Web Advanced II: Toolthema

JS: Validierung, Kompression, Obfuscation

Felix Trojan

Bauhaus-University Weimar, Germany
June 10, 2013

Warum Code validieren?

JS: Validierung

Anwenden bestimmter Richtlinien (Style Guides) in Form von Regeln auf den Code, nach denen dieser umgeformt wird.

- Code wird lesbarer und einheitlicher
- automatische Code-Formatierung spart Zeit
- vermeidet häufige Programmierfehler

Tools

- JSLint
- Google Closure Linter

Tools

- JSLint
- Google Closure Linter

Googles Styleguide

- Lanuage Rules
  - Semikolons
  - Schleifen richtig verwenden
  - var
  - ...
- Style Rules
  - Naming Convention: unctionNamesLikeThis, variableNamesLikeThis, ...
  - explizite Scopes
  - Angabe von Annotations (JSDoc)
  - ...

Warum Code komprimieren?

JS: Komprimierung

Transformiert den Code in eine möglichst kompakte Form durch verschiedene Optimierungen.

- \+ Zusammenfassung von var/const Definitionen
- \+ toten Code entfernen
- \+ Konstanten evaluieren
- \+ Leerzeichen, Kommentare, Tabs entfernen
- \+ Conditionals/Vergleiche optimieren
- \+ kompakterer, schnellerer Code
- \+ Ressourcen sparen (Speicher)
- \+ bieten oft Obfuscation

Warum Code komprimieren?

JS: Komprimierung

Transformiert den Code in eine möglichst kompakte Form durch verschiedene Optimierungen.

+ Zusammenfassung von var/const Definitionen
+ toten Code entfernen
+ Konstanten evaluieren
+ Leerzeichen, Kommentare, Tabs entfernen
+ Conditionals/Vergleiche optimieren
+ kompakterer, schnellerer Code
+ Ressourcen sparen (Speicher)
+ bieten oft Obfuscation

- Code wird schwer lesbar, hart zu debuggen
- zu aggressive Optimierung kann das Verhalten des Codes verändern

Tools

- Google Closure Compiler
- YUI Compressor
- jsmin
- uglifyjs

Tools

- Google Closure Compiler
- YUI Compressor
- jsmin
- uglifyjs

Spezialfall: Binäre Kompression

- gzip

Für viele Tools auch als IDE-Plugins: Google Closure Plugin

Code-Verschleierung = Obfuscation

Encryption ≠ Obfuscation!

JS: Obfuscation

Code wird so transformiert, dass das Nachvollziehen der Arbeitsweise erschwert wird. Soll vor allem Reverse Engineering und den damit verbundenen Diebstahl des Quellcodes verhindern. Die Funktionalität des umgeformten Codes ist dabei die gleiche wie die des Ursprungscodes.

- Oft: Kompression als Grundlage
- Umformung von Anweisungen
- Sinnlose Sprünge nach Vergleichen die immer True/False ergeben
- Aufruf vieler Subroutinen
- toten Code einfügen
- Umbenennung von Funktionen, Variablen, Konstanten
- Arrays umformen

Code-Verschleierung = Obfuscation

Encryption ≠ Obfuscation!

JS: Obfuscation

Code wird so transformiert, dass das Nachvollziehen der Arbeitsweise erschwert wird. Soll vor allem Reverse Engineering und den damit verbundenen Diebstahl des Quellcodes verhindern. Die Funktionalität des umgeformten Codes ist dabei die gleiche wie die des Ursprungscodes.

- Oft: Kompression als Grundlage
- Umformung von Anweisungen
- Sinnlose Sprünge nach Vergleichen die immer True/False ergeben
- Aufruf vieler Subroutinen
- toten Code einfügen
- Umbenennung von Funktionen, Variablen, Konstanten
- Arrays umformen

  - es gibt genauso viele Deobfuscation-Tools
  - macht es nur aufwendiger den Code zu verstehen, aber nicht unmöglich!
  - kann Code verlangsamen
  - kann Code-Verhalten ändern

- Compression/Obfuscation Pitfalls

Tools

- Closure-Compiler
- YUI Compressor
- uglifyjs