“**Build with Us | Speedrun: Your First Ontology Function**”是 learn.palantir.com 上的一门进阶实战课程，由 Ontologize 团队提供。该课程旨在指导用户如何通过编写 **TypeScript** 函数（**Functions**）来扩展 Palantir Foundry 本体（Ontology）的功能，并将这些逻辑集成到 **Workshop** 应用程序中 1-3。
以下是根据来源对该课程内容的详细解释：

1. 核心目标与业务场景

- 业务背景：课程模拟了一个医疗财务分析场景，重点在于研究医院如何将其服务转化为实际收入 2。
- 核心指标：计算 **DSO**（**Day Sales Outstanding**，应收账款周转天数），这是一个衡量诊所收回收入速度的关键指标 4。
- 最终成果：构建一个“诊所财务追踪（**Clinic Finance Tracker**）”应用程序，该程序包含由函数驱动的指标卡、动态图表和操作按钮 2, 5。

2. 开发环境与资源准备

- 数据准备：通过 **Marketplace** 安装名为 “Speedrun: Your First Ontology Function” 的产品包，其中包含“诊所（Clinic）”和“财务（Financial）”两个核心对象类型 6, 7。
- 工具选择：主要在 **Code Repositories**（代码存储库 ） 中进行开发。用户需要创建一个类型为 “**Functions**” 的存储库，并选择 **TypeScript** 作为开发语言 3。
- 本体导入：在编写代码前，必须将本体中的对象类型导入到存储库的资源导入（Resource Imports）中，以便在代码中引用它们 7, 8。

3. 课程涵盖的四种函数类型

课程通过四个具体的函数示例，展示了函数在 Foundry 中的不同应用模式：

A. 基础指标计算函数 (Metric Calculation)

- 逻辑：编写 calculateDaysSalesOutstanding 函数。它使用 groupBy 方法按月聚合账单和收入数据，并通过“倒序计数”逻辑计算 DSO 值 4, 9, 10。
- **Workshop** 应用：在 Workshop 中创建一个函数驱动的数值变量，用于填充指标卡（Metric Card），实时显示所有诊所或特定诊所的 DSO（示例值为 40.5）11, 12。

B. 函数驱动的衍生列 (Function-backed Column)

- 逻辑：编写 functionColumnDaysSalesOutstanding 函数。该函数创建一个映射（Map），将每个诊所与其计算出的 DSO 值关联起来 13, 14。
- **Workshop** 应用：在 Workshop 的对象表格（Object Table）中添加一个衍生列。用户可以为该列设置数值格式（显示单位为“天”）和条件格式（例如：DSO > 40 显示为红色危险状态 ）15, 16。

C. 动态图表聚合函数 (Function-backed Chart)

- 逻辑：编写 calculateDaysSalesOutstandingMonthOverMonth 函数。它计算月度 DSO 的变化，并返回一个聚合结果 17, 18。
- **Workshop** 应用：在 Workshop 中配置 XY 图表，将其数据源从“对象集”更改为“函数”。该图表能根据用户在表格中选择的不同诊所实时更新趋势 19。

D. 函数驱动的操作逻辑 (Ontology Edit Function)

- 逻辑：使用 **@OntologyEditFunction** 装饰器编写函数。该函数不仅计算数据，还能修改本体对象。逻辑包括：切换诊所的“复核中（Under Review）”状态，并自动设定一个 DSO 改进目标（通常为当前值的 70%-95%） 20, 21。
- **Workshop** 应用：在 **Ontology Manager** 中将该函数包装成一个操作（**Action**），然后在 Workshop 中通过按钮触发，直接修改底层数据 5, 22。

4. 关键技术概念

- 装饰器（**Decorators**）：使用 @Function 暴露普通读取逻辑，使用 @OntologyEditFunction 暴露数据修改逻辑 4, 20。
- 异步处理：利用 TypeScript 的 Promise（如 Promise.all）并行处理多个诊所的复杂计算，以提高应用性能 4, 14。
- 实时预览与调试：在代码存储库中使用 **Live Preview** 功能，在不发布代码的情况下输入参数并查看函数运行结果，确保逻辑正确 11, 14, 21。

通过本课程，用户能够掌握如何超越标准的 UI 配置，利用代码的力量在 Foundry 中实现复杂的业务逻辑和高度定制化的用户交互 1, 23。