Distributive

BROWSER WORKER ↘

Updated: 6 Jan 2026

Quickstart instructions for integrating a DCP Browser Worker into any webpage. The worker can join the Global Network and/or Private Compute Groups. Visitors' devices execute CPU and GPU tasks while the page is open. Earned Compute Credits are deposited into a preconfigured DCP bank account, created via the DCP Portal at https://dcp.cloud, providing an alternative to ad-based monetization.

Browser Worker Overview

Supported Platforms

Chrome, Firefox, Safari, Edge, Opera, Brave (64-bit; x86-64, arm64)

Deployment Model

Browser Workers are embedded into webpages using the **DistributiveWorker** API. Any host visiting a page containing the embedded Worker will execute it within their browser context. This enables instant, zero-install deployment across individual users, home computers, institutional desktops, or cloud-hosted web apps.

Execution Model

Browser Workers are deployments of the core DCP Worker within a web browser environment. They require no installation and execute entirely within the browser's security and isolation model.

In browser deployments:

- The **Worker** runs as browser-hosted JavaScript rather than as a Node.js process.
- **Evaluators** execute within browser-managed execution contexts using the browser's native JavaScript engine and WebGPU implementation.

The browser itself provides process isolation, scheduling, and lifecycle management in place of system services such as systemd, but does not provide any automated load management.

Browser Worker Quickstart

Add the following into the <head> of any webpage to automatically create and start a Browser Worker on the Global Network while a visitor is viewing the page:

```html
<script src="https://scheduler.distributed.computer/dcp-client/dcp-client.js"></script>
<script type="module">
  const id = await new dcp.wallet.Keystore(null,''); // generates random id
  await dcp.identity.set(id);

  const worker = new dcp.worker.DistributiveWorker({
    paymentAddress: new dcp.wallet.Address(
      '0x08d9468dddc9238b08cd72e6b6fdbadf03ae466d
    '),
  });

  await worker.start();
</script>
```

Same example, but configured to join a Private Compute Group and opt out of the Global Network:

```bash
<script src="https://scheduler.distributed.computer/dcp-client/dcp-client.js"></script>
<script type="module">
  const id = await new dcp.wallet.Keystore(null,'');
  await dcp.identity.set(id);

  const worker = new dcp.worker.DistributiveWorker({
    paymentAddress: new dcp.wallet.Address(
      '0x08d9468dddc9238b08cd72e6b6fdbadf03ae466d
    '),
    computeGroups: [{ joinKey: 'demo', joinSecret: 'dcp' }],
    leavePublicGroup: true,
    cores: { cpu: 4, gpu: 1 }
  });

  await worker.start();
</script>
```

See full examples and documentation on GitHub:

- DistributiveWorker template: github.com/dan-distributive/DistributiveWorker-template
    - auto-start demo: jsfiddle.net/DCP_team/g29or41j/
- DistributiveWorker demo: github.com/dan-distributive/DistributiveWorker-demo
    - button-start demo: dcp.network

Operational Characteristics

Browser Workers preserve the same core execution and sandboxing guarantees as Standalone DCP Workers but differ operationally:

- Execution is ephemeral and tied to the browser session.
- Resource availability is governed by browser policies and user activity.

Browser Workers are typically used for ad hoc, volunteer, or opportunistic compute rather than persistent, managed deployments.

Imagine this...

If the code snippet on the next page were embedded into **youtube.com**, every person watching a video — about 160 million people at any given moment — would instantly become part of a DCP Compute Group.

That's **160 million compute nodes**, humming in parallel, doing video transcoding, computational science, accelerating AI training, solving simulations, or optimizing medical schedules — all in the background while you watch cat videos or the latest music drop.

Youtube, Vimeo, New York Times, LinkedIn, Instagram, Tiktok, and any other online community can use DCP to replace web advertising revenue with web compute revenue.

Yesterday's internet was paid for by clicks. Tomorrow's will be powered by cycles.

Distributive

Happy computing