Д.А. Новиков, Е.В. Симонова

АВТОМАТИЗАЦИЯ СБОРКИ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ

(Самарский национальный исследовательский университет
имени академика С.П. Королева)

Введение

В связи с развитием веб-технологий происходит усложнение структуры веб-приложений. Современные веб-приложения могут содержать сотни зависимостей от других библиотек, фреймворков, подпроектов. В свою очередь, каждая такая зависимость может включать в себя другие зависимости. Основной проблемой является то, что разные части веб-приложения содержат одинаковые зависимости, из-за чего происходит дублирование кода. Еще одной проблемой является обновление этих зависимостей. Часто встречается ситуация, когда одно и то же веб-приложение работает на одном персональном компьютере и не работает на другом. Это происходит из-за установки разных версий зависимостей у веб-приложения. Подобные проблемы решают менеджеры пакетов.

Следствием усложнения структуры является увеличение количества и размеров файлов, используемых в веб-приложении. Конечный пользователь не хочет долго ждать загрузки веб-страницы с веб-приложением и скачивать тысячи файлов, необходимых для его работы. Задачу по сборке веб-приложений решают так называемые сборщики проектов.

Подключение зависимостей с помощью ссылок на внешние скрипты

Изначально для подключения каких-либо зависимостей в веб-приложение использовались ссылки на внешние скрипты. Для каждой зависимости в разметку веб-приложения добавлялся тег <script>, в атрибуте src которого указывался путь до исполняемого файла зависимости.

Такой подход позволяет использовать одни и те же общие функции на разных веб-страницах веб-приложения и ускоряет их загрузку, т.к. внешний файл кэшируется в браузере при первой загрузке и скрипт вызывается быстрее при следующих вызовах.

Недостатки такого подхода:

- отсутствует какой-либо контроль за зависимостями;
- отсутствует версионирование зависимостей;
- нет возможности автоматического обновления зависимостей;
- код повторяется при пересечении зависимостей в разных частях веб-приложения;
- отсутствует возможность установки зависимостей из облака.

Все манипуляции с зависимостями осуществляются вручную. Это привело к созданию автоматизированной системы контроля зависимостей – менеджеру пакетов. Рассмотрим два менеджера пакетов – npm и Bower на основе программной платформы Node.js.

Node.js – программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Это асинхронная событийная среда выполнения, предназначенная для создания масштабируемых сетевых приложений на JavaScript [1].

Менеджеры пакетов в веб-приложениях

1) npm

npm (node package manager) – менеджер пакетов, автоматически устанавливающийся вместе с Node.js.

При инициализации npm в веб-приложении будет создан файл package.json, который содержит в себе информацию о приложении: название, версия, зависимости и т.п. Любая директория, в которой есть этот файл, интерпретируется как Node.js-пакет [2].

npm предоставляет следующие возможности:

- скачивание и установка пакетов из облачного сервера npm;
- предоставление данных по текущим зависимостям в веб-приложении;
- автоматическое обновления пакетов;
- публикация пакета.

Однако npm не решает проблему дублирования зависимостей в разных частях веб-приложения, так как npm устанавливает зависимости для каждого пакета отдельно, в итоге получается большое дерево пакетов (node_modules/grunt/node_modules/glob/node_modules/…), где может быть несколько версий одного и того же пакета. Это недопустимо: нельзя подключить в одно приложение две версии jQuery или любой другой библиотеки.

2) Bower

Главное отличие npm от Bower – подход к установке зависимостей пакетов. В Bower каждый пакет устанавливается один раз (jQuery всегда будет в папке bower_components/jquery, сколько бы пакетов от него не зависело). В случае конфликта зависимостей Bower не станет устанавливать пакет, не совместимый с ранее установленными.

Например, если создать новое веб-приложение и установить пару зависимостей, дерево зависимостей будет выглядеть следующим образом (рисунок 1). Т.к. все пакеты зависят от jQuery, Bower смог найти подходящую всем версию – jQuery 2.1.0.

```
bowertest#0.0.0 /Users/admin/bowertest
├┬ fotorama#4.5.1
│ └── jquery#2.1.0 (2.1.1-beta1 available)
├── jquery#2.1.0 (2.1.1-beta1 available)
├┬ jquery-icheck#1.0.2
│ └── jquery#2.1.0 (2.1.1-beta1 available)
└┬ social-likes#3.0.2
  └── jquery#2.1.0
```

Рисунок 8 – Дерево зависимостей веб-приложения

Дерево файловой системы проекта представлено на рисунке 2.

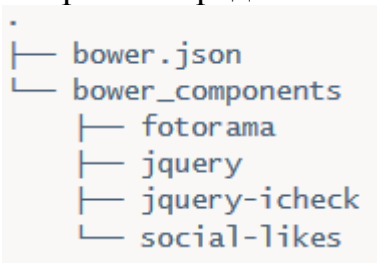

Рисунок 2 – Дерево файловой системы веб-приложения

Каждый пакет устанавливается в свою папку, вложенных пакетов нет, jQuery встречается только один раз.

Сборщики проектов в веб-приложениях

Сборщики проектов осуществляют минификацию и конкатенацию файлов веб-приложения, значительно уменьшая их итоговый размер и количество файлов. Они также способны при сборке учитывать неиспользуемые зависимости и не включать их в готовое веб-приложение.

Минификация – процесс, направленный на уменьшение размера исходного кода путём удаления ненужных символов без изменения его функциональности. Сокращение исходного кода является особенно полезным для программ на интерпретируемых языках, которые развернуты и передаются в интернете, так как это уменьшает объем передаваемых данных. При минификации удаляются переводы строк, комментарии, лишние отступы и пробелы, сокращаются имена переменных, за счет этого достигается ускорение загрузки сайта и оптимизация запросов [3].

Рассмотрим два сборщика проектов – Gulp и Webpack.

1) Gulp

Gulp – это потоковый сборщик JavaScript проектов. Достоинства Gulp [4]:

- автоматизация – gulp позволяет м автоматизировать трудоемкие и длительные задачи в процессе разработки веб-приложения;
- интегрируемость – gulp встроен во многие IDE и может быть использован для автоматизации задач в PHP, .NET, Node.js, Java, и в других платформах;
- сильная экосистема – можно использовать множество npm пакетов;
- простота – gulp очень прост в изучении и использовании, т.к. предоставляет минимальное API.

Возможности Gulp:

- создание веб-сервера и автоматическая перезагрузка страницы в браузере при сохранении кода, слежение за изменениями в файлах проекта;
- использование различных JavaScript, CSS и HTML препроцессоров;
- минификация CSS и JS кода, оптимизация и конкатенация отдельных файлов проекта в один;
- автоматическое создание приставок к названию CSS свойства, которые добавляют производители браузеров для нестандартных свойств;

- управление файлами и папками в рамках проекта – создание, удаление, переименование;
- работа с изображениями – сжатие, создание спрайтов, изменение размера изображения;
- отправка на внешний сервер проекта по FTP, SFTP, Git.

Связывание файлов в веб-приложениях

Webpack – это связыватель модулей (module bundler), объединяющий в себе функционал менеджера пакетов и сборщика проектов.

Webpack создает граф всех зависимостей в веб-приложении. Начальная точка этого графа называется точкой вхождения. Точка вхождения сообщает webpack, откуда начать движение по графу для выполнения связывания. В процессе связывания возможно осуществить конкатенацию и минификацию исходных файлов. В результате связывания получается набор статических ресурсов – js, html, css файлов и изображений, представленных на рисунке 3.

Основным преимуществом Webpack перед Gulp является то, что при связывании он строит граф зависимостей и проходит по нему, отбрасывая неиспользуемые модули и зависимости. С помощью процедуры, называемой Tree Shaking, webpack находит неиспользуемые функции и части кода внутри веб-приложения и отбрасывает их. Все это приводит к значительному уменьшению размера собранного веб-приложения [5].

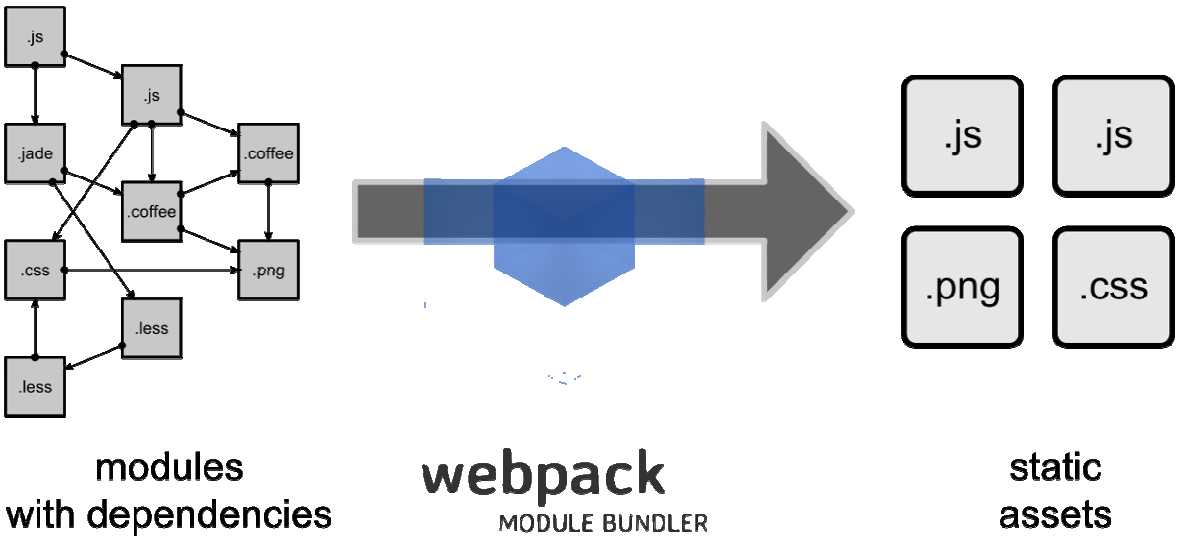

Рисунок 3 – Процесс связывания файлов веб-приложения с помощью Webpack

Заключение

В статье рассмотрены основные решения по управлению зависимостями и автоматизации сборки современных веб-приложений. Современные технологии позволяют автоматизировать процессы сборки, минификации, конкатенации веб-приложения, уменьшить размер веб-приложения, отбрасывая неиспользуемые зависимости и части кода, что уменьшает время загрузки веб-приложений по сети и увеличивает скорость их работы.

Литература

1. Node.js. – Режим доступа: https://nodejs.org/en/
2. npm. – Режим доступа: https://www.npmjs.com/
3. Минификация. – Режим доступа: https://ru.wikipedia.org/wiki/Минификация
4. Gulp. – Режим доступа: http://gulpjs.com/
5. Webpack. – Режим доступа: https://webpack.github.io/

А.М. Ольшанский, А.В. Игнатенков

О НЕКОТОРЫХ СВОЙСТВАХ МНОГОСЛОЙНОЙ ИСКУССТВЕННОЙ НЕЙРОННОЙ СЕТИ С ПЕРЕМЕННОЙ ПРОВОДИМОСТЬЮ СИГНАЛА

(ФГБОУ ВПО «Самарский государственный университет путей сообщения, ОАО «Научно-исследовательский и проектно-конструкторский институт информатизации, автоматизации и связи на железнодорожном транспорте»)

Рассмотрим многослойную искусственную нейронную сеть с переменной проводимостью сигнала (далее – сеть), на вход которой поступает определенный вектор и на выходе снимается значение ответа сети. Сеть служит для построения графика движения поездов на двухпутном участке железнодорожной сети с рядом ограничений [3]. Цель настоящей работы - проанализировать некоторые формальные свойства указанной сети. Одной из предпосылок к созданию этой работы явился нестационарный и нелинейный характер поведения функции ошибки сети (см. рис. 1-2).

Замечено, что её поведение может быть описано следующим образом:

1. Автокорреляционная функция первого порядка свидетельствует о том, что ярко выраженного тренда нет, незначительное преобладание сигнала с периодом 3 единицы и наличие ряда уровней корреляции в районе 0.5-0.562 не позволяет сделать вывод о выраженной периодичности сигнала ошибки сети. Наличие уровней корреляции, которые по своей величине превышают уровень, которым можно пренебречь, не позволяет подтвердить предположение о марковском характере случайного процесса.
2. Периодограмма на рис. 2 также содержит в себе значительное количество частот с высоким значением спектральной плотности, что говорит о том, что нет ярко преобладающих гармоник, объясняющих большую часть периодической составляющей ряда.

Таким образом, налицо нелинейный хаотический характер поведения функции ошибки сети.

Авторы полагают, что подобный режим образовался в силу следующих положений. Поведение функции ошибки сети в общем зависит от поведения следующих её элементов:

1. Нейроны в каждом слое
2. Связи нейронов между слоями сети и между нейронами одного слоя