

Переменные:

Const:

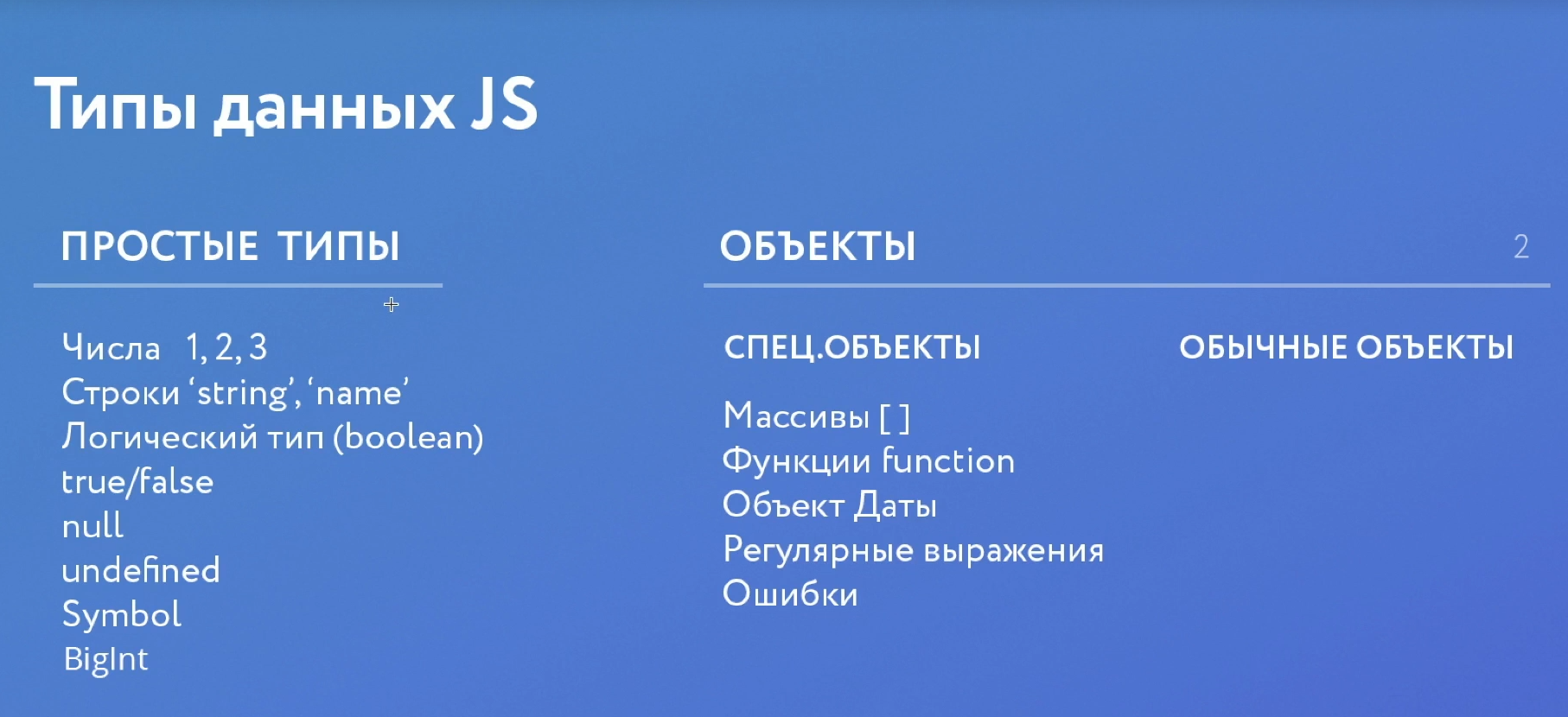
* Блочная область видимости: Переменные const также имеют блочную область видимости и видны только внутри блока, в котором они объявлены.
* Недоступны до объявления: Переменные const не поднимаются вверх в области видимости и не могут быть использованы до фактического объявления.
* Нельзя переобъявлять и перезаписывать значение: Переменные const не могут быть переобъявлены или перезаписаны после инициализации. Они являются константами и должны быть инициализированы при объявлении.

Let:

* Блочная область видимости: Переменные let имеют блочную область видимости. Они видны только внутри блока, в котором они объявлены, и не видны за пределами этого блока.
* Недоступны до объявления: Переменные let не поднимаются вверх в области видимости, поэтому они не могут быть использованы до фактического объявления в коде.
* Нельзя переобъявлять в той же области видимости: Переменные let не могут быть повторно объявлены в той же области видимости, что помогает избежать конфликтов и ошибок.

Var( устарел, вызывает ошибки, но все еще может встречаться)

* Функциональная область видимости: Переменные var имеют функциональную область видимости. Они видны внутри функции, в которой объявлены, и могут быть доступны за пределами блока, в котором они объявлены.
* Поднятие переменных (hoisting): Объявления переменных var поднимаются вверх в области видимости, что означает, что их можно использовать до фактического объявления в коде.
* Переобъявление и перезапись: Переменные var могут быть переобъявлены в той же области видимости, и их значения могут быть перезаписаны.
* Типы данных:



Операторы

1. Арифметические операторы:
   * + - сложение
   * - - вычитание
   * \* - умножение
   * / - деление
   * % - остаток от деления
   * ++ - инкремент
   * -- - декремент
2. Операторы присваивания:
   * = - присваивание
   * += - присваивание с добавлением
   * -= - присваивание с вычитанием
   * \*= - присваивание с умножением
   * /= - присваивание с делением
   * %= - присваивание с остатком от деления
3. Операторы сравнения:
   * == - равенство
   * != - неравенство
   * === - строгое равенство
   * !== - строгое неравенство
   * > - больше
   * < - меньше
   * >= - больше или равно
   * <= - меньше или равно
4. Логические операторы:
   * && - логическое И
   * || - логическое ИЛИ
   * ! - логическое НЕ
5. Битовые операторы:
   * & - битовое И
   * | - битовое ИЛИ
   * ^ - битовое исключающее ИЛИ (XOR)
   * ~ - битовое НЕ
   * << - сдвиг влево
   * >> - сдвиг вправо
   * >>> - беззнаковый сдвиг вправо
6. Операторы условия:
   * ? : - тернарный оператор
7. Остальные операторы:
   * typeof - возвращает тип переменной
   * instanceof - проверяет, является ли объект экземпляром определенного класса
   * in - проверяет наличие свойства в объекте
   * void - оператор
   * delete - удаляет свойство объекта
   * new - создает новый объект

Условные операторы:

* + if: Выполняет блок кода, если условное выражение истинно.
  + else: Выполняет блок кода, если предыдущее условное выражение в if было ложным.
  + else if: Позволяет проверить несколько условных выражений и выполнить первый блок кода, который истинен.
  + Тернарный оператор (condition ? exprIfTrue : exprIfFalse): Выполняет одно из двух выражений в зависимости от условия.

Циклы:

* + for: Повторяет выполнение блока кода определенное количество раз.
  + while: Повторяет выполнение блока кода, пока условие остается истинным.
  + do...while: Повторяет выполнение блока кода один раз, а затем продолжает повторять, пока условие истинно.
  + for...in: Перебирает свойства объекта.
  + for...of (ES6): Перебирает значения итерируемых объектов (например, массивов).

Управление потоком и обработка ошибок:

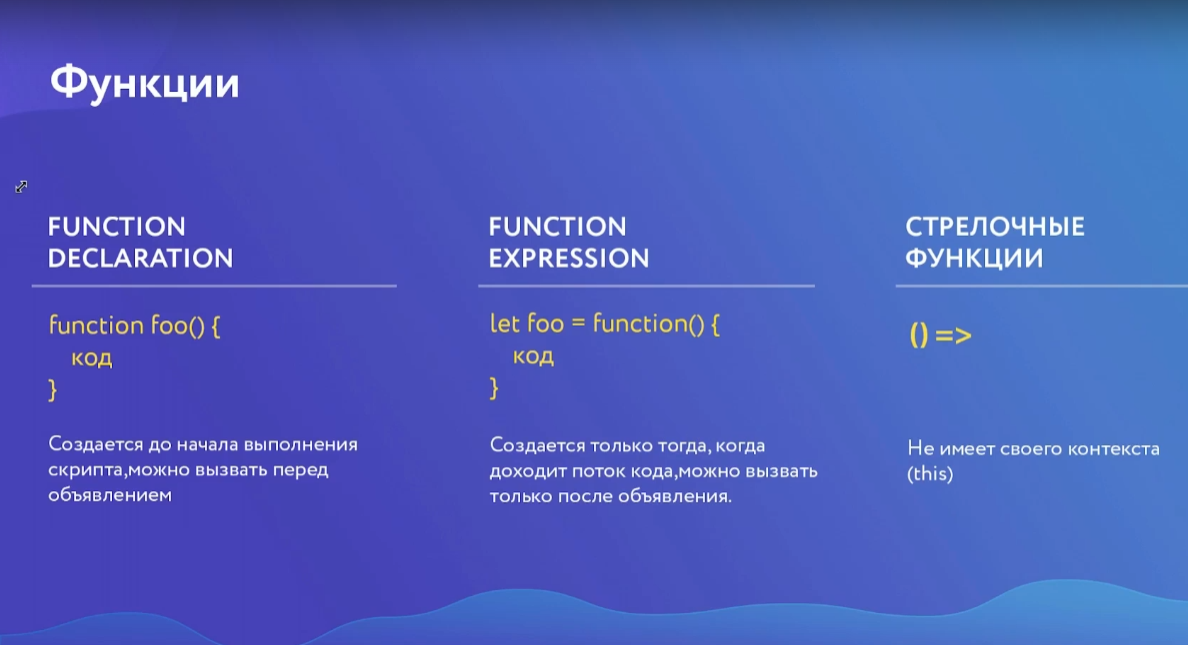
* + break: Прерывает выполнение цикла и переходит к выполнению кода после цикла.
  + continue: Прерывает текущую итерацию цикла и переходит к следующей.
  + try...catch...finally: Позволяет "поймать" ошибки и обработать их, а также гарантировать выполнение некоторого кода в любом случае.
  + throw: Вызывает пользовательскую ошибку.

Оператор switch: Позволяет выполнять различные действия в зависимости от значения переменной или выражения.

Loops

1. for (цикл for): Цикл for используется для выполнения блока кода определенное количество раз.
2. while (цикл while): Цикл while выполняет блок кода, пока условие истинно.
3. do...while (цикл do...while): Цикл do...while выполняет блок кода, а затем проверяет условие. Если условие истинно, цикл продолжается.
4. for...in (цикл for...in): Цикл for...in используется для перебора свойств объекта или индексов элементов массива
5. for...of (цикл for...of): Цикл for...of используется для перебора значений элементов массива или других итерируемых объектов.
6. forEach (цикл forEach): Цикл forEach является методом массива, который выполняет указанную функцию обратного вызова один раз для каждого элемента массива.
7. for await...of (цикл for await...of): Цикл for await...of используется для перебора значений элементов объекта AsyncIterable (например, для обхода асинхронного генератора).

Functions



**Function Declaration (Объявление функции)**: Функции, объявленные с помощью ключевого слова function. Они могут быть объявлены как с именем, так и без имени (анонимные функции).

Function Expression (Функциональное выражение): Функции, которые присваиваются переменным. Имя функции становится доступным только внутри самой функции.

Arrow Function (Стрелочная функция): Сокращенный синтаксис для объявления функций с использованием стрелочного оператора =>. Они автоматически привязывают значение this к окружению, в котором они были созданы.

Method (Метод объекта): Функции, которые являются свойствами объекта. В контексте метода this ссылается на сам объект, в котором он определен.

Generator Function (Генератор): Функции, которые возвращают объект генератора, представляющий последовательность значений, которые можно итерировать с помощью цикла for...of.

Promises и async/await в JavaScript используются для работы с асинхронными операциями, такими как обращения к серверу, таймауты или другие действия, которые не выполняются мгновенно.

* Promise является объектом, который представляет итог асинхронной операции. Promise может находиться в одном из трех состояний: pending (в ожидании), fulfilled (успешно выполнено), или rejected (отклонено из-за ошибки).

**Async/Await** это способ работы с Promise, который позволяет писать асинхронный код, как будто он синхронный. Ключевое слово async используется для объявления асинхронной функции, а await ставится перед Promise, чтобы "подождать" его выполнения.

**2. Objects**

Объекты в JavaScript — это коллекции пар ключ/значение. В объектах значения могут быть любого типа: числа, строки, булевы значения, функции, массивы, другие объекты и т.д.

1. Используйте let и const вместо var: В ES6 введены новые ключевые слова для объявления переменных - let и const, которые имеют блочную область видимости и поэтому более предсказуемы и безопасны для использования.
2. Стрелочные функции: Стрелочные функции предоставляют более короткий и удобный синтаксис для объявления функций и автоматически привязывают контекст this.
3. Модули: Модули ES6 позволяют вам структурировать свой код, импортируя и экспортируя функции, объекты или значения.
4. Promise и async/await: Эти конструкции позволяют работать с асинхронным кодом на более высоком уровне абстракции.
5. Шаблонные строки: Шаблонные строки (обозначаемые косыми кавычками ``) позволяют встраивать выражения прямо в строки и поддерживают многострочный текст.
6. Расширение объектов и массивов (spread и rest): Операторы расширения (...) позволяют вам легко копировать и модифицировать массивы и объекты.
7. Деструктуризация: Деструктуризация позволяет извлекать данные из массивов и объектов в отдельные переменные.
8. Опциональная цепочка (?.) и объединение nullish (??): Эти операторы (введенные в ES2020) упрощают работу с возможно неопределенными или null-значениями.

Примечание Подробнее про изменения в ES6 можно посмотреть в нашей статьях [«Что и как в ES6: хитрости, лучшие практики и примеры. Часть первая»](https://tproger.ru/translations/es6-cheatsheet-1/) и [«Что и как в ES6: хитрости, лучшие практики и примеры. Часть вторая»](https://tproger.ru/translations/es6-cheatsheet-2/).

ES7

1. оператор возведения в степень \*\*;
2. метод Array.prototype.includes, который проверяет, содержится ли переданный аргумент в массиве.

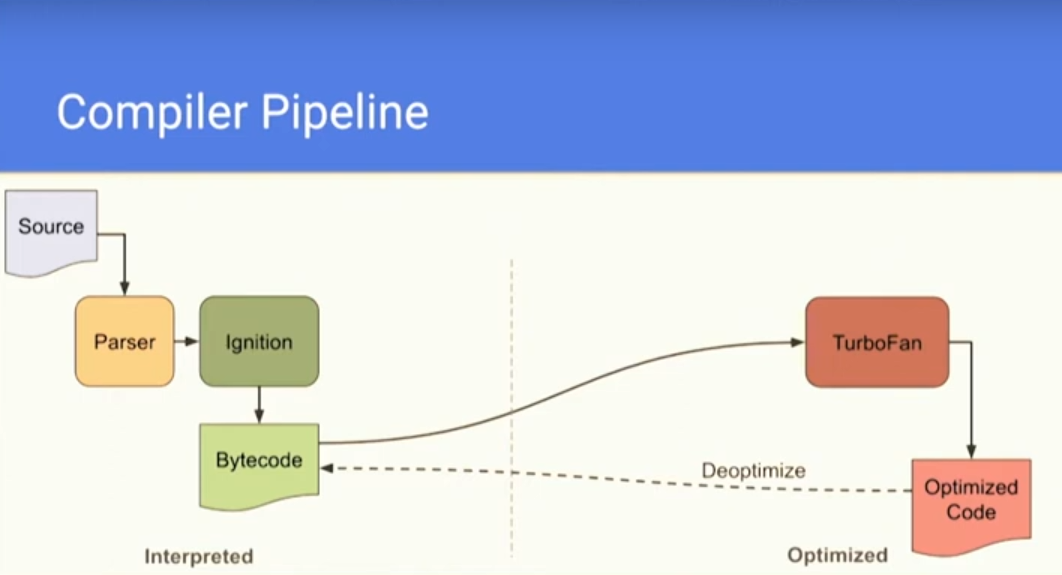
ES8

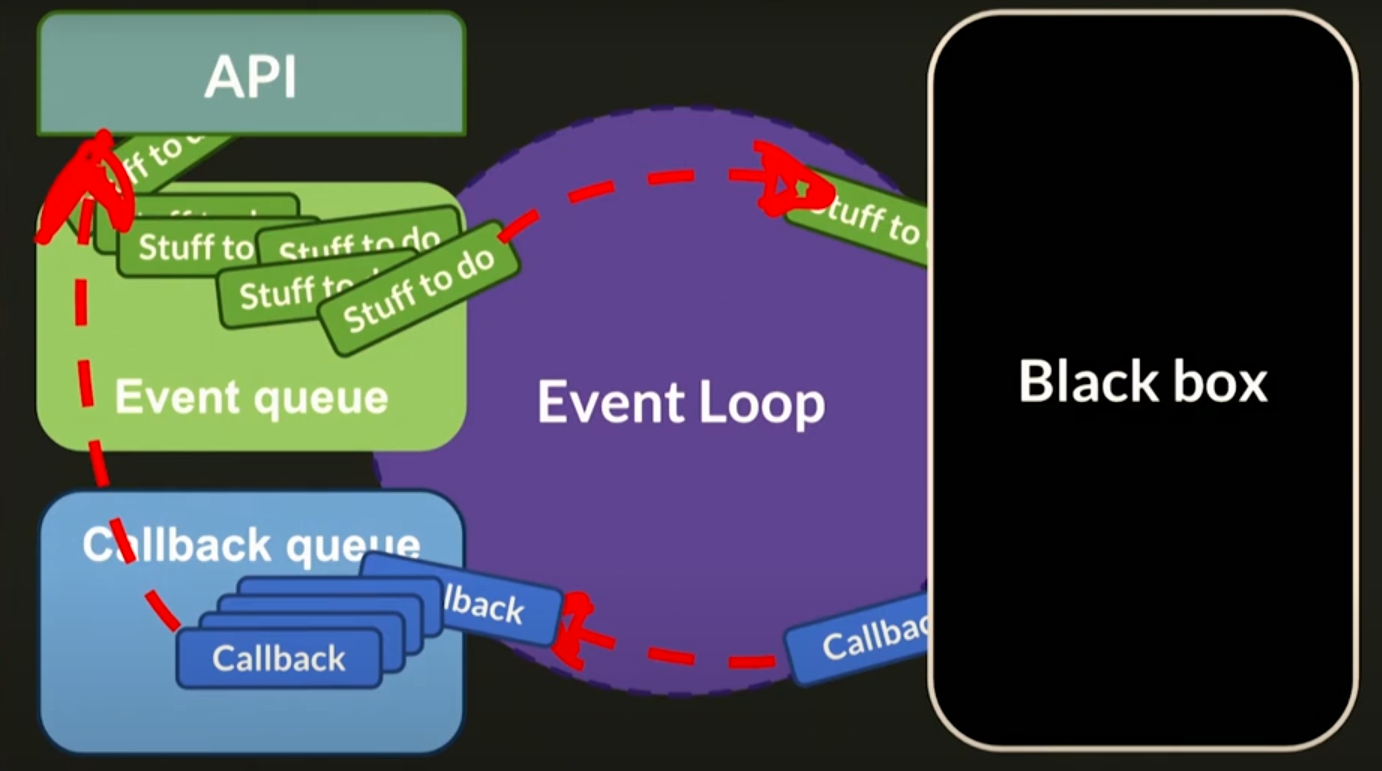
* 1. асинхронность теперь официально поддерживается (async/await);
  2. «висячие» запятые в параметрах функций. Добавлена возможность ставить запятые в конце списка аргументов функций;
  3. добавлено два новых метода для работы со строками: padStart() и padEnd(). Метод padStart() подставляет дополнительные символы слева, перед началом строки. А padEnd(), в свою очередь, справа, после конца строки;
  4. добавлена функция Object.getOwnPropertyDescriptors(), которая возвращает массив с дескрипторами всех собственных свойств объекта;
  5. добавлено разделение памяти и объект Atomics.

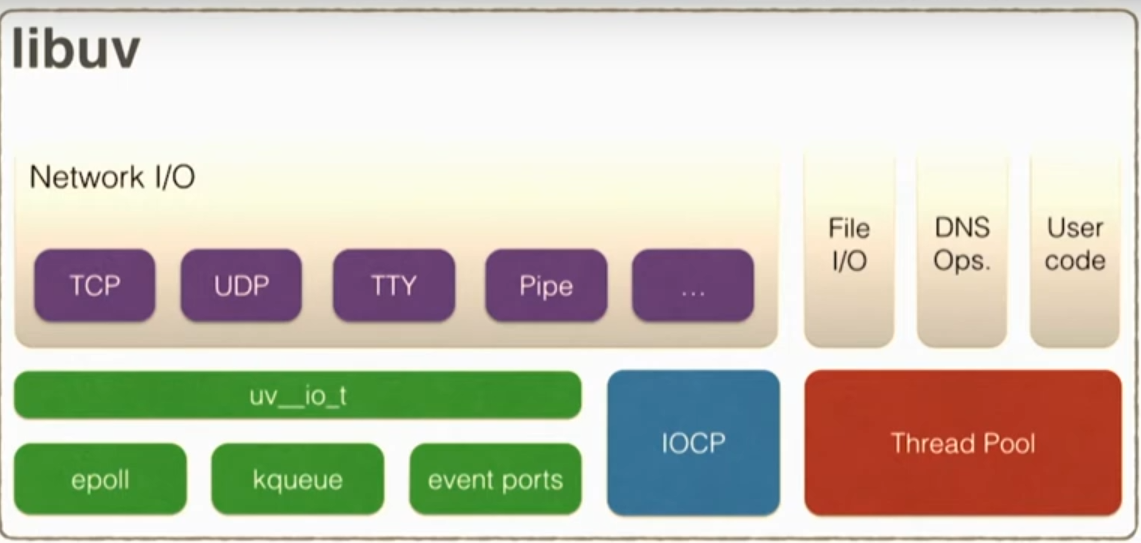
ES9

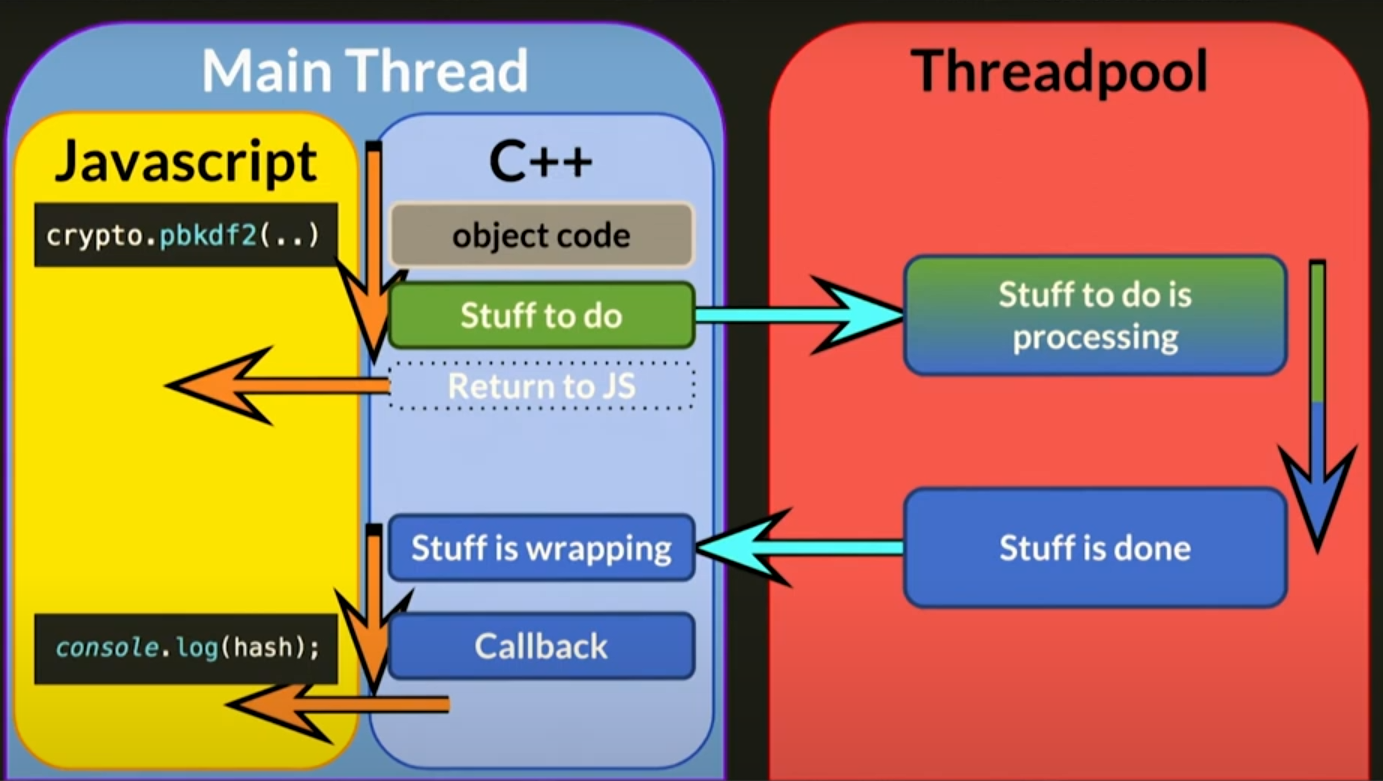
1. Применение операторов spread и rest к объектам.
2. Асинхронные итераторы.
3. Метод Promise.prototype.finally().
4. Улучшения регулярных выражений.

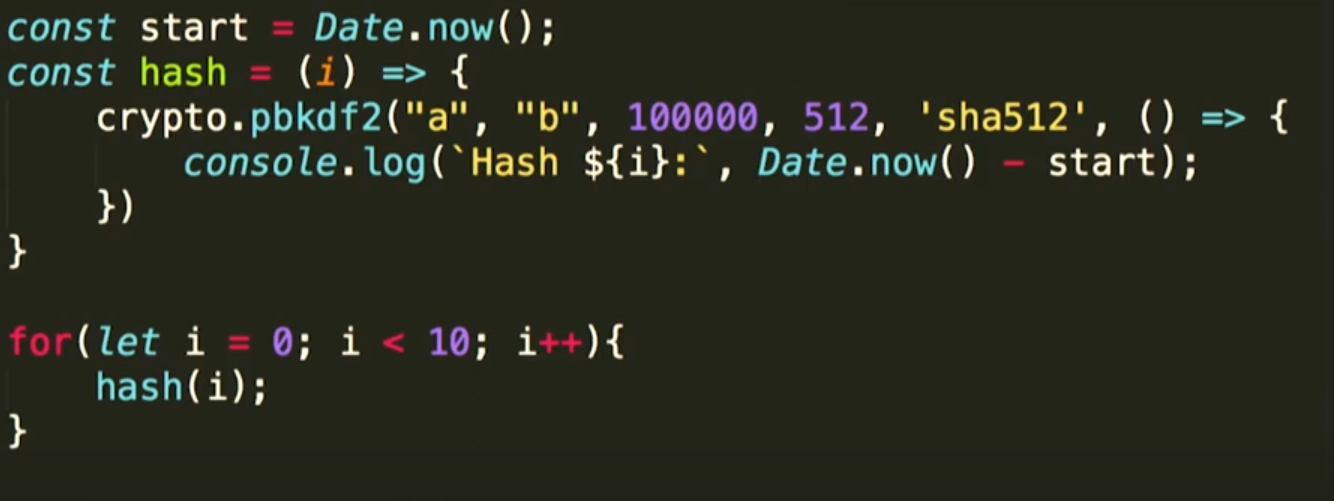
Чуточку про в8











Js однопоточен а тредпул имеет уже 4

