HMR `hotUpdate` Plugin Hook

- Motivation
- Migration Guide

::: tip Feedback Give us feedback at Environment API feedback discussion :::

We're planning to deprecate the `handleHotUpdate` plugin hook in favor of `hotUpdate` hook to be Environment API aware, and handle additional watch events with `create` and `delete`.

Affected scope: `Vite Plugin Authors`

::: warning Future Deprecation `hotUpdate` was first introduced in `v6.0`. The deprecation of `handleHotUpdate` is planned for a future major. We don't recommend moving away from `handleHotUpdate` yet. If you want to experiment and give us feedback, you can use the `future.removePluginHookHandleHotUpdate` to `"warn"` in your vite config. :::

Motivation

The `handleHotUpdate` hook allows to perform custom HMR update handling. A list of modules to be updated is passed in the `HmrContext`.

```
interface HmrContext {
  file: string
  timestamp: number
  modules: Array<ModuleNode>
  read: () => string | Promise<string>
  server: ViteDevServer
}
```

This hook is called once for all environments, and the passed modules have mixed information from the Client and SSR environments only. Once frameworks move to custom environments, a new hook that is called for each of them is needed.

The new `hotUpdate` hook works in the same way as `handleHotUpdate` but it is called for each environment and receives a new `HotUpdateOptions` instance:

```
interface HotUpdateOptions {
  type: 'create' | 'update' | 'delete'
  file: string
  timestamp: number
  modules: Array<EnvironmentModuleNode>
  read: () => string | Promise<string>
  server: ViteDevServer
}
```

The current dev environment can be accessed like in other Plugin hooks with `this.environment`. The `modules` list will now be module nodes from the current environment only. Each environment update can define different update strategies.

This hook is also now called for additional watch events and not only for `'update'`. Use `type` to differentiate between them.

Migration Guide

Filter and narrow down the affected module list so that the HMR is more accurate.

```
handleHotUpdate({ modules }) {
  return modules.filter(condition)
}

// Migrate to:

hotUpdate({ modules }) {
  return modules.filter(condition)
}
```

Return an empty array and perform a full reload:

```
handleHotUpdate({ server, modules, timestamp }) {
  // Invalidate modules manually
```

```
  const invalidatedModules = new Set()
  for (const mod of modules) {
    server.moduleGraph.invalidateModule(
      mod,
      invalidatedModules,
      timestamp,
      true
    )
  }
  server.ws.send({ type: 'full-reload' })
  return []
}

// Migrate to:

hotUpdate({ modules, timestamp }) {
  // Invalidate modules manually
  const invalidatedModules = new Set()
  for (const mod of modules) {
    this.environment.moduleGraph.invalidateModule(
      mod,
      invalidatedModules,
      timestamp,
      true
    )
  }
  this.environment.hot.send({ type: 'full-reload' })
  return []
}
```

Return an empty array and perform complete custom HMR handling by sending custom events to the client:

```
handleHotUpdate({ server }) {
  server.ws.send({
    type: 'custom',
    event: 'special-update',
    data: {}
  })
  return []
}
```

```
// Migrate to...

hotUpdate() {
  this.environment.hot.send({
    type: 'custom',
    event: 'special-update',
    data: {}
  })
  return []
}
```

□□□ #2
□ □□□ □□□ 2026-10-31 13:13:14 UTC
□ □□□ □□□ 2026-10-31 13:13:15 UTC