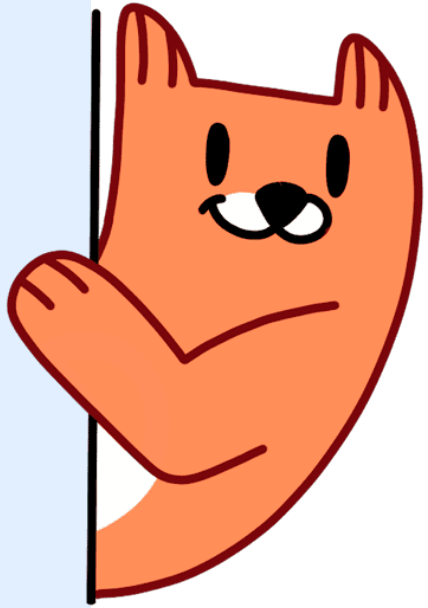
Webpack watch в экстремальных условиях

Николай Рябов, ВКонтакте

Николай Рябов, ВКонтакте

- Играюсь с вебпаком в команде Frontend Infrastructure
- Упарывался 3,5 года перформанс-тестами в Avito
- Погулял пару лет по стартапам (энтерпрайз лучше)
- Люблю графики

О чём?



1. Особенности сборки нашего фронтенда
2. Наследие, которое хоть как-то работало
3. Глубокая кастомизация webpack-dev-server
4. Кастомный HMR, который позволил довести до конца задуманное

ВКонтакте — это монолит

Сборка всего и вся

(webpack и его друзья)

Entrypoints

Индивидуальный рантайм
для каждого энтрипойнта

Отдельная сборка для CSS

Главный энтрипойнт со сборной
солянкой всего подряд

Обилие кэш-групп

StaticManagerPlugin

К каждому чанку мы добавляем специальное окончание с вызовом функции статик-менеджера.

В каждом энтрипойнте должен быть вручную вставленный вызов **stManager.done()** для правильной отработки загрузки энтрипойнта.

```
chunk.files.forEach((file) => {
  if (!file.endsWith('.css')) {
    compilation.assets[file] = new ConcatSource(
      compilation.assets[file],
      `try{stManager.done('${publicPath.slice(1)}${file}');}catch(e){}`,
    );
  }
});
```

config_versions.json

- В результате сборки генерится несколько stats.json файлов
- В каждом статс-файле есть информация обо всех чанках и энтрипойнтах, которые участвовали в сборке
- Все статс-файлы парсятся и преобразовываются в конфиг, нужный для работы **static_manager.js**
- Итоговый конфиг содержит версии всех чанков и зависимости всех энтрипойнтов

```
{
  "versions": {
    "chunk_name.js": "c16781146f02d08ebefe"
  },
  "deps": {
    "entry_name": [
      "entry_name.js",
      "chunk_name.js",
      "chunk_name_foo.js",
      "chunk_name_bar.js"
    ]
  }
}
```

Дополнительные JSON

Инлайн-скрипты

Ключи переводов

Селекторы
для Critical CSS

После сборки

(всё ещё сложнее)

VK/StaticFiles/Common.php

Сбор данных для страницы

- Готовим список энтрипойнтов, которые нужны для конкретной страницы
- Получаем все их зависимости и версии зависимостей
- Собираем данные об уже подключённой статике для глобальных переменных

Подготовка вёрстки страницы

- Все нужные скрипты и стили
- Инлайн-скрипты
- Переводы
- Глобальные переменные с уже подключённой статикой

loader_nav.js

- Да, это PHP
- Зависимости всех энтрипойнтов, которые нужны для **static_manager.js**
- Версии всех чанков и энтрипойнтов
- Роуты и их энтрипойнты

static_manager.js

- Контролирует загрузку самих энтрипойнтов
- Использует **loader_nav.js** как источник данных
- Работает с глобальными переменными версий чанков и уже загруженной статики
- Загружает новые энтрипойнты по запросу вместе с их зависимостями через **stManager.add()**
- Следит за версиями чанков
- Активно используется при переходах между страницами для подгрузки новых энтрипойнтов

Вроде работает

Старый добрый watch

(неудачно состарился)

Сложности

- Как подружить динамические обновления в процессе кодинга с системой **static_manager.js** - **config_versions.json** - **php**?
- Как не утечь по памяти?
- Как ускорить ребилд после инвалидации вотча с такой большой кодовой базой?
- Как собрать то, что нужно, если есть 600+ энтрипойнтов?
- Как ускорить инициализацию вотча?

build

- Собирает иконки
- Билдит воркспейсы
- Отдельно билдит старые CSS-файлы
- Билдит отдельно весь JS- и TS-код
- Порождает все нужные JSON (например, **config_versions.json**)

watch

- Использует **.env** для получения списка нужных для сборки энтрипойнтов
- Собирает энтрипойнты индивидуально
- Не использует стандартные кэш-группы
- Каждый энтрипойнт использует свои собственные инстансы всех библиотек
- Отдельно вотчит CSS и JS
- Использует данные, сгенерированные **build**

watch

- Утекает по памяти
- Всё равно долго инициализируется
- Сложно конфигурируется
- Может ломать сборку
- Совсем неконсистентен с **build**, не говоря уж о продакшн билде
- Так или иначе нужно запускать **build**


build


watch


watch-dynamic


build


watch

watch-dynamic

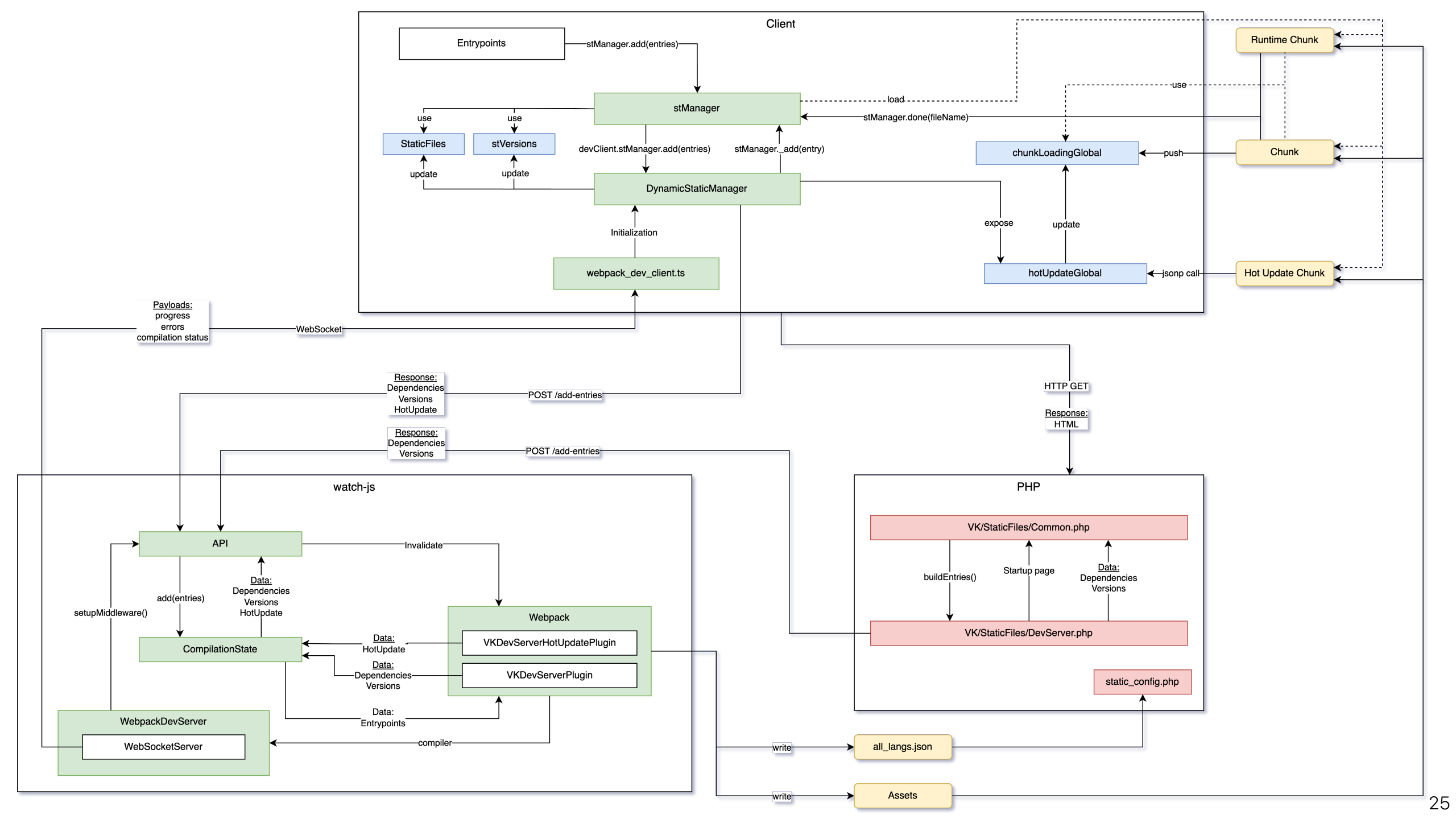
(кродеться)

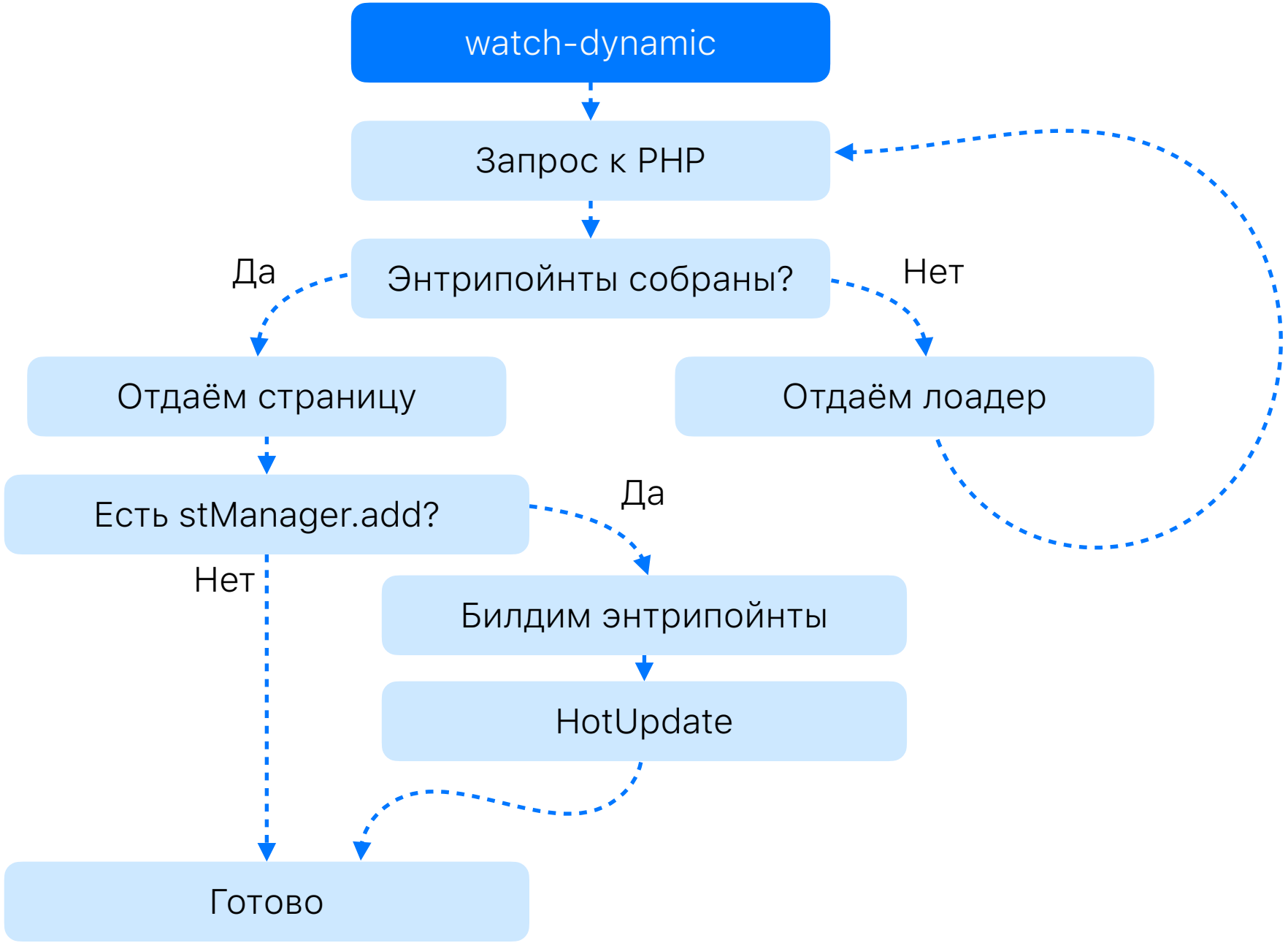
watch-dynamic

- Не утекает по памяти
- Быстро инициализируется
- Не надо конфигурировать
- Не ломает сборку
- Полностью консистентен с **build**
- Не нужно запускать **build**
- Собирает все сорцы одним инстансом вебпака

watch-dynamic

Но как он
это делает?


Client
Entrypoints
stManager.add(entries)
stManager
load
stManager.done(fileName)
use
use
StaticFiles
stVersions
update
update
devClient.stManager.add(entries)
stManager._add(entry)
DynamicStaticManager
Initialization
webpack_dev_client.ts
Runtime Chunk
use
chunkLoadingGlobal
push
Chunk
expose
update
hotUpdateGlobal
jsonp call
Hot Update Chunk
Payloads:
progress
errors
compilation status
WebSocket
Response:
Dependencies
Versions
HotUpdate
POST /add-entries
HTTP GET
Response:
HTML
Response:
Dependencies
Versions
POST /add-entries
watch-js
API
Invalidate
setupMiddleware()
add(entries)
Data:
Dependencies
Versions
HotUpdate
CompilationState
Webpack
VKDevServerHotUpdatePlugin
VKDevServerPlugin
Data:
HotUpdate
Data:
Dependencies
Versions
Data:
Entrypoints
WebpackDevServer
WebSocketServer
compiler
PHP
VK/StaticFiles/Common.php
buildEntries()
Startup page
Data:
Dependencies
Versions
VK/StaticFiles/DevServer.php
static_config.php
write
all_langs.json
write
Assets


watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

Подготовка

- Генерирует инлайн-скрипты
- Собирает Service Worker
- Собирает базовый набор энтрипойнтов, которые присутствуют на всех страницах

Готовим webpack как боженьки

(medium rare)

Динамические энтрипойнты

Позволяют менять список энтрипойнтов даже у активной сборки

Dynamic entry

If a function is passed then it will be invoked on every make event.

> *Note that the `make` event triggers when webpack starts and for every invalidation when watching for file changes.*

```
module.exports = {
  //...
  entry: () => './demo',
};
```

or

```
module.exports = {
  //...
  entry: () => new Promise((resolve) => resolve(['./demo', './demo2'])),
};
```

For example: you can use dynamic entries to get the actual entries from an external source (remote server, file system content or database):

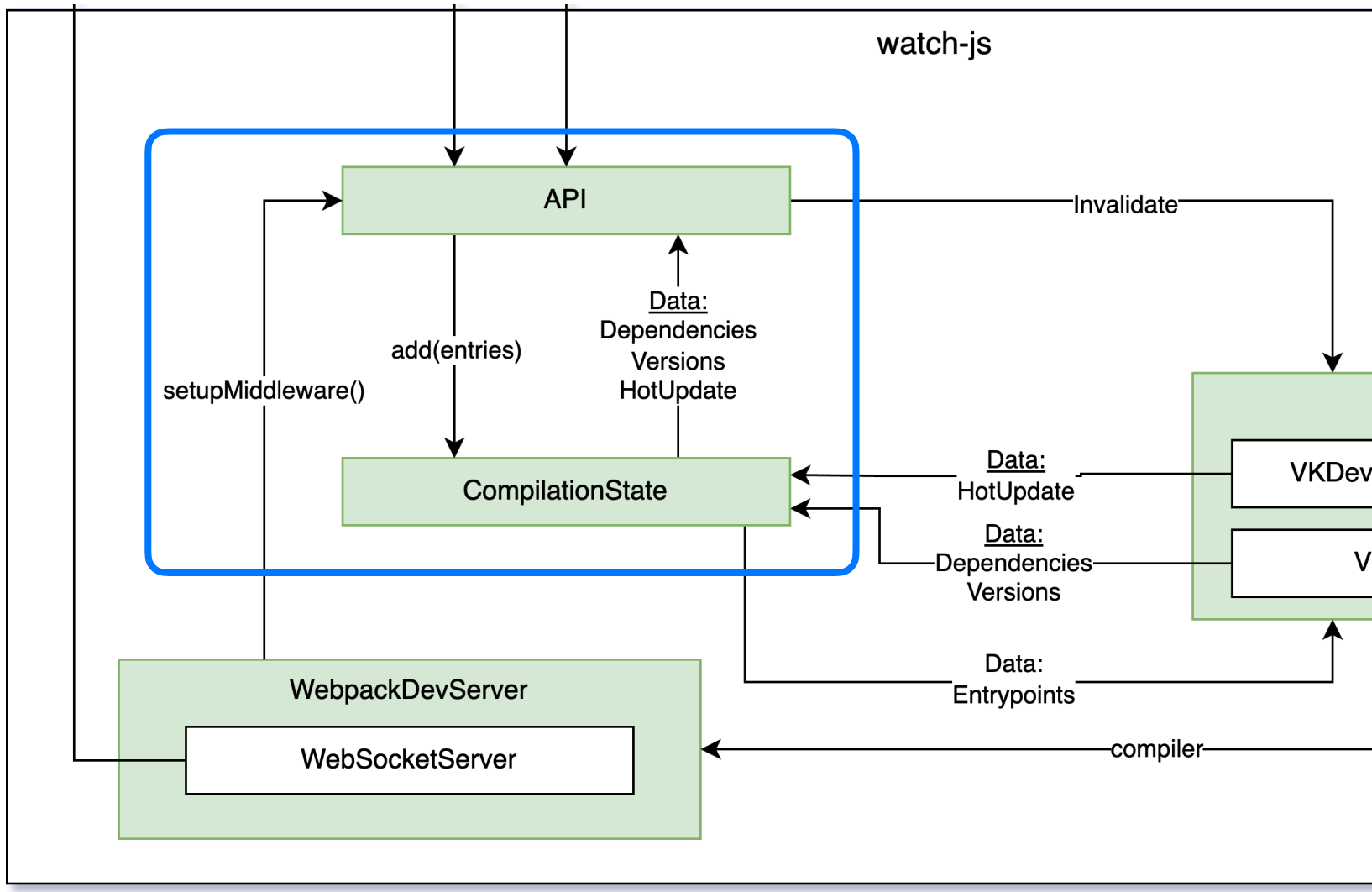
Динамические энтрипойнты

```
compilationState.addEntries(newEntries);
// для сборки новых ентрипоинтов необходимо инвалидировать вотч
devServer.compiler.watching.invalidate();
```

```
// набор ентрипоинтов меняется при запросе /add-entries
webpackConfig.entry = () => compilationState.getEntryOption();
```

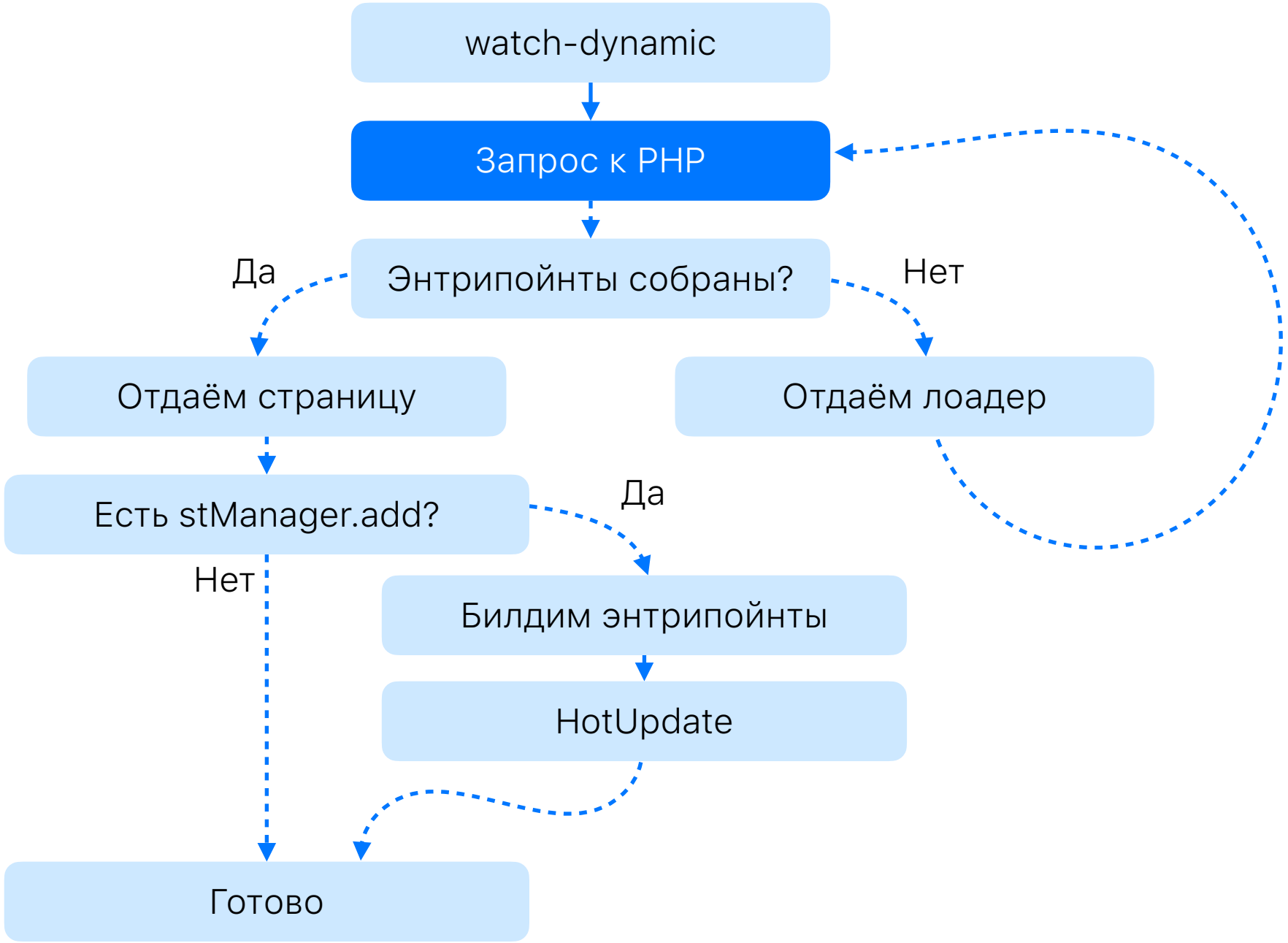
Список энтрипойнтов

Список энтрипойнтов

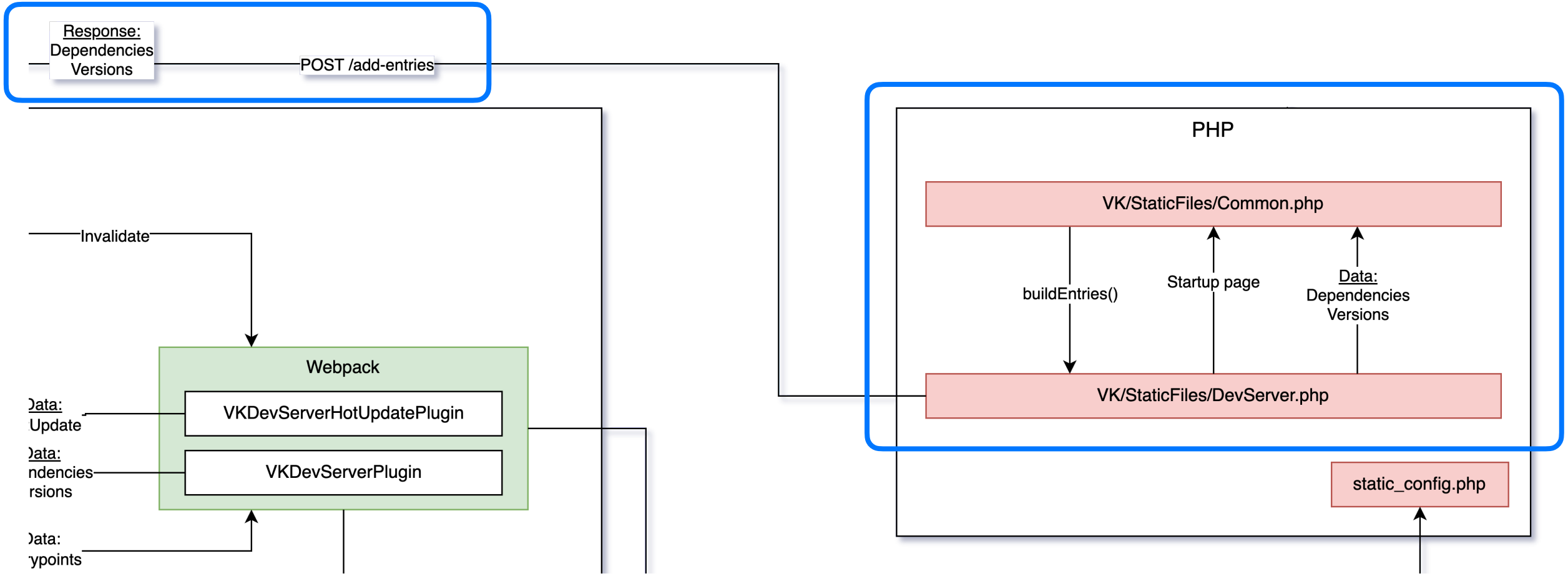


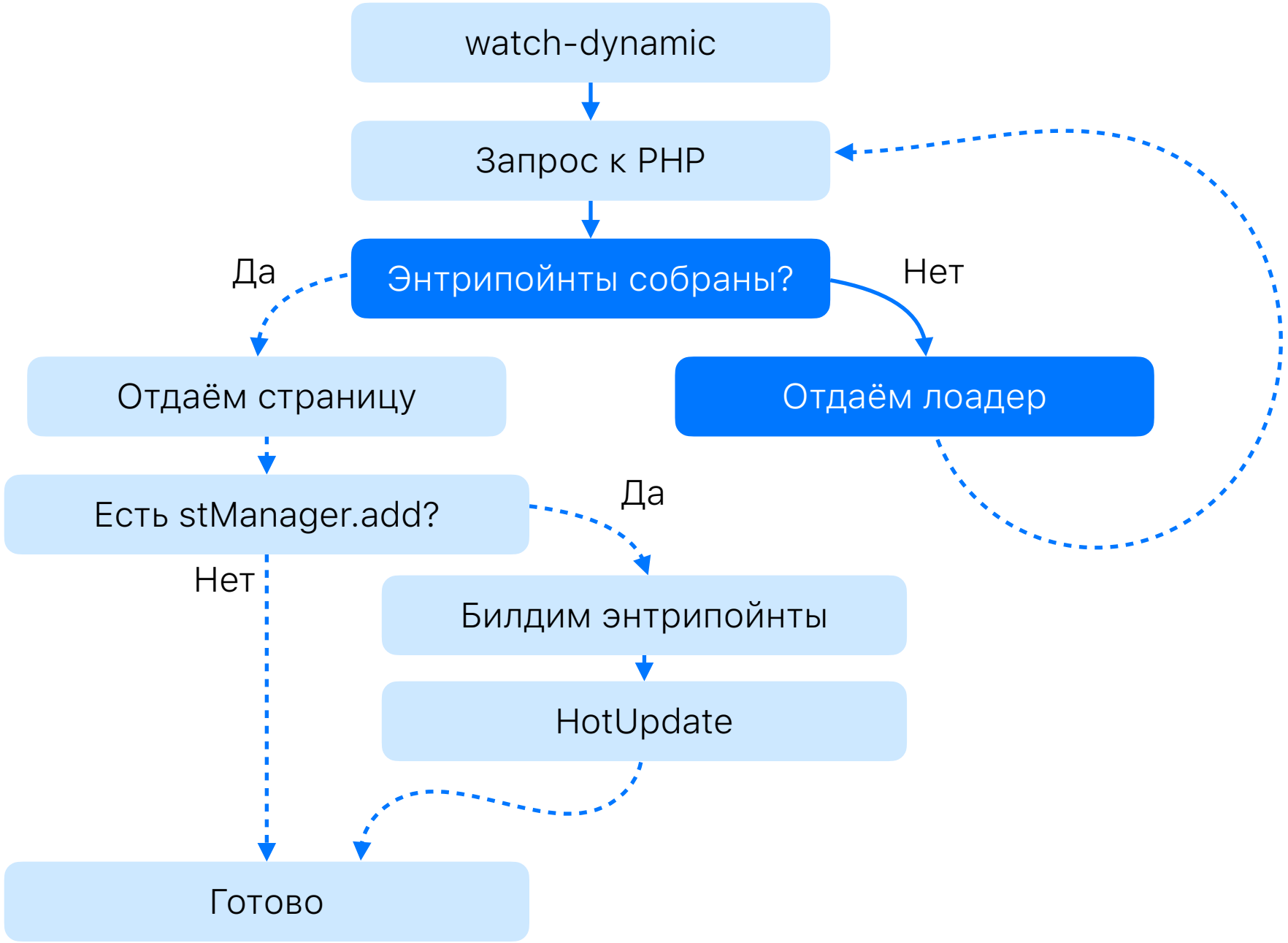
Список энтрипойнтов

```
webpackConfig.entry = () => compilationState.getEntryOption();

const devServerConfig = {
  setupMiddlewares(middlewares, devServer) {
    devServer.app.post('/add-entries', (req, res) => {
      compilationState.addEntries(req.body.entries);
      devServer.compiler.watching.invalidate();

      if (req.body.waiting) {
        compilationState.once('compilation-end', () =>
          res.json({
            versions: compilationState.getVersions(),
            dependencies: compilationState.getDependencies(),
          }),
        );
      } else {
        res.json({ inProgress: true });
      }
    });

    return middlewares;
  },
};

const compiler = webpack(webpackConfig);
const server = new WebpackDevServer(devServerConfig, compiler);
```

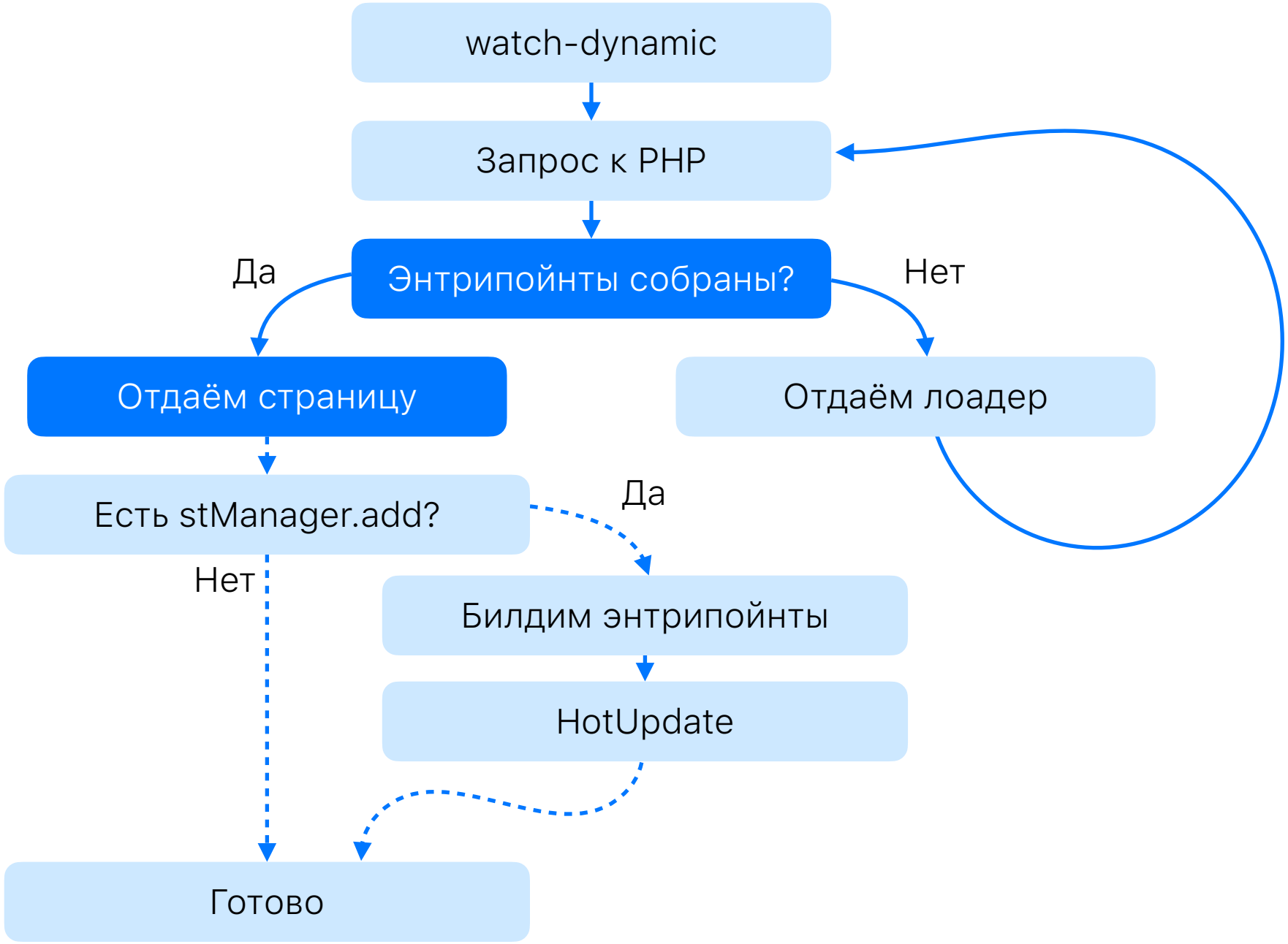

watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

PHP общается с вебпаком

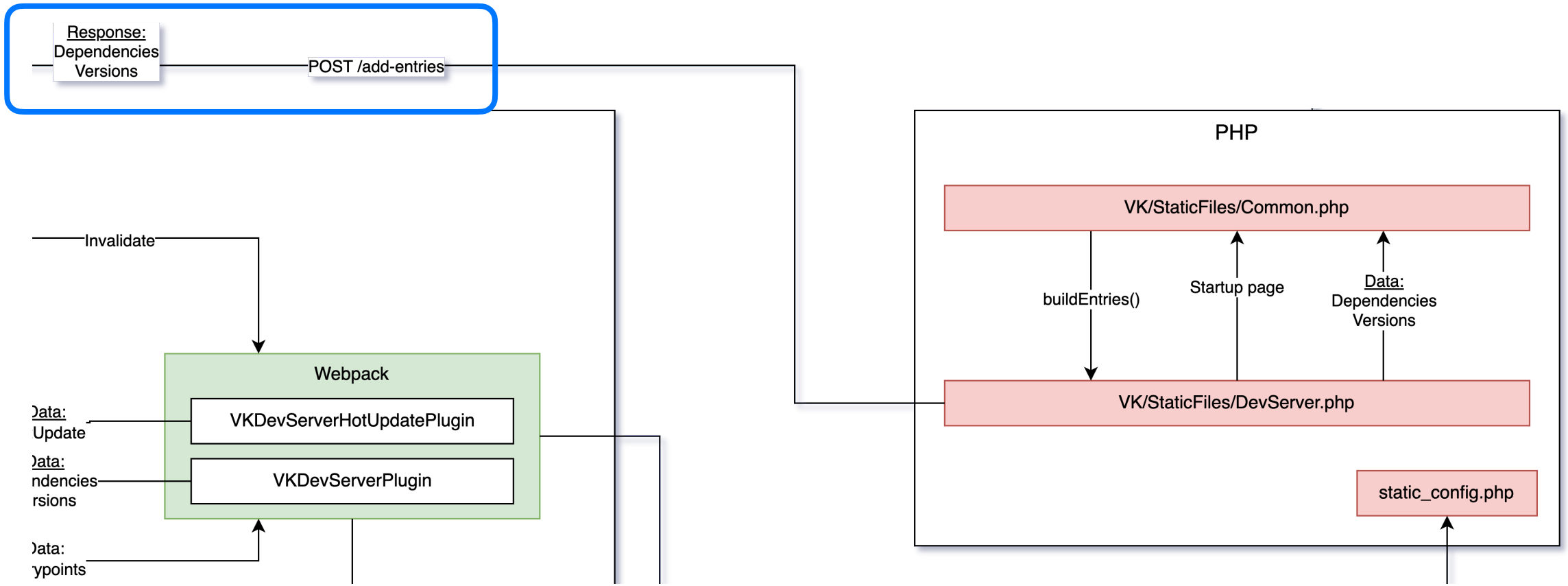



watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

ВКонтактик собирается

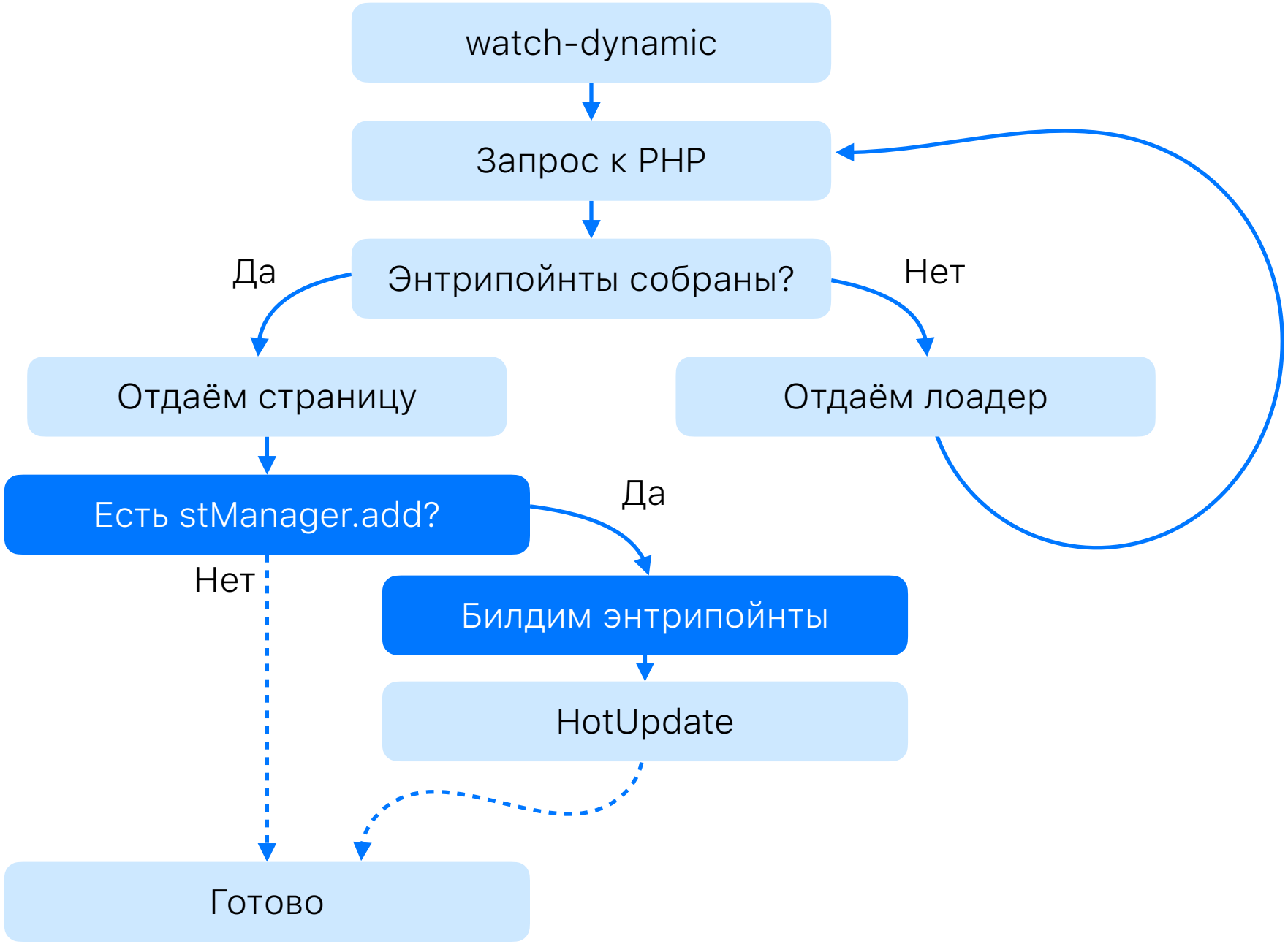

watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

Динамический config_versions.json

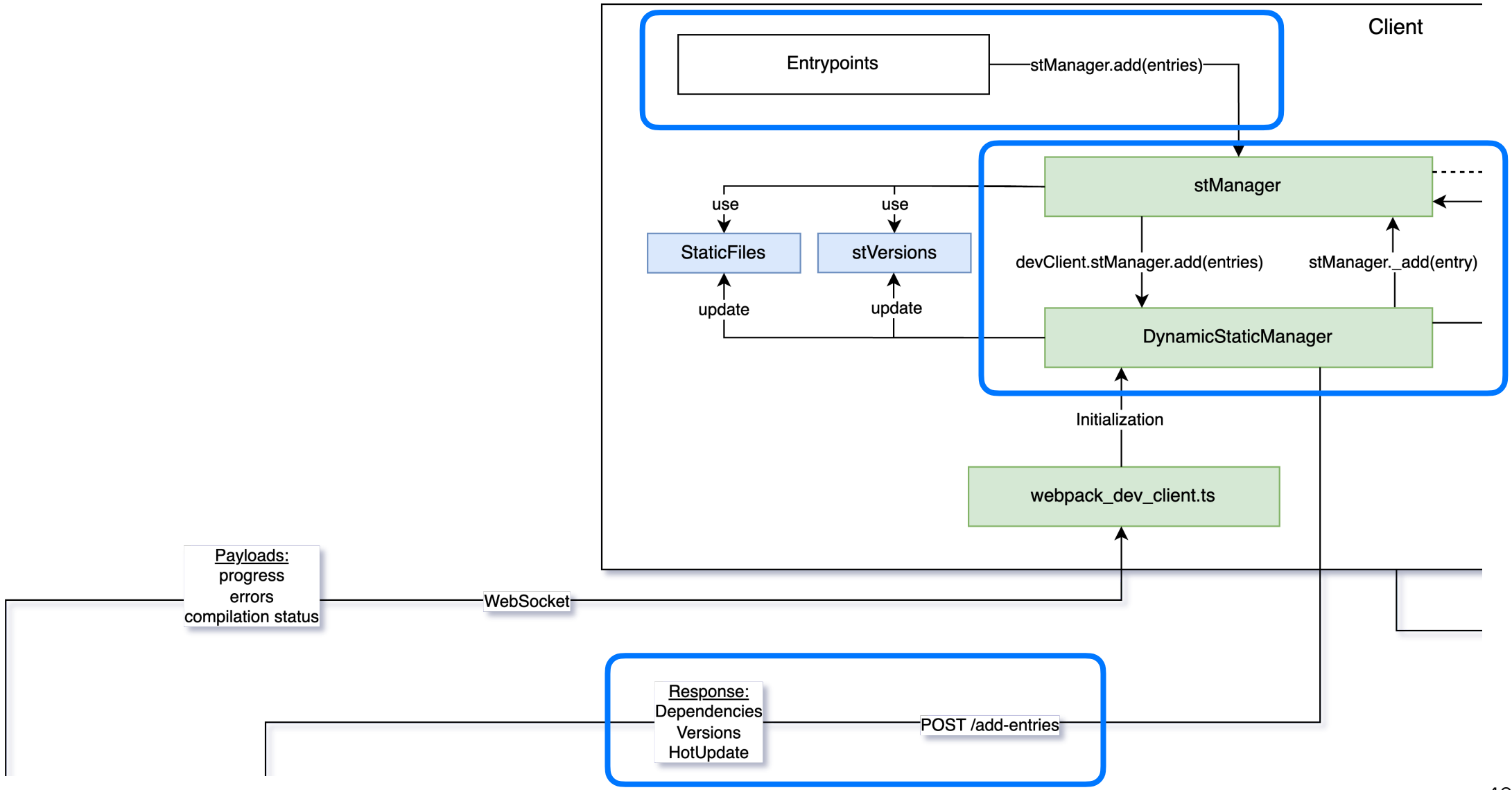


VKDevServerPlugin

```
compiler.hooks.done.tap('VkDevServerPlugin', (stats) => {
      const entries = Array.from(stats.compilation.entrypoints.entries()).map(([entryName, entry]) => [
        entryName,
        entry.chunks
          .map((chunk) =>
            Array.from(chunk.files).map((file) => [file, Object.values(chunk.contentHash).join('_')]),
          )
          .flat(),
      ]);

      const entriesDeps = Object.fromEntries(
        entries.map(([entryName, files]) => [entryName, files.map(([fileName]) => fileName)]),
      );
      const chunkVersions = Object.fromEntries(entries.map(([, files]) => files).flat());

      this.compilationState.update({
        entriesDeps,
        chunkVersions,
      });
      this.compilationState.emit('compilation-end');
    });
```

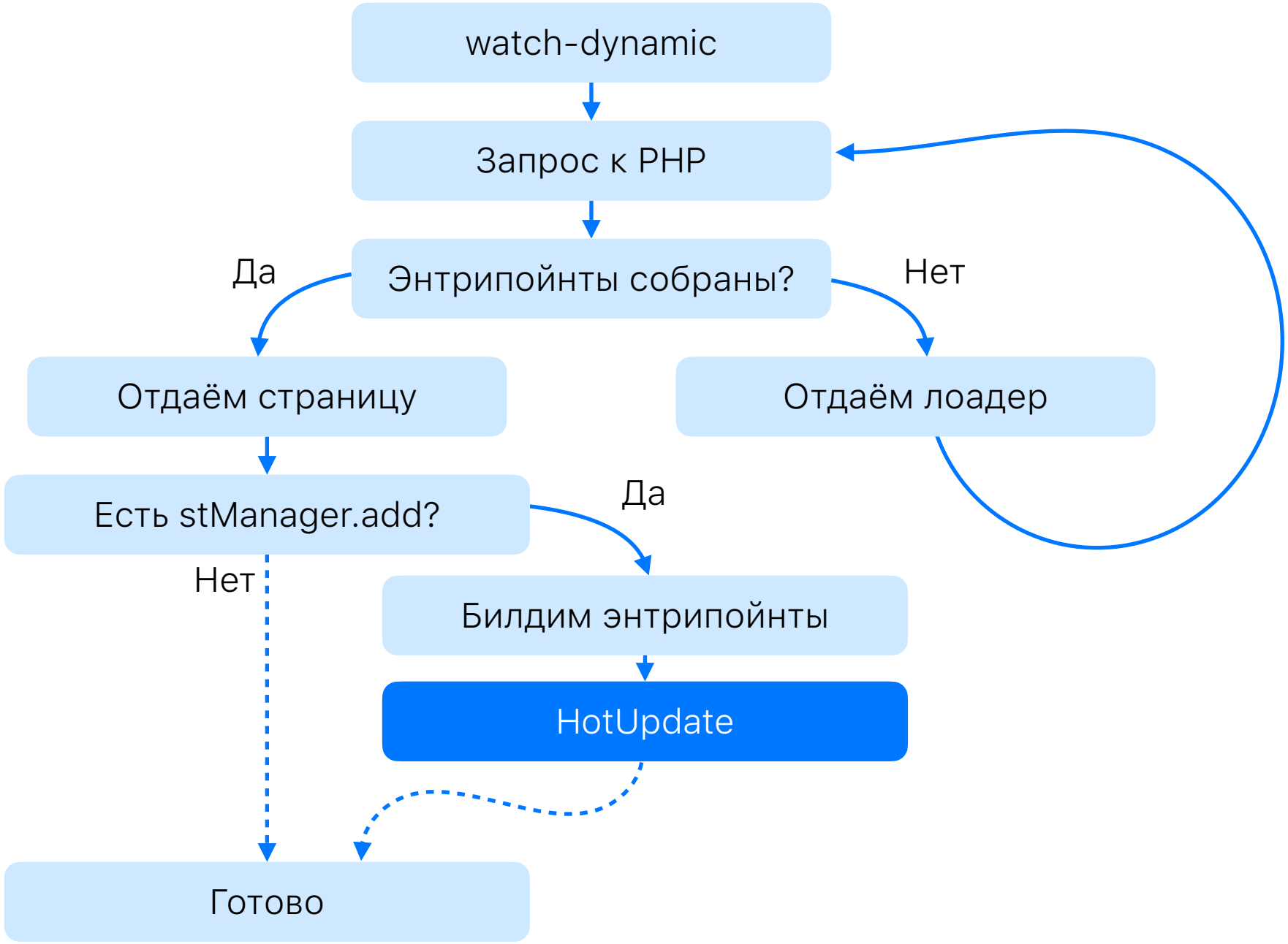

watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

DynamicStaticManager

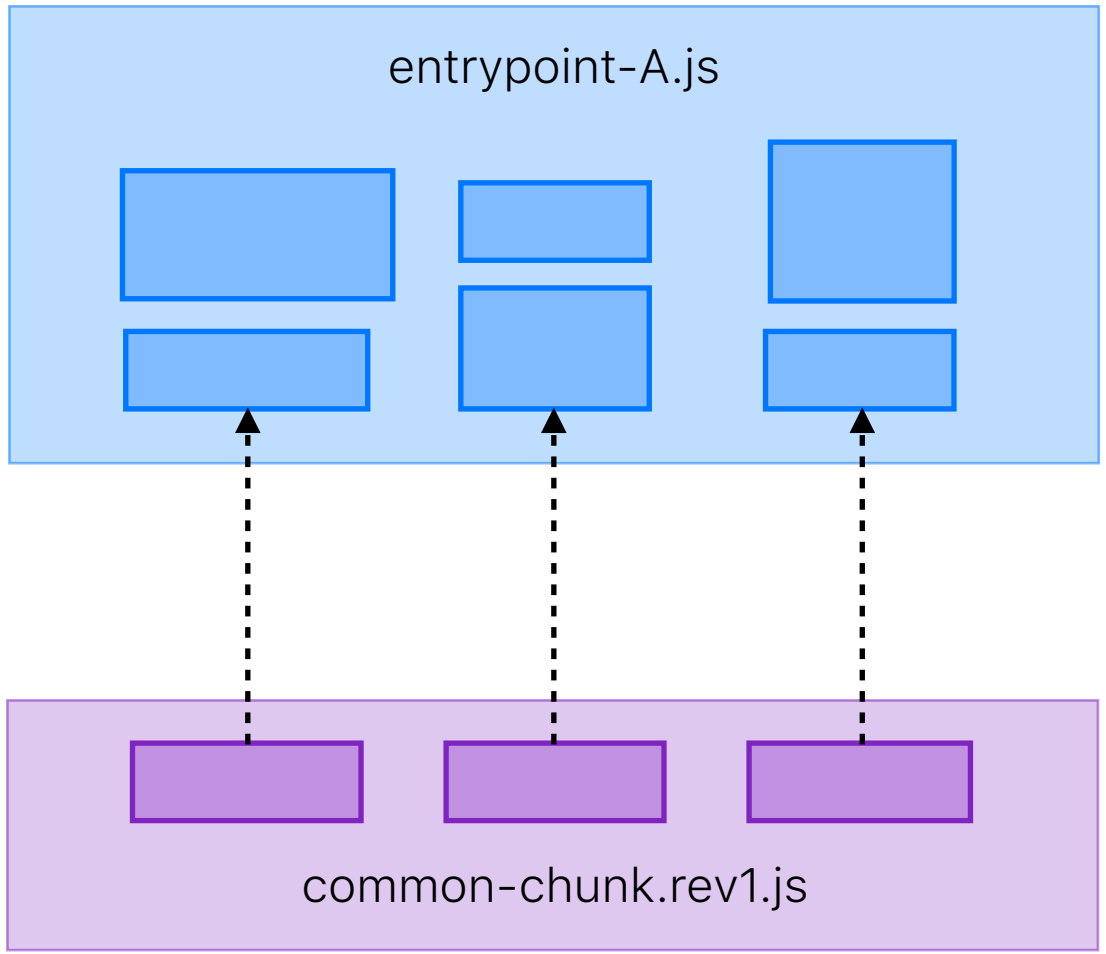


HMR

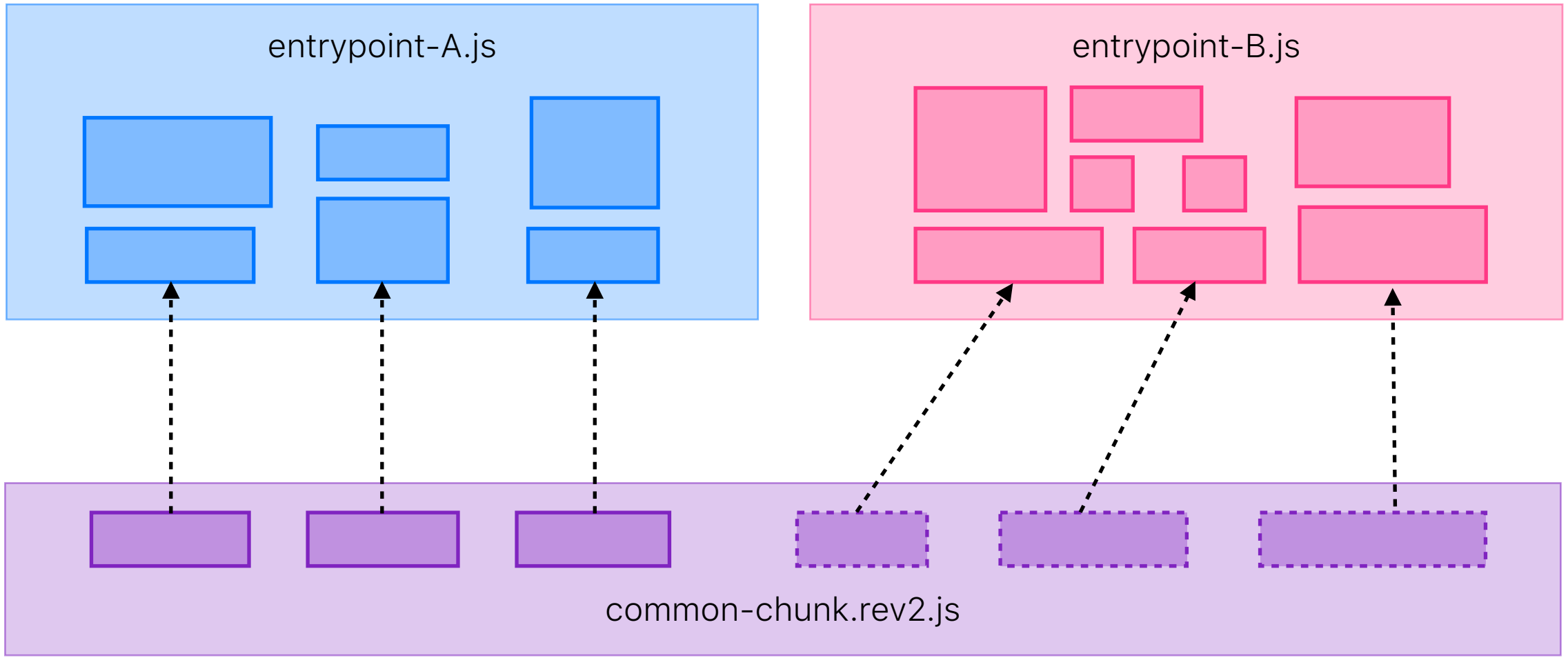
(списали, но
не точь-в-точь)


watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

HMR



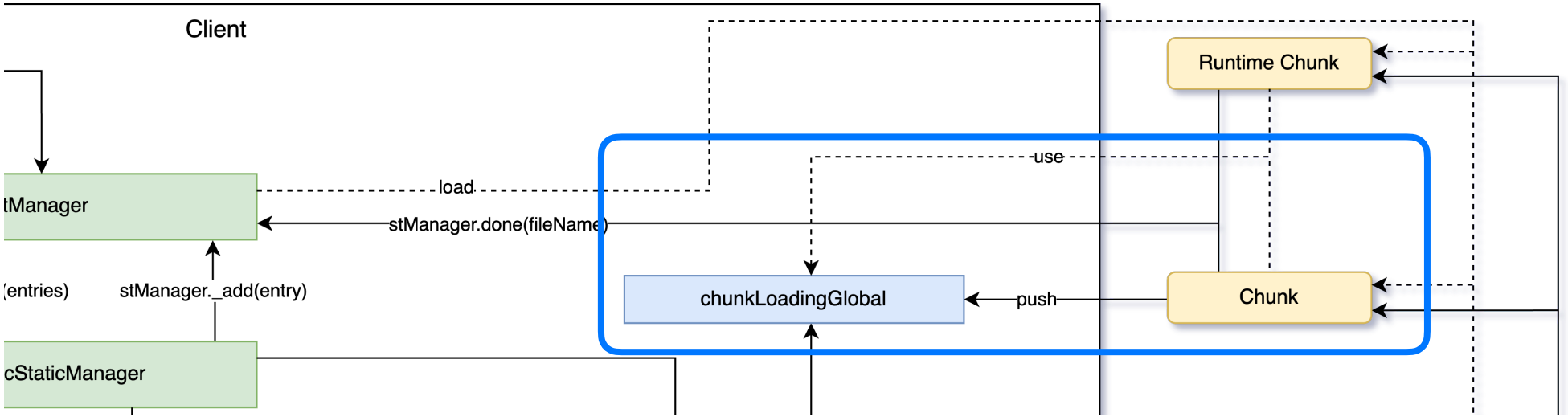
HMR



HMR

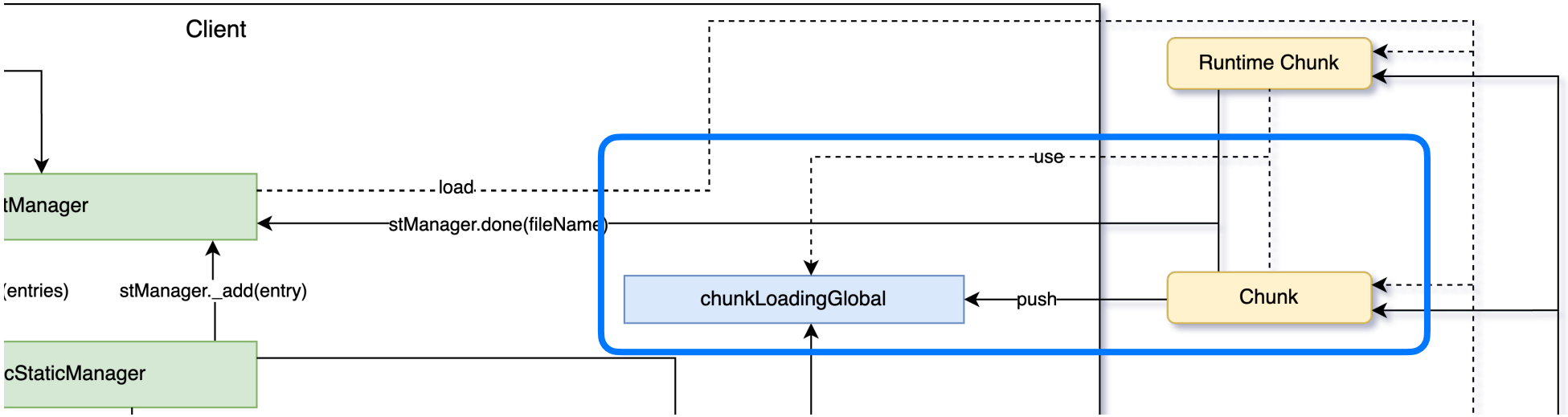
- Не работает со множественными рантаймами
- Перезапускает рантайм после каждого обновления
- Его никак не подружить со **static_manager.js**

HMR




```
(self['chunkLoadingGlobal'] = self['chunkLoadingGlobal'] || []).push([['web/chunks/react'], {
  './node_modules/prop-types/checkPropTypes.js':
   ((module, __unused_webpack_exports, __webpack_require__) => {
     /* module code */
   }),
  /* ...other modules */
}]);
```

HMR

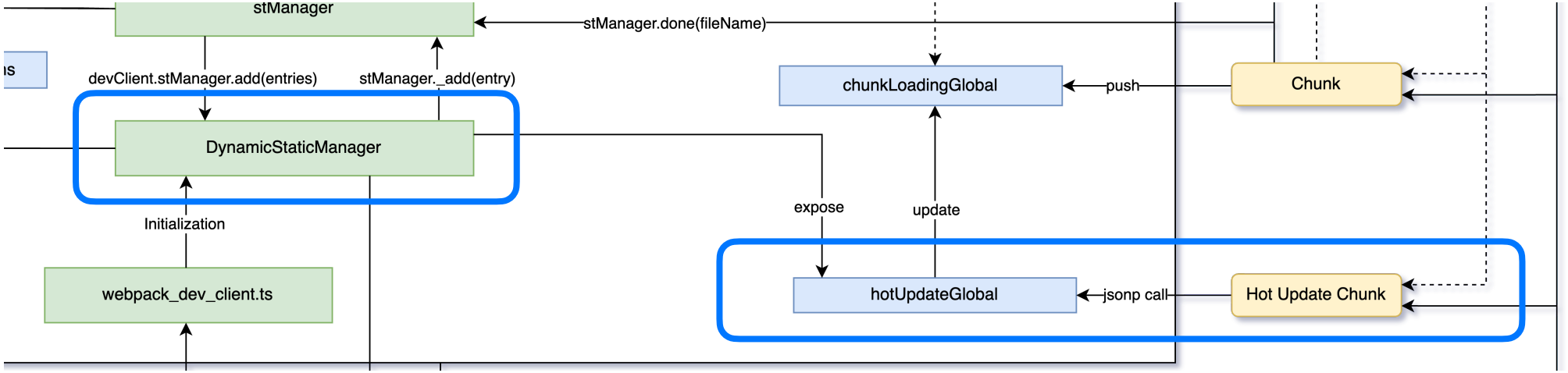



```
window.chunkLoadingGlobal = [
  [
    ['dev/chunks/dev-client-modules'],
    {
      './node_modules/@swc/helpers/esm/_async_to_generator.js':
        (__unused_webpack___webpack_module__, __webpack_exports__, __webpack_require__) => { /* module code */ },
      './node_modules/@swc/helpers/esm/_extends.js':
        (__unused_webpack___webpack_module__, __webpack_exports__, __webpack_require__) => { /* module code */ }
    }
  ],
  /* ...rest chunks */
]
```

HMR

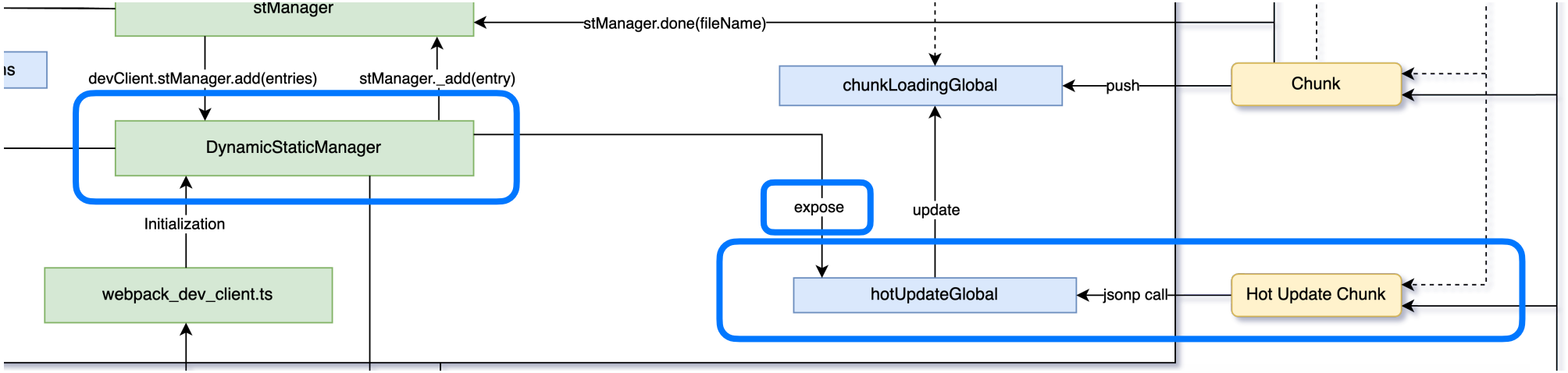
```
const installedChunks = {
  'some_chunk_name': 0
};

const webpackJsonpCallback = (data) => {
  const [chunkIds, moreModules] = data;

  if (chunkIds.some((id) => (installedChunks[id] !== 0))) {
    for (let moduleId in moreModules) {
      __webpack_require__.moduleFactories[moduleId] = moreModules[moduleId];
    }
  }
}

const chunkLoadingGlobal = window['chunkLoadingGlobal'] || [];
chunkLoadingGlobal.forEach(webpackJsonpCallback);
chunkLoadingGlobal.push = webpackJsonpCallback;
```

HMR



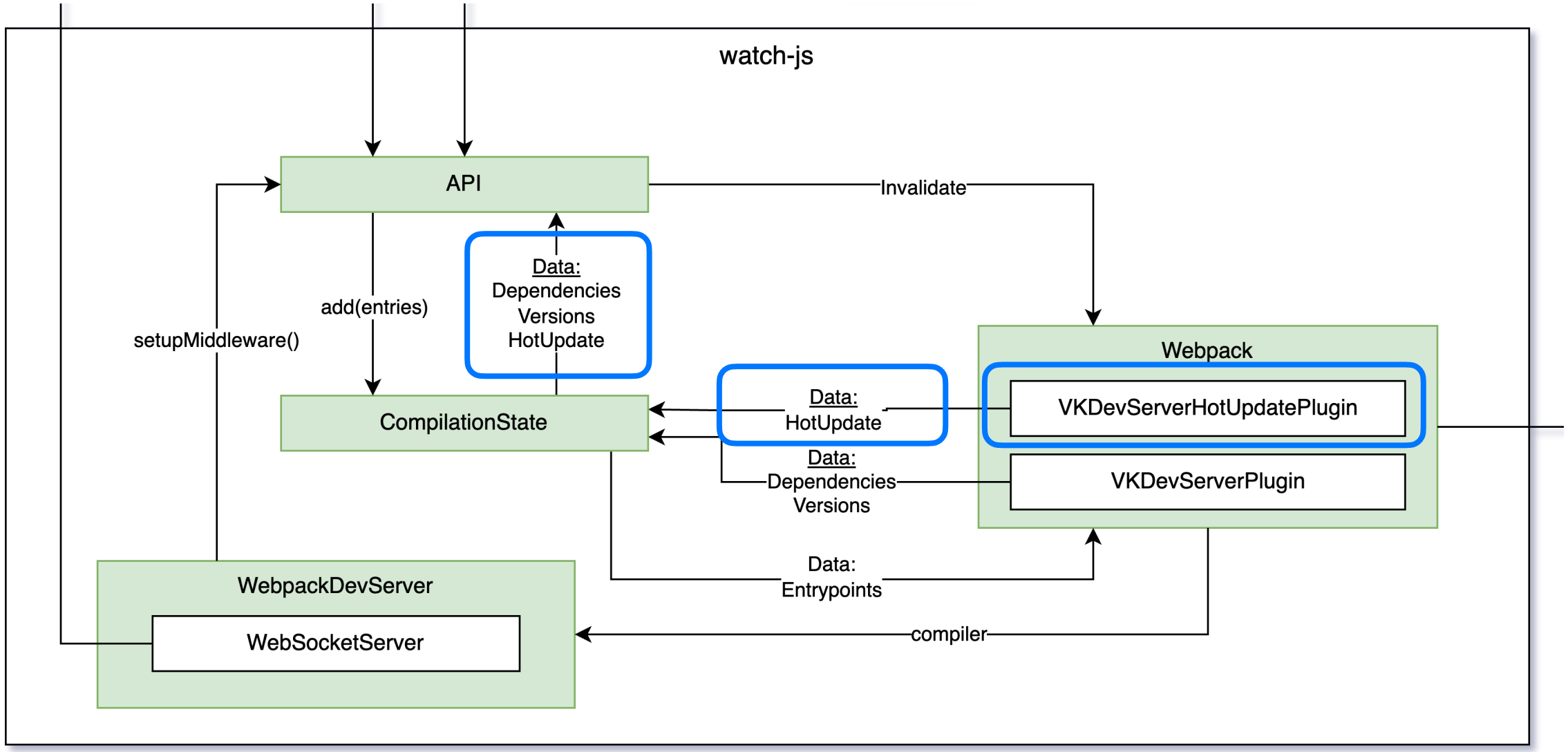

```
self['hotUpdateGlobal']('web/chunks/react', {
  './node_modules/react-dom/cjs/react-dom-server-legacy.node.development.js':
    ((__unused_webpack_module, exports, __webpack_require__) => {
      /* module code */
    }),
  /* ...other modules */
})
```

HMR

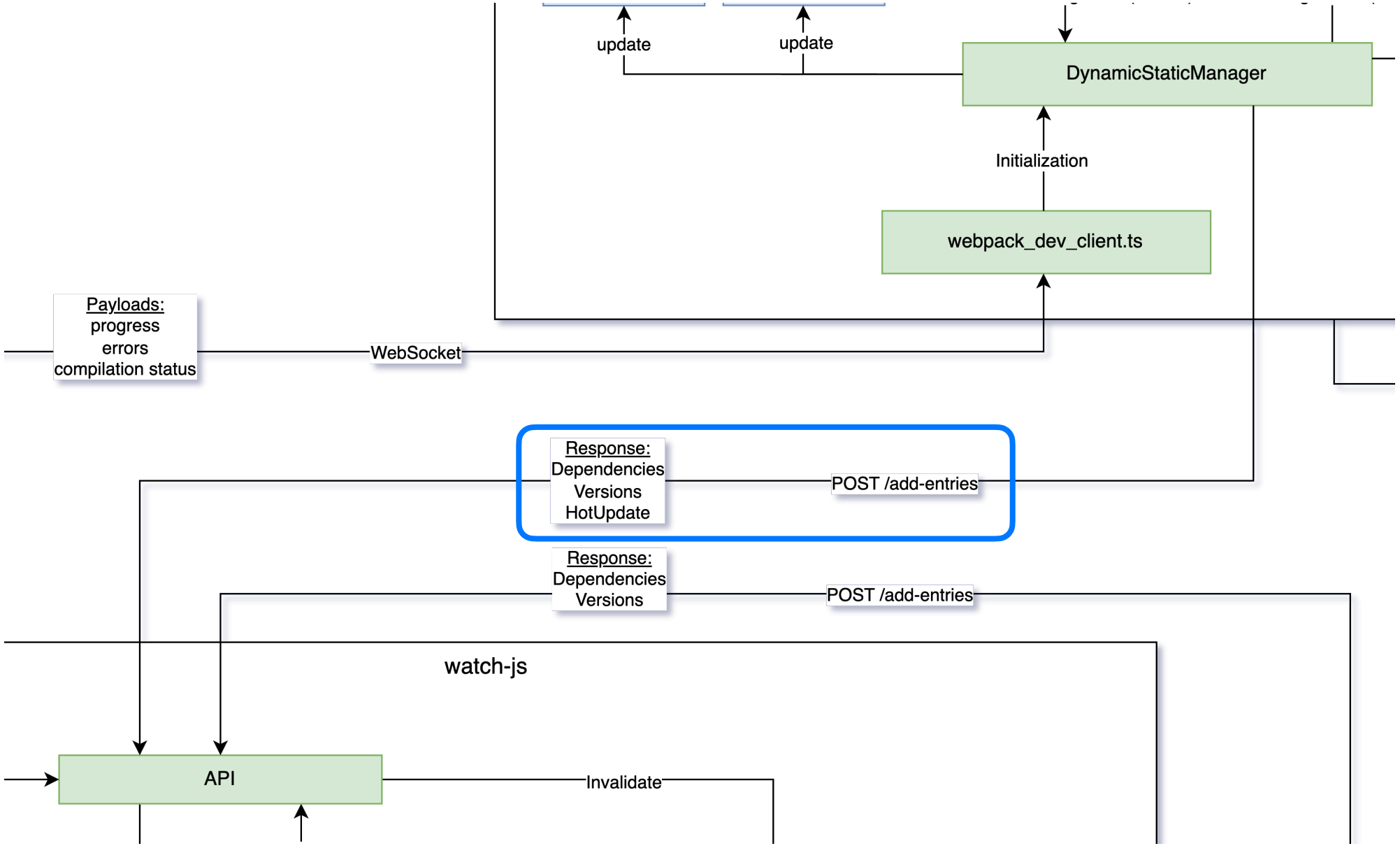



```
window['hotUpdateGlobal'] = (chunkId, source) => {
  const chunksGlobal = window['chunkLoadingGlobal'];
  const existedChunk = chunksGlobal.find(([[chunkName]]) => chunkName === chunkId);

  if (existedChunk) {
    const [, modules] = existedChunk;

    Object.entries(source).forEach(([moduleName, moduleFactory]) => {
      modules[moduleName] = moduleFactory;
    });
  }
};
```

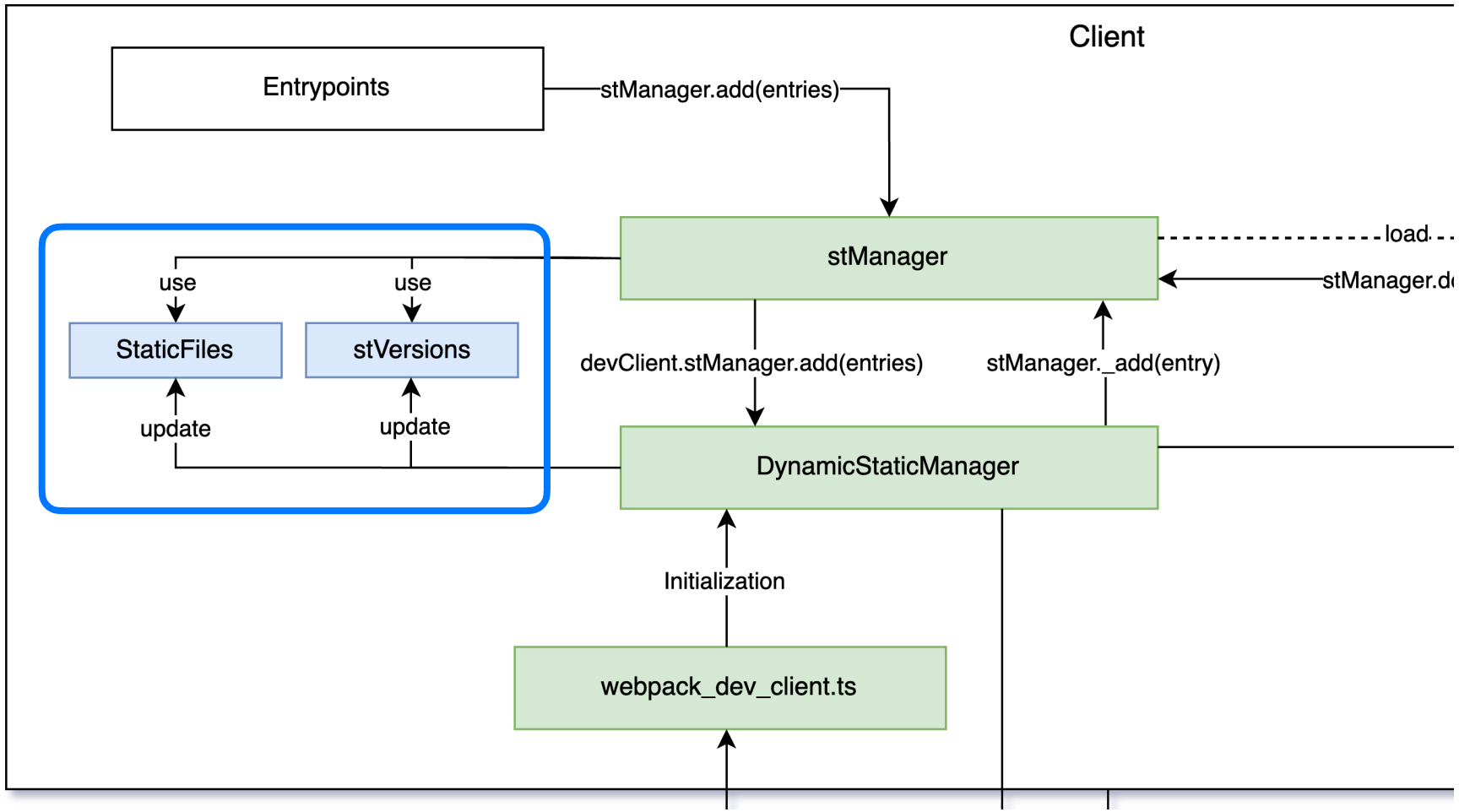
HMR


watch-js
API
Invalidate
add(entries)
setupMiddleware()
Data:
Dependencies
Versions
HotUpdate
Webpack
VKDevServerHotUpdatePlugin
Data:
HotUpdate
CompilationState
Data:
Dependencies
Versions
VKDevServerPlugin
Data:
Entrypoints
WebpackDevServer
WebSocketServer
compiler

HMR

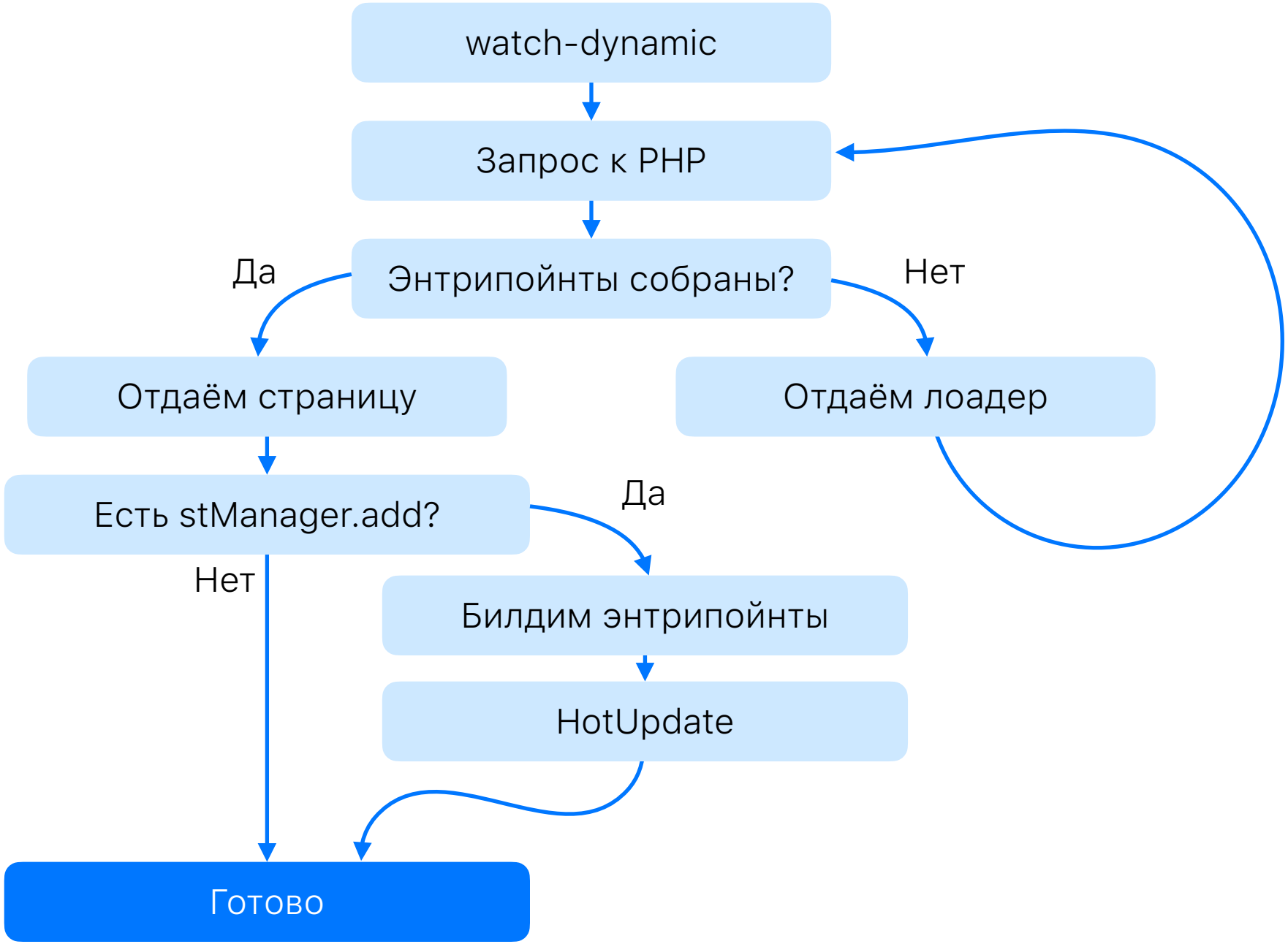

update
update
DynamicStaticManager
Initialization
webpack_dev_client.ts
Payloads:
progress
errors
compilation status
WebSocket
Response:
Dependencies
Versions
HotUpdate
POST /add-entries
Response:
Dependencies
Versions
POST /add-entries
watch-js
API
Invalidate

HMR


Client
Entrypoints
stManager.add(entries)
stManager
load
stManager.d
use
use
StaticFiles
stVersions
update
update
devClient.stManager.add(entries)
stManager._add(entry)
DynamicStaticManager
Initialization
webpack_dev_client.ts

HMR

Было сложно, но мы таки списали


watch-dynamic
Запрос к PHP
Энтрипойнты собраны?
Да
Нет
Отдаём страницу
Отдаём лоадер
Есть stManager.add?
Да
Нет
Билдим энтрипойнты
HotUpdate
Готово

После полной загрузки

Можно переходить
на другие страницы
и добавлять в сборку
новые энтрипойнты.

Профиты

- Не нужно перезапускать **watch**, чтобы собрать что-то новое
- Сборка будет бесшовной для всего проекта

После полной загрузки

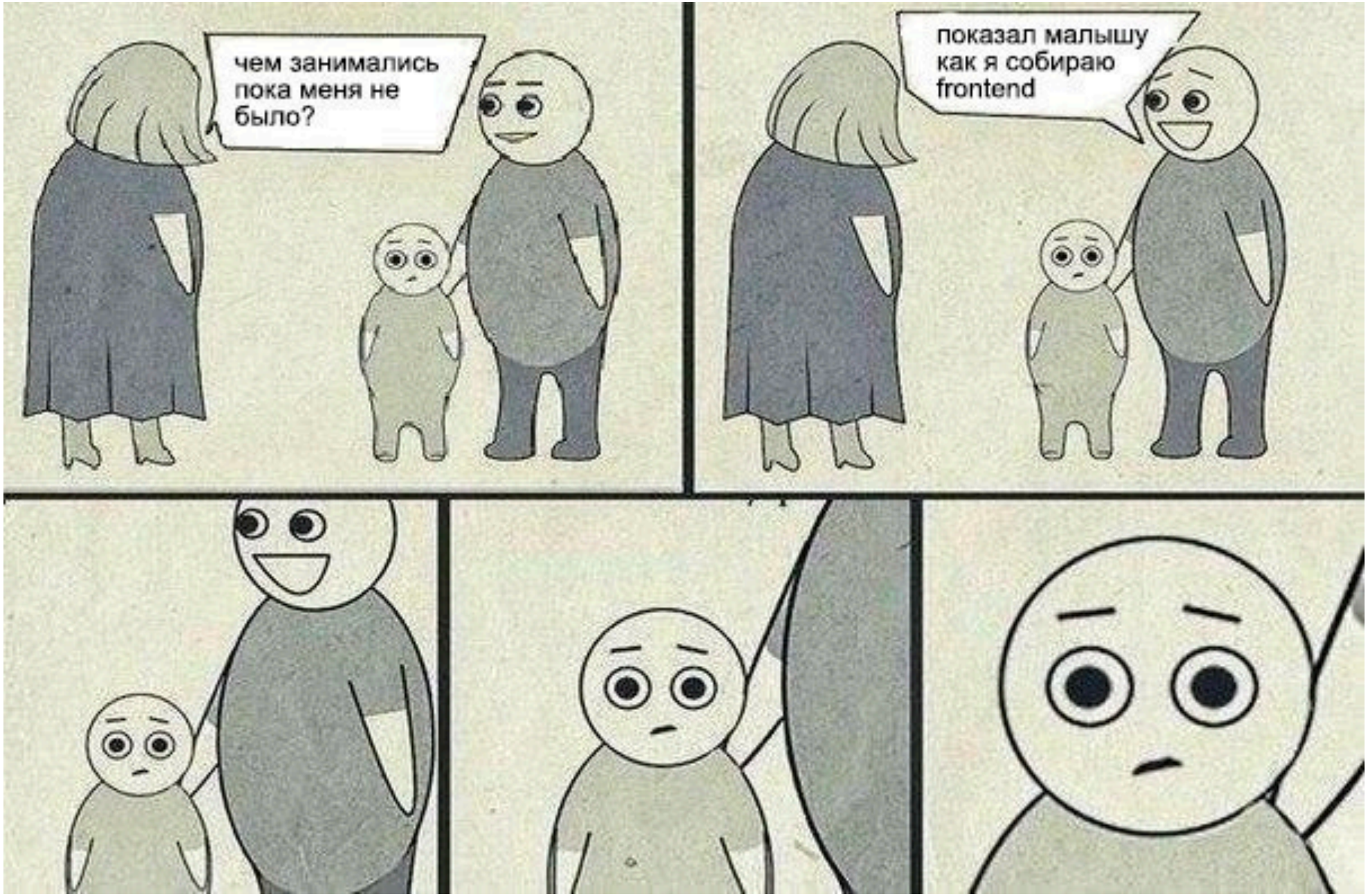
Можно сохранить набор энтрипойнтов и впоследствии его реиспользовать при повторном запуске watch.

Профиты

- Ускорение инициализации
- Не нужно дожидаться окончания докидываний энтрипойнтов и хот-апдейтов
- Получение сразу готовой для работы страницы

Главные итоги

	build + watch	watch-dynamic
Холодный запуск	530s + 80s	120s
Горячий запуск	350s + 40s	80s
Ребилд	20s	12s
Конфигурация	.env	auto


чем занимались пока меня не было?
показал малышу как я собираю frontend

Спасибо
за внимание!