VIT®

BHOPAL

www.vitbhopal.ac.in

VIT BHOPAL

UNIVERSITY

A PLACE TO LEARN; A CHANCE TO GROW

WELCOME TO

CSE 4001 - INTERNET AND WEB PROGRAMMING

Unit 4

XML and AJAX PL/SQL: Declaring PL/SQL Variables, Writing Executable Statements, Using SQL Statements Within a PL/SQL XML Basics – Parser – DOM – XPath – XSL/XSLT – XQuery – DTD – Schema – Namespaces. AJAX Basics – Request and Response- AJAX using XML- AJAX using Java Script and JQuery.

Text Books:

- 1. Thomas Powell, HTML and CSS, Complete Reference, Fifth Edition, Mc Graw Hill, 2010
- 2. Thomas Powell, Fritz Schneider , JavaScript The complete reference, Mc Graw Hill, 2013
- 3. Tom Christiansen, Nathan Torkington, Perl Cookbook, O'Reilly, 2012
- 4. David Powers, PHP Solutions, Dynamic web page design made easy, Apress, 2010
- 5. Joe Fawcett, Danny Ayers, Liam R. E. Quin, Beginning XML, 5th Edition, Wrox, 2012

Reference Books:

- 1. Paul Dietel, Harvey Dietel and Abbey Dietel, Internet and World Wide Web How to program, 5th International Edition, Pearson, 2012

- PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java.
- This tutorial will give you great understanding on PL/SQL to proceed with Oracle database and other advanced RDBMS concepts.

- The PL/SQL programming language was developed by Oracle Corporation in the late 1980s as procedural extension language for SQL and the Oracle relational database.

Following are certain notable facts about PL/SQL –

- PL/SQL is a completely portable, high-performance transaction-processing language.
- PL/SQL provides a built-in, interpreted and OS independent programming environment.
- PL/SQL can also directly be called from the command-line **SQL*Plus interface**.

- Direct call can also be made from external programming language calls to database.
- PL/SQL's general syntax is based on that of ADA and Pascal programming language.
- Apart from Oracle, PL/SQL is available in **TimesTen in-memory database** and **IBM DB2**.

Features of PL/SQL

PL/SQL has the following features −

- PL/SQL is tightly integrated with SQL.
- It offers extensive error checking.
- It offers numerous data types.
- It offers a variety of programming structures.
- It supports structured programming through functions and procedures.
- It supports object-oriented programming.
- It supports the development of web applications and server pages.

Advantages of PL/SQL

PL/SQL has the following advantages −

- SQL is the standard database language and PL/SQL is strongly integrated with SQL.
- PL/SQL supports both static and dynamic SQL.
- Static SQL supports DML operations and transaction control from PL/SQL block.
- In Dynamic SQL, SQL allows embedding DDL statements in PL/SQL blocks.
- PL/SQL allows sending an entire block of statements to the database at one time.
- This reduces network traffic and provides high performance for the applications.

- PL/SQL gives high productivity to programmers as it can query, transform, and update data in a database.
- PL/SQL saves time on design and debugging by strong features, such as exception handling, encapsulation, data hiding, and object-oriented data types.

Applications written in PL/SQL are fully portable.

- PL/SQL provides high security level.
- PL/SQL provides access to predefined SQL packages.
- PL/SQL provides support for Object-Oriented Programming.
- PL/SQL provides support for developing Web Applications and Server Pages.

PL/SQL - Basic Syntax

- We will discuss the Basic Syntax of PL/SQL which is a **block-structured** language; this means that the PL/SQL programs are divided and written in logical blocks of code. Each block consists of three sub-parts −

1**Declarations**

- This section starts with the keyword **DECLARE**.
- It is an optional section and defines all variables, cursors, subprograms, and other elements to be used in the program.

2**Executable Commands**

- This section is enclosed between the keywords **BEGIN** and **END** and it is a mandatory section.
- It consists of the executable PL/SQL statements of the program.
- It should have at least one executable line of code, which may be just a **NULL command** to indicate that nothing should be executed.

3**Exception Handling**

- This section starts with the keyword **EXCEPTION**.
- This optional section contains **exception(s)** that handle errors in the program.

- Every PL/SQL statement ends with a semicolon (;).
- PL/SQL blocks can be nested within other PL/SQL blocks using **BEGIN** and **END**.

Following is the basic structure of a PL/SQL block −

- DECLARE <declarations section>
- BEGIN <executable command(s)>
- EXCEPTION <exception handling>
- END; **EX:**
- The **end;** line signals the end of the PL/SQL block. To run the code from the SQL command line, you may need to type / at the beginning of the first blank line after the last line of the code. When the above code is executed at the SQL prompt, it produces the following result −

The PL/SQL Identifiers

- PL/SQL identifiers are constants, variables, exceptions, procedures, cursors, and reserved words.
- The identifiers consist of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters.
- By default, **identifiers are not case-sensitive**. So you can use **integer** or **INTEGER** to represent a numeric value.
- You cannot use a reserved keyword as an identifier.
- The PL/SQL Delimiters
- A delimiter is a symbol with a special meaning. Following is the list of delimiters in PL/SQL −

DelimiterDescription

- **+, -, *, /**
- Addition, subtraction/negation, multiplication, division
- **%**Attribute indicator
- 'Character string delimiter**.**
- Component selector**(,)**
- Expression or list delimiter**:**
- Host variable indicator
- **,**Item separator
- "Quoted identifier delimiter
- **=**Relational operator

- **@**Remote access indicator
- **;**Statement terminator
- **:=**Assignment operator
- **=>**Association operator
- **||**Concatenation operator
- ****Exponentiation operator
- **<<, >>**Label delimiter (begin and end)
- **/*, *//**Multi-line comment delimiter (begin and end)
- **--**Single-line comment indicator
- **..**Range operator**<, >, <=, >=**Relational operators**<>, '=, ~=,**

  ^=Different versions of NOT EQUAL

The PL/SQL Comments

- Program comments are explanatory statements that can be included in the PL/SQL code that you write and helps anyone reading its source code. All programming languages allow some form of comments.
- The PL/SQL supports single-line and multi-line comments. All characters available inside any comment are ignored by the PL/SQL compiler. The PL/SQL single-line comments start with the delimiter -- (double hyphen) and multi-line comments are enclosed by /* and */.
- **EX:**

PL/SQL Program Units

- A PL/SQL unit is any one of the following −
- PL/SQL block
- Function
- Package
- Package body
- Procedure
- Trigger
- Type
- Type body

- We will discuss the Data Types in PL/SQL. The PL/SQL variables, constants and parameters must have a valid data type, which specifies a storage format, constraints, and a valid range of values. We will focus on the **SCALAR** and the **LOB** data types in this chapter. The other two data types will be covere
- S.NoCategory & Description

1**Scalar**

- Single values with no internal components, such as a **NUMBER, DATE,** or **BOOLEAN**.

2**Large Object (LOB)**

- Pointers to large objects that are stored separately from other data items, such as text, graphic images, video clips, and sound waveforms.

3**Composite**

- Data items that have internal components that can be accessed individually. For example, collections and records.

4Reference

- Pointers to other data items.
- PL/SQL Scalar Data Types and Subtypes
- PL/SQL Scalar Data Types and Subtypes come under the following categories −
- S.NoDate Type & Description

1Numeric

- Numeric values on which arithmetic operations are performed.

2Character

- Alphanumeric values that represent single characters or strings of characters.

3Boolean

- Logical values on which logical operations are performed.

4Datetime

- Dates and times.
- PL/SQL provides subtypes of data types. For example, the data type NUMBER has a subtype called INTEGER. You can use the subtypes in your PL/SQL program to make the data types compatible with data types in other programs while embedding the PL/SQL code in another program, such as a Java program.

- PL/SQL Numeric Data Types and Subtypes
- Following table lists out the PL/SQL pre-defined numeric data types and their sub-types −

S.NoData Type & Description

1**PLS_INTEGER**

- Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits

2**BINARY_INTEGER**

- Signed integer in range -2,147,483,648 through 2,147,483,647, represented in 32 bits

3**BINARY_FLOAT**

- Single-precision IEEE 754-format floating-point number

4**BINARY_DOUBLE**

- Double-precision IEEE 754-format floating-point number

5**NUMBER(prec, scale)**

- Fixed-point or floating-point number with absolute value in range 1E-130 to (but not including) 1.0E126. A NUMBER variable can also represent 0

6**DEC(prec, scale)**

- ANSI specific fixed-point type with maximum precision of 38 decimal digits

7**DECIMAL(prec, scale)**

- IBM specific fixed-point type with maximum precision of 38 decimal digits

8**NUMERIC(pre, secale)**

- Floating type with maximum precision of 38 decimal digits

9**DOUBLE PRECISION**

- ANSI specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)

10**FLOAT**

- ANSI and IBM specific floating-point type with maximum precision of 126 binary digits (approximately 38 decimal digits)

11**INT**

- ANSI specific integer type with maximum precision of 38 decimal digits

12**INTEGER**

- ANSI and IBM specific integer type with maximum precision of 38 decimal digits

13**SMALLINT**

- ANSI and IBM specific integer type with maximum precision of 38 decimal digits

14**REAL**

- Floating-point type with maximum precision of 63 binary digits (approximately 18 decimal digits)
- **EX:**

- we will discuss Variables in PL/SQL. A variable is nothing but a name given to a storage area that our programs can manipulate. Each variable in PL/SQL has a specific data type, which determines the size and the layout of the variable's memory; the range of values that can be stored within that memory and the set of operations that can be applied to the variable.
- The name of a PL/SQL variable consists of a letter optionally followed by more letters, numerals, dollar signs, underscores, and number signs and should not exceed 30 characters. By default, variable names are not case-sensitive. You cannot use a reserved PL/SQL keyword as a variable name.

- PL/SQL programming language allows to define various types of variables, such as date time data types, records, collections, etc. which we will cover in subsequent chapters. For this chapter, let us study only basic variable types.

Variable Declaration in PL/SQL

- PL/SQL variables must be declared in the declaration section or in a package as a global variable. When you declare a variable, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.
- The syntax for declaring a variable is −
- variable_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial_value]

- Where, *variable_name* is a valid identifier in PL/SQL, *datatype* must be a valid PL/SQL data type or any user defined data type which we already have discussed in the last chapter. Some valid variable declarations along with their definition are shown below −
- sales number(10, 2);
- pi CONSTANT double precision := 3.1415;
- name varchar2(25);
- address varchar2(100);

- When you provide a size, scale or precision limit with the data type, it is called a **constrained declaration**. Constrained declarations require less memory than unconstrained declarations.
- For example −
- sales number(10, 2); name varchar2(25); address varchar2(100);
- Initializing Variables in PL/SQL
- Whenever you declare a variable, PL/SQL assigns it a default value of NULL. If you want to initialize a variable with a value other than the NULL value, you can do so during the declaration, using either of the following −

- The **DEFAULT** keyword
- The **assignment** operator
- For example −
- counter binary_integer := 0;
- greetings varchar2(20) DEFAULT 'Have a Good Day';
- You can also specify that a variable should not have a **NULL** value using the **NOT NULL** constraint. If you use the NOT NULL constraint, you must explicitly assign an initial value for that variable.
- It is a good programming practice to initialize variables properly otherwise, sometimes programs would produce unexpected results. Try the following example which makes use of various types of variables
- **EX:**

Variable Scope in PL/SQL

- PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block. If a variable is declared within an inner block, it is not accessible to the outer block. However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks. There are two types of variable scope −
- **Local variables** − Variables declared in an inner block and not accessible to outer blocks.
- **Global variables** − Variables declared in the outermost block or a package.
- Following example shows the usage of **Local** and **Global** variables in its simple form − **EX:**

- Assigning SQL Query Results to PL/SQL Variables
- You can use the **SELECT INTO** statement of SQL to assign values to PL/SQL variables. For each item in the **SELECT list**, there must be a corresponding, type-compatible variable in the **INTO list**. The following example illustrates the concept. Let us create a table named CUSTOMERS −
- **EX:**
- Let us now insert some values in the table −
- **EX:**
- The following program assigns values from the above table to PL/SQL variables using the **SELECT INTO clause** of SQL −
- **EX:**

PL/SQL - Constants and Literals

- we will discuss **constants** and **literals** in PL/SQL. A constant holds a value that once declared, does not change in the program. A constant declaration specifies its name, data type, and value, and allocates storage for it. The declaration can also impose the **NOT NULL constraint**.

Declaring a Constant

- A constant is declared using the **CONSTANT** keyword. It requires an initial value and does not allow that value to be changed. For example

The PL/SQL Literals

- A literal is an explicit numeric, character, string, or Boolean value not represented by an identifier. For example, TRUE, 776, NULL, ‘Internet’ are all literals of type Boolean, number, or string. PL/SQL, literals are case-sensitive. PL/SQL supports the following kinds of literals −
- Numeric Literals
- Character Literals
- String Literals
- BOOLEAN Literals
- Date and Time Literals
- The following table provides examples from all these categories of literal values.

- 1**Numeric Literals**
- 050 78 -14 0 +32767
- 6.6667 0.0 -12.0 3.14159 +7800.00
- 6E5 1.0E-8 3.14159e0 -1E38 -9.5e-3
- 2**Character Literals**
- 'A' '%' '9' ' ' 'z' '('
- 3**String Literals**
- 'Hello, world!‘
- '19-NOV-12'
- 4**BOOLEAN Literals**
- TRUE, FALSE, and NULL.

- 5**Date and Time Literals**
- DATE '1978-12-25';
- TIMESTAMP '2012-10-29 12:01:01';

EX: 2

PL/SQL - Operators

- we will discuss operators in PL/SQL. An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulation. PL/SQL language is rich in built-in operators and provides the following types of operators .
- Arithmetic operators
- Relational operators
- Comparison operators
- Logical operators
- String operators

- Here, we will understand the arithmetic, relational, comparison and logical operators one by one.
- Arithmetic Operators
- Following table shows all the arithmetic operators supported by PL/SQL. Let us assume **variable A** holds 10 and **variable B** holds 5, then −

Operator	Description	Example
+	Adds two operands	A + B will give 15
-	Subtracts second operand from the first	A - B will give 5
*	Multiplies both operands	A * B will give 50
/	Divides numerator by de-numerator	A / B will give 2
**	Exponentiation operator, raises one operand to the power of other	A ** B will give 100000

Relational Operators

Relational operators compare two expressions or values and return a Boolean result. Following table shows all the relational operators supported by PL/SQL. Let us assume **variable A** holds 10 and **variable B** holds 20, then

Operator	Description	Example
=	Checks if the values of two operands are equal or not, if yes then condition becomes true.	(A = B) is not true.
!= <> ~=	Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.	(A != B) is true.
>	Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.	(A <= B) is true

EX:4

- Comparison Operators
- Comparison operators are used for comparing one expression to another. The result is always either **TRUE, FALSE** or **NULL.** EX:5

Operator	Description	Example
LIKE	The LIKE operator compares a character, string, or CLOB value to a pattern and returns TRUE if the value matches the pattern and FALSE if it does not.	If 'Zara Ali' like 'Z% A_i' returns a Boolean true, whereas, 'Nuha Ali' like 'Z% A_i' returns a Boolean false.
BETWEEN	The BETWEEN operator tests whether a value lies in a specified range. x BETWEEN a AND b means that x >= a and x <= b.	If x = 10 then, x between 5 and 20 returns true, x between 5 and 10 returns true, but x between 11 and 20 returns false.
IN	The IN operator tests set membership. x IN (set) means that x is equal to any member of set.	If x = 'm' then, x in ('a', 'b', 'c') returns Boolean false but x in ('m', 'n', 'o') returns Boolean true.
IS NULL	The IS NULL operator returns the BOOLEAN value TRUE if its operand is NULL or FALSE if it is not NULL. Comparisons involving NULL values always yield NULL.	If x = 'm', then 'x is null' returns Boolean false.

- Logical Operators
- Following table shows the Logical operators supported by PL/SQL. All these operators work on Boolean operands and produce Boolean results. Let us assume **variable A** holds true and **variable B** holds false, then **EX: 6**

Operator	Description	Examples
and	Called the logical AND operator. If both the operands are true then condition becomes true.	(A and B) is false.
or	Called the logical OR Operator. If any of the two operands is true then condition becomes true.	(A or B) is true.
not	Called the logical NOT Operator. Used to reverse the logical state of its operand. If a condition is true then Logical NOT operator will make it false.	not (A and B) is true.

- PL/SQL Operator Precedence
- Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator.
- For example, **x = 7 + 3 * 2**; here, **x** is assigned **13**, not 20 because operator * has higher precedence than +, so it first gets multiplied with **3*2** and then adds into **7**.

- Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.
- The precedence of operators goes as follows: =, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN.

Operator	Operation
**	exponentiation
+, -	identity, negation
*, /	multiplication, division
+, -, ||	addition, subtraction, concatenation
comparison	
NOT	logical negation
AND	conjunction
OR	inclusion

XML Basics

XML - Schemas XML Schema Definition (XSD).

- XML stands for eXtensible Markup Language.
- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.

```
<?xml version="1.0" encoding="UTF-8"?>
    <note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
    </note>
```

The Difference Between XML and HTML

- XML and HTML were designed with different goals:
- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML Does Not Use Predefined Tags

- The XML language has no predefined tags.
- The tags in the example above (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.
- HTML works with predefined tags like <p>, <h1>, <table>, etc.
- With XML, the author must define both the tags and the document structure.

XML is Extensible

- Most XML applications will work as expected even if new data is added (or removed).
- Imagine an application designed to display the original version of note.xml (<to> <from> <heading> <body>).
- Then imagine a newer version of note.xml with added <date> and <hour> elements, and a removed <heading>.
- The way XML is constructed, older version of the application can still work:

How can XML be used?

- XML is used in many aspects of web development.
- XML is often used to separate data from presentation.

XML Separates Data from Presentation

- XML does not carry any information about how to be displayed.
- The same XML data can be used in many different presentation scenarios.
- Because of this, with XML, there is a full separation between data and presentation.

XML is Often a Complement to HTML

- In many HTML applications, XML is used to store or transport data, while HTML is used to format and display the same data.

XML Separates Data from HTML

- When displaying data in HTML, you should not have to edit the HTML file when the data changes.
- With XML, the data can be stored in separate XML files.
- With a few lines of JavaScript code, you can read an XML file and update the data content of any HTML page.

The XML Tree Structure

XML Syntax Rules

- The syntax rules of XML are very simple and logical. The rules are easy to learn, and easy to use.
- XML Documents Must Have a Root Element
- XML documents must contain one **root** element that is the **parent** of all other elements:
- <root>
  <child>
  <subchild>.....</subchild>
  </child>
  </root>
- In this example **<note>** is the root element:
- <?xml version="1.0" encoding="UTF-8"?>
  <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>

- The XML Prolog
- This line is called the XML **prolog**:
- <?xml version="1.0" encoding="UTF-8"?>
- All XML Elements Must Have a Closing Tag
- In XML, it is illegal to omit the closing tag. All elements **must** have a closing tag:
- This is a paragraph.

- XML Tags are Case Sensitive
- XML tags are case sensitive. The tag <Letter> is different from the tag <letter>.
- Opening and closing tags must be written with the same case:
- <message>This is correct</message>

- "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.
- XML Elements Must be Properly Nested
- In HTML, you might see improperly nested elements:
- This text is bold and italic
- In XML, all elements **must** be properly nested within each other:
- This text is bold and italic

- "Opening and closing tags" are often referred to as "Start and end tags". Use whatever you prefer. It is exactly the same thing.
- XML Elements Must be Properly Nested
- In HTML, you might see improperly nested elements:
- This text is bold and italic
- In XML, all elements **must** be properly nested within each other:
- This text is bold and italic

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- Name Conflicts
- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.
- This XML carries HTML table information:
- <table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>

- This XML carries information about a table (a piece of furniture):
- <table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
  </table>
- If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.
- A user or an XML application will not know how to handle these differences.

- This XML carries information about a table (a piece of furniture):
- <table>
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
  </table>
- If these XML fragments were added together, there would be a name conflict. Both contain a <table> element, but the elements have different content and meaning.
- A user or an XML application will not know how to handle these differences.
- Solving the Name Conflict Using a Prefix
- Name conflicts in XML can easily be avoided using a name prefix.
- This XML carries information about an HTML table, and a piece of furniture:

```
<h:table>
    <h:tr>
      <h:td>Apples</h:td>
      <h:td>Bananas</h:td>
    </h:tr>
  </h:table>

  <f:table>
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
```

•XML Namespaces - The xmlns Attribute

•When using prefixes in XML, a **namespace** for the prefix must be defined.

•The namespace can be defined by an **xmlns** attribute in the start tag of an element.

•The namespace declaration has the following syntax. xmlns:*prefix*="*URI*".

•<root>

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

•<f:table xmlns:f="https://www.w3schools.com/furniture">
<f:name>African Coffee Table</f:name>
<f:width>80</f:width>
<f:length>120</f:length>
</f:table>
</root>

- In the example above:
- The xmlns attribute in the first <table> element gives the h: prefix a qualified namespace.
- The xmlns attribute in the second <table> element gives the f: prefix a qualified namespace.
- When a namespace is defined for an element, all child elements with the same prefix are associated with the same namespace.

•Namespaces can also be declared in the XML root element:

•<root xmlns:h="http://www.w3.org/TR/html4/" xmlns:f="https://www.w3schools.com/furniture">

•<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
</root>

•**Note:** The namespace URI is not used by the parser to look up information.

•The purpose of using an URI is to give the namespace a unique name.

•However, companies often use the namespace as a pointer to a web page containing namespace information.

- Uniform Resource Identifier (URI)
- A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource.
- The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address. Another, not so common type of URI is the **Uniform Resource Name** (URN).
- Default Namespaces
- Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:
- xmlns="*namespaceURI*"

- This XML carries HTML table information:
- <table xmlns="http://www.w3.org/TR/html4/">
  <tr>
  <td>Apples</td>
  <td>Bananas</td>
  </tr>
  </table>
- This XML carries information about a piece of furniture:
- <table xmlns="https://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
  </table>

- This XML carries HTML table information:
- <table xmlns="http://www.w3.org/TR/html4/">
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
</table>
- This XML carries information about a piece of furniture:
- <table xmlns="https://www.w3schools.com/furniture">
  <name>African Coffee Table</name>
  <width>80</width>
  <length>120</length>
</table>

- Displaying XML
- ❮ PreviousNext ❯
- Raw XML files can be viewed in all major browsers.
- Don't expect XML files to be displayed as HTML pages.
- Viewing XML Files
- <?xml version="1.0" encoding="UTF-8"?>
  - <note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>
- Look at the XML file above in your browser: note.xml
- Most browsers will display an XML document with color-coded elements.

- Often a plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure.
- To view raw XML source, try to select "View Page Source" or "View Source" from the browser menu.
- **Note:** In Safari 5 (and earlier), only the element text will be displayed. To view the raw XML, you must right click the page and select "View Source".
- Viewing an Invalid XML File
- If an erroneous XML file is opened, some browsers will report the error, and some will display it, or display it incorrectly.
- <?xml version="1.0" encoding="UTF-8"?>
  - <note>
  <to>Tove</to>
  <From>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
  </note>

- XML Parser
- ❮ PreviousNext ❯
- All major browsers have a built-in XML parser to access and manipulate XML.
- XML Parser
- The XML DOM (Document Object Model) defines the properties and methods for accessing and editing XML.
- However, before an XML document can be accessed, it must be loaded into an XML DOM object.
- All modern browsers have a built-in XML parser that can convert text into an XML DOM object.
- Parsing a Text String
- This example parses a text string into an XML DOM object, and extracts the info from it with JavaScript:

XML DOM

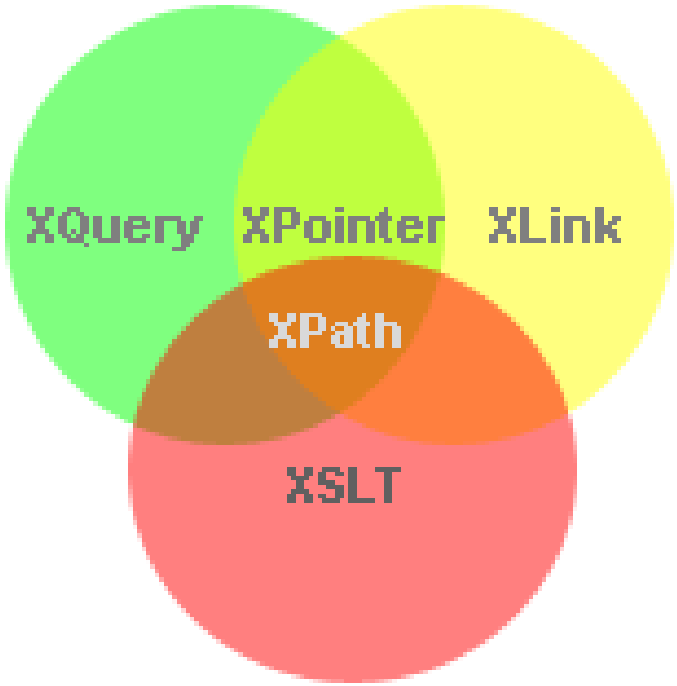
- What is the DOM?
- The Document Object Model (DOM) defines a standard for accessing and manipulating documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- The **HTML DOM** defines a standard way for accessing and manipulating HTML documents. It presents an HTML document as a tree-structure.
- The **XML DOM** defines a standard way for accessing and manipulating XML documents. It presents an XML document as a tree-structure.

- Understanding the DOM is a must for anyone working with HTML or XML.
- The HTML DOM
- All HTML elements can be accessed through the HTML DOM.
- This example changes the value of an HTML element with id="demo":

Ex:

What is XPath?

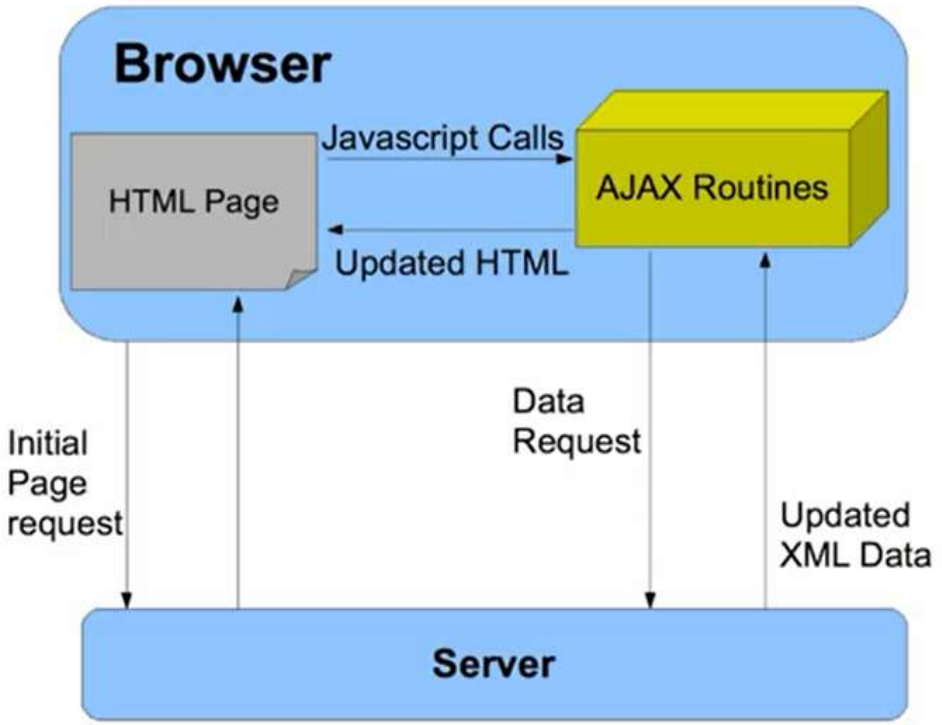
- XPath is a major element in the XSLT standard.
- XPath can be used to navigate through elements and attributes in an XML document.
- XPath is a syntax for defining parts of an XML document
- XPath uses path expressions to navigate in XML documents
- XPath contains a library of standard functions
- XPath is a major element in XSLT and in XQuery
- XPath is a W3C recommendation

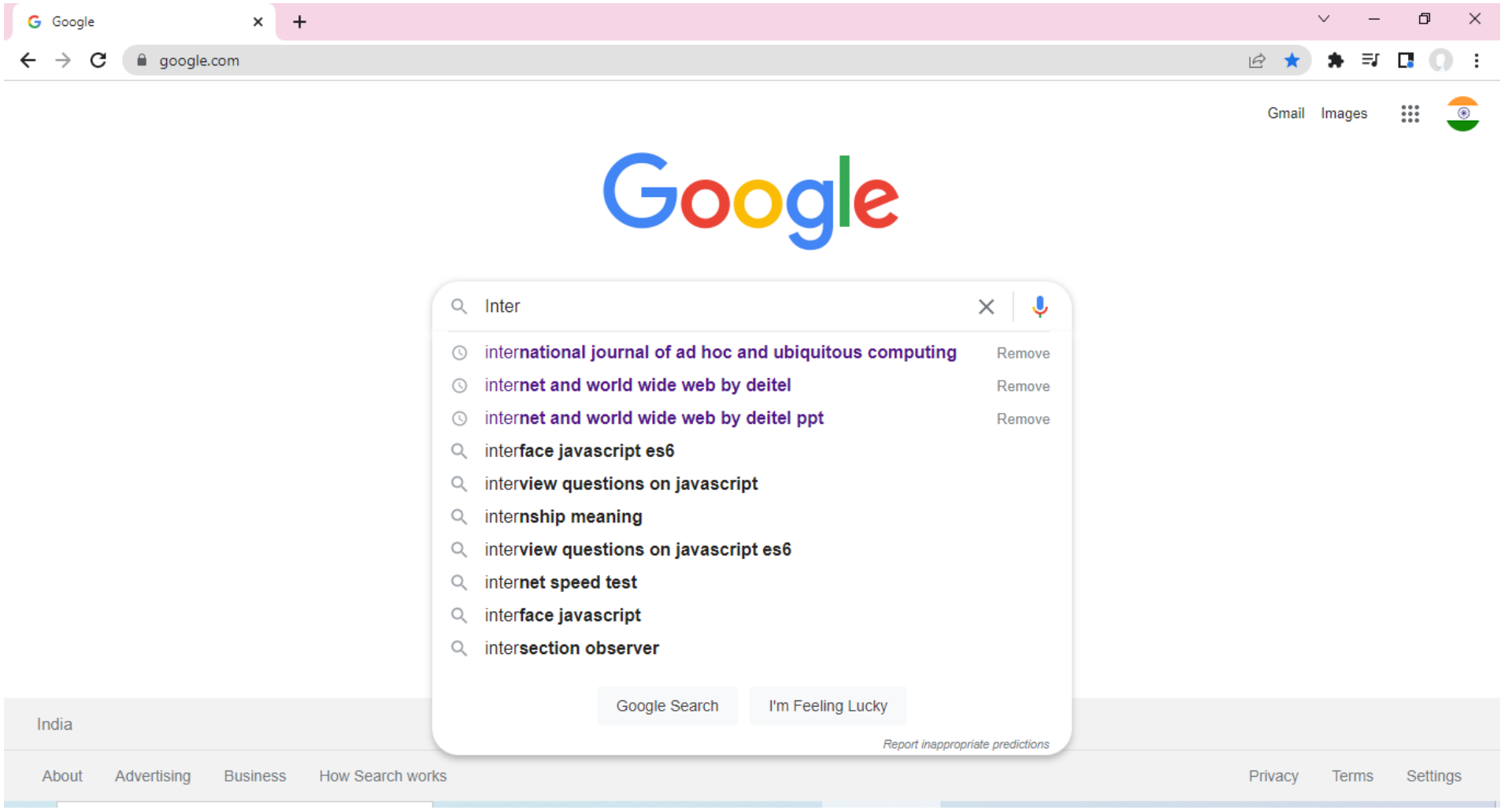


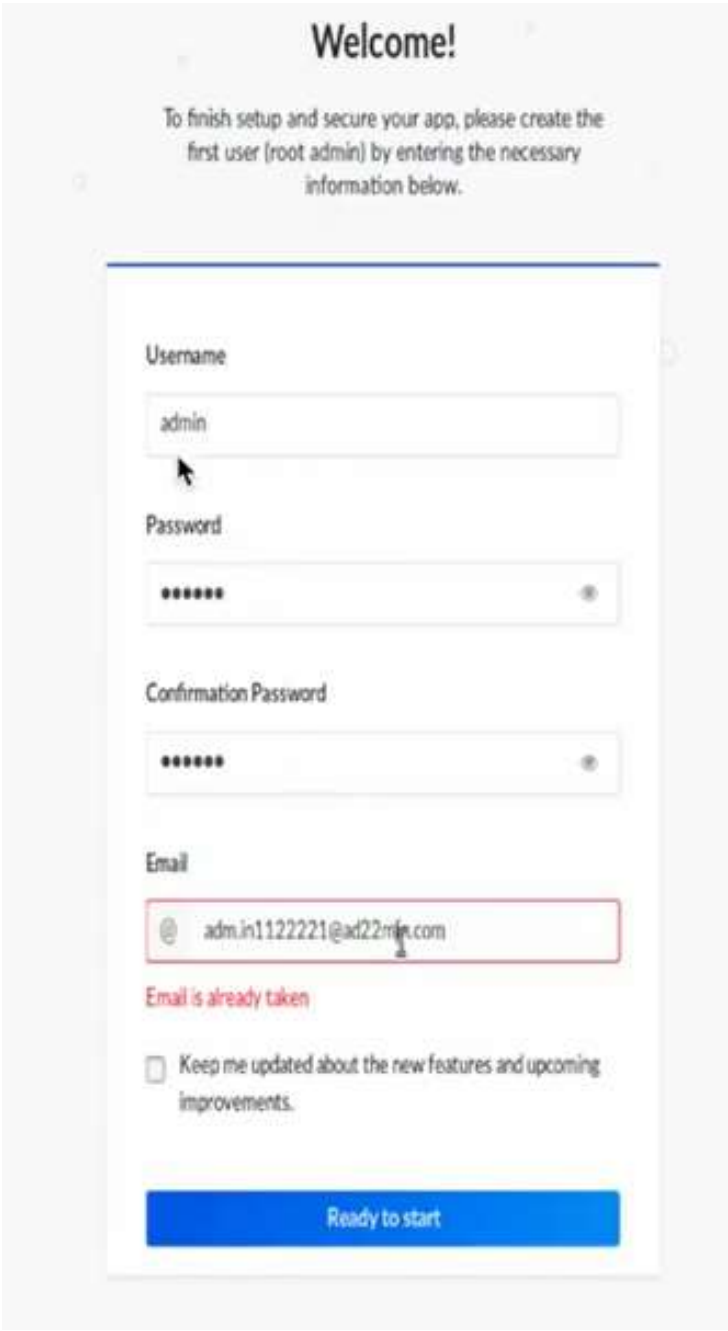
- XPath Path Expressions
- XPath uses path expressions to select nodes or node-sets in an XML document. These path expressions look very much like the expressions you see when you work with a traditional computer file system.
- XPath expressions can be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages.
- XPath is Used in XSLT
- XPath is a major element in the XSLT standard.
- With XPath knowledge you will be able to take great advantage of XSL.

AJAX

- AJAX stands for **A**synchronous **J**ava**S**cript and **X**ML.
- AJAX is a new technique for creating better, faster, and more interactive web applications
- XMLHttpRequest object that performs asynchronous interaction with the server.




Google
google.com
Gmail
Images
Google
Inter
international journal of ad hoc and ubiquitous computing
Remove
internet and world wide web by deitel
Remove
internet and world wide web by deitel ppt
Remove
interface javascript es6
interview questions on javascript
internship meaning
interview questions on javascript es6
internet speed test
interface javascript
intersection observer
Google Search
I'm Feeling Lucky
Report inappropriate predictions
India
About
Advertising
Business
How Search works
Privacy
Terms
Settings


Welcome!
To finish setup and secure your app, please create the first user (root admin) by entering the necessary information below.
Username
admin
Password
Confirmation Password
Email
adm.in1122221@ad22m
.com
Email is already taken
Keep me updated about the new features and upcoming improvements.
Ready to start


Love
Like
Comment
Share
92
Top Comments

AJAX Introduction

- AJAX is a developer's dream, because you can:
- Update a web page without reloading the page
- Request data from a server - after the page has loaded
- Receive data from a server - after the page has loaded
- Send data to a server - in the background

AJAX Example

- HTML Page
- <!DOCTYPE html>
  <html>
  <body>

  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change Content</button>

  </body>
  </html>

- The HTML page contains a section and a <button>.
- The section is used to display information from a server.
- The <button> calls a function (if it is clicked).
- The function requests data from a web server and displays it:
- Function loadDoc()
- function loadDoc() {

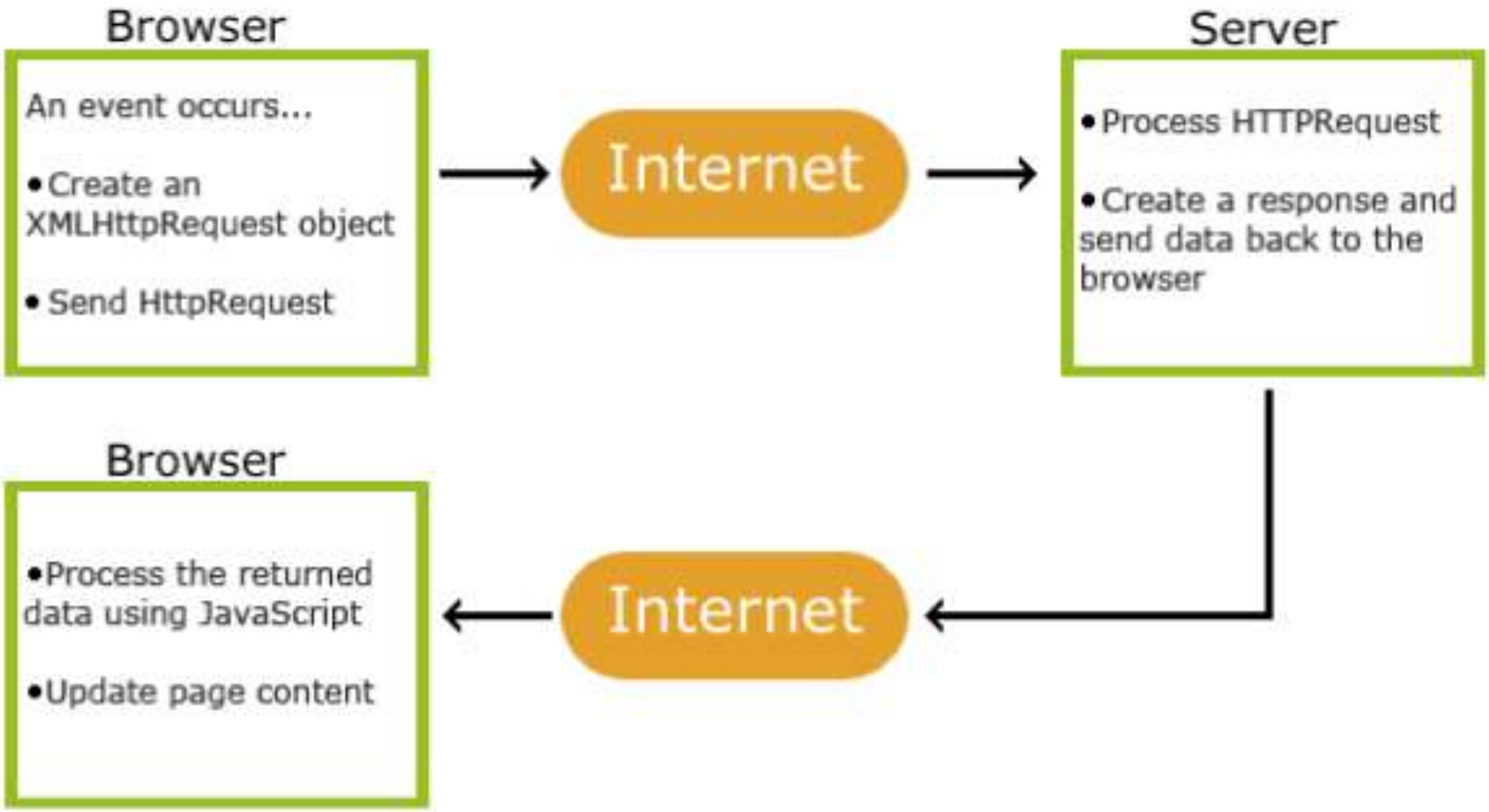
```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
  if (this.readyState == 4 && this.status == 200) {
   document.getElementById("demo").innerHTML = this.responseText;
  }
};
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();    }
```

The "ajax_info.txt" file used in the example above, is a simple text file and looks like this:

Subash Chandra Bose

- What is AJAX?
- AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is not a programming language.
- AJAX just uses a combination of:
- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

- AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Internet
Browser
• Process the returned data using JavaScript
• Update page content

- 1. An event occurs in a web page (the page is loaded, a button is clicked)
- 2. An XMLHttpRequest object is created by JavaScript
- 3. The XMLHttpRequest object sends a request to a web server
- 4. The server processes the request
- 5. The server sends a response back to the web page
- 6. The response is read by JavaScript
- 7. Proper action (like page update) is performed by JavaScript

- The keystone of AJAX is the XMLHttpRequest object.
- The XMLHttpRequest Object
- All modern browsers support the XMLHttpRequest object.
- The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.
- Create an XMLHttpRequest Object
- All modern browsers (Chrome, Firefox, Edge (and IE7+), Safari, Opera) have a built-in XMLHttpRequest object.
- Syntax for creating an XMLHttpRequest object:
- *variable* = new XMLHttpRequest();

- var xhttp = new XMLHttpRequest();
- Try it Yourself »
- The "ajax_info.txt" file used in the example above, is a simple text file and looks like this:

Subash Chandra Bose

- Access Across Domains
- For security reasons, modern browsers do not allow access across domains.
- This means that both the web page and the XML file it tries to load, must be located on the same server.
- The examples on W3Schools all open XML files located on the W3Schools domain.

- If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.