入門スライド ／ 学生・初学者向け

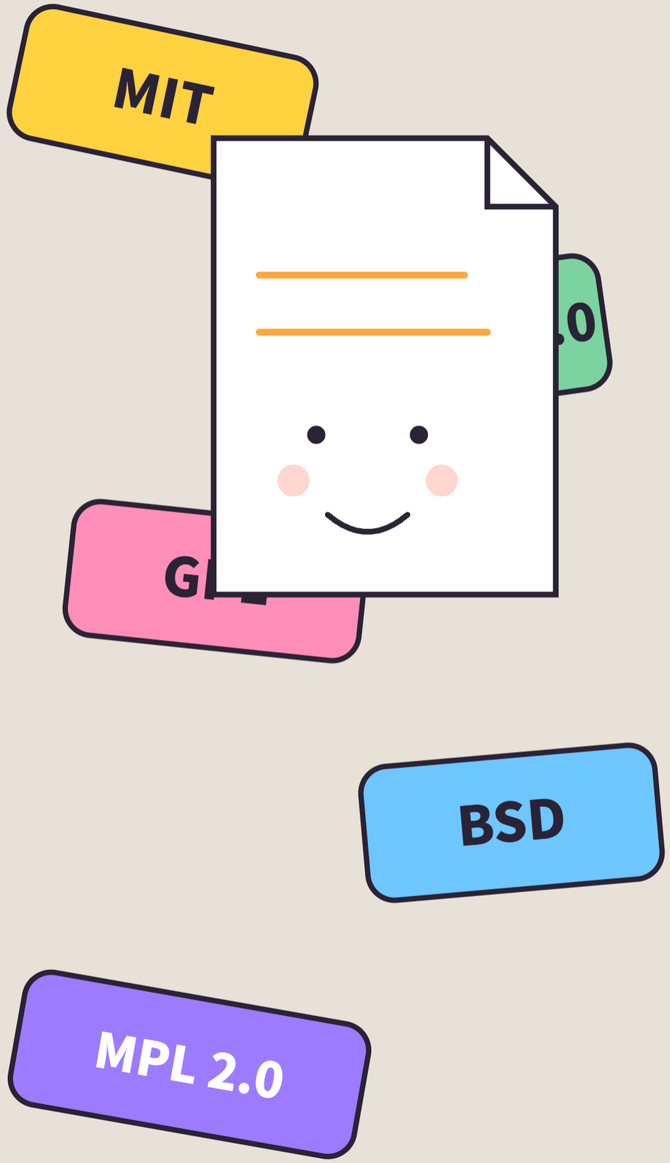
むずかしくない オープンソース ライセンス。

～ きほんの「き」と、いまの動向 ～

オープンソースライセンス研究所 顧問
吉田 行男

このスライドは法律相談ではありません。実務上は専門家にも相談してね。

🚀 全30枚



WHY — なぜ、いま学ぶのか

OSSライセンスは、「知らなかった」では済まない

🌎 もう「OSSなしの開発」はあり得ない

いまやアプリやサービスのコードの **7〜9割** は、世界中の誰かが書いたOSSの組み合わせでできている。

スマホもクラウドも家電も車も、その土台はOSS。 私たちは毎日、 無数のライセンス の上で暮らしている。

🧩 **OSSは「部品」ではなく、もはや「土台」**

⚠️ ライセンス違反のリスク

- **著作権侵害** として法的責任を問われる
- 製品の**出荷停止・回収** に発展した事例も
- 非公開のはずのソースの **公開を迫られる**
- 企業の**信用**を一瞬で失う

🤔 「うっかり」で起きてしまう

コピペ1行、ライブラリ1個から違反は始まる。 悪意がなくても **知らなければ防げない** 。

🌱 理解すれば「武器」になる

① **安心してOSSを使い倒せる**

② 自分の成果物を**正しく世界に公開** できる

③ OSS文化に**貢献する側**に回れる

ルールを知ることは、 **「自由を正しく使うための地図」** を 手に入れること。

SPEAKER — Self Introduction

講演者：吉田 行男 ／ YOSHIDA Yukio

吉田 行男

OSSコンサルタント／独立コントリビューター

🏆 受賞・表彰

2018/11 北東アジアOSS推進フォーラム OSS貢献賞 受賞 （第13回 日中韓 オープンソースアワード）

📖 経歴

入社当時
金融端末のソフトウェア開発に従事

2000年頃～
Linux／OSSのビジネス開発を担当

2012年～
オープンソース専門組織に所属

2019年～
定年退職を機に**独立** 。OSSコンサルタントとして活動

💼 現在の業務

- OSSを活用したビジネス構築の支援
- 新しい技術／OSSの発掘・評価検証
- ビジネス・ソリューションの立ち上げ支援、人材育成
- **OSSコンプライアンス管理** ・ガイドライン作成・社内プロセス構築
- 各種講演、執筆、技術ドキュメント翻訳

📰 最近の活動

- **記事執筆** （Think IT）「OSSの持続可能性を問う ープリザンター10周年記念イベントで語られたコミュニティと継続の流儀」など
- **翻訳** 「オープンソースとAIの未来」（LF Japan）
- **YouTube** 「OSS界隈のニュース深読み」配信

CONTENTS

きょうのもくじ

CHAPTER 1

01

そもそもOSSって、なに？

「無料で使えるソフト」だけじゃない、もう一歩深い話。

OSSは「自由が保証された」ソフトウェア

オープンソースソフトウェア（OSS）は、ソースコードが公開されているだけではなく、**OSI（Open Source Initiative）**が定めた**「オープンソースの定義」（OSD）**を満たすライセンスで配布されているもののことを指すよ。

⚠️ よくある誤解

❌「無料で見れる＝OSS」
❌「GitHubに上がってる＝OSS」
✅ **ライセンス**を読んで初めてわかる！

OSDが求めるもの（抜粋・全10項目）

① 自由な再配布が認められる

② ソースコードが入手できる

③ 派生物の作成・配布が認められる

④ 作者のソースコードの完全性

⑤ 個人・グループへの差別をしない

⑥ 利用分野への差別をしない

⑦〜⑩ ライセンス配布／製品縛り禁止／ 他ソフトウェア縛り禁止／技術中立

「フリー」には、2つの意味がある

Free beer

＝無料のビール

↓

お金を払わずに

使える

「無償」

Free speech

＝言論の自由

↓

改変・再配布など

「行動の自由」が保証

「自由」

OSSは こっちの「自由」

OSSが大切にしているのは **"自由"** のほう。

実際には無償なものが多いが、**「お金を取ってもOK」**。有償でOSSを売ることもライセンス的には完全に合法！

なぜ「ライセンス」が必要なの？

📜 デフォルトは「全部禁止」

コードを書いた瞬間、自動的に**著作権**が発生する。 作者の許可なしには、コピーも改変も再配布もできない。

🎁 ライセンス＝「使っていいよ」の手紙

作者が「こういう条件なら使っていいよ」と みんなに **あらかじめ許可** を出すための文書。 これがあるから、知らない人のコードも使える！

📦 GitHubに置いても自動でOSSにはならない

ライセンスファイルがないリポジトリは、 **「全部禁止」状態のまま** 。これが意外と落とし穴！

ライセンスが書かれる場所

```
my-project/
  ├── LICENSE   ←  これ！
  ├── README.md
  ├── package.json
  └── src/
        └── index.js
```

ファイル名は **LICENSE** または **LICENSE.txt** が定番。 GitHubも自動で認識して 📄 MIT みたいなバッジを表示してくれるよ。

CHAPTER 2

02

ライセンスの 2大派閥

「自由」をどこまで強制するか。これが分かれ目。

大きく分けると、2つの考え方

パーミッシブ系

Permissive ／ 寛容型

「ご自由にどうぞ。 名前だけ載せてね」

改変したコードを**非公開のまま** 商用製品に組み込んでもOK。 ビジネス側にやさしく、エコシステムが広がりやすい。

MIT　BSD　Apache 2.0　ISC

VS

コピーレフト系

Copyleft ／ 強い継承型

「自由を受け取ったら、 次の人にも自由を渡してね」

派生物も**同じライセンス** で公開する義務がある。 「自由」がウイルスのように伝播していく仕組み。

GPL　AGPL　LGPL　MPL

「自由の伝染力」にはグラデーションがある

CHAPTER 3

03

代表的な ライセンス図鑑

6つの「定番ライセンス」のキャラを紹介していくよ。

いちばん人気の MIT & BSD

MIT

パーミッシブ

Massachusetts Institute of Technology Licence

「3行で書ける」シンプル王者

本文がたった数百語。条件は実質 **「著作権表示とライセンス文を残す」** だけ。

- ✅ 商用利用
- ✅ 改変
- ✅ 再配布
- ✅ クローズド化
- 📝 著作権表示の保持
- 🚫 無保証

例：React・Vue・jQuery・Rails・Node.js…ほぼ全部

BSD

パーミッシブ

Berkeley Software Distribution（2-clause / 3-clause）

MITに似た古参の良ライセンス

MITとほぼ同じ寛容さ。3条項版には **「名前を宣伝に使うな」** 条項が追加されている。

- ✅ 商用利用
- ✅ 改変
- ✅ 再配布
- ✅ クローズド化
- 📝 著作権表示の保持
- 🚫 名前の無断利用

例：FreeBSD・Nginx・Go言語・PostgreSQL・Valkey

企業が好む Apache 2.0

Apache 2.0

パーミッシブ

MITに「特許」の安心感をプラス

パーミッシブだけど、より厳密でビジネス向き。 最大の特徴は**特許条項**。

- 📜 **特許ライセンス**が明示的に付与される
- ⚔️ **特許訴訟**を起こした人は、ライセンスが取消される（特許防衛）
- 📝 変更したファイルには**変更通知**を残す
- 🏷️ NOTICEファイルがあれば再配布物にも引き継ぐ

例：Kubernetes・Android・Apache HTTP Server・TensorFlow・Swift

💡 なぜ企業はApacheが好き？

MITは特許について**沈黙** している。「使わせるよ」と書いていても、 あとで特許侵害で訴えられるリスクがゼロじゃない。

Apache 2.0なら**特許の扱いまで明文化** されているから、 法務的にゴーサインが出しやすい。

🛡 **「迷ったら Apache 2.0」と言われる理由**

最強コピーレフトの GPL ／ AGPL

GPL v2 / v3

強コピーレフト

GNU General Public License

「自由を継承させる」憲章

GPLコードを取り込んで作ったソフトを**配布**するなら、 全体を **同じGPL** でソース公開しなければならない。

- 🧬 派生物も自動的にGPL化
- 📤 配布時にソース提供
- 🆚 v3は特許条項・Tivoization対策を強化

例：Linux Kernel(v2) ／ Bash(v3) ／ GIMP

AGPL v3

最強

Affero General Public License

SaaSの「抜け穴」を塞ぐGPL

GPLは「配布」が引き金。 →サーバで動かすだけなら配布じゃない＝公開不要…という **SaaS抜け穴** があった。

AGPLは**「ネットワーク越しに使わせる＝配布扱い」** として、 SaaS提供時もソースコードを利用者に提供する義務を課す。

例：MongoDB(旧)・Mastodon・Nextcloud・Grafana(旧)・Redis 8.0+

ちょうどいい LGPL ／ MPL 2.0

LGPL

弱コピーレフト

GNU Lesser General Public License

「ライブラリだけGPL」モード

ライブラリ自体はLGPLで公開しないといけないけど、 **動的リンク** して使うだけなら、自分のアプリは **非公開でもOK**。

商用ソフトから安心して呼び出せるバランス型。

例：glibc・FFmpeg(一部)・Qt(LGPL版)・GTK

MPL 2.0

弱コピーレフト

Mozilla Public License v2.0

「ファイル単位」で共有する

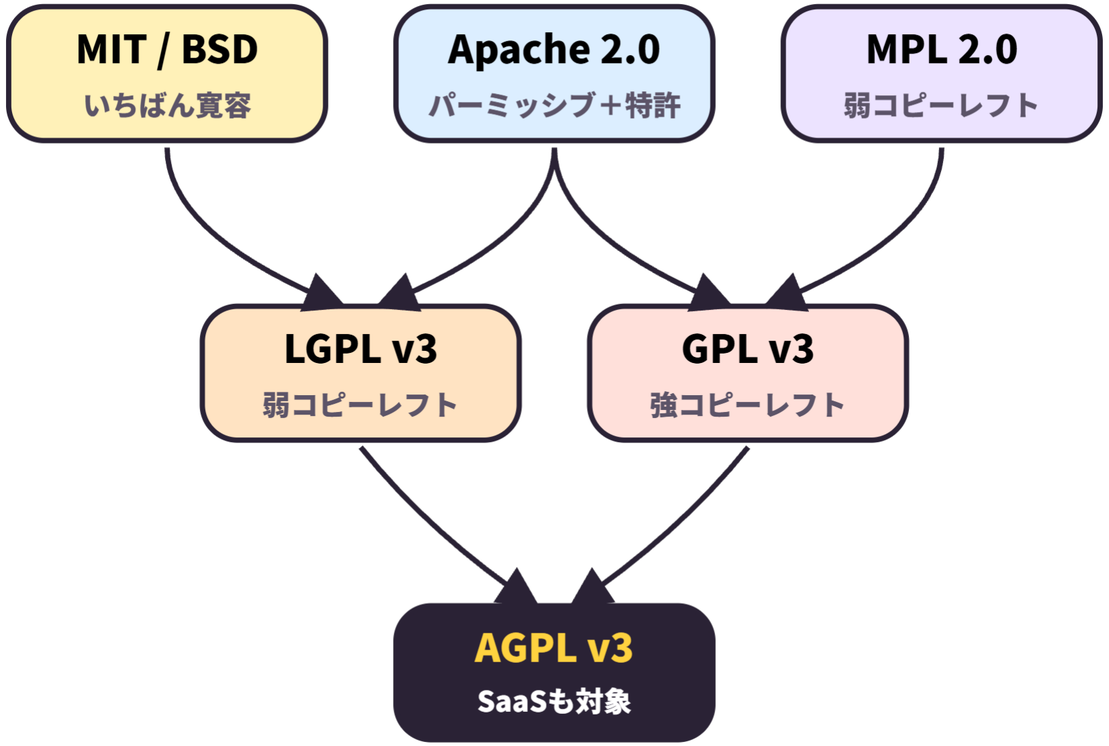
変更を加えた**そのファイルだけ** をMPLで公開すればOK。 他のファイルは独自の（クローズドな）ライセンスでも構わない。

プロプライエタリ製品との**共存** がしやすく、特許条項もちゃんとあり。

例：Firefox・Thunderbird・Rust(一部)・OpenTofu

混ぜていい？ ライセンス互換マップ

🌊 互換性は「下流に流れる」イメージ



↓矢印は「混ぜたら全体がこのライセンスになる」方向。下に行くほど制約が強くなる。

🚫 やっちゃダメな組み合わせ

たとえば **GPL v2 only** と **Apache 2.0** は **非互換** 。一緒のバイナリにできない！

✅ 大事な原則

複数のOSSを混ぜたソフトは、 **いちばん厳しいライセンス** がプロジェクト全体に適用される。

🔍 迷ったら

SPDXのID・ **choosealicense.com**・ **tldrlegal.com** などのサイトでチェック！

CHAPTER 4

04

使うとき／公開するとき

実務で「あっやばい」となりがちな7つのポイント。

OSSを「使う」ときの3つのお作法

① ライセンスを 読む（or 種類を確認）

npm / pip / Mavenにポンと足す前に **ライセンス名** だけでも確認。 GPL系・AGPL系がproductive内に紛れていないか要注意。

```
$ npm-license-checker
$ pip-licenses
```

② 著作権表示と ライセンス文を残す

パーミッシブでも**ほぼ全て** に 「著作権表示／ライセンス文の保持」義務がある。

アプリの**「ライセンス情報」画面**や**NOTICE/THIRD_PARTY ファイル** に列挙するのが定番。

③ バージョンと出典を 記録（SBOM）

どのOSSを、どのバージョンで、どこから入れたか。 Software Bill of Materials を **SPDX/CycloneDX** 形式で残しておくと、 脆弱性対応もライセンス監査もぐっと楽になる。

SBOM＝部品表

自分のコードを「公開する」ときの4ステップ

STEP 1

何をしたいか はっきりさせる

「広く使われたい」「派生も自由にしたい」のか、「商用利用は守りたい」のか。 ゴールでライセンスは決まる。

→

STEP 2

既存の依存と「ぶつからない」か確認

GPLライブラリを使っているのに「MITで公開！」はNG。 あなたのライセンスは依存に縛られる。

→

STEP 3

LICENSEファイルを 正しく置く

本文を「自分で書き直さない」。 OSI公式の文面を**そのまま** コピペする。 年と著作者名だけ書き換える。

→

STEP 4

CLA / DCOで 外部貢献を整える

他人がPRをくれたとき、その著作権の扱いを定める仕組み。 無いと将来ライセンス変更ができなくなることも！

CLA : Contributor License Agreement
DCO : Developer Certificate of Origin

CHAPTER 5

05

最近の動向 2023 → 2026

OSSが、いま大きく揺れている。

TREND #1 — Re-licensing wave

「ソースアベイラブル」化の連鎖

2018年あたりから「**クラウド大手にタダ乗りされたくない**」という理由で、OSSプロジェクトが **OSI非承認のライセンス** へ移る流れが続いている。

2018
MongoDB：AGPL → **SSPL**へ

2021
Elasticsearch / Kibana：Apache → SSPL／Elastic License v2 → AWSが **OpenSearch** をフォーク

2023.8
HashiCorp(Terraform/Vault等)：MPL → **BUSL 1.1** → **OpenTofu**誕生

2024.3
Redis：BSD → **RSALv2** ／ **SSPL** → **Valkey**誕生

2025
Redis 8.0でAGPL v3を選択肢に追加して**方針転換**...

👄 共通する動機

「自分達のOSSをマネタイズしているクラウドベンダー（特にAWS）が 売上を吸い上げている」という危機感。

⚖️ BUSL / SSPL とは？

ソースは見られて改変もできるけど、「**競合SaaSとして提供してはダメ**」など制限つき。 OSI的には**OSSではない**。

🍴 結果：コミュニティはフォークで応戦

OpenTofu／Valkey／OpenSearchはどれも **Linux Foundation** 傘下で開発が続いている。

CASE STUDY — Redis vs Valkey

2024年最大の事件：Redis → Valkey フォーク

Redis

Redis Ltd（旧Labs）が運営

- 2024.3.20 BSDから **RSALv2 / SSPL** へ
- クラウド事業者は**商用契約が必要**に
- 多くの外部コントリビューターが離脱
- 2025 Redis 8.0 で**AGPL v3**を追加し方針修正

FORK

Valkey

Linux Foundation傘下のフォーク

- Redis 7.2.4（最後のBSD版）からフォーク
- AWS / Google / Oracle / Ericsson / Snap 等が支援
- ライセンス変更からわずか**2週間**でLF採択
- Valkey 8系で **I/O multithreading** など独自進化

📌 教訓

コミュニティはライセンス変更に対して**「数週間でフォーク」**できる時代になった。 OSSの「BSDライセンスだから安心」は、 **歴史を遡って初版だけが該当** することがある点に注意。

TREND #2 — Supply chain attack

2024年3月「xz utils バックドア事件」

🕵️ なにが起きた？

Linuxの**圧縮ライブラリ** xz/liblzma に、 悪意のあるメンテナーが **2年以上かけて信頼を獲得** し、 最終的に**SSH経由でリモートコード実行できる** バックドアを忍び込ませた。

Microsoftの研究者 Andres Freund が SSHログインの**CPU使用率の異常** に気づき発覚。 CVSSスコアは満点の **10.0**。

💥 史上最大級のサプライチェーン攻撃未遂と評される

🧠 OSS的な学び

- **「みんなで見ているから安全」** は神話。 重要なライブラリでもメンテナー1人状態は珍しくない。
- 「燃え尽きた」メンテナーに**メンテナー権限を譲り受ける** ソーシャルエンジニアリング。
- 悪意あるコードはGitリポジトリには無く、 **tarballのbuildスクリプトに** 。
- OSSメンテナンスへの**持続可能な資金支援** と コードレビュー・ **SBOM**運用の重要性が再認識された。

TREND #3 — AI & Licensing

AI時代の「学習データ」はライセンス的にOK？

🤖 LLMの学習材料

ChatGPT・Claude・Copilotなどの大規模モデルは、**大量のOSSコード** で訓練されている。

GPLコードで学習したモデルが吐き出すコードは、GPLの「派生物」？──法的に未解決な問題。

📚 GitHub Copilot 訴訟（2022〜）など継続中

⚠️ 開発者目線の注意点

- AI生成コードの**著作権の所在**はまだ曖昧
- そっくりなコードが出力される **リグルジテーション** のリスク
- 多くの企業がAI支援開発に**ガイドライン**を整備中
- **OSSライセンスの遵守責任** はAIに任せきりにできない

🆕 「Open Source AI」も再定義中

OSIは2024年に **「Open Source AI Definition (OSAID) v1.0」** を公開。モデル本体だけでなく、**学習データ・学習コード・ウェイト** すべての公開要件を定義。

Llama や Gemma が「Open Source」を名乗れるかが議論の焦点。

TREND #4 — Regulation

国も動く：SBOM義務化 と EU CRA

📦 SBOM（部品表）の義務化

ソフトウェアを構成する **すべてのOSSとそのバージョン** を一覧にしたファイル。SPDX / CycloneDX が標準フォーマット。

- 米国：大統領令でフェデラル調達の前提に
- 日本：経産省のセキュリティガイドラインで推奨
- SaaSでも顧客に提示を求められる時代へ

EU EU Cyber Resilience Act

2024年成立。EU市場で **「デジタル要素を持つ製品」** を売る企業に セキュリティ要件・脆弱性対応・SBOM提出を課す。

- 純粋なOSSは原則対象外（個人＆非商用）
- ただし**OSS Steward** (財団など)には軽量義務
- 製品にOSSを**組み込んで売る側** は責任発生

🌏 日本でも

- 経産省「ソフトウェア管理に向けたSBOM導入手引」
- 金融庁・防衛省でもガバナンス整備が進む
- Linux財団Japan・OpenChain JWG が標準化推進

OSSは **「タダで使える素材」から「ガバナンス対象」** へ。

TREND #5 — New licenses

「第三の道」を探す新ライセンスたち

🌅 Fair Source

Sentryが提唱した「ソース公開＋商用条件付き＋ **2年後にOSI承認OSS化** （DOSP）」する仕組み。

BUSL・FSLなどがこの考え方に近い。 **「ソースアベイラブル＋将来OSS」** のいいとこ取りを狙う。

🎨 FUTO License

Eron Wolfが立ち上げた「ユーザの権利を最優先」する思想のライセンス。 個人利用・改変は自由、商用配布は制限。

OSI的にはOSSではない けど、エンドユーザの自由を強く守る方針。

🌱 Polyform / PolyForm Shield

弁護士有志が作ったモジュラー型のライセンス集。 「Noncommercial」「Internal Use」「Small Business」など **用途を限定して配布** する選択肢を整理。

SSPLやBUSLよりも **条文がスッキリ** している。

💭 共通テーマは **「OSSの理念は守りたいが、ビジネスとして持続可能にしたい」** 。 OSSが社会基盤になった結果、**純粋なOSSの外側** に新しい契約形態が生まれている。

SUMMARY

きょうの5つの持ち帰り

01
OSS＝「自由が保証された」ソフト
無料 ≠ OSS。OSDが基準。

02
パーミッシブ vs コピーレフト
継承の強さに段階あり。

03
迷ったらMIT or Apache 2.0
特許まで気にするならApache。

04
表示・SBOM・互換性チェックを忘れずに
使う側にも責任がある。

05
OSSは変化の真っ只中
ライセンスは「凍結」しない。常にウォッチ。

📚 まずは LICENSE を1度ちゃんと読もう！
短いMITなら3分で終わるよ。

FURTHER READING

もっと深掘りしたい人へ

🌐 公式・基礎

- **opensource.org** OSI公式（OSDとライセンス一覧）
- **gnu.org/licenses** FSF公式（GPL/AGPL/LGPL）
- **spdx.org** ライセンスID標準
- **opensource.guide** GitHub公式の入門ガイド

🛠 実務でつかうツール

- **choosealicense.com** 迷ったときの選択フロー
- **tldrlegal.com** ライセンスを「1分で要約」
- **FOSSA / Snyk / Black Duck** 商用OSS管理
- **ScanCode / REUSE.software** OSSなツール

📖 日本語の良書・記事

- 姉崎章博『OSSライセンスの教科書』
- 可知豊『知る、読む、使う！オープンソースライセンス』
- OSSライセンスFAQ（IPA／OpenChain JWG）
- はてブ「OSSライセンス」タグの定番エントリ

ありがとう！

OSSライセンス、ちょっと身近に感じてもらえたかな？

📄 まずLICENSEを読む　📦 SBOMを作ってみる　🌳 動向を追い続ける

※ 本スライドは入門的な解説であり、法的アドバイスではありません。実務では必ず一次資料・専門家に当たってください。