

Peter the Great
Saint-Petersburg Polytechnic University

Введение в язык Ocaml



Презентацию подготовили:

- Дрекалов Никита
- Соколов Артём

План презентации

- История появления
- Область применения
- Способы быстрого запуска
- Пример: программа «Здравствуй, мир!»

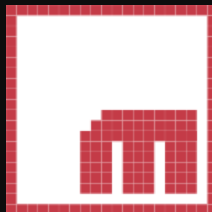
История появления



- **OCaml** — это функционально-императивный язык из семейства ML, разработанный в 1996 году во французском институте INRIA группой исследователей, включая Ксавье Леруа, Дамьена Долиге и Дидье Рем
- Название Objective Caml отражает добавление объектной модели к исходному языку **Caml** (Categorical Abstract Machine Language)
- **OCaml** поддерживается активным сообществом и компанией Jane Street, которая применяет язык для высокочастотной торговли

Область применения

- Финансы, торговые системы (Jane Street, Bloomberg)
- Исследования и образование (компиляторы, парсеры, анализаторы)
- Формальная верификация (проект Coq, SPARK)
- Blockchain (язык Michelson для Tezos)
- Научные и инженерные расчёты



Способы быстрого запуска

- На [DockerHub](https://hub.docker.com/r/ocaml/opam) есть официальный образ с предустановленной средой разработки: <https://hub.docker.com/r/ocaml/opam>
 - Интерактивный запуск контейнера

```
docker run -it ocaml/opam
```
 - Компиляция внутри файла `ocaml.ml` и запуск программы

```
ocamlc hello.ml -o hello && ./hello
```
- Также доступны онлайн способы запуска
 - <https://ocaml.org/play>
 - <https://try.ocamlpro.com/>



Пример: программа «Здравствуй, мир!»

- `hello.ml`:

```
print_endline "Здравствуй, мир!"
```

- В консоли:

```
ocamlc hello.ml -o hello
./hello
Здравствуй, мир!
```

- Объяснение:

- функция `print_endline` — встроенная, печатает строку и добавляет перевод строки
- `OCaml` не требует явного `main`, программа выполняется сверху вниз

План следующих 5 презентаций

1. Синтаксис и базовые типы
 - Переменные, функции, кортежи, списки, паттерн-матчинг
2. Модули и интерфейсы
 - Как структурировать код, module и .mli файлы
3. Функциональное программирование
 - Каррирование, функции высшего порядка, иммутабельность
4. Императивные и объектные возможности
 - Работа с состоянием, классы и объекты
5. Сборка и экосистема
 - Opam, dune, тестирование, деплой через Docker

Спасибо за внимание!