JS Webinar #1

Introduction to webpack

NAVEEN PETE

SUNDAY, SEP 10, 2017

Agenda

- Introduction
- Why Build Tools?
- Modularization in Web Apps
- Challenges with Modularization
- Core Purpose of webpack
- Core Concepts
- webpack in Action
- Q & A

Introduction

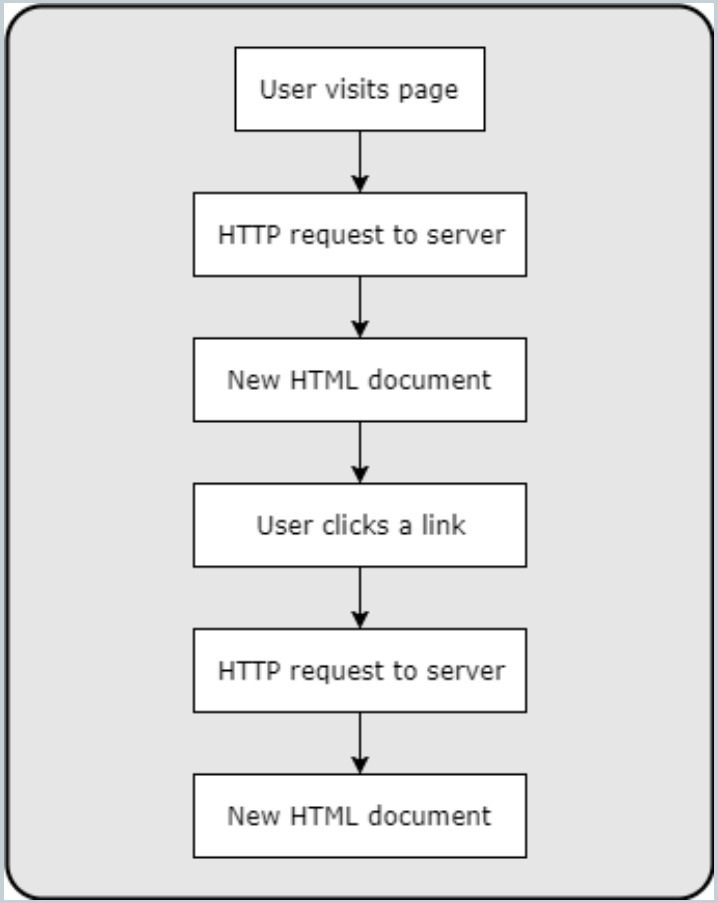
- webpack
  - module bundler for modern JavaScript applications
  - build tool

Why Build Tools?

- Rich web apps
  - Apps that have a lot of dynamic features around them
  - Not static texts, pictures
- Approaches
  - Server Side Templating (SST)
    - Back end server creates an HTML document and sends it to the user
    - HTML document is a fully rendered document, i.e., it has complete information
  - Single Page App (SPA)
    - Server sends a bare-bones HTML document
    - JavaScript runs on the users machine to assemble a full web page

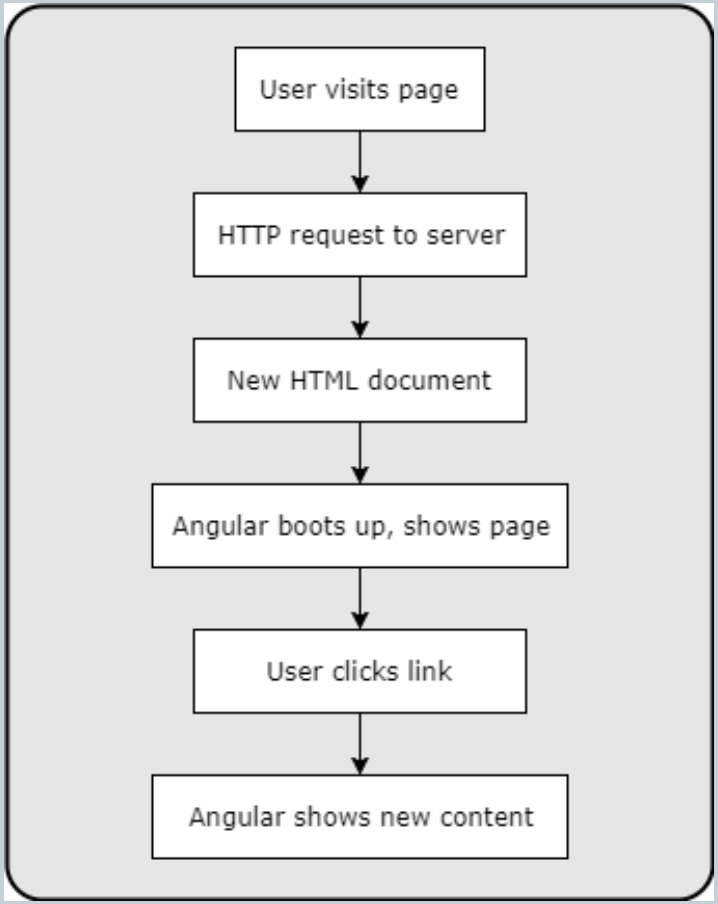
Why Build Tools?

- Server Side Templating (SST)



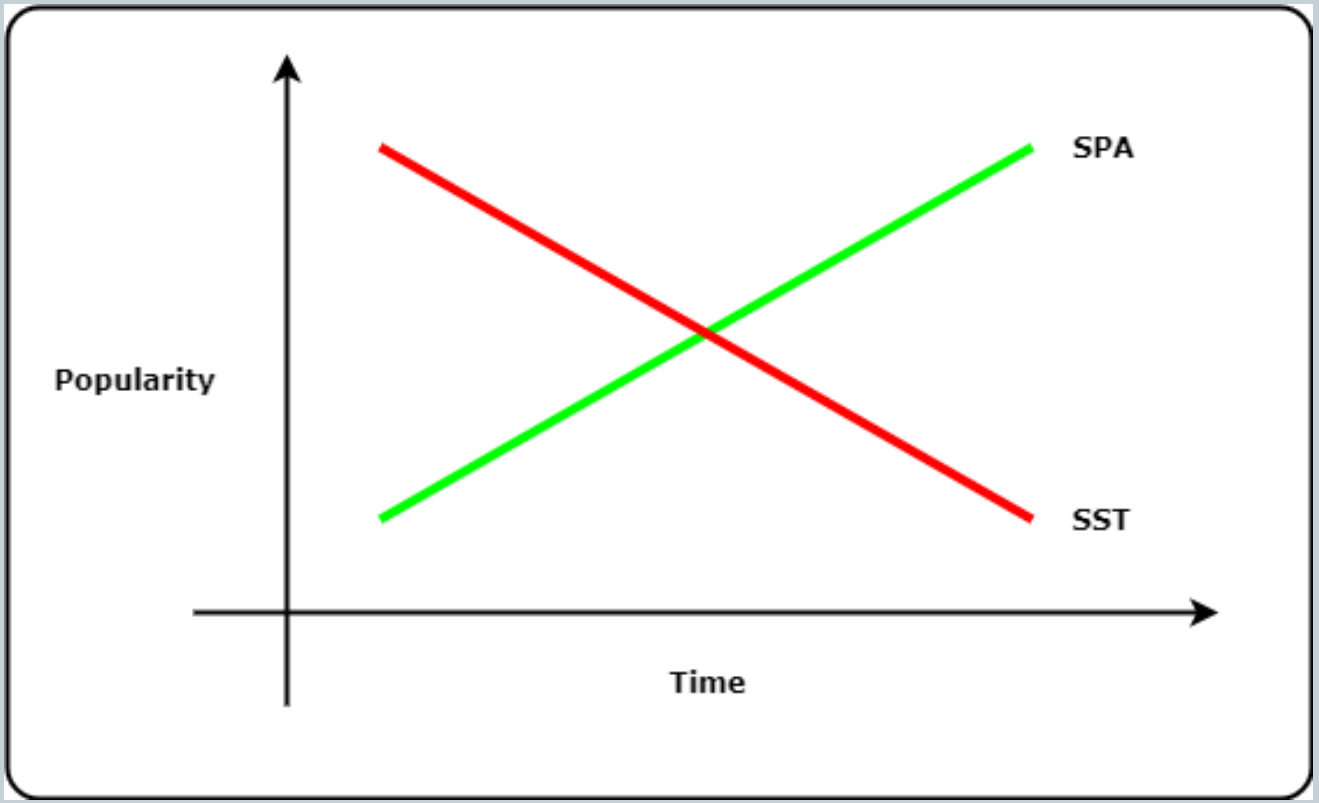
Why Build Tools?

- Single Page App (SPA)



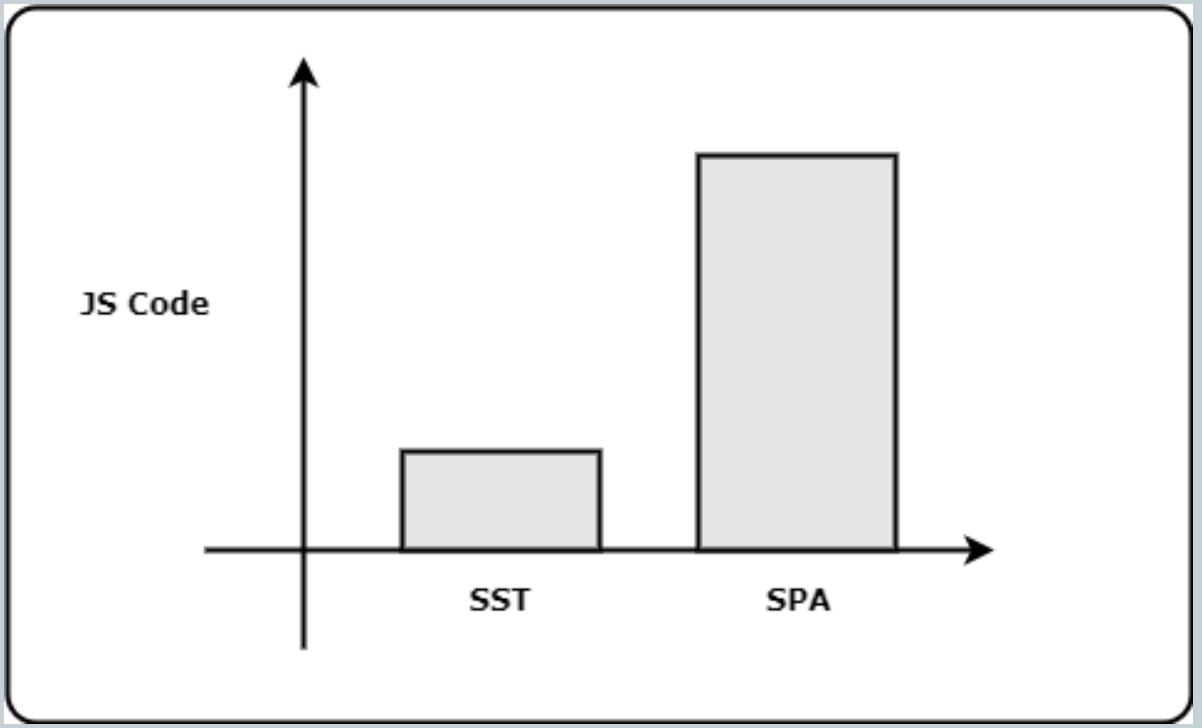
Why Build Tools?

- Popularity – SST vs SPA



Why Build Tools?

- Amount of JS Code



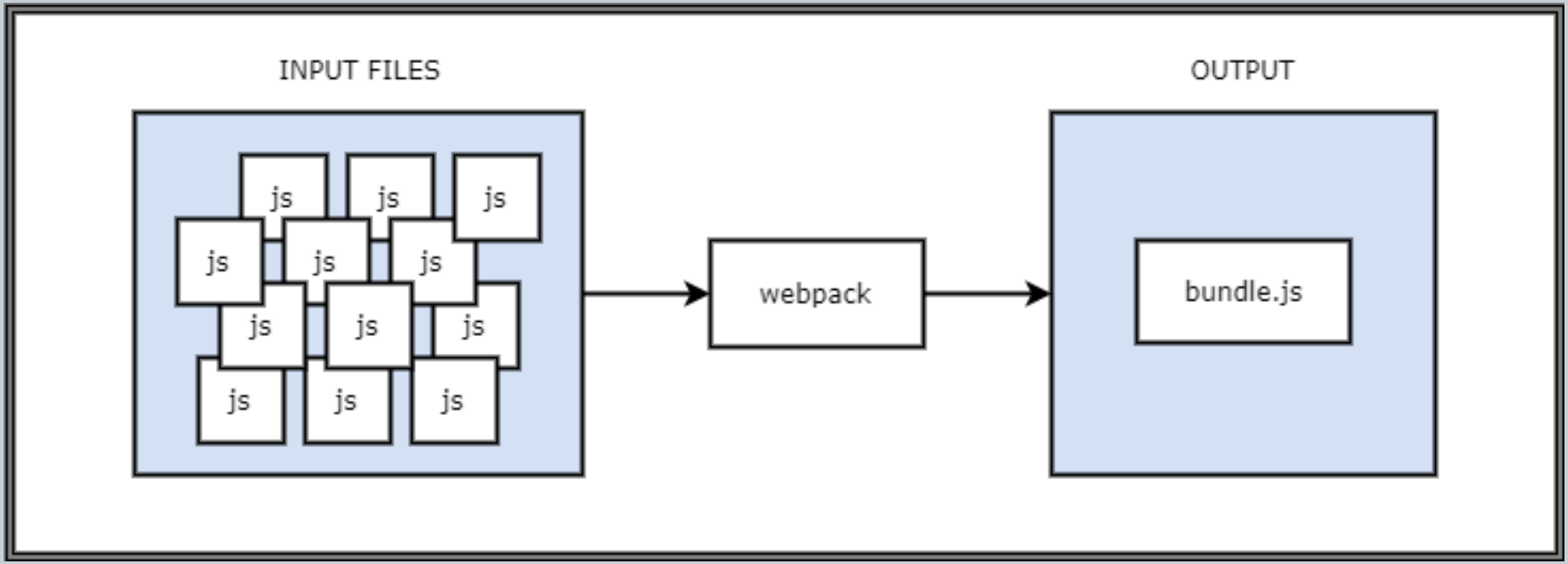
Modularization in Web Apps

- Monolithic app – several thousands of lines of code in few files
  - main.js
  - home.js
  - util.js
- Modular app – organize files into multiple smaller files
  - Components
    - header.js
    - footer.js
    - navigation.js
  - Services
    - users.js
    - products.js

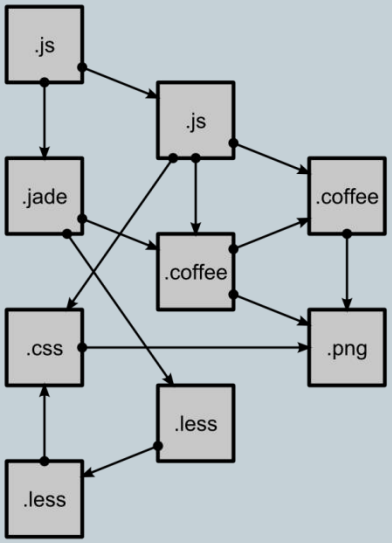
Challenges with Modularization

- Load order
- Performance – load time
  - More files, slower load time of web page
  - Mobile devices

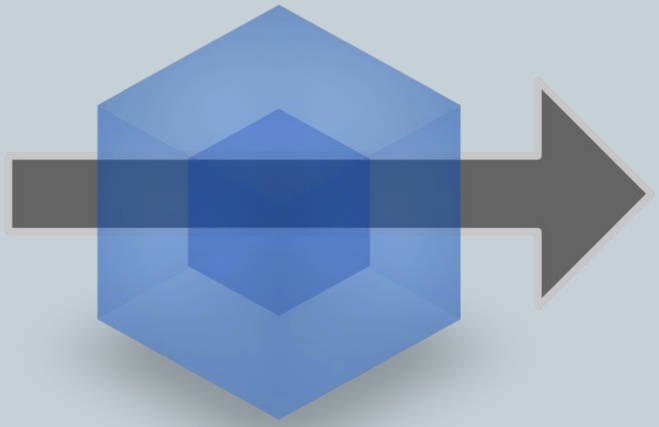
Core Purpose of webpack



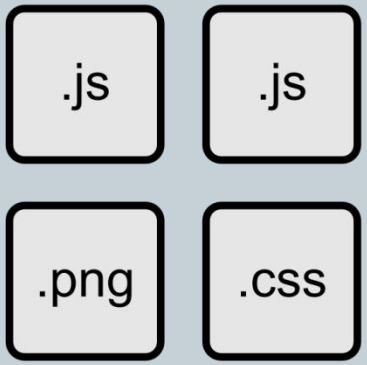
Core Purpose of webpack



modules
with dependencies



webpack
MODULE BUNDLER



static
assets

Core Concepts

- Entry
  - Starting point of the graph of app dependencies
  - Tells webpack where to start
  - The first file to kick off your app
- Output
  - Tells webpack how to treat bundled code
  - *output.filename*: specifies the name of our bundle file
  - *output.path*: specifies the location for placing the bundle

Core Concepts

- Loaders
  - Transform files into modules
  - Used to perform some pre-processing on files before they are added to the bundle
  - *test*: Identify which file(s) should be transformed
  - *use*: Specify the loader that transforms identified files
- Plugins
  - Applied on the bundle before it is output
  - Used to add functionality typically related to bundles
  - Most commonly used to perform actions and custom functionality on "compilations" or "chunks" of your bundled modules

webpack in Action

- Make a new app
- Create two JS modules
- Install and configure webpack
- Run webpack

- Make a new app
  - Create a directory – demo-basic
  - Change the directory to demo-basic
  - Run npm init command, provide necessary details

webpack in Action

- Create two JS modules
  - Launch code editor
  - Create a new folder below the root folder, for e.g., name is as ‘src’
  - Create two files inside ‘src’ folder
    - product.js – utility functions related to product details
    - index.js – calls functions from sum.js, then prints the result
- JS Module Systems
  - CommonJS – require – module.exports
  - ES2015 – import – export

webpack in Action

- Install and configure webpack
  - Run command: npm install webpack –save-dev
  - Create webpack.config.js
    - Define config object
    - Provide “entry” and “output” properties on config object
- Run webpack
  - Within package.json, include “build”: “webpack” within “scripts” property
  - Run command: npm run build
- Running app in the browser
  - Create index.html in the app root folder
  - Include a reference to build/bundle.js in index.html
  - Launch index.html in a browser

webpack in Action

- Demos
  - Basic Demo
  - Loader Demo
  - Plugin Demo
  - Code Splitting Demo

Q & A

- Thank you!