Background processing using web workers

Web workers lets you run CPU-intensive computations in a background thread, freeing the main thread to update the user interface. If you find your application performs a lot of computations, such as generating CAD drawings or doing heavy geometrical calculations, using web workers can help increase your application's performance.

The CLI does not support running Angular itself in a web worker.

Adding a web worker

To add a web worker to an existing project, use the Angular CLI `ng generate` command.

```
ng generate web-worker <location>
```

You can add a web worker anywhere in your application. For example, to add a web worker to the root component, `src/app/app.component.ts`, run the following command.

```
ng generate web-worker app
```

The command performs the following actions.

- Configures your project to use web workers, if it isn't already.
- Adds the following scaffold code to `src/app/app.worker.ts` to receive messages.

  addEventListener('message', ({ data }) => { const response = `worker response to ${data}`; postMessage(response); });

- Adds the following scaffold code to `src/app/app.component.ts` to use the worker.

  if (typeof Worker !== 'undefined') { // Create a new const worker = new Worker(new URL('./app.worker', import.meta.url)); worker.onmessage = ({ data }) => { console.log(`page got message: ${data}`); }; worker.postMessage('hello'); } else { // Web workers are not supported in this environment. // You should add a fallback so that your program still executes correctly. }

After you generate this initial scaffold, you must refactor your code to use the web worker by sending messages to and from the worker.

Some environments or platforms, such as `@angular/platform-server` used in Server-side Rendering, don't support web workers. To ensure that your application will work in these environments, you must provide a fallback mechanism to perform the computations that the worker would otherwise perform.