1. 函数式编程.

1) lodash 的 reverse 为非纯函数.

fp 的 reverse 为纯函数

fp 的函数都是 curry 之后的 (fp 的文档在 fp Guide 中，较为有用)

2) 函子在开发中的使用场景 (对值的内部的值进行处理)

① 控制副作用 (IO)、处理异常 (Either)、异步任务 (Task)

② folktale 库.

3) 函数式编程 { 纯函数 / 柯里化 / 函数组合

4) 柯里化的实现原理.

{ 参数为一元 / 固化参数 (闭包) / 延迟执行 .

柯里化之后的函数在使用之后是否要手动释放，否则会造成内存泄漏.

5) lodash 的 chain . tap . thru ⟶ 有返回值，使用返回值作下一步结果.

(tap) 无返回值，直接改变值

6) 显示调用 chain 与隐式调用的区别：

_.chain( ) → 惰性求值，需要调用 value( )

_( ) ⟶ 对于只返回单值的方法，会结束链式调用，也是惰性求值.

2. Javascript 性能优化.

1) forEach 快于 for ⟶ 有函数作用域.

↓ 对应的不调用函数 .

2) 函数的活动对象.

① 执行上下文
- 全局
- 函数
- eval

② 执行阶段
- 准备阶段：创建执行上下文：调用了但在执行之前
- 执行阶段：将变量对象转为活动对象.
  - arguments、变量、函数声明
  - 作用域链.
  - this 指向

[[FunctionLocation]]：代码位置.

[[Scopes]]：数组

3. 其他问题.

1) 闭包：
- 外部对内部有引用
- 在另一作用域访问一作用域内的局部变量.

2) gulp & webpack 的区别.
- webpack → 处理 web 组件大
- gulp → 其他非 web 的项目.