transaction with the *COMMIT, ROLLBACK or SAVEPOINT* command.

SELECT Statements in PL/SQL

TEKNOTURF

Retrieve data with a SELECT statement.

Syntax

```
SELECT select_list
INTO {variable_name[, variable_name]... | record_name}
FROM table
[WHERE condition];
```

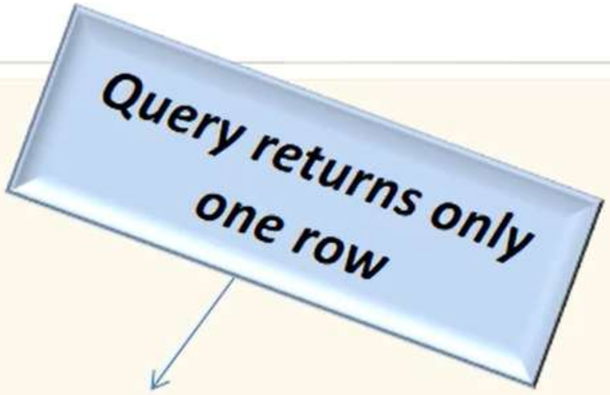
The INTO clause is required.

Queries must return one and only one row.

Retrieving Data in PL/SQL

TEKNOTURF

```
DECLARE
        v_emp_id number(6) := 1010;
        v_name   varchar2(20);
        v_salary NUMBER(7,2);
BEGIN
        SELECT name,salary INTO v_name, v_salary FROM employees
                WHERE emp_id=v_emp_id;
        dbms_output.put_line('Employee Name' || v_name);
        dbms_output.put_line('Employee Salary' || v_salary);
        END;
/
```



SELECT Statements in PL/SQL

TEKNOTURF

```
DEFINE emp_id = 1010
DECLARE
        employee_info employee%ROWTYPE;
BEGIN
      SELECT * INTO employee_info FROM employees WHERE emp_id =
                                                        &emp_id;
       insert into retired_emps (emp_id,name,salary,address,dor)
                values (employee_info.emp_id,employee_info.name,
                employee_info.salary,employee_info.address,SYSDATE);
       COMMIT;
END;
/
```

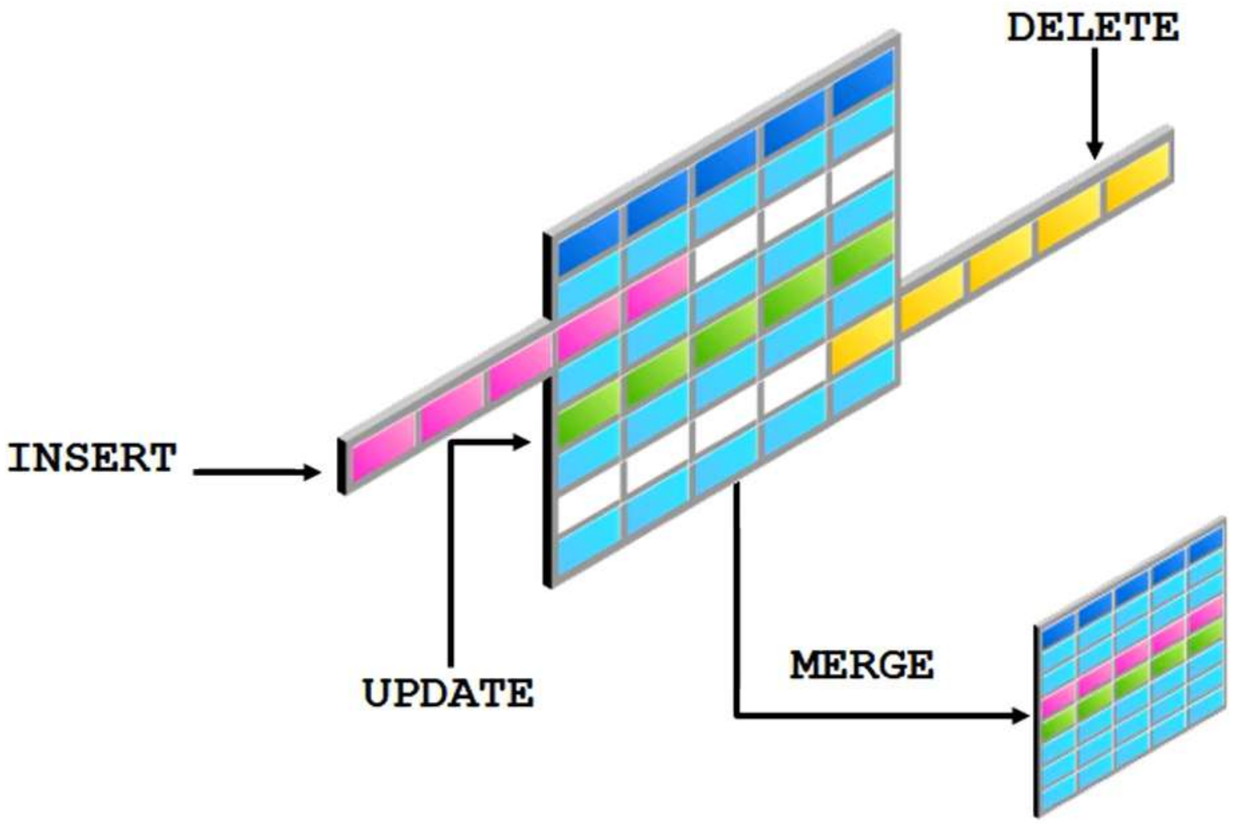
The record of emp_id 1010 is inserted into the retired_emps table

Using PL/SQL to Manipulate Data

TEKNOTURF

Make changes to database tables by using DML commands:

- INSERT
- UPDATE
- DELETE
- MERGE



Control Structure

TEKNOTURF

»**Conditional**
»**Iterative**

PL/SQL manipulates program data as a part of its business logic using conditional / iterative statements like

- IF-THEN-ELSE statement
- CASE statement
- FOR-LOOP statement
- WHILE-LOOP statement
- EXIT-WHEN statement and
- GOTO statement.

Control Structure

TEKNOTURF

Syntax for IF-ELSIF-ELSE

```
IF condition 1 THEN
        statement 1;
        statement 2;
ELSIF condtion2 THEN
        statement 3;
ELSE
        statement 4;
END IF;
```

Control Structure - Example

TEKNOTURF

```
DECLARE
    v_emp_id number(6) := 1010;
    v_salary NUMBER(7,2);
BEGIN
    select  salary  into  v_salary  where  emp_id = v_emp_id;
    IF v_salary < 25000 THEN
        UPDATE employees SET salary = 25000 WHERE emp_id = v_emp_id;
    ELSIF v_salary < 50000 THEN
        UPDATE employees SET salary = salary + salary * 5 / 100 WHERE emp_id =  v_emp_id;
    ELSIF v_salary < 100000 THEN
        UPDATE employees SET salary = salary + salary * 2 / 100 WHERE emp_id = v_emp_id;
    ELSE
        UPDATE employees SET salary = salary * 5  WHERE emp_id = v_emp_id;
    END IF;
END;
/
```

CASE Expression

TEKNOTURF

```
DECLARE
      v_pct      employees.commission_pct%TYPE;
      v_eval    VARCHAR2(10);
      v_emp_id    employees.emp_id%TYPE := 145;
BEGIN
            --get commission percentage
      SELECT  commission_pct  INTO  v_pct   FROM   employees   WHERE   emp_id = v_emp_id;
            --evaluate commission percentage
      CASE   v_pct
            WHEN 0 THEN
                  v_eval := 'N/A';
            WHEN 0.1 THEN
                  v_eval := 'Low';
            WHEN 0.4 THEN
                  v_eval := 'High';
            ELSE
                  v_eval := 'Fair';
      END CASE;
      DBMS_OUTPUT.PUT_LINE('Employee ' || v_emp_id || ' commission ' ||  TO_CHAR(v_pct)
                                                || '  which is  ' || v_eval);
END;
/
```

Iterative Control

TEKNOTURF

Used when one or more statements need to be executed repeatedly for specific number of times to achieve a business task.

Types of loops supported by PL / SQL

Simple loop - loop that keeps track of

FOR loop - repeats the iteration for a specified number of times

WHILE loop - repeats the iteration when the condition is true

Iterative Control – Simple Loop

TEKNOTURF

A simple loop executes a set of statements at least once before terminating the loop.

The base condition to exit a loop is specified using the EXIT keyword.

The loop terminates when the exit condition becomes true

Syntax

```
LOOP
        statements;
        EXIT;
        {or EXIT WHEN condition;}
END LOOP;
```

Iterative Control – Simple Loop

TEKNOTURF

```
DECLARE
   area NUMBER(7);
   pi CONSTANT NUMBER :=3.14;
   radius NUMBER :=1;
BEGIN
LOOP
   area := pi*power(radius,2);
   insert into AREAS values (radius, area);
   radius := radius+1;
      EXIT WHEN  area >100;
END LOOP;
END;
/
```

Iterative Control – WHILE Loop

TEKNOTURF

WHILE loop repeats execution of set of statements till the condition remains true

The loop terminates when the condition becomes false.

```
DECLARE
        area NUMBER(7) :=0;
        pi CONSTANT NUMBER :=3.14;
        radius NUMBER :=1;
BEGIN
        WHILE   area < 100
        LOOP
                area := pi*power(radius,2);
                insert into AREAS values (radius, area);
                radius := radius+1;
        END LOOP;
END;
/
```

Iterative Control – FOR Loop

TEKNOTURF

FOR LOOP executes a set of statements for a specified number of times.

Iteration starts from the initial value and continues, till it reaches the final value.

The loop counter value is incremented by 1.

```
DECLARE
        area NUMBER(7);
        pi CONSTANT NUMBER :=3.14;
BEGIN
        FOR radius in 1..10 LOOP
                area := pi*power(radius,2);
                insert into AREAS values (radius, area);
        END LOOP;
END;
/
```

In case of reverse counting, FOR radius in **REVERSE** 1..10 LOOP

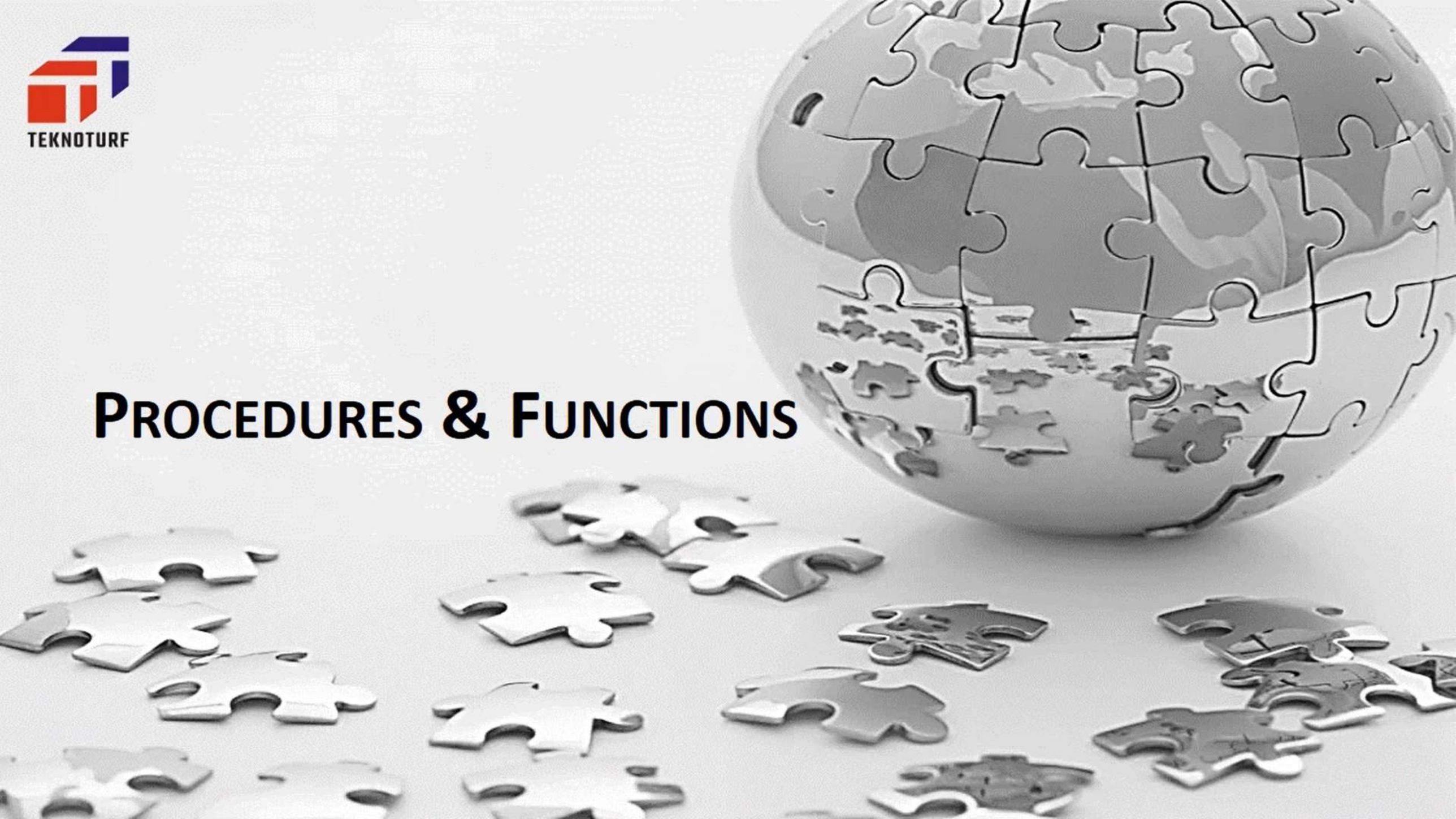
Summary

TEKNOTURF

- PL/SQL Basics
- PL/SQL Block Structure
- Handling Variables in PL/SQL
- Scope and Visibility of Variables
- SQL in PL/SQL
- Programmatic Constructs

TEKNOTURF

PROCEDURES & FUNCTIONS



In this Module you will learn

TEKNOTURF

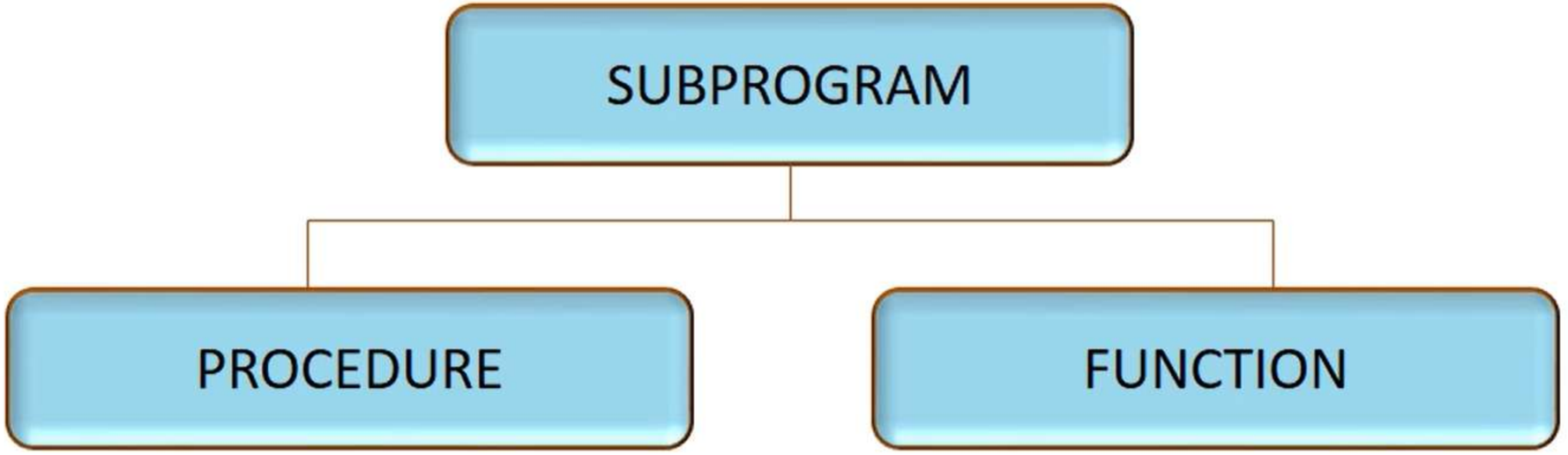
- Subprograms in PL/SQL
- Anonymous Blocks
- Procedures
- Functions

Subprogram

TEKNOTURF

Is a named PL/SQL block that can accept parameters and be invoked from a calling environment

Is based on standard PL/SQL block structure



Procedure

TEKNOTURF

Is a named PL/SQL block which performs one or more specific tasks.

Contains a header and a body.

Can be stored in the database, as a schema object, for repeated execution.

Parameters can be passed to procedures in three ways.

- IN-parameters
- OUT-parameters
- IN OUT-parameters

A procedure may or may not return any value.

Procedure

TEKNOTURF

Syntax

```
CREATE [OR REPLACE] PROCEDURE procedure_name
[(parameter1 [mode1] datatype1,
parameter2 [mode2] datatype2,. . .)]
IS|AS
PL/SQL Block;
```

```
CREATE OR REPLACE PROCEDURE employer_details IS
        CURSOR emp_cur IS  SELECT empid,empname, salary FROM employee;
        emp_rec emp_cur%rowtype;
 BEGIN
        FOR emp_rec in emp_cur  LOOP
            dbms_output.put_line(emp_rec.empid || ' ' ||emp_rec.empname);
        END LOOP;
END;
/
```

Procedure

TEKNOTURF

Procedure is executed as

EXECUTE [or EXEC] *procedure_name*

EXECUTE employer_details;

Can be executed within another procedure using the procedure name.

Procedure with Parameters

TEKNOTURF

```
CREATE OR REPLACE PROCEDURE raise_salary (emp_id int, sal_increase int) IS
        current_sal int;
        salary_missing EXCEPTION;
BEGIN
        SELECT salary INTO current_sal FROM  employee  WHERE empid = emp_id;
        IF current_sal IS NULL THEN
                RAISE salary_missing;
        ELSE
                UPDATE employee SET salary=sal_increase WHERE  empno = emp_id;
        END IF;
 EXCEPTION
        WHEN NO_DATA_FOUND THEN
                INSERT INTO emp_audit VALUES (emp_id, 'No such number');
        WHEN salary_missing THEN
                INSERT INTO emp_audit VALUES (emp_id, 'Salary is null');
END raise_salary;
/
```

To Execute

```
EXECUTE raise_salary(1,20000)
```

Removing a Procedure

TEKNOTURF

- When a stored procedure is no longer required, DROP PROCEDURE command can be used to remove the stored procedure.
- Syntax

```
DROP PROCEDURE  procedure_name;
```

- Example

```
DROP PROCEDURE  raise_salary;
```

Functions

TEKNOTURF

A function is a named PL/SQL Block which is similar to a procedure.

A function must always return a value.

Procedure and Function have the same structure, except that

- A function heading must include a RETURN clause that specifies the data type of the return value whereas a procedure heading cannot have a RETURN clause.
- A function must have at least one RETURN statement in its executable part whereas in a procedure, the RETURN statement is optional.

Functions

TEKNOTURF

Syntax

```
CREATE [OR REPLACE] FUNCTION function_name
 [(argument1 [mode1] datatype1,
  argument2 [mode2] datatype2,
  . . .)]
RETURN datatype
IS|AS
function_body;
```

Size of the data type should not be included in the return data type. Returned value should be of the same data type defined in the header section.

Creating a Function

TEKNOTURF

```
CREATE OR REPLACE FUNCTION get_salary(p_empid
employees.emp_id%TYPE)
RETURN NUMBER
IS
      v_salary employees.salary%TYPE :=0;
BEGIN
SELECT salary INTO v_salary FROM employees WHERE emp_id =
p_empid;
RETURN v_salary;
END get_salary;
/
```

Invoking a Function

TEKNOTURF

```
CREATE OR REPLACE PROCEDURE get_designation
IS
     CURSOR c_empid is SELECT emp_id FROM
                                        employees;
     v_designation  varchar2(30);
     v_salary employees.salary%TYPE;
BEGIN
     FOR v_empid in c_empid
     LOOP
          v_salary := get_salary(v_empid.emp_id);
          IF(v_salary > 50000) THEN
              v_designation := 'Project Lead';
          ELSIF(v_salary > 30000) THEN
              v_designation := 'Team Lead';
          ELSE
              v_designation :='Developer';
     END IF;
     DBMS_OUTPUT.PUT_LINE(
        v_empid.emp_id||' has
              designation '||
                   v_designation);
     END LOOP;
END get_designation;
/
```

Can also be invoked using select statement as

SELECT get_salary(101) from dual;

Functions – Passing Parameter

TEKNOTURF

```
CREATE  OR REPLACE FUNCTION   check_salary(p_empid  employees.emp_id%TYPE)
RETURN Boolean
IS
            v_dept_id employees.department_id%TYPE;
            v_sal     employees.salary%TYPE;
            v_avg_sal employees.salary%TYPE;
BEGIN
   SELECT salary,department_id INTO v_sal,v_dept_id FROM employees  WHERE emp_id= p_empid;
   SELECT avg(salary) INTO v_avg_sal FROM employees WHERE  department_id = v_dept_id;
    IF v_sal > v_avg_sal THEN
            RETURN TRUE;
    ELSE
            RETURN FALSE;
    END IF;
EXCEPTION
     WHEN NO_DATA_FOUND THEN
            RETURN NULL;
END;
```

Invoking a Function

TEKNOTURF

Invoking a Function with parameter

```
DECLARE
        v_empno employees.employee_id%TYPE;
BEGIN
         v_empno=205;
         IF (check_salary(v_empno) IS NULL) THEN
                DBMS_OUTPUT.PUT_LINE('The function returned
NULL');
          ELSIF (check_salary(v_empno)) THEN
              DBMS_OUTPUT.PUT_LINE('Salary > average');
           ELSE
                DBMS_OUTPUT.PUT_LINE('Salary < average');
           END IF;
END
/
```

Removing a Function

TEKNOTURF

- When a stored function is no longer required, DROP FUNCTION command can be used to remove the stored function.
- Syntax

```
DROP FUNCTION  function_name;
```

- Example

```
DROP FUNCTION  check_salary;
```

Parameters in Procedures

TEKNOTURF

Parameter Type	Description
IN (Default)	Used to send values to stored procedures
OUT	Used to get values from stored procedures (Similar to a return type in functions)
IN OUT	Used to pass and retrieve values from stored procedures

Example – IN, OUT Parameter

TEKNOTURF

```
CREATE OR REPLACE PROCEDURE  get_salary(p_empid
employees.emp_id%TYPE,
                                                v_salary OUT
NUMBER)
IS
BEGIN
SELECT salary INTO v_salary FROM employees WHERE emp_id =
p_empid;
END get_salary;
/
```

Retrieving the out parameter

TEKNOTURF

Retrieving the out parameter in a program

```
DECLARE
        v_salary NUMBER;
        CURSOR id_cur IS SELECT emp_id FROM employees;
BEGIN
        FOR empid in id_cur LOOP
            get_salary(empid.emp_id, v_salary);
            dbms_output.put_line('The employee with id ' || empid.emp_id || ' has
                                                        salary ' || v_salary);
        END LOOP;
END;
 /
```

The out parameter value is captured in v_salary

Retrieving the out parameter

TEKNOTURF

Retrieving the out parameter using host variable

```
VARIABLE g_salary  NUMBER;
EXECUTE
get_salary(101,:g_salary);
PRINT g_salary;
```

The out parameter value is captured in the host variable g_salary and printed to the console

Example – IN OUT Parameter

TEKNOTURF

```
CREATE OR REPLACE PROCEDURE get_emp_salary_increase (p_empid IN
        employees.emp_id%type, salary_inc IN OUT employees.salary%type)
IS
        tmp_sal number;
BEGIN
        SELECT salary INTO tmp_sal FROM employees  WHERE emp_id = p_empid;
        IF tmp_sal < 10000 THEN
                salary_inc := tmp_sal * 0.01;
        ELSIF  tmp_sal between 10000 and 20000 THEN
                salary_inc := tmp_sal * 0.012;
        ELSIF tmp_sal between 20000 and 30000 THEN
                salary_inc := tmp_sal * 0.013;
        ELSIF tmp_sal > 30000 THEN
                salary_inc := tmp_sal * 1.4;
        END IF;
END;
```

Example – IN OUT Parameter

TEKNOTURF

```
DECLARE
        CURSOR updated_sal IS SELECT emp_id,salary FROM employees;
        pre_sal  NUMBER;
 BEGIN
        FOR emp_rec IN updated_sal
        LOOP
                pre_sal := emp_rec.salary;
                getemp_salary_increase(emp_rec.emp_id, emp_rec.salary);
                dbms_output.put_line('The salary of ' || emp_rec.emp_id ||
                        ‘  increased by '|| pre_sal || ' to '||emp_rec.salary);
        END LOOP;
END;
/
```

Summary

TEKNOTURF

- Subprograms in PL/SQL
- Anonymous Blocks
- Procedures
- Functions

TEKNOTURF

Press Esc to exit full screen

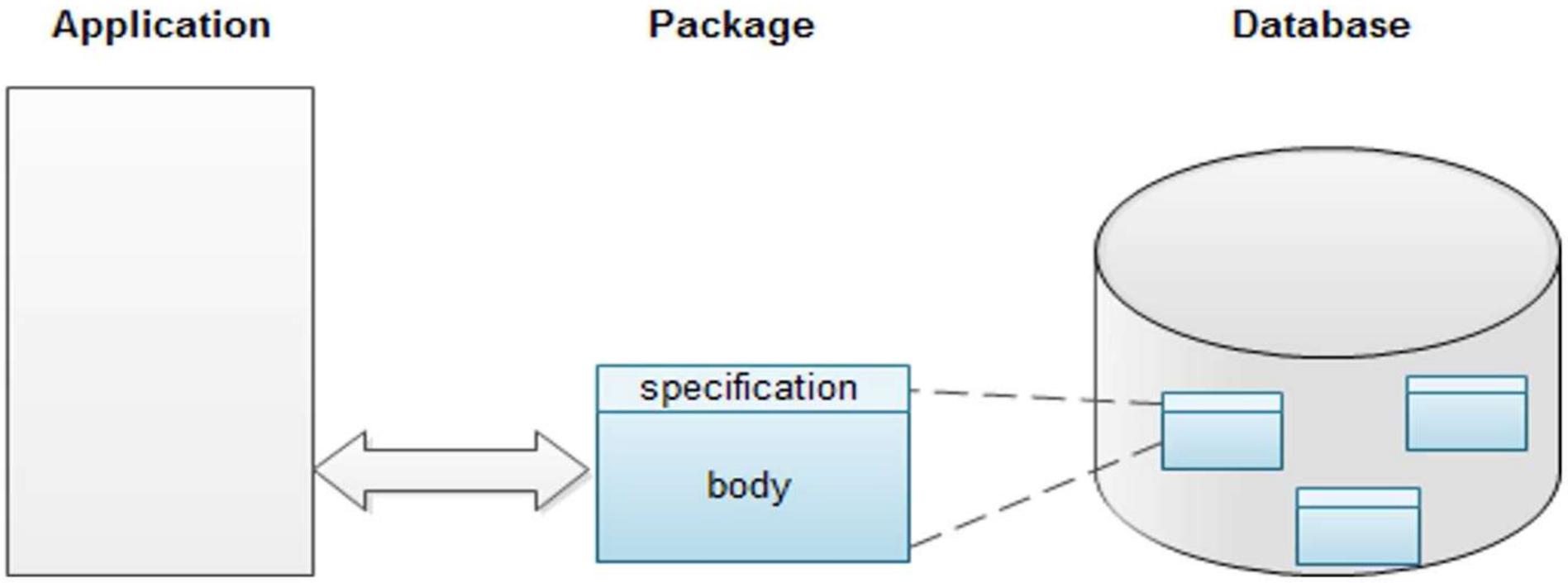
PACKAGES & TRIGGERS

In this Module you will learn

TEKNOTURF

- Packages
- Creating & Deleting Packages
- Database Triggers
- Types of Triggers

Packages



Packages

TEKNOTURF

- A schema object that groups logically related
  - PL/SQL types
  - variables
  - constants
  - subprograms
  - cursors
  - exceptions.
- A package is compiled and stored in the database, where many applications can share its contents.

Parts of a package

TEKNOTURF

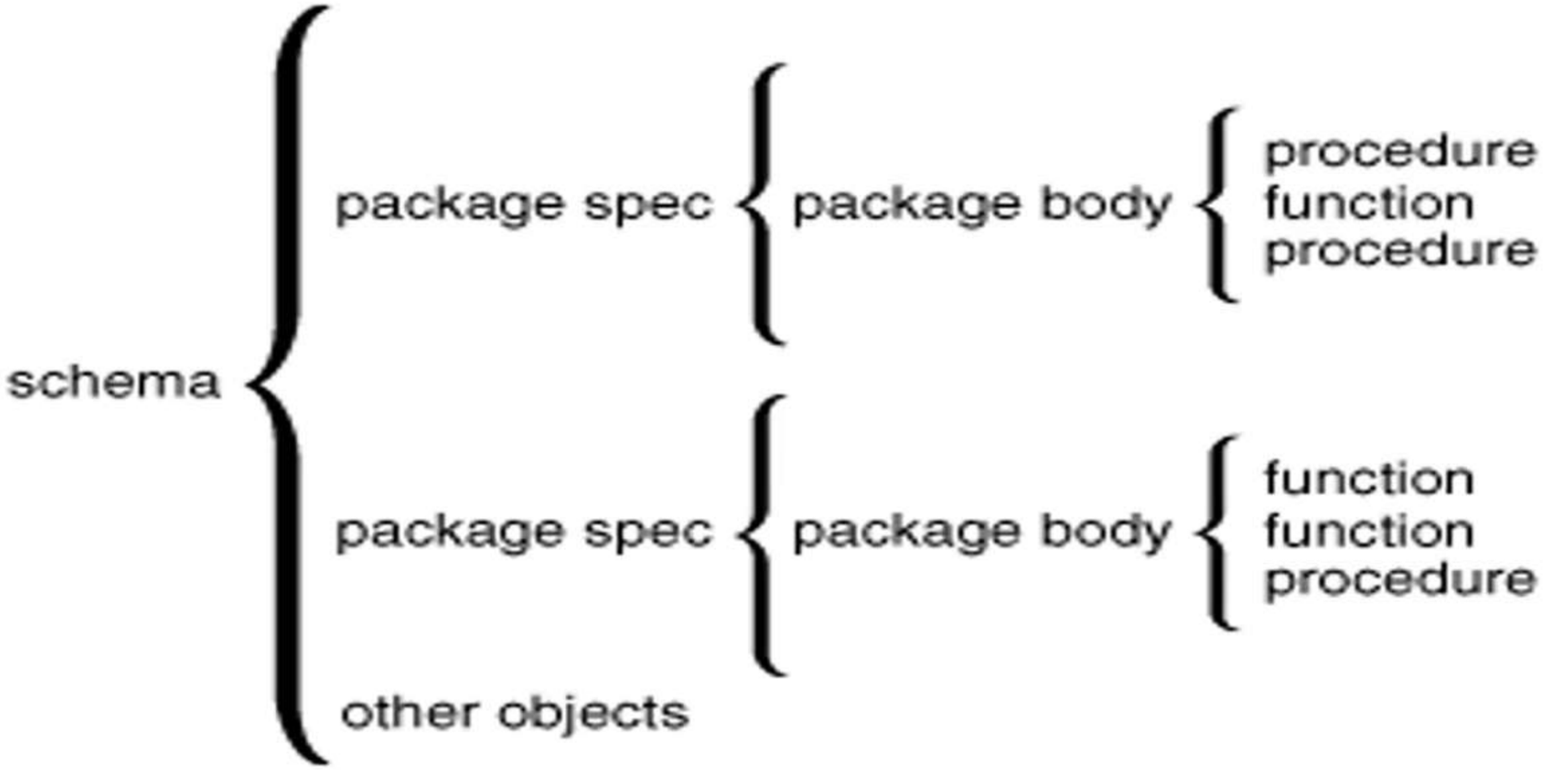
Package Specification

- Declares package constructs, which include names and parameters, publicly available procedures and functions.

Package Body

- Can declare additional private package constructs which may not be available publicly.
- Defines all the package constructs be it public or private.

Packages


schema
package spec
package body
procedure
function
procedure
package spec
package body
function
function
procedure
other objects

Package Specification

TEKNOTURF

```
CREATE [OR REPLACE] PACKAGE package_name
[ AUTHID { CURRENT_USER | DEFINER } ]
{ IS | AS }
   [definitions of public TYPES
  ,declarations of public variables, types, and objects
  ,declarations of exceptions
  ,pragmas
  ,declarations of cursors, procedures, and functions
  ,headers of procedures and functions]
END [package_name];
```

Package Body

TEKNOTURF

```
CREATE [OR REPLACE] PACKAGE BODY package_name
  { IS | AS }
  [definitions of private TYPEs
  ,declarations of private variables, types, and objects
  ,full definitions of cursors
  ,full definitions of procedures and functions]
[BEGIN
  sequence_of_statements
  . . .
[EXCEPTION
  exception_handlers ] ]
END [package_name];
```

Creating a package

TEKNOTURF

Package Specification:

CREATE PACKAGE **emp_bonus** AS PROCEDURE

calc_bonus (date_hired employees.hire_date%TYPE);

END emp_bonus;

/

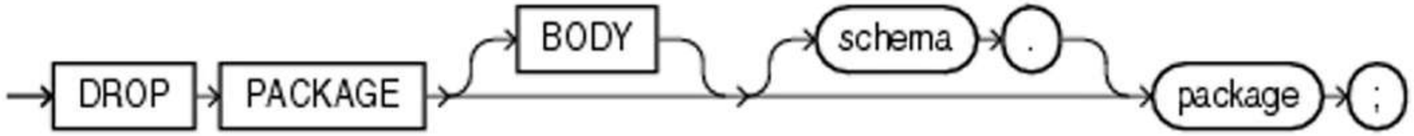
Creating a package

TEKNOTURF

Package Body:

```
CREATE PACKAGE BODY emp_bonus AS PROCEDURE
calc_bonus (date_hired employees.hire_date%TYPE) IS
BEGIN
DBMS_OUTPUT.PUT_LINE ('Employees hired on ' ||
date_hired || ' get bonus.');
END;
END emp_bonus;
/
execute emp_bonus.calc_bonus('07-APR-2008');
```

Dropping a package



- Using the DROP PACKAGE statement, the entire package/body of the package can be dropped.
- Example

```
DROP PACKAGE emp_bonus;
```

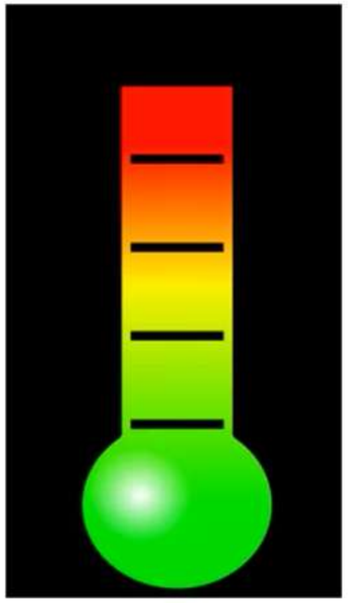
Uses of packages

TEKNOTURF

- Highly Modular
  - Easier application design
- Information hiding
- Simplified Security
  - GRANT EXECUTE privilege on the entire package
- Better performance
  - First time a package is invoked, the Database loads the whole package onto memory

Triggers

TEKNOTURF



Database Triggers

TEKNOTURF

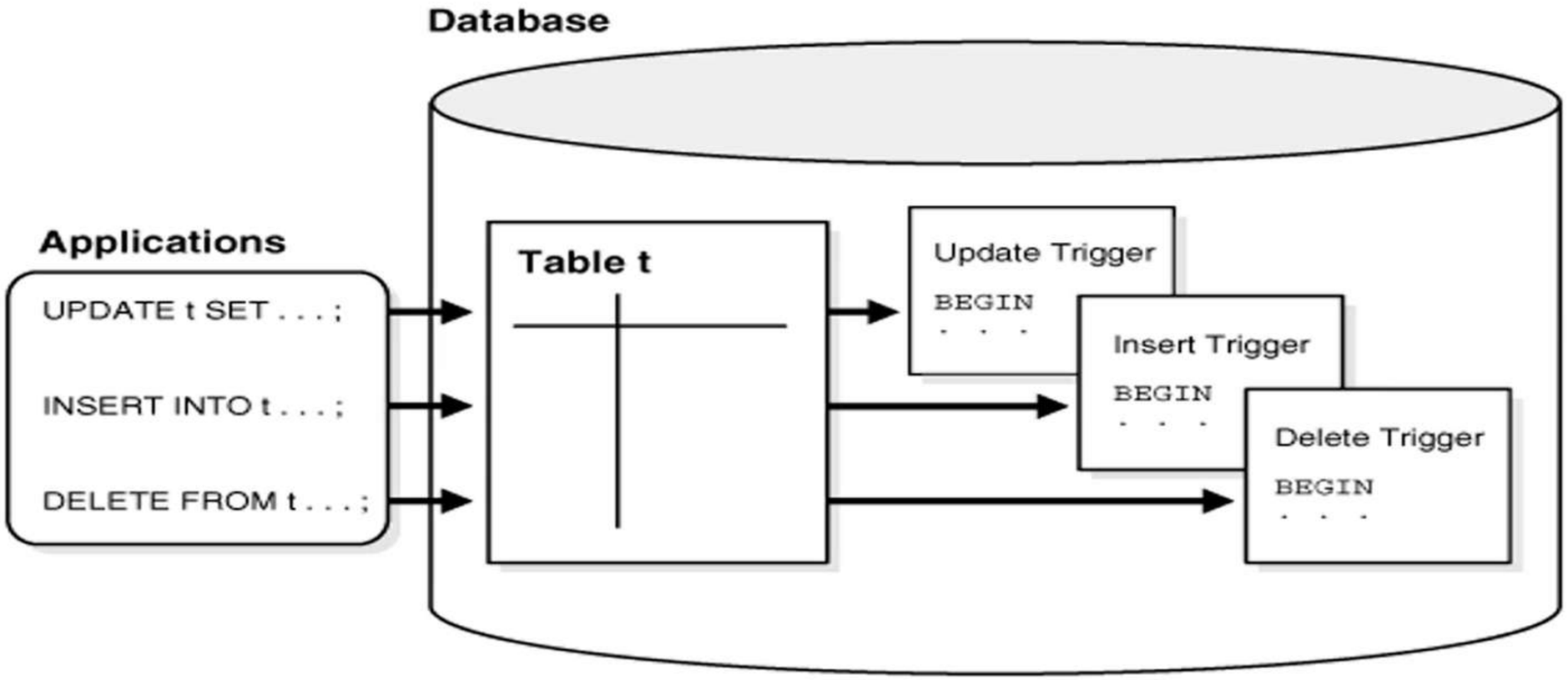
- A database trigger is a stored procedure that is fired automatically when an insert/delete/update statement is issued against the table.

- Syntax of a trigger:

```
create or replace trigger <trigger name>
[before/after]  [insert/update/delete] on <table-name>
[referencing {old [as] old/new as new}]
[for each statement/for each row] [when <condition>]
pl/sql block
```

Triggers

TEKNOTURF



Types of Triggers

TEKNOTURF

Before/After option

For each row/For each statement

- Before update row/statement
- Before delete row/statement
- Before insert row/statement
- After update row/statement
- After delete row/statement
- After insert row/statement

Example of Triggers

TEKNOTURF

```
create or replace trigger t_insert after insert on customer for
each row
begin
 dbms_output.put_line('ROW IS INSERTED');
end;

create or replace trigger t_trig1 before update on customer
for each row
begin
  if new.custno<old.custno then
    raise_application_error(-20001,'CANNOT BE UPDATED');
 end if;
end;
```

Example of Triggers

TEKNOTURF

```
create or replace trigger t_delete after delete on
customer for each  row
begin
  dbms_output.put_line('Deleted');
end;
```

Enabling/Disabling & Dropping Triggers

TEKNOTURF

Enabling and disabling triggers

Alter trigger <trigger-name> disable;

Alter table <table-name> disable all triggers;

Alter table <table-name> enable all triggers;

Dropping triggers

Drop trigger <trigger-name>;

Summary

TEKNOTURF

- Packages
- Creating & Deleting Packages
- Database Triggers
- Types of Triggers

TEKNOTURF

CURSORS & EXCEPTION HANDLING

In this Module you will learn

TEKNOTURF

- Introduction to Cursors
- Implicit Cursors
- Explicit Cursors
- Exception Handling
- Predefined Exception
- User Defined Exceptions
- Other Exception Handler

CURSORS

TEKNOTURF

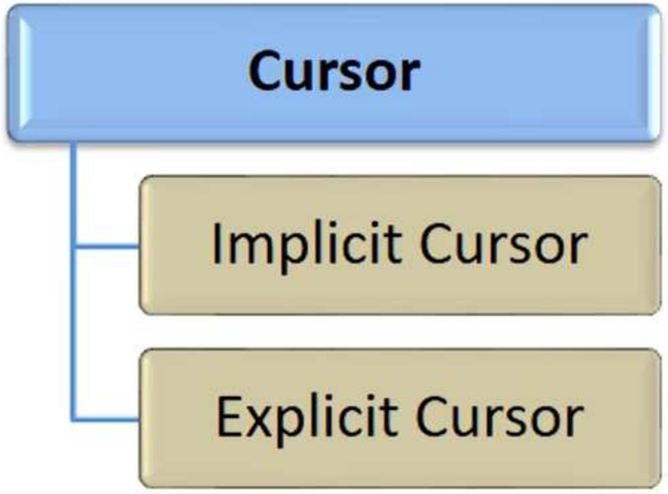
A cursor is a pointer to the private memory area temporarily allocated by the Oracle Server when an SQL statement is executed.

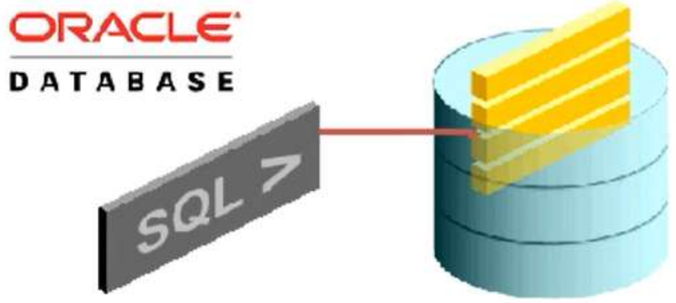
A Cursor holds multiple rows but can process a single row at a time.

The total number of rows a Cursor holds is called an active set.

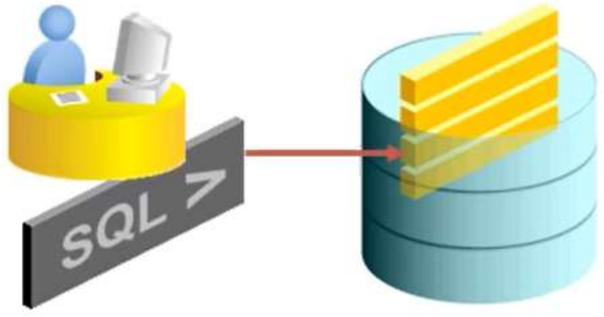
Cursor Types

TEKNOTURF





Implicit Cursor



Explicit Cursor

Implicit Cursor

TEKNOTURF

- Created and managed internally by the Oracle Server to process SQL statements
- Implicit cursors are called SQL cursors.
- Programmers cannot control the implicit cursors and the information in it.
- Information about the status and the number of rows that are affected by the query can be tracked through its attributes.

Cursor Attributes

TEKNOTURF

Attributes	Return Value
%FOUND	The return value is TRUE, if the DML statements like INSERT, DELETE and UPDATE affect at least one row and if SELECT ….INTO statement returns at least one row. The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE do not affect row and if SELECT….INTO statement do not return a row.
%NOTFOUND	The return value is FALSE, if DML statements like INSERT, DELETE and UPDATE affect at least one row and if SELECT ….INTO statement return at least one row. The return value is TRUE, if a DML statement like INSERT, DELETE and UPDATE do not affect even one row and if SELECT ….INTO statement does not return a row.
%ROWCOUNT	Return the number of rows affected by the DML operations INSERT, DELETE, UPDATE, SELECT.

Implicit Cursor

TEKNOTURF

```
DECLARE
      v_rows number(5);
BEGIN
      UPDATE employees SET salary = salary + 1000;
      IF SQL%NOTFOUND THEN
            dbms_output.put_line('None of the salaries where updated');
     ELSIF SQL%FOUND THEN
            v_rows := SQL%ROWCOUNT;
            dbms_output.put_line('Salaries for ' || v_rows || 'employees are
                  updated');
    END IF;
END;
/
```

In this example, the salaries of all the employees in the ‘employee’ table are updated and the message is printed accordingly.

Explicit Cursor

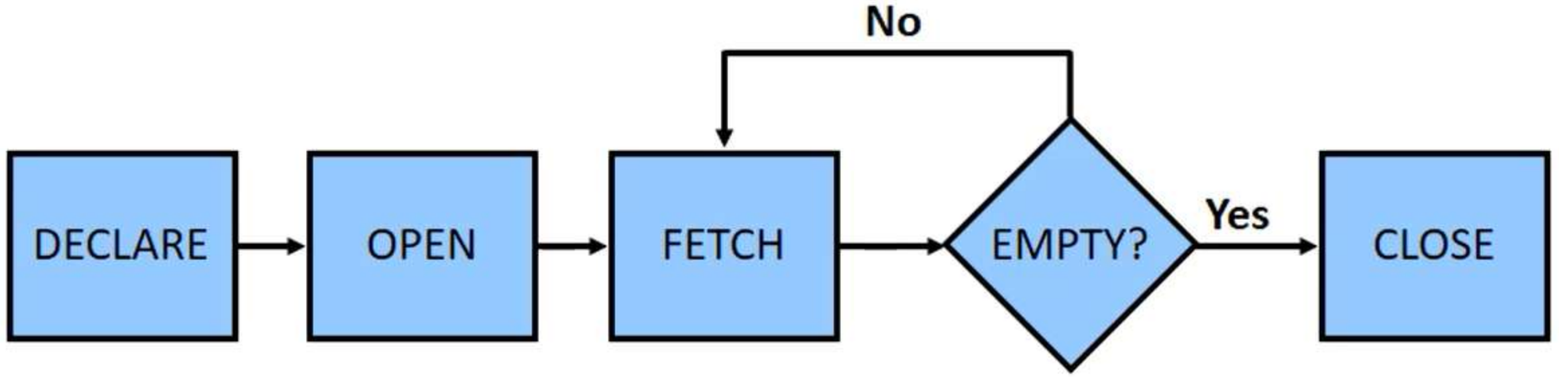
TEKNOTURF

An explicit cursor is defined on a SELECT statement when it returns more than one row.

A Cursor should be declared in the declaration section of a PL/SQL block, by giving a cursor name and an associated query to it.

Cursor can process only one row at a time and that is called as a current row.

Controlling Explicit Cursor

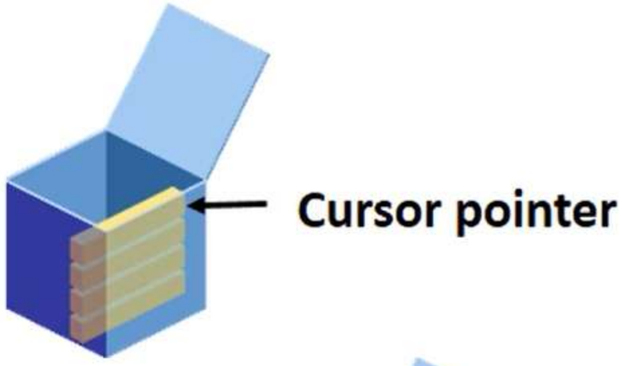


- Create a named SQL area.
- Identify the active set.
- Load the current row into variables.
- Test for existing rows.
- Return to FETCH if rows are found.
- Release the active set.

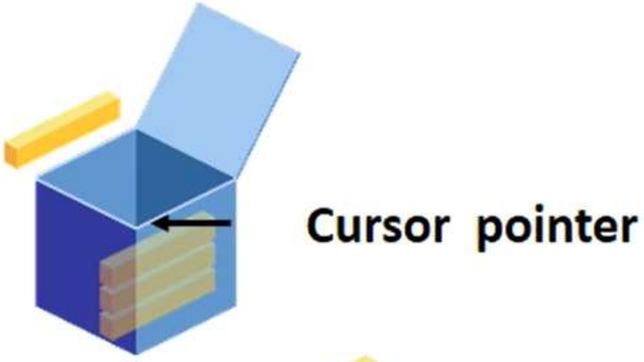
Controlling Explicit Cursor

TEKNOTURF

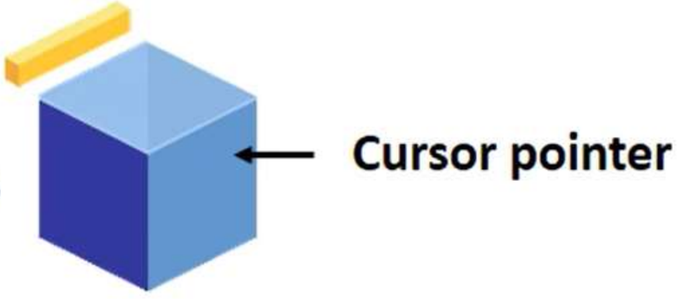
1. Open the cursor.



2. Fetch a row.



3. Close the cursor.



Explicit Cursor

TEKNOTURF

```
DECLARE
  CURSOR c_emp_cursor IS
      SELECT emp_id, last_name FROM employees
                  WHERE department_id =30;
  v_empid employees.emp_id%TYPE;
  v_lname employees.last_name%TYPE;
BEGIN
  OPEN c_emp_cursor;
  LOOP
    FETCH c_emp_cursor INTO v_empid, v_lname;
    EXIT WHEN c_emp_cursor%NOTFOUND;
    DBMS_OUTPUT.PUT_LINE( v_empid ||' '||v_lname);
  END LOOP;
  CLOSE  c_emp_cursor;
END;
/
```

Declare Cursor

Open Cursor

Fetch Cursor

Check for Empty

Close Cursor

Cursor FOR Loop

TEKNOTURF

Syntax

- Implicit open, fetch, exit, and close occur.
- The record is implicitly declared.
- The cursor FOR loop is a shortcut to process explicit cursors.

```
FOR record_name IN cursor_name LOOP
  statement1;
  statement2;
  . . .
END LOOP;
```

Cursor FOR Loop

TEKNOTURF

```
DECLARE
        emp_rec   employees%ROWTYPE ;
        CURSOR emp_cur IS SELECT * FROM employee WHERE salary > 10000;
BEGIN
        FOR emp_rec   IN   emp_cur
        LOOP
            dbms_output.put_line ('Name:'||emp_rec.name);
            dbms_output.put_line ('Address:'||emp_rec.address);
        END LOOP;
END;
/
```

Explicit Cursor Attributes

TEKNOTURF

%ISOPEN Attribute

Rows can be fetched from a cursor only when the cursor is open.

%ISOPEN cursor attribute can be used to test whether the cursor is open.

Example for %ISOPEN Attribute

```
IF NOT emp_cursor%ISOPEN THEN
OPEN emp_cursor;
END IF;
LOOP
  FETCH emp_cursor...
```

Explicit Cursor Attributes

TEKNOTURF

%ROWCOUNT and %NOTFOUND Attributes

```
DECLARE
 CURSOR c_emp_cursor IS SELECT employee_id,
  last_name FROM employees;
 v_emp_record  c_emp_cursor%ROWTYPE;
BEGIN
 OPEN c_emp_cursor;
 LOOP
  FETCH c_emp_cursor INTO v_emp_record;
  EXIT WHEN c_emp_cursor%ROWCOUNT > 10 OR
            c_emp_cursor%NOTFOUND;
  DBMS_OUTPUT.PUT_LINE( v_emp_record.employee_id
          ||' '||v_emp_record.last_name);
 END LOOP;
 CLOSE c_emp_cursor;
END ;
/
```

Cursor with Parameters

TEKNOTURF

- Parameter can be passed to a Cursor and it can be used in the query
- Values can be passed only through parameters to a Cursor
- The datatype of the parameter alone needs to be mentioned in the parameter and not its length
- Optionally, default value can be passed to the parameter, in which case this value will take effect if no value is passed
- Cursor becomes more reusable with cursor parameters
- The scope of the cursor parameter is local to the cursor

Cursor with Parameters

TEKNOTURF

```
DECLARE
      CURSOR cur_dept is select deptno,deptname from dept order by deptno;
      CURSOR  cur_emp (par_dept varchar2) is select empid,empname,empsal from emp
                                          where deptno=par_dept order by empid;
       v_totSal number;
BEGIN
      FOR v_dept in cur_dept   LOOP
            DBMS_OUTPUT.PUT_LINE ('Department ID: ' || v_dept.deptno ||
                                  ' Department Name  '||v_dept.deptname);
             v_totSal=0;
             FOR v_emp in   cur_emp(v_dept.deptno)  LOOP
                  DBMS_OUTPUT.PUT_LINE(v_emp.empid || ' ' || v_emp.empname || ' ' ||
                                                                  v_emp.empsal);
                   v_totSal :=v_totSal+v_emp.empsal;
            END LOOP;
            DBMS_OUTPUT.PUT_LINE ('Total Salary for Department : '||v_totSal);
            END LOOP;
END;
/
```

Exception Handling

TEKNOTURF

```
DECLARE
  v_name varchar2(30);
BEGIN
  select studname into v_name from student;
  DBMS_OUTPUT.PUT_LINE(v_name);
END;
/
```

Code does not work as expected. SELECT statement retrieves multiple rows.

```
ERROR at line 1:
ORA-01422: exact fetch returns more than requested
number of rows
ORA-06512: at line 4
```

Such errors that occur at run time are called *exceptions*.

Exception Handling

TEKNOTURF

When an exception occurs, the PL/SQL block is terminated.

An exception is a PL/SQL error that is raised during program execution.

An exception can be raised

- Implicitly by the Oracle Server
- Explicitly by the program

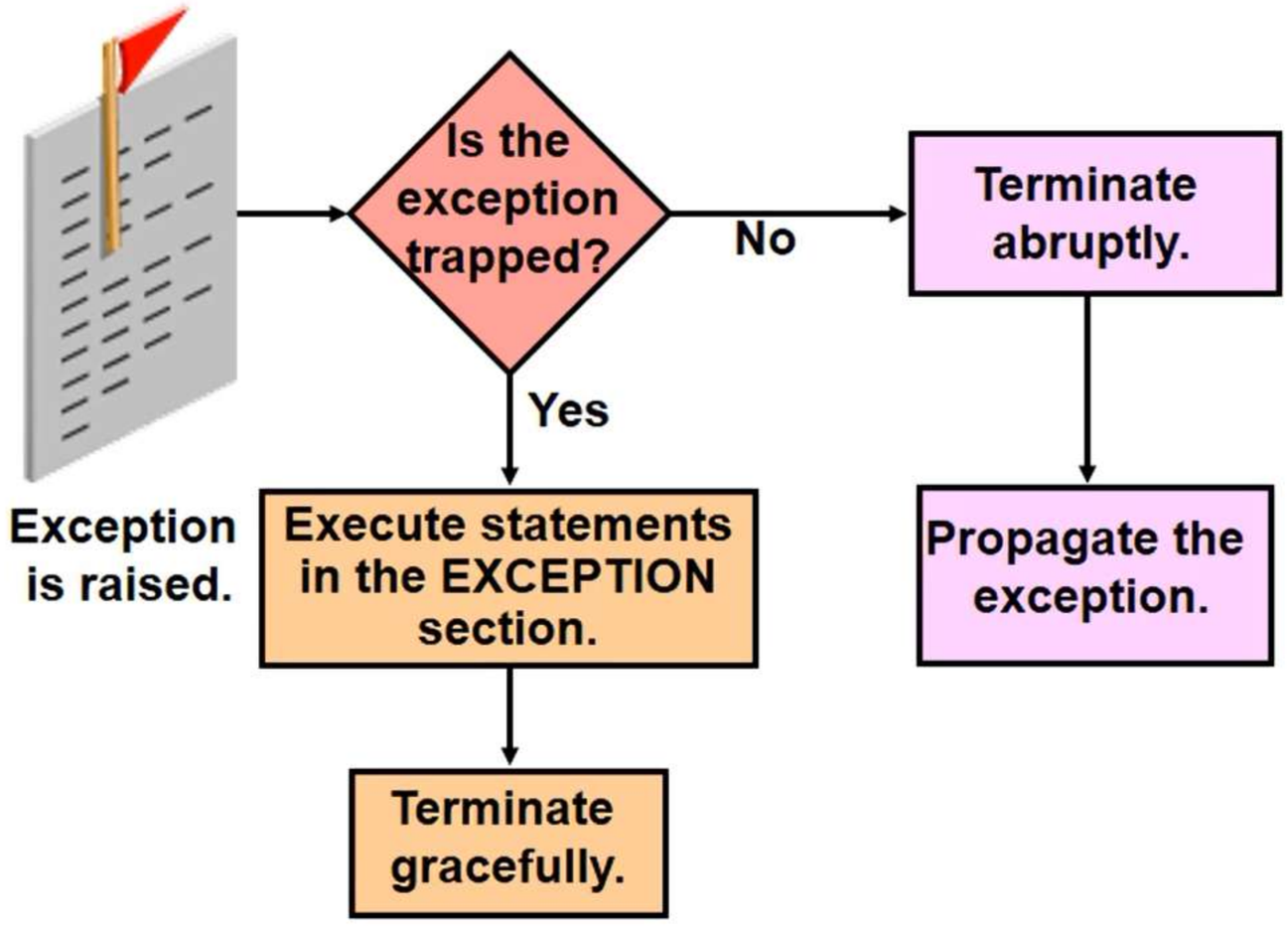
An exception can be handled

- By trapping it with a handler
- By propagating it to the calling environment

When an exception is raised,

- normal execution of the PL/SQL block or subprogram stops
- control transfers to its exception-handling part
- control does NOT return to where the exception was raised.

Exception Handling



Exception Handling

TEKNOTURF

Syntax

```
DECLARE
        Declaration section
BEGIN
        Exception section
EXCEPTION
        WHEN ex_name1 THEN
                -Error handling statements
        WHEN ex_name2 THEN
                -Error handling statements
        WHEN Others THEN
                -Error handling statements
END;
/
```

Exception Handling

TEKNOTURF

- Exception types
  - Predefined Oracle Server
  - Nonpredefined Oracle Server

  } Implicitly raised
- User-defined
- Sample predefined exceptions — Explicitly raised

Exception Name	Meaning
TOO_MANY_ROWS	Raised if a SELECT INTO statement returns more than one row.
ZERO_DIVIDE	Raised if you try to divide a number by zero.
NO_DATA_FOUND	Raised if a SELECT INTO statement returns no rows.
INVALID_CURSOR	Raised if you try an illegal cursor operation. For example, if you try to CLOSE an unopened cursor.
DUP_VAL_ON_INDEX	Raised if you try to store duplicate values in a database column that is constrained by a unique index.

Pre-Defined Exception

TEKNOTURF

Example

```
DECLARE
    v_emp_id number(6) := 1010;
    employee_info employees%ROWTYPE;
BEGIN
     SELECT * INTO emp_info FROM employees
                        WHERE emp_id = v_emp_id;
     dbms_output.put_line(employee_info.name);
     dbms_output.put_line(employee_info.salary);
EXCEPTION when NO_DATA_FOUND then
     dbms_output.put_line('Record not found');
END;
/
```

Search for an employee with employee id 1010. Raises an exception if record not found. The exception is handled by displaying a message "Record not found".

User Defined Exception

TEKNOTURF

```
DECLARE
        invalid_age EXCEPTION;
        name varchar2(15):='&name';
        age number:=&age;
BEGIN
        if age>=18 then
            dbms_output.put_line(name|| 'is eligible to vote');
        else
            RAISE  invalid_age;
        end if;
EXCEPTION
        WHEN invalid_age THEN
            dbms_output.put_line(name||‘  is not eligible to vote');
END;
 /
```

RAISE_APPLICATION_ERROR

TEKNOTURF

RAISE_APPLICATION_ERROR is a procedure used to issue user-defined error messages.

It can be used to report errors to application and avoid returning unhandled exceptions.

Syntax

```
RAISE_APPLICATION_ERROR(error_number, message);
```

Summary

TEKNOTURF

- Introduction to Cursors
- Implicit Cursors
- Explicit Cursors
- Exception Handling
- Predefined Exception
- User Defined Exceptions
- Other Exception Handler

TEKNOTURF

XML AND JSON

XML WITH JSON

TEKNOTURF

You will learn about

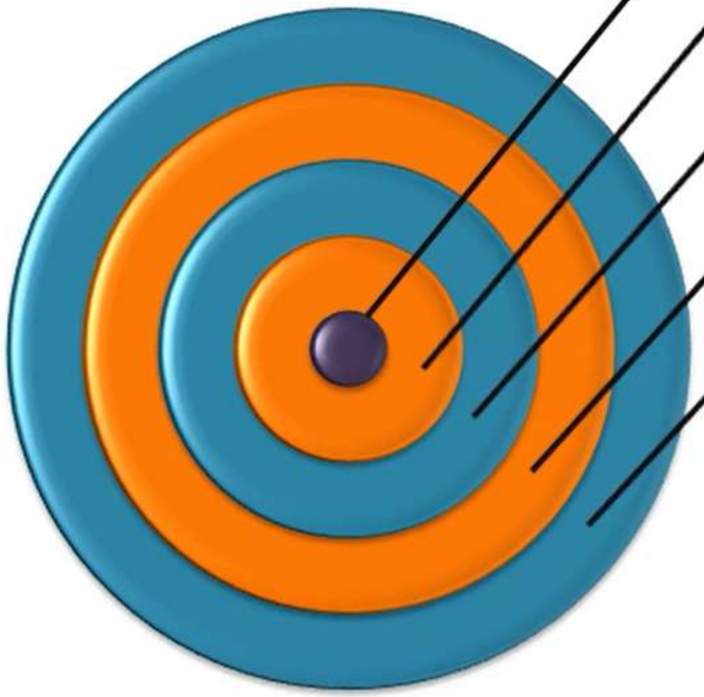
- Introduction to XML
- Working with Elements, Attributes, Entities And Processing Instructions
- XML Schema Definition
- Introduction To JSON
- Comparison between JSON and XML

TEKNOTURF

XML

Introduction To Extensible Markup Language

TEKNOTURF



XML: A W3C standard to complement HTML

HTML describes presentation

XML describes content

XML is a platform-independent way to represent data

Ideal as “Data Interchange” format

An Example

TEKNOTURF

1. Let us assume that you want to use XML to represent information about a transaction.
2. This transaction originates on salesman's iPad, it will then be sent to the Windows server, and ultimately mainframe, so it needs to be very flexible.

```
<?xml version="1.0"?>
<transaction  ID="THX1138">
  <salesperson>bluemax</salesperson>
  <order>
    <product productNumber="3263827">
      <quantity>1</quantity>
      <unitprice currency="standard">3000000</unitprice>
      <description>Medium  Trash  Compactor</description>
    </product>
  </order>
</transaction>
```

Represent the Data this way, so that it is adaptable to different environments.

XML In Use

TEKNOTURF

Storing data

- The most obvious use of XML is to store data in Data Bases that support XML Data

Web services

- All of the leading methods of Web services, SOAP, REST, and even XML-RPC, are based in XML

Podcasting and other data syndication

Does XML lend itself to application development?

- The Document Object Model (DOM)
- The Simple API for XML (SAX)

Transforming XML data (XSLT); manipulation you want to do with XML doesn't even require programming

Can I use XML with my favorite programming language?

- Java/ PHP/ Perl/ Python/ C++/ Ruby/ JavaScript

XML Syntax

TEKNOTURF

Comments:

- <!-- this is a comment -->

Naming conventions:

The following styles are popular:

- Camel Case Notation
- Dot . notation
- Underscore _ notation

Best Practice:

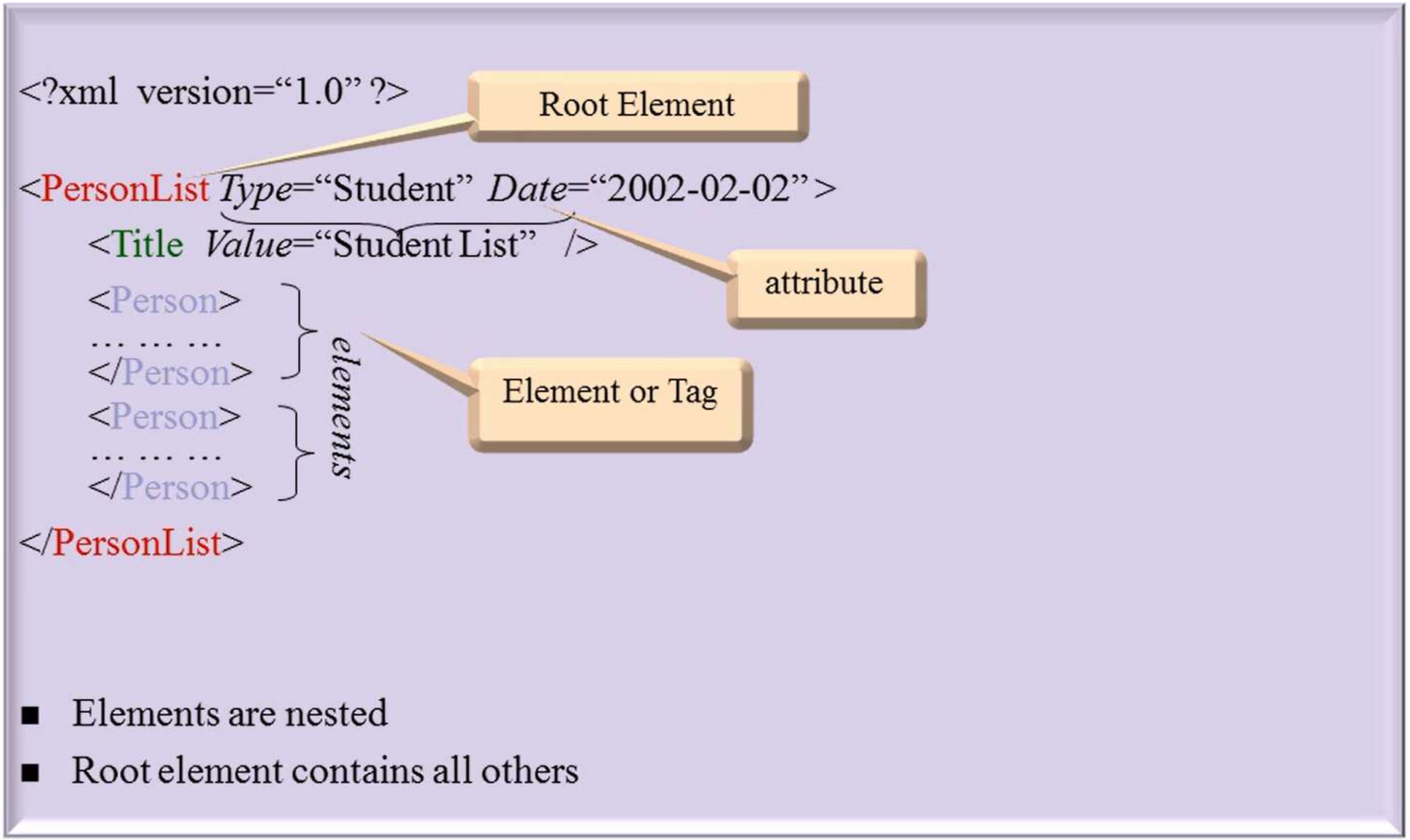
Use indentation to represent the document's hierarchy

Predefined entities exist to address ambiguous syntax

Example:

- <range>> 6 & < 20</range>

Terminology



- Elements are nested
- Root element contains all others

XML Declaration

```
<?xml  version="version_number"
     encoding="encoding_declaration"
     standalone="standalone_status" ?>
```

Example:

```
<?xml  version="1.0" encoding="UTF-8" standalone="yes" ?>
```

XML Declaration

TEKNOTURF

Possible Attribute Values

Attributes	Possible Values
version	1.0
encoding	UTF-8, UTF-16, ISO-10646-UCS-2, ISO-10646-UCS-4, ISO-8859-1 to ISO-8859-9, ISO-2022-JP, Shift_JIS, EUC-JP
standalone	yes, no

Processing Instruction

TEKNOTURF

Embed application-specific instructions in documents

Syntax:

- <? target arg1 arg2 ... ?>

The target name immediately follows "<?" and is used to associate the PI with an application

PIs may include zero or more arguments

"Processing instruction" is often abbreviated as PI in documentation.

Well Formed XML Documents

TEKNOTURF

There must be a single root element

- All other elements are nested inside the root element

Elements must be properly terminated

- For every opening tag "<...>" there must be a matching closing tag "</...>“
- Tags with no content (or body) may be self-terminated ("<.../>")

Elements must be properly nested underneath a parent tag (except for the single, root element)

- A nested tag-pair may not overlap another tag
- There is no limit to the nesting level of child elements

Well Formed XML Documents

TEKNOTURF

Element names or Tag names are case sensitive

- All tag and attribute names, attribute values, and data must comply with XML naming rules

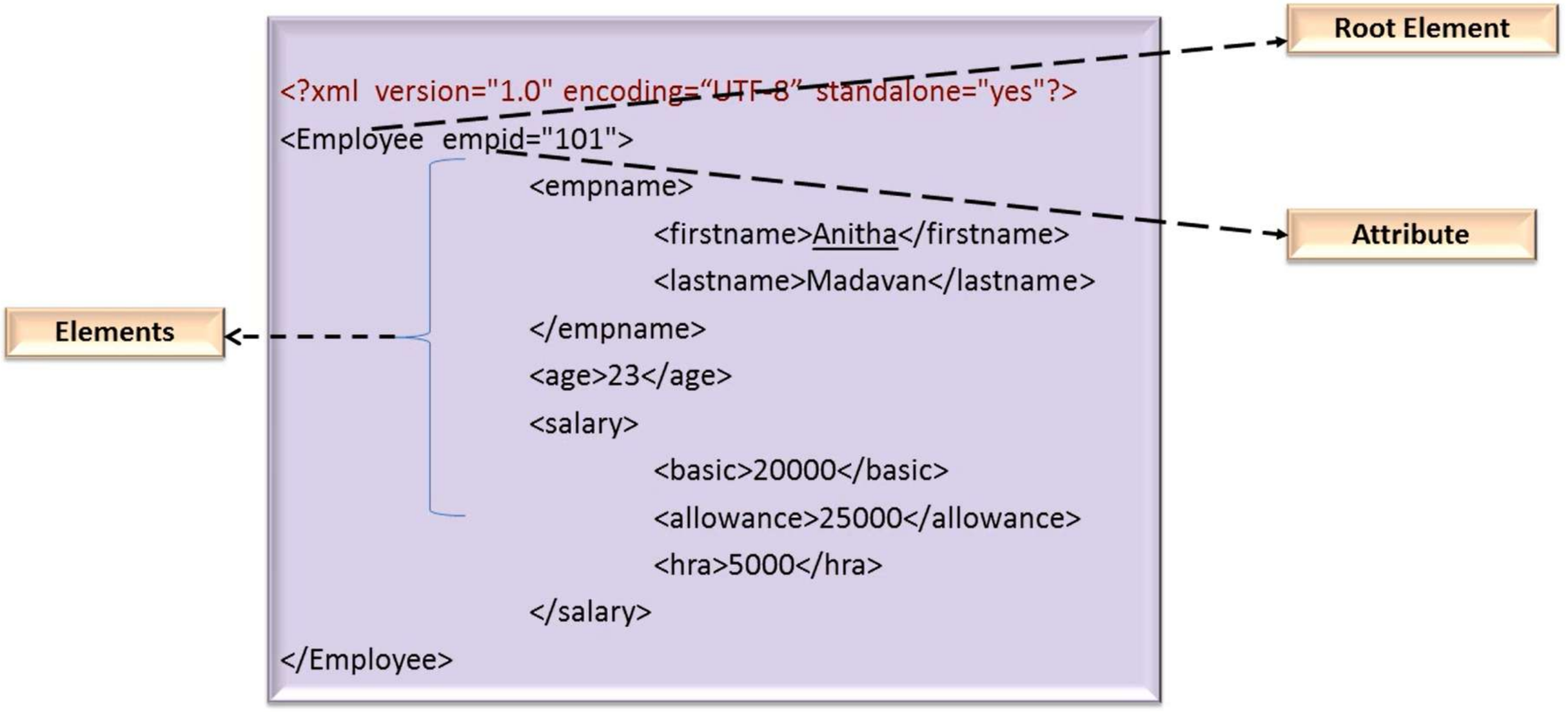
Attributes ("extra" information that can be provided for elements) must be properly quoted

- That is, all attribute values must lie inside double quotation marks

The first line should contain a prolog that identifies the version of the XML specification to apply:

- <?xml version="1.0" ?>

XML sample


Root Element
Attribute
Elements
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Employee empid="101">
<empname>
<firstname>Anitha</firstname>
<lastname>Madavan</lastname>
</empname>
<age>23</age>
<salary>
<basic>20000</basic>
<allowance>25000</allowance>
<hra>5000</hra>
</salary>
</Employee>

What is DTD?

TEKNOTURF

DTD means Document Type Definition

Purpose of DTD

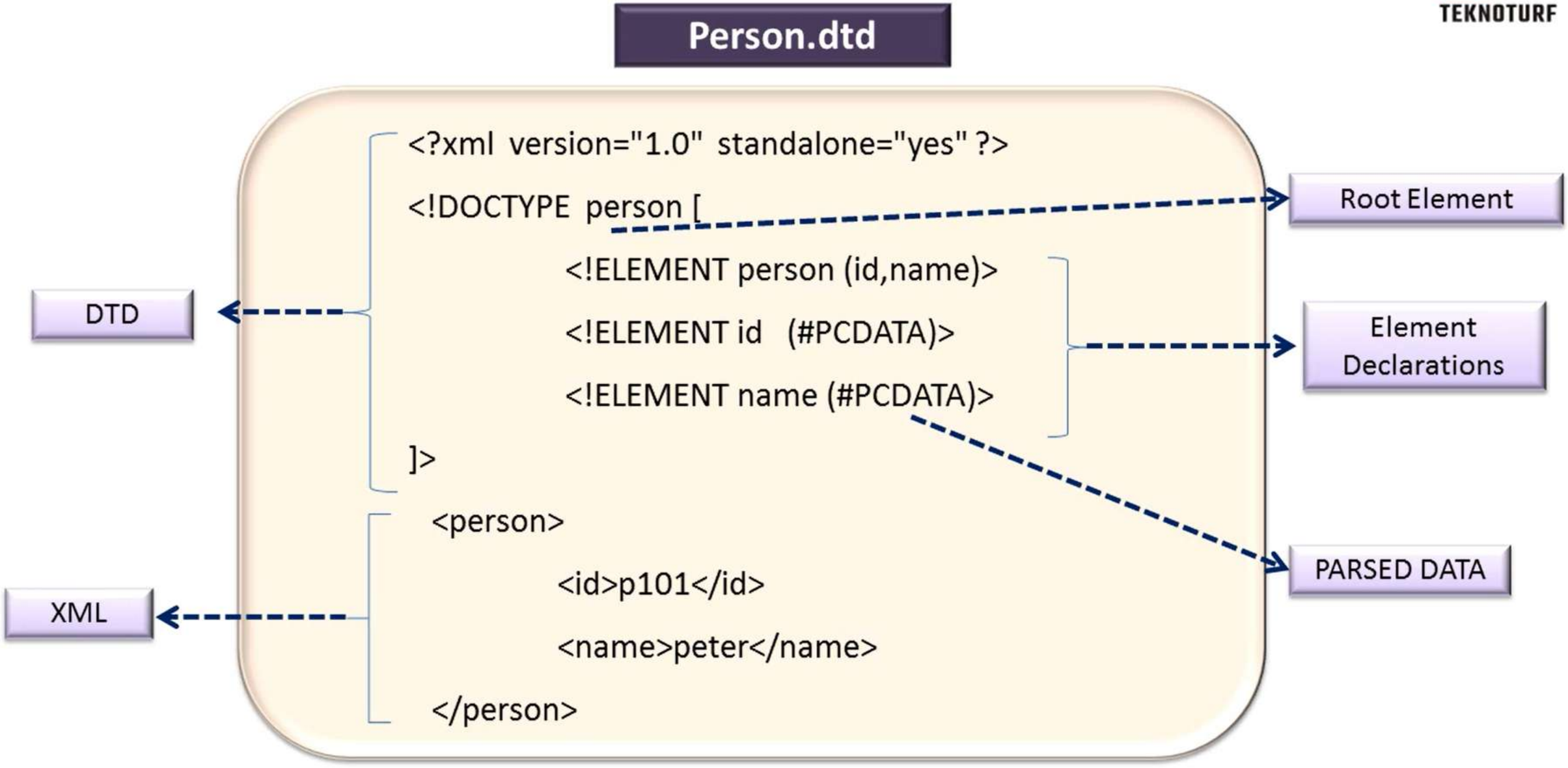
- Define the structure of the XML document.
- To ensure that XML files conform to a known structure, writing DTD is preferred.
- DTD specifies which tags to use, what attributes these tags can contain and which tags can occur inside which other tags.

An XML document is well-formed if it obeys the XML rules for tags.

An XML document is valid if

- It is well formed
- It conforms to the rules stated in the publicly available DTD

DTD - Example


Person.dtd
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE person [
<!ELEMENT person (id,name)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
]>
<person>
<id>p101</id>
<name>peter</name>
</person>
DTD
XML
Root Element
Element Declarations
PARSED DATA

Entities in DTD

TEKNOTURF

- ENTITIES are used to reference data that act as an abbreviation or can be found at an external location.
- The first character of an ENTITY value must be a letter, '_', or ':'
- It is used in DTD

DTD

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE company [
<!ELEMENT company (#PCDATA)>
<!NOTATION jpeg SYSTEM "JPG">
<!ENTITY cr "Copyright (C) ABC Solutions Ltd" >
<!ENTITY logo SYSTEM "logo.jpg" NDATA jpeg>
]>
<company>
    <copyright>&cr; </copyright>
    <companylogo>logo</companylogo>
</company>
```

<!ENTIT Y entity-name "entity-value">

Predefined Entities

TEKNOTURF

Predefined Entities	Declaring Entities in DTD
&lt;	<!ENTITY lt "(<">
&gt;	<!ENTITY gt "A">
&amp;	<!ENTITY amp "''">
&apos;	<!ENTITY apos "(">
&quot;	<!ENTITY quot "#">

XML Schemas

TEKNOTURF

XML schema is a richer, more powerful way of establishing vocabularies (or “grammar”) for XML instance documents

XML schema is an XML-based markup language

The XML schema definition language is often abbreviated as XSD

XML schema files are identified by .xsd extension

When we say “XML Schemas,” we usually mean the W3C XML Schema Language

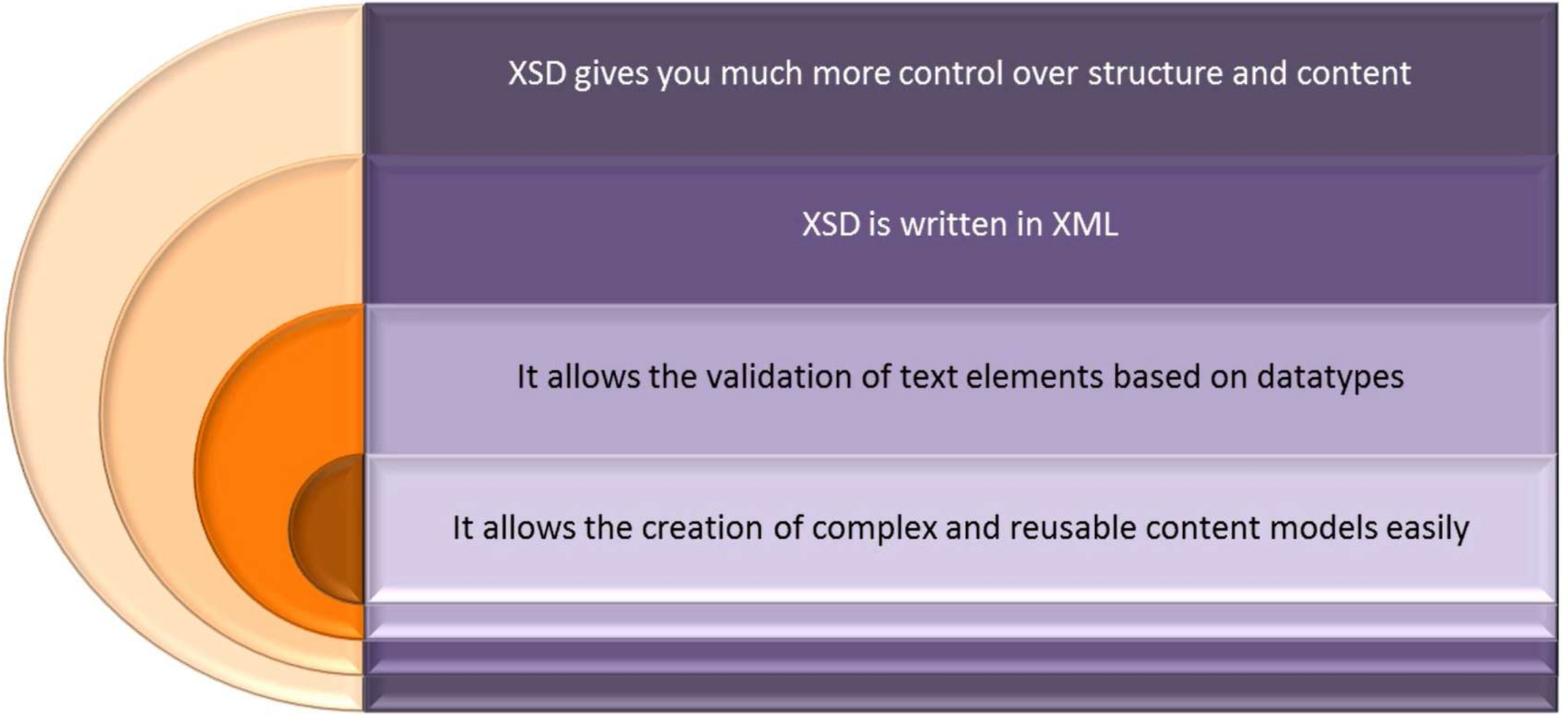
An XML document is well-formed if it obeys the XML rules for tags

An XML document is valid if

- It is well formed and
- It conforms to the rules stated in the schema

Why XML Schemas?

TEKNOTURF



Referring To A Schema

TEKNOTURF

To refer to an XML Schema in an XML document, the reference goes in the root element:

```
<?xml version="1.0"?>
  <rootElement
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          (The XML Schema Instance reference is required)
    xsi:noNamespaceSchemaLocation="url.xsd">
          (This is where your XML Schema definition can be found)
     ...
  </rootElement>
```

XSD Document

TEKNOTURF

The root element is <schema>

The XSD starts like this:

- <?xml version="1.0"?>

  <xs:schema xmlns:xs="http://www.w3.rg/2001/XMLSchema">

The <schema> element may have attributes:

- xmlns:xs="http://www.w3.org/2001/XMLSchema"
  - This is necessary to specify where all XSD tags are defined
- elementFormDefault="qualified"
  - This means that all XML elements must be qualified

Example For XML Schema

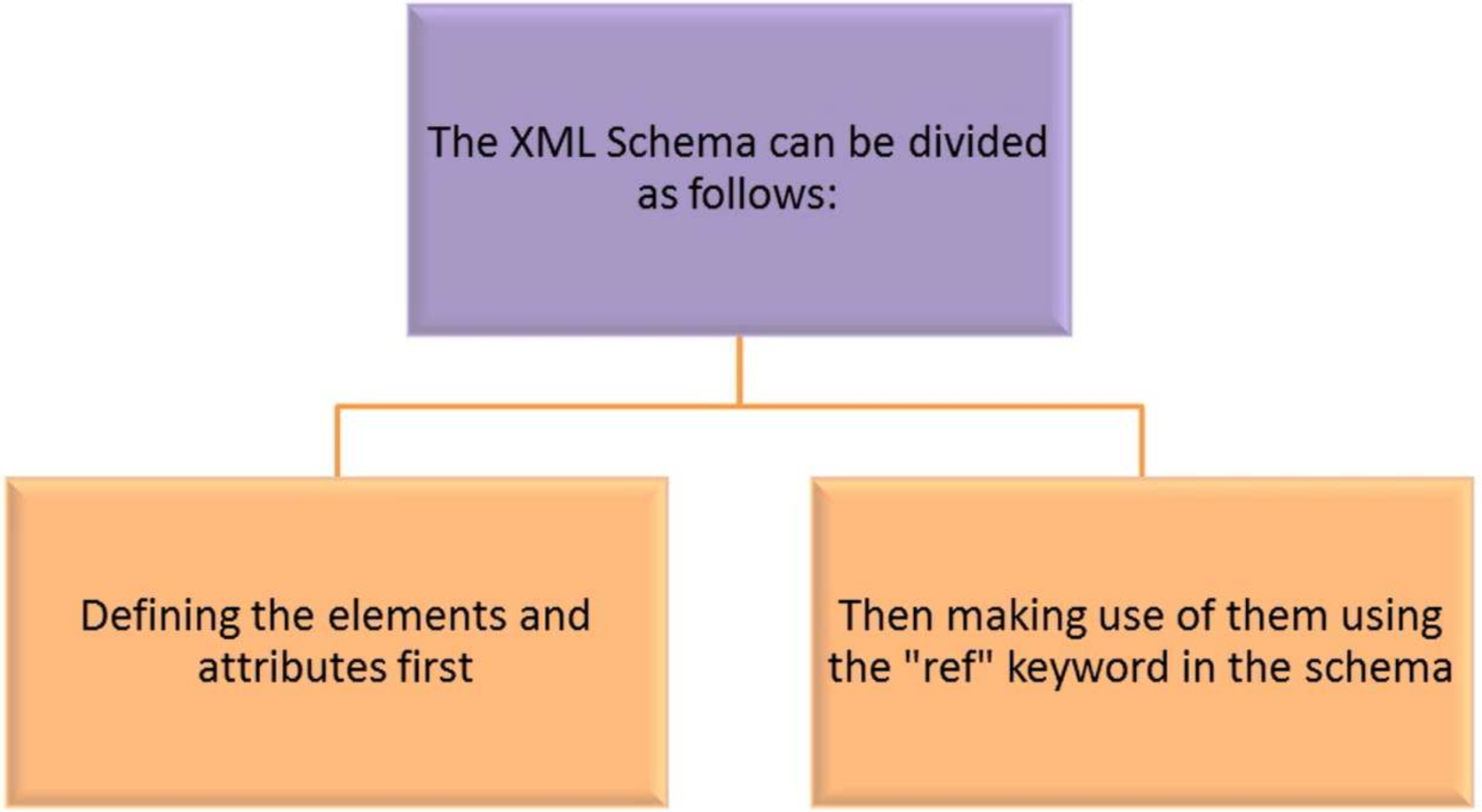
TEKNOTURF

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:element name="books">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="book">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="title" type="xs:string"></xs:element>
              </xs:sequence>
              <xs:attribute name="size" type="xs:string"></xs:attribute>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book size="oversize">
    <title>Computer Choreography</title>
  </book>
</books>
```

Dividing XML Schema

TEKNOTURF



Example for Dividing XML Schema using " ref "

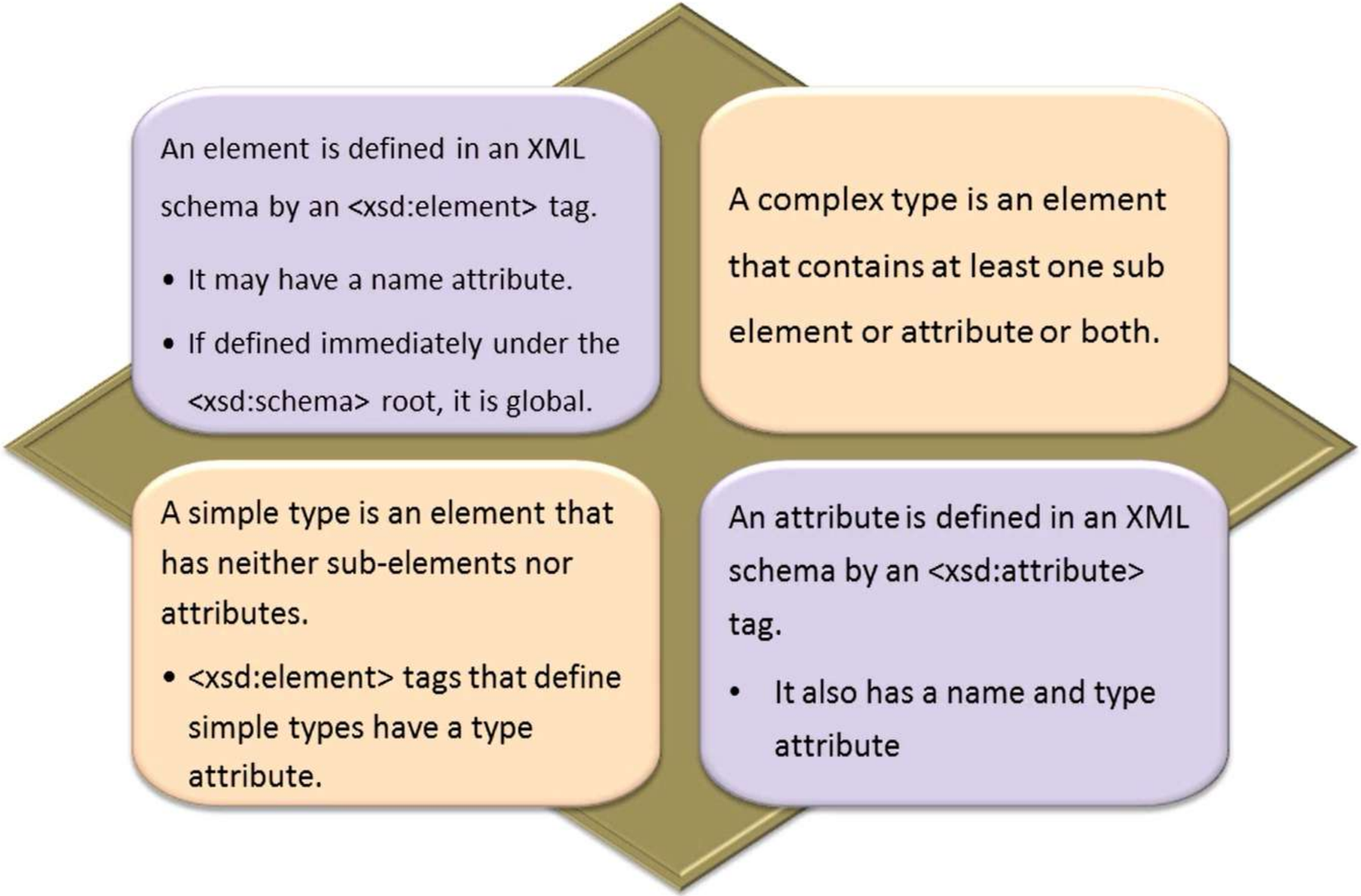
TEKNOTURF

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <xsd:element name="title" type="xsd:string"/>
   <xsd:element name="book">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="title"/>
       </xsd:sequence>
       <xsd:attribute name="size" type="xsd:string"/>
     </xsd:complexType>
   </xsd:element>
   <xsd:element name="books">
     <xsd:complexType>
       <xsd:sequence>
         <xsd:element ref="book"/>
       </xsd:sequence>
     </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book size="oversize">
    <title>Computer Choreography</title>
  </book>
</books>
```

XSD Terminology

TEKNOTURF



Defining A Simple Element

TEKNOTURF

A simple element is defined as
<xs:element name="name" type="type" /> where:

- name is the name of the element
- The most common values for type are
  xs:boolean xs:integer
  xs:date xs:string
  xs:decimal xs:time

Other attributes a simple element may have:

- default="default value" if no other value is specified
- fixed="value"

Restrictions Or Facets

TEKNOTURF

The general form for putting a restriction on a text value is:

Syntax

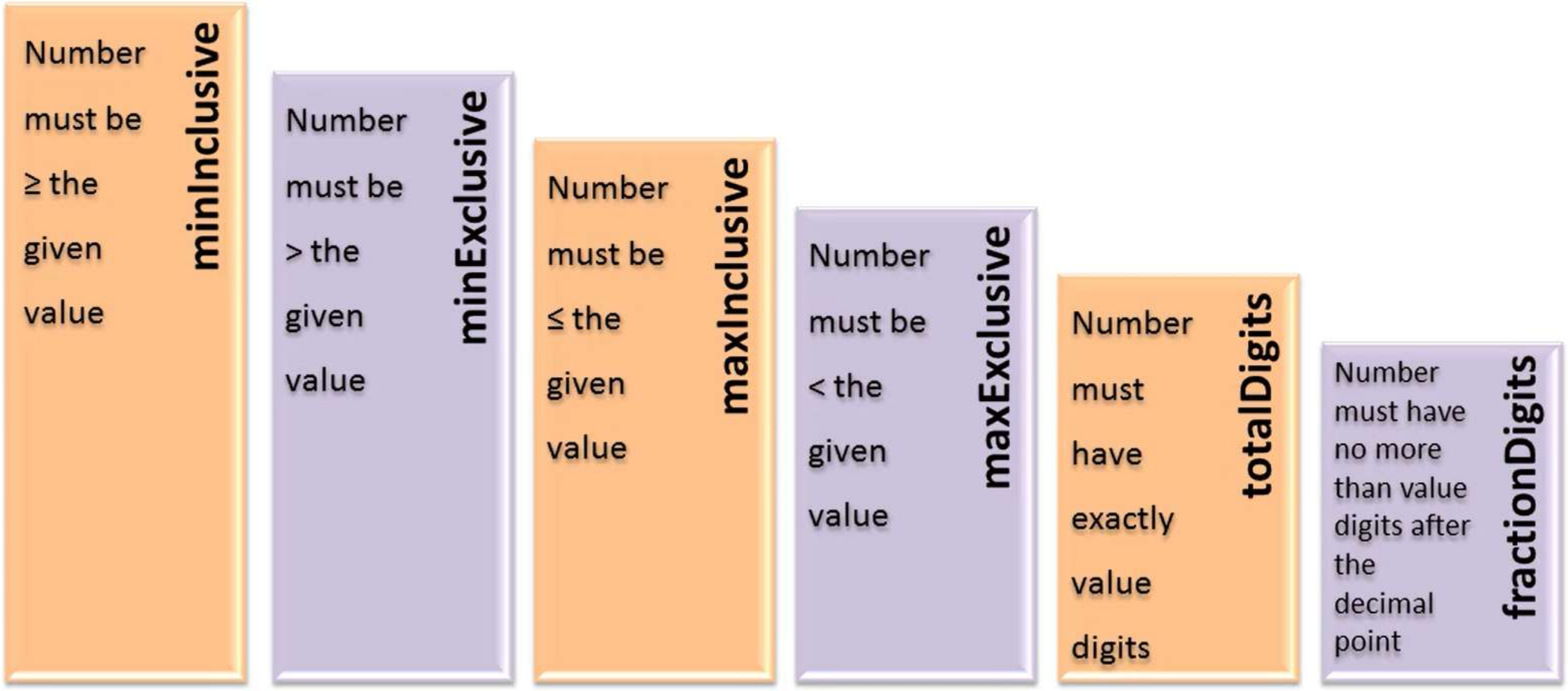
```
<xs:element name="name"> [or xs:attribute ]
  <xs:simpleType> [ or <xs:complexType> ]
    <xs:restriction base="type">
      ... the restrictions ...
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

For example:

```
<xs:element name="age">
<xs:simpleType>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="100"/>
  </xs:restriction>
<xs:simpleType>
</xs:element>
```

Restriction On Numbers

TEKNOTURF



Restriction On Strings

TEKNOTURF

length -- the string must contain exactly value characters

minLength -- the string must contain at least value characters

maxLength -- the string must contain no more than value characters

whiteSpace -- tells what to do with whitespace

- value="preserve" Keep all whitespace
- value="replace" Change all whitespace characters to spaces
- value="collapse" Remove leading and trailing whitespace, and replace all sequences of whitespace with a single space

Enumeration

TEKNOTURF

An enumeration restricts the value to be one of a fixed set of values

Example:

```
<xs:element name="season">
    <xs:simpleType>
        <xs:restriction  base="xs:string">
            <xs:enumeration value="Spring"/>
            <xs:enumeration value="Summer"/>
            <xs:enumeration value="Autumn"/>
            <xs:enumeration value="Fall"/>
            <xs:enumeration value="Winter"/>
        </xs:restriction>
    </xs:simpleType>
</xs:element>
```

Complex Elements

TEKNOTURF

A complex element is defined as

```
<xs:element   name="name">
   <xs:complexType>
      ... information about the complex type...
   </xs:complexType>
</xs:element>
```

Example:

```
<xs:element   name="person">
   <xs:complexType>
      <xs:sequence>
         <xs:element  name="firstName"  type="xs:string" />
         <xs:element  name="lastName"  type="xs:string" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

xs:sequence

TEKNOTURF

Sequence indicator defines that elements must occur in the specified order

```
<xs:element name="address">
 <xs:complexType>
    <xs:sequence>
       <xs:element name= "name" type="xs:string" />
       <xs:element name= "doorno" type="xs:positiveInteger"
/>
       <xs:element name= "streetname" type="xs:string" />
       <xs:element name= " city" type="xs:string" />
       <xs:element name= " pincode" type="xs:positiveInteger"
/>
   </xs:sequence>
  </xs:complexType>
</xs:element>
```

XML file can be written as:

```
<address>
  <name> Tom </name>
  <doorno> 12 </doorno>
  <streetname> Nehru Street
</streetname>
   <city> Chennai </city>
   <pincode> 600003 </pincode>
</address>
```

xs:all

TEKNOTURF

xs:all allows elements to appear in any order

```
<xs:element   name="person">
  <xs:complexType>
     <xs:all>
        <xs:element  name="firstName"  type="xs:string" />
        <xs:element  name="lastName"  type="xs:string" />
     </xs:all>
  </xs:complexType>
</xs:element>
```

- Despite the name, the members of an xs:all group can occur once or not at all
- You can use minOccurs="n" and maxOccurs="n" to specify how many times an element may occur (default value is 1)
  - In this context, n may only be 0 or 1

xs:choice

TEKNOTURF

Choice indicators define that either one of the child elements must occur within the element

```
<xs:element name="student">
   <xs:complexType>
         <xs:choice>
               <xs:element name="email"
type="xs:string"/>
               <xs:element name="mobile"
type="xs:integer"/>
         </xs:choice>
   </xs:complexType>
</xs:element>
```

The XML File can be written in both ways:

```
<student>
  <email> tom123@gmail.com </email>
</student>
```

```
<student>
  <mobile> 9456723145 </mobile>
</student>
```

Mixed Elements

TEKNOTURF

- Mixed elements may contain both text and elements
- We add mixed="true" to the xs:complexType element
- The text itself is not mentioned in the element, and may go anywhere (it is basically ignored)

```
<xs:complexType name="paragraph" mixed="true">
  <xs:sequence>
    <xs:element name="someName" type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

Predefined Date And Time Types

TEKNOTURF

xs:date

A date in the format CCYY-MM-DD, for example, 2002-11-05

xs:time

A date in the format hh:mm:ss (hours, minutes, seconds)

xs:dateTime

Format is CCYY-MM-DD hh:mm:ss

Allowable restrictions on dates and times:

enumeration, minInclusive, maxExclusive, maxInclusive,

maxExclusive, pattern, whiteSpace

Predefined Numeric Types

TEKNOTURF

Here are some of the predefined numeric types:

- xs:decimal
- xs:positiveInteger
- xs:byte
- xs:negativeInteger
- xs:short
- xs:nonPositiveInteger
- xs:int
- xs:nonNegativeInteger
- xs:long

Allowable restrictions on numeric types:

- enumeration, minInclusive, maxExclusive, maxInclusive, maxExclusive, fractionDigits, totalDigits, pattern, whiteSpace

Namespaces in XSD

TEKNOTURF

- Namespaces are used to avoid naming conflicts
- Namespaces are declared as an attribute of an element
- They can be declared either at the root element or can be declared at any element in the XML document
- A namespace is declared as follows:

- **xmlns** -> A reserved word which is used for binding namespaces

Example for Namespace in XML Schema

TEKNOTURF

Multiple Namespaces

```
<?xml version="1.0" encoding="UTF-8"?>
<students xmlns:stud="http://www.lms.com/student"
        xmlns:dept="http://www.lms.com/department">
    <student>
        <stud:rollno>1</stud:rollno>
        <stud:name>Tom</stud:name>
        <dept:name>ECE</dept:name>
    </student>
</students>
```

Example For Default Namespaces

TEKNOTURF

Elements Using Default Namespace

Default Namespace

```
<?xml version="1.0"?>
<Book xmlns="http://www.library.com">
        <Title>Nineteen Eighty-Four</Title>
        <Author>George Orwell.</Author>
        <purchase xmlns="http://www.otherlibrary.com">
                <Title>The Lord of the Rings</Title>
                <Author>George and Weedon Grossmith</Author>
        </purchase>
        <Title>Middlemarch</Title>
         <Author>George Eliot</Author>
</Book>
```

The elements Book, Title and Author of "Nineeen Eighty-Four" and "Middlemarch" are associated with the namespace "http://www.library.com"

The elements purchase, Title and Author of "The Lord of the Rings“ are associated with the namespace "http://www.otherlibrary.com"

Target Namespace

TEKNOTURF

- It is possible to place the definitions for each schema file within a distinct namespace
- It can be done by adding the attribute targetNamespace into the schema element in the XSD file

```
<?xml version="1.0"?>
        <xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="myNamespace"> ... </xs:schema>
```

where targetNamespace -> Unique Identifier

- If the targetNamespace attribute is placed at the top of the XSD schema, then all the entities defined in it are part of this namespace

Example for Target Namespace

TEKNOTURF

```
<xsd:schema  targetNamespace="http://www.SampleStore.com/Account"
        xmlns:xsd="http://www.w3.org/1999/XMLSchema"
        xmlns:ACC= "http://www.SampleStore.com/Account">
<xsd:element name="InvoiceNo" type="xsd:positive-integer"/>
<xsd:element name="ProductID" type="ACC:ProductCode"/>
<xsd:simpleType name="ProductCode"  base="xsd:string">
        <xsd:pattern value="[A-Z]{1}d{6}"/>
</xsd:simpleType>
```

- The names defined in a schema are said to belong to its *target namespace*
- Definitions and declarations in a schema can refer to names that may belong to other namespaces referred to as *source namespaces*

Include And Import

TEKNOTURF

Reusable schemas can be created by including parts of other schemas or even whole schemas into a parent schema.

XML schema provides two mechanisms for this:

- **<xsd:include schemaLocation="..." />**
- The included schema must have the same **targetNamespace as the** including schema
- **<xsd:import namespace="..." schemaLocation="..." />**
  - The imported schema may specify a different **targetNamespace.**

Example For import

TEKNOTURF

Person.xsd (Root XML Schema)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
            targetNamespace="http://www.example.com/person"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.example.com/person"
            xmlns:phno="http://www.example.com/phoneno"
            elementFormDefault="qualified">

 <xs:import schemaLocation="http://www.example.com/phoneno.xsd"
            namespace="http://www.example.com/phoneno"/>

 <xs:element name="person">
    <xs:complexType>
       <xs:sequence>
          <xs:element name="name" type="xs:string"/>
         <xs:element ref="phno:phoneno minOccurs="0"

maxOccurs="unbounded"/>
       </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

phoneno.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.com/phoneno"
xmlns="http://www.example.com/phoneno"
     elementFormDefault="qualified">
  <xs:element name="phoneno">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="type" type="xs:string"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Example For include

order.xsd (Root XML Schema)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://example.org/ord"
        xmlns="http://example.org/ord">

 <xs:include schemaLocation="order-no.xsd"/>
 <xs:include schemaLocation="customer.xsd"/>

 <xs:element name="order" type="OrderType"/>
 <xs:complexType name="OrderType">
  <xs:sequence>
   <xs:element name="number" type="OrderNumType"/>
   <xs:element name="customer" type="CustomerType"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

order-no.xsd

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns="http://example.org/ord"
        targetNamespace="http://example.org/ord">
 <xs:simpleType name="OrderNumType">
  <xs:restriction base="xs:string"/>
 </xs:simpleType>
</xs:schema>
```

customer.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:complexType name="CustomerType">
  <xs:sequence>
   <xs:element name="name" type="CustNameType"/>
   <xs:element name="number" type="xs:integer"/>
  </xs:sequence>
 </xs:complexType>
 <xs:simpleType name="CustNameType">
  <xs:restriction base="xs:string"/>
 </xs:simpleType>
</xs:schema>
```

Validating XML Schema

TEKNOTURF

PersonList.xml

```
<?xml version="1.0"  encoding="UTF-8"?>
<PersonList
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
     xsi:noNamespaceSchemaLocation="PersonList.xsd">
     <Person>
          <adhaarno>414356782345</adhaarno>
          <name>
                    <firstname>Zeenath</firstname>
          </name>
          <age>28</age>
    </Person>
</PersonList>
```

Validating XML Schema

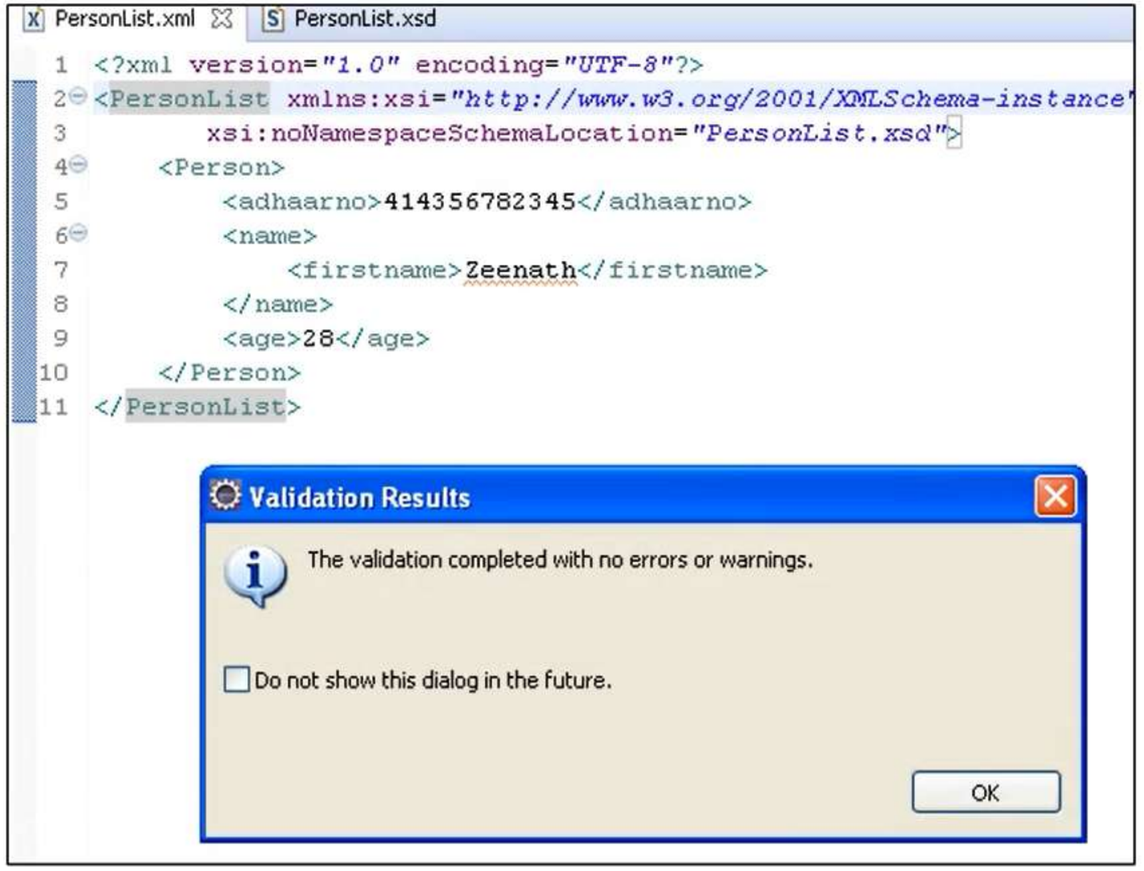
TEKNOTURF

PersonList.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:element name="PersonList">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="Person">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="adhaarno" type="xs:long"></xs:element>
                            <xs:element name="name">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="firstname"
                                            type="xs:string"></xs:element>
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="age" type="xs:int"></xs:element>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Validating XML Schema

- Many parsers are available to check
- Eclipse Tool has a built-in-parser



ENTITY in XML SCHEMA

TEKNOTURF

- In XML Schema there is a type to represent ENTITY type element **xs:ENTITY**
- The **type xs:ENTITY** represents a reference to an **unparsed entity**.
- The **xs:ENTITY type** is most often used to include information from another location that is not in XML format, such as **picture** etc.,
- An **xs:ENTITY value** must be an NCName – Non Qualified name like **picture**, **pattern [\i-[:]][\c-[:]]*, White Space**
- An **xs:ENTITY value** carries the additional constraint that it must match the name of an unparsed entity in a **Document Type Definition (DTD)**.

xs:ENTITY Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
 <xs:element name="catalog">
  <xs:complexType>
   <xs:sequence>
    <xs:element maxOccurs="unbounded" ref="product"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="product">
  <xs:complexType>
   <xs:sequence>
    <xs:element ref="number"/>
    <xs:element ref="picture"/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name="number" type="xs:integer"/>
 <xs:element name="picture">
  <xs:complexType>
   <xs:attribute name="location" use="required" type="xs:ENTITY"/>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

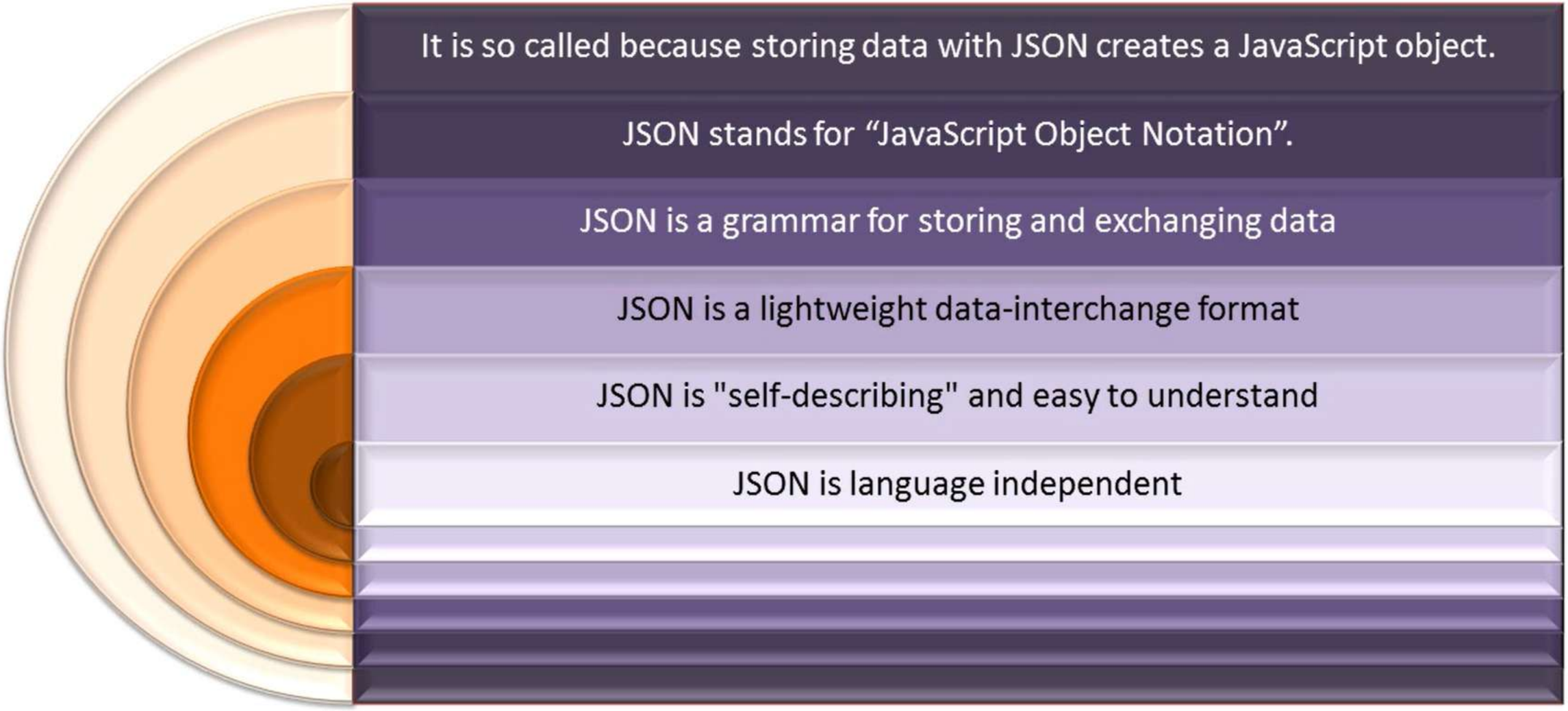
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog [
<!NOTATION jpeg SYSTEM "JPG">
<!ENTITY prod557 SYSTEM "prod557.jpg" NDATA jpeg>
<!ENTITY prod563 SYSTEM "prod563.jpg" NDATA jpeg>
]>

<catalog xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
     xsi:noNamespaceSchemaLocation="Product.xsd">
 <product>
  <number>557</number>
  <picture location="prod557"/>
 </product>
 <product>
  <number>563</number>
  <picture location="prod563"/>
 </product>
</catalog>
```

TEKNOTURF

JSON

Introduction To JSON

TEKNOTURF



Why Use JSON

TEKNOTURF

Standard Structure: JSON objects have a standard structure that makes developers' job easy to read and write code.

Light weight: Since JSON is light weighted, it becomes easier to get and load the requested data quickly. When exchanging data between a browser and a server, the data can only be text. Since JSON format is text, it can easily be sent to and from a server

Scalable: JSON is language independent. JSON structure is same for all the languages. So it can work well with most of the modern programming languages.

JSON Syntax Rules

TEKNOTURF

JSON syntax is derived from JavaScript object notation syntax:

- JSON Data is in name/value pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

A name/value pair consists of a field name (in double quotes), followed by a colon, followed by a value:

Example: "name":"John“

In JSON, *keys* must be strings, written with double quotes

A *value* can be:

a string, a number, true, false, null, an object, or an array

- Values can be nested

JSON Datatypes

TEKNOTURF

In JSON, *values* must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- null

JSON values **cannot** be one of the following data types:

- a function
- a date
- *undefined*

JSON Objects

TEKNOTURF

A JSON object is an unordered set of name/value pairs

- The pairs are enclosed within braces { }
- Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).
- There is a colon between the name and the value
- Pairs are separated by commas

Example

- { “no” : “1”,
- “name” : ”Sam”,
- “gender” :”male”,
- “phone”:”555 1234567”}

JSON Datatypes

TEKNOTURF

JSON Strings : Strings in JSON must be written in double quotes and can contain the usual assortment of escaped characters

Example : { "name":"John" }

JSON Numbers : Numbers in JSON must be an integer or a floating point.

Example: { "age":30 }

JSON Booleans : Values in JSON can be true/false.

Example:{ "sale":true }

JSON Booleans : Values in JSON can be true/false.

Example:{ "sale":true }

JSON Datatypes

TEKNOTURF

JSON Objects : Values in JSON can be objects.
Example:

```
{
"employee":{ "name":"John", "age":30, "city":"New York" }
}
```

Objects as values in JSON must follow the same rules as JSON objects.

JSON Arrays : Values in JSON can be arrays.
Example

```
{
"employees":[ "John", "Anna", "Peter" ]
}
```

JSON Objects

TEKNOTURF

Accessing Object Values : You can access the object values by using dot (.) notation.

Example:

```
myObj = { "name":“Sathish", "age":20, "car":null };

x = myObj.name;
```

You can also access the object values by using bracket ([]) notation:

Example:

```
myObj = { "name":“Sathish", "age":20, "car":null };

x = myObj["name"];
```

JSON Objects

TEKNOTURF

- **Nested JSON Objects :** Values in a JSON object can be another JSON object.

  Example:

```
myObj = {
  "name":“Sathish",
  "age":20,
  "cars": {
    "car1":"Ford",
    "car2":"BMW",
    "car3":"Fiat"
  }
}
```

- **Accessing nested JSON Objects:**

  Example:

```
x = myObj.cars.car2;
  ( OR )
x = myObj.cars["car2"];
```

JSON Arrays

TEKNOTURF

- An *array* is an ordered collection of values
  - The values are enclosed within brackets [ ]
  - Values are separated by commas

Example:

```
[
        {
            "name": "html",
            "years": 5
        },
        {
            "name": "jquery",
            "years": 6
        }
]
```

XML And JSON

TEKNOTURF

- Both JSON and XML are "self describing"
- Both of them are hierarchical
- Both of them can be parsed and used by lots of programming languages
- Both of them can be fetched with an XMLHttpRequest

XML And JSON

TEKNOTURF

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
 ]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName><lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName><lastName>Smith</lastName>
  </employee>
</employees>
```

Comparison Between XML And JSON

TEKNOTURF

JSON	XML
JSON supports only text and number data type.	XML supports many data types such as text, number, images, charts, graphs etc.,
JSON supports array.	XML doesn't support array
JSON is simple to read and write.	XML is less simpler than JSON.
JSON is data-oriented.	XML is document-oriented.
JSON doesn't provide display capabilities.	XML provides the capability to display data because it is a markup language.
JSON is less secured than XML.	XML is more secured.

XML WITH JSON - Summary

TEKNOTURF

- Introduction to XML
- Working with Elements, Attributes, Entities And Processing Instructions
- XML Schema Definition
- Introduction To JSON
- JSON Datatypes
- Comparison between JSON and XML

TEKNOTURF

JQUERY

In this module you will learn

TEKNOTURF

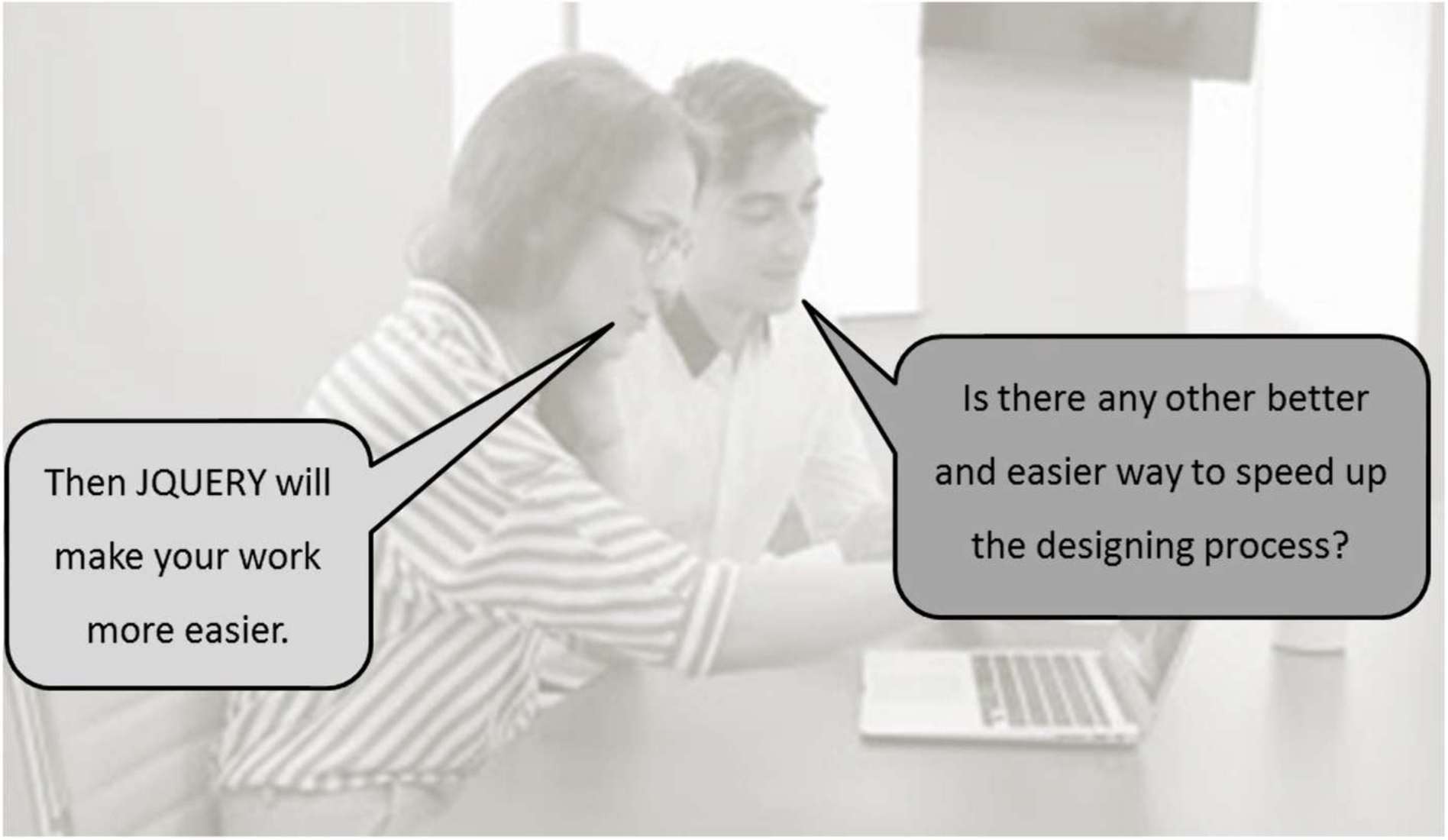
- Introduction to jQuery
- Using jQuery Libraries
- Event Handling in jQuery
- jQuery by examples
- Handling User Scrolling
- Handling Resizing
- Images and Slideshows

Overview

TEKNOTURF

Overview

TEKNOTURF



Introduction to jQuery

TEKNOTURF

jQuery is a JavaScript Library created by John Resig in 2006.

JQuery is a lightweight, open-source JavaScript library that simplifies interaction between HTML and JavaScript.

jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development.

Why jQuery?

TEKNOTURF

Cross Browser Support

Extensibility through plug-ins

DOM manipulation

Event Handling.

AJAX Processing

Creating effects / animations

Using jQuery Libraries

TEKNOTURF

There are two ways to use jQuery:

Local Installation - Can download jQuery library on the local machine and include it into the HTML code

CDN Based Version - Can include jQuery library in the HTML code directly from Content Delivery Network (CDN)

Local Installation

TEKNOTURF

- Download the latest version of Jquery script file from the following URL

  https://jquery.com/download/

- Version used - **Jquery 3.1.1**
- Copy the **JS file** into the needed location and refer that file within the **html code** of a web page like:
- **<script type="text/javascript" src="jquery/jquery-3.1.1.js"></script>**

.js file is present inside the folder jquery (relative path)

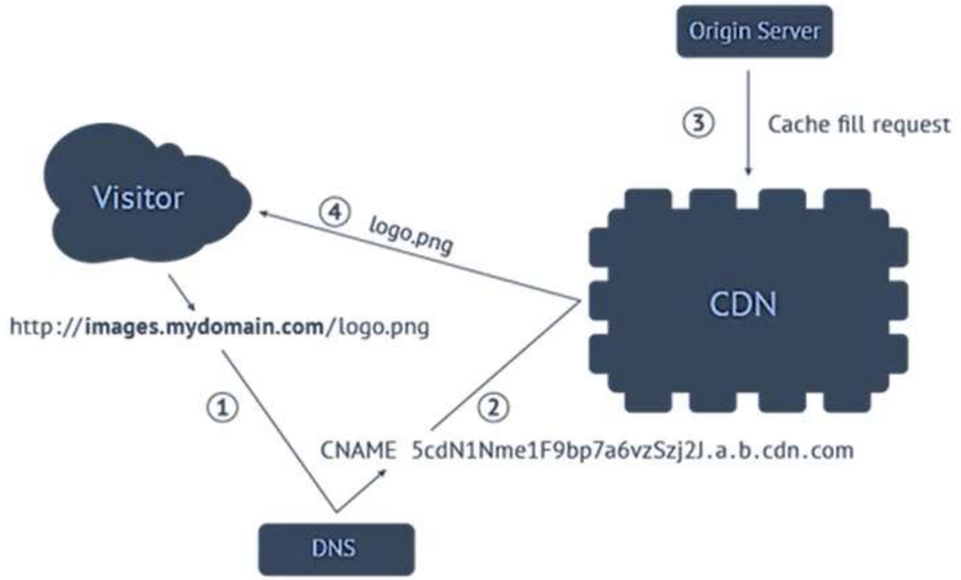
CDN Installation

TEKNOTURF

- You can refer the jQuery library within your HTML code directly from Content Delivery Network (CDN)
- Both Google and Microsoft host jQuery files

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
```



Production version (compressed): The production version is meant to be used in a working application.

we can download jquery.min.js and access it locally using relative address mode

Working with jQuery

TEKNOTURF

All the JavaScript works that need to be performed, happens after the document is ready.

Example - adding events

```
$(document).ready(function()
{
        //JavaScript code
} );
```

Everything inside it will load as soon as the DOM is loaded, and before the page contents are loaded.

Example

Here, **'document'** refers to the HTML element and **ready()** is the action on the selected element.

```
<html>
<head>
<script type="text/javascript" src="jquery/jquery-3.1.1.js"> </script>
<script type="text/javascript" >
$(document).ready(function()
{
        document.write("Welcome to JQuery");
});
</script>
</head>
<body>
</body>
</html>
```

Output

jQuery Selectors

TEKNOTURF

Selectors are used to select one or more HTML elements using jQuery.

jQuery selectors start with the dollar sign and parentheses – $()

Selectors can be

- **Tag name** - $('name of the tag') - $('p')
  - Selects all elements which match with the given element Name.
- **Id** - $('#p_id')
  - Selects a single element which matches with the given ID.
- **Class** - $('.p_class')
  - Selects all elements which match with the given Class.

The factory function $() is a synonym of jQuery() function. You can use function jQuery() instead of $().

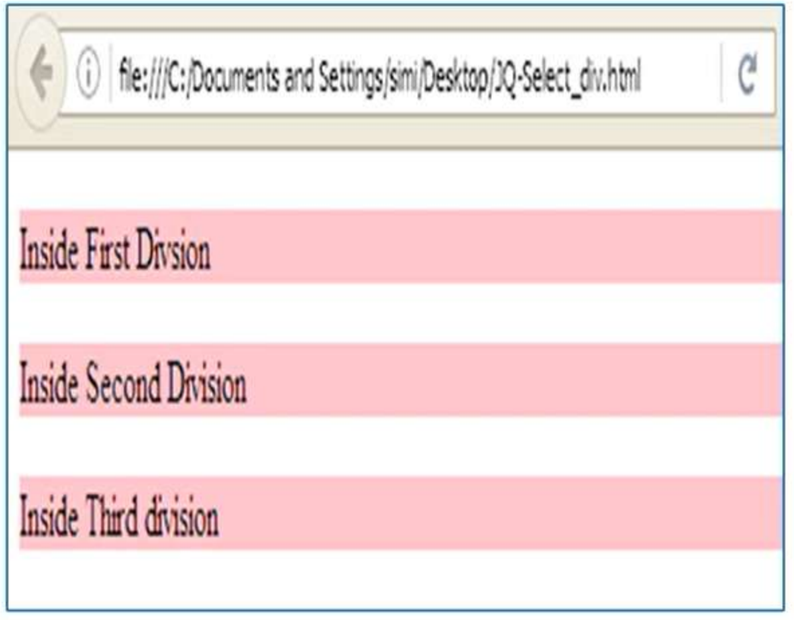
Using Tag Name Selector

TEKNOTURF

```
<html>
<head>
<script type = "text/javascript"
src = "https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
<script type = "text/javascript"
language = "javascript">
$(document).ready(function()
{
    $("div").css("background-color", "pink");
});
 </script>
</head>
<body>
     <div class = "big" id = "div1">
         <p>Inside First Division</p>
      </div>
      <div class = "medium" id = "div2">
         <p>Inside Second Division</p>
      </div>
      <div class = "small" id = "div3">
         <p>Inside Third division</p>
      </div>
   </body>
</html>
```

This would select all the **div tags** in the html file

Output



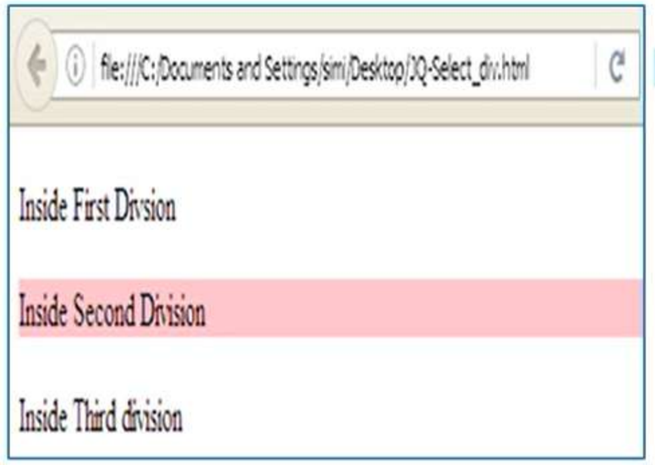
Using #id Selector

TEKNOTURF

```
<html>
<head>
<script type = "text/javascript"
    src = "https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
    $(document).ready(function()
    {
        $("#div2").css("background-color", "pink");
    });
 </script>
</head>
<body>
     <div class = "big" id = "div1">
         <p>Inside First Division</p>
      </div>
      <div class = "medium" id = "div2">
         <p>Inside Second Division</p>
      </div>
      <div class = "small" id = "div3">
         <p>Inside Third division</p>
      </div>
   </body>
</html>
```

This would select the **div** tag which has the **id div2**

Output



Using .class Selector

TEKNOTURF

```
<html>
<head>
<script type = "text/javascript"
    src = "https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">
</script>
<script type = "text/javascript" language = "javascript">
    $(document).ready(function()
    {
        $(".big").css("background-color", "pink");
    });
</script>
</head>
<body>
     <div class = "big" id = "div1">
         <p>Inside First Division</p>
      </div>
      <div class = "medium" id = "div2">
         <p>Inside Second Division</p>
      </div>
      <div class = "small" id = "div3">
         <p>Inside Third division</p>
      </div>
   </body>
</html>
```

This would select only the division which belongs to the class 'big'

Output

.css() is an action which sets the css attributes to the specified elements selected by the selector

jQuery Selectors

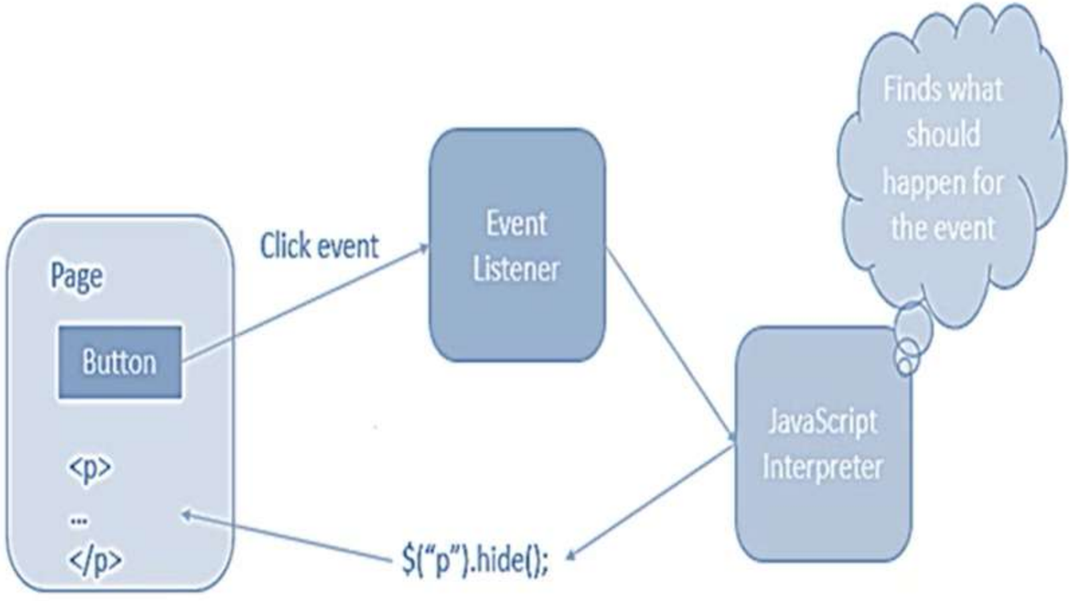
TEKNOTURF

Selector	Description
$('*')	This selector selects all elements in the document
$("p > *")	This selector selects all elements that are children of a paragraph element
$("li:not(.myclass)")	Selects all elements matched by ** ** that do not have **class="myclass"**
$("p a.specialClass")	This selector matches links with a class of specialClass declared within **<p>** elements
$("ul li:first")	This selector gets only the first ** ** element of the ** **.
$(":empty")	Selects all elements that have no children
$("p:empty")	Selects all elements matched by **<p>** that have no children
$("div[p]")	Selects all elements matched by ** ** that contain an element matched by **<p>**
$("input[@name=myname]")	Selects all elements matched by **<input>** that have a name value exactly equal to myname

Event Handling

TEKNOTURF

- Events are user's actions.
- jQuery events are the actions that can be detected by your web application.
- The term "fires/fired" is often used with events.



"The **click event** is fired, the moment when you click a button".

Types of Events

TEKNOTURF

Here are some common **DOM** events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Mouse Events

TEKNOTURF

```
<script>
    $(document).ready(function()
    {
            $("button").click(function()
        {
                    $("p").hide();
            });
    });
</script>
```

When click event is fired, **click()** will be executed

```
<script>
    $(document).ready(function()
    {
            $("h1,h2,h3").click(function()
        {
                    $(this).hide();
            });
        });
</script>
```

When you click on h1 or h2 or h3 tag, the contents within those tags will be **hidden**

```
<script>
    $(document).ready(function()
    {
            $("#p1").mouseenter(function()
        {
                    alert("You have Entered !!!");
            });
    });
</script>
```

When the mouse pointer enters into the paragraph, it shows the **alert box**

Mouse Events

TEKNOTURF

```
<script>
    $(document).ready(function()
    {
            $("p").on({
                    mouseenter: function(){
                    $(this).css("background-color", "lightgray"); },
                    mouseleave: function(){
                    $(this).css("background-color", "lightblue");},
                    click: function(){
                    $(this).css("background-color", "yellow");  },
                    });
    });
</script>
```

The **on()** method attaches one or more event handlers for the selected elements

```
<script>
    $(document).ready(function()
    {
            $("#p1").hover(function(){
                    alert("You entered into paragraph!");
                },
        function(){
                alert("Bye! You now leave the Praragraph");
            });
    });
</script>
```

hover() method takes two functions and is a combination of the **mouseenter()** and **mouseleave()** methods.

Scroll Events

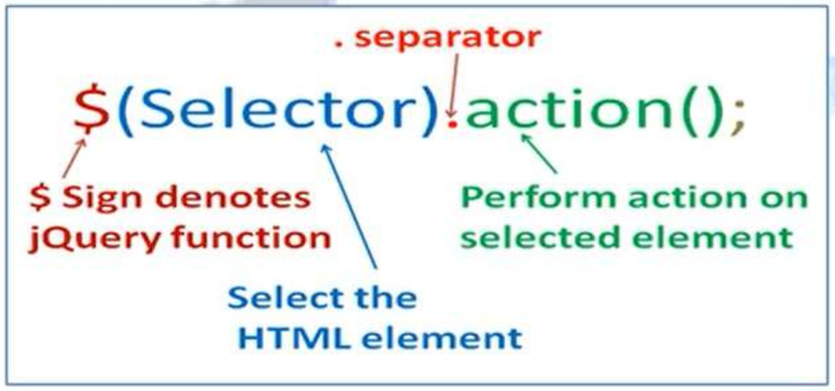
TEKNOTURF

The scroll event occurs when the user scrolls the specified element.

The scroll event works for all scrollable elements and the window object (browser window).

The scroll() method triggers the scroll event, or attaches a function to run when a scroll event occurs.

Syntax



Example

$(selector).scroll()

or

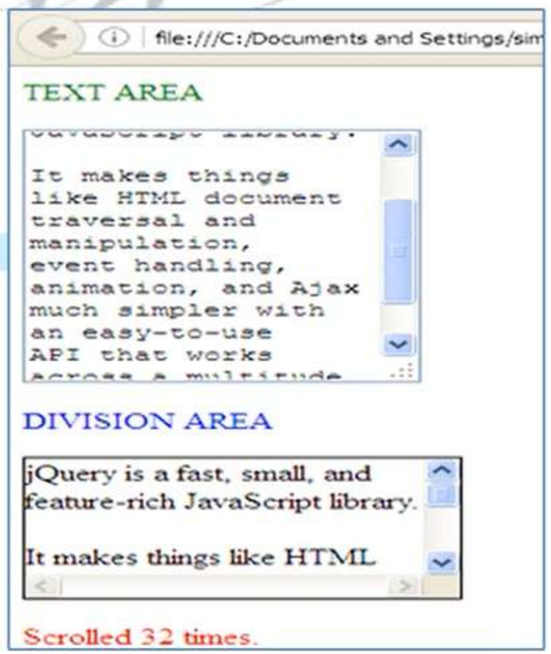
$(selector).scroll(function)

Scroll Events

```
<html>
<head>
<script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
<script>
    $(document).ready(function()
    {
        $(txtscroll).scroll(function(){
        alert ("Scroll event occured!");
    });
    x = 0;
    $("div").scroll(function(){
        $("span").text( x+= 1);
        }); });
</script>
</head>
<body>
<p><font color="Green"> TEXT AREA </font></p>
<textarea cols="20" rows="10" id="txtscroll">
 jQuery is a fast, small, and feature-rich JavaScript library.
<br><br>
    It makes things like HTML document traversal and manipulation,
    event handling, animation, and Ajax much simpler with an easy-to-use
    API that works across a multitude of browsers.
</textarea>
<P><font color="blue"> DIVISION AREA </font></p>
<div style="border:1px solid black;width:200px;height:100px;overflow:scroll;">
   JQuery is a fast, small, and feature-rich JavaScript library.
<br><br>
    It makes things like HTML document traversal and
    manipulation, event handling, animation, and Ajax
    much simpler with an easy-to-use
    API that works across a multitude of browsers.
</div>
<p><font color="red">Scrolled <span>0</span> times.</font></p>
</body>
</html>
```

The scroll event occurs when the user scrolls in the specified element

Output



Resize Events

TEKNOTURF

The **resize()** event is sent to the window element when the size of the browser window changes

Code in a resize handler should never rely on the number of times the handler is called.

As the **.resize()** method is just a shorthand for **.on("resize", handler),** detaching is possible using **.off("resize")**

Example

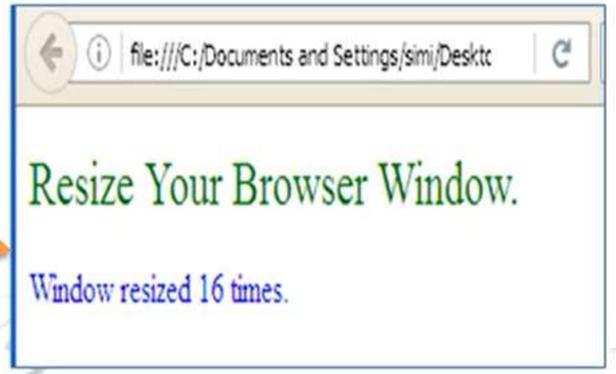
$(selector).resize()
or
$(selector).resize(handler function)

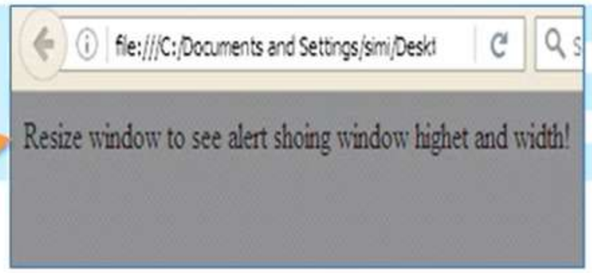
Resize Events - Example

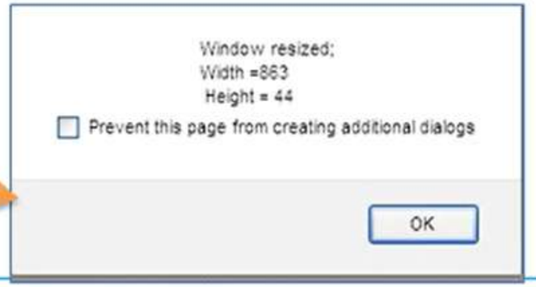
```
<script>
        x = 0;
        $(document).ready(function()
    {
            $(window).resize(function(){
                $("span").text(x += 1);
        });
         });
</script>
```

```
<script>
    $(document).ready(function()
    {
        $(window).resize(function(){
            alert ("Window resized;" + "\n" + "Width =" +
        $(window).width() + "\n" + " Height = " + $(window).height());
        });
    });
</script>
```

Output





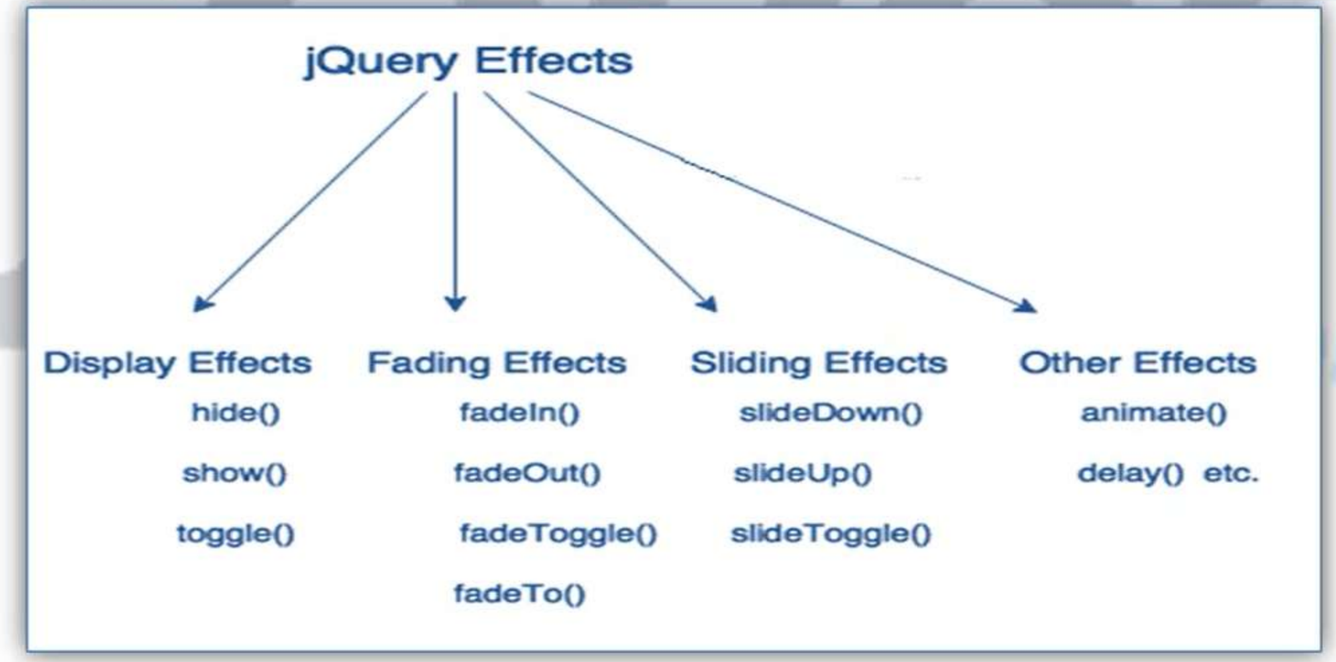


jQuery Effects

TEKNOTURF

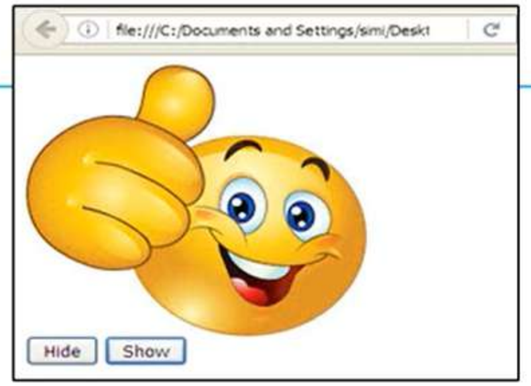
jQuery enables us to add effects on a web page.

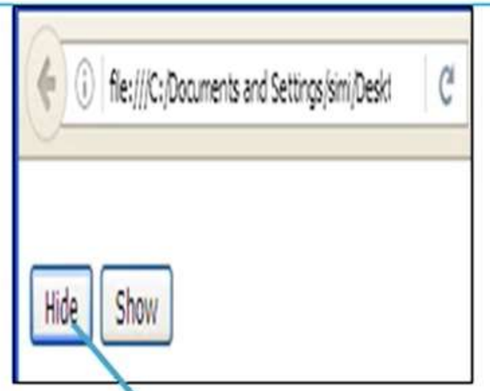
jQuery effects can be categorized into display, fading, sliding, hiding/showing and other animation effects.



Display Effects

```
<script type="text/javascript" >
    $(document).ready(function()
    {
        $('#hide').click(function(){
            $('img').hide();
        });
        $('#show').click(function(){
            $('img').show();
        });
    });
</script>
```





Hide and show can take an optional time in milliseconds to have transition

After clicking the hide button, image is hidden

```
<script type = "text/javascript" language = "javascript">
    $(document).ready(function()
    {
                $(".clickme").click(function(event){
                    $(".target").toggle('slow', function(){
                });  }); });
</script>
```

toggles() method toggles between the **hide()** and **show()** methods.

Fading Effects

TEKNOTURF

fadeIn() - Syntax

```
$(selector).fadeIn([speed, callback]);
```

Specifies the duration of the effect

It is a function to be executed after fading completes

fadeOut() - Syntax

```
$(selector).fadeOut([speed, callback]);
```

fadeToggle() - Syntax

```
$(selector).fadeToggle([speed, callback]);
```

fadeTo() - Syntax

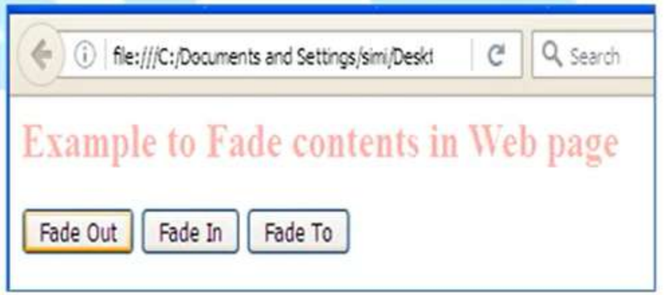
```
$(selector).fadeTo([speed, opacity, callback]);
```

Opacity value between 0 and 1

Fading Effects - Example

```
<script type="text/javascript" >
    $(document).ready(function() {
        $('#fadeout').click(function(){
            $('h3').fadeOut(1000);
        });
        $('#fadein').click(function(){
            $('h3').fadeIn(1000);
        });
        $('#fadeto').click(function(){
            $('h3').fadeTo(1000,.5);
        });
    });
</script>
```

Output



Sliding Effects

TEKNOTURF

slideDown() - Syntax

```
$(selector).slideDown([speed, callback]);
```

slideUp() - Syntax

```
$(selector).slideUp([speed, callback]);
```

slideToggle() - Syntax

```
$(selector).slideToggle([speed, callback]);
```

Sliding Effects - Example

```
<script type="text/javascript" >
    $(document).ready(function()
    {
        $('#slidedown').click(function(){
        $('h3').slideDown("slow");
    });
    $('#slideup').click(function(){
        $('h3').slideUp("fast");
    });
    $('#slidetoggle').click(function(){
        $('h3').slideToggle(1000);
    });});
</script>
```

Output

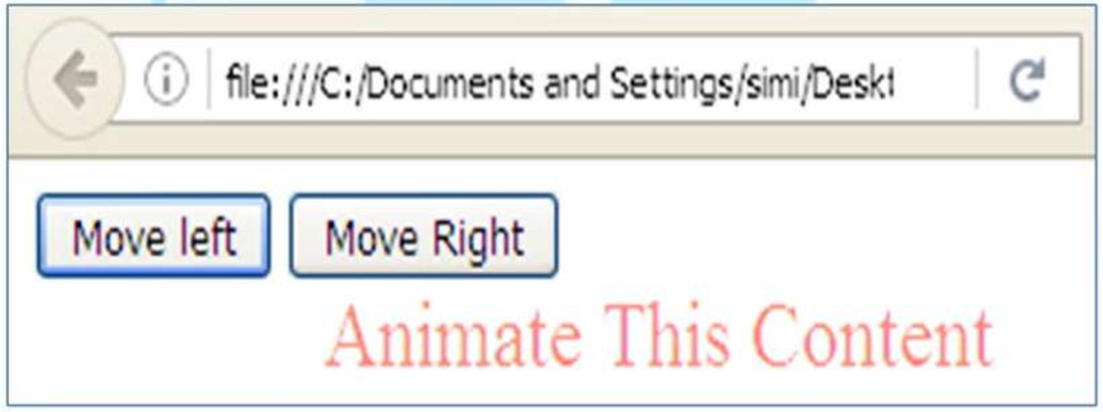
animate() - Example

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
    $(document).ready(function()
    {
                $('#left').click(function()
        {
        $('div').animate( {left: "-=50px" , height:'toggle', opacity:0.5}, "slow");
            });
            $('#right').click(function(){
        $('div').animate( {left: "+=50px", width : 'toggle',opacity:1}, "slow");
    }); });
</script>
<head>
<body>
    <button id="left">Move left</button>
    <button id="right">Move Right</button>
    <div style="left: 100px ; position: absolute ">
          <font size=5 color="red">Animate This Content</font></div>
</body>
</html>
```

Syntax

```
$(selector).animate({params}, speed, callback);
```

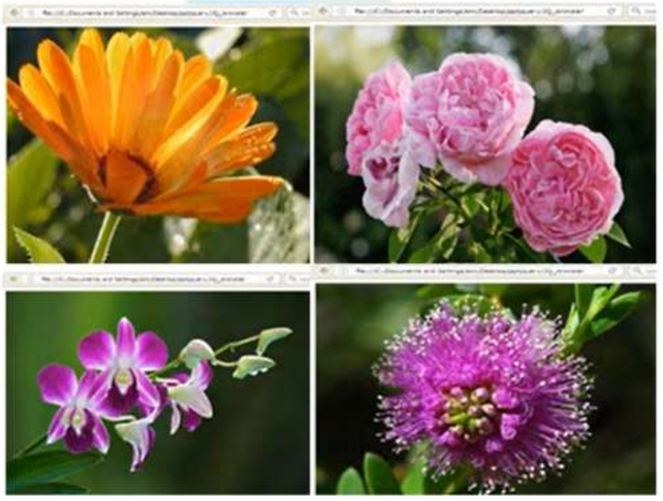
Output



SlideShow - Example

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js"></script>
<script>
function slideSwitch() {
    var $active = $('#slideshow IMG.active');
     if ( $active.length == 0 )
           $active = $('#slideshow IMG:last');
        var $next =  $active.next().length ? $active.next() : $('#slideshow IMG:first');
                    $active.addClass('last-active');
                    $next.css({opacity: 0.0}) .addClass('active') .animate({opacity: 1.0}, 1000,
        }
        $(function() {
                setInterval( "slideSwitch()", 1000 );
        });
</script>
<style type="text/css">
    .active{
        z-index:99;
    }
</style>
</head>
<body>
     <div id="slideshow">
        <img src="images/flower1.jpg" width=50% height=80%
            style="position:absolute;" class="active" />
        <img src="images/flower2.jpg" width=50% height=80%
                        style="position:absolute;"/>
          <img src="images/flower3.jpg" width=50% height=80%
                        style="position:absolute;"/>
          <img src="images/flower4.jpg" width=50% height=80%
                        style="position:absolute;" />
</div>
</body>
</html>
```

Output



Summary

TEKNOTURF

- Introduction to jQuery
- Using jQuery Libraries
- Event Handling in jQuery
- jQuery by examples
- Handling User Scrolling
- Handling Resizing
- Images and Slideshows