Written Assignment 2

Questions:

Q1) What is Refs? When to Use Refs And when not to use Refs?

- Refs is the shorthand use of References in React.
- It is very similar to keys in React.
- It is an attribute which makes it possible to store a reference to particular DOM or react elements.
- It Is used when we want to change the value of a child component without making the use of props.
- Refs can be created using *React.createRef()* function and attached it with the React element via the *ref* attribute.
- When a component is constructed the Refs are commonly assigned to an instance property so that they can be referenced in the component. -
Code: Creation of React Refs

```
class MyComponent extends React.Component {
 constructor(props) {
 super(props);
 this.myCallRef = React.createRef();
 }
 render() {
 return <div ref={this.myCallRef} />;
 }
}
```

- Accessing Refs
- In React, A reference to the node becomes accessible at the *current* attribute of the *ref* when a *ref* is passed to an element in the *render*.
  - **When to Use Refs?**
- Managing focus, text selection or media playback
- Triggering imperative animations
- Integrating with third party dom libraries.
  - **When to not Use Refs?**

- Its use should be avoided for anything that can be done effectively. - You should have to avoid overuse of Refs.

Q2) Write a short note on

a) NPM

- **npm** is a short form of ***Node Package Manager,*** which is the world's largest software registry. The open-source web project developers use it from the entire world to **share** and **borrow** packages. The npm also acts as a command-line utility for the Node.js project for installing packages in the project, dependency management, and even version management.
- **Components of NPM**

npm mainly consists of three different components, these are:

- **Website:** The npm official website is used to find packages for your project, create and set up profiles to manage and access private and public packages. - **Command Line Interface (CLI):** The CLI runs from your computer's terminal to interact with npm packages and repositories.
- **Registry:** The registry is a large and public database of JavaScript projects and meta-information. You can use any supported npm registry that you want or even your own. You can even use someone else's registry as per their terms of use.

npm comes with **Node.js,** which means you have to install Node.js to get installed automatically on your personal computer. There are two different methods to install npm on a computer.

- **Using nvm (Node Version Manager)**
- **Using Node installer**

b) REPL

- **REPL (READ, EVAL, PRINT, LOOP)** is a computer environment similar to Shell (Unix/Linux) and command prompt. Node comes with the

REPL environment when it is installed. System interacts with the user through outputs of commands/expressions used. It is useful in writing and debugging the codes. The work of REPL can be understood from its full form:

- **Read** : It reads the inputs from users and parses it into JavaScript data structure. It is then stored to memory.
  Eval : The parsed JavaScript data structure is evaluated for the results.
  Print : The result is printed after the evaluation.
  Loop : Loops the input command. To come out of NODE REPL, press **ctrl+c** twice

Q3) Explain Routing in Express JS with an Example.

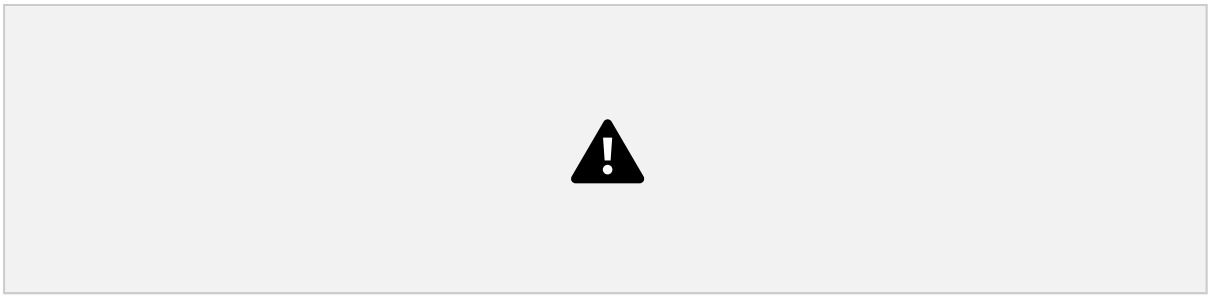
Routing is made from the word route. It is used to determine the specific behavior of an application. It specifies how an application responds to a client request to a particular route, URI or path and a specific HTTP request method (GET, POST, etc.). It can handle different types of HTTP requests.

Let's take an example to see basic routing.

File: routing_example.js

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
 console.log("Got a GET request for the homepage");
 res.send('Welcome to JavaTpoint!');
})
app.post('/', function (req, res) {
 console.log("Got a POST request for the homepage");
 res.send('I am Impossible! ');
})
app.delete('/del_student', function (req, res) {
 console.log("Got a DELETE request for /del_student");
 res.send('I am Deleted!');
})
app.get('/enrolled_student', function (req, res) {
```

```
 console.log("Got a GET request for /enrolled_student");
 res.send('I am an enrolled student.');
})
// This responds a GET request for abcd, abxcd, ab123cd, and so
on  app.get('/ab*cd', function(req, res) {
 console.log("Got a GET request for /ab*cd");
 res.send('Pattern Matched.');
})
var server = app.listen(8000, function () {
var host = server.address().address
 var port = server.address().port
console.log("Example app listening at http://%s:%s", host, port)
})
```



You see that server is listening.

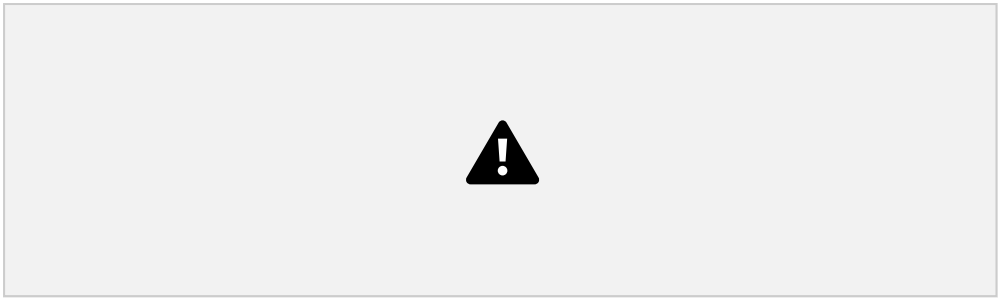
PlayNext
Unmute

Loaded: 6.60%
Â
Fullscreen

Now, you can see the result generated by server on the local host http://127.0.0.1:8000

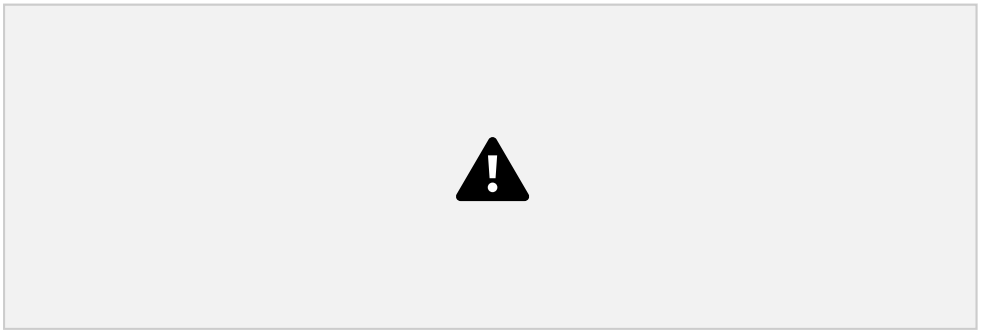
Output:

This is the homepage of the example app.

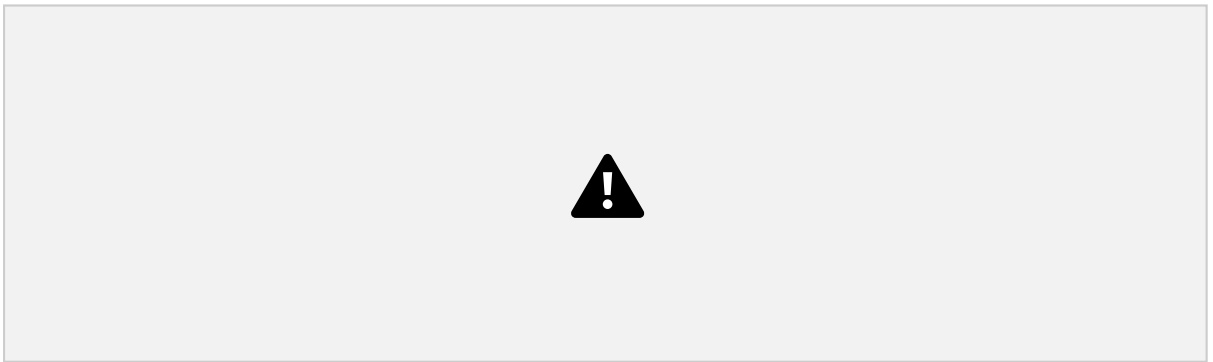


Note: The Command Prompt will be updated after one successful response.

You can see the different pages by changing routes.
http://127.0.0.1:8000/enrolled_student

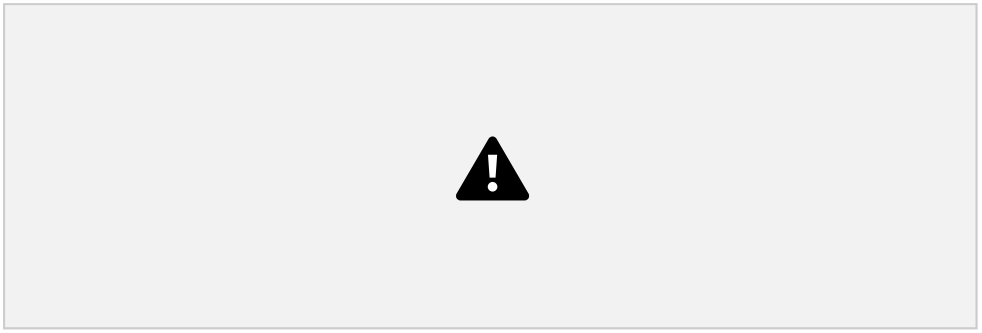


Updated command prompt:

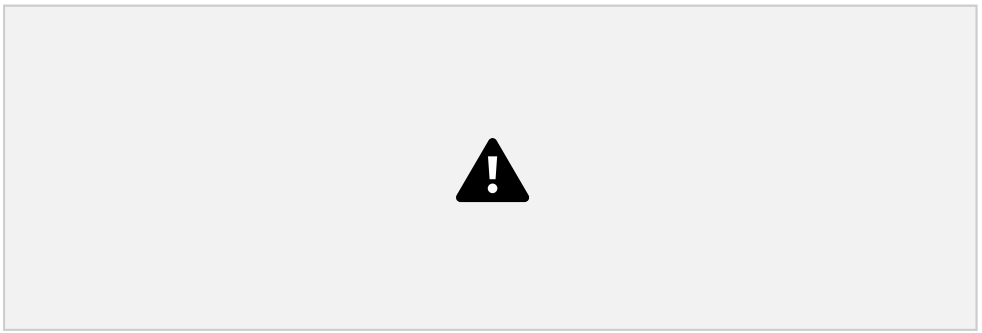


This can read the pattern like abcd, abxcd, ab123cd, and so on.

Next route **http://127.0.0.1:8000/abcd**



Next route http://127.0.0.1:8000/ab12345cd



Updated command prompt: