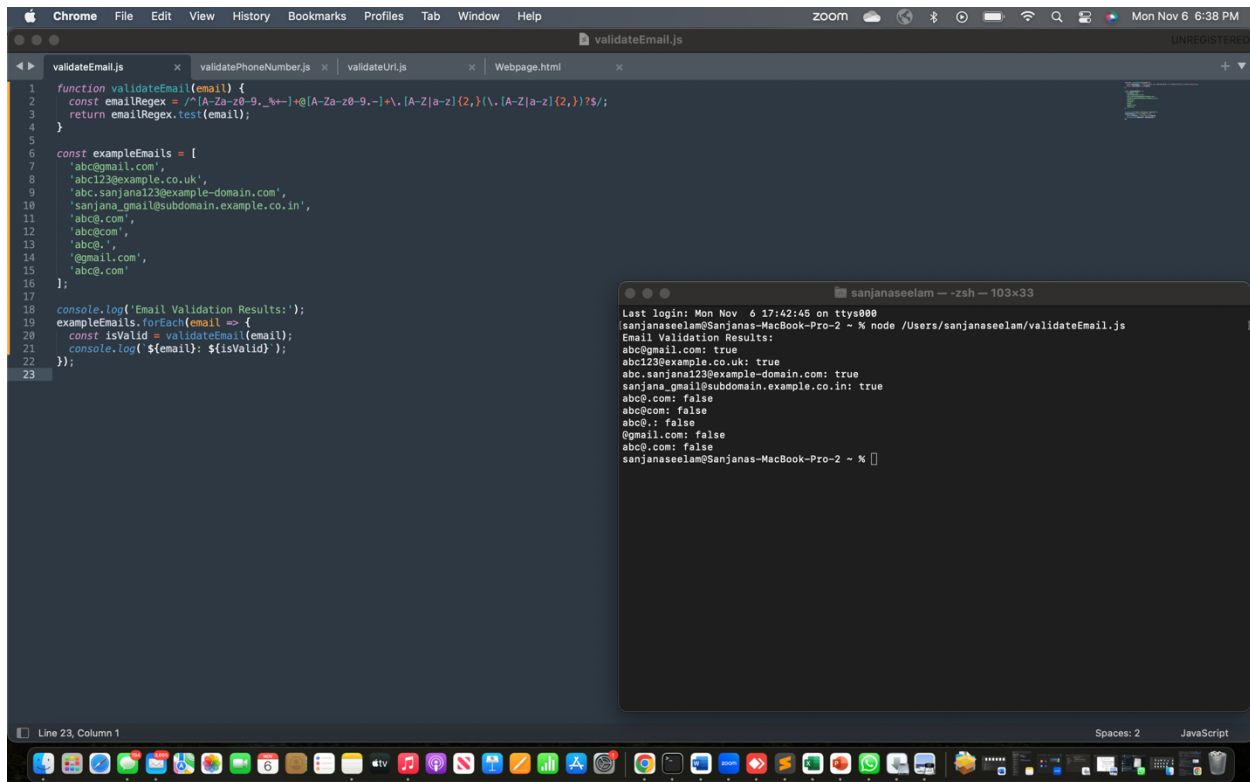


## Output Screenshots:

### Email Validation:



The screenshot shows a web browser with a JavaScript file named `validateEmail.js` open. The script defines a function `validateEmail(email)` that uses a regular expression to check if an email is valid. It then tests this function against a list of example email addresses. A terminal window is open in the background, showing the output of the script.

```
function validateEmail(email) {
  const emailRegex = /^[A-Za-z0-9_!+@][A-Za-z0-9-!+\\. [A-Z][a-z]{2,}\\.[A-Z][a-z]{2,})?$/;
  return emailRegex.test(email);
}

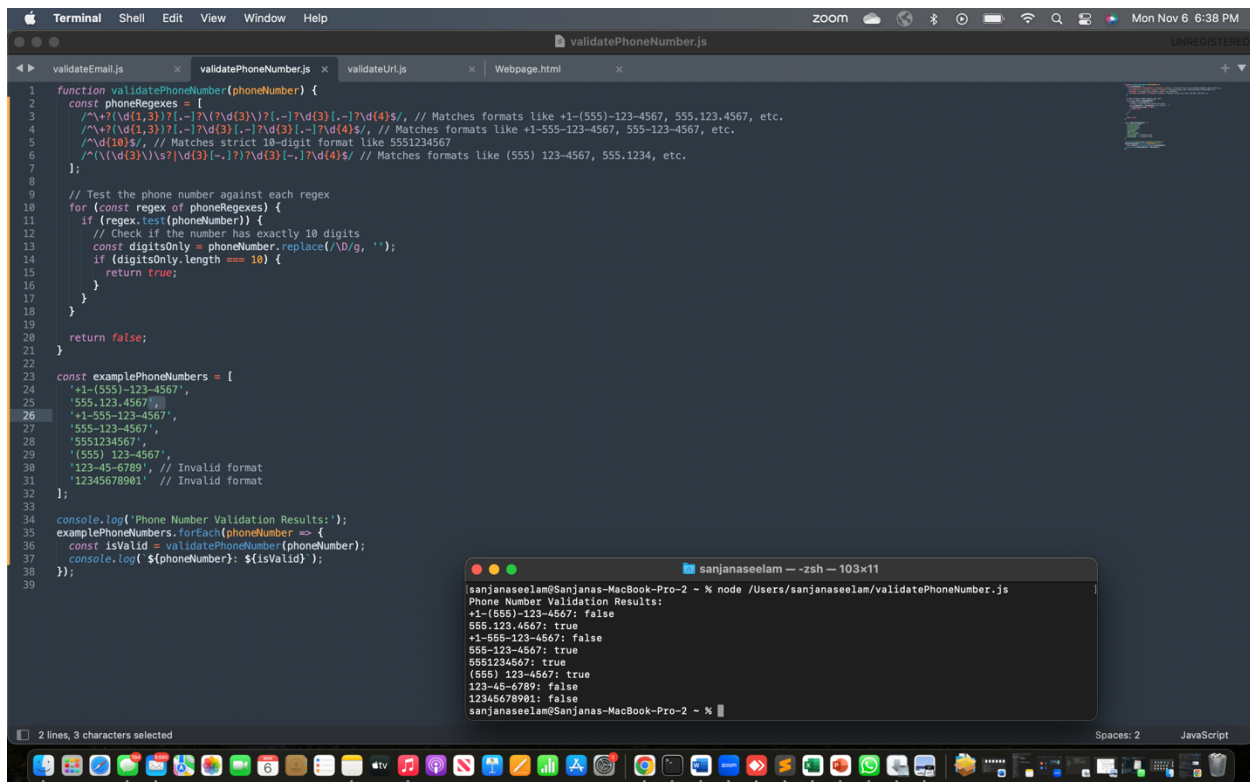
const exampleEmails = [
  'abc@gmail.com',
  'abc123@example.co.uk',
  'abc.sanjana123@example-domain.com',
  'sanjana_gmail@subdomain.example.co.in',
  'abc@com',
  'abc@com',
  'abc@',
  '@gmail.com',
  'abc@com'
];

console.log('Email Validation Results:');
exampleEmails.forEach(email => {
  const isValid = validateEmail(email);
  console.log(`${email}: ${isValid}`);
});
```

Terminal Output:

```
Last login: Mon Nov 6 17:42:46 on ttys000
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ % node /Users/sanjanaseelam/validateEmail.js
Email Validation Results:
abc@gmail.com: true
abc123@example.co.uk: true
abc.sanjana123@example-domain.com: true
sanjana_gmail@subdomain.example.co.in: true
abc@com: false
abc@com: false
abc@: false
@gmail.com: false
abc@com: false
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ %
```

### Phone Number Validation:



The screenshot shows a web browser with a JavaScript file named `validatePhoneNumber.js` open. The script defines a function `validatePhoneNumber(phoneNumber)` that uses a regular expression to check if a phone number is valid. It then tests this function against a list of example phone numbers. A terminal window is open in the background, showing the output of the script.

```
function validatePhoneNumber(phoneNumber) {
  const phoneRegexes = [
    /^(\d{1,3})?(\d{3})?(\d{3})?(\d{4})$/, // Matches formats like +1-(555)-123-4567, 555.123.4567, etc.
    /^(\d{1,3})?(\d{3})?(\d{3})?(\d{4})$/, // Matches formats like +1-555-123-4567, 555-123-4567, etc.
    /^(\d{10})$/, // Matches strict 10-digit format like 5551234567
    /^(\d{3})\s?(\d{3})?(\d{3})?(\d{4})$/ // Matches formats like (555) 123-4567, 555.1234, etc.
  ];

  // Test the phone number against each regex
  for (const regex of phoneRegexes) {
    if (regex.test(phoneNumber)) {
      // Check if the number has exactly 10 digits
      const digitsOnly = phoneNumber.replace(/\D/g, '');
      if (digitsOnly.length === 10) {
        return true;
      }
    }
  }

  return false;
}

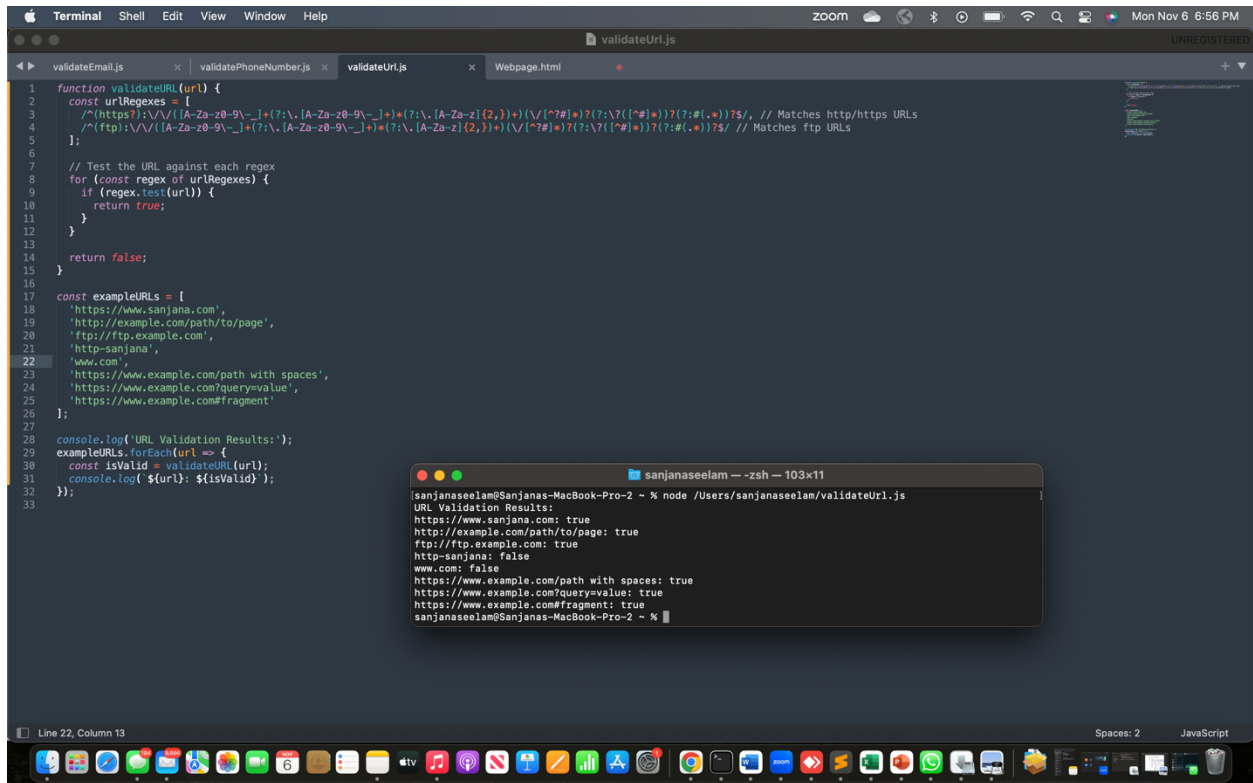
const examplePhoneNumbers = [
  '+1-(555)-123-4567',
  '555.123.4567',
  '+1-555-123-4567',
  '555-123-4567',
  '5551234567',
  '(555) 123-4567',
  '123-45-6789', // Invalid format
  '12345678901' // Invalid format
];

console.log('Phone Number Validation Results:');
examplePhoneNumbers.forEach(phoneNumber => {
  const isValid = validatePhoneNumber(phoneNumber);
  console.log(`${phoneNumber}: ${isValid}`);
});
```

Terminal Output:

```
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ % node /Users/sanjanaseelam/validatePhoneNumber.js
Phone Number Validation Results:
+1-(555)-123-4567: false
555.123.4567: true
+1-555-123-4567: false
555-123-4567: true
5551234567: true
(555) 123-4567: true
123-45-6789: false
12345678901: false
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ %
```

# Url Validation:



The screenshot shows a Mac terminal window with a code editor open. The code editor displays a JavaScript file named `validateUri.js` with the following content:

```
1 function validateURL(url) {
2   const urlRegexes = [
3     /^(https?):\/\/([A-Za-z0-9_-]+(?:\.[A-Za-z0-9_-]+)*(\.[A-Za-z]{2,})+)(\/(?:~?#)?(?:\?|#|.)?)*$/ // Matches http/https URLs
4     /^(ftp):\/\/([A-Za-z0-9_-]+(?:\.[A-Za-z0-9_-]+)*(\.[A-Za-z]{2,})+)(\/(?:~?#)?(?:\?|#|.)?)*$/ // Matches ftp URLs
5   ];
6
7   // Test the URL against each regex
8   for (const regex of urlRegexes) {
9     if (regex.test(url)) {
10      return true;
11    }
12  }
13
14  return false;
15 }
16
17 const exampleURLs = [
18   'https://www.sanjana.com',
19   'http://example.com/path/to/page',
20   'ftp://ftp.example.com',
21   'http-sanjana',
22   'www.com',
23   'https://www.example.com/path with spaces',
24   'https://www.example.com?query=value',
25   'https://www.example.com#fragment'
26 ];
27
28 console.log('URL Validation Results:');
29 exampleURLs.forEach(url => {
30   const isValid = validateURL(url);
31   console.log(`${url}: ${isValid}`);
32 });
33
```

A terminal window titled `sanjanaseelam — zsh — 103x11` is open in the foreground, showing the output of the script:

```
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ % node /Users/sanjanaseelam/validateUri.js
URL Validation Results:
https://www.sanjana.com: true
http://example.com/path/to/page: true
ftp://ftp.example.com: true
http-sanjana: false
www.com: false
https://www.example.com/path with spaces: true
https://www.example.com?query=value: true
https://www.example.com#fragment: true
sanjanaseelam@Sanjanas-MacBook-Pro-2 ~ %
```

The terminal window is positioned over the code editor, showing the execution results for each URL in the `exampleURLs` array. The output indicates that valid URLs return `true` and invalid ones return `false`.