Response object in Node.js

The object res represents the HTTP Response that the Express application sends when receiving an HTTP Request.

The object res represents the HTTP Response that the Express application sends when receiving an HTTP Request.

Properties of the Response object in Node.js

The following table lists some properties of the Response object in Node.js.

Stt Properties & Description1 **res.app**
This property holds a reference to the Express application that is using Middleware. 2 **res.headersSent**
Value true, false to indicate whether the application has sent the HTTP Header for Response. 3 **res.locals**
An object contains local variables of Response in the Request range.

Method of Response object in Node.js

Method res.append (field [, value])

```
res . append ( field  [,  value ])
```

This method appends a specific value to the HTTP Header of Response field. For example:

```
res . append ( 'Link' , [ '  ' , '  ' ]);  res . append ( 'Set-Cookie' , 'foo=ba
```

Res.attachment ([filename]) method

```
res . attachment ([ filename ])
```

This method is used to send a File as an Attachment in the HTTP Response. For example:

```
res . attachment ( 'path/to/logo.png' );
```

Res.cookie method (name, value [, options])

```
res . cookie ( name ,  value  [,  options ])
```

This method is used to set the name of Cookie to value. The value parameter can be a string or an object converted to JSON. For example:

```
res . cookie ( 'name' , 'tobi' , {  domain : '.example.com' ,  path : '/admin' ,
```

Res.clearCookie method (name [, options])

```
res . clearCookie ( name  [,  options ])
```

This method is used to delete specific cookies specified by the name parameter. For example:

```
res . cookie ( 'name' , 'tobi' , {  path : '/admin' });  res . clearCookie ( 'na
```

Method res.download (path [, filename] [, fn])

```
res . download ( path  [,  filename ] [,  fn ])
```

This method is used to transfer the file in the given path as an Attachment. In general, the browser will prompt the user to download. For example:

```
res . download ( '/report-12345.pdf' );  res . download ( '/report-12345.pdf' ,
```

Res.end ([data] [, encoding] method)

```
res . end ([ data ] [,  encoding ])
```

This method is used to end the Response process. For example:

```
res . end ();  res . status ( 404 ). end ();
```

Res.format (object) method

```
res . format ( object )
```

This method is used to execute the Accept HTTP Header section format on the Request object when performing. For example:

```
res . format ({ 'text/plain' : function (){  res . send ( 'hey' ); }, 'text/htm
```

hey

```
' ); }, 'application/json' : function (){  res . send ({  message : 'hey' }); },
```

Res.get method (field)

```
res . get ( field )
```

This method is used to return a certain field in the HTTP Response Header. For example:

```
res . get ( 'Content-Type' );
```

Res.json method ([body])

```
res . json ([ body ])
```

This method is used to send a response in JSON format. For example:

```
res . json ( null )  res . json ({  user : 'tobi' })  res . status ( 500 ). jso
```

Res.jsonp method ([body])

```
res.jsonp ([body])
```

This method is used to send a response in JSON format with the help of JSONP. For example:

```
res.jsonp (null)
res.jsonp ({user: 'tobi'})
res.status (500) .jsonp ({error: 'message'})
```

Res.location method (path)

```
res.location (path)
```

This method is used to set the Location field of the HTTP Header based on the given path parameter. For example:

```
res.location ('/ foo / bar');
res.location ('foo / bar');
res.location ('http://example.com');
```

Res.redirect method ([status,] path)

```
res.redirect ([status,] path)
```

This method is used to redirect to a URL from a specific path path, with a certain Status Code. For example:

```
res.redirect ('/ foo / bar');
res.redirect ('http://example.com');
res.redirect (301, 'http://example.com');
```

Res.render (view [, locals] [, callback] methods)

```
res.render (view [, locals] [, callback])
```

This method is used to render a view and send the rendered HTML string to the Client. For example:

```
// Guess a view to client
res.render ('index');

// Access a screen to view
res.render ('user', {name: 'Tobi'}, function (err, html) {
// .
});
```

Method res.send ([body])

```
res.send ([body])
```

This method is used to send HTTP Response. For example:

```
res.send (new Buffer ('whoop'));
res.send ({some: 'json'});
res.send ('
some html ');
```

Res.sendFile method (path [, options] [, fn])

```
res.sendFile (path [, options] [, fn])
```

This method is used to transmit the file at the given path address. Set Content-Type based on the filename extension. For example:

```
res.sendFile (fileName, options, function (err) {
// .
});
```

Res.sendStatus method (statusCode)

```
res.sendStatus (statusCode)
```

This method is used to set the Status Code and send its string representation as a body of Response. For example:

```
res.sendStatus (200);  // tuong duong phuong res.status (200) .send ('OK')
res.sendStatus (403);  // tuong duong phuong res.status (403) .send ('Forbidde
res.sendStatus (404);  // tuong duong phuong res.status (404) .send ('Not Foun
res.sendStatus (500);  // In the same way, res.status (500) .send ('Internal S
```

Res.set method (field [, value])

```
res.set (field [, value])
```

This method is used to set the field field of the HTTP Header to the value value. For example:

```
res.set ('Content-Type', 'text / plain');

res.set ({
'Content-Type': 'text / plain',
'Content-Length': '123',
'ETag': '12345'
})
```

Res.status (code)

```
res.status (code)
```

This method is used to set HTTP Status for Response. For example:

```
res.status (403) .end ();
res.status (400) .send ('Bad Request');
res.status (404) .sendFile ('/ absolute / path / to / 404.png');
```

Res.type (type) method

```
res.type (type)
```

This method is used to set the Content-Type of the HTTP Header to the MIME type type. For example:

```
res.type ('. html');  // => 'text / html'
res.type ('html');  // => 'text / html'
res.type ('json');  // => 'application / json'
res.type ('application / json');  // => 'application / json'
res.type ('png');  // => image / png:
```

According to Tutorialspoint

Previous post: Request object in Node.js