BHARATIVIDYAPEETH

COLLEGE OF ENGINEERING

DEPARTMENT OF INFORMATION TECHNOLOGY

ACADEMIC YEAR: 2021-2022

COURSE NAME: MAD&PWA

COURSECODE	**ITL604**
EXPERIMENTNO.	**09**
EXPERIMENTTITLE	**To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.**
NAMEOFSTUDENT	**Prathamesh Sable**
ROLLNO.	**60**
CLASS	**TE-IT**
SEMESTER	**VI**
GIVEN DATE	**30/03/2022**
SUBMISSIONDATE	**4/06/2022**
CORRECTIONDATE	
REMARK	

TIMELY SUBMISSION		**PRESENTATION**		**UNDERSTANDING**		**TOTALMARKS**	
	04		**04**		**07**		**15**
NAME&SIGN.							

OFFACULTY	**Prof.S.M.Patil**	

Experiment No 9

Aim: To code and register a service worker, and complete the install and activation process for a new service worker for the E-commerce PWA.

Theory:

The Web App Manifest identifies our website as an app to the browser. The ServiceWorker is the last piece of the puzzle, the piece that will take our plain, old, website and give it superpowers like loading offline.

The ServiceWorker is a JavaScript worker that sits between your application and the network. With it (and some supporting APIs like Cache API) we're able to have full control over how our application behaves in any network situation. But before we get ahead of ourselves, let's create an empty ServiceWorker and register it.

In the root of your application, create a file sw.js. You can leave it empty for now.

Next, we need to register our ServiceWorker so that our browser can install it. Go into app.js, and add the following line in the load event listener.

App.js

```
window.addEventListener('load', e => {
newPWAConfApp();
registerSW();
});
```

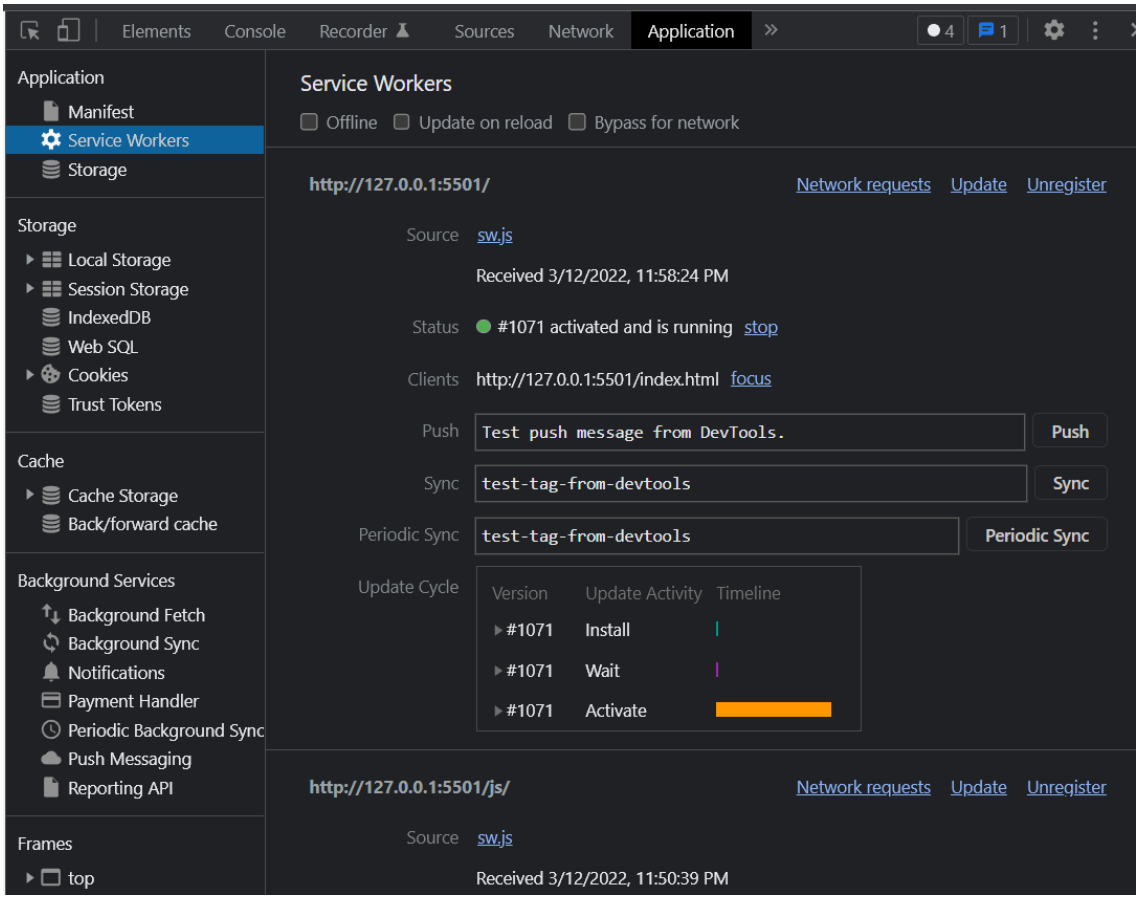
Before we register the ServiceWorker, we want to ensure that the browser supports it. Add the following method after the load even listener:

App.js

```
async function registerSW() {
  if ('serviceWorker' in navigator) {
    try {
awaitnavigator.serviceWorker.register('./sw.js');
    } catch (e) {
alert('ServiceWorker registration failed. Sorry about
that.');
    }
  } else {
document.querySelector('.alert').removeAttribute('hidden'
                                    );
                                     }
}
```

1. We're using an async function for registering the ServiceWorker for easier to read code.
2. Ensure that the browser supports ServiceWorker before trying to register one.
3. Register the ServiceWorker with navigator.serviceWorker.register.
4. Let the user know if things failed.
5. Show an unintrusive notice that offline is not supported if the browser doesn't support ServiceWorker.

Reload your browser and switch to the *ServiceWorker* section of the *Application* tab in DevTools. You should see your service worker listed. To make our lives easier while developing, check"Update on reload"**.** Normally, a new ServiceWorker is only loaded once all tabs using it have been closed, so we would not get our new version visible by just refreshing the page while developing otherwise.



Conclusion: Service workers are event driven. Both the installation and activation processes trigger corresponding install and activate events to which the service workers can respond.