ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Initial Load Performance
Assets & Build

Alex Thalhammer

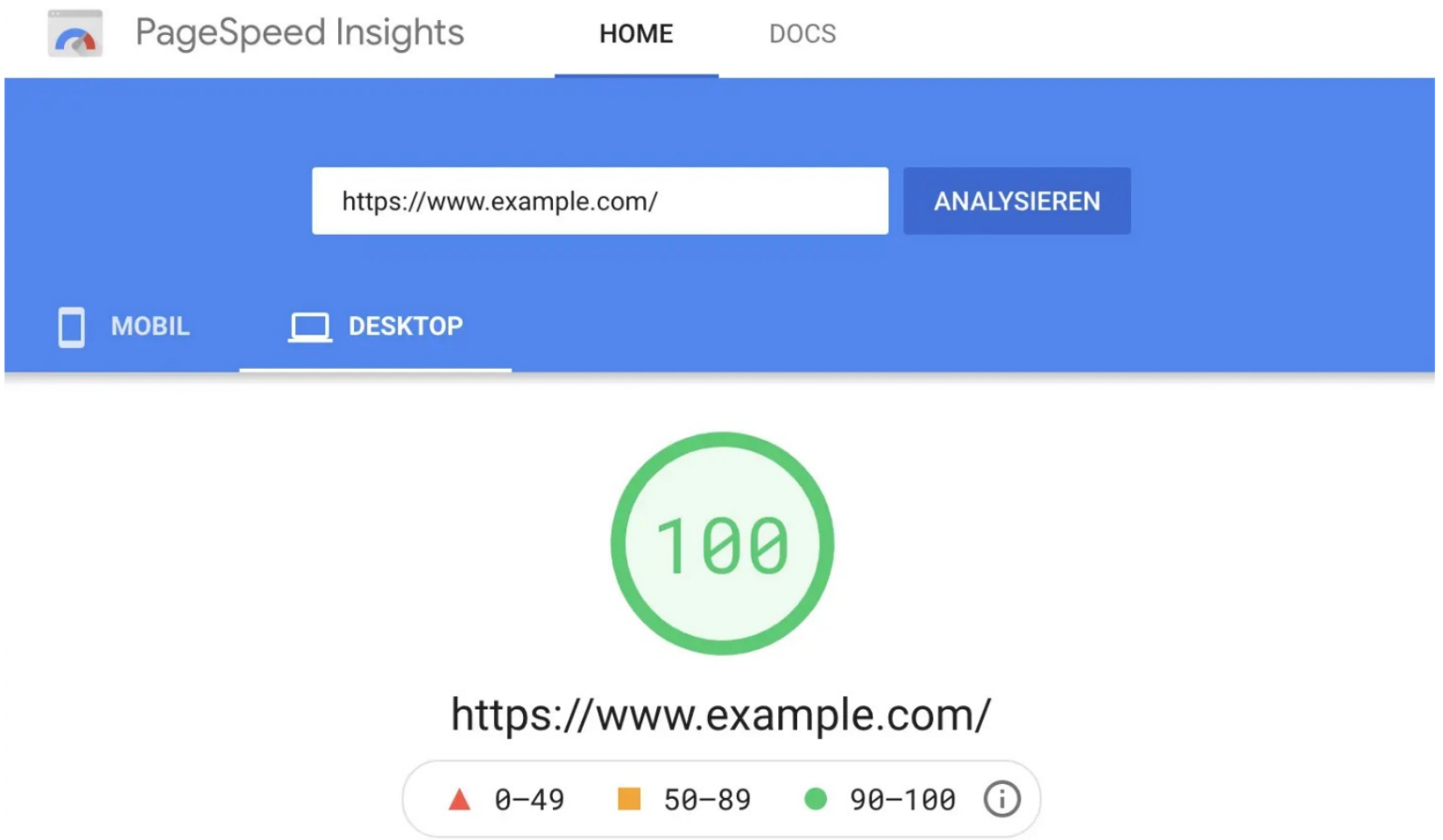
Outline 02 - Initial Load Performance

- Assets & Build
- Lazy Loading & Deferrable Views
- SSR & SSG

Outline 02a - Assets & Build

- Use web performance best practices
- Use NgOptimizedImage (since NG 14.2.0)
- Use build optimization
- Avoid large 3rd party libs / CSS frameworks

Web Performance Best Practices



Use web performance best practices - I

Problems:

- *Images not optimized*
- *Images not properly sized*
- *Slow server infrastructure*
- *Unused JS code or CSS styles*
- *Too large assets, too many assets*
- *Caching not configured correctly*
- *Compression not configured correctly*
- *…*

Use web performance best practices - II

Identify:

- Lighthouse & PageSpeed Insights
- WebPageTest.org or
- Chrome DevTools

Use web performance best practices - III

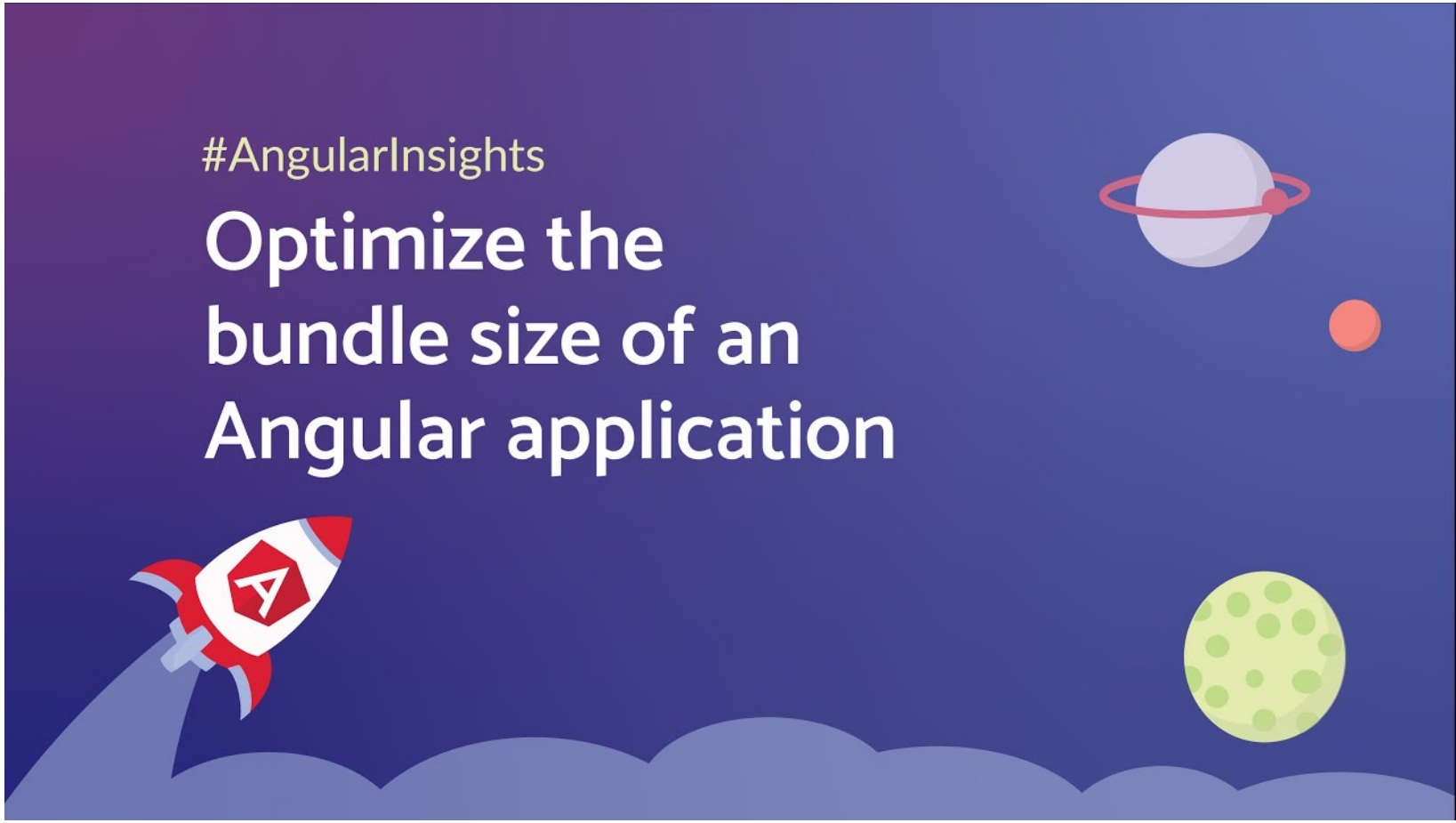
Solutions:

- *Images not optimized → Use .webp, .avif or .svg*
- *Images not properly sized → Use srcsets*
- *Too large assets, too many assets → Clean up & lazy load assets*
- *Unused JS code or CSS styles → Clean up & lazy load assets*
- *Slow server infrastructure → HTTP/2, CDN*
- *Caching not configured correctly → Configure it*
- *Compression not configured correctly → Brotli or Gzip*
- *…*

Use NgOptimizedImage (since NG 14.2.0)

- Problem: *Lighthouse or PageSpeed report image errors / warnings*
- Identify: Lighthouse & PageSpeed Insights / WebPageTest or DevTools
- Solution: Use NgOptimizedImage's ngSrc instead of src attribute
  - Takes care of intelligent lazy loading
  - Prioritization of critical images ("above-the-fold")
  - Also creates srcset & sizes attributes (for responsive sizes, since NG 15.0.0)
  - Also supports high-resolution devices ("Retina images")

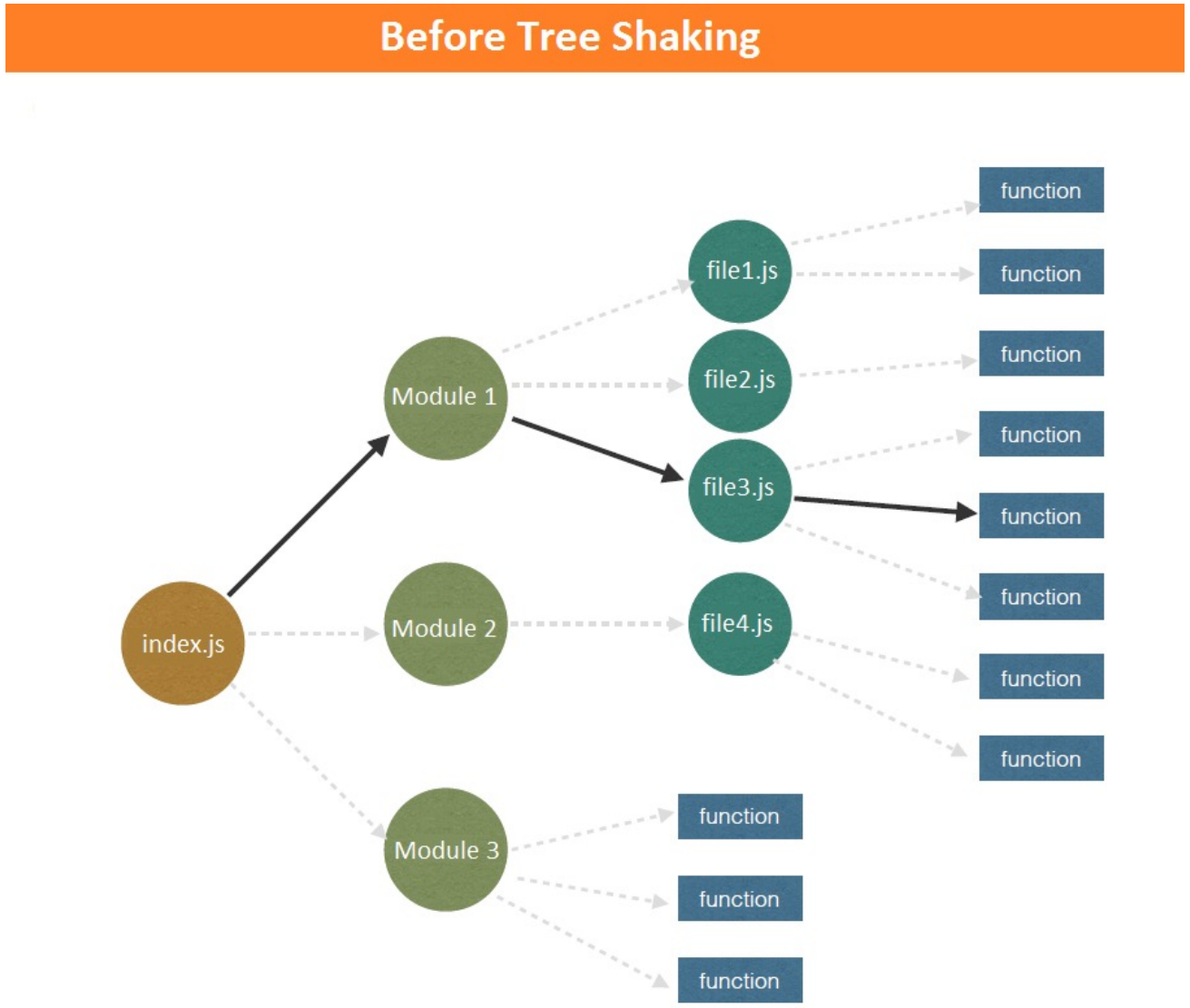
Build optimization



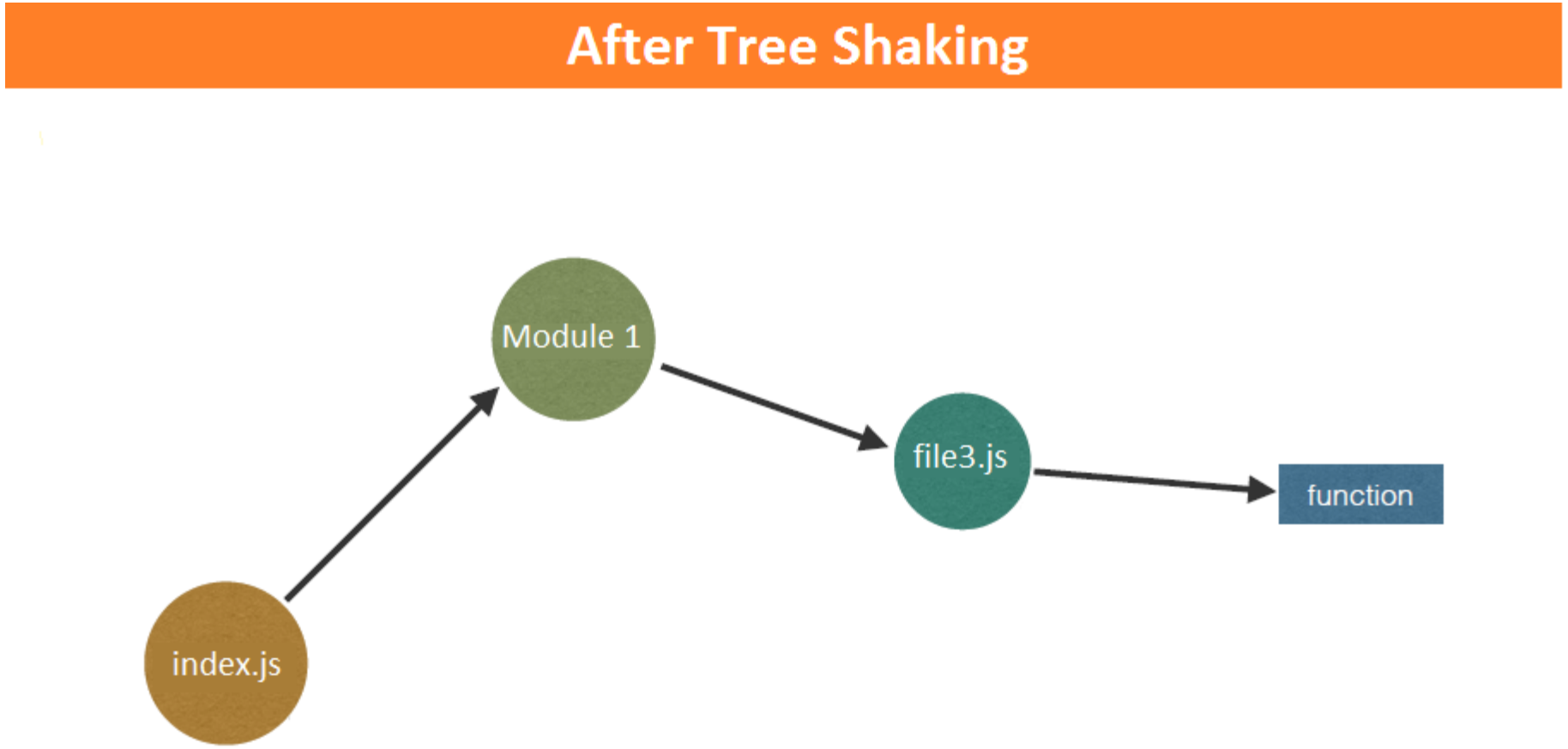
Advantages of Angular Ivy (since V9)

- Angular ViewEngine itself was not tree-shakable
- Default since NG 10, for libs default since NG 12
- AOT per default → You don't need to include the compiler!
- Ivy also does a lot of under the hood optimization
- Tools can easier analyse the code
  - Remove unneeded parts of frameworks
  - Called Tree Shaking
    - Also 3rd party and
    - Even our own libs

Tree Shaking



Tree Shaking



Use Build Optimization – I

Problem:

- *Too large build*
- *Downloading the Angular App takes too much time / ressources*

Identify:

- CSS / JS Files not minimized
- Unused JS code included in the build

Use Build Optimization – II

Solution:

- Use production build
  - ng b(uild) (--c production)

- Set up angular.json correctly

```
"production": {
  "buildOptimizer": true,
  "optimization": true,
  "vendorChunk": true
}
```

DEMO – Build Configuration

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Avoid large 3rd party libs / CSS frameworks

- Problem: *Importing large 3rd party libraries that are not treeshakable*
  - *moment*
  - *lodash*
  - *charts*
  - …
- Identify: Source Map Analyzer or Webpack Bundle Analyzer
- Solution: Remove or replace that lib / framework
  - *moment → luxon, day.js or date-fns*
  - *lodash → lodash-es*
  - …

DEMO – Large Libs

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE
ARCHITECT

Lab

Initial Load Performance

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

SOFTWARE ARCHITECT

Recap

- Use web performance best practices
- **Use NgOptimizedImage (since NG 14.2.0)**
- **Use build optimization**
- Avoid large 3rd party libs / CSS frameworks

References

- Optimize the bundle size of an Angular application
  - https://www.youtube.com/watch?v=19T3O7XWJkA

- Angular Docs
  - NgOptimizedImage
  - NG Build