web worker vs service worker vs worklet

refs

- <https://www.jianshu.com/p/e2cdc78ff47c>
- <https://www.smashingmagazine.com/2021/06/web-workers-2021/>
- <https://surma.dev/things/omt-for-three-xr/index.html>
- <https://yarin.dev/nodejs-cpu-bound-tasks-worker-threads/>

web worker

> □□□□□□□□□□□□□□□□ server□□□□□□□□ rpc □□□□□□□□□api□
> onMessage/postMessage □□□□

```
// worker.js
// □□□□□□□□□□
addEventListener("message", (message) => {
if (message.data.command === "run") {
run(message.data.args);
}
});

function run(args) {
// ...

postMessage(/* value to return */);
}

// main.js
const worker = new Worker("./worker.js");

document.querySelector("#run").addEventListener("click", () => {
const args = document.querySelector("#args").value;
worker.postMessage({
command: "generate",
args,
});
});

// □ worker □□□□□□□□□□□□□□callback□□ message □□□□□
// worker.onMessage = (message) => {
worker.addEventListener("message", (message) => {
document.querySelector("#output").textContent =
`Finished by web worker, return is ${message.data}!`;
});

// □□ worker
// worker.terminate();

// □□□□
// worker.onError = (error: ErrorEvent) => {};
```

worker □□□ lib

```
// □□□□□□□□□□□□□□□□□□□□□□□□
importScripts(); /* □□□□□□□ */
importScripts("foo.js"); /* □□□ "foo.js" */
importScripts("foo.js", "bar.js"); /* □□□□□□□ */
importScripts("//example.com/hello.js"); /* □□□□□□□□□□□□□□ */
```

service worker

PWA □□□□□□□

```
/* main.js */
```

```
navigator.serviceWorker.register('/service-worker.js');


/* service-worker.js */
// Install
self.addEventListener('install', function(event) {
// ...
});
// Activate
self.addEventListener('activate', function(event) {
// ...
});
// Listen for network requests from the main document
self.addEventListener('fetch', function(event) {
// Return data from cache
event.respondWith(
caches.match(event.request);
);
});
```

> service worker □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
> □□□□□□□service worker □□□□□□□□□□□□□□□□□□

worklet

```
/* main.js */
CSS.paintWorklet.addModule('myWorklet.js');

/* myWorklet.js */
registerPaint('myGradient', class {
paint(ctx, size, properties) {
var gradient = ctx.createLinearGradient(0, 0, 0, size.height / 3);

gradient.addColorStop(0, "black");
gradient.addColorStop(0.7, "rgb(210, 210, 210)");
gradient.addColorStop(0.8, "rgb(230, 230, 230)");
gradient.addColorStop(1, "white");

ctx.fillStyle = gradient;
ctx.fillRect(0, 0, size.width, size.height / 3);
}
});
```

```
div {
background-image: paint(myGradient);
}
```

> worklet □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□

NodeJS: child_process

```
const http = require('http');
const path = require('path');
const { fork } = require('child_process');

const port = 3000;

http
.createServer((req, res) => {
const url = new URL(req.url, `http://${req.headers.host}`);
console.log('Incoming request to:', url.pathname);

if (url.pathname === '/fibonacci') {
const n = Number(url.searchParams.get('n'));
```

```
  console.log('Calculating fibonacci for', n);

  const childProcess = fork(path.join(__dirname, 'fibonacci-fork'));

  childProcess.on('message', (message) => {
  res.writeHead(200);
  return res.end(`Result: ${message}`);
  });

  childProcess.send(n);
  } else {
  res.writeHead(200);
  return res.end('Hello World!');
  }
  })
  .listen(port, () => console.log(`Listening on port ${port}...`));
```

Date: 2024-04-13
Words: 576
Time to read: 2 mins

Newer Older

2024-04-14
[react] re-render

2024-04-07
HTTP http/1.x http/2 http/3