Production Build

Cory House

@housecor bitnative.com

Here's the Plan

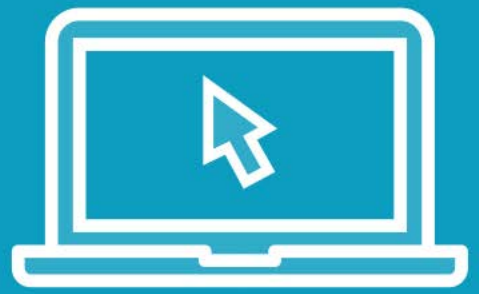
- Minification
- Sourcemaps
- Dynamic HTML
- Cache busting
- Bundle splitting
- Error logging

Minification

How Does Minification Work?

- Shortens variable and function names
- Removes comments
- Removes whitespace and new lines
- Dead code elimination / Tree-shaking
- Debug via sourcemap

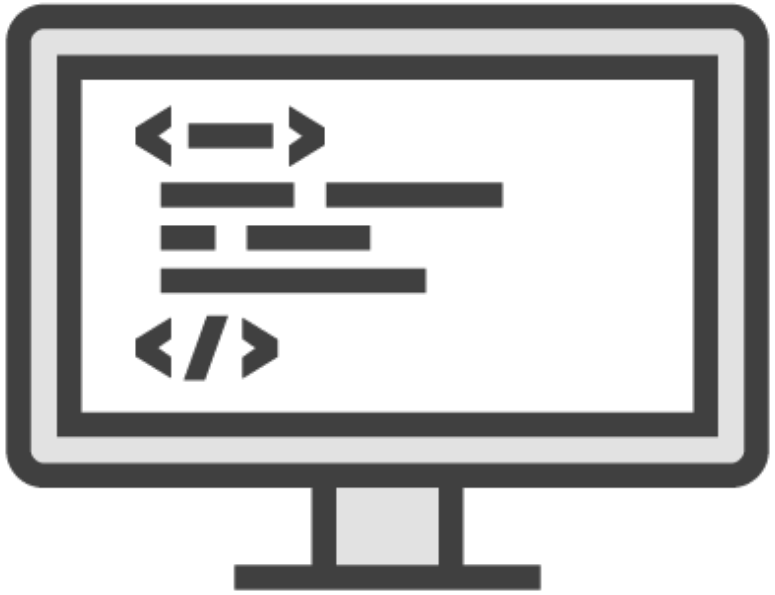
Demo



Set up Minification

Dynamic HTML

Why Manipulate HTML for Production?



- Reference bundles automatically
- Handle dynamic bundle names
- Inject production only resources
- Minify

```
<html>
  <head>
    <title>Your mom</title>
  </head>
  <body>
    <script src="bundle.js"></script>
  </body>
</html>
```

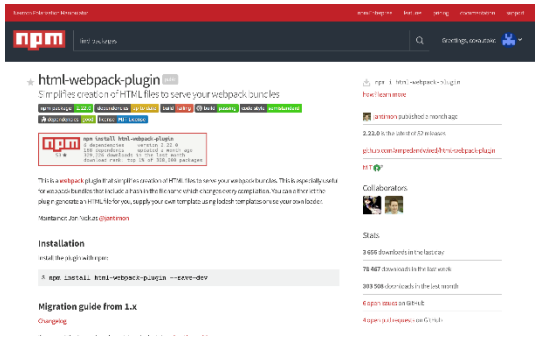
HTML

Referencing Bundled Assets in HTML

```
<html>
<head></head>
<body>
  <script src="bundle.js"></script>
</body>
</html>
```

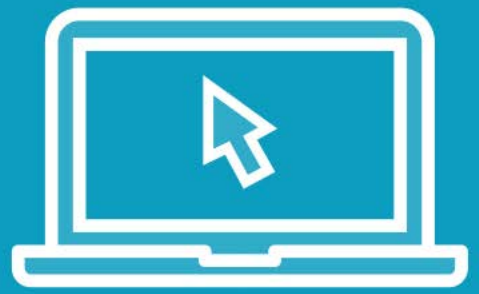
Hard code

Manipulate via Node



html-webpack-plugin

Demo



Handling Dynamic HTML

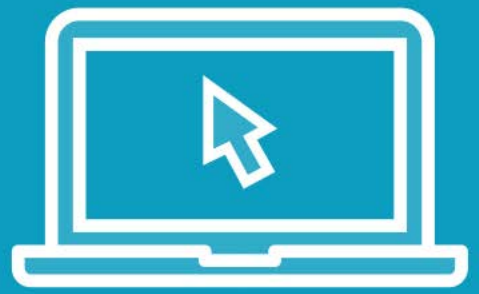
Bundle Splitting

Why Bundle Splitting?

Speed initial page load

Avoid re -downloading all libraries

Demo



Bundle splitting with Webpack

Cache Busting

Why Bust Cache?

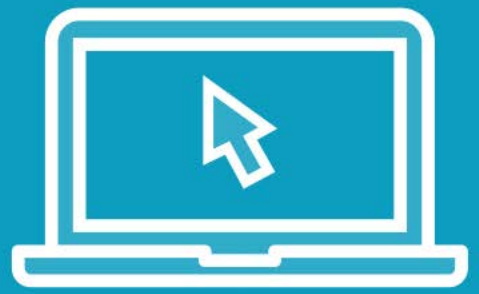
Save HTTP Requests

Force request for latest version

Here's the Plan for Busting Cache

1. Hash bundle filename
2. Generate HTML dynamically

Demo



Set up cache busting

Production Error Logging

Error Logging

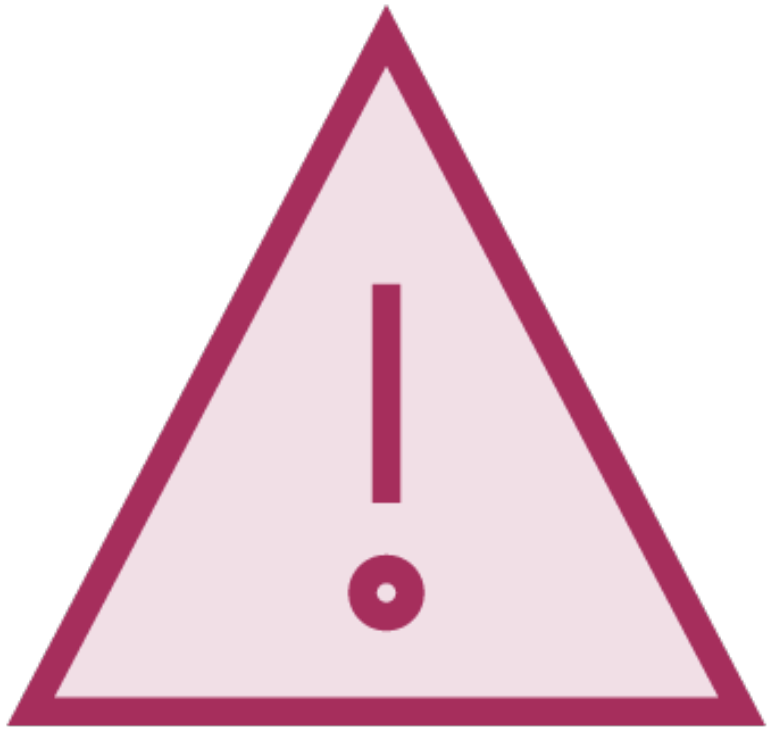
TrackJS

Sentry

New Relic

Raygun

JS Error Logging: What to Look For



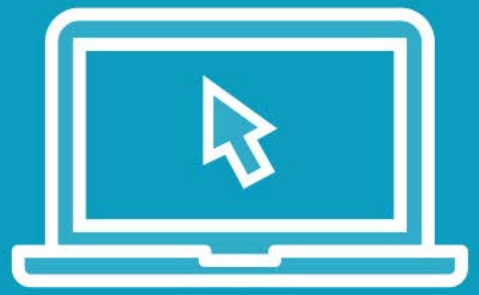
Error Metadata
- Browser
- Stack trace
- Previous actions
- Custom API for enhanced tracking

Notifications & integrations

Analytics and filtering

Pricing

Demo



Set up error tracking via Track.js

Wrap Up

Minification

Sourcemaps

Minified HTML and dynamic script tags

Cache busting

Bundle splitting

Error logging

Dynamic HTML via EmbeddedJS

Final module: Production deploy and updates