动手实验：设计一个渐进式网页应用

预计所需时间： 30分钟

在这个实验中，您将学习如何将常规网页应用转换为可以离线工作的渐进式网页应用，并对资源进行缓存。

目标：

完成此实验后，您将能够：

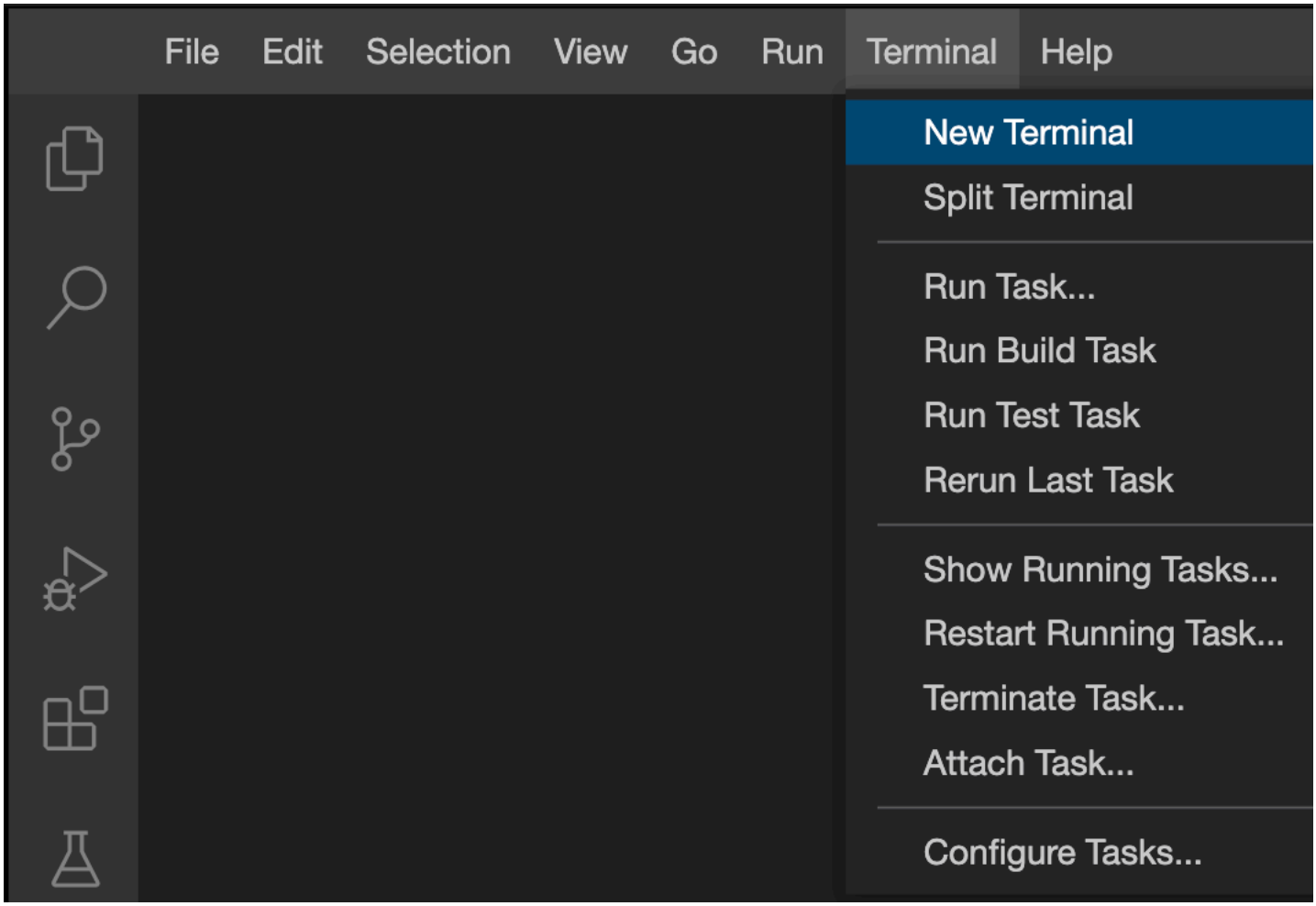
- 将一个简单的网页应用转换为渐进式网页应用
- 利用服务工作者使您的PWA能够离线工作
- 应用步骤使PWA可安装

开发工具：

- 为了充分利用此实验，您需要在支持调试的计算机浏览器上访问开发工具。在本实验中，示例使用Chrome给出，但也可以使用FireFox或Safari等其他浏览器。

设置：创建应用程序

1. 通过编辑器中的菜单打开终端窗口：**终端 > 新终端**。



2. 如果您当前不在项目文件夹中，请复制并粘贴以下代码以切换到您的项目文件夹。

```
cd /home/project
```

3. 运行以下命令以克隆包含本实验所需起始代码的 Git 仓库，如果 Git 仓库尚不存在的话。

```
[ ! -d 'atdch-c2f_Converter' ] && git clone https://github.com/ibm-developer-skills-network/atdch-c2f_Converter.git
```

4. 切换到目录 **atdch-c2f_Converter** 开始实验。

```
cd atdch-c2f_Converter
```

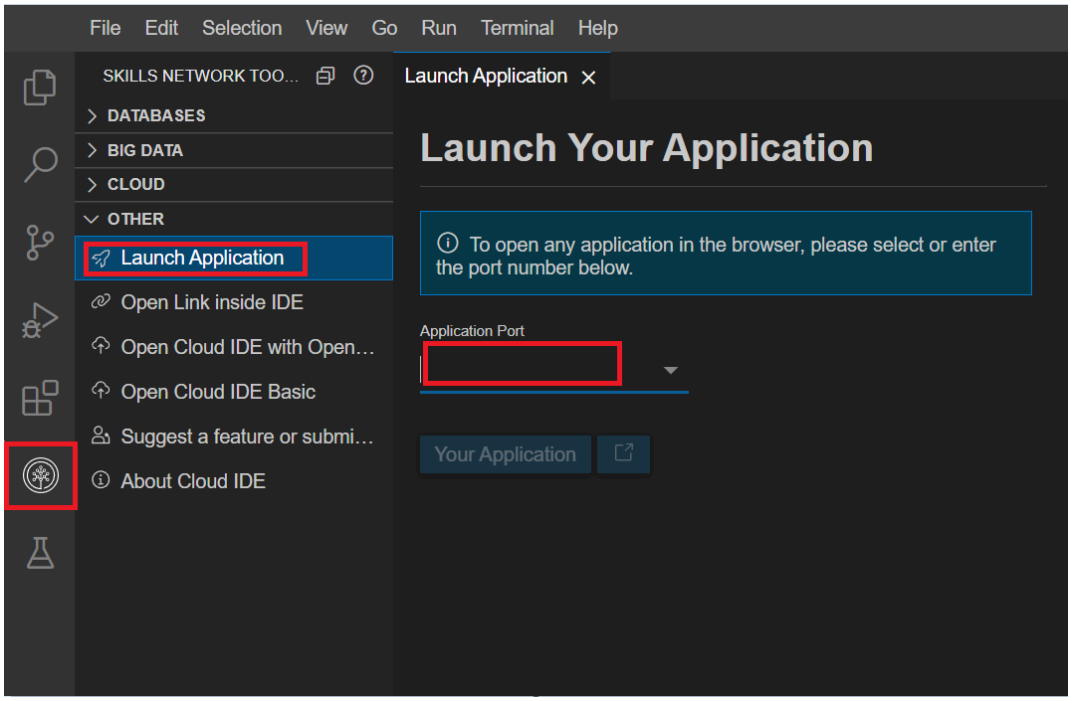
5. 列出该目录的内容，以查看本实验的工件。

```
ls
```

6. 在终端运行以下命令以托管您的网页。

```
npx -y http-server
```

7. 要通过浏览器窗口验证相同的内容，请单击左侧的技能网络按钮，它将打开“技能网络工具箱”。然后点击**其他**，再点击**启动应用程序**。从那里，您应该能够输入服务器运行的端口号，即`8080`并启动。

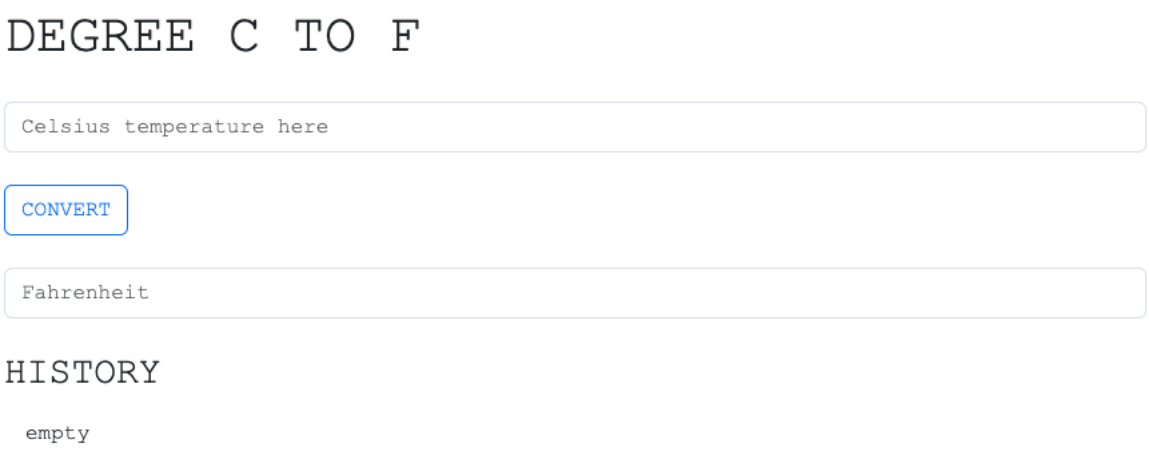


一个新的浏览器窗口将会打开，如下所示。(*注意：如果您的浏览器设置不允许弹出窗口，则可能不会打开新的浏览器窗口*)

8. 或者，您可以点击“启动应用程序”按钮，在浏览器中测试您的应用程序。

Launch Application

9. 它将看起来像这样：



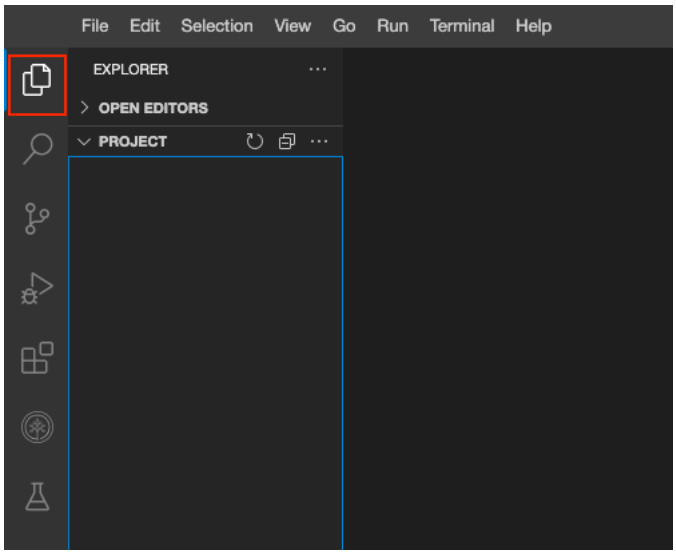
10. 您可以复制URL并在自己的浏览器中运行。

在云IDE中处理文件

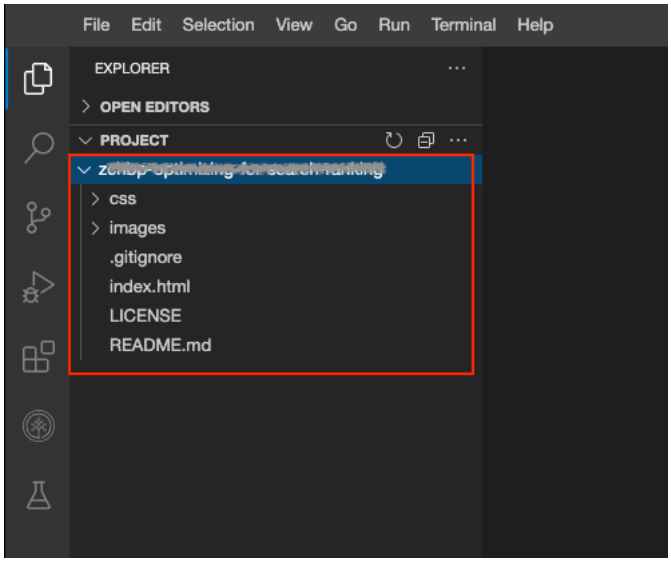
如果您是云IDE的新手，本节将向您展示如何在云IDE中创建和编辑项目中的文件。

注意：本节供学习者参考，以便在Theia实验室中工作。

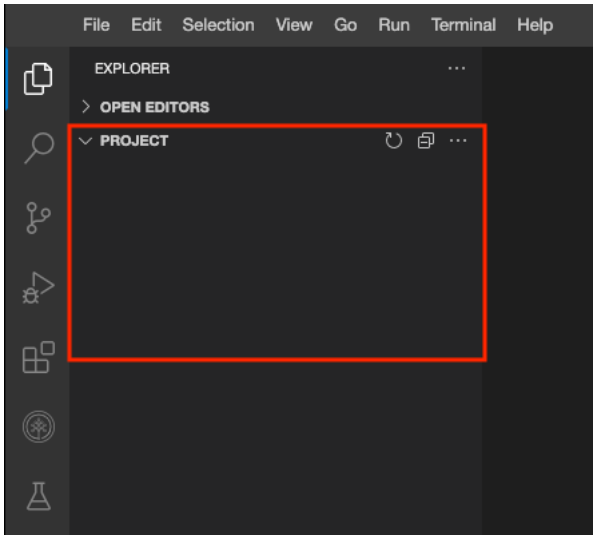
要查看您在云IDE中的文件和目录，请单击此文件图标以显示它。



如果您已克隆（使用`git clone`命令）模板/起始代码，则它将如下所示：

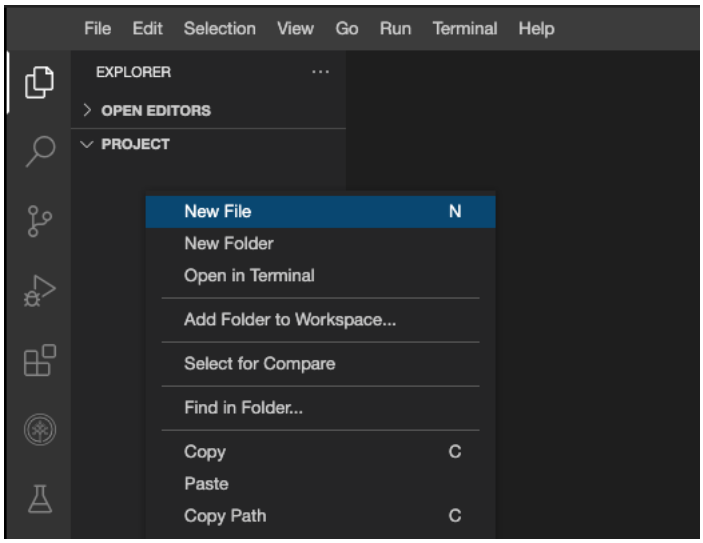


否则，空项目看起来是这样的：



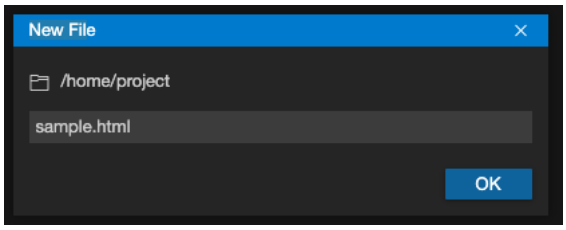
创建新文件

您可以右键单击并选择“新建文件”选项在项目中创建文件。

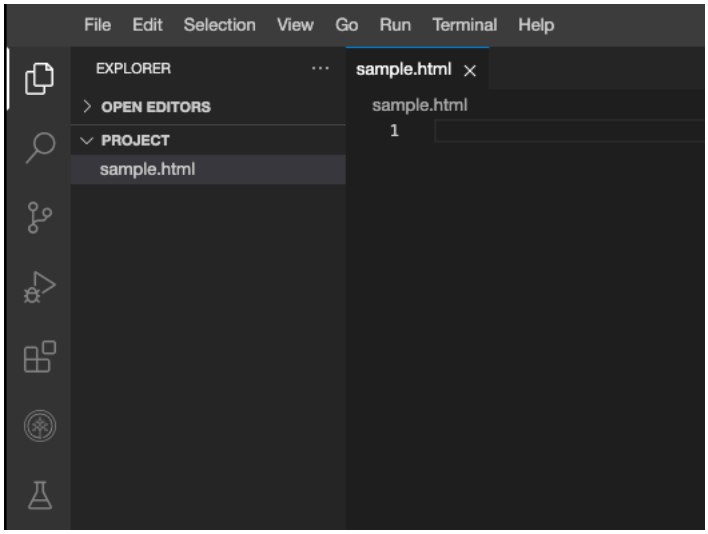


您也可以选择文件 -> 新建文件来实现相同的操作。

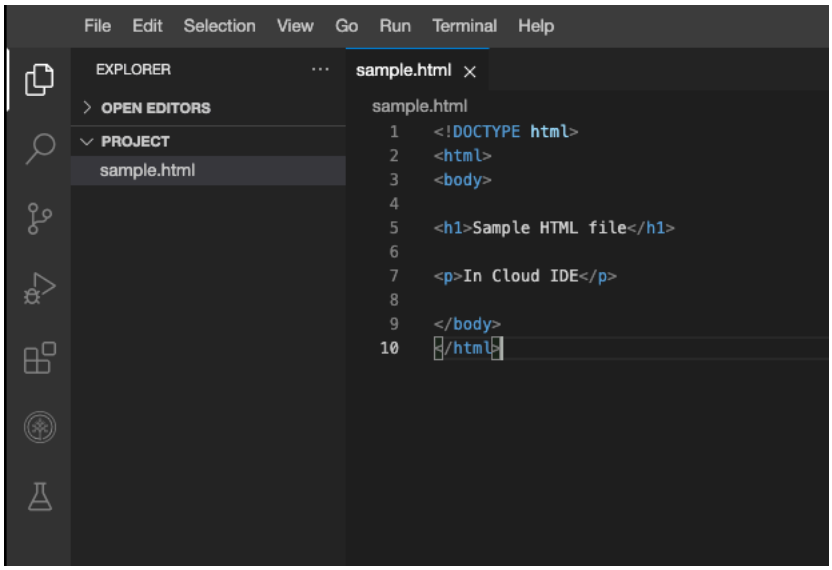
然后会提示您输入新文件的名称。在下面的示例中，我们创建sample.html。



在目录结构中单击文件名sample.html将打开右侧窗格中的文件。您可以创建各种类型的文件；例如，JavaScript文件可以命名为FILE_NAME.js。

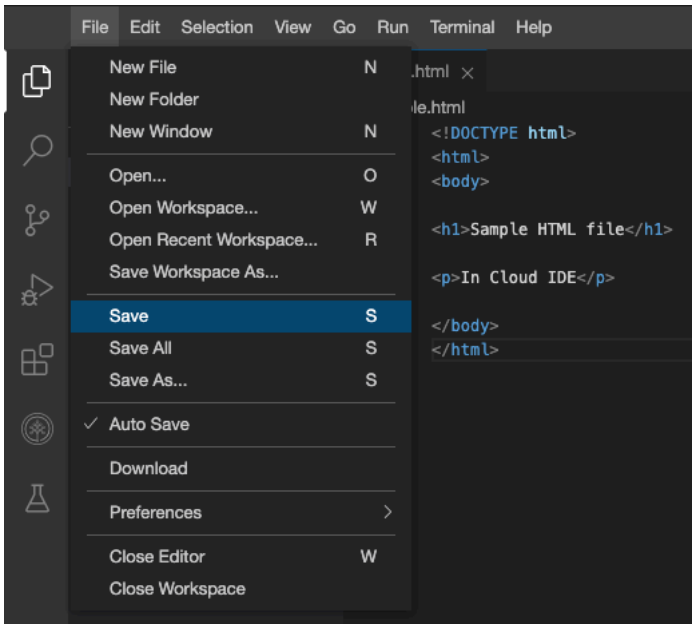


在示例中，我们刚刚粘贴了一些基本的html代码，然后保存了文件。



并通过以下方式保存它：

- 进入菜单。
- 在Mac上按⌘ Cmd + S或在Windows上按CTRL + S。
- 或者它也可以为您自动保存。



练习 1：不是 PWA

现在您的应用程序正在运行，在第一个输入框中输入一些值并按下 `CONVERT`。您将看到转换后的华氏温度值。

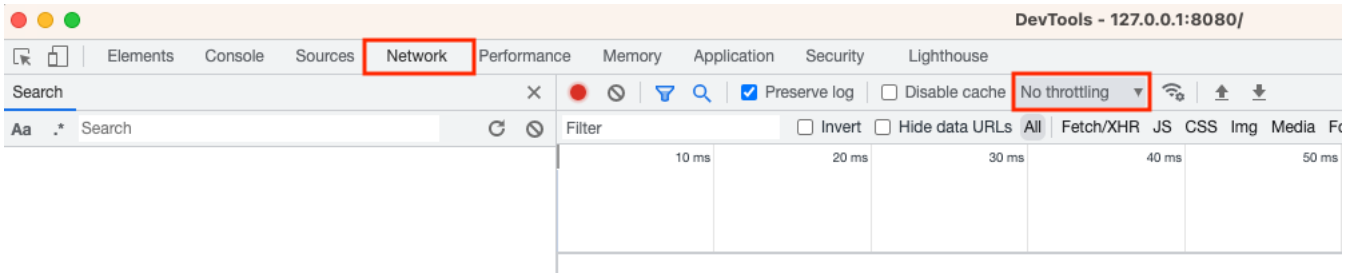
该应用程序还使用 window.localStorage 来保留最近的 3 次转换记录。

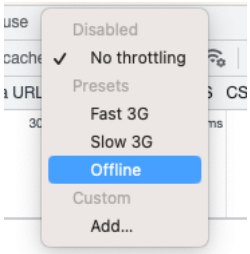
离线工作

我们现在将打开开发者工具，看看我们的应用程序是否可以离线工作。在这个例子中，我们使用的是 Chrome，但所有浏览器都支持某种形式的开发者工具进行调试。

- 要访问 DOM 或 CSS，请右键单击页面上的一个元素，然后选择 **检查** 以打开元素面板。您也可以使用 `Command+Option+C (Mac)` 或 `Control+Shift+C (Windows, Linux, ChromeOS)`。
- 要访问控制台面板并查看日志消息或运行 JavaScript，请按 `Command+Option+J (Mac)` 或 `Control+Shift+J (Windows, Linux, ChromeOS)`。

在 网络 选项卡中，将其设置为 离线。这允许您在浏览器中测试您的应用程序，而无需实际断开互联网连接。





如果您现在通过按 `⌘ Cmd + Shift + R` 在 Mac 上或 `CTRL + F5` 在 Windows 上强制刷新浏览器，您将看到显示 `无互联网` 或类似的消息。

在接下来的步骤中，我们将使这个应用程序即使在离线时也能工作。

但现在，请将其设置回 `无限流量限制`。否则，Chrome 将表现得像是未连接到互联网。

练习 2：添加服务工作者

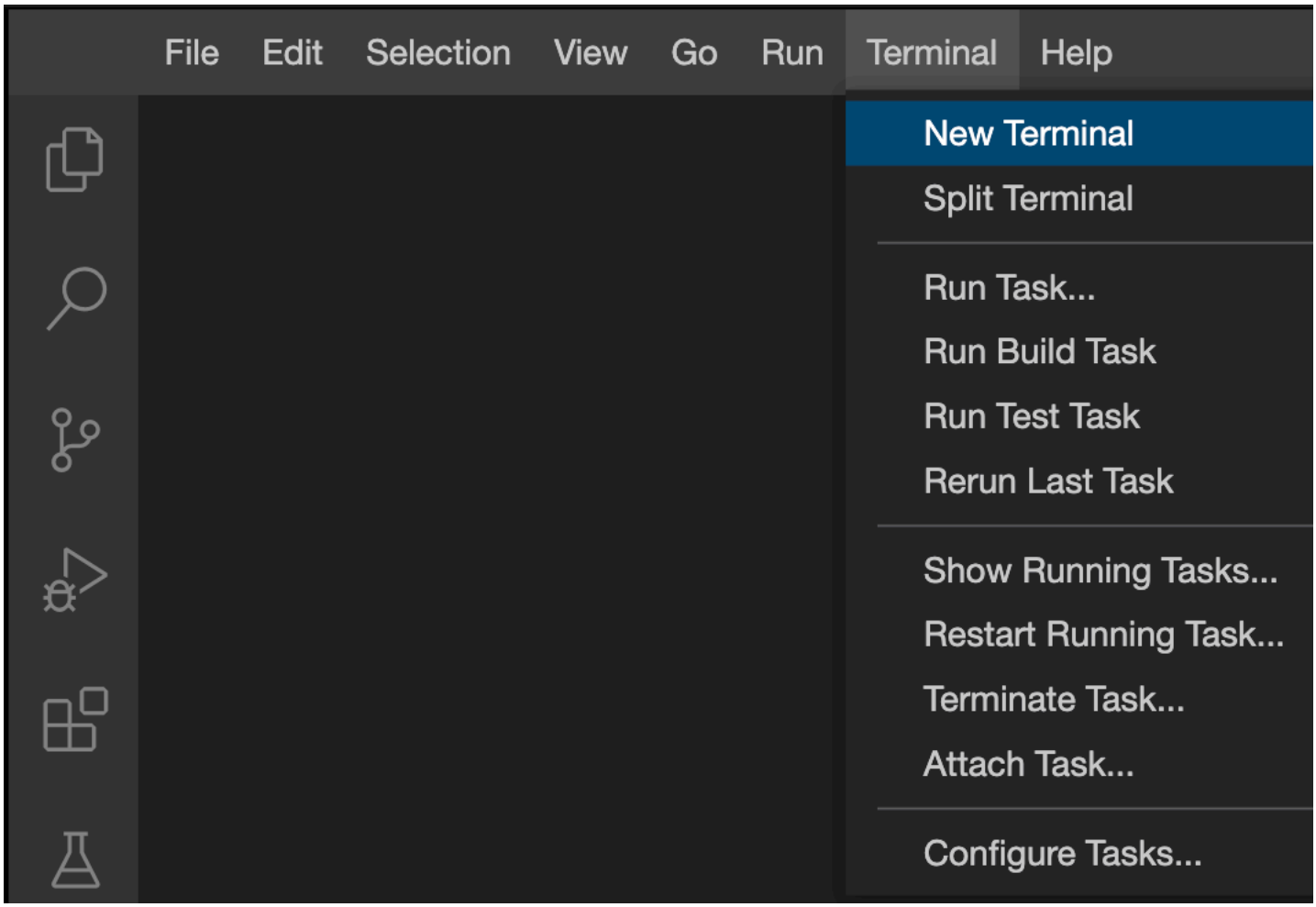
服务工作者是一种 Web Worker，它在渐进式网络应用（PWA）的后台运行，提供了一种缓存内容和资产以便离线访问、处理推送通知以及拦截网络请求的方式，从而启用额外的功能，如后台同步和地理定位。

通过缓存内容和资产，服务工作者可以通过加快加载速度并在没有互联网连接时仍然正常工作，从而提高 PWA 的性能。这可以创造更无缝和可靠的用户体验，这对那些旨在在移动设备上使用的 PWA 尤为重要。

服务工作者使用 JavaScript 编写，并在网页应用的主线程中注册。它们可以被视为一种代理服务器，拦截并处理 PWA 发出的网络请求。由于服务工作者独立于主线程运行，即使用户的设备面临高网络流量或其他性能问题，它们也能提供可靠且一致的性能。

让我们将服务工作者作为我们 PWA 的第一步来添加。要更改代码，您可以手动应用更改，或者运行预定义的脚本为您下载更新的文件。

通过编辑器中的菜单打开一个新的终端窗口：**终端 > 新建终端**。



将目录更改为项目文件夹。

```
cd /home/project/atdch-c2f_Converter
```

并运行以下脚本：

```
curl https://raw.githubusercontent.com/ibm-developer-skills-network/atdch-c2f_Converter/wip/sw.js -o sw.js
```

或者，将以下代码粘贴到新创建的文件 `sw.js` 中。

```
const CACHE_NAME = 'c-to-f-converter';
// Event listener for the 'install' event, triggered when the service worker is first installed
self.addEventListener('install', event => {
    event.waitUntil((async () => {
        // Open (or create) the cache with the specified name
        const cache = await caches.open(CACHE_NAME);
        // Add files to the cache for offline use
        await cache.addAll([
            '/',                // Homepage
            '/script.js',       // JavaScript file
            '/style.css',       // CSS file
            '/bootstrap.min.css', // Bootstrap CSS file
        ]);
    })());
});
// Event listener for the 'fetch' event, triggered for each network request made by the page
self.addEventListener('fetch', event => {
    event.respondWith((async () => {
        // Open the cache to check for cached resources
        const cache = await caches.open(CACHE_NAME);
        try {
            // Try to fetch the resource from the network
            const fetchResponse = await fetch(event.request);
            // If successful, store the network response in the cache for future use
            cache.put(event.request, fetchResponse.clone());
            // Return the network response to the page
            return fetchResponse;
        } catch (e) {
            // If the network request fails (e.g., offline), try to fetch from the cache
            const cachedResponse = await cache.match(event.request);
            // If the resource is found in the cache, return it
            if (cachedResponse) {
                return cachedResponse;
            } else {
                // If the resource is not available in the cache, return a custom error response
                return new Response('Network error occurred and no cache available', {
                    status: 408, // Request Timeout status
                    statusText: 'Network error occurred'
                });
            }
        }
    })());
});
```

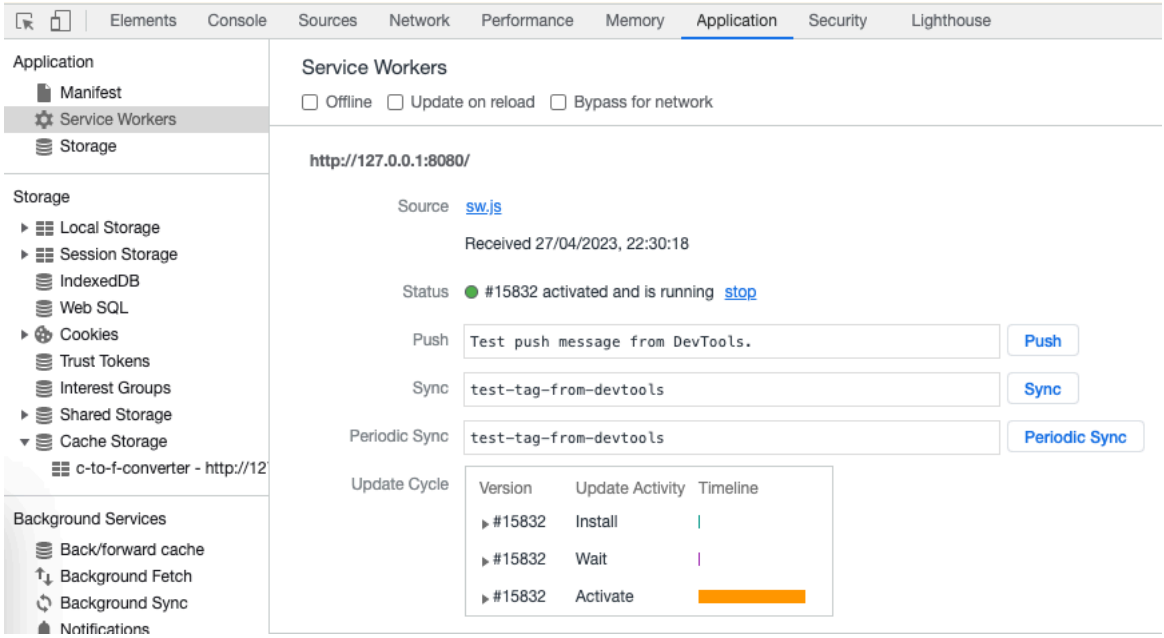
现在我们在 `index.html` 文件中引用工人。您可以运行以下脚本：

```
curl https://raw.githubusercontent.com/ibm-developer-skills-network/atdch-c2f_Converter/wip/index.html -o index.html
```

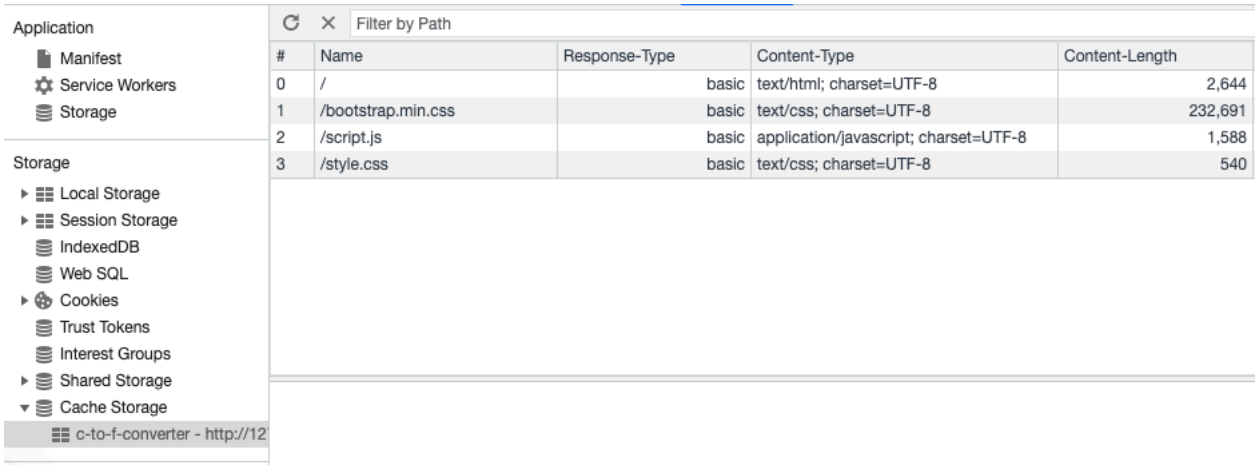
或者在 `index.html` 的 `head` 部分添加以下内容。

```
<script>
if('serviceWorker' in navigator) {
  navigator.serviceWorker.register('/sw.js', { scope: '/' });
}
</script>
```

再次打开 Chrome DevTools，然后转到 `Application` 部分。这将显示您刚刚引入的服务工作者。

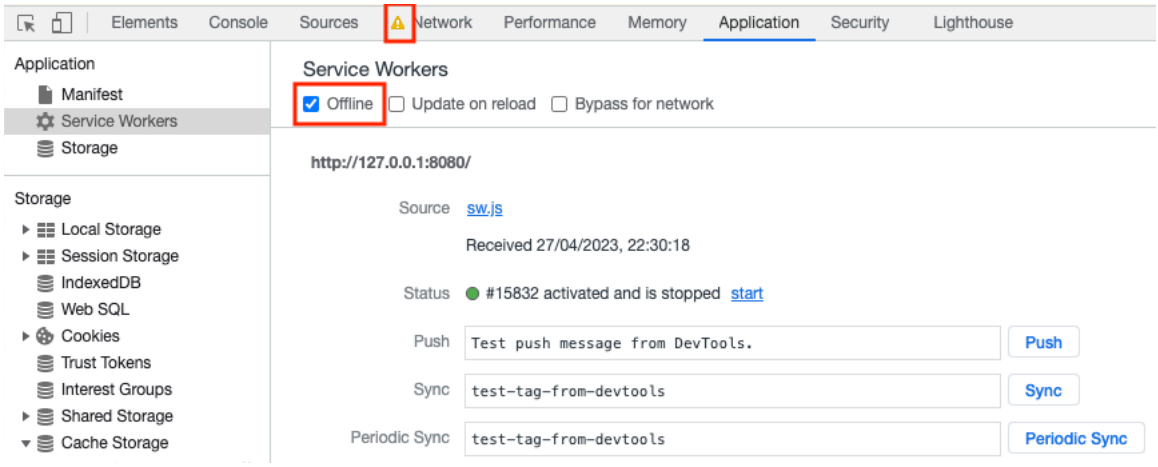


要查看它正在缓存的内容，请转到 `Cache Storage`



练习 3：离线测试

在开发者工具中，我们可以让它看起来像是与互联网断开连接，就像我们之前做的那样。这也可以在 `Application` 选项卡中完成。



如果你强制刷新浏览器，我们的摄氏度到华氏度转换器应用程序应该仍然按预期工作。

练习 4：使其可安装

一个 Web 应用清单 是一个包含有关您应用程序的元数据的 JSON 文件，例如其名称、描述、图标以及它使用的各种操作系统功能。

让我们使用脚本创建一个文件 `manifest.json`。

```
curl https://raw.githubusercontent.com/ibm-developer-skills-network/atdch-c2f_Converter/wip/manifest.json -o manifest.json
```

或 手动使用以下内容。

```
{
    "lang": "en-us",
    "name": "c-to-f app",
    "short_name": "celsius to fahrenheit converter",
    "description": "A basic temperature converter application that can convert from Celsius to Fahrenheit",
    "start_url": "/",
    "orientation": "any",
    "display": "standalone",
    "icons": [
        {
            "src": "/icon512.png",
            "sizes": "512x512"
        }
    ]
}
```

并在 `head` 部分的 `index.html` 中添加引用。

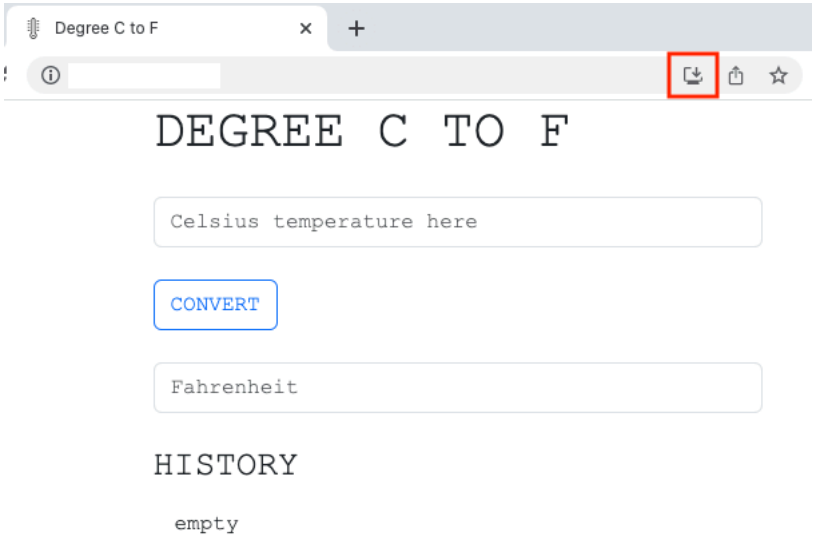
使用脚本：

```
curl https://raw.githubusercontent.com/ibm-developer-skills-network/atdch-c2f_Converter/wip/index-final.html -o index.html
```

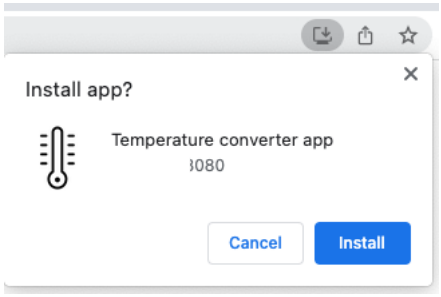
或者 手动，

```
<link rel="manifest" href="/manifest.json">
```

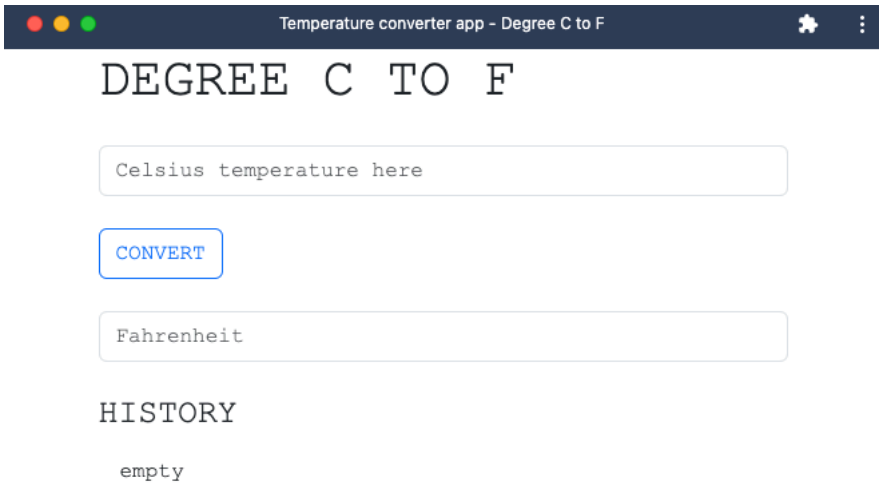
如果您现在通过在Mac上按`⌘ Cmd + Shift + R`或在Windows上按`CTRL + F5`强制刷新浏览器，您将会在浏览器中看到这个额外的选项。



一旦您点击它，您将看到一个提示，类似于这个来安装应用程序。



最后，安装后，在Mac上看起来是这样的；在Windows上也会类似。



恭喜！您已完成渐进式网络应用实验。

总结：

在这个实验中，您将一个普通的JavaScript应用程序转换为渐进式网络应用，使其能够离线工作并可安装。

Author(s)

Muhammad Yahya

(C) IBM Corporation 2023. All rights reserved.