rollup.js

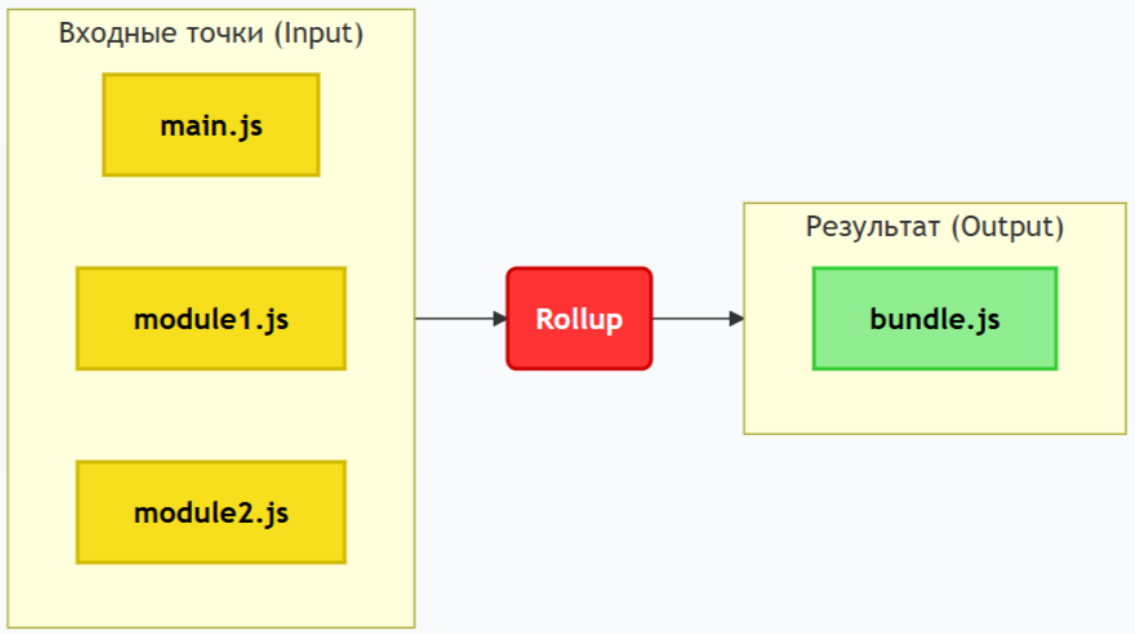
Зинкин Станислав, Усманов Игорь
гр. 5040102/50201

Сборщик модулей (Module Bundler)

Что это: инструмент, который объединяет JS-файлы и зависимости в один или несколько файлов (бандлов)

Зачем нужен:

- Упрощает структуру проекта
- Оптимизирует код для браузера
- Управляет зависимостями между модулями




```
1 // file1.js
2 export function greet(name) {
3   return `Hello, ${name}!`;
4 }
5
6 // file2.js
7 import { greet } from './file1.js';
8 console.log(greet('World'));
```

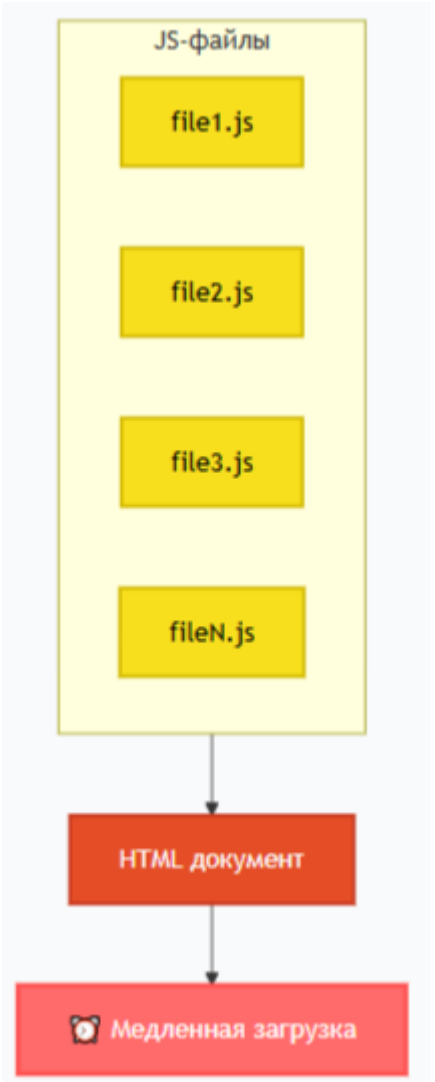
Без сборщика: каждый файл подключается отдельно. Сборщик объединяет их в один бандл для браузера.

Проблемы при работе без сборщика

- Несколько скриптов на сайте → трудно поддерживать
- Медленная загрузка страниц
- Конфликты зависимостей между файлами

```
// file1.js
function greet() { return "Hello!"; }

// file2.js
function greet(name) { return "Hello, " + name; }

// file3.js
console.log(greet("World"));
```

Конфликт функций и порядок загрузки файлов важен. Без сборщика легко допустить ошибки

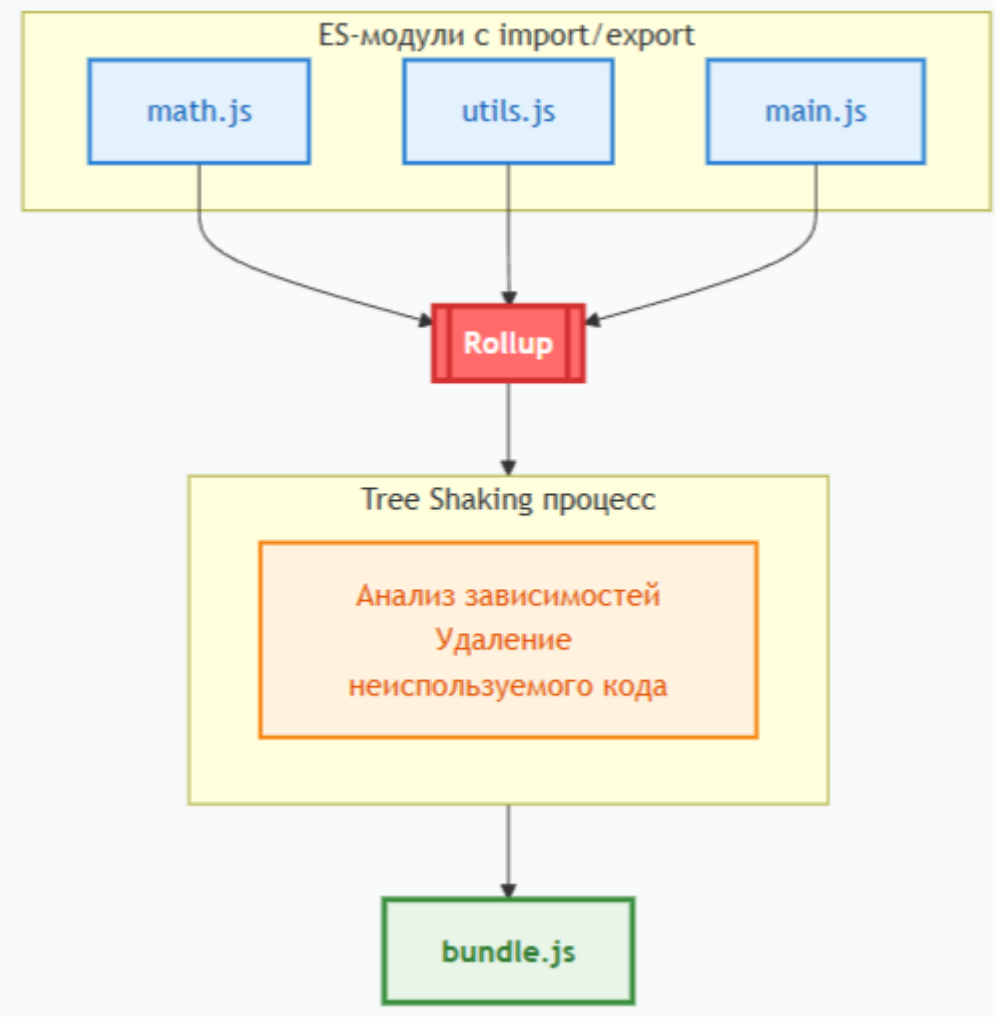


Rollup — современный сборщик модулей для JavaScript

- Использует ES-модули (import/export)
- Фокус на tree shaking — удаляет неиспользуемый код
- Идеален для библиотек и небольших проектов

```
1 // file1.js
2 export function add(a, b) { return a + b; }
3
4 // file2.js
5 export function multiply(a, b) { return a * b; }
6
7 // file3.js
8 import { add } from './file1.js';
9 console.log(add(2, 3));
```

Rollup объединяет модули, удаляя неиспользуемые.



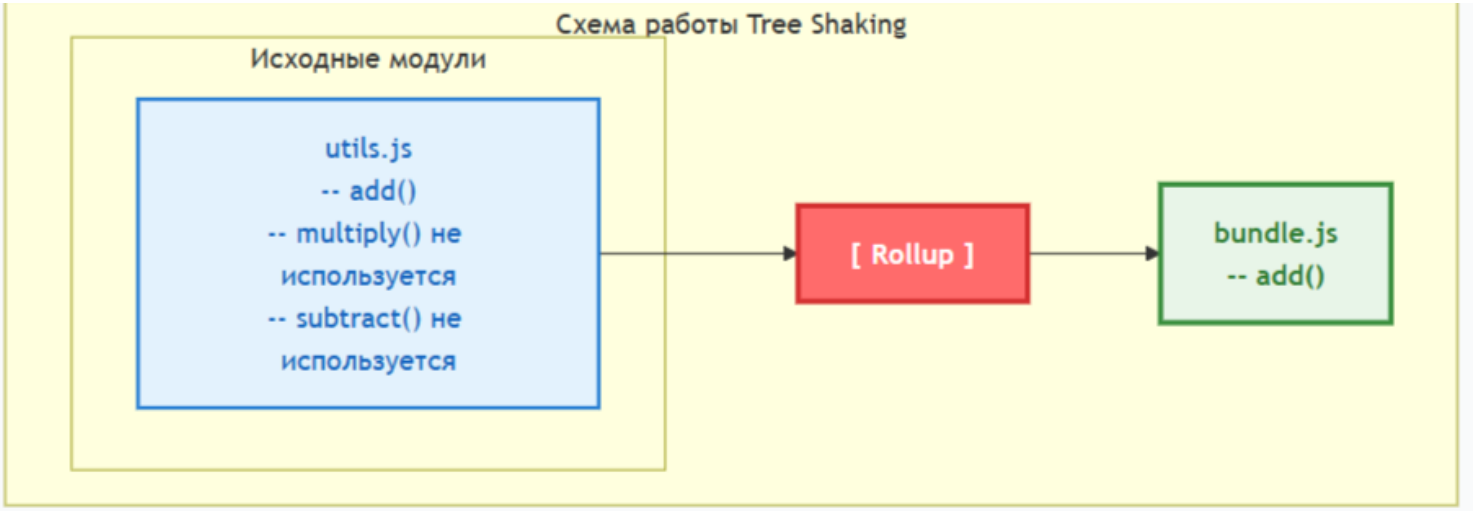
ES-модули (ES6)

- Используют import / export
- Позволяют разбивать код на независимые файлы
- Упрощают управление зависимостями

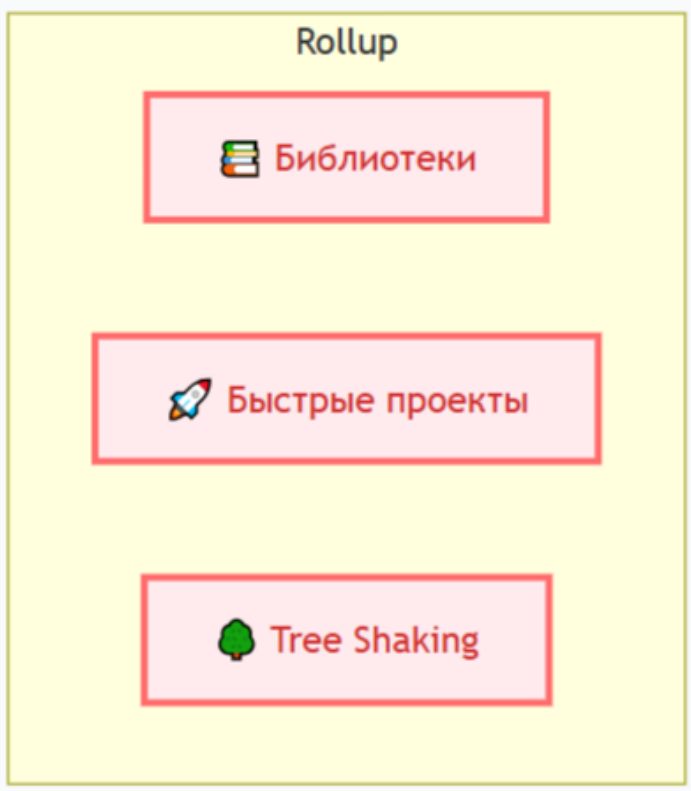
```
// utils.js
export function add(a, b) { return a + b; }
export function multiply(a, b) { return a * b; }

// main.js
import { add } from './utils.js';
console.log(add(2, 3));
```

Tree Shaking

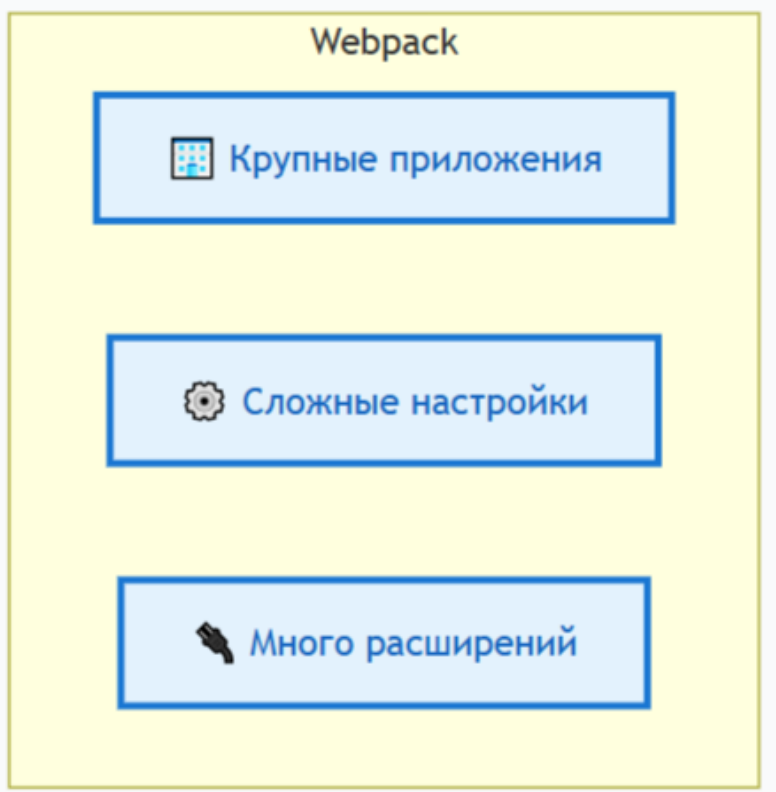
- Rollup удаляет неиспользуемый код
- Итоговый бандл содержит только то, что реально используется



Rollup vs Webpack



Характеристика	Rollup	Webpack
Конфигурация	Простая, минимальная	Более сложная
Фокус	Библиотеки, небольшие проекты	Большие приложения
Tree shaking	Отлично	Есть, но сложнее
Размер бандла	Минимальный	Может быть больше




```
export default {
  input: 'src/main.js',
  output: {
    file: 'bundle.js',
    format: 'esm'
  }
};
```

```
module.exports = {
  entry: './src/main.js',
  output: { filename: 'bundle.js' },
  module: { rules: [] }
};
```

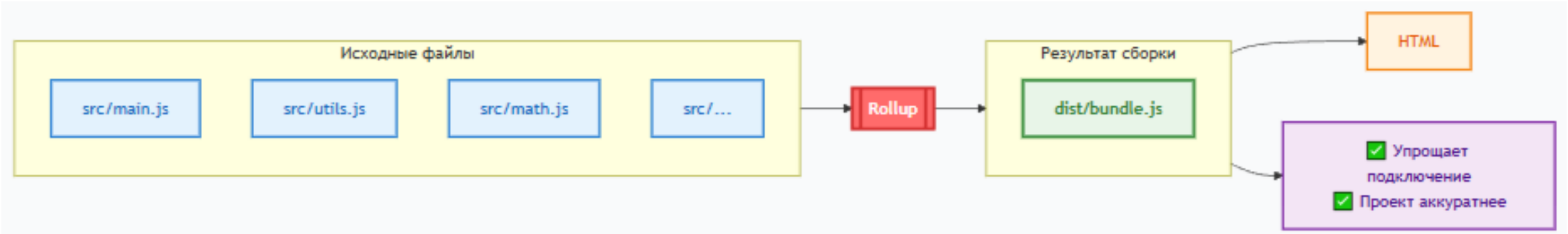
Установка Rollup

- Через npm:

$ npm install --save-dev rollup

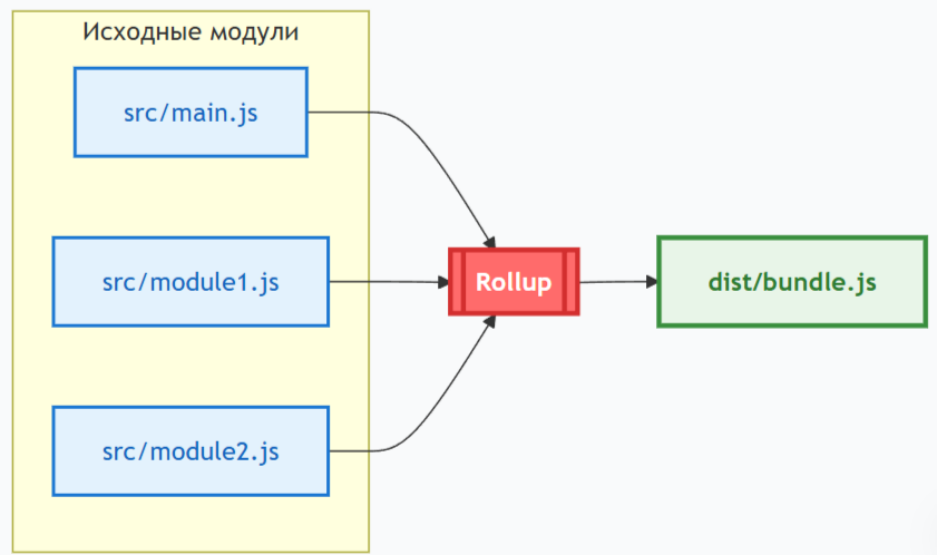
- Базовая конфигурация (rollup.config.js):

```
1 // rollup.config.js
2 export default {
3   input: 'src/main.js',     // точка входа
4   output: {
5     file: 'dist/bundle.js', // файл бандла
6     format: 'esm'           // формат модуля
7   }
8 };
```



Пример работы Rollup

- Проект: 2–3 JS-файла
- Конфигурация Rollup: rollup.config.js
- Сборка: npx rollup -c
- Результат: один бандл JS

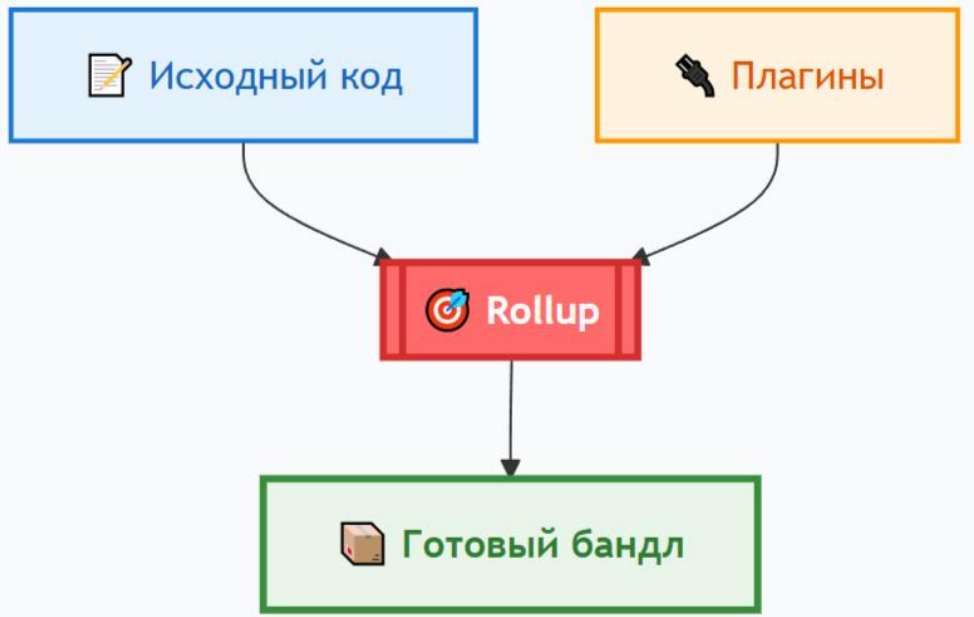



```
// module1.js
export function add(a, b) { return a + b; }

// module2.js
export function multiply(a, b) { return a * b; }

// main.js
import { add } from './module1.js';
console.log(add(2, 3));
```

```
// rollup.config.js
export default {
  input: 'src/main.js',
  output: {
    file: 'dist/bundle.js',
    format: 'esm'
  }
};
```

Плагины Rollup

- Для чего нужны: расширяют возможности Rollup



- Популярные плагины:
  - @rollup/plugin-node-resolve — подключение модулей из node_modules
  - @rollup/plugin-commonjs — поддержка CommonJS
  - rollup-plugin-terser — минификация кода

```
// rollup.config.js
import resolve from '@rollup/plugin-node-resolve';
import commonjs from '@rollup/plugin-commonjs';
import { terser } from 'rollup-plugin-terser';

export default {
  input: 'src/main.js',
  output: {
    file: 'dist/bundle.js',
    format: 'esm'
  },
  plugins: [
    resolve(),
    commonjs(),
    terser()
  ]
};
```

Где используется Rollup?

- Библиотеки JavaScript:
  1. Lodash
  2. D3.js и другие

- Фреймворки и небольшие фронтенд-проекты
  1. Минимальный размер бандла
  2. Чистая структура кода

- Публикация npm-пакетов
  1. ES-модули + CommonJS
  2. Tree shaking из коробки
  3. Оптимизированный итоговый билд

Преимущества Rollup

- Лёгкий и простой в настройке 
- Эффективный tree shaking 
- Отлично подходит для библиотек 
- Чистый и минимальный бандл 

Недостатки Rollup

- Меньше интеграций, чем у Webpack 
- Ограниченный плагин-экосистема 
- Не всегда удобен для больших приложений 

Итоги

- Rollup — простой и эффективный сборщик ES-модулей
- Основные преимущества: минимальный бандл и сильный tree shaking
- Применение: библиотеки, небольшие фронтенд-проекты

Рекомендации

- Rollup — для библиотек и небольших приложений
- Webpack / Vite — для крупных SPA и сложных проектов