INTRODUCTION NODE JS

Server side Scripting language

WHAT IS NODE JS

- Server Side JavaScript Runtime
- Features:
  - Non-Blocking I/O
  - Asynchronous event driven architecture
  - Single threaded but scalable
  - V8 JavaScript Engine
  - Cross platform support

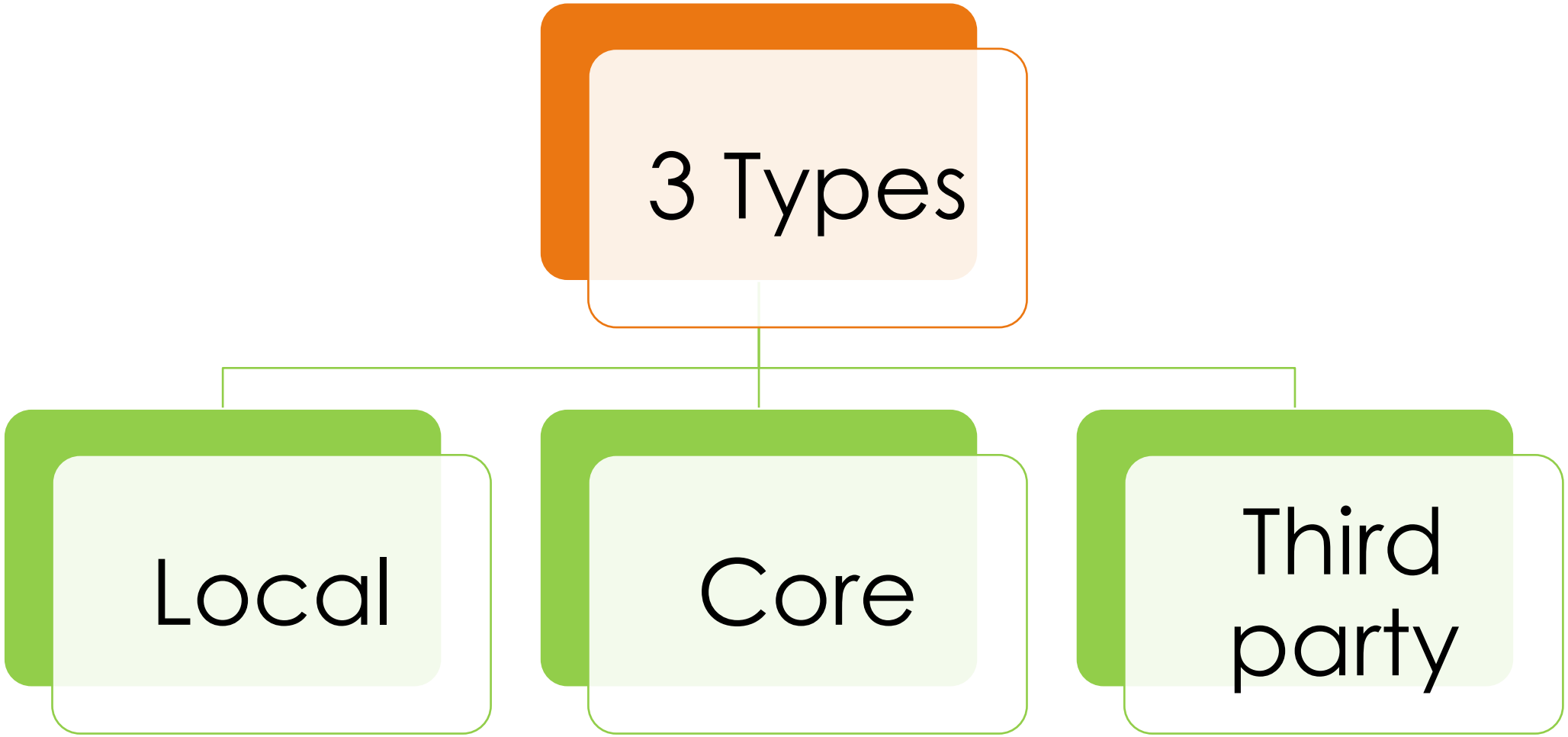
WHY NODE?

- Fast Execution
- Efficient for I/O operations
- Large Ecosystem
- npm packages
- Community support

NODE REPL

- Read Evaluate Print Loop
- How to start: node and press enter
- You will be inside node terminal where you can directly execute all js lines of code.
- You can also create .js file and run directly using node command.

NODE MODULES



CORE MODULES IN NODEJS

- Modules comes as an inbuilt modules in JS called as Core modules
- Fs (file system)
- http (HTTP)
- url (URL)
- path
- os

URL MODULE

- Parsing and formatting URL
- From parsed object retrive
  - Path
  - Hostname
  - Port
  - Protocol
  - Query etc..
- Extracting query parameters

FS MODULE

- It is used to execute file operations
- CRUD opeartions
  - Create: fs.writeFile()
  - Read: fs.readFile()
  - Update : fs.appendFile()
  - Delete: fs.unlink()

UNDERSTANDING SYNC AND ASYNC OPERATIONS

- Use FileSystem nodules functions
  - Fs.writeFileSync()
  - Fs.readFileSync()
  - Fs.appendFileSync()
  - Fs.unlinkSync()

HTTP MODULE

- For creating http server
- Handling multiple requests
- Providing responses in different formats
  - Text
  - Json
  - Html

PACKAGE.JSON

- It's a configuration file of node projects
- You can configure project name, version, author name, policies, dependencies, dev dependencies etc.
- How to create: 2 ways
- Npm init (prompts for some questions and create)
- Npm init –y (direct creates)

NODE PACKAGES

- How to install third party node packages
- Using npm install
  - Global installation
  - Local installation
  - Development dependency
- How to uninstall dependencies
- Using uninstall command
  - uninstall locally
  - Uninstall globally

HTTP SERVER

- The HTTP module in Node.js allows you to create a web server that can listen to client requests and respond accordingly.
- Let’s create one folder: mkdir my-server
- Move to the folder: cd my-server
- Create package.json file: npm init –y
- Create the server code in file named index.js
- Save the file
- Execute the file: node index.js

PROVIDE HTML FILE AS RESPONSE

- Read the file data using filesystem module
- Show that data in http server response to provide the response.

INTRO TO EXPRESS

- Fast, unopinionated, minimalist web framework for Node.js
- Using express we can create:
  - Web application
  - Web APIs
  - Middlewares
- How to install express?
  - Npm install express

LET'S CREATE ONE SERVER USING EXPRESS

```
const express = require('express')
const app = express()
const port = 3000

app.get('/', (req, res) => {
  res.send('Hello World!')
})

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

CREATING APIS USING EXPRESS

- Get method
- Post method
- Put method
- Delete method
- Check them using postman

ACTIVITY 1: FILESYSTEM ASSIGNMENT

- You are tasked with building a command-line tool named "FileOps" that performs various file operations using the Node.js filesystem module.
- **Reading Files**
  - Create a Node.js script named fileops.js
  - Implement a function named readFile that reads the contents of a text file specified by the user.
  - If the file exists, print its contents to the console.
  - If the file does not exist, handle the error gracefully and display an appropriate message.
- **Writing to Files**
  - Implement a function named writefile that writes a user-specified message to a text file.
  - If the file already exists, overwrite its contents with the new message.
  - If the file does not exist, create a new file with the specified message.

ACTIVITY 1: FILESYSTEM ASSIGNMENT

- **Copying Files**
  - Implement a function named copyfile that copies the contents of one text file to another.
  - Prompt the user for the source and destination file names.
  - Verify that the source file exists before proceeding with the copy operation.
- **Listing Files in a Directory**
  - Implement a function named listfiles that lists all files in a specified directory.
  - Allow the user to input the directory path.
- **Creating and Removing Directories**
  - Implement functions named createDirectory and removeDirectory
  - Allow the user to create a new directory and specify the path.
  - Allow the user to remove an existing directory and its contents.

ACTIVITY 2: USER AUTHENTICATION MODULE

- Create a custom module named authModule that provides functions for user authentication:
- Register User:
  - Implement a function registerUser that takes a username and password as parameters and registers a new user. Store the user information securely, considering basic security practices.
- Login User:
  - Create a function loginUser that takes a username and password as parameters and checks if the provided credentials are valid. Return a success message if the login is successful, and handle incorrect credentials.
- Change Password:
  - Implement a function changePassword that allows a logged-in user to change their password. The function should take the username, current password, and new password as parameters.
- Logout User:
  - Create a function logoutUser that logs out the currently logged-in user.
- Once you have created the authModule module, use it in a separate file (index.js) to perform user authentication operations. (use import or require to use it in separate file)

ACTIVITY 3: PRODUCT INVENTORY MANAGEMENT

- Create a express application which implements followings APIs
- **Add Product:**
  - Implement a function addProduct that takes the product name, price, and quantity as parameters and inserts a new product into the products array.
- **View Product Information:**
  - Create a function viewProduct that takes a product name as a parameter and retrieves information about the product from the products array.
- **Update Product Quantity:**
  - Implement a function updateProductQuantity that allows updating the quantity of a specific product. The function should take the product name and the new quantity as parameters.
- **List All Products:**
  - Create a function listAllProducts that retrieves and displays the names and quantities of all products in the inventory.