# Режем HTML на куски

## как ускоряли загрузку m.vk.com стримингом

# Обо мне

– работаю в VK;

– 8 лет в разработке;

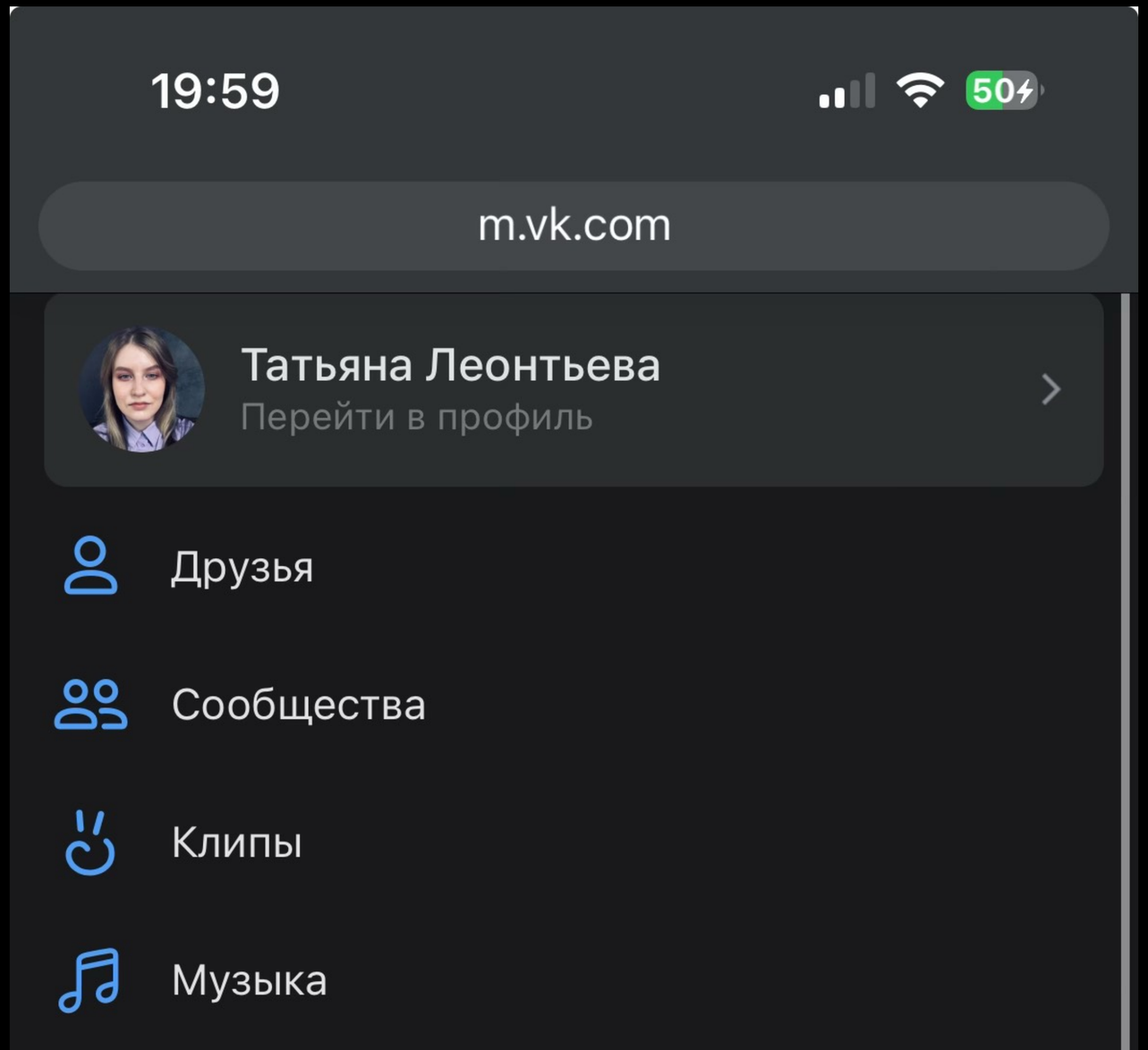
– люблю писать блоги.

# Обо мне

– работаю в VK;

– 8 лет в разработке;

– люблю писать блоги.



# Что обсудим?

# Что обсудим?

- Какую проблему решаем?

# Что обсудим?

- Какую проблему решаем?
- Немного теории про стриминг данных;

# Что обсудим?

- Какую проблему решаем?
- Немного теории про стриминг данных;
- Как разрабатывали и тестировали;

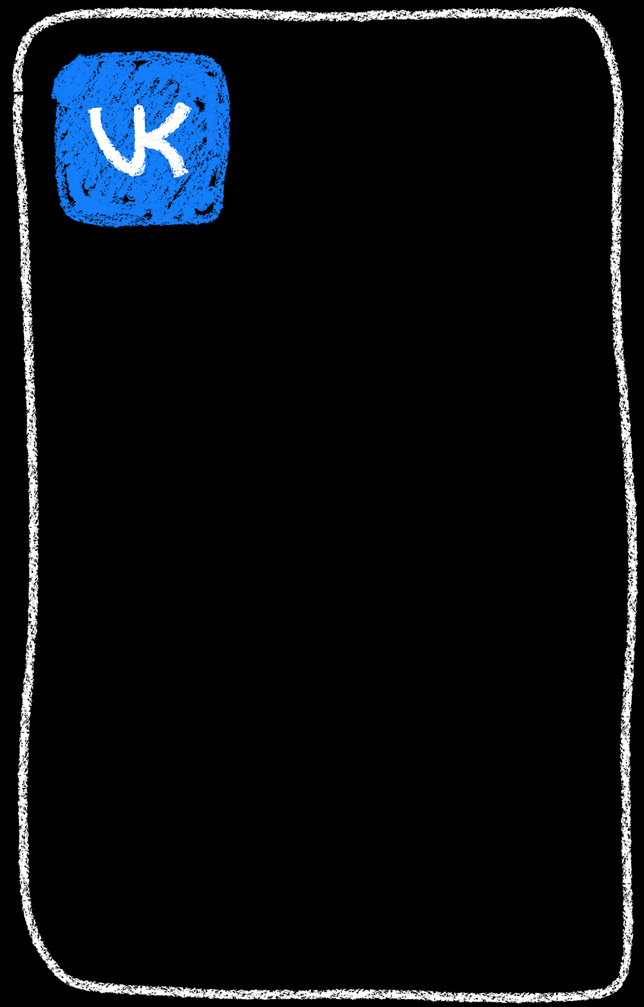
# Что обсудим?

- Какую проблему решаем?
- Немного теории про стриминг данных;
- Как разрабатывали и тестировали;
- Разбираем результаты: Web Vitals

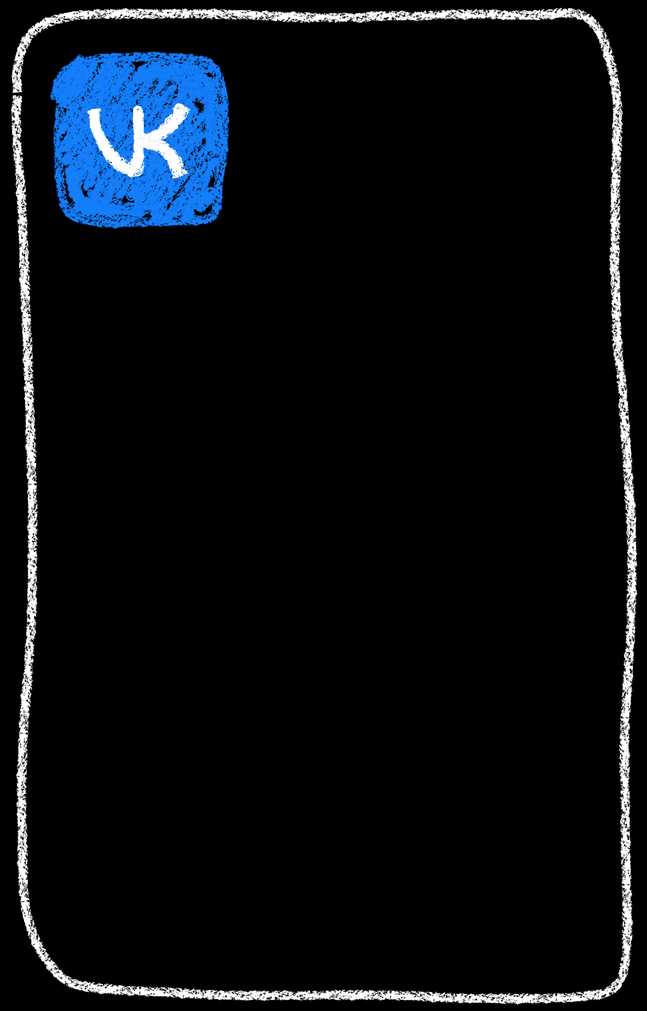
# Что обсудим?

- Какую проблему решаем?
- Немного теории про стриминг данных;
- Как разрабатывали и тестировали;
- Разбираем результаты: Web Vitals
- Подводим итоги

# Это наш монолит

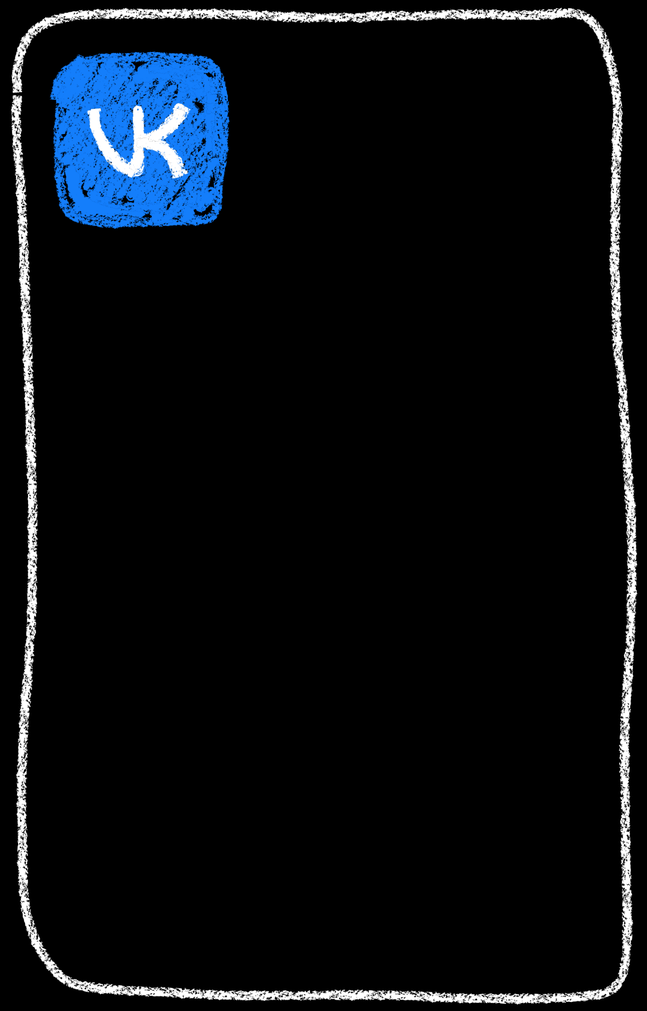


# Это наш монолит

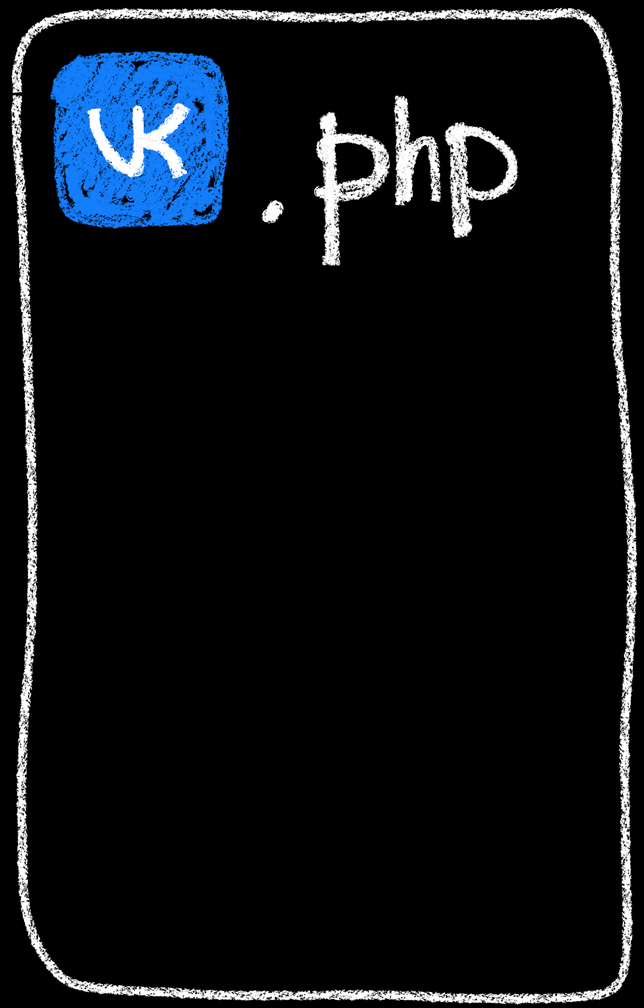


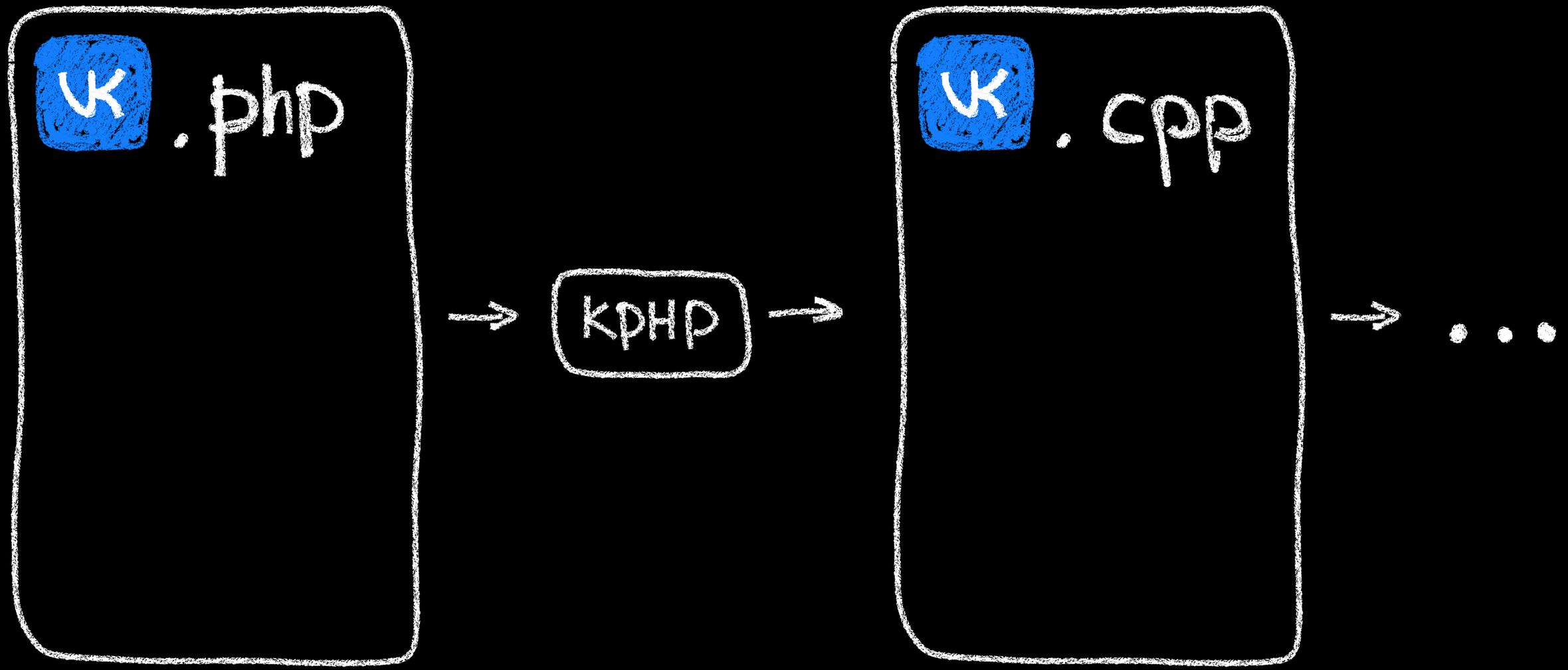
- js;
- php;
- тесты;
- ...

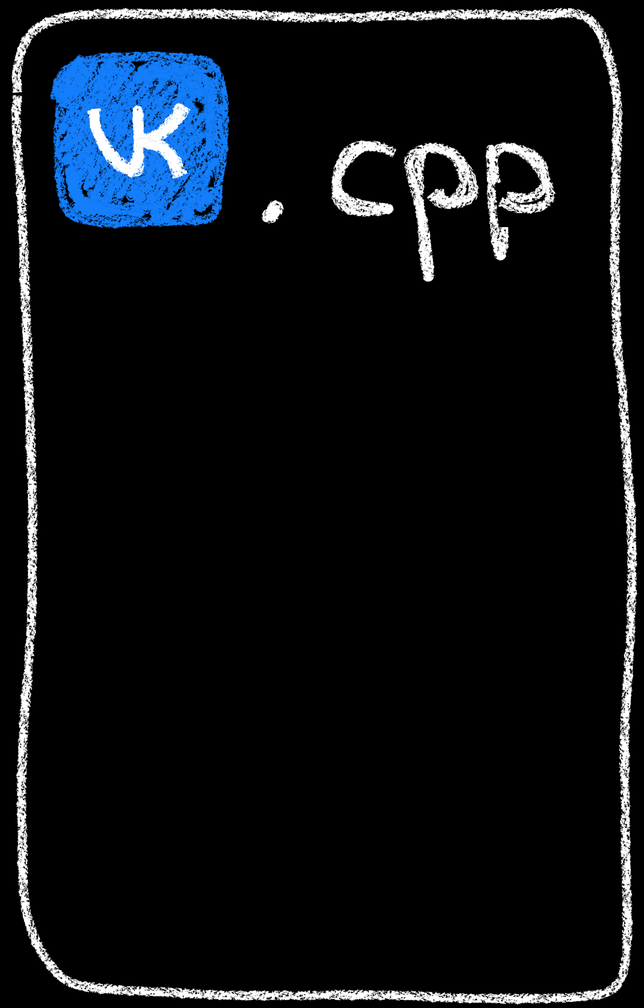
# Это наш монолит

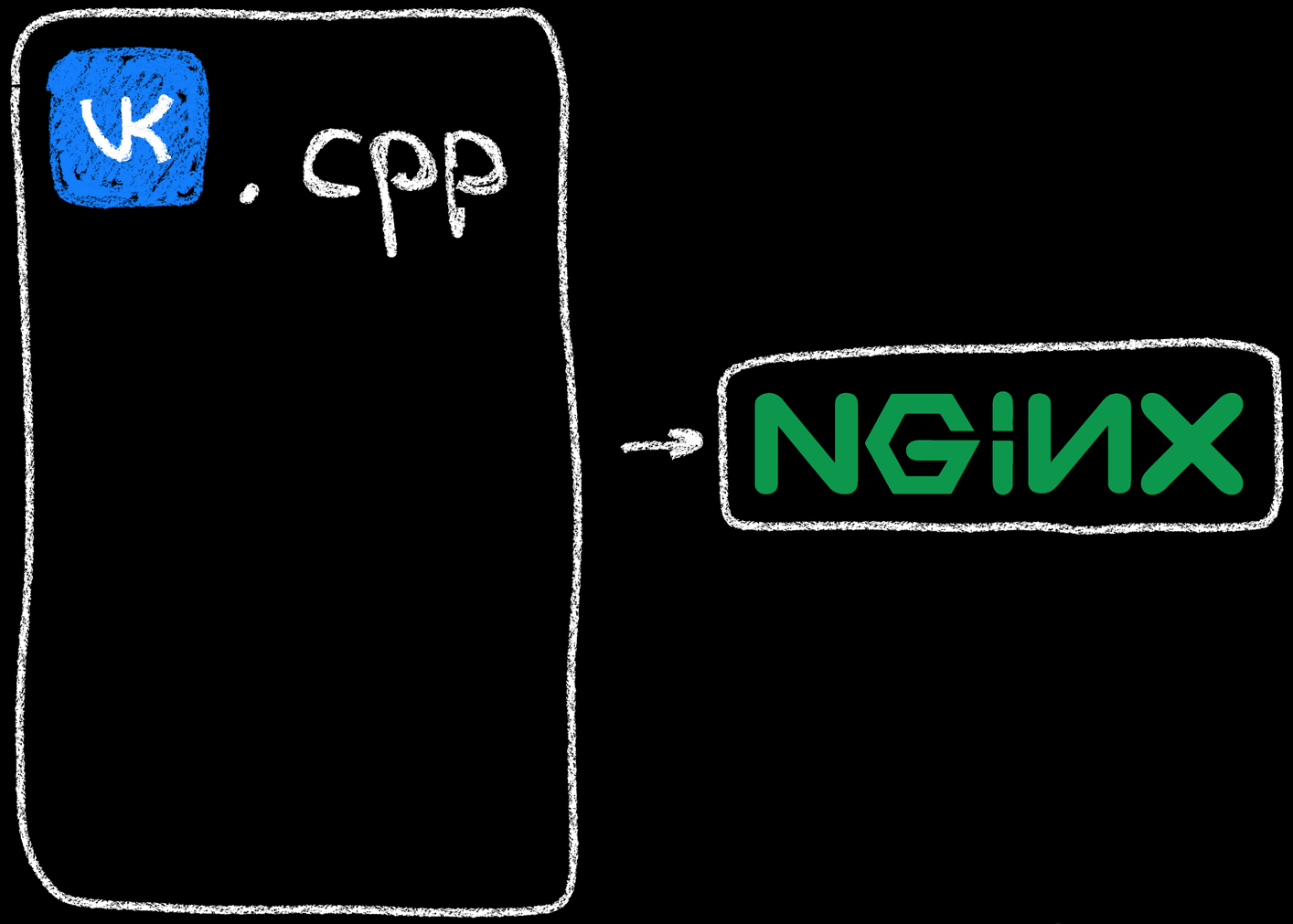


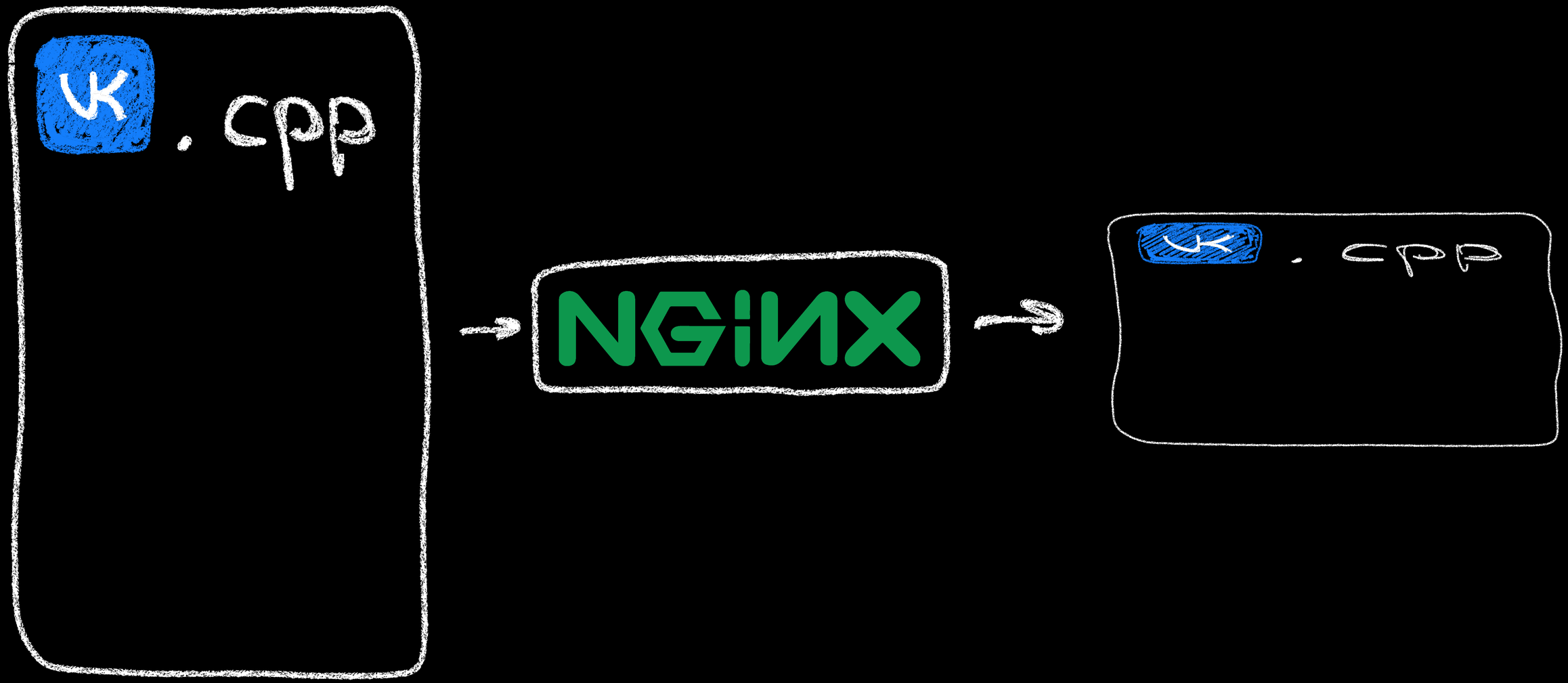
- > 9 млн. строк кода на PHP;
- 25 000 .php файлов;

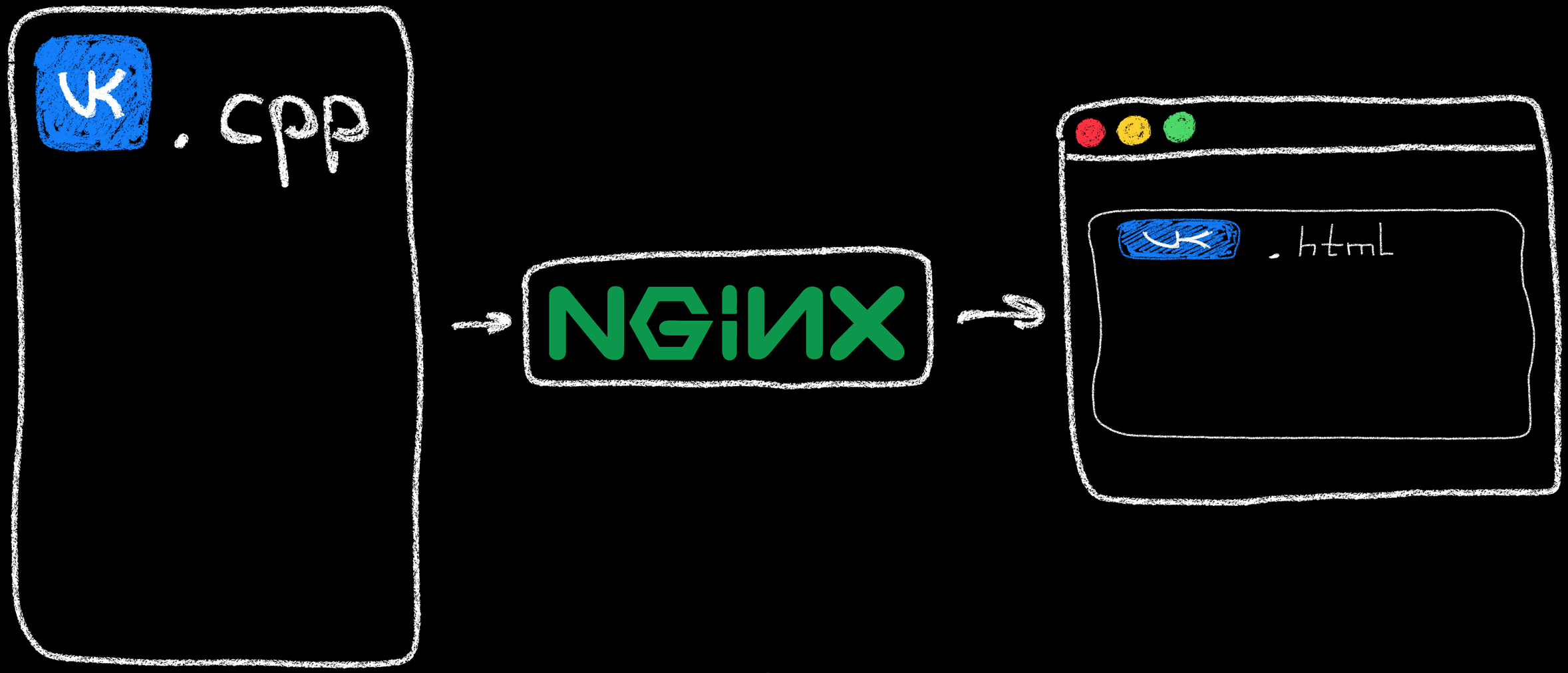

.php


.php
KPHP
.cpp
...


.cpp

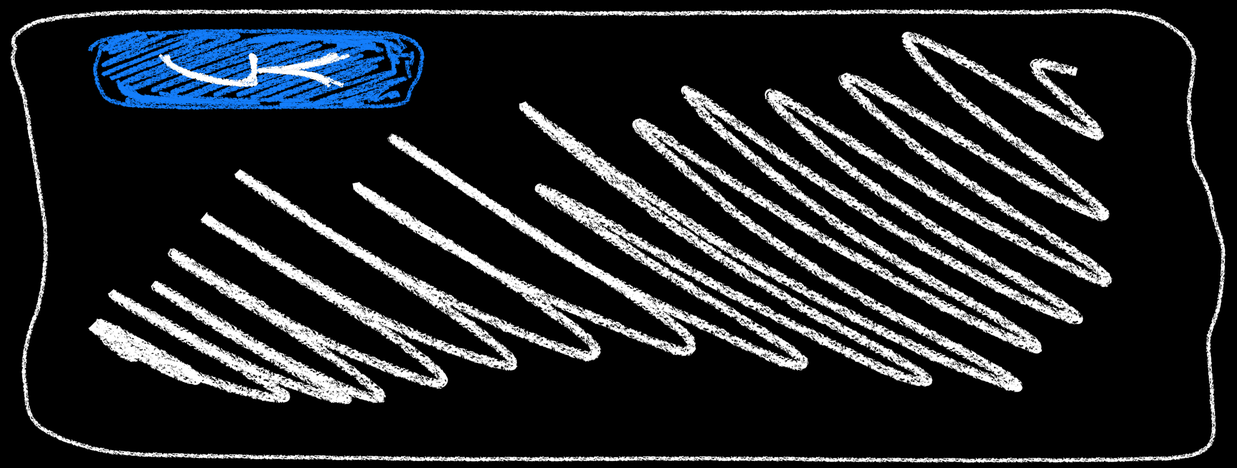

.cpp
NGINX

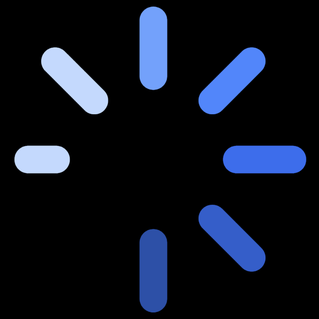

.cpp
NGINX
.cpp


.cpp
NGINX
.html

# жать недостаточно

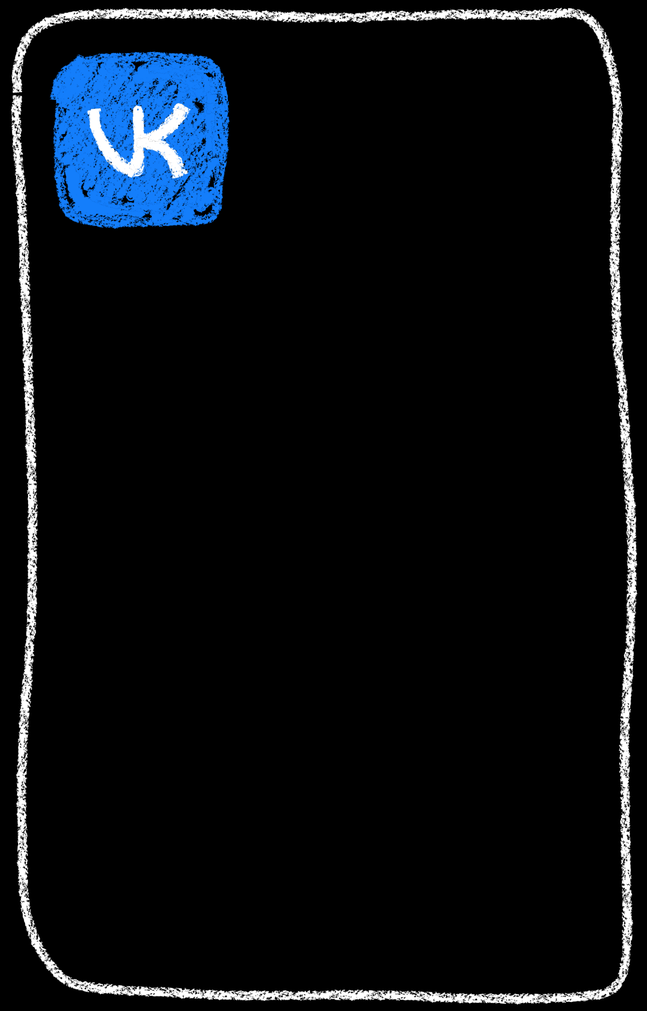
# Белый экран смерти



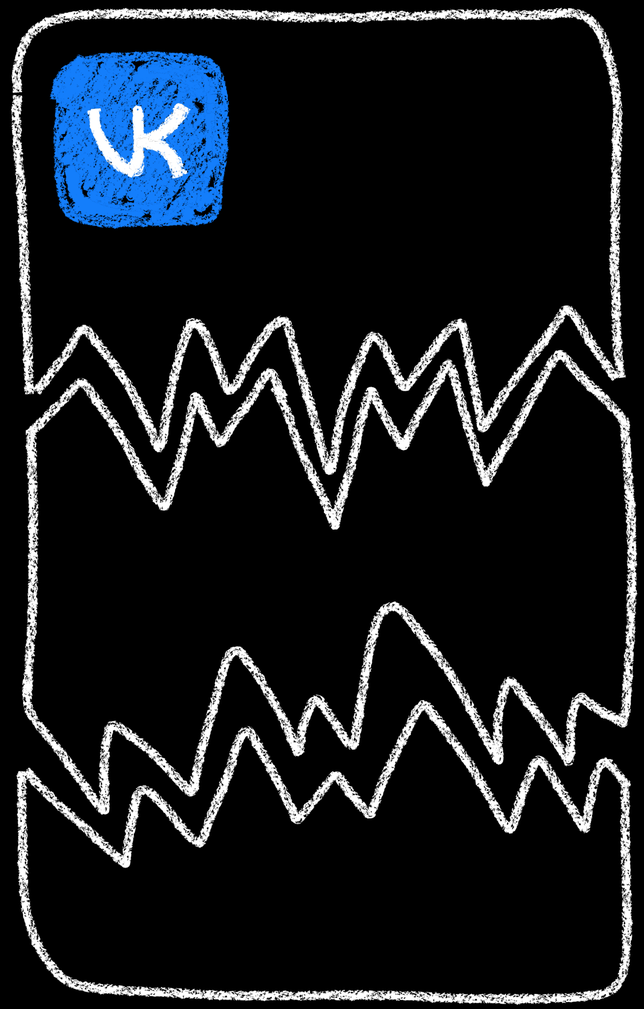


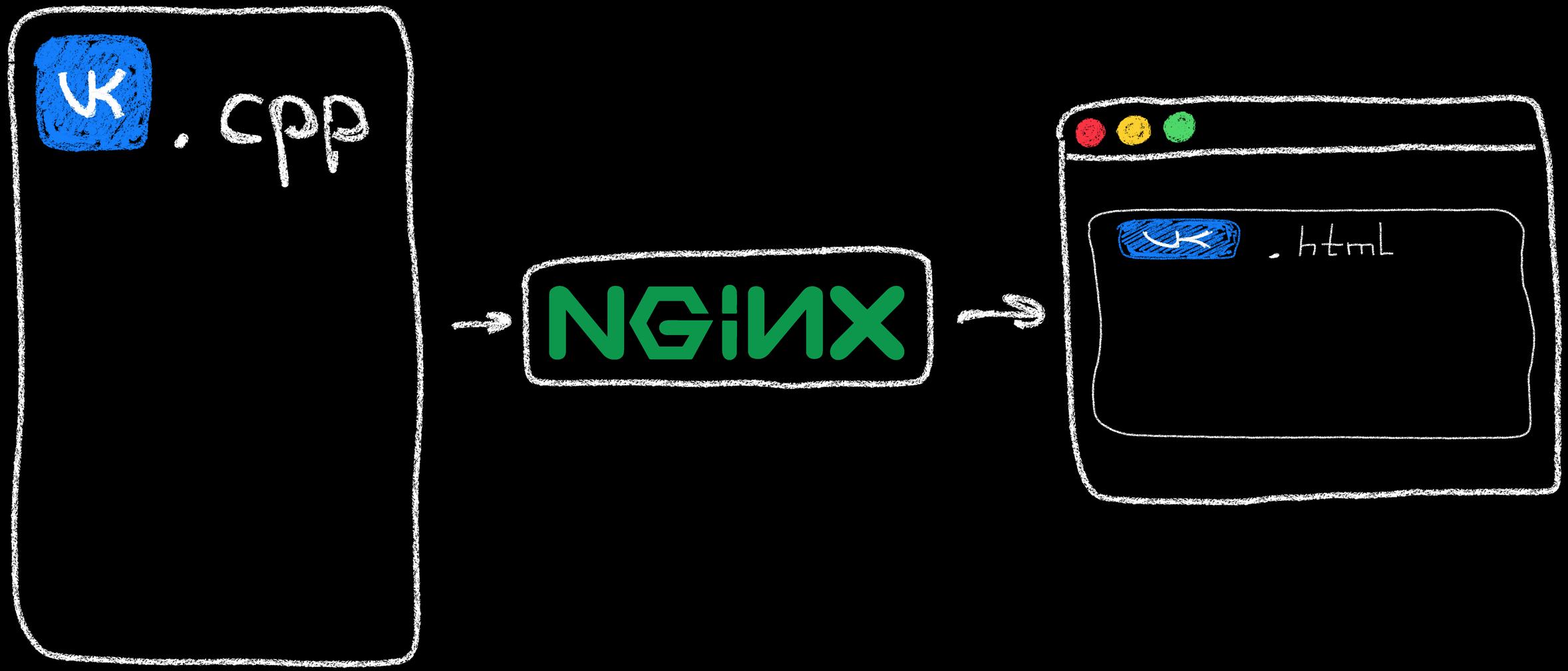
Долгий бэк?

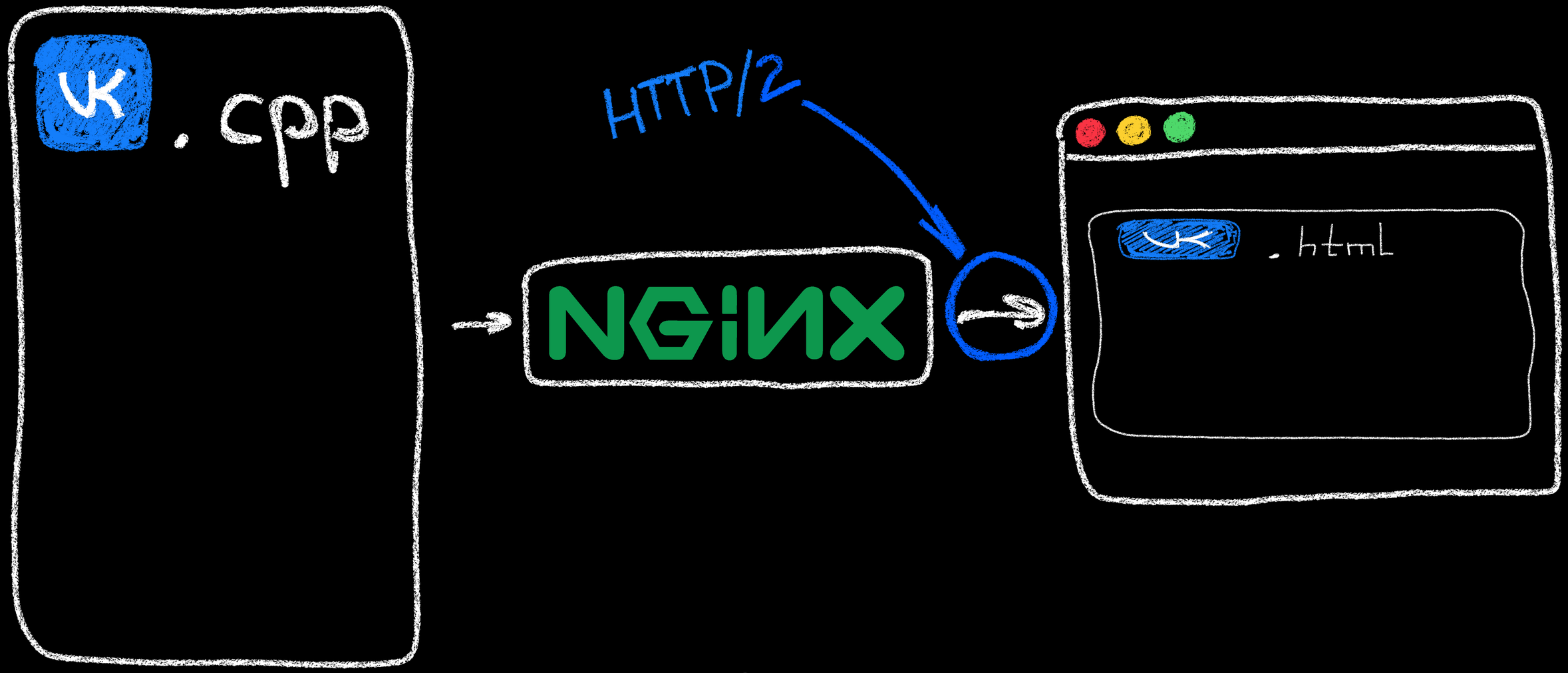
# Меняем стратегию

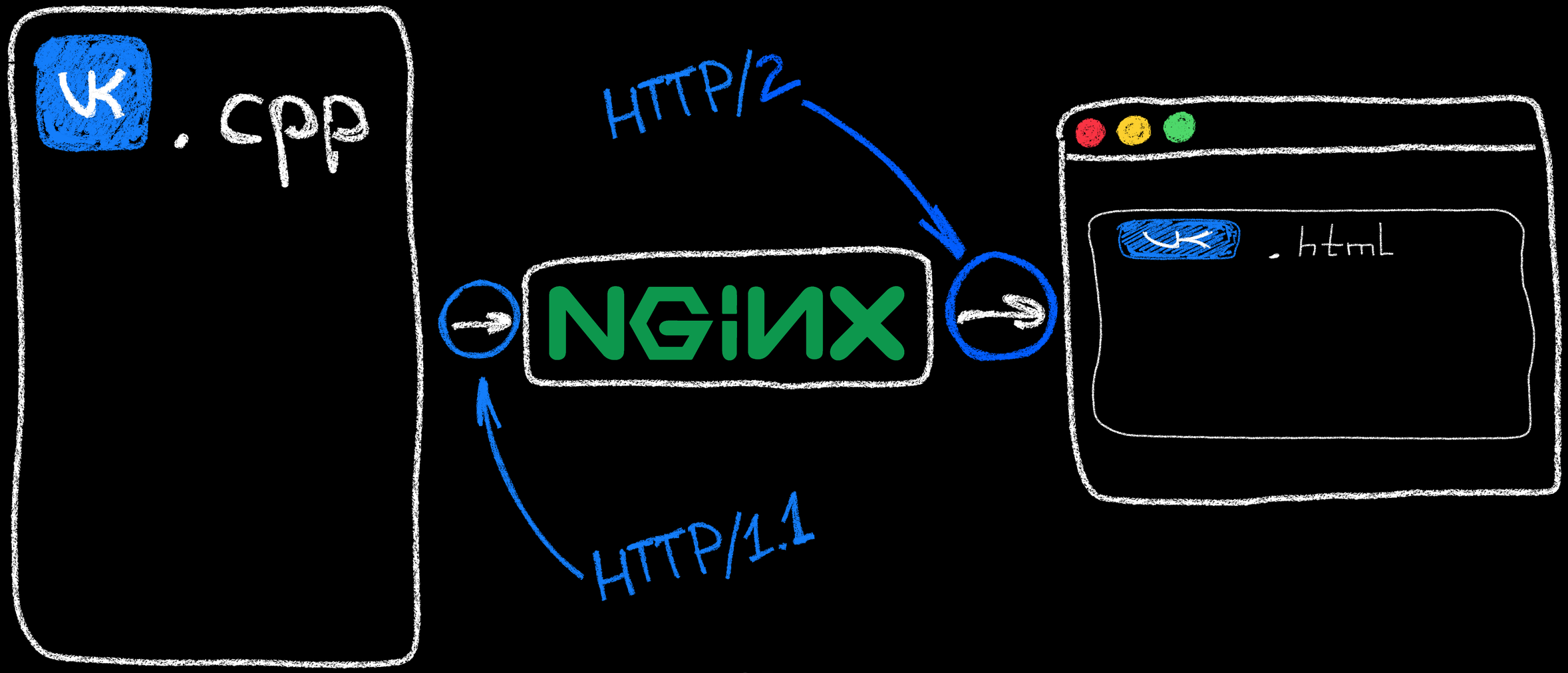


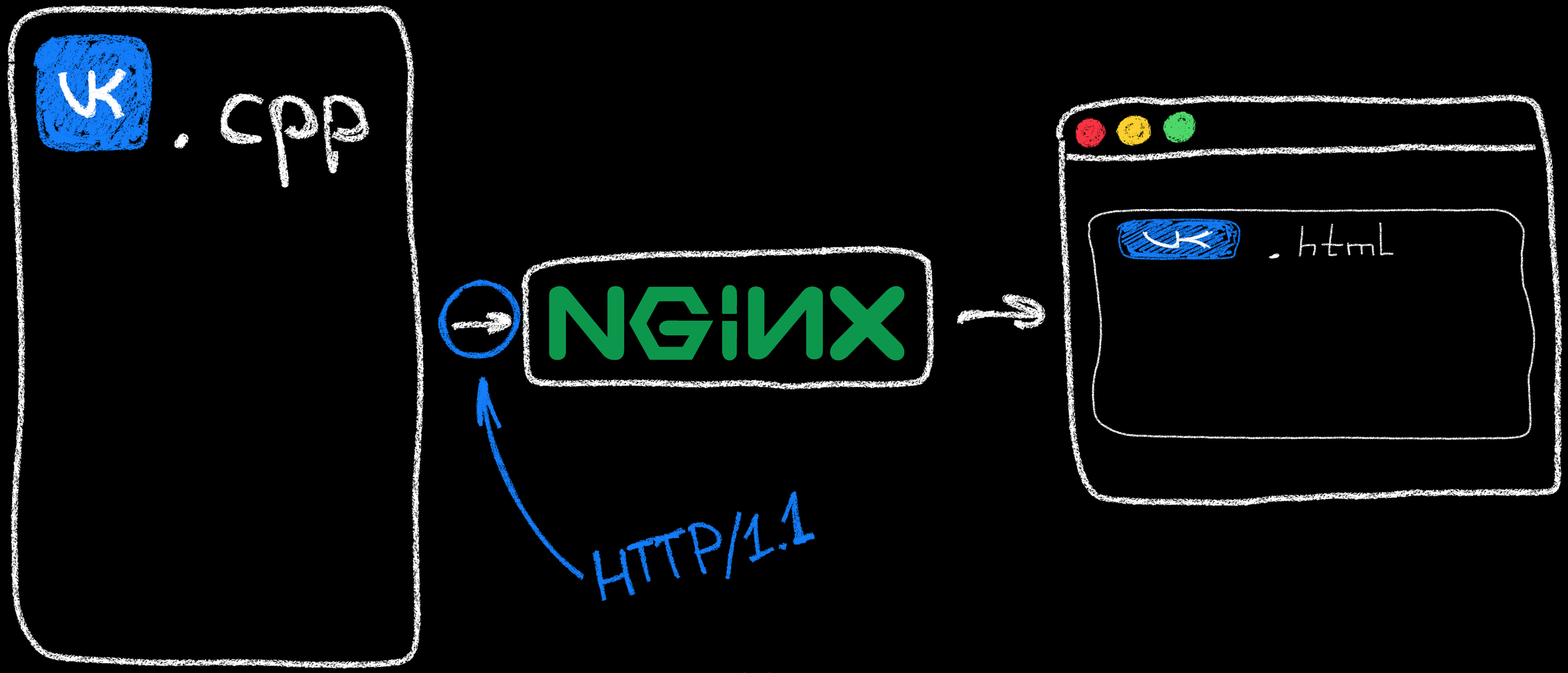
# Меняем стратегию



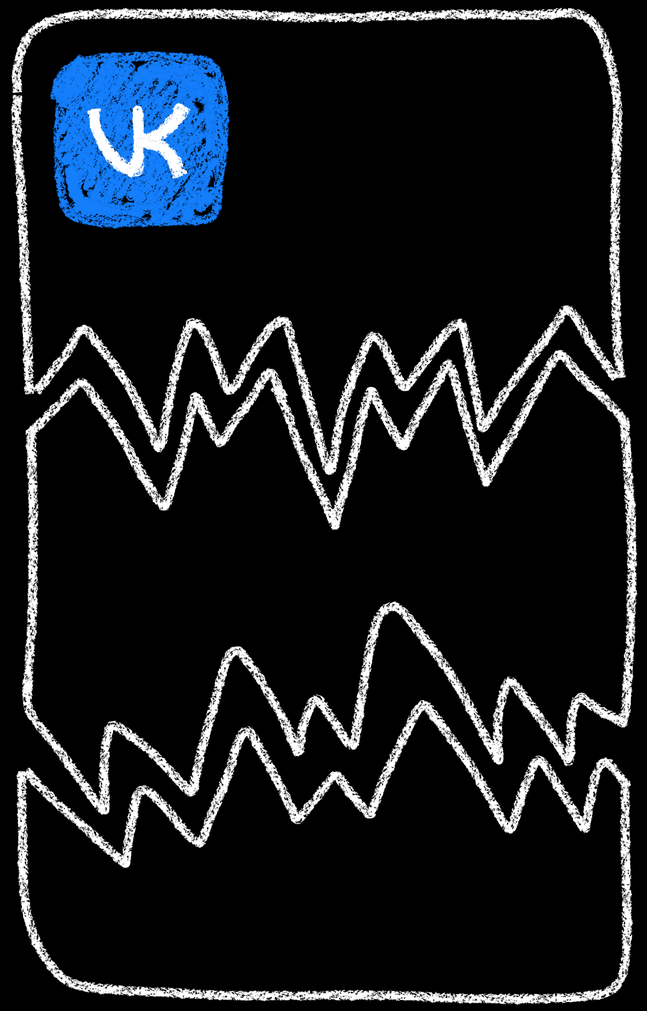

.cpp
NGINX
.html


.cpp
HTTP/2
NGINX
.html

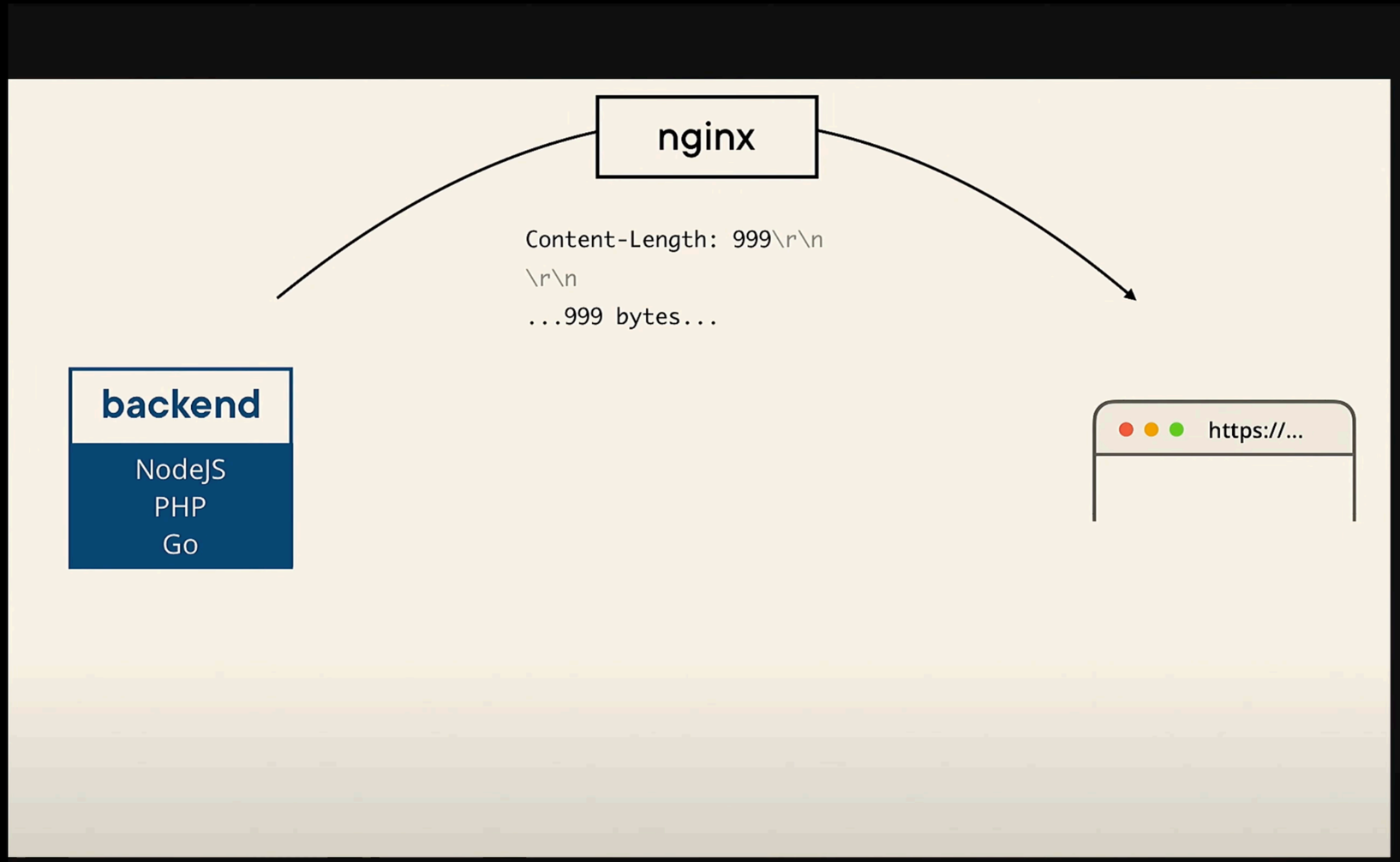

.cpp
HTTP/2
NGINX
.html
HTTP/1.1

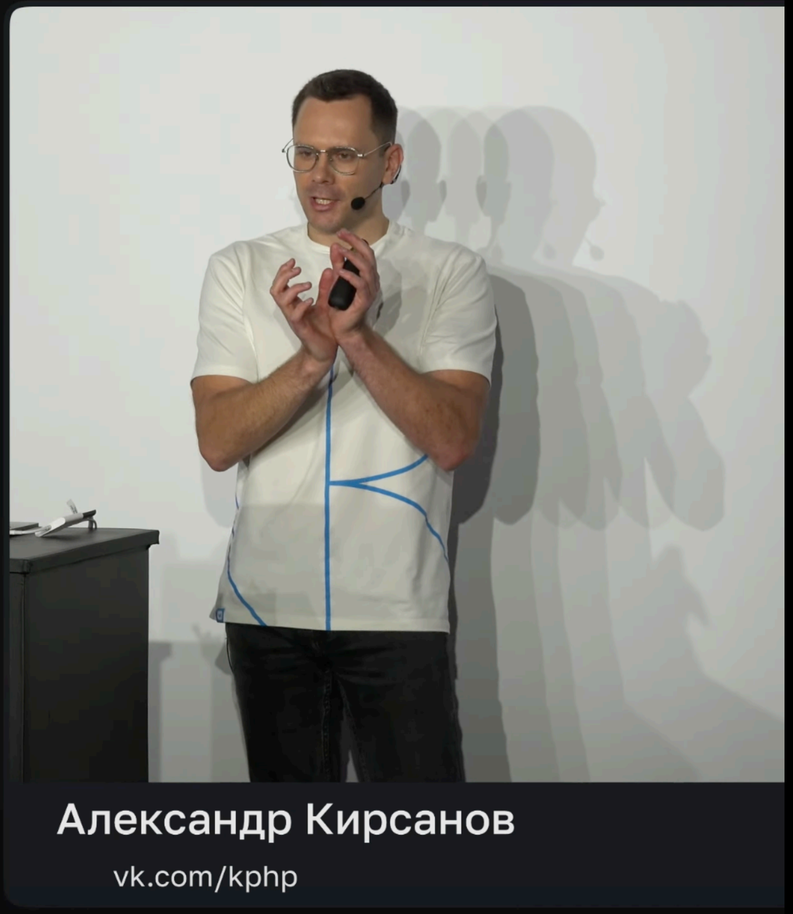

.cpp
NGINX
.html
HTTP/1.1

# Меняем стратегию

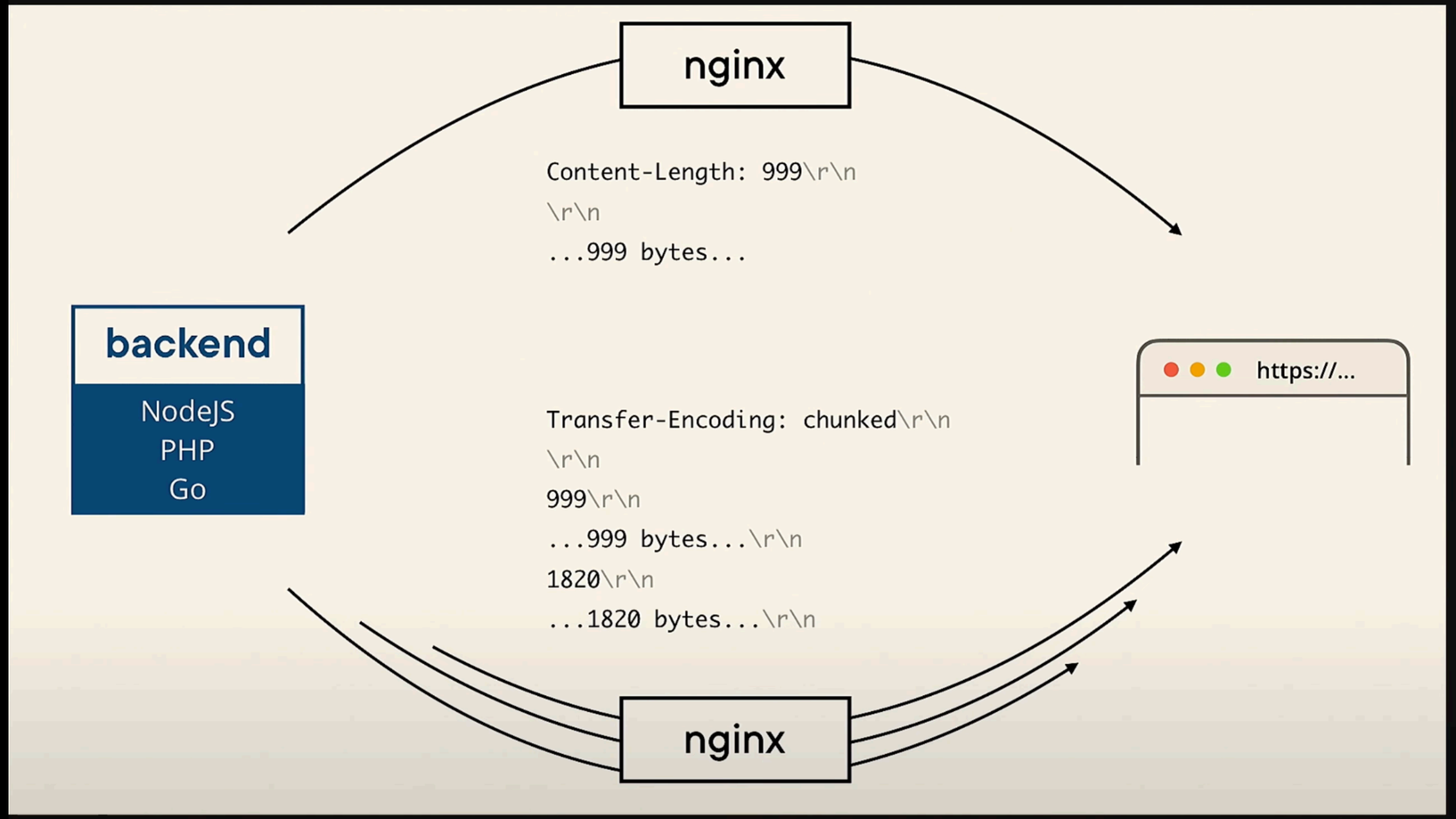


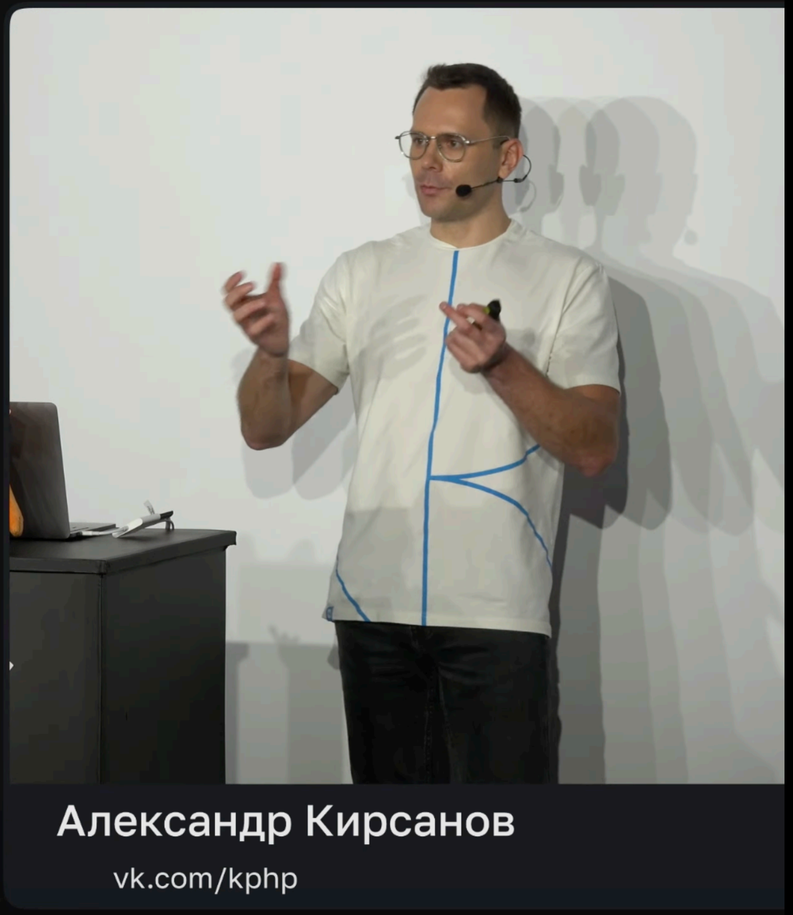
```
Transfer-Encoding: chunked
```


nginx
Content-Length: 999\r\n
\r\n
...999 bytes...
backend
NodeJS
PHP
Go
https://...

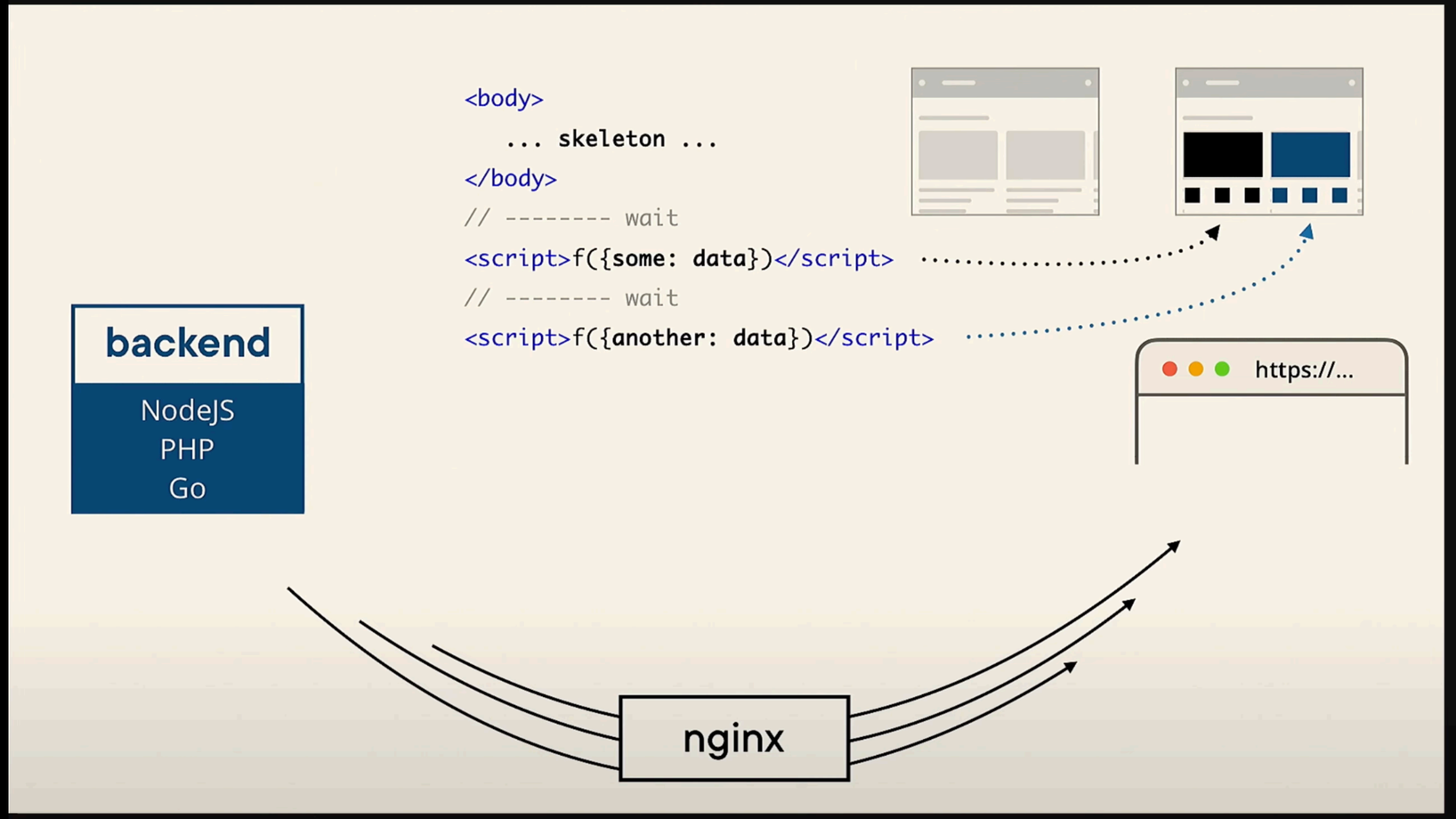

Александр Кирсанов
vk.com/kphp

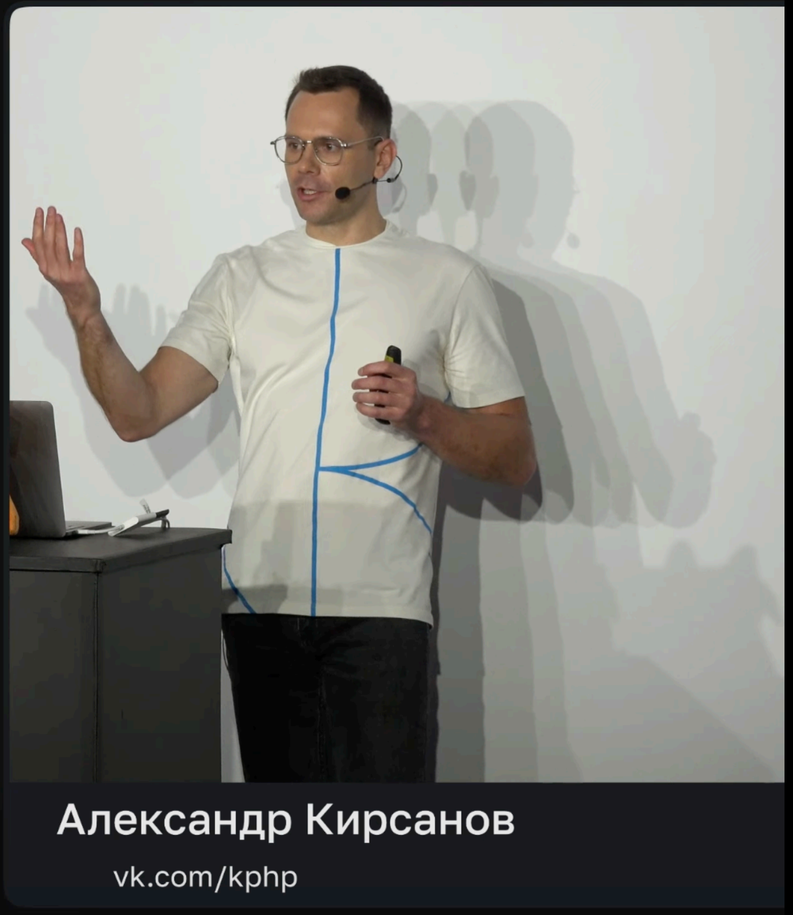
Александр Кирсанов — Про фронтенд с точки зрения плюсовика-компиляторщика



Александр Кирсанов — Про фронтенд с точки зрения плюсовика-компиляторщика



# пример: Node

# Как в ноде?

```
http.createServer(function (request, response) {
  // ...
  response.write(`<h1>Большой заголовок</h1>`);
  response.write(`<p>Текст через 2 секунды</p>`);
  response.write(`<p>Текст через 5 секунд</p>`);
  // ...
  response.end();
}).listen(process.env.VMC_APP_PORT || 8080, null);
```

# Как в ноде?

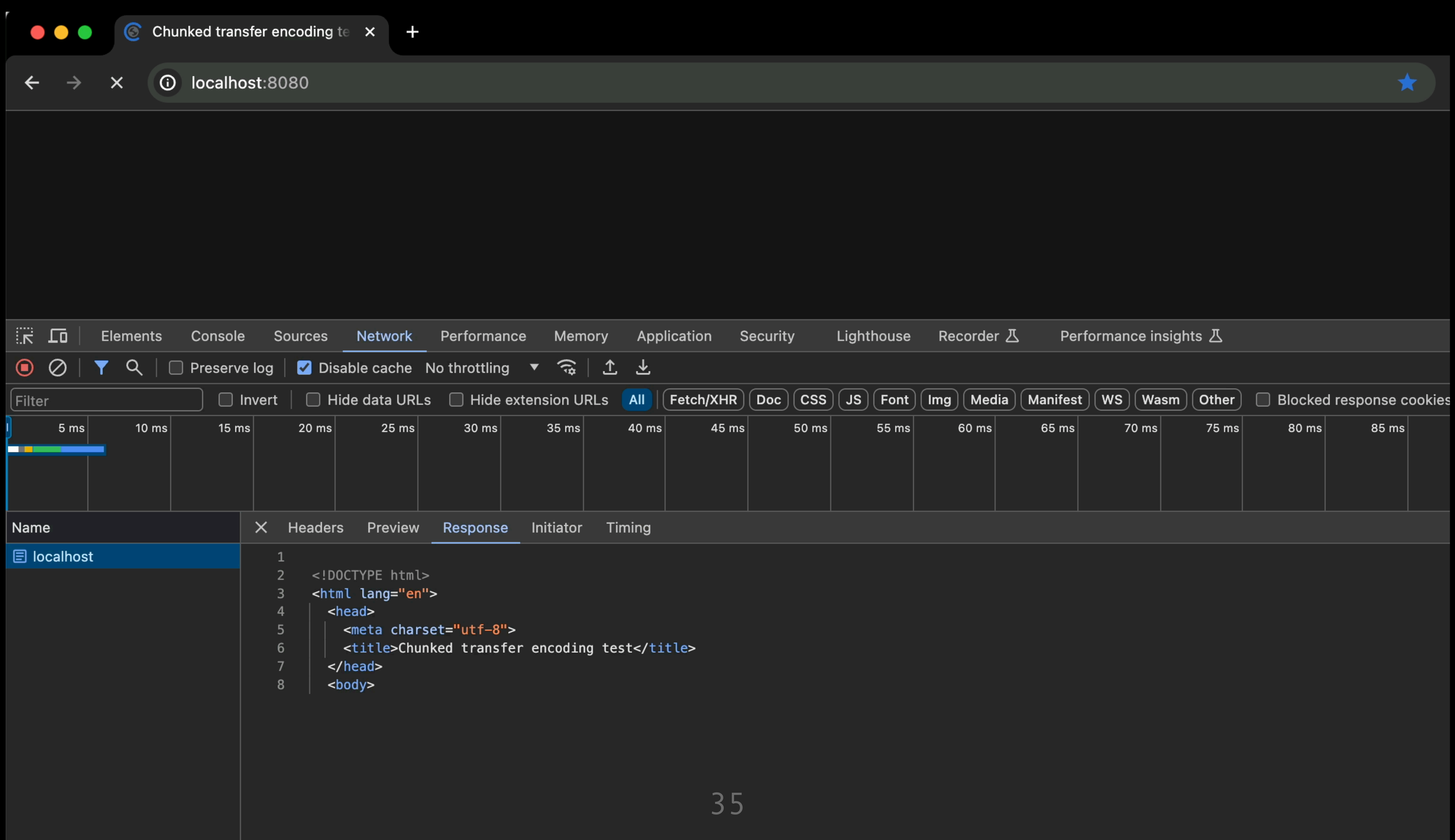
```
http.createServer(fu
  // ...
  response.write(`<h
  response.write(`<p
  response.write(`<p
  // ...
  response.end();
}).listen(process.en
```

```
yepstepz@Tatianas-MacBook-Pro
> Host: localhost:8080
> User-Agent: curl/8.1.2
> Accept: */*
> Connection: close
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 09 May 2024 15:25:57 GMT
< Connection: close
< Transfer-Encoding: chunked
<
```

# Как в ноде?

```
http.createServer(fu
  // ...
  response.write(`<h
  response.write(`<p
  response.write(`<p
  // ...
  response.end();
}).listen(process.en
```

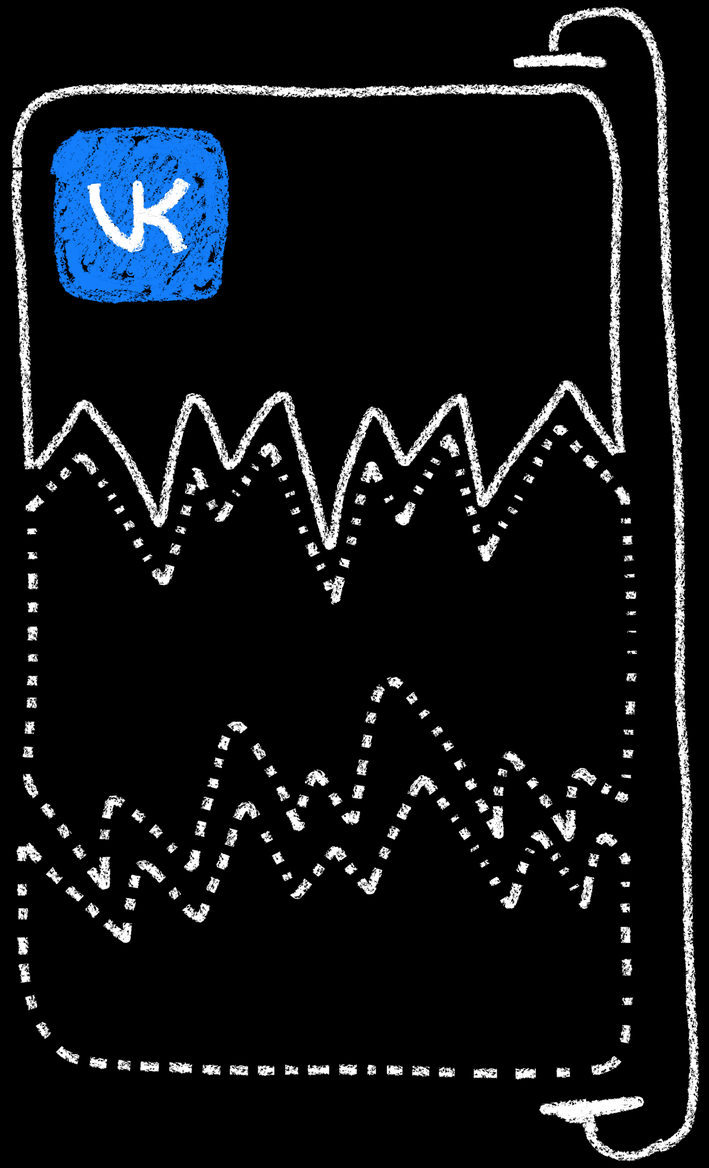
```
yepstepz@Tatianas-MacBook-Pro
> Host: localhost:8080
> User-Agent: curl/8.1.2
> Accept: */*
> Connection: close
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=UTF-8
< Date: Thu, 09 May 2024 15:25:57 GMT
< Connection: close
< Transfer-Encoding: chunked
<
```


Chunked transfer encoding te
localhost:8080
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
Hide extension URLs
All
Fetch/XHR
Doc
CSS
JS
Font
Img
Media
Manifest
WS
Wasm
Other
Blocked response cookies
5 ms
10 ms
15 ms
20 ms
25 ms
30 ms
35 ms
40 ms
45 ms
50 ms
55 ms
60 ms
65 ms
70 ms
75 ms
80 ms
85 ms
Name
Headers
Preview
Response
Initiator
Timing
localhost
1
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5 <meta charset="utf-8">
6 <title>Chunked transfer encoding test</title>
7 </head>
8 <body>

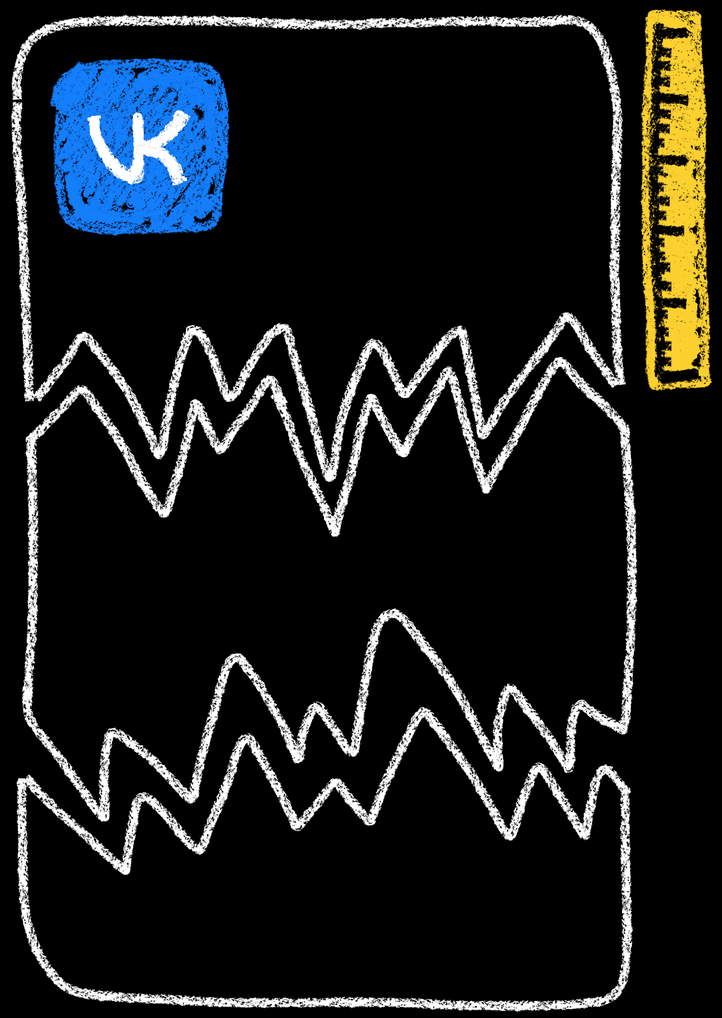
```php
<?php
// ...
    echo "<h1>Большой заголовок</h1>";
    flush();
    ob_flush();

    echo "<p>Текст через 2 секунды</p>";
    flush();
    ob_flush();

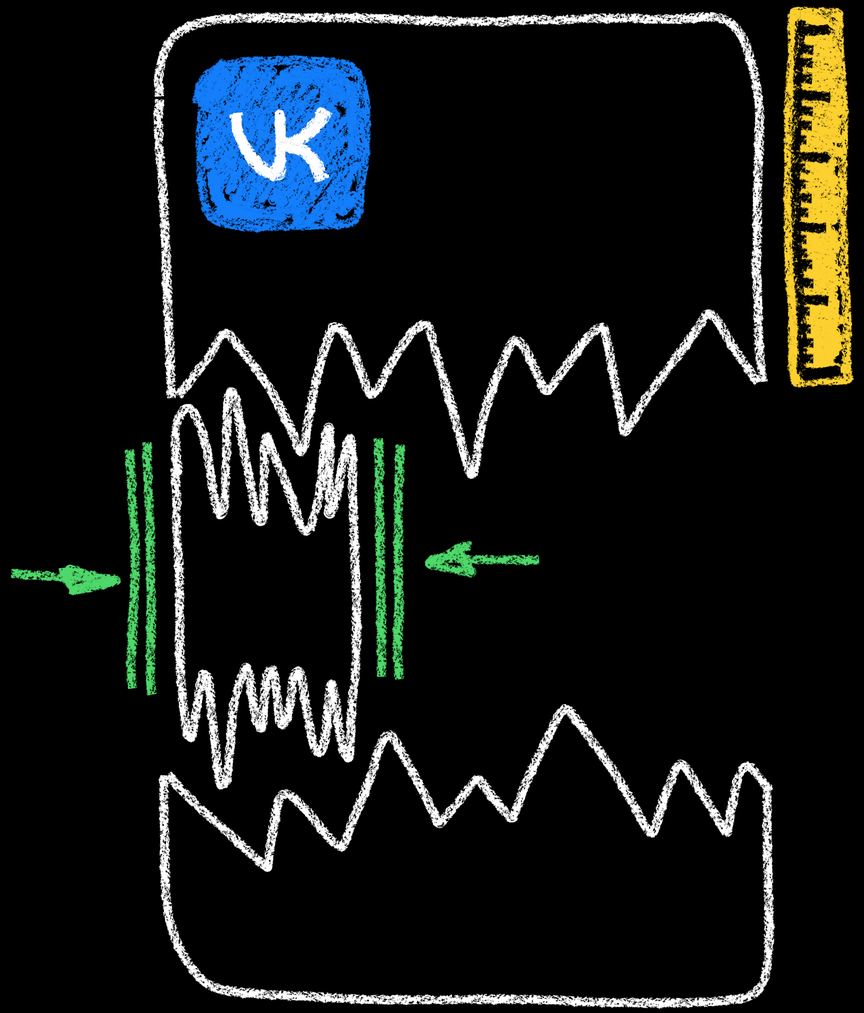
    echo "<p>Текст через 5 секунд</p>";
    flush();
    ob_flush();
// ...
?>
```



SIZE ?

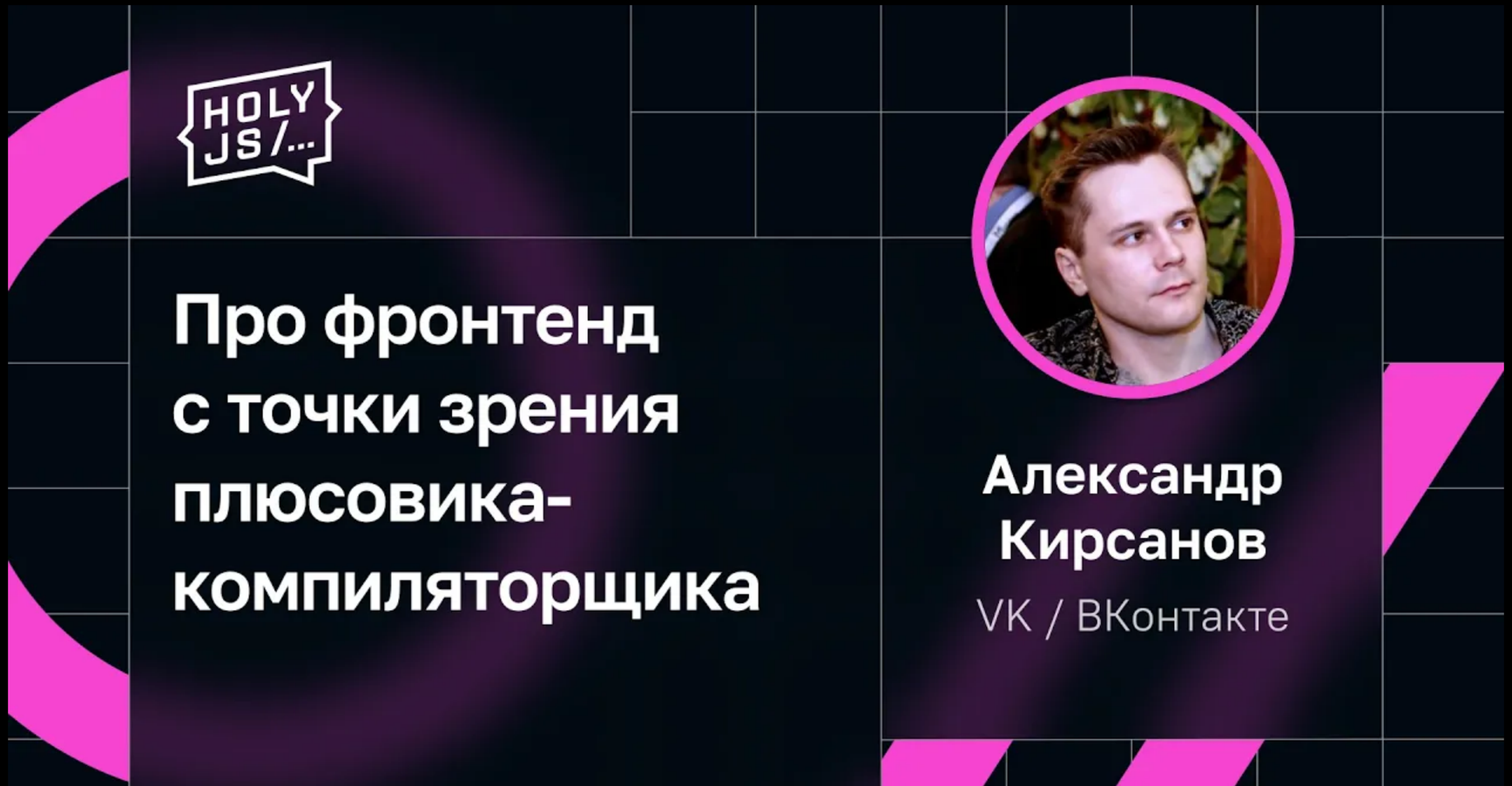


```
Transfer-Encoding: chunked\r\n
\r\n
999\r\n
...999 bytes...\r\n
1820\r\n
...1820 bytes...\r\n
```



# другая история...

# можно посмотреть здесь



# так родился flush()

# сделаем по-быстрому

сделаем по-быстрому
(так мы думали)

# начало разработки

# пишем мега-класс

# Мега-класс

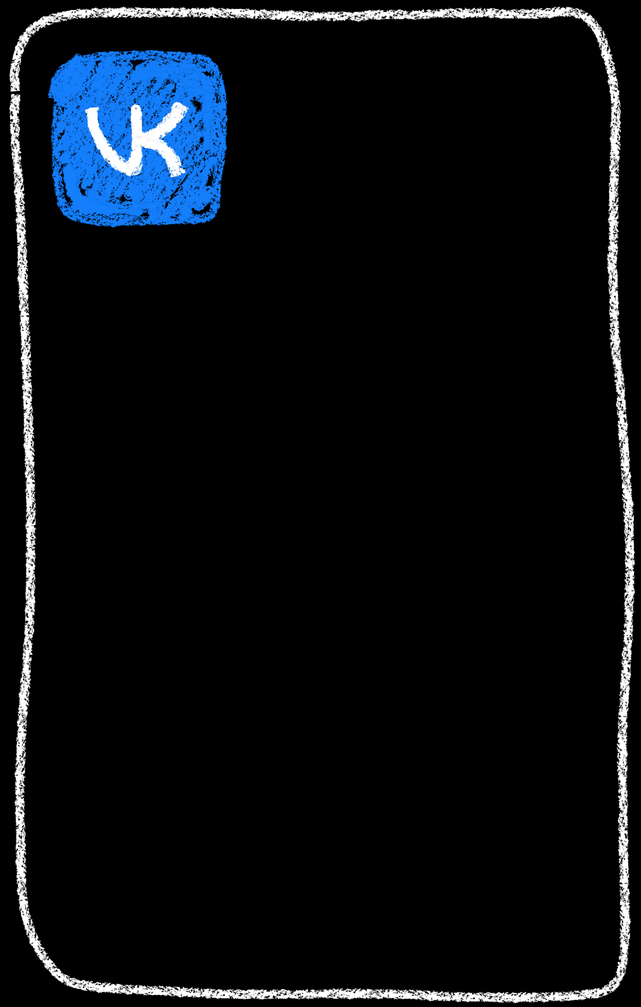
```
class FlushChunks {

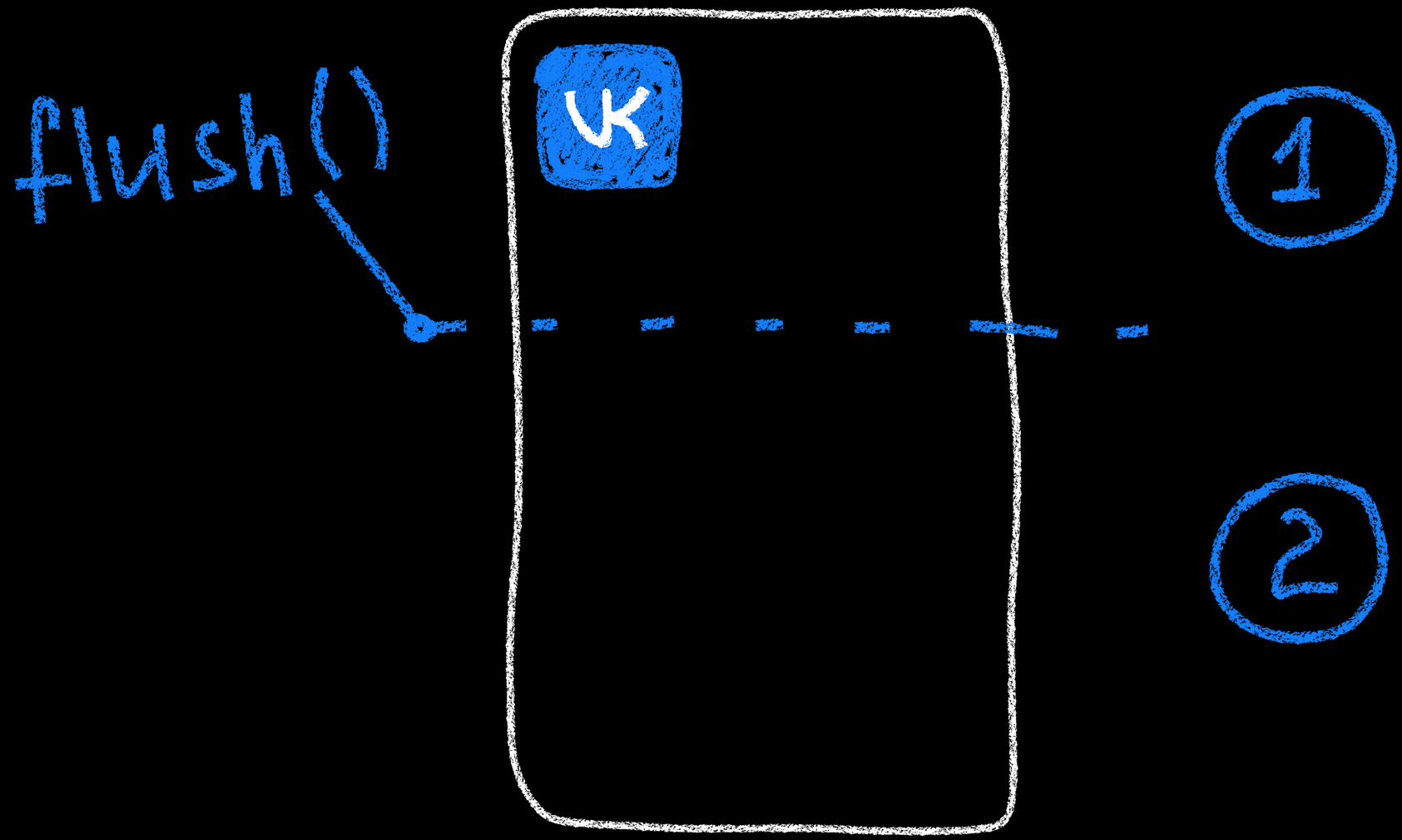
  public static function isFlushAvailable(): bool {
    ...
  }

  public static function earlyFlush(): void {
    flush();
  }

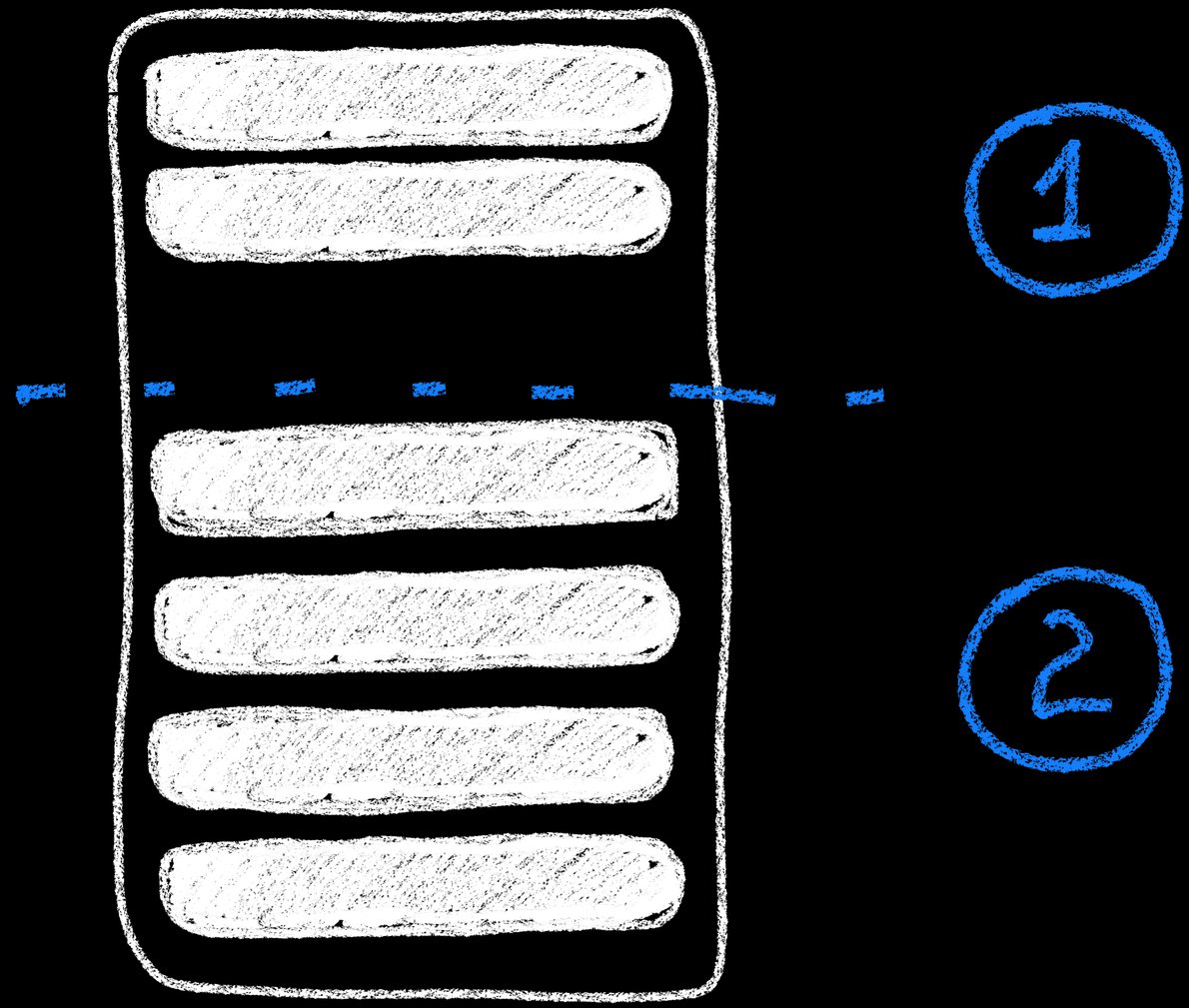
}
```

# Вспоминаем как выглядит страничка




flush()
1
2

# Так люди писали в монолите



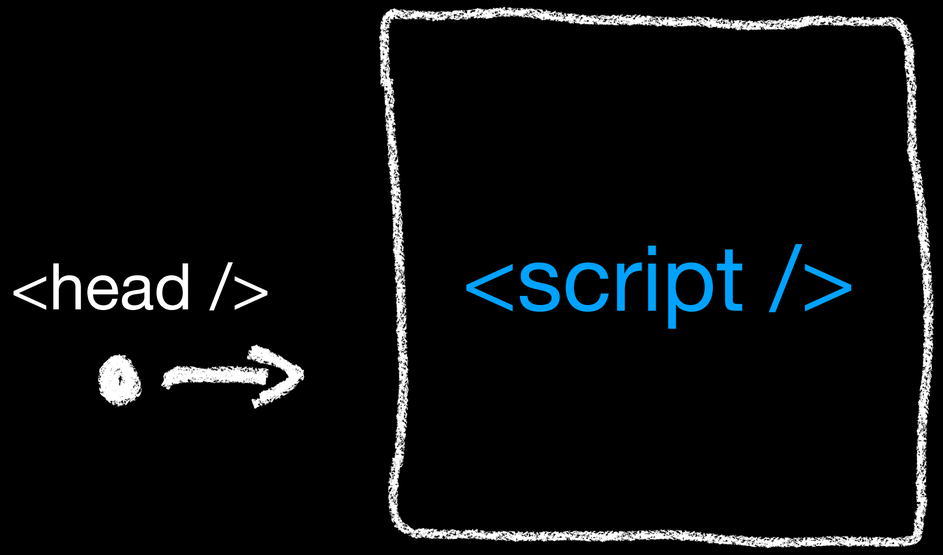
# Так люди писали в монолите

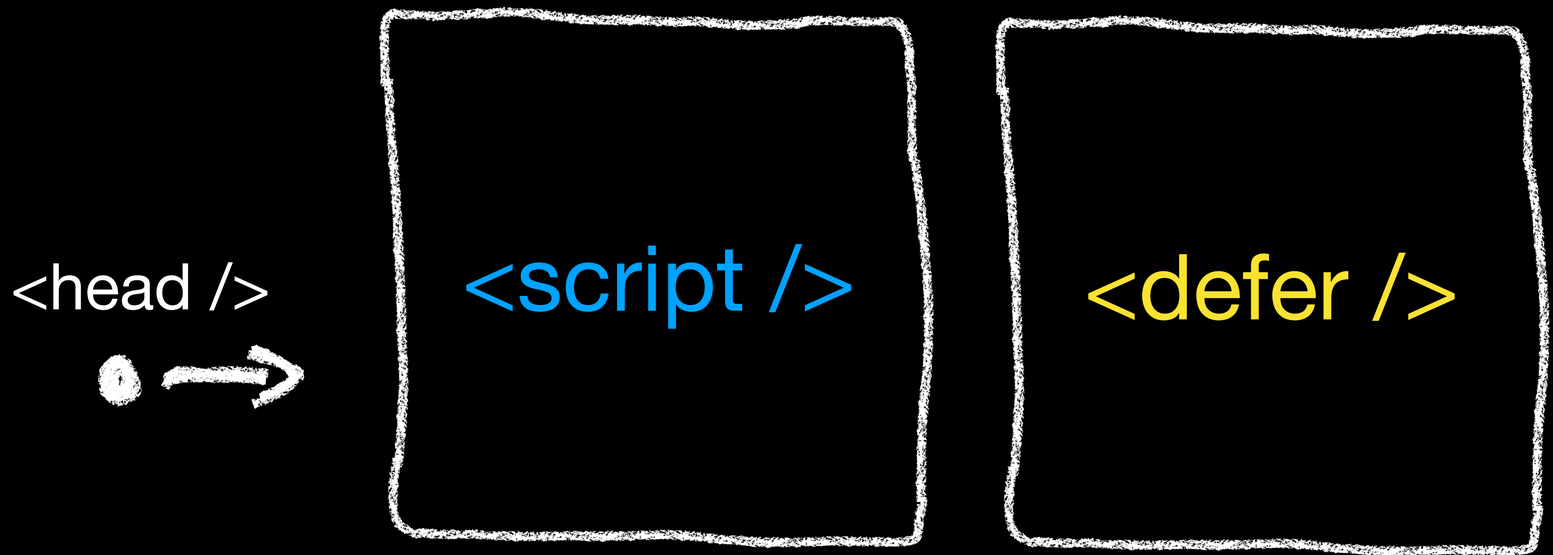
```
<html>
  <head>
      <meta/>
      <link/>
      <script></script>
      <title />
      {...}
  </head>
  <body>
    ...
  </body>
</html>
```

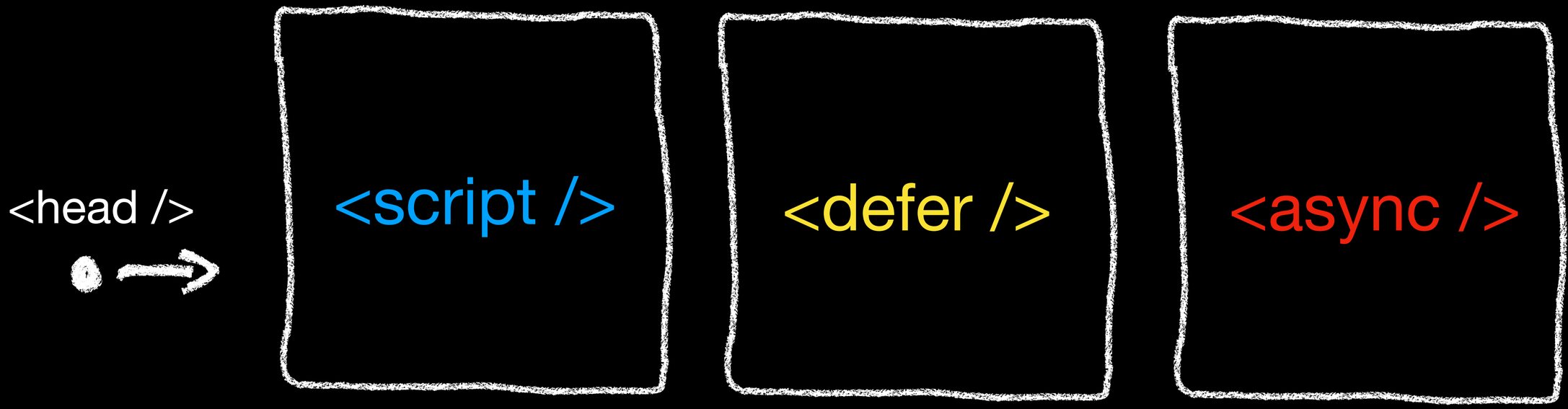
# где поставить разделение?

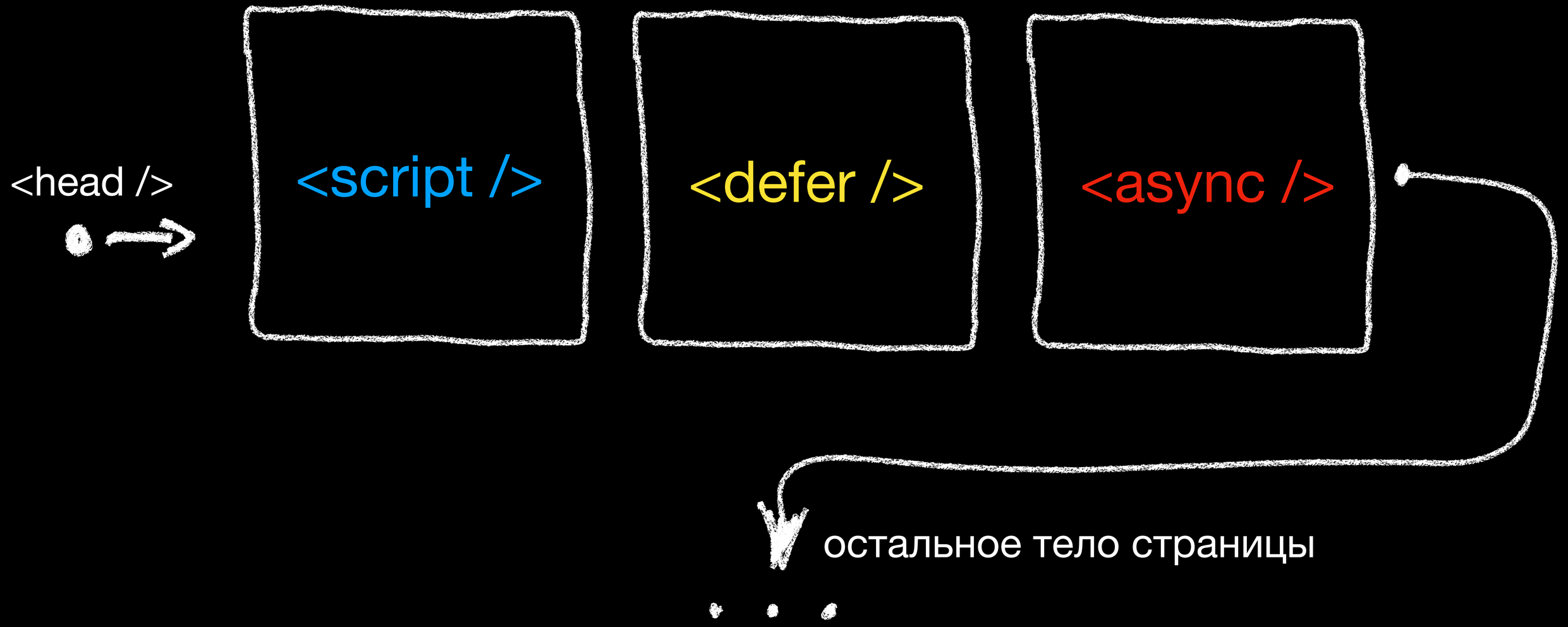
# начнем с общих скриптов

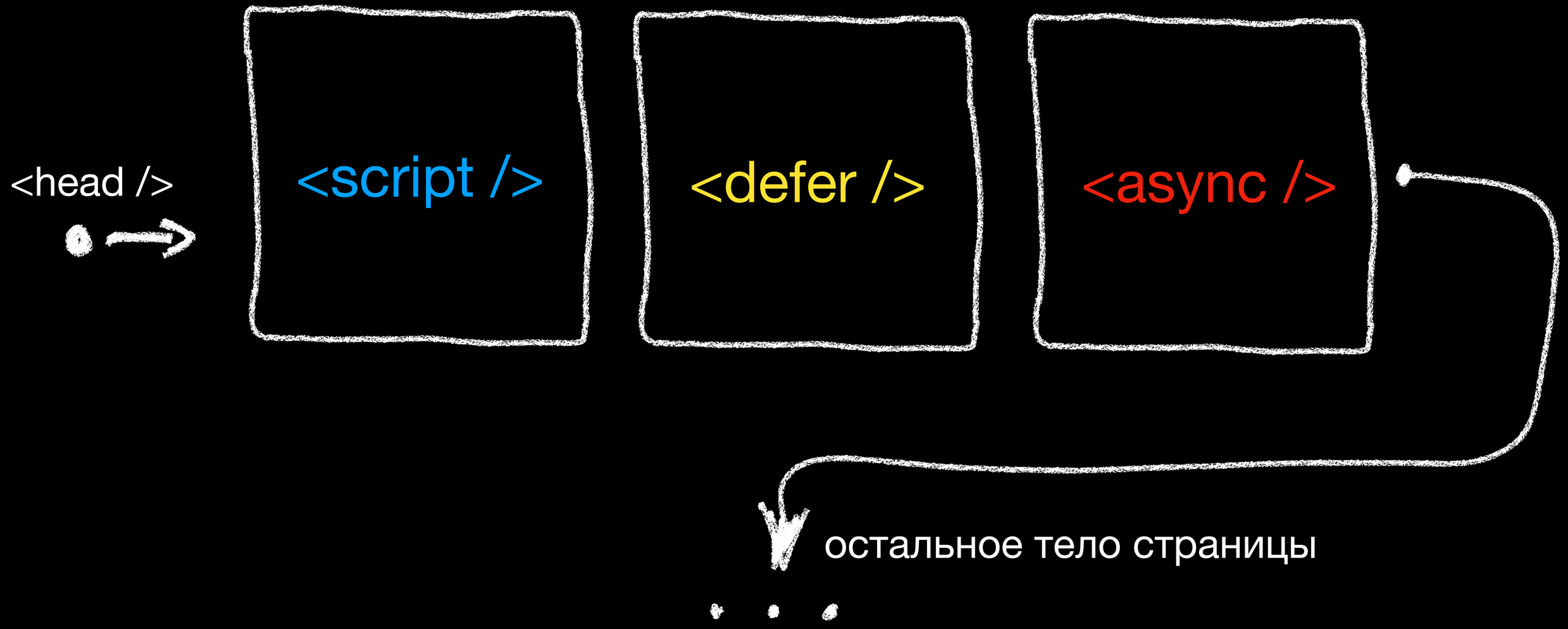
<head />

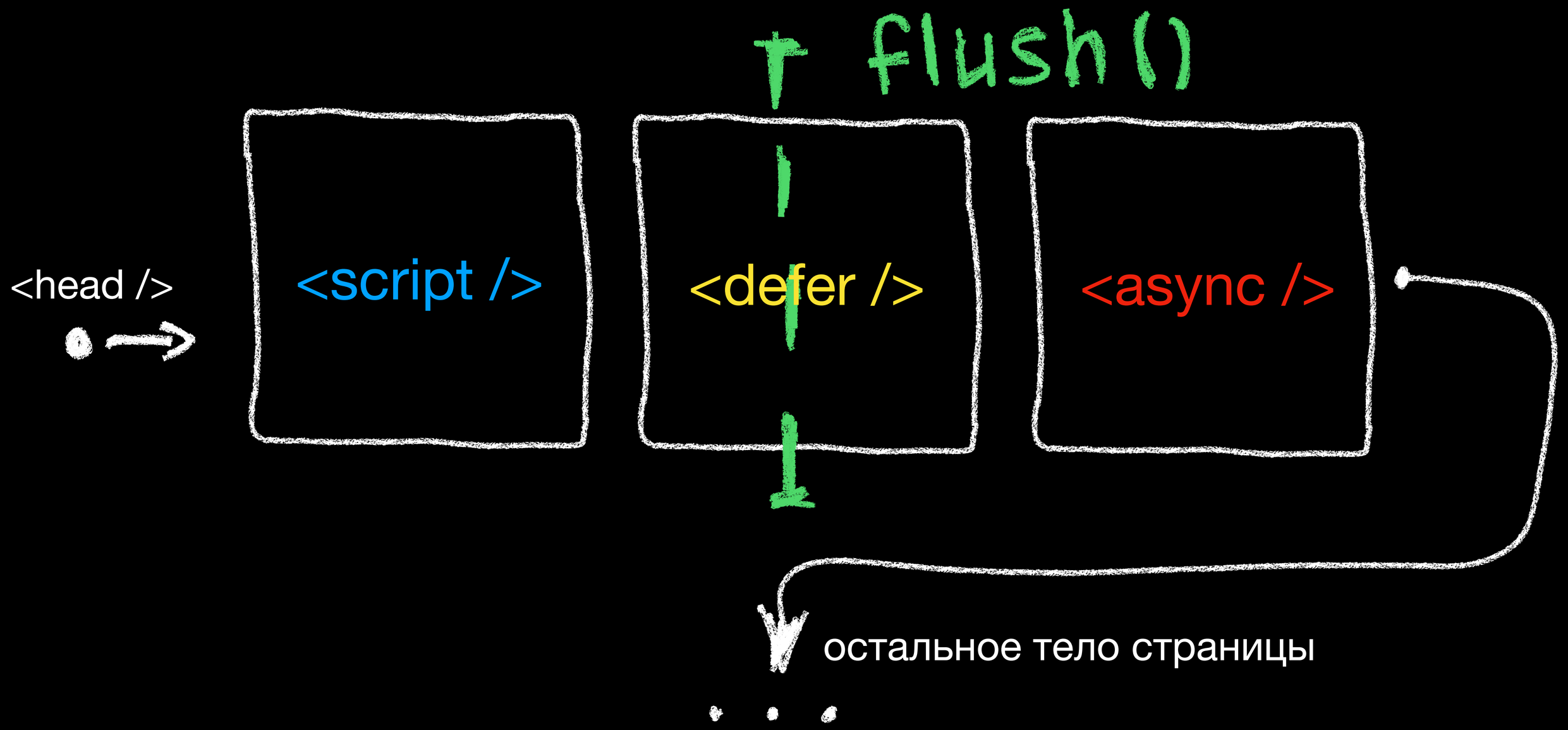

<head />
<script />


<head />
<script />
<defer />


<head />
<script />
<defer />
<async />


<head />
<script />
<defer />
<async />
остальное тело страницы


<head />
<script />
<defer />
<async />
остальное тело страницы
страница интерактивна


flush()
<head />
<script />
<defer />
<async />
остальное тело страницы
. . .
страница интерактивна

# Кто отвечает за скрипты

1) Webpack

# Кто отвечает за скрипты

1) Webpack
2) Static Manager

# Кто отвечает за скрипты

1) ~~Webpack~~ 

2) Static Manager

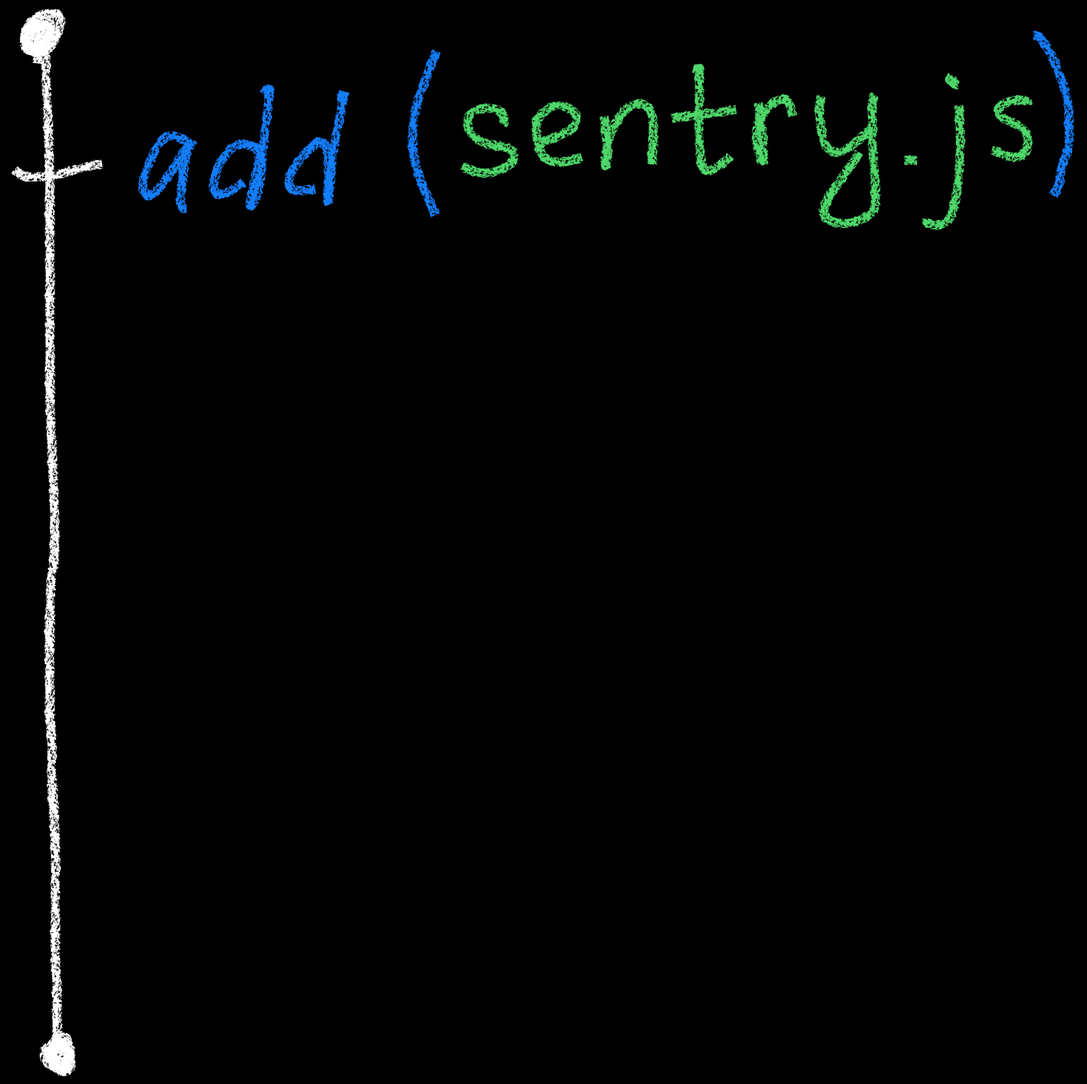
# Static Manager

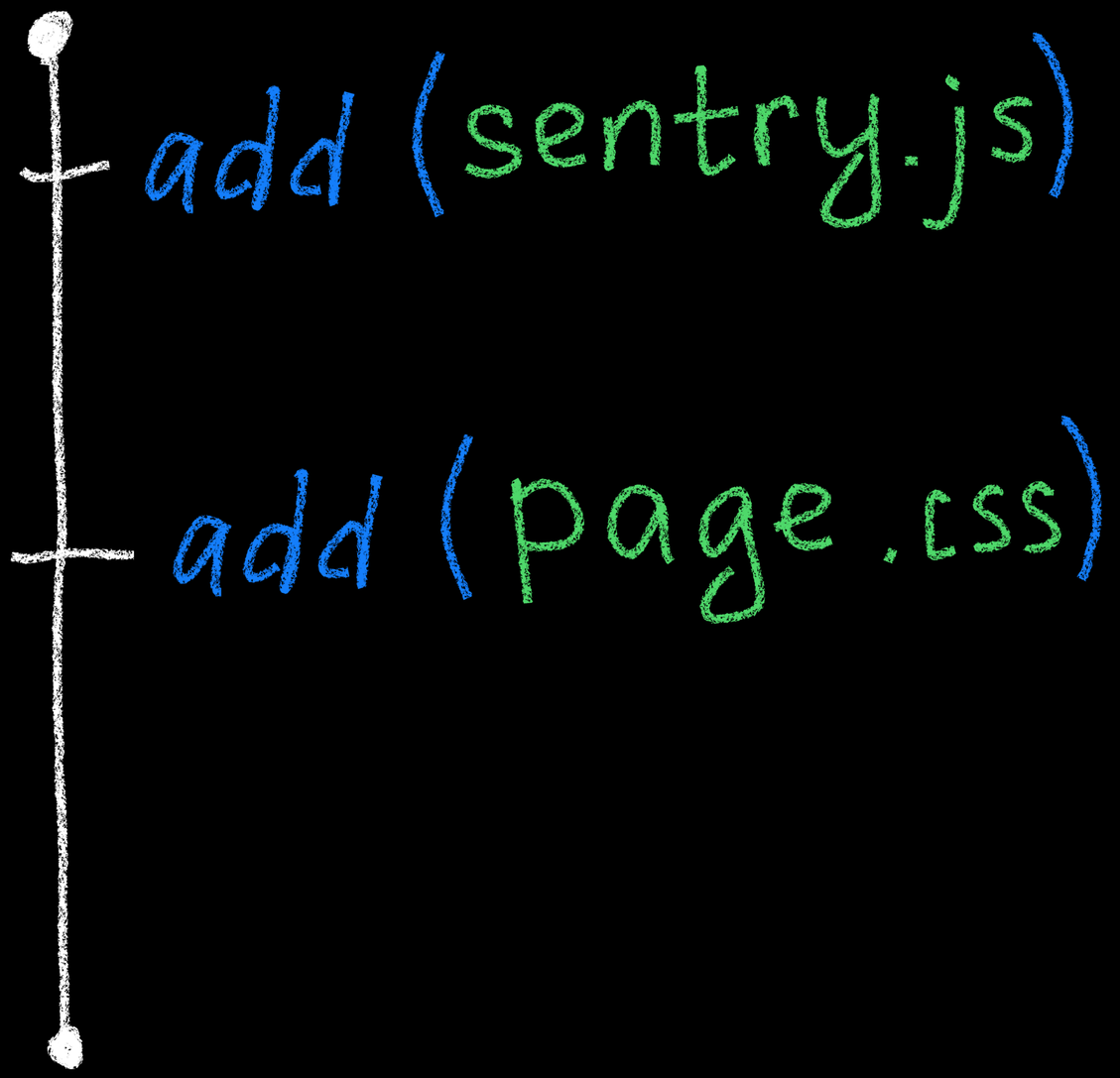
```
class StaticManager {

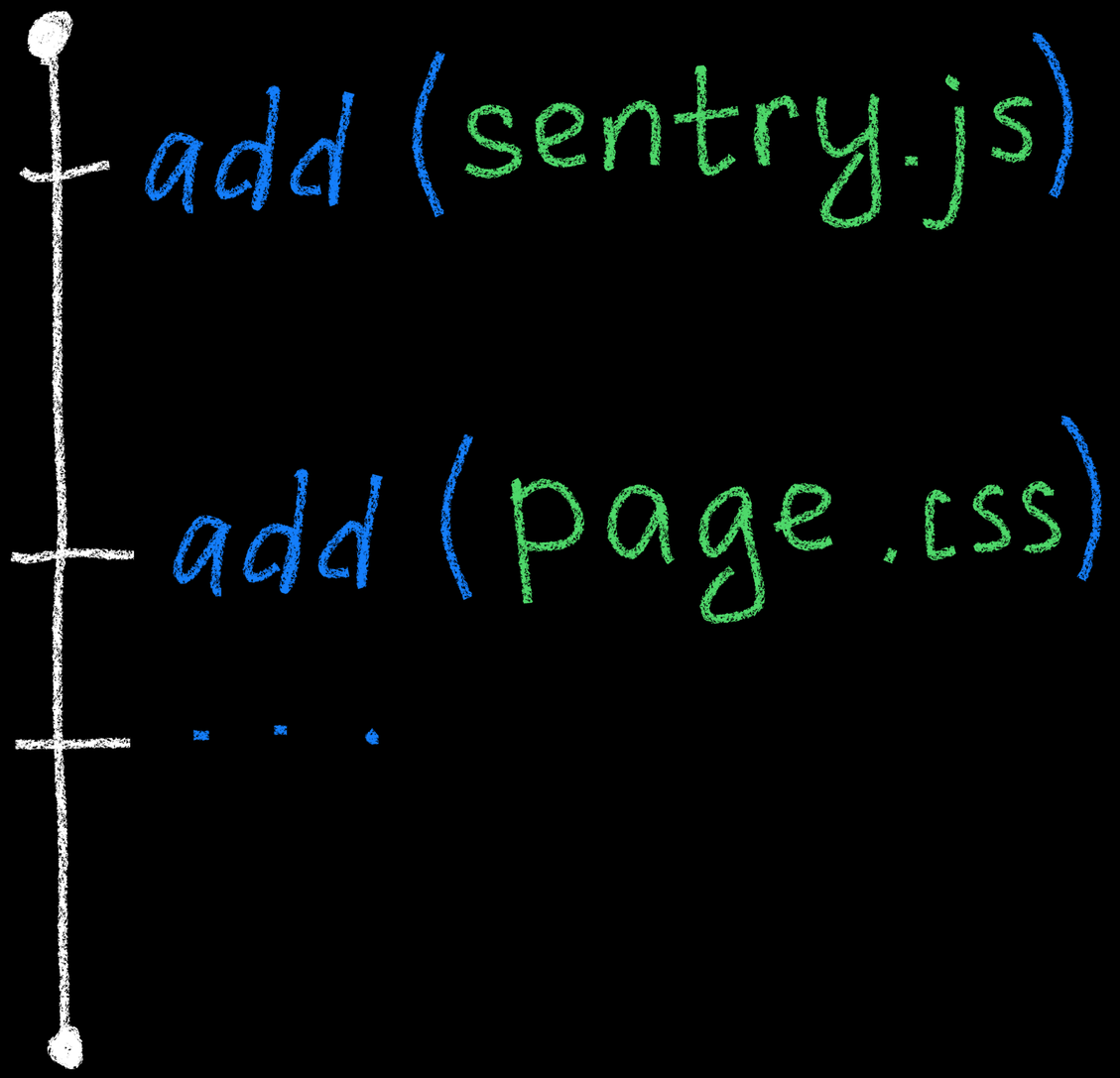
  public static function add (...$entry) {
    ...
  }

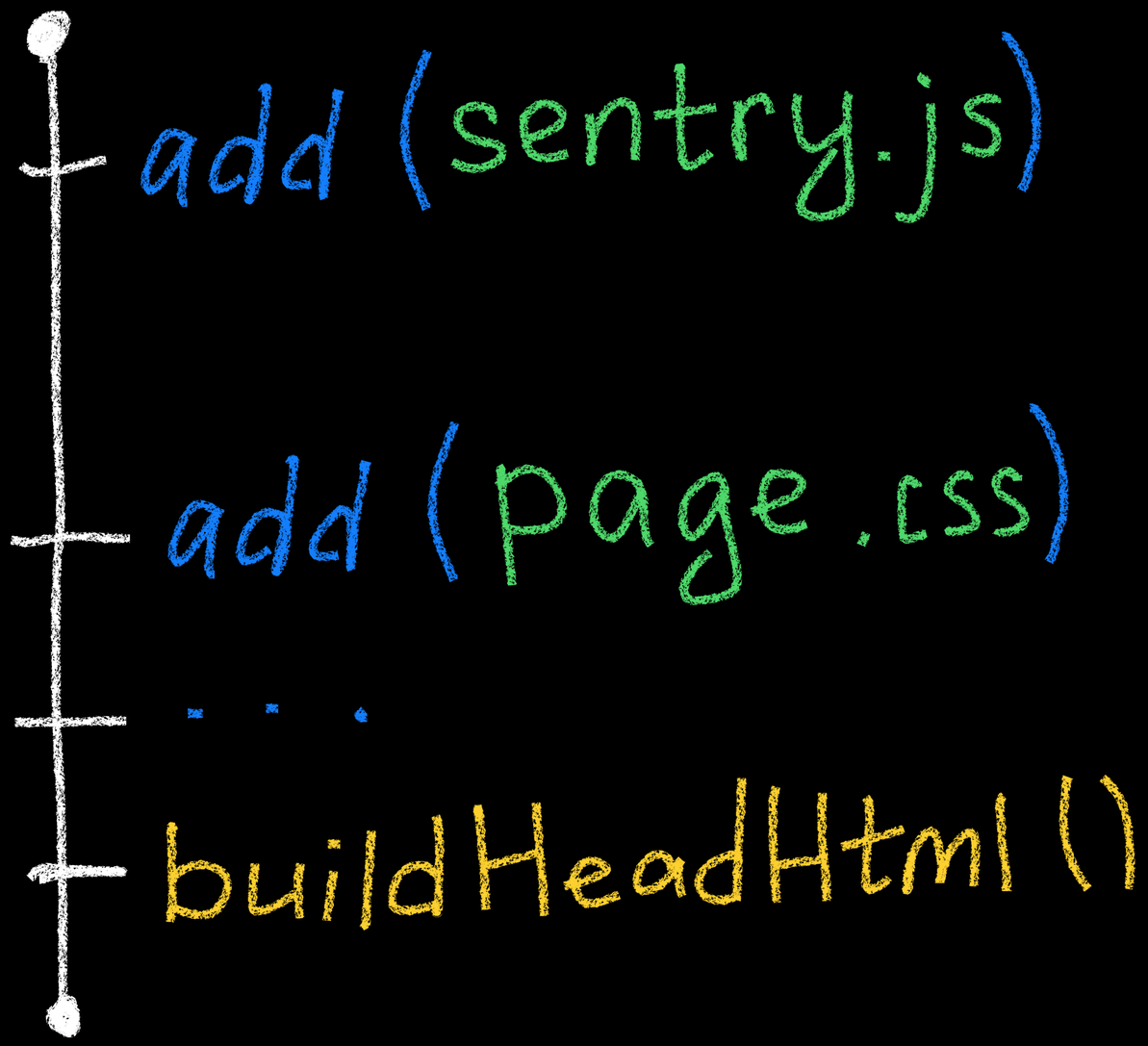
  public static function buildHeadHTML () {
    ...
  }

}
```


add (sentry.js)


add (sentry.js)
add (page.css)


add (sentry.js)
add (page.css)
. . .


add (sentry.js)
add (page.css)
. . .
buildHeadHtml ()

add (sentry.js)

add (page.css)

...

buildHeadHtml()

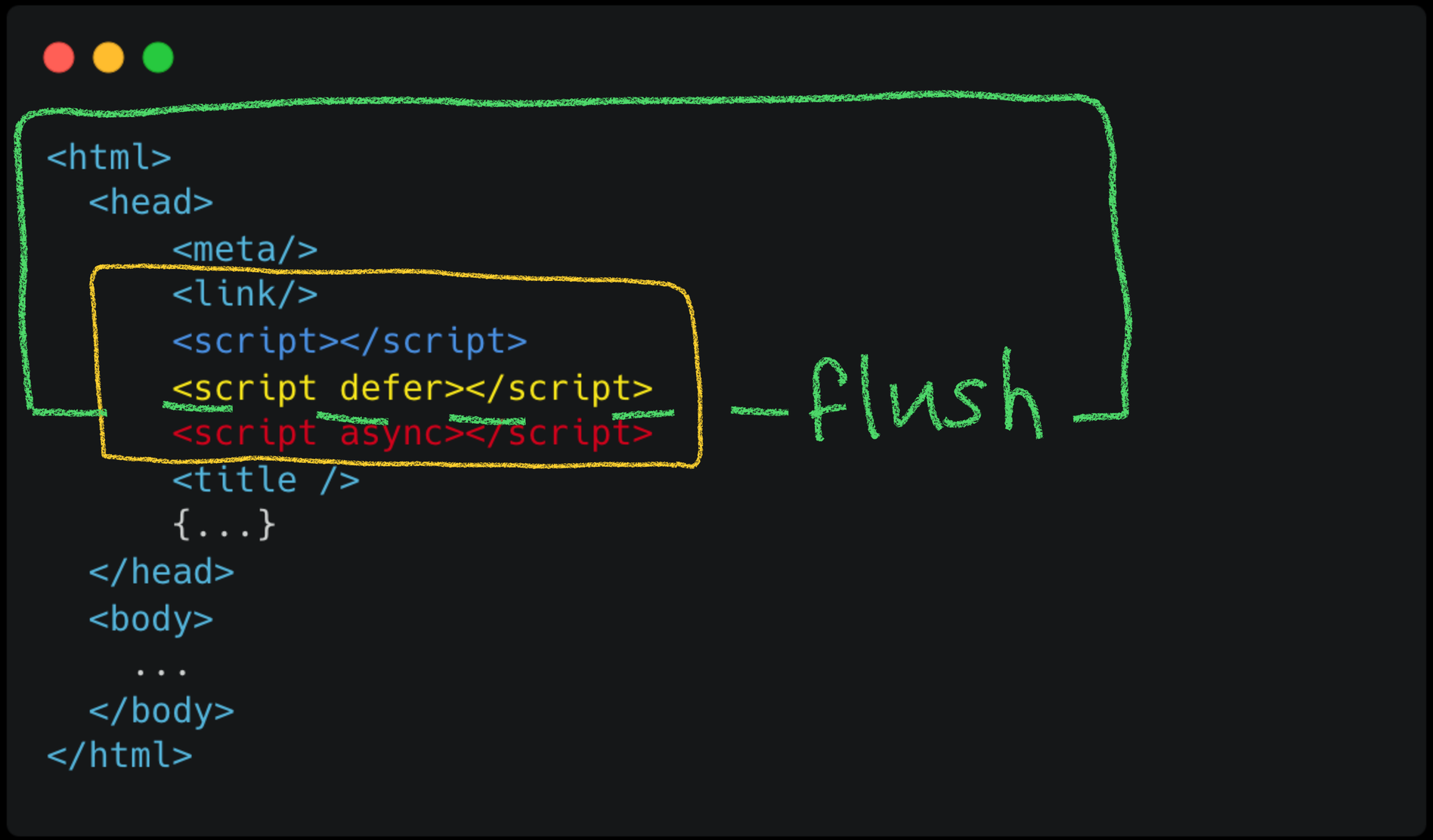
echo (


)

```
<html>
  <head>
      <meta/>
      <link/>
      <script></script>
      <script defer></script>
      <script async></script>
      <title />
      {...}
  </head>
  <body>
    ...
  </body>
</html>
```

```
<html>
  <head>
      <meta/>
      <link/>
      <script></script>
      <script defer></script>
      <script async></script>
      <title />
      {...}
  </head>
  <body>
    ...
  </body>
</html>
```

build Head Html


<html>
<head>
<meta/>
<link/>
<script></script>
<script defer></script>
<script async></script>
<title />
{...}
</head>
<body>
...
</body>
</html>
flush

add (sentry.js)

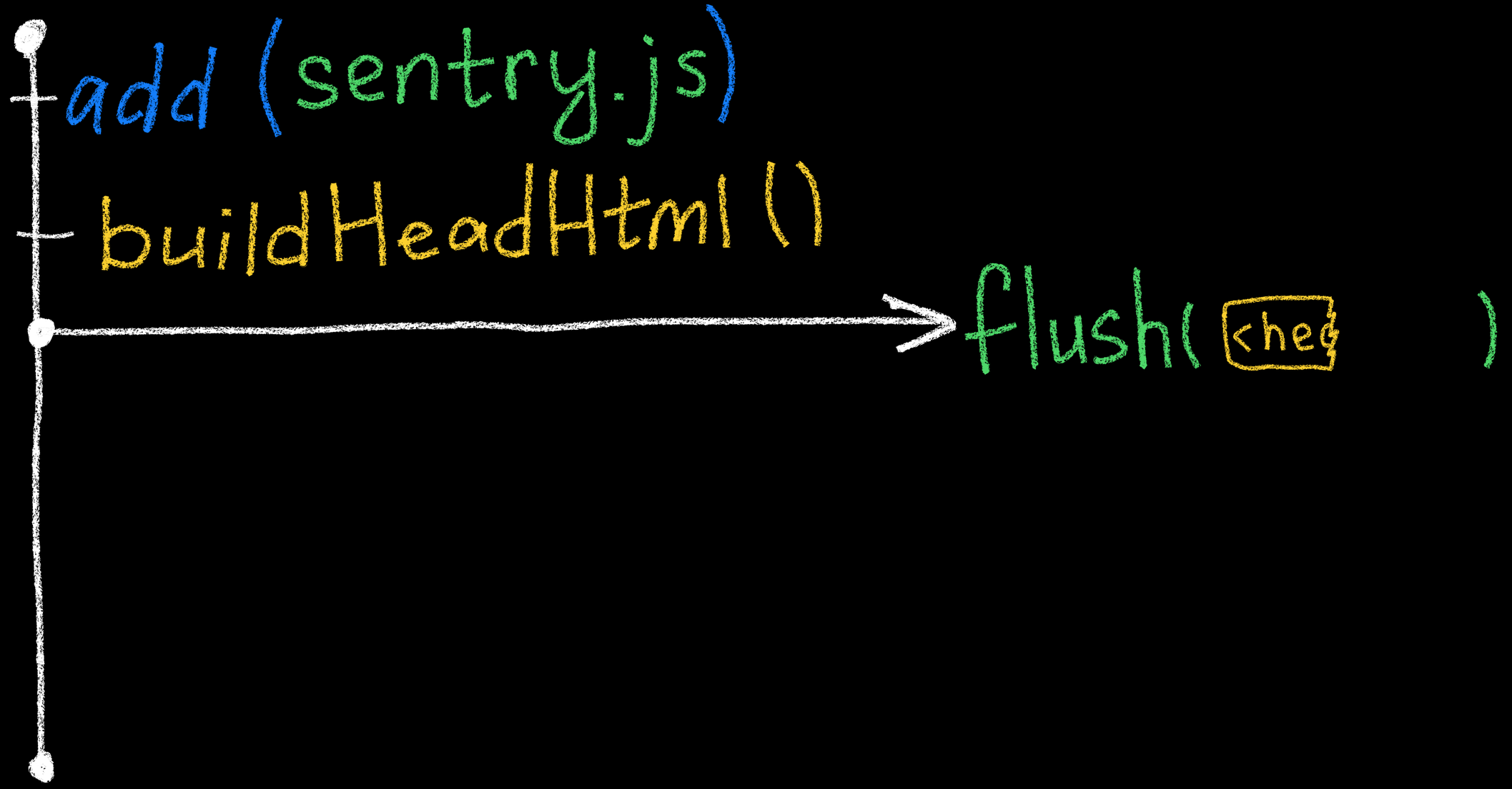
add (page.css)

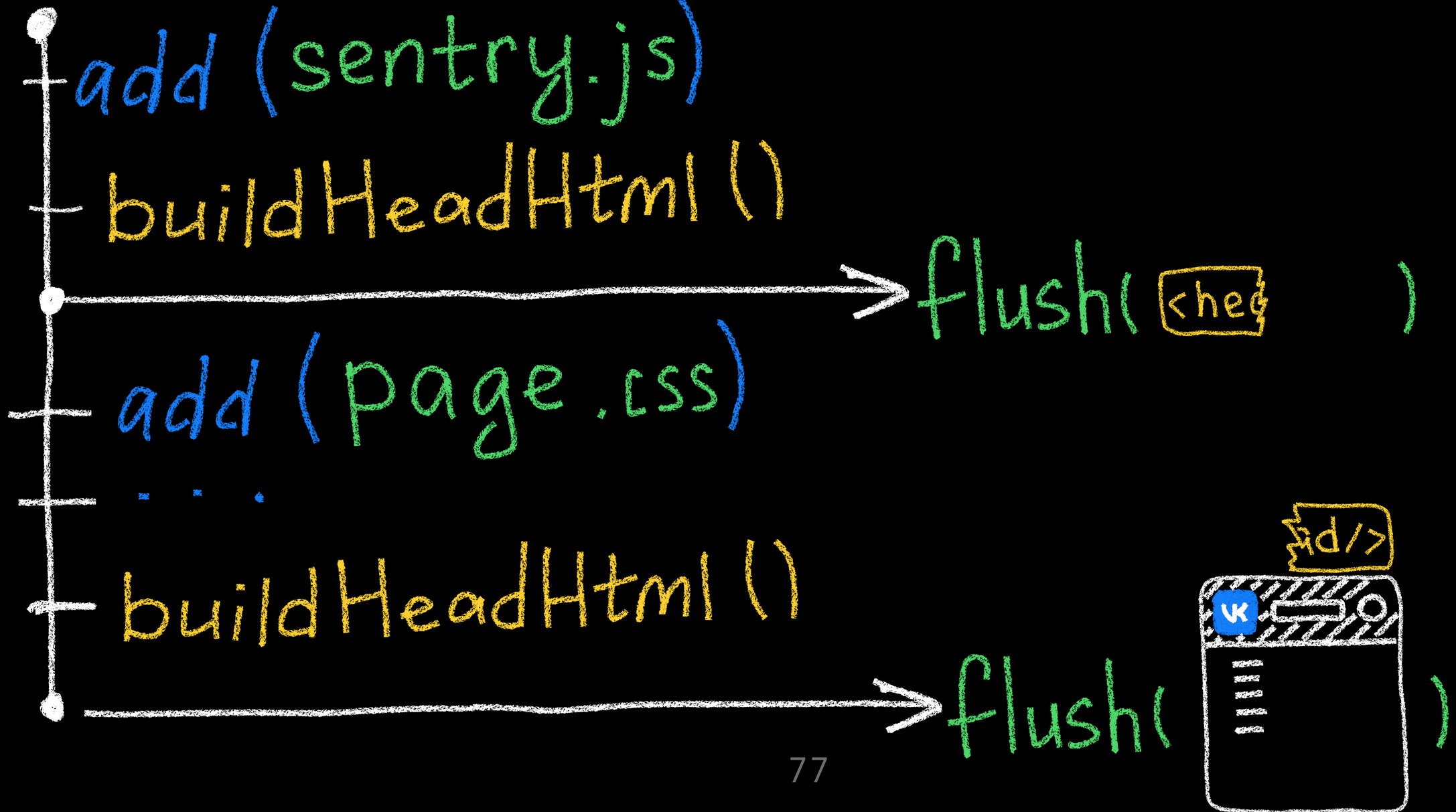
...

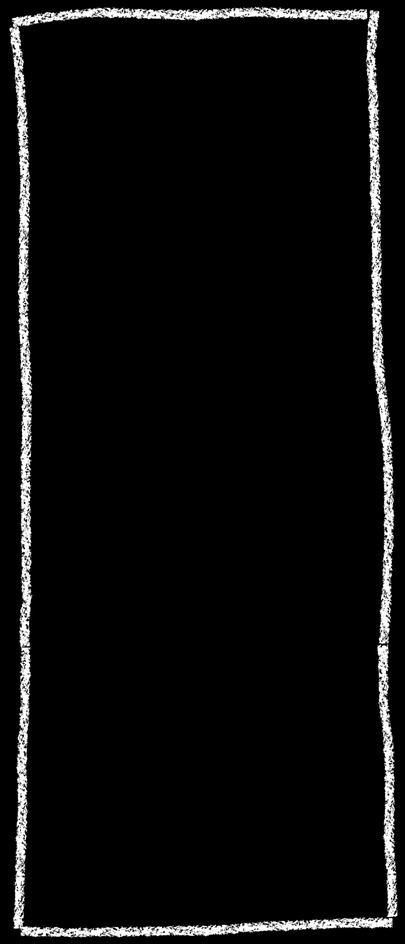
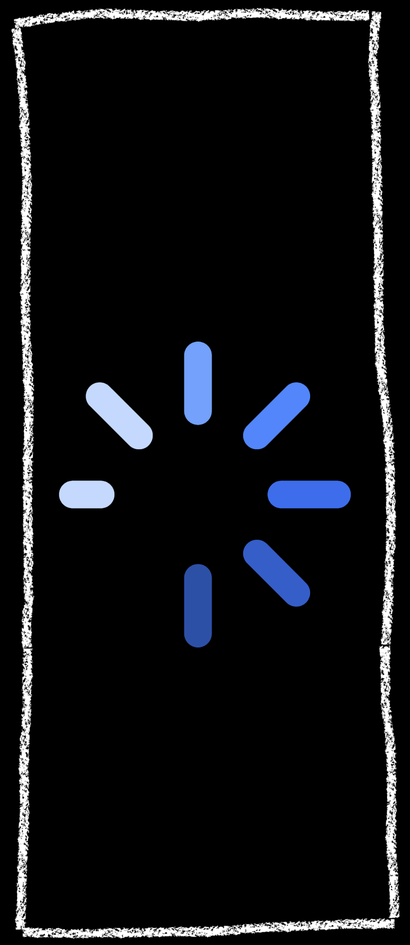
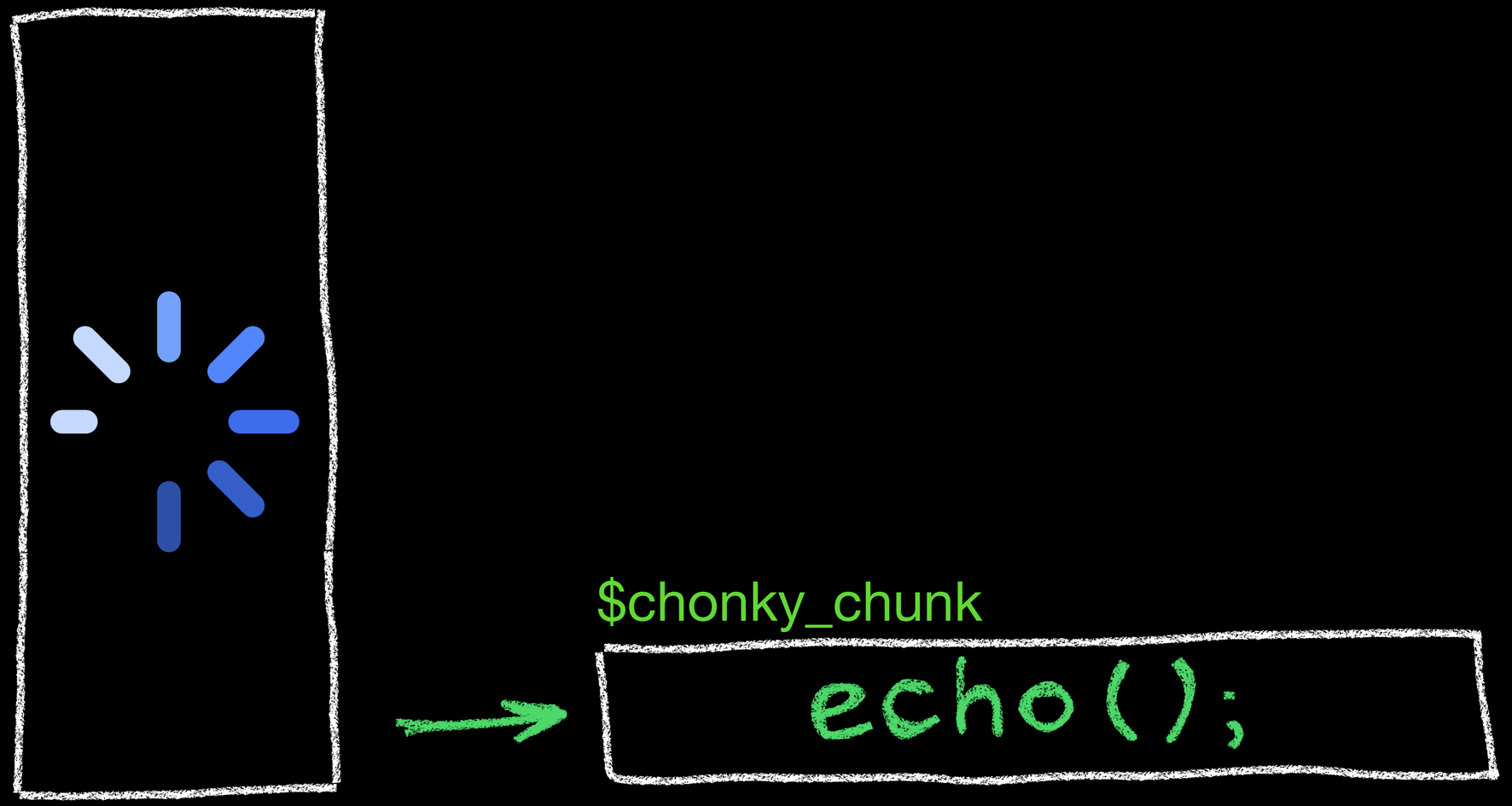
buildHeadHtml()

echo (


)

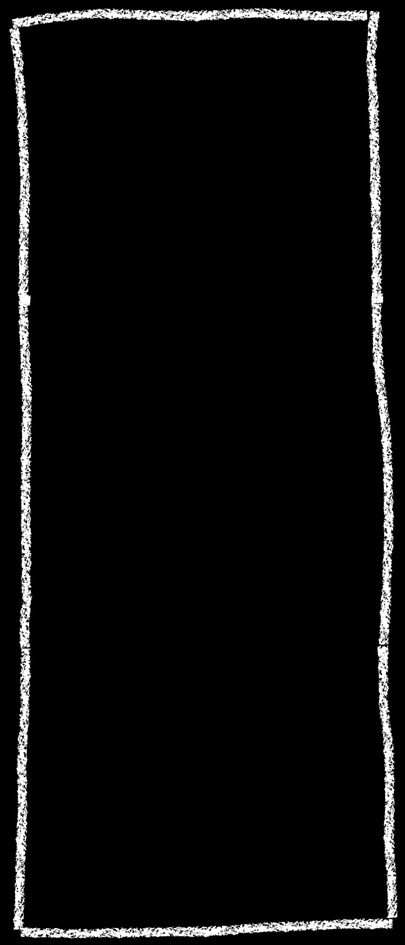

add (sentry.js)
buildHeadHtml ()
flush(
<he


add (sentry.js)
buildHeadHtml ()
flush( <he
add (page.css)
. . .
buildHeadHtml ()
flush(
d/>

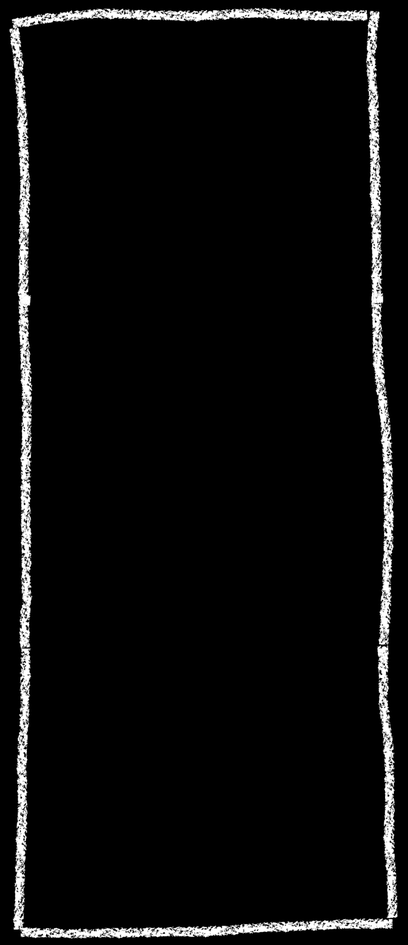
```
$chonky_chunk = ```HTML<html>
  <head>
      <meta/>
      <link/>
      <script></script>
      <script defer></script>
      <script async></script>
      <title />
      {...}
  </head>
  <body>
    ...
  </body>
</html>
```;
```
```






$chonky_chunk
echo();

```
$first_chunk = ```<html>
  <head>
      <meta/>
      <link/>
      <script></script>
      <script defer></script>
      <script async></script>
 ```;
```

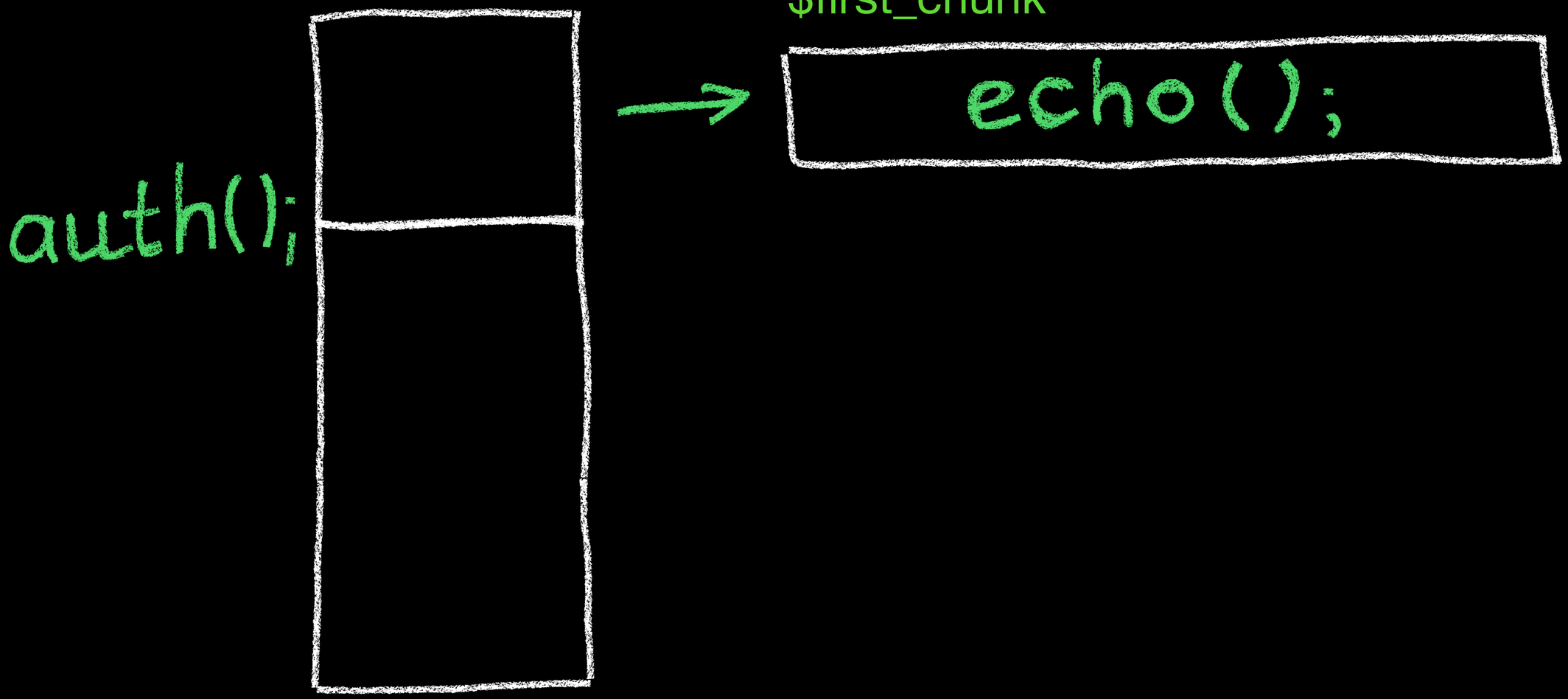
```
$rest_chunk = ```
      <title />
      {...}
  </head>
  <body>
    ...
  </body>
</html>
```;
```

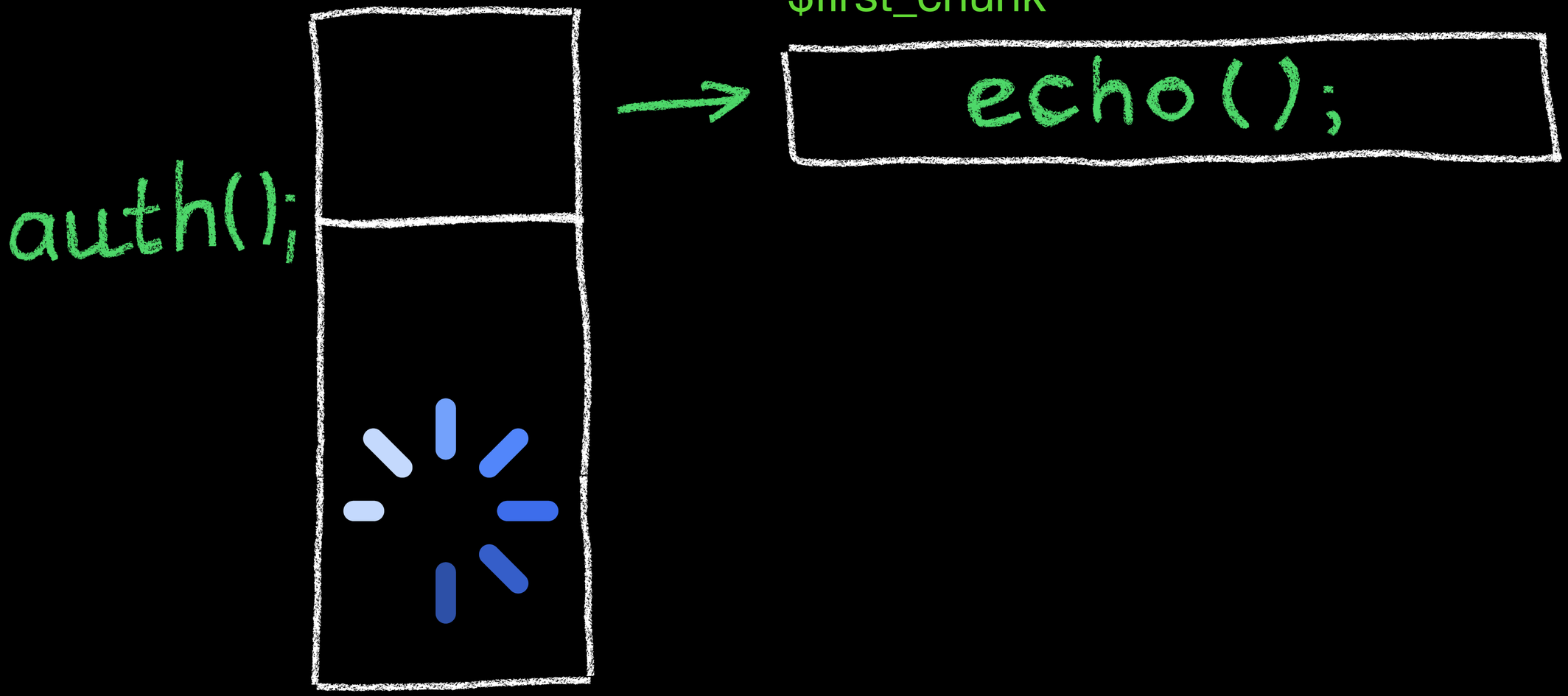
```
function doAuth() {

  ...

  if (FlushChunks::isFlushAvailable()) {
     FlushChunks::earlyFlush($first_chunk);
  }

  ...

}
```

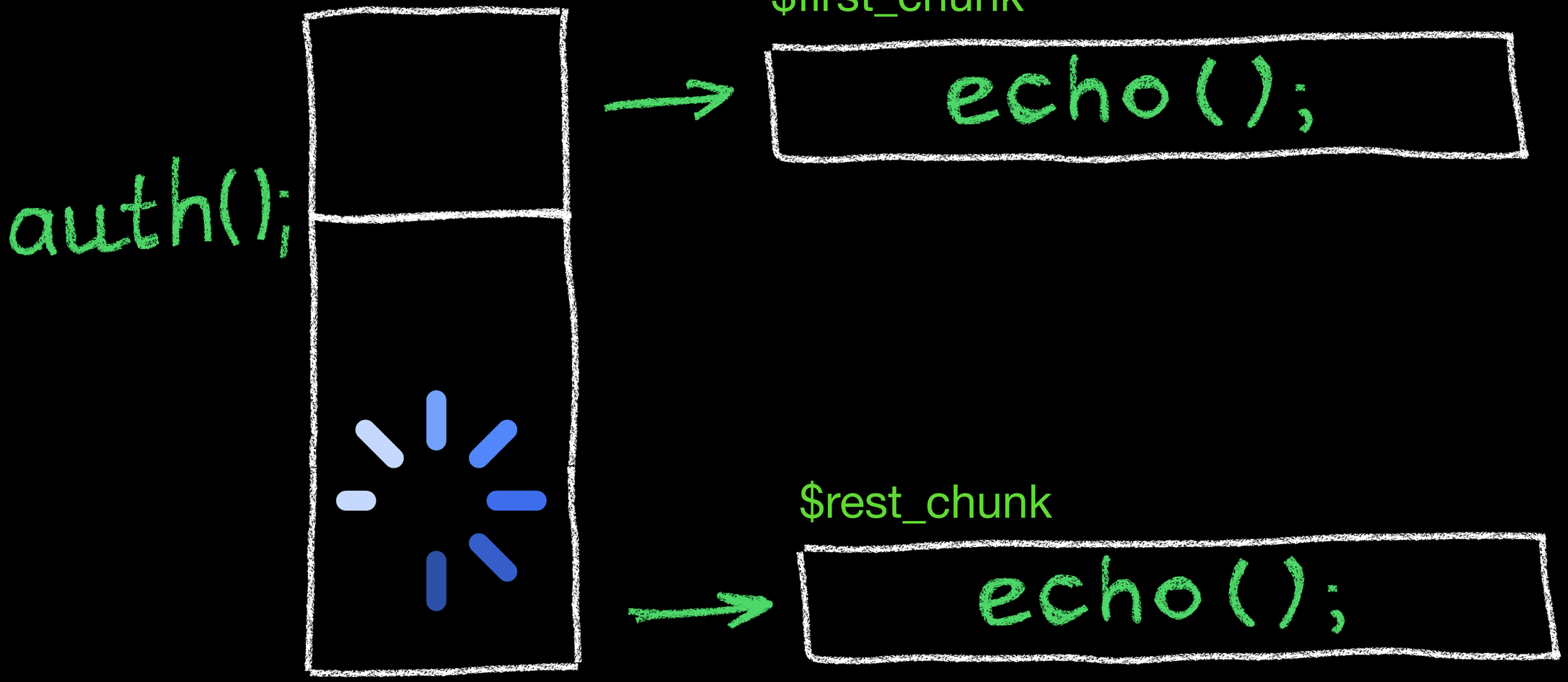


auth();



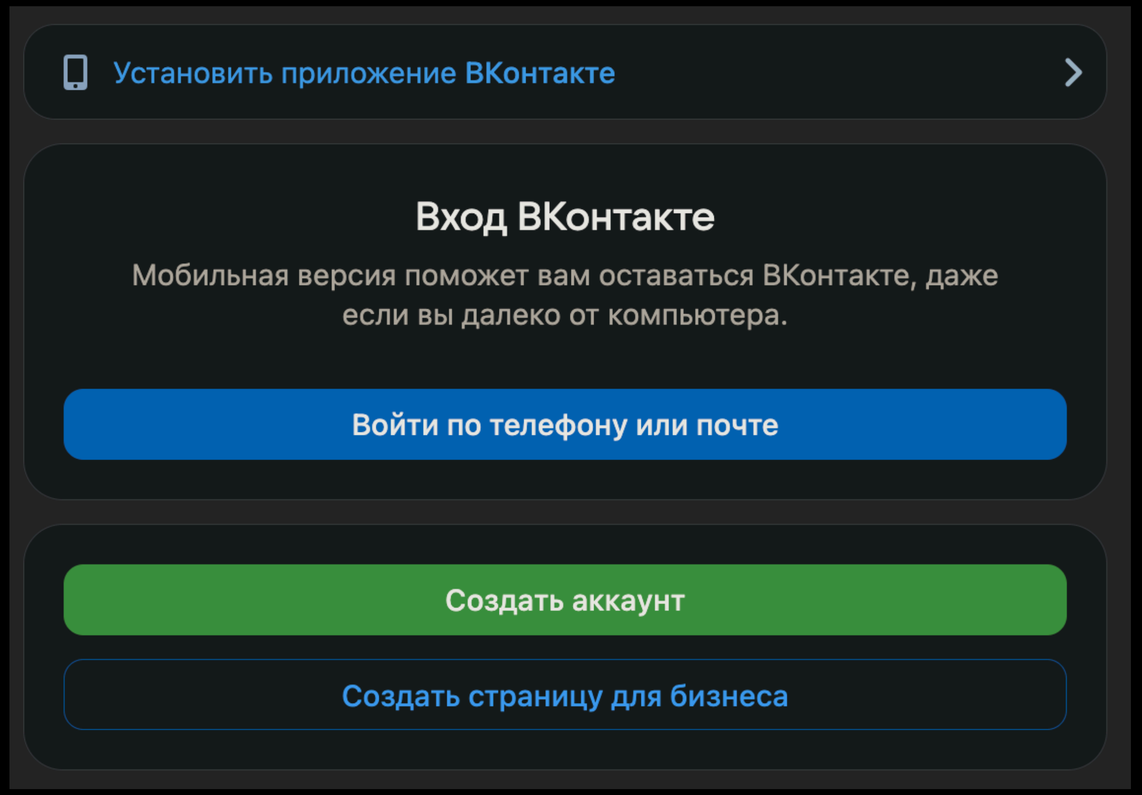

$first_chunk
echo();
auth();


$first_chunk
echo();
auth();

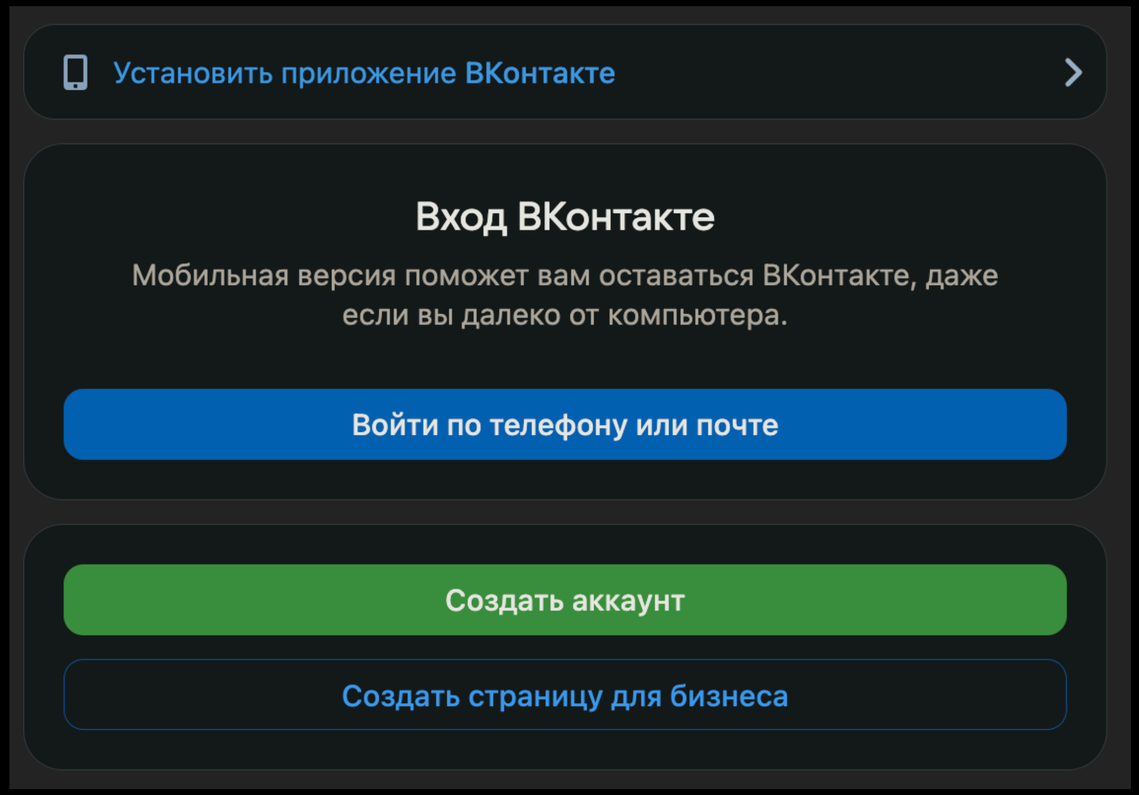

$first_chunk
auth();
echo();
$rest_chunk
echo();

# тестируем на себе

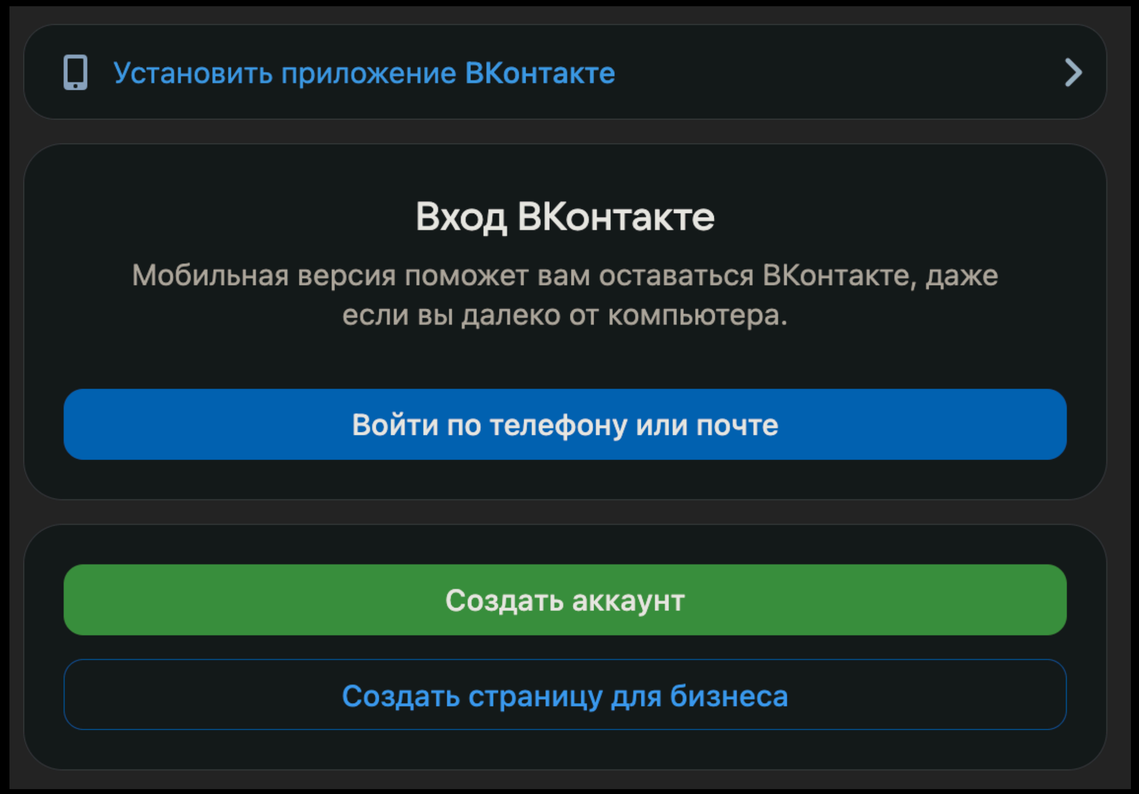
# Логинимся


Установить приложение ВКонтакте
Вход ВКонтакте
Мобильная версия поможет вам оставаться ВКонтакте, даже если вы далеко от компьютера.
Войти по телефону или почте
Создать аккаунт
Создать страницу для бизнеса

# Логинимся


Установить приложение ВКонтакте
Вход ВКонтакте
Мобильная версия поможет вам оставаться ВКонтакте, даже если вы далеко от компьютера.
Войти по телефону или почте
Создать аккаунт
Создать страницу для бизнеса

# Логинимся


Установить приложение ВКонтакте
Вход ВКонтакте
Мобильная версия поможет вам оставаться ВКонтакте, даже если вы далеко от компьютера.
Войти по телефону или почте
Создать аккаунт
Создать страницу для бизнеса

# Логинимся

```
Manifest: Line: 1, column: 1, Syntax error.
GET https://tk-deflux.m.cs7777.vk.com/manifest.webmanifest?ver=228 net::ERR_HTTP2_PROTOCOL_ERROR 200 (OK)
▶ GET https://mincifry-cert.vk.com/ net::ERR_CERT_AUTHORITY_INVALID
Manifest: Line: 1, column: 1, Syntax error.
▶ GET https://tk-deflux.m.cs7777.vk.com/
>
```

```
Manifest: Line: 1, column: 1, Syntax error.
GET https://tk-deflux.m.cs7777.vk.com/manifest.webmanifest?ver=228 net::ERR_HTTP2_PROTOCOL_ERROR 200 (OK)
▶ GET https://mincifry-cert.vk.com/ net::ERR_CERT_AUTHORITY_INVALID
Manifest: Line: 1, column: 1, Syntax error.
▶ GET https://tk-deflux.m.cs7777.vk.com/manifest.webmanifest?ver=228 net::ERR_HTTP2_PROTOCOL_ERROR 200 (OK)
>
```

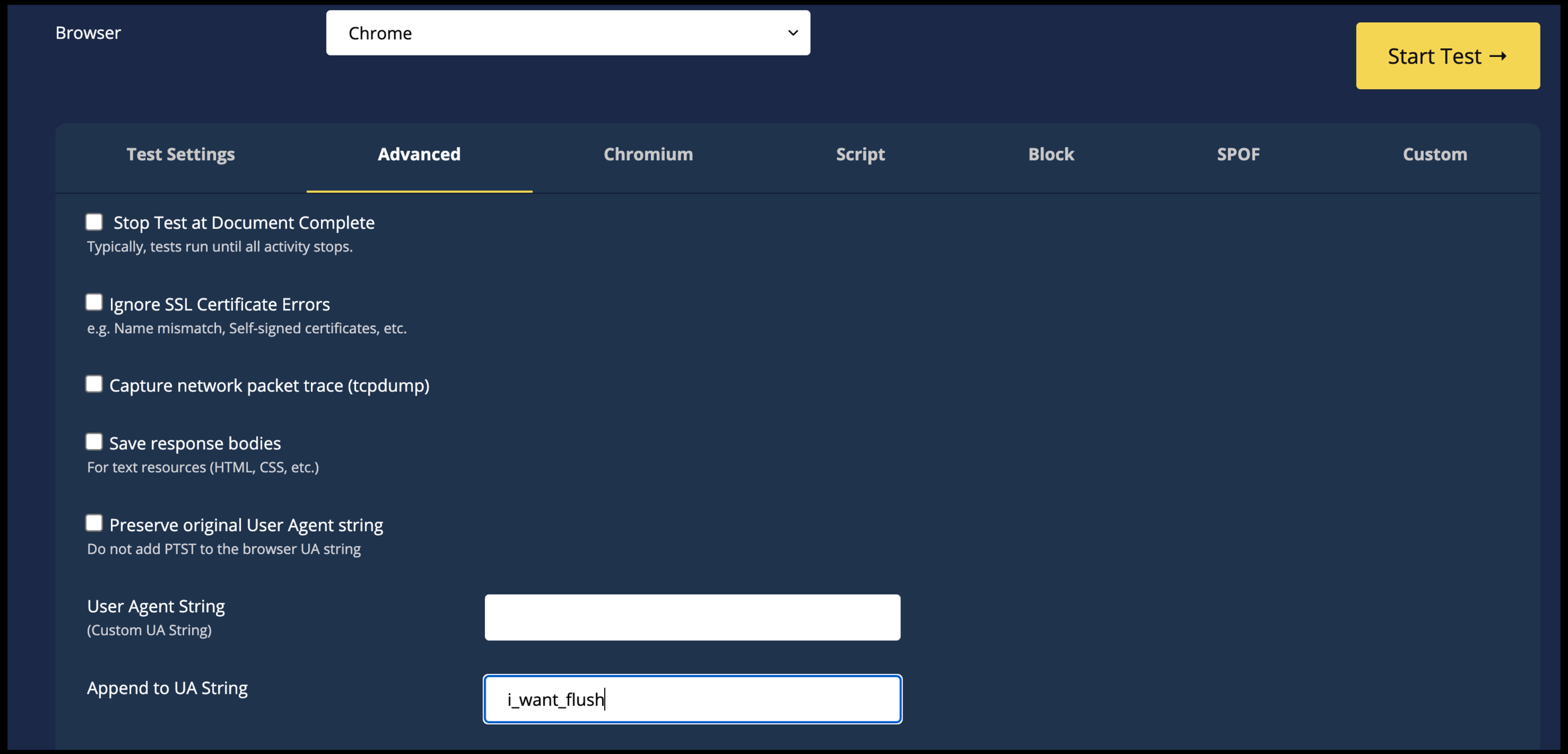
```
hthouse  Recorder  Performance insights                                  1
                                                          Default levels ▼  N
4723.b04f0…pGRTJ6UXg0OXZkRk15TkVENGU1aTBVYjVVcEtfT3ZDNE9UNThOZ05oT0VmV1dXOWwza0Njdg-- net::ERR_CONTENT_DECODING_FAILED 200 (OK)  /login?slogin_h=d544…mV1dXOWwza0Nj
```

по-быстрому
не получилось

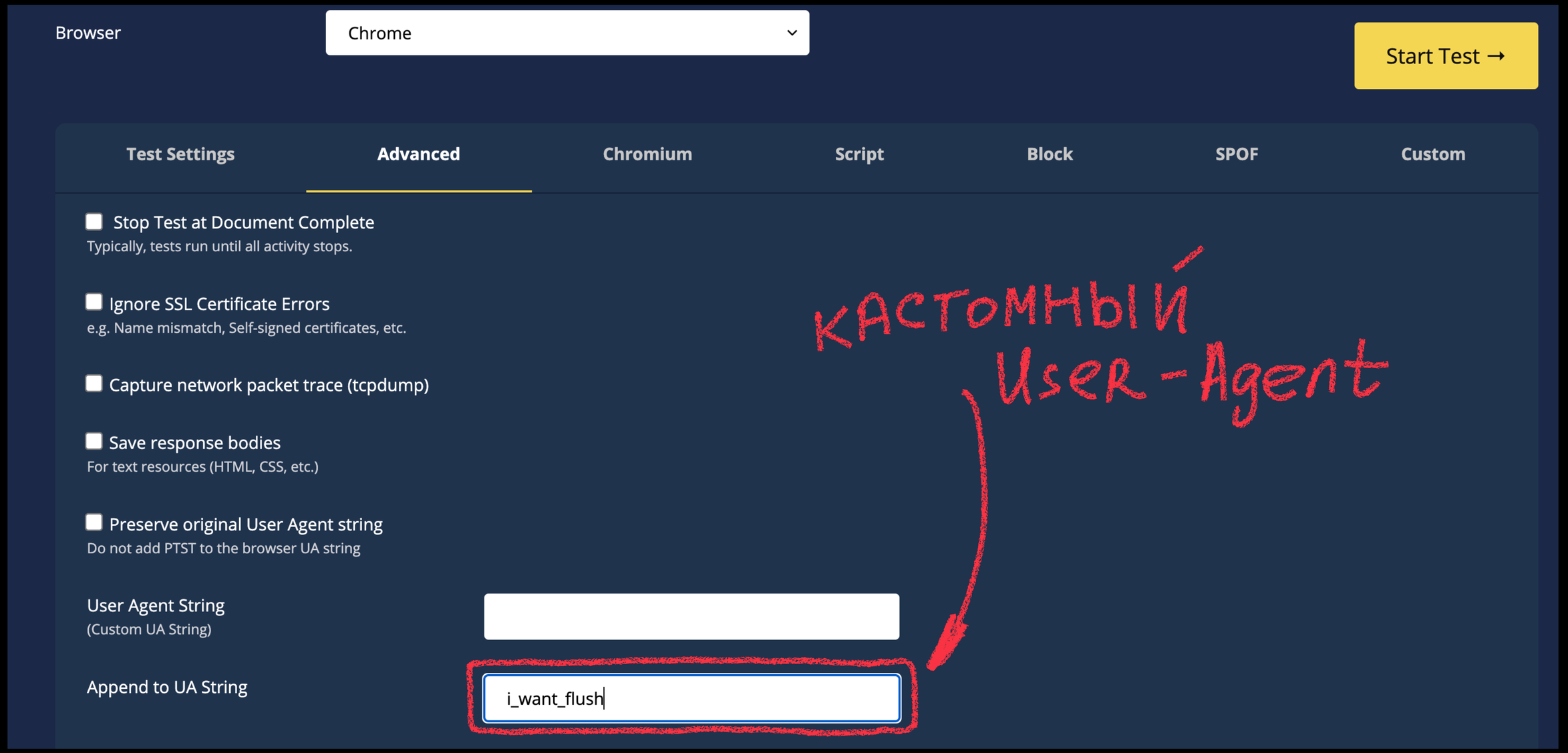
# тестим на незарегах

# webpagetest – грубый замер

# webpagetest



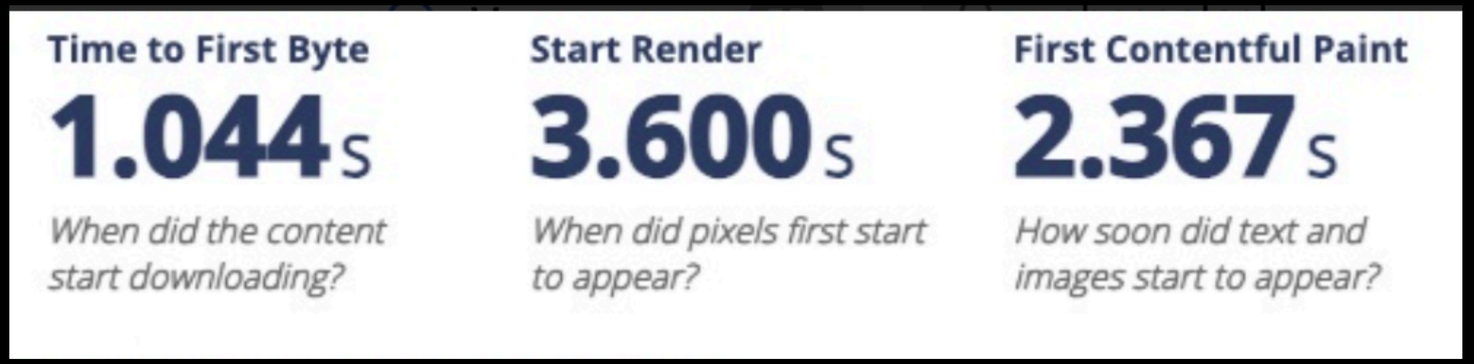
# webpagetest



# webpagetest

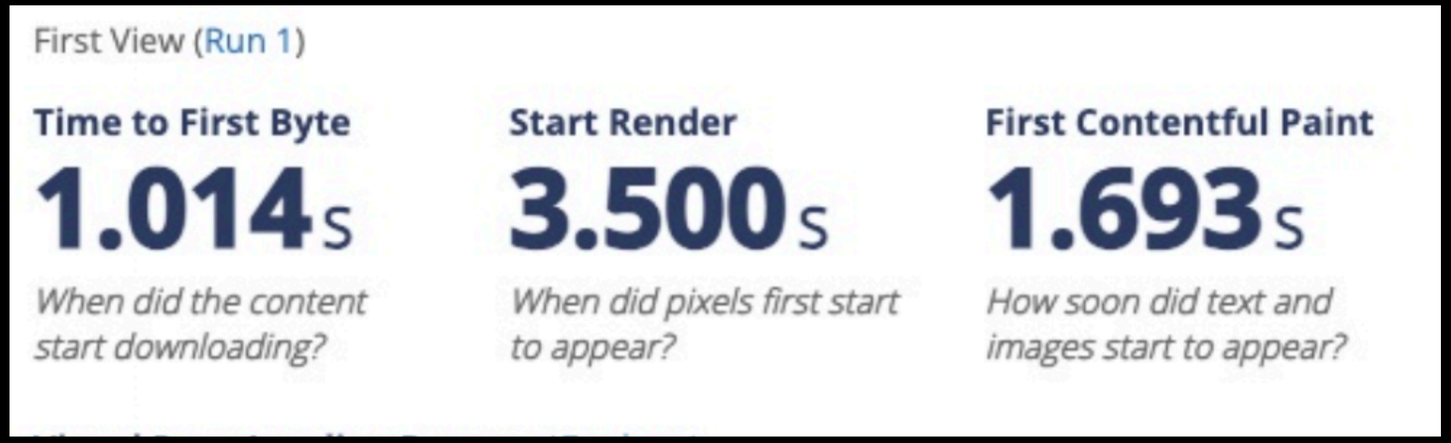

Без флаша: 2.367s

# webpagetest

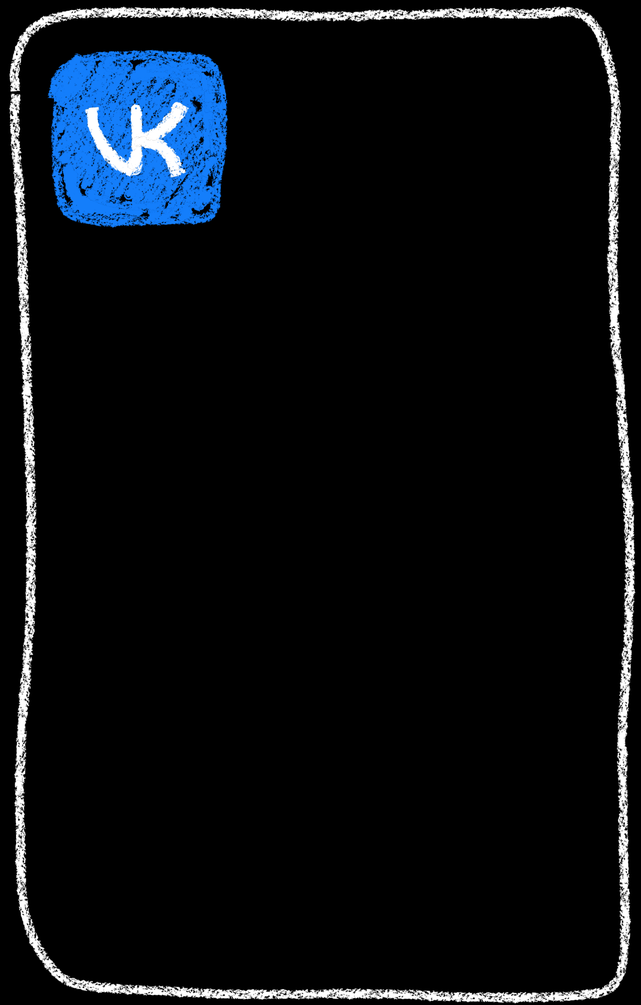


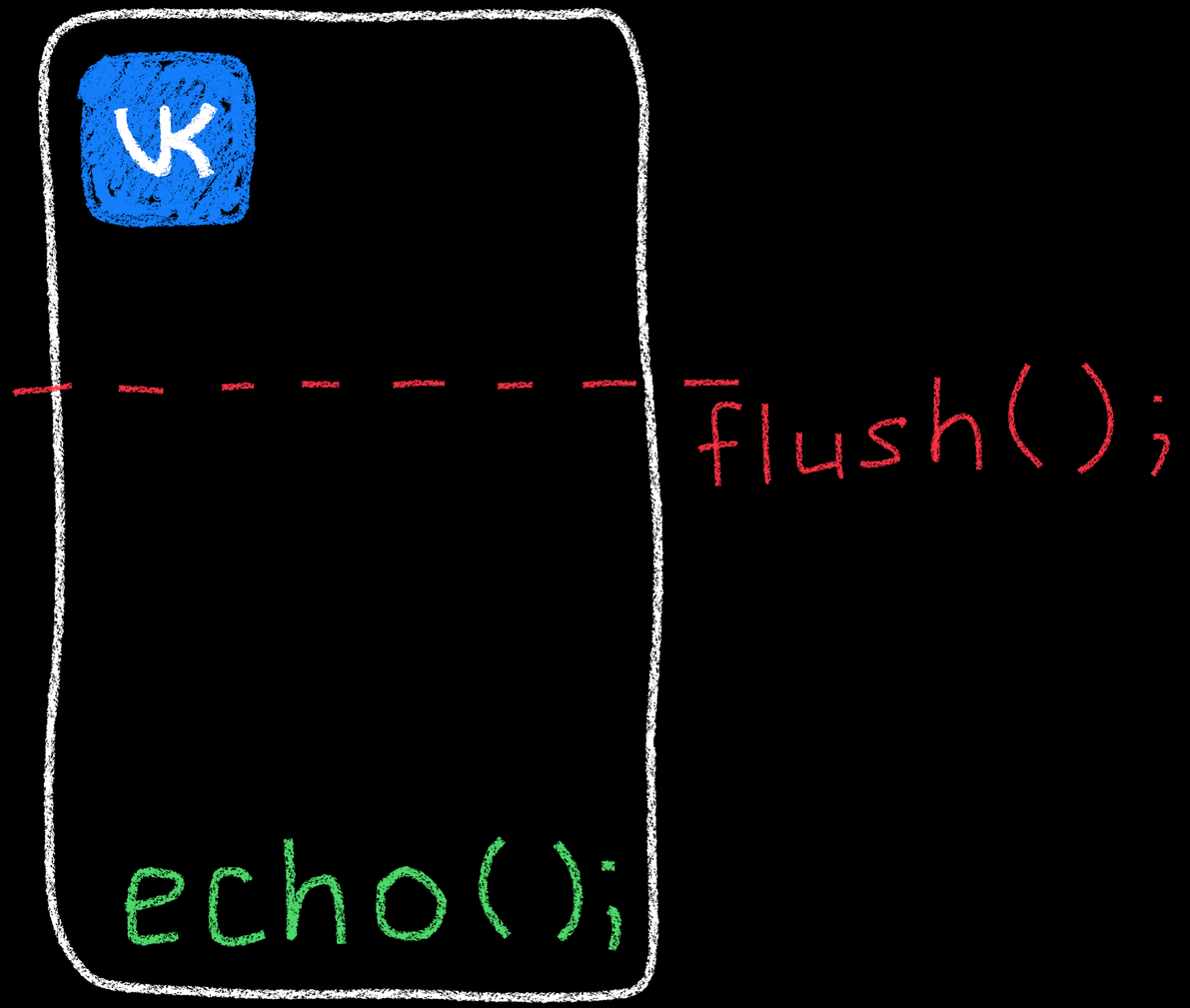

Без флаша: 2.367s

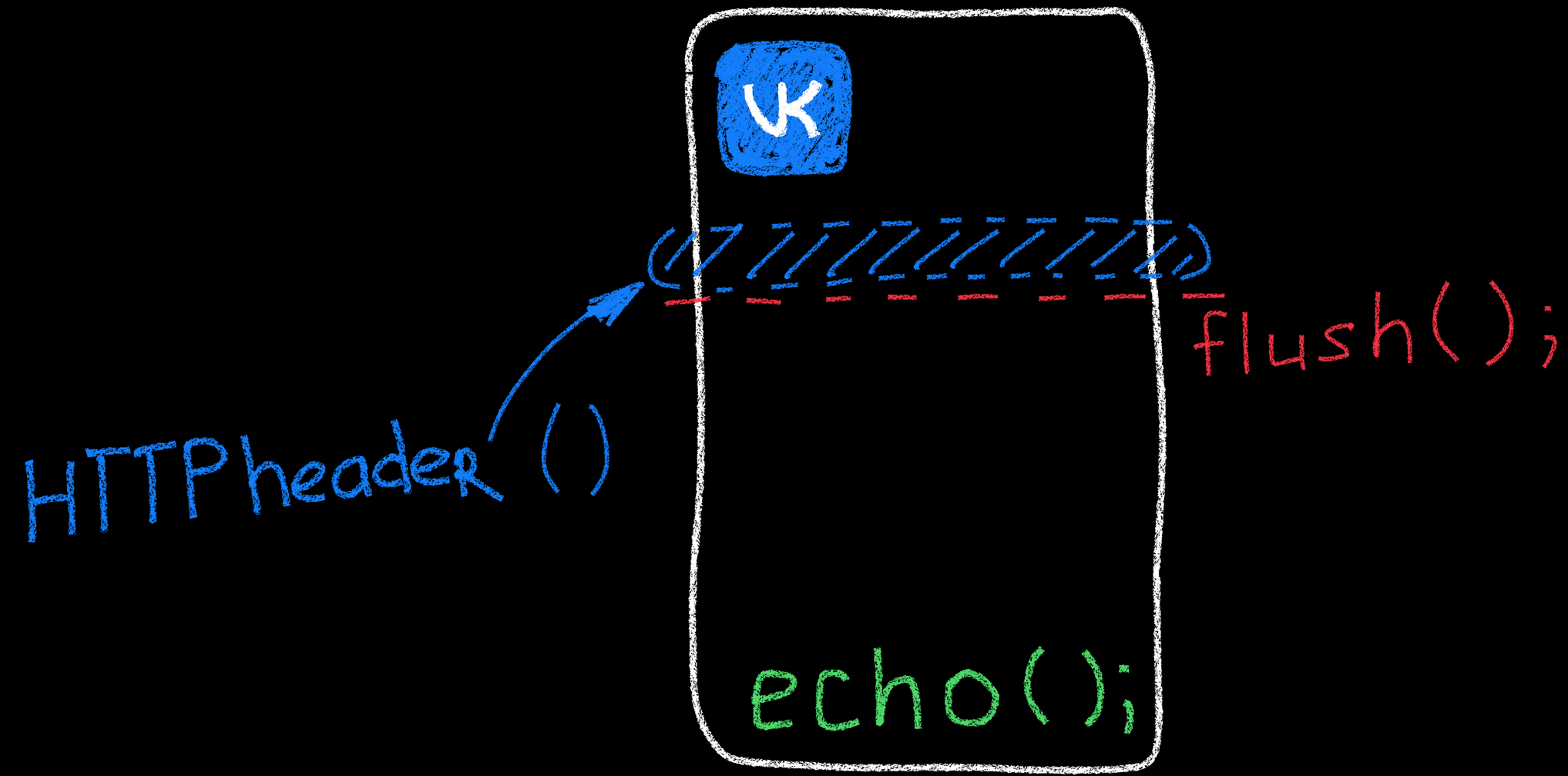
Флаш: 1.693s

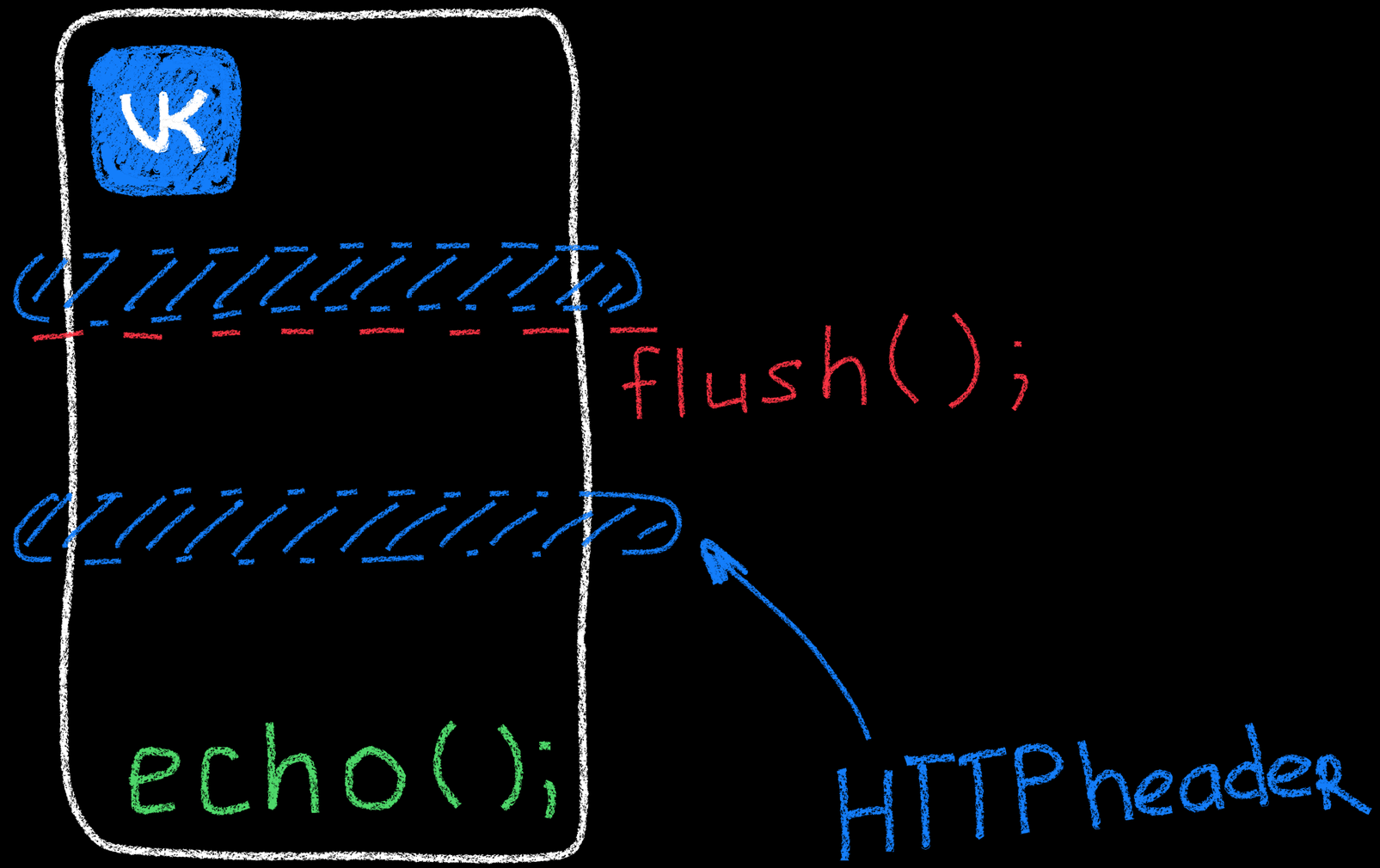


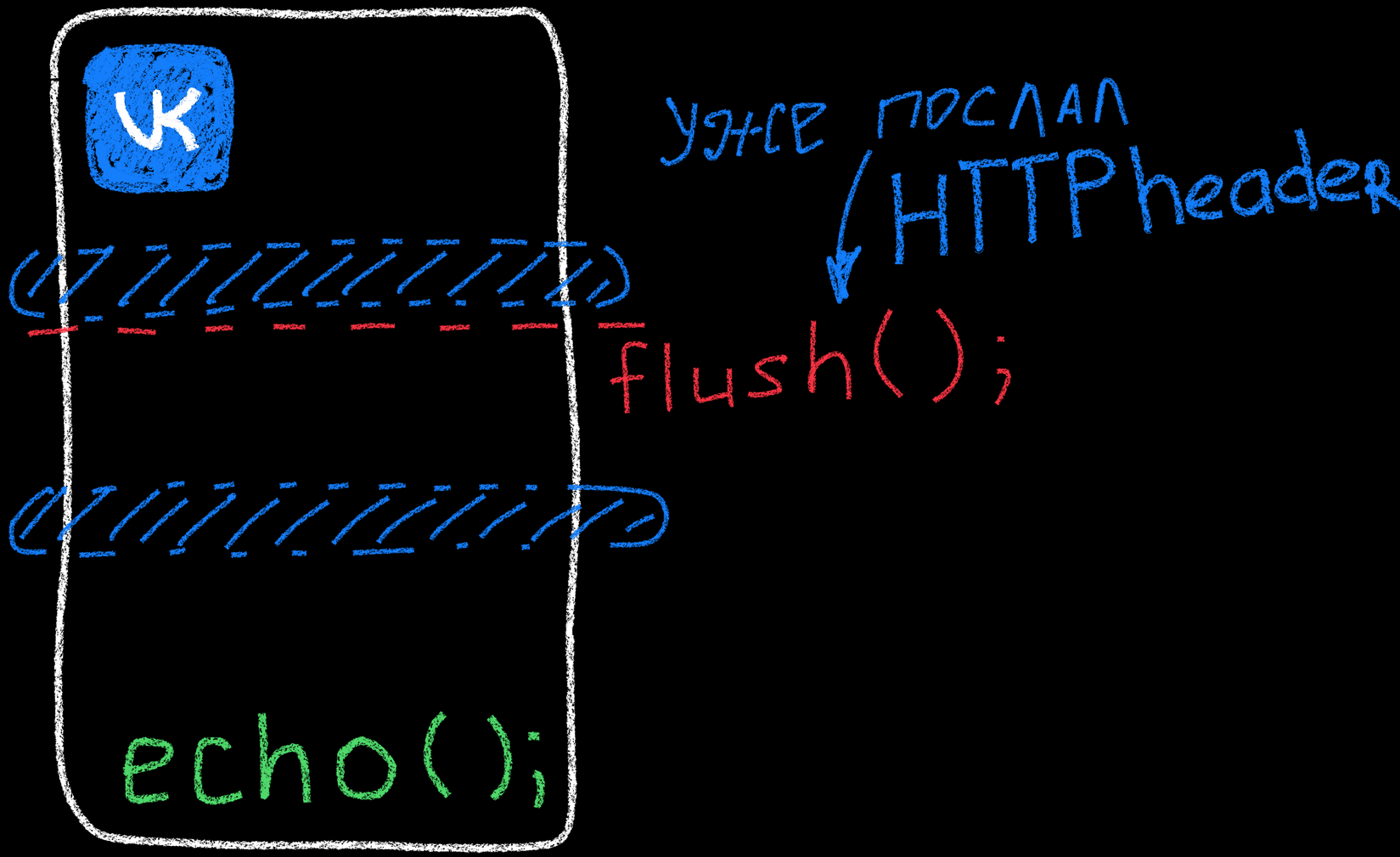
# профит есть. чиним логин

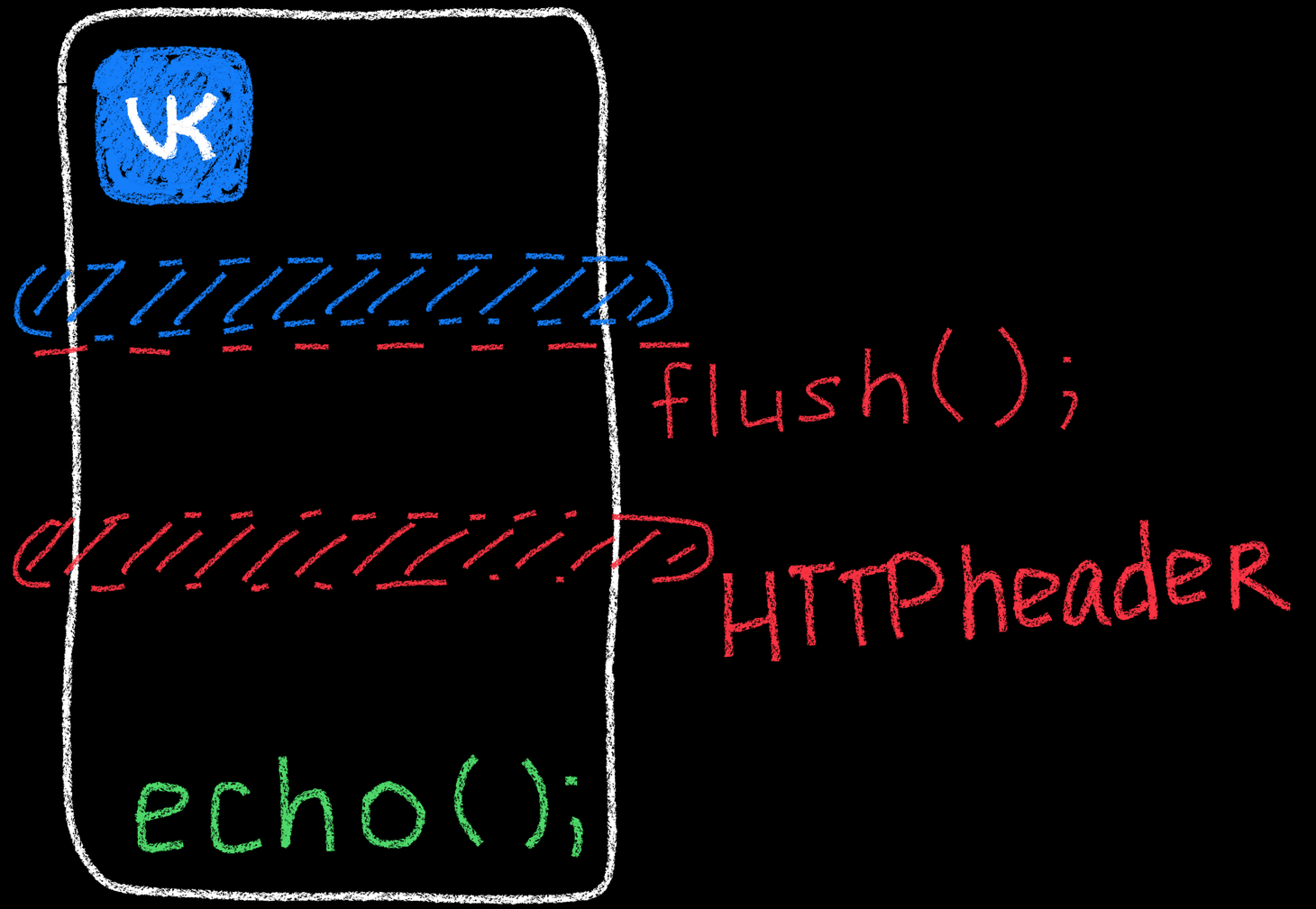



flush();
echo();

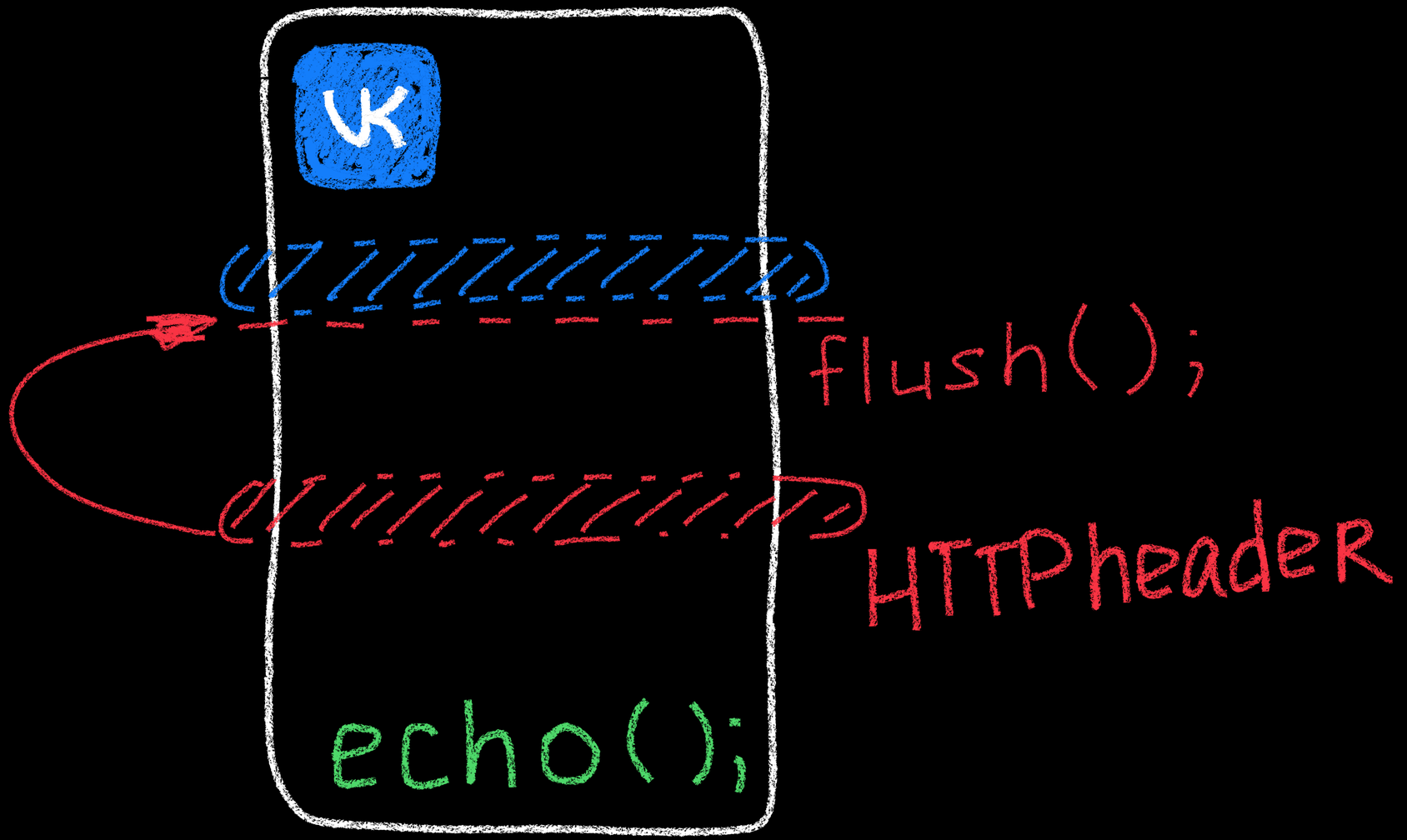

HTTPheader ()
flush();
echo();


flush();
echo();
HTTP header

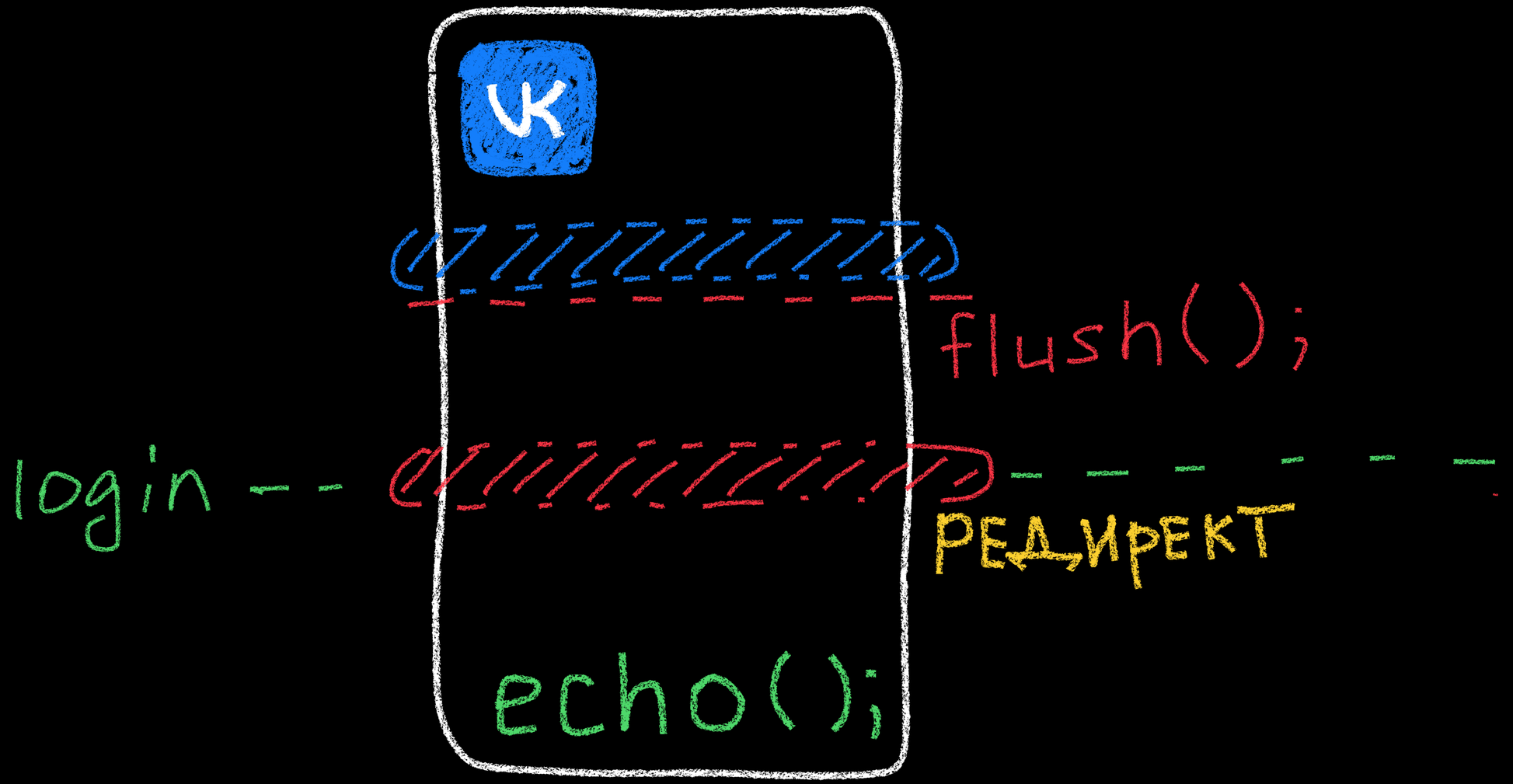

уже послал
HTTP header
flush();
echo();


flush();
HTTPheader
echo();

# Собираем и переносим



# Собираем и переносим



редиректы происходят через единую функцию

переадресовываем
с помощью мета-тега

```
<meta
  http-equiv="refresh" // переадресовать
  content="..." // куда
/>
```

```
<meta
  http-equiv="refresh" // переадресовать
  content=“0; $url" // через сколько, куда
/>
```

# Дополняем мега-класс

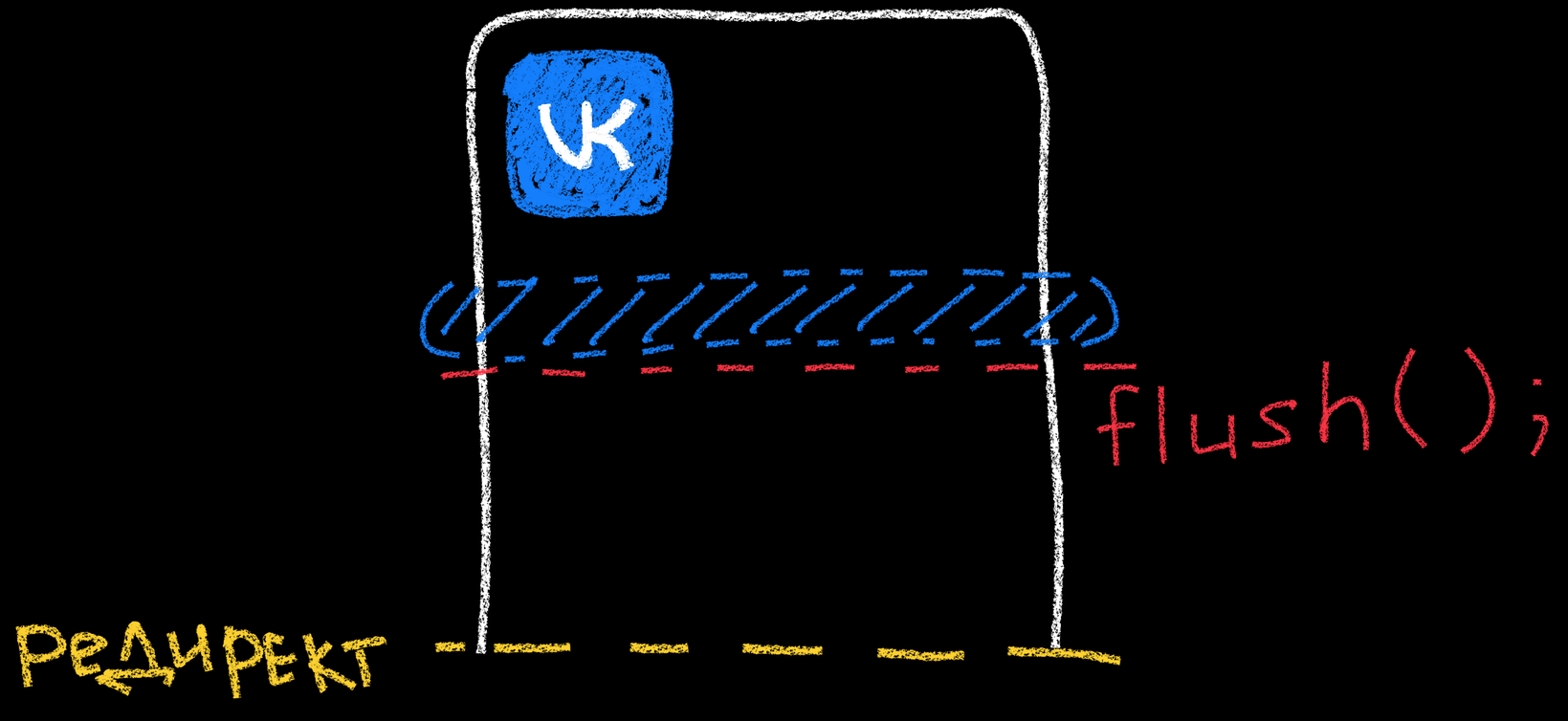
```
class FlushChunks {

  public static function isFlushAvailable(): bool {
    ...
  }

  public static function earlyFlush(): void {
    flush();
  }

}
```

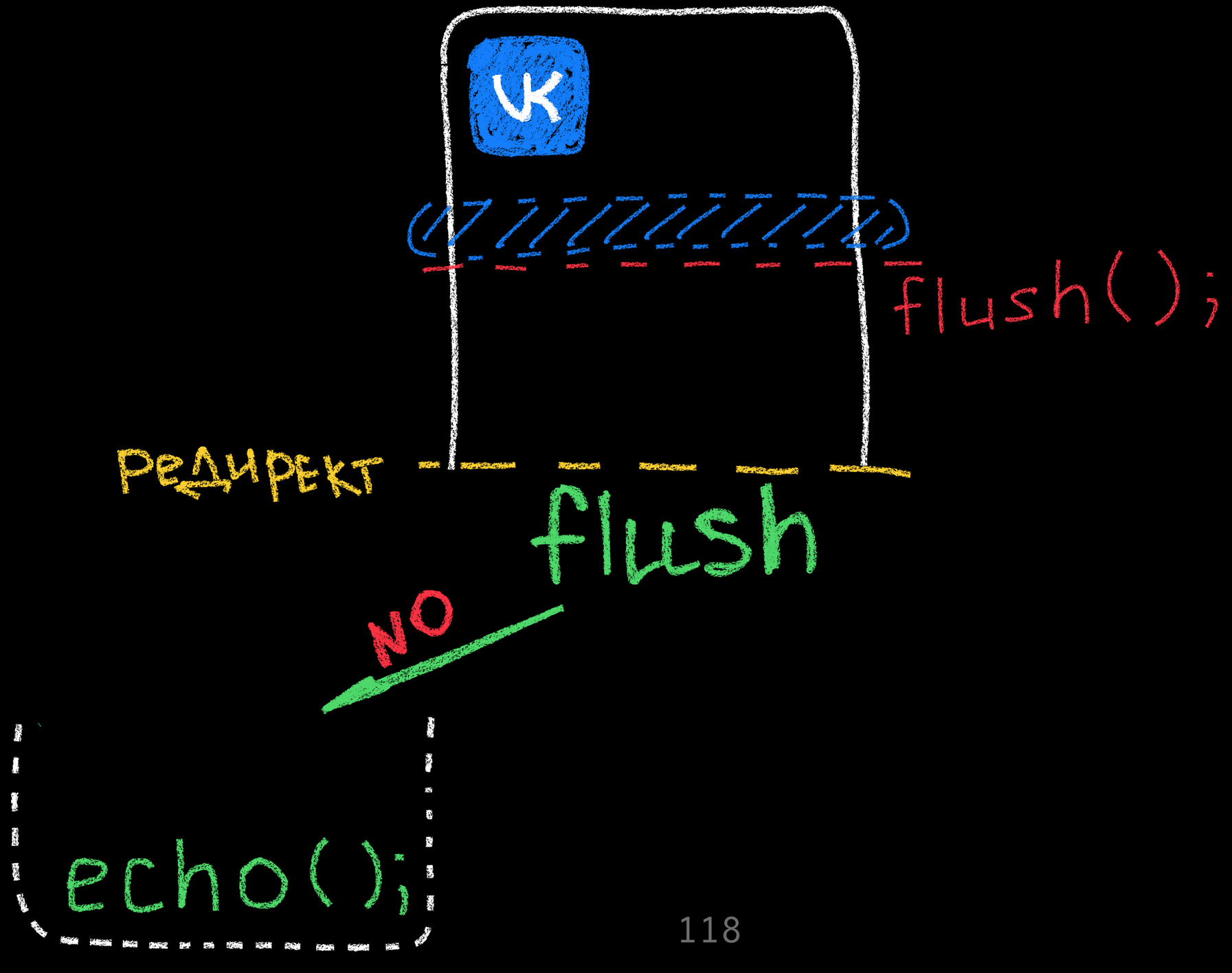
# Дополняем мега-класс

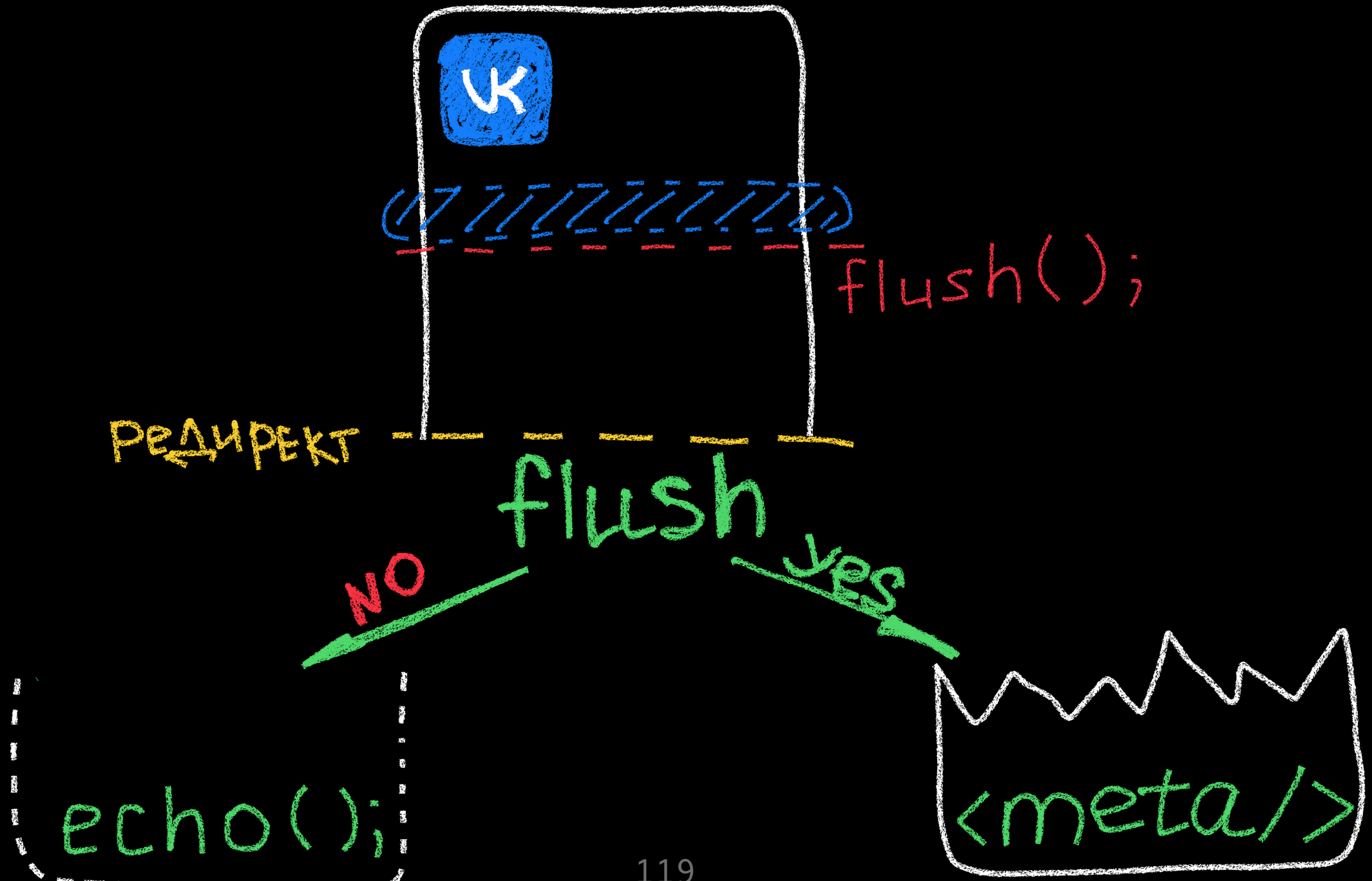
```php
class FlushChunks {

  public static function isFlushAvailable(): bool {
    ...
  }

  public static function earlyFlush(): void {
    flush();
    $flushed = true;
  }

}
```

# Итоговое условие

```
if ($flushed) {
  flush(<meta http-equiv="refresh" ... />)
} else {
  header('Location: ' . $redirect_url);
}
```


flush();
редирект

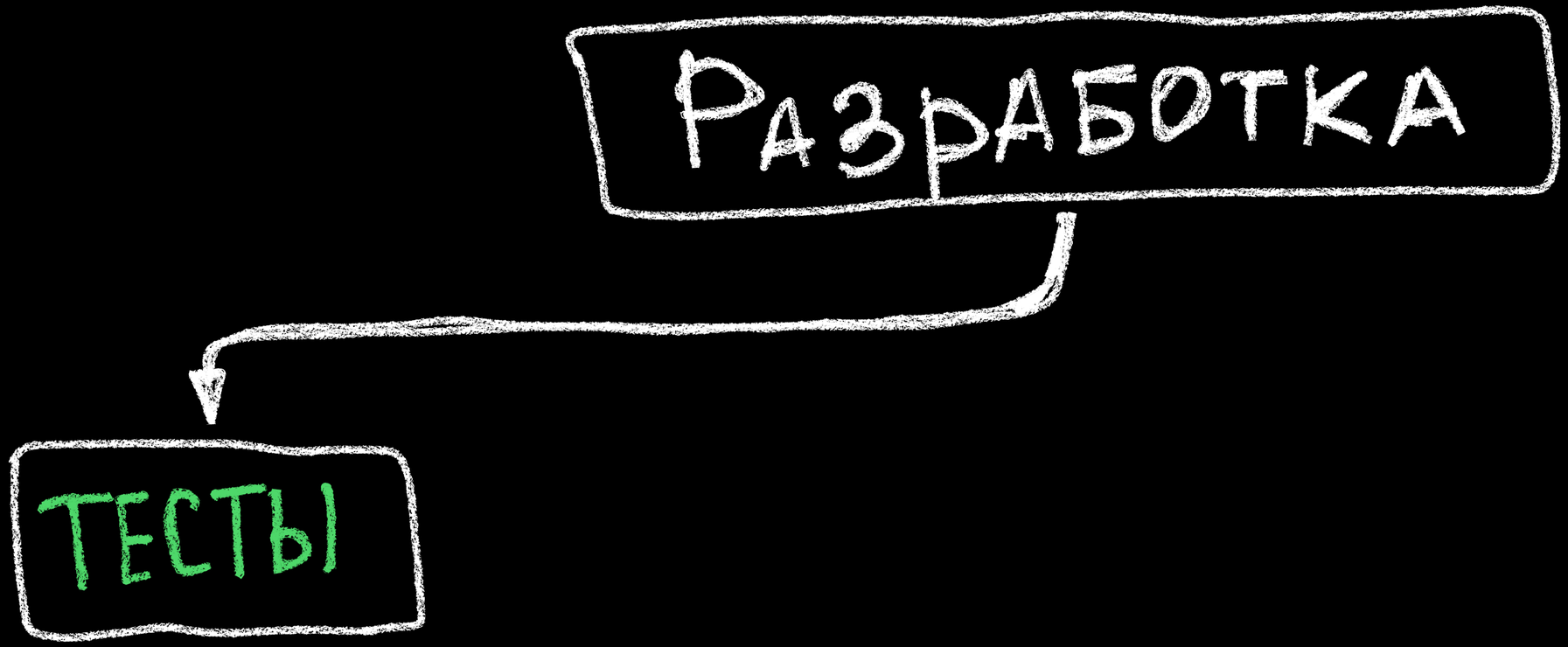

flush();
редирект
NO
flush
echo();


flush();
редирект
flush
NO
YES
echo();
<meta/>

запускаем...

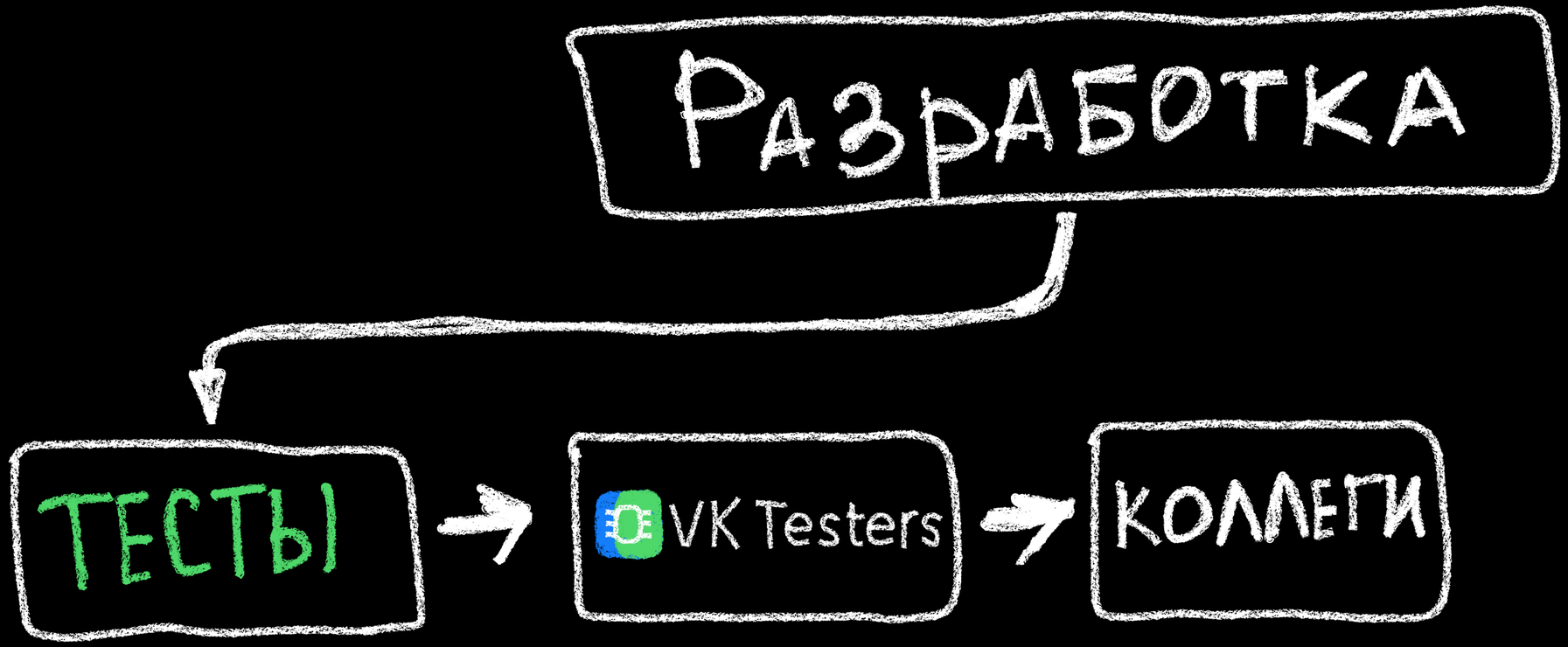
# запускаем… в наш цикл разработки

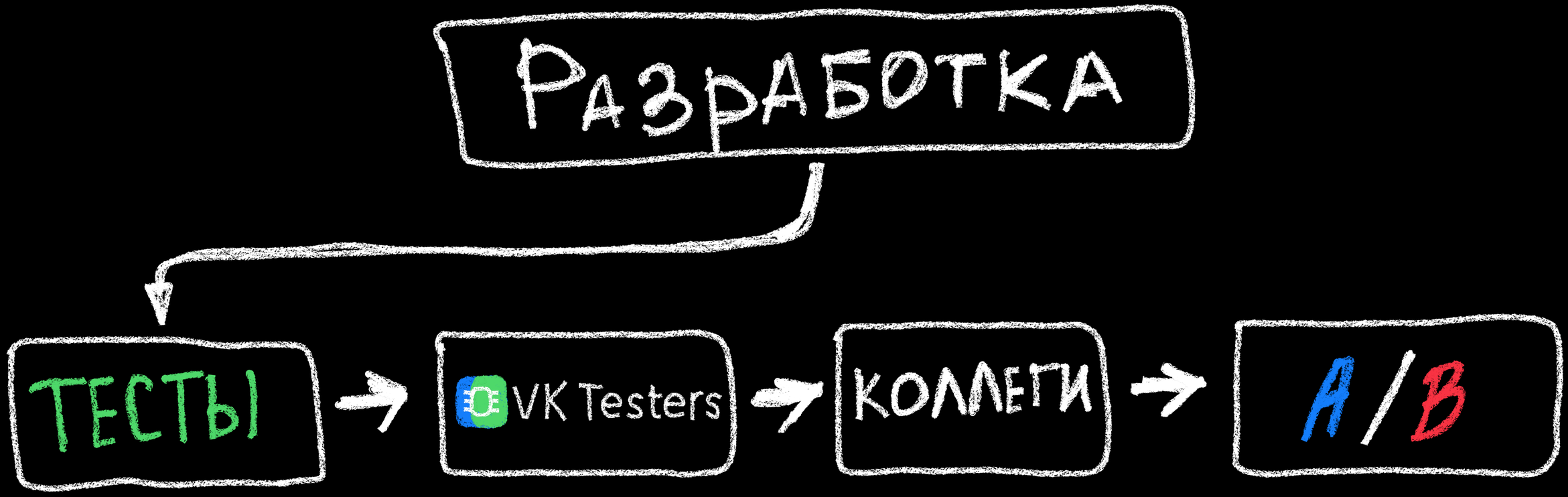
# Разработка

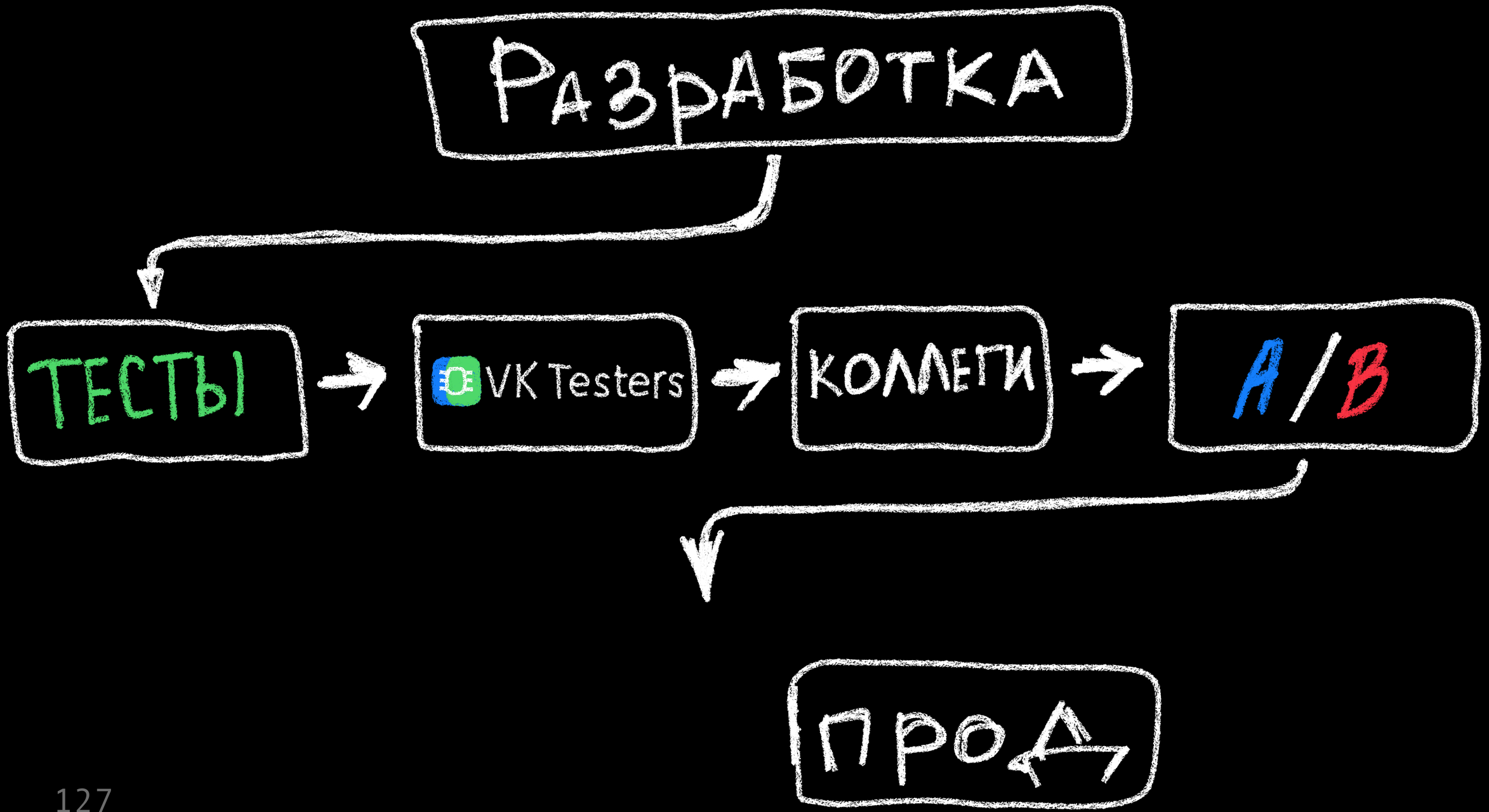

Разработка
тесты

Разработка

Тесты → 


Разработка
Тесты
VK Testers
Коллеги

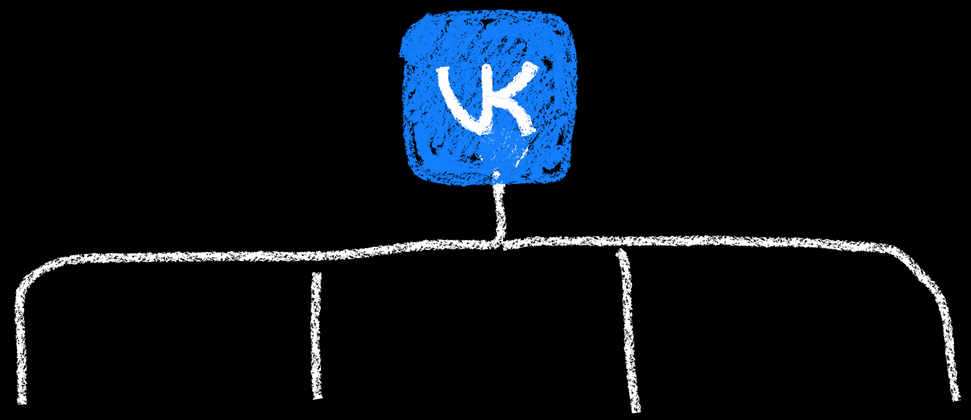

Разработка
Тесты
VK Testers
Коллеги
A/B

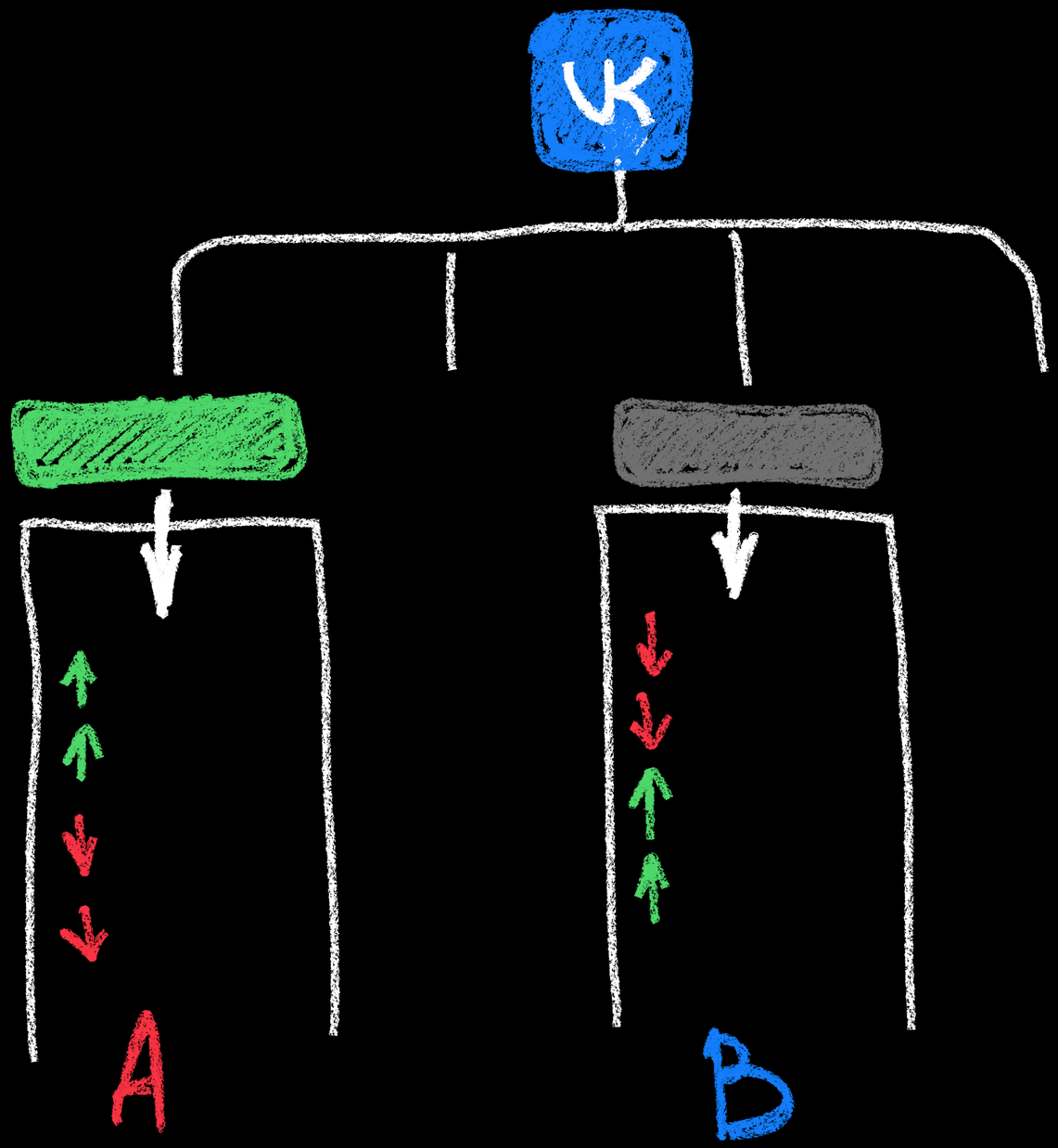

Разработка
тесты
VK Testers
коллеги
A/B
прод

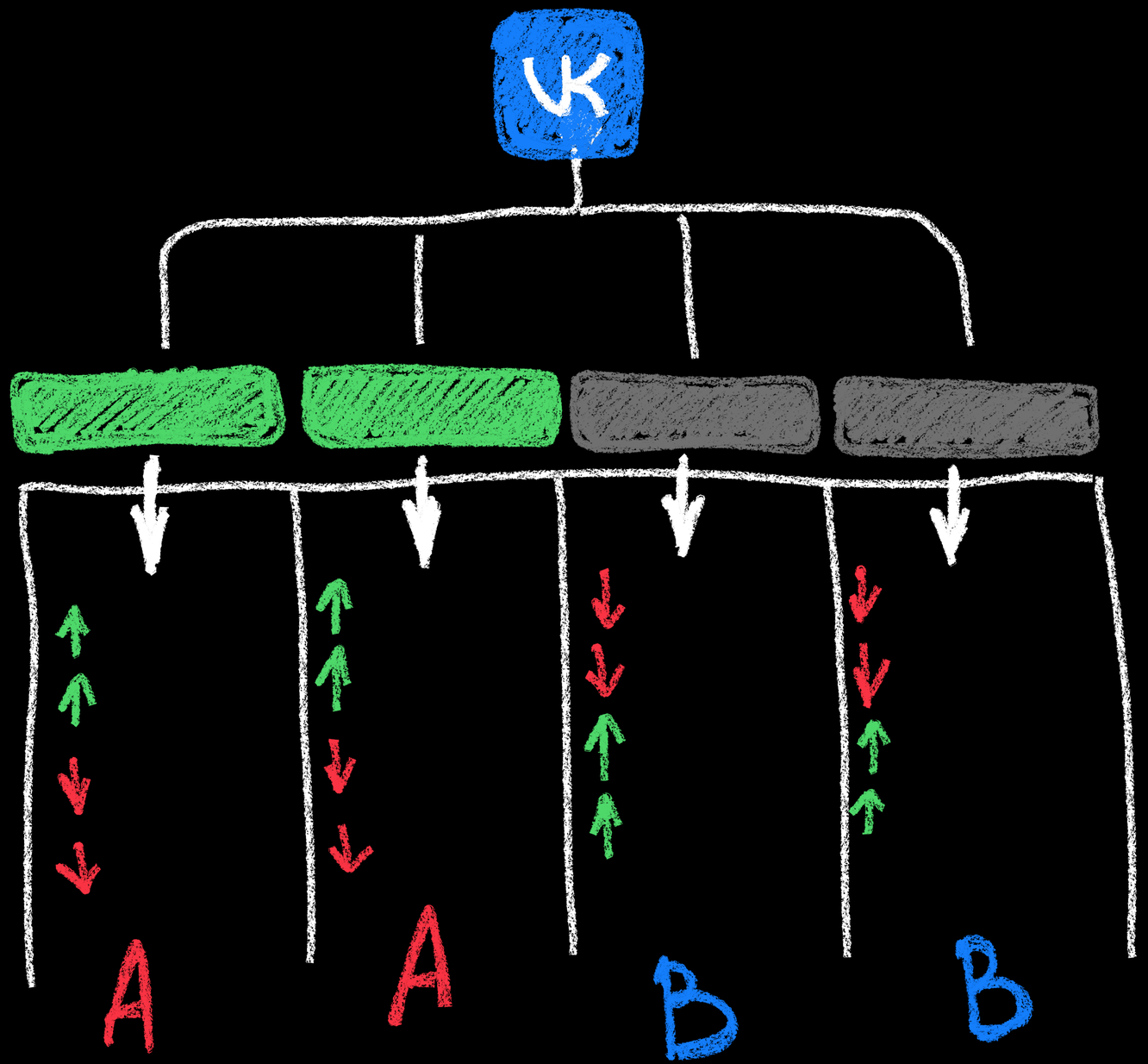

Разработка
тесты
VK Testers
коллеги
A/B
прод

# что нашли в АБ

# что нашли в ААББ




A
B


A
A
B
B

# Пример отчетов

**Краткая сводка результатов эксперимента 8054 за 2024-03-14**

Ссылка на дэшборд: https://vk.cc/cv4UEY

В сравнении **AB** контрольной и тестовой групп:

⬆ : 10 метрик, ⬇ : 27 метрик

| Метрика | Отн.эффект, % | Границы отн.эффекта, % | MDE, % | p-value | Платформа |
|---|---|---|---|---|---|
| **total_menu_calls_taps** | ⬆**18.69** | [3.08, 34.3] | 22.76 | 0.0202 | all |
| **bookmarks_open_uniques** | ⬇**-7.38** | [-10.6, -4.15] | 4.62 | 0.0007 | all |
| **comments_created** | ⬇**-9.26** | [-16.8, -1.73] | 10.79 | 0.0218 | all |
| **feed_cls_mvk_uniques** | ⬆**21.0** | [18.06, 23.94] | 4.21 | 0.0 | mvk |
| **timespent_feed_mvk** | ⬆**13.65** | [11.69, 15.61] | 2.8 | 0.0 | mvk |
| **timespent_feed_mvk_uniques** | ⬆**5.19** | [3.77, 6.62] | 2.04 | 0.0 | mvk |
| **feed_ttfb_mvk_uniques** | ⬆**3.68** | [1.97, 5.39] | 2.44 | 0.002 | mvk |
| **feed_ttfb_mvk_uniques** | ⬆**3.68** | [1.97, 5.39] | 2.44 | 0.002 | mvk |
| **feed_fcp_mvk_uniques** | ⬆**3.63** | [2.01, 5.26] | 2.33 | 0.0025 | mvk |
| **feed_fcp_mvk_uniques** | ⬆**3.63** | [2.01, 5.26] | 2.33 | 0.0025 | mvk |
| **feed_ttlb_mvk_uniques** | ⬆**3.62** | [1.91, 5.32] | 2.44 | 0.0027 | mvk |
| **timespent_mvk_uniques** | ⬆**2.31** | [1.27, 3.34] | 1.48 | 0.0102 | mvk |

# Находили всякое

# Находили всякое

- Упали просмотры статей;

# Находили всякое

- Упали просмотры статей;
- Сломались гифки;

# Находили всякое

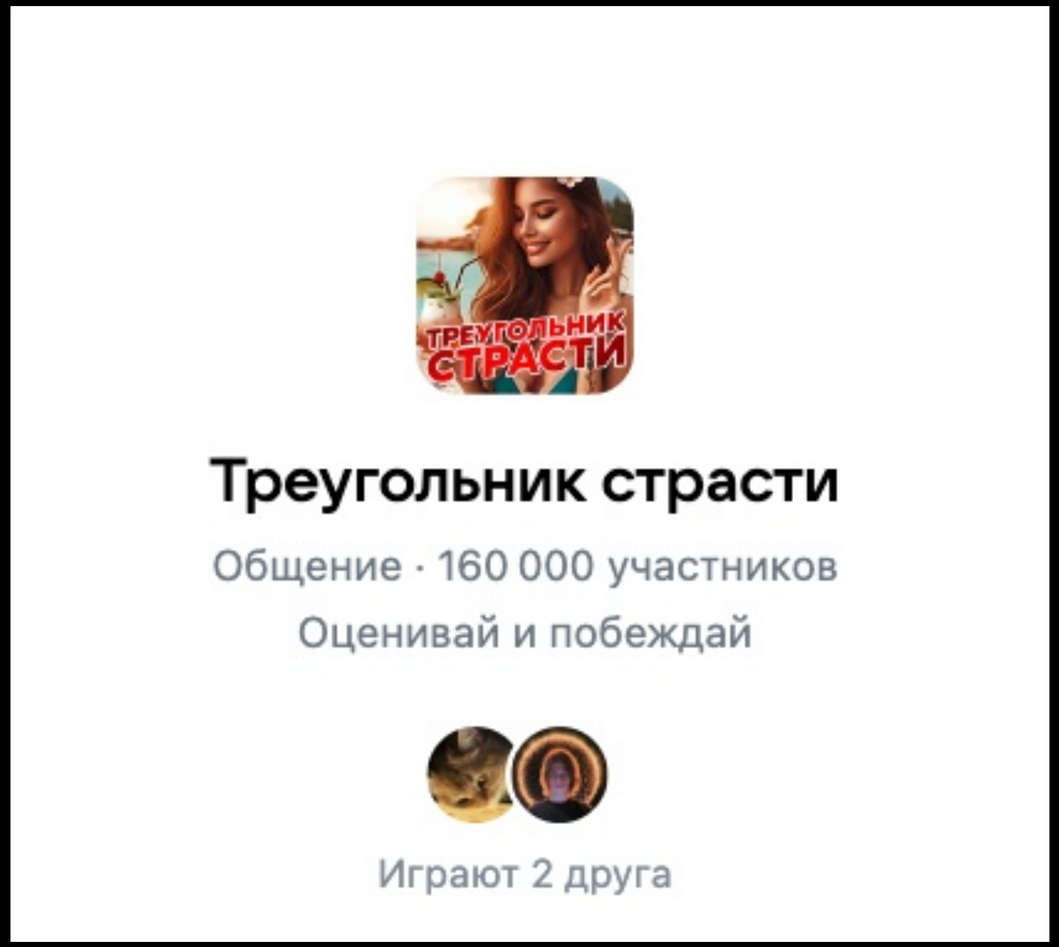
- Упали просмотры статей;
- Сломались гифки;
- Флаш не флашился;

# Находили всякое

- Упали просмотры статей;
- Сломались гифки;
- Флаш не флашился;
- Скрипты прелоадились с другого домена;

# Находили всякое

- Упали просмотры статей;
- Сломались гифки;
- Флаш не флашился;
- Скрипты прелоадились с другого домена;
- Сломались игры.



# Добавляем $blacklisted. Бан за плохой флаш.

# Дополняем мега-класс

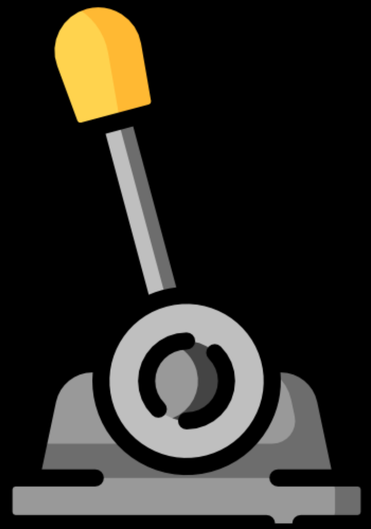
```php
class FlushChunks {

  public static function isFlushAvailable(): bool {
    ...
  }

  public static function earlyFlush(): void {
    flush();
    $flushed = true;
  }

}
```

# Дополняем мега-класс

```php
class FlushChunks {

  public static function isFlushAvailable(): bool {
    if ($blacklisted) {
      return false;
    }
  }

  public static function earlyFlush(): void {
    flush();
    $flushed = true;
  }

}
```

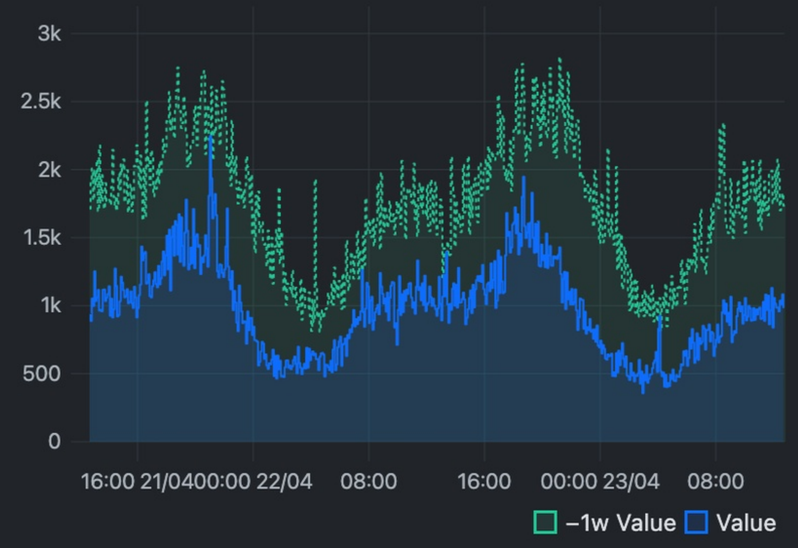
тестировали и чинили
~1 месяц

# настало время ехать в прод

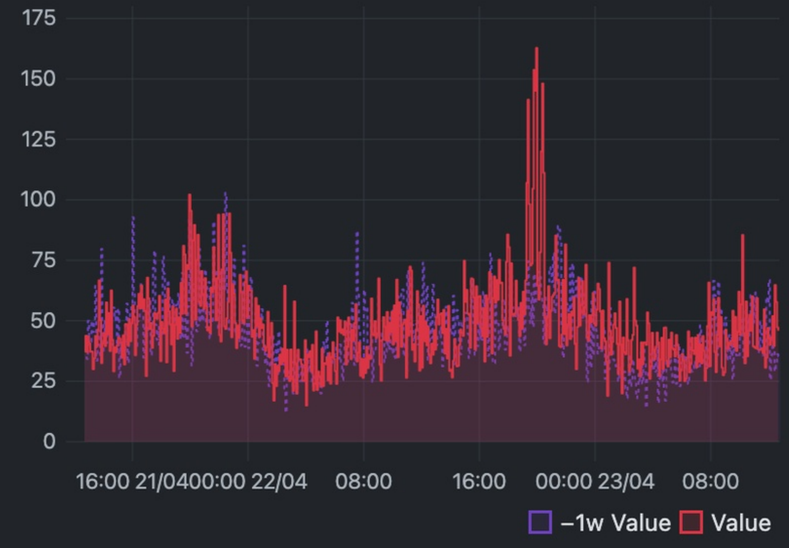


# Крутим и следим

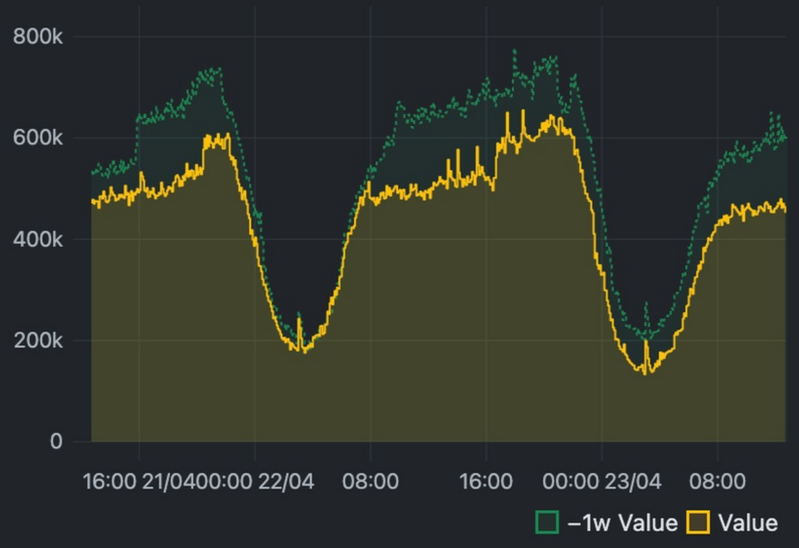
JS ошибки




Страницы ошибок



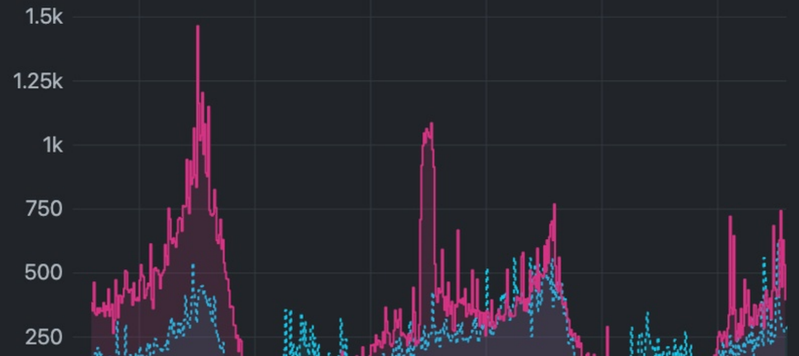

Отданные страницы



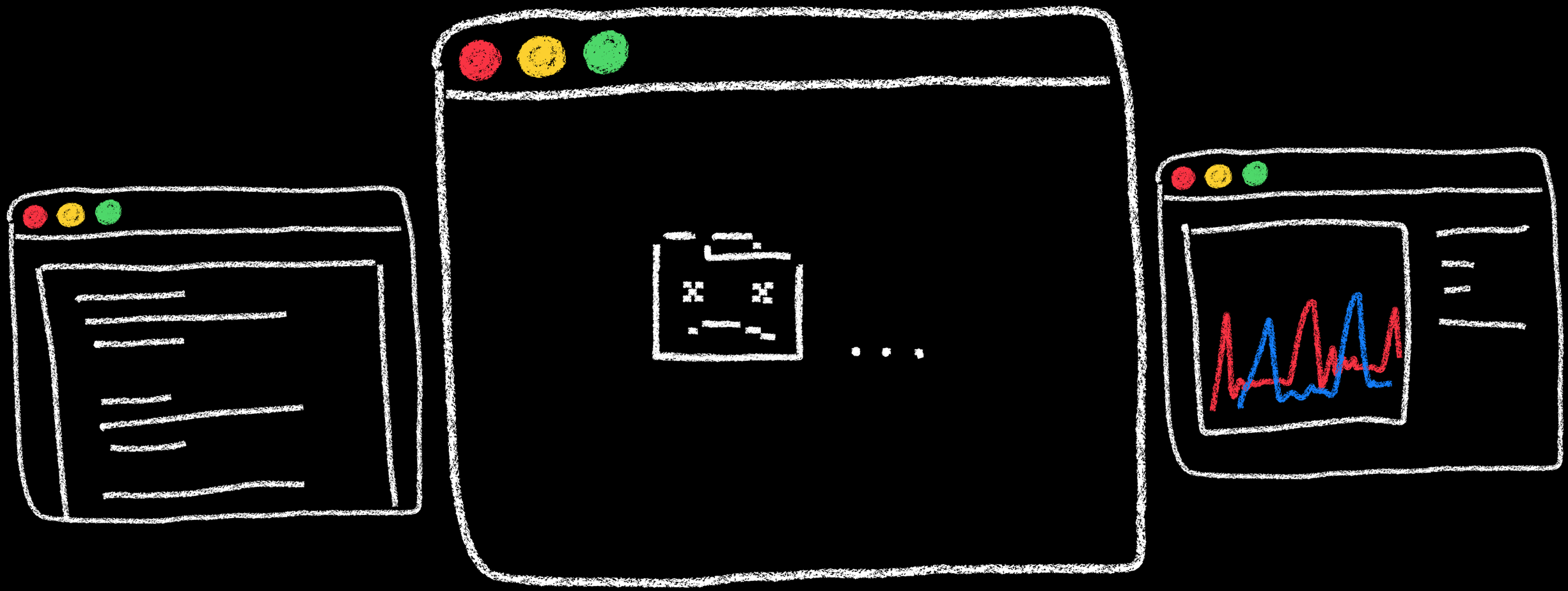

Отданные страницы без ботов и вебв


Обновления страницы после деплоя


KPHP ошибки (все проекты)

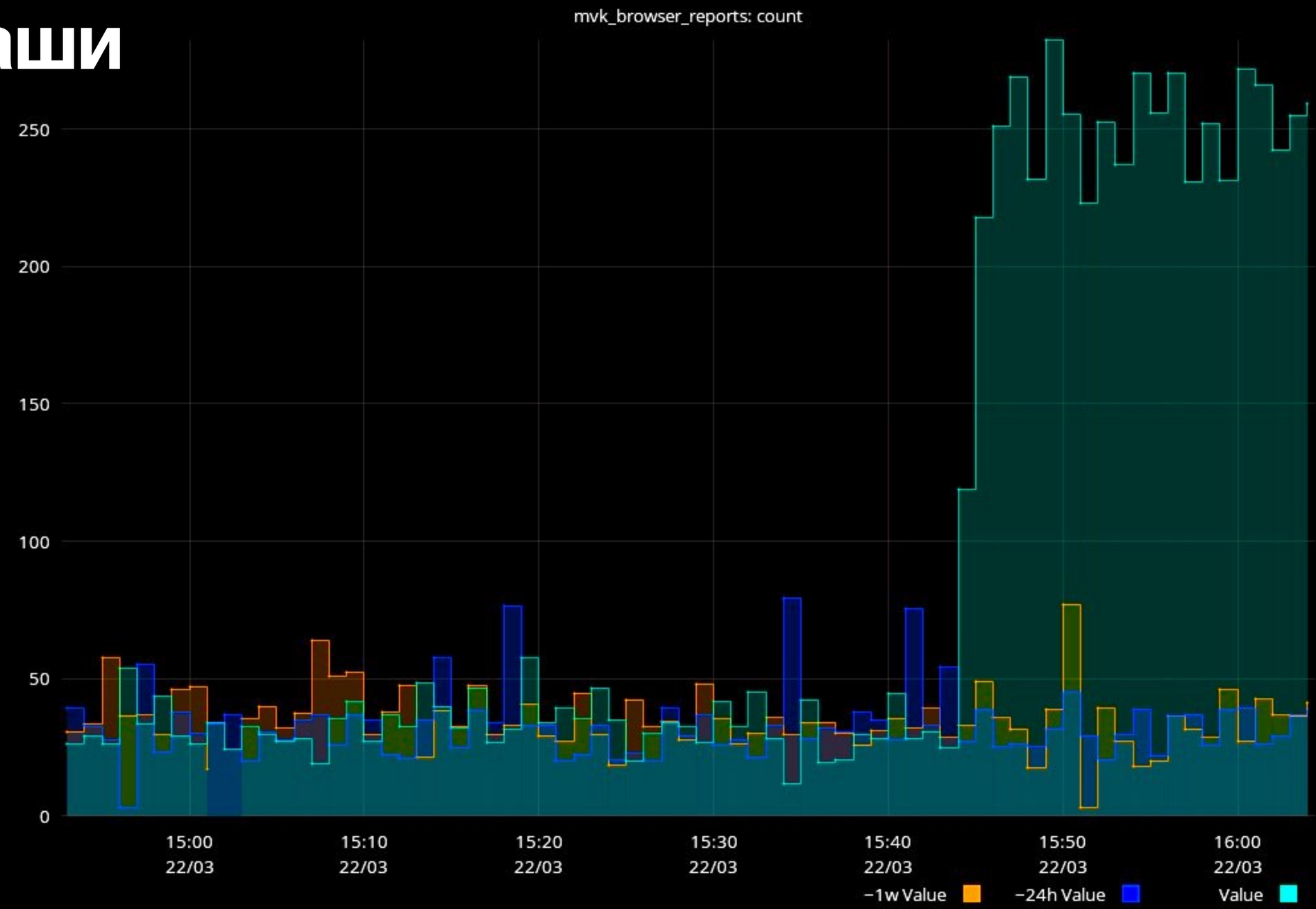


# Reporting API



10%…25%…50%

# Краши


mvk_browser_reports: count
250
200
150
100
50
0
15:00
22/03
15:10
22/03
15:20
22/03
15:30
22/03
15:40
22/03
15:50
22/03
16:00
22/03
−1w Value
−24h Value
Value

дебажим.


nthouse Recorder Performance insights 1

Default levels

-723.b04f0…pGRTJ6UXg0OXZkRk15TkVENGU1aTBVYjVVcEtfT3ZDNE9UNThOZ05oT0VmV1dXOWwza0Njdg-- net::ERR_CONTENT_DECODING_FAILED 200 (OK) /login?slogin_h=d544…mV1dXOWwza0Nj

меняем стратегию: $whitelisted.
даем флаш только
по пригласительному

# Дополняем мега-класс

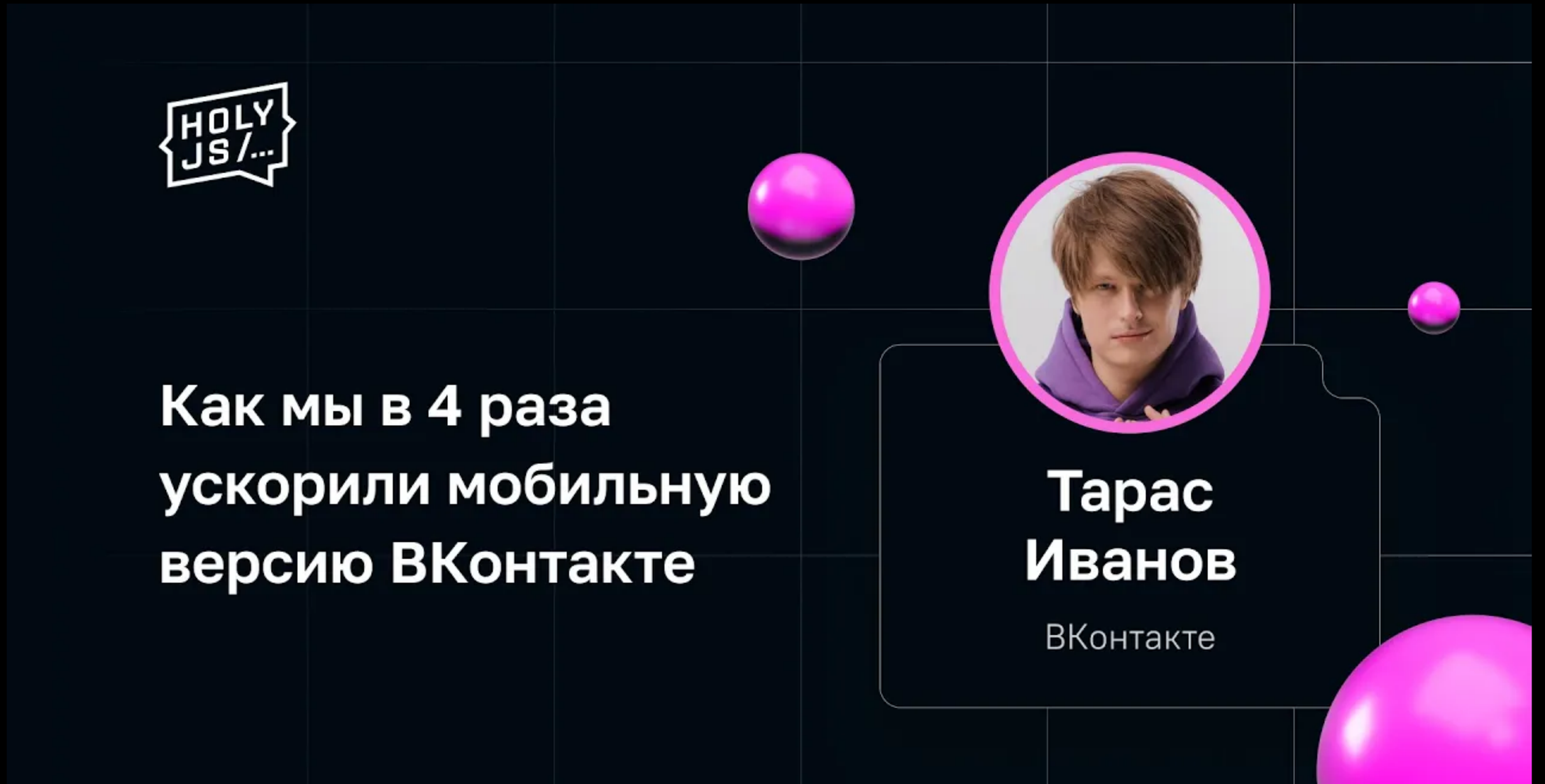
```php
class FlushChunks {

  public static function isFlushAvailable(): bool {
    if ($blacklisted) {
      return false;
    }
  }

  public static function earlyFlush(): void {
    flush();
    $flushed = true;
  }

}
```

# Белый список

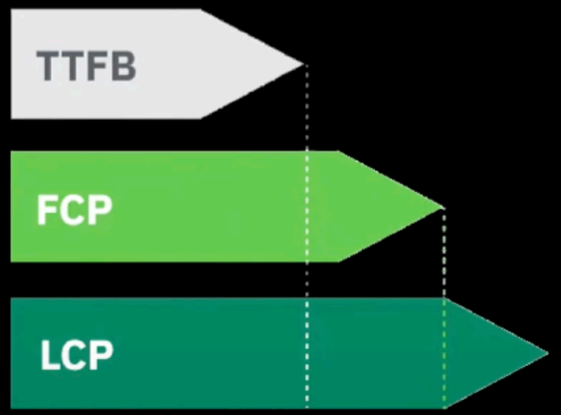
```php
class FlushChunks {

  public static function isFlushAvailable(): bool {
    if ($whitelisted) {
      return false;
    }
  }

  public static function earlyFlush(): void {
    flush();
    $flushed = true;
  }

}
```

какой профит?

# Про метрики


HOLY JS
Как мы в 4 раза ускорили мобильную версию ВКонтакте
Тарас Иванов
ВКонтакте

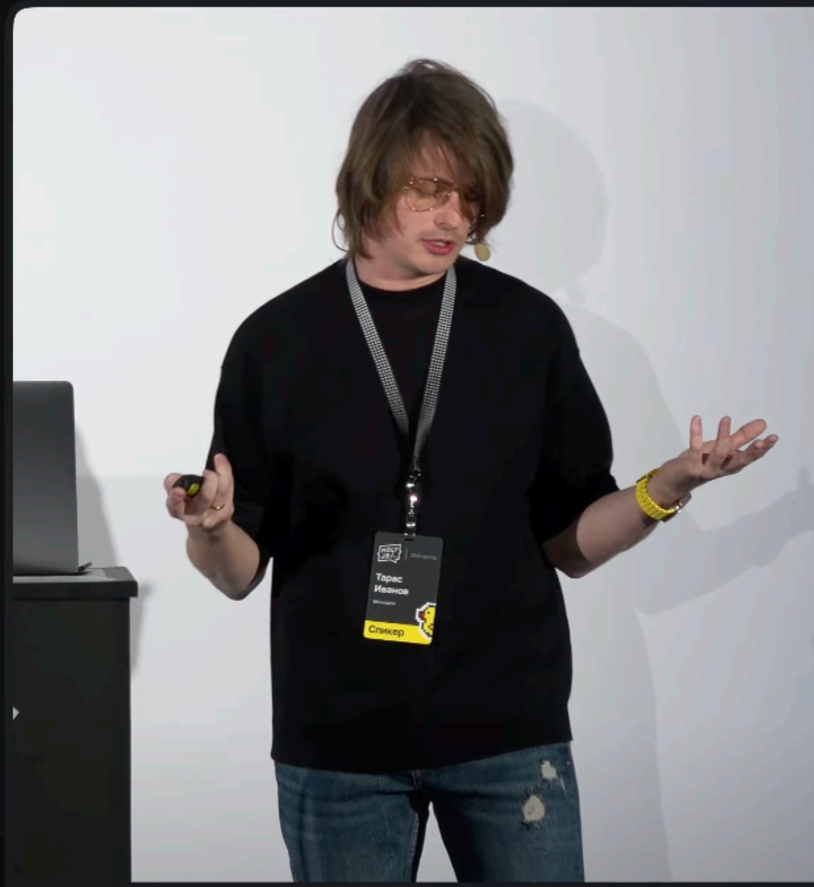
# Web Vitals




TTFB — время до получения первого байта (вычисления на сервере + сеть)

FCP — время до того, как на экране появится хоть что-нибудь

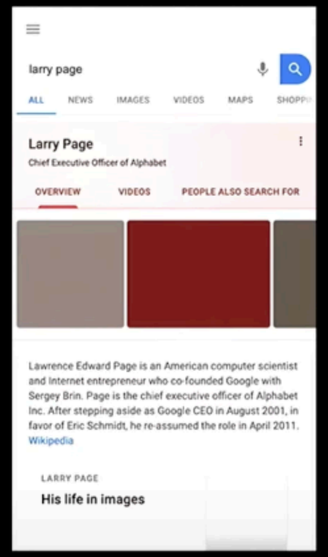
LCP — время до того, как на экране появится полезный контент

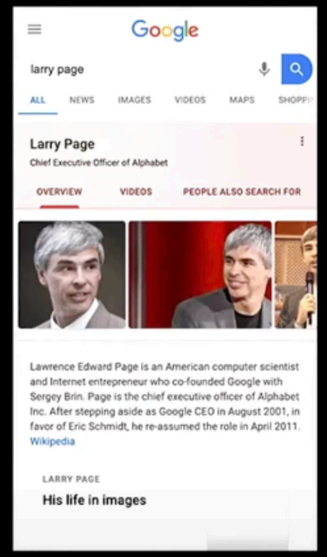


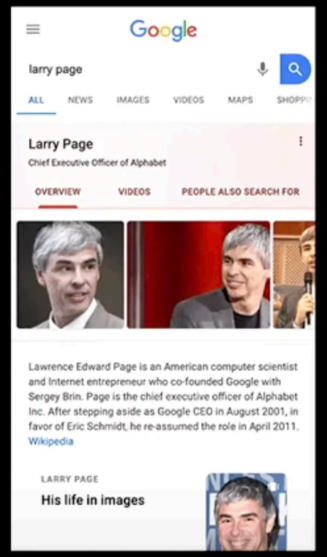

Тарас Иванов

https://vk.com/ti

TTFB FCP LCP

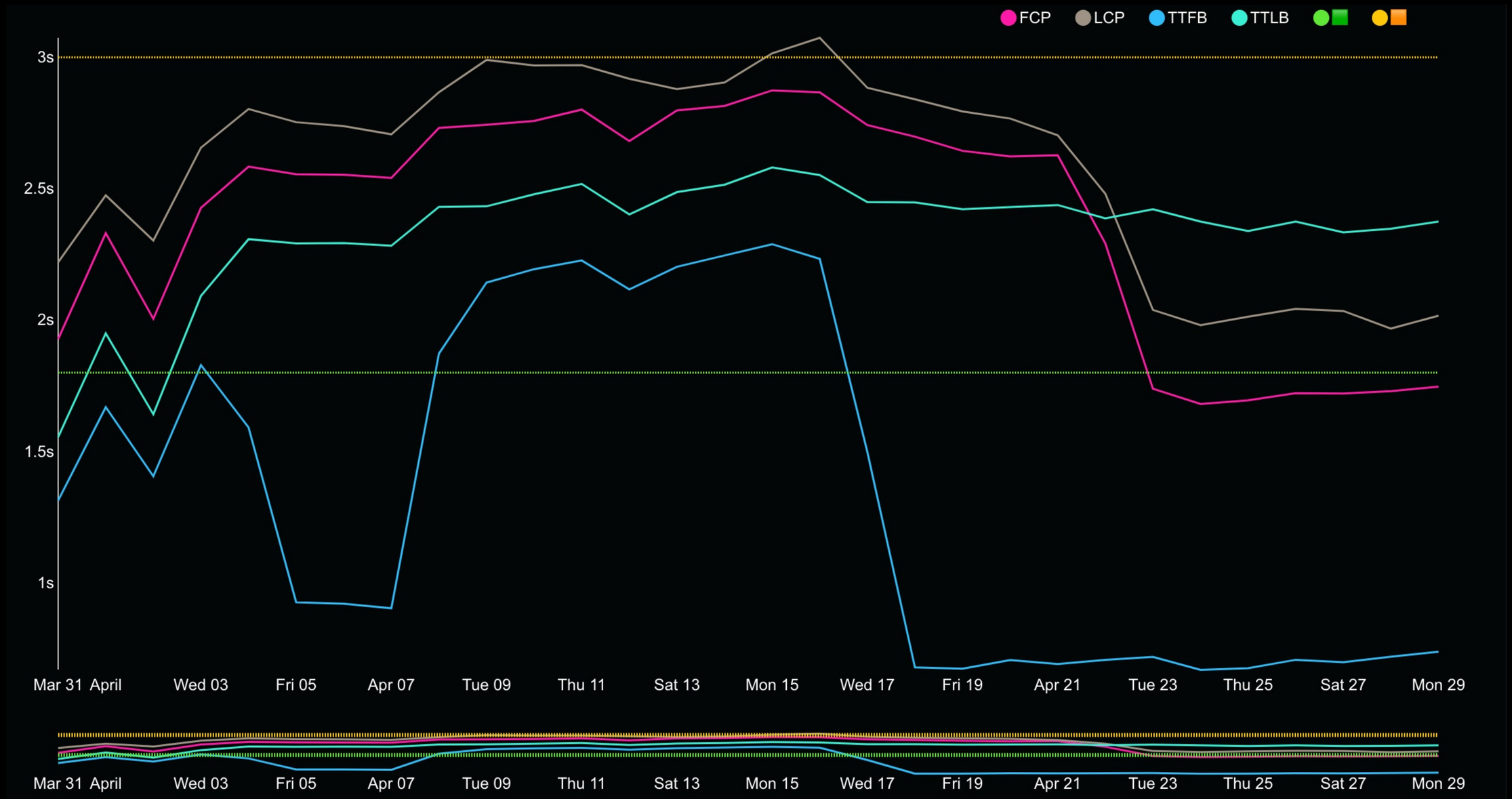



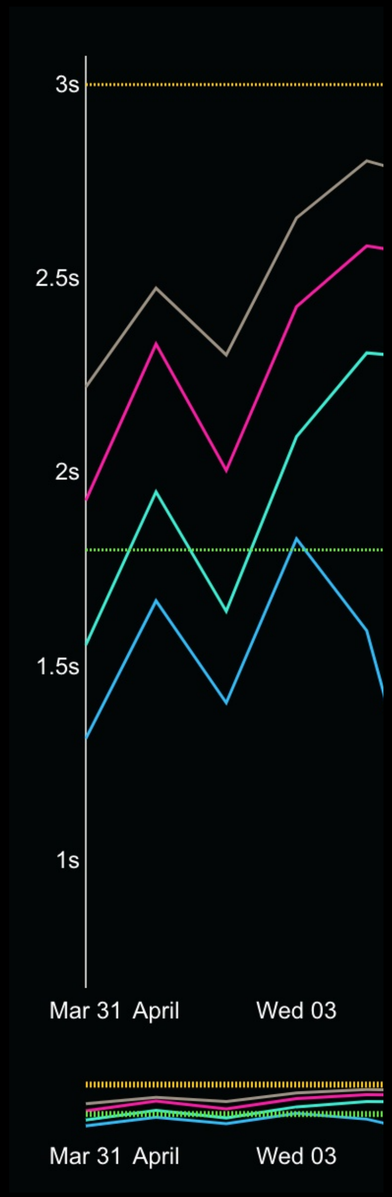

Тарас Иванов

https://vk.com/ti

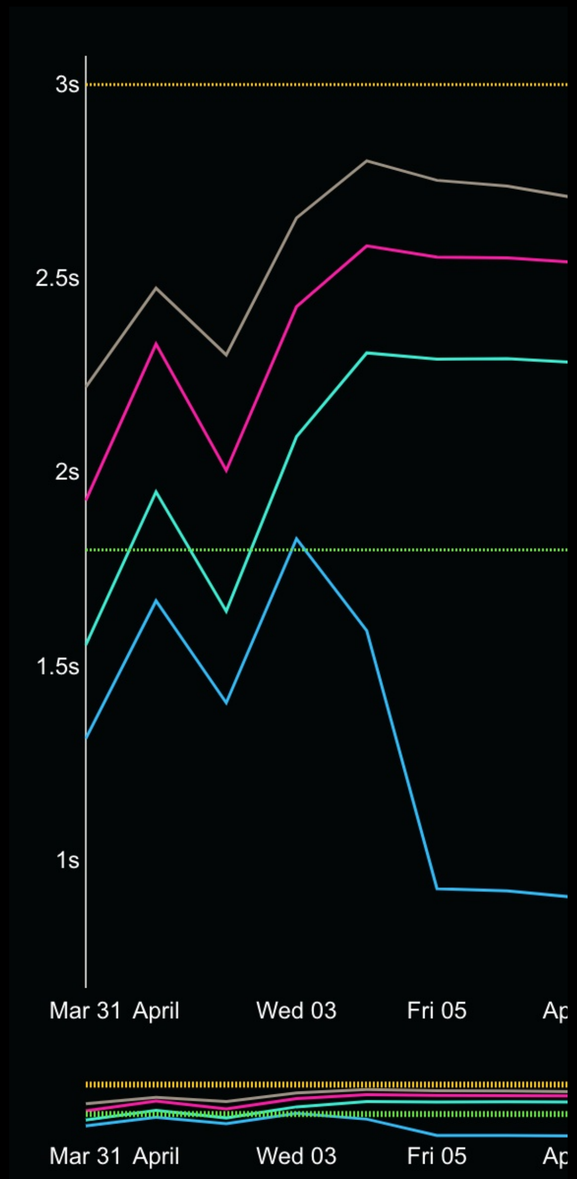
TTFB (First Byte) для души,
TTLB (Last Byte) для контроля

смотрим графики

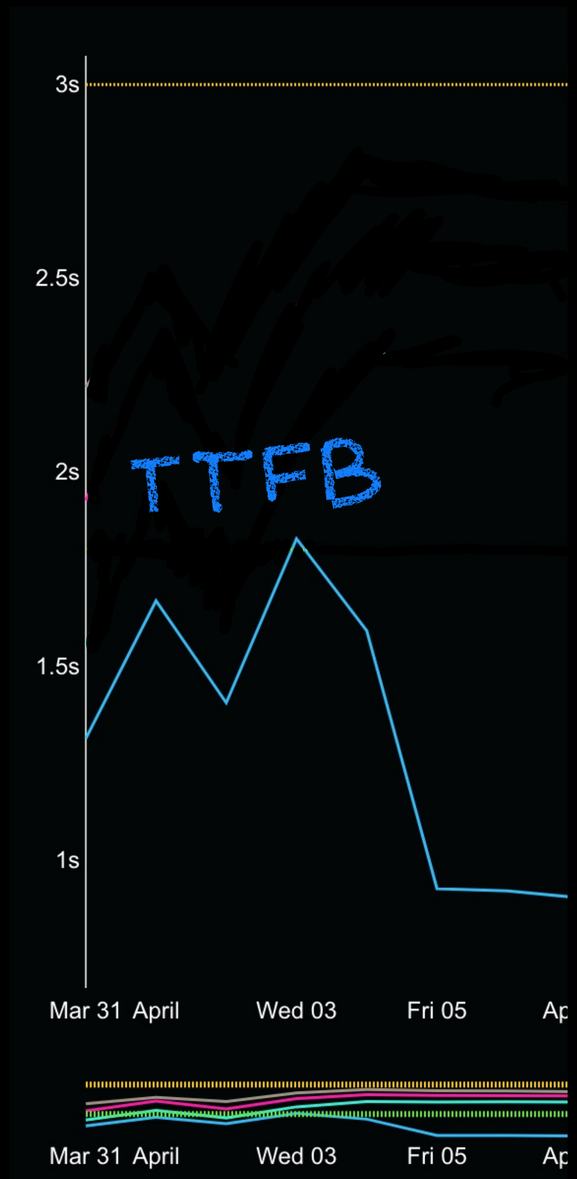

FCP
LCP
TTFB
TTLB
3s
2.5s
2s
1.5s
1s
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29



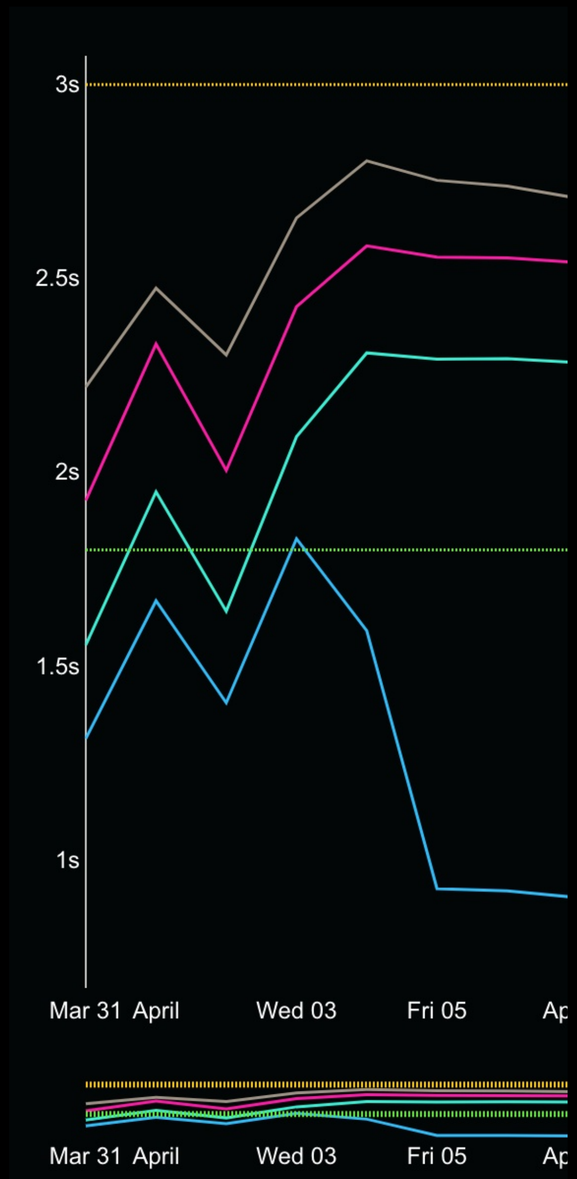

# График АААББ



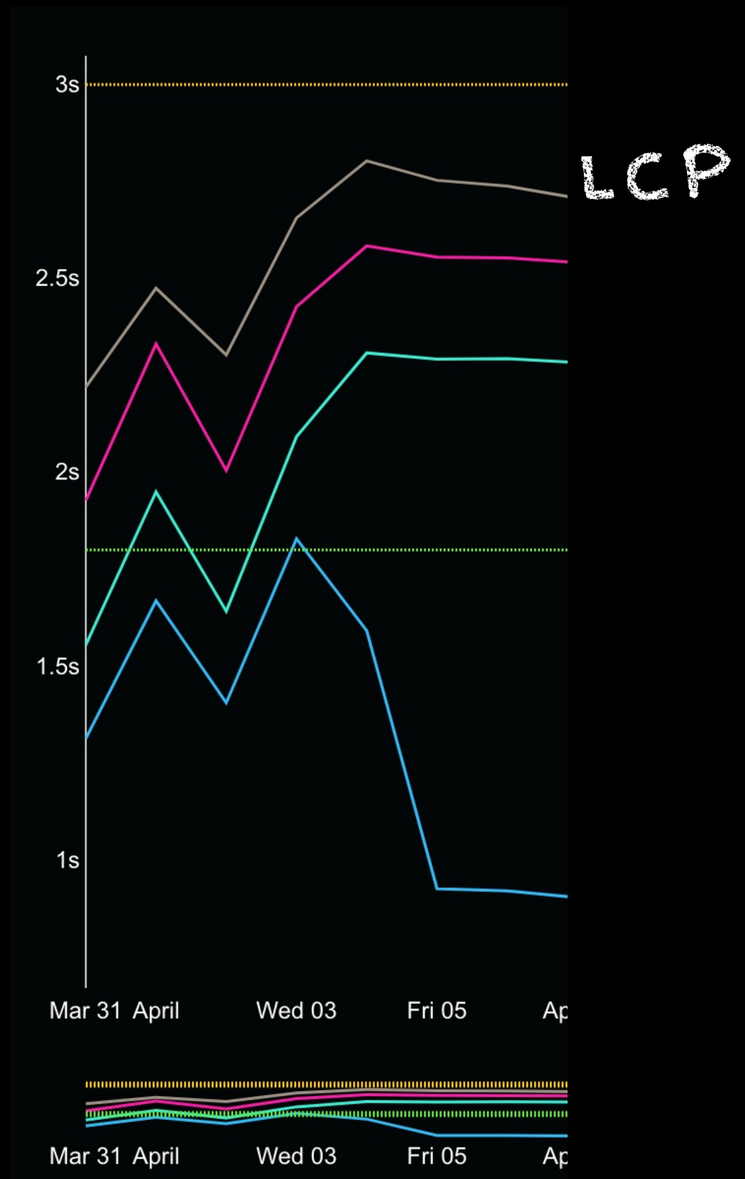

# Первое включение



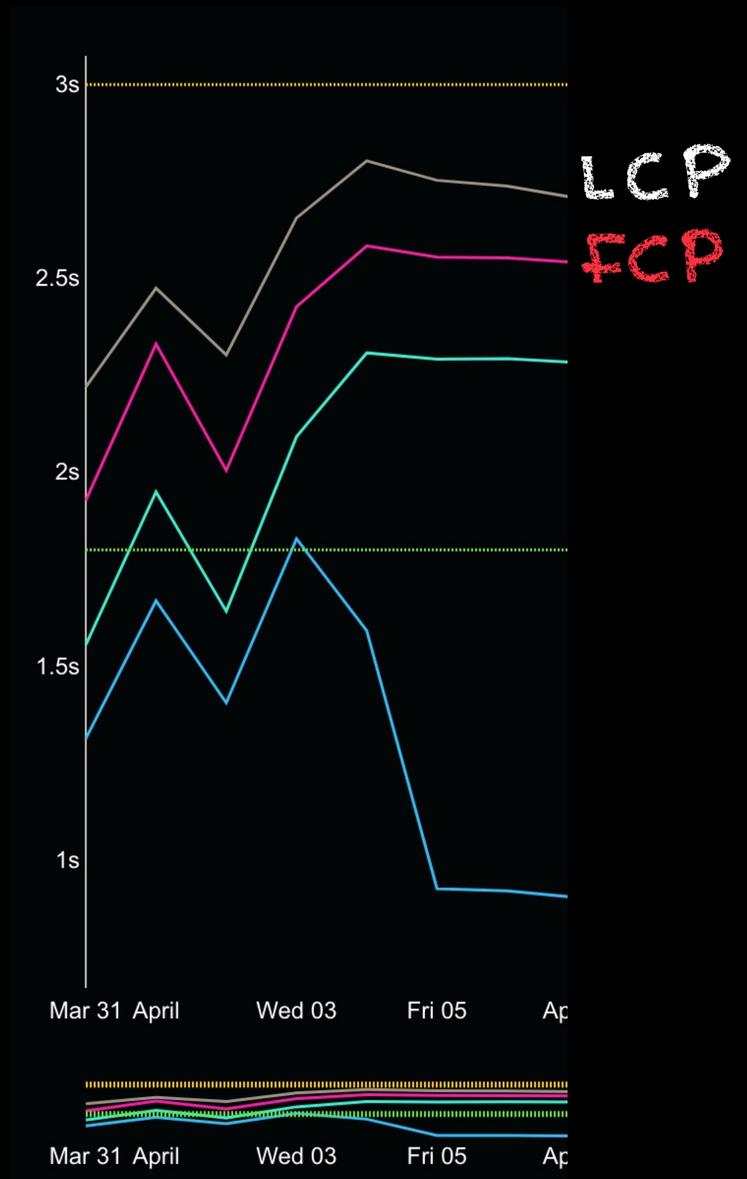

# Первое включение



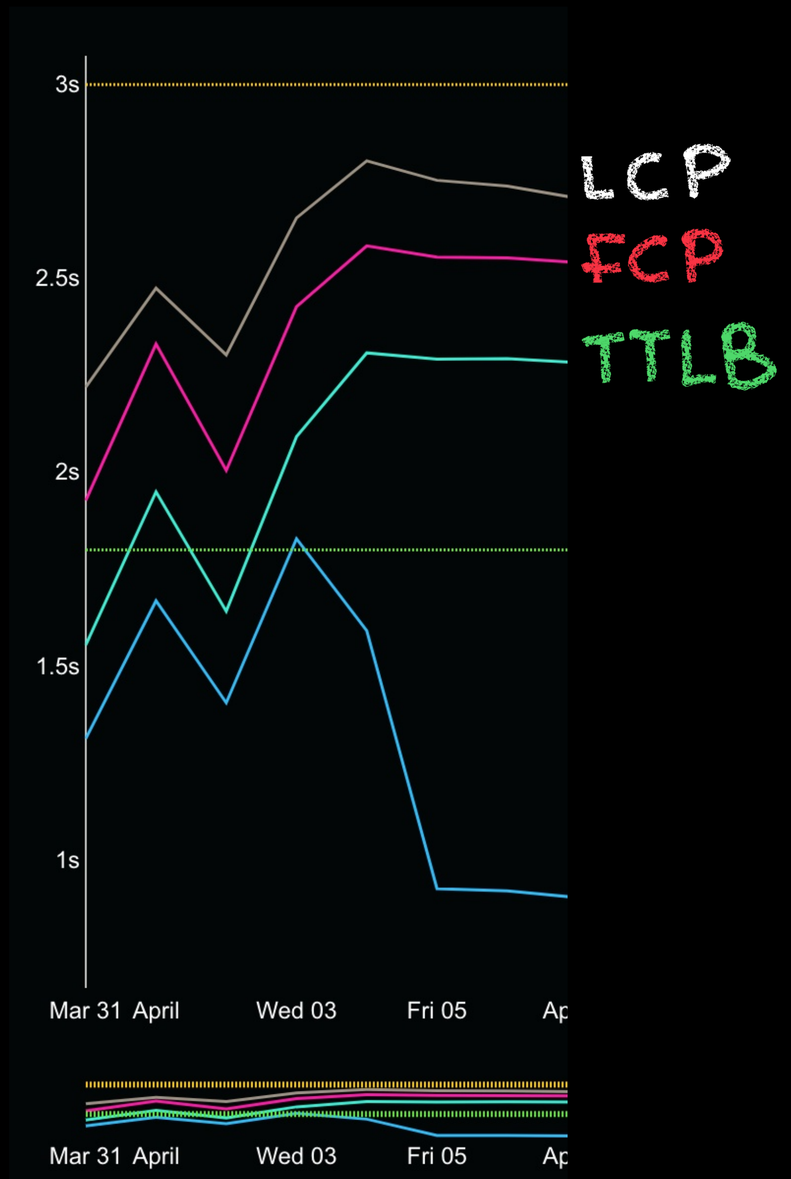

# Первое включение




# Первое включение

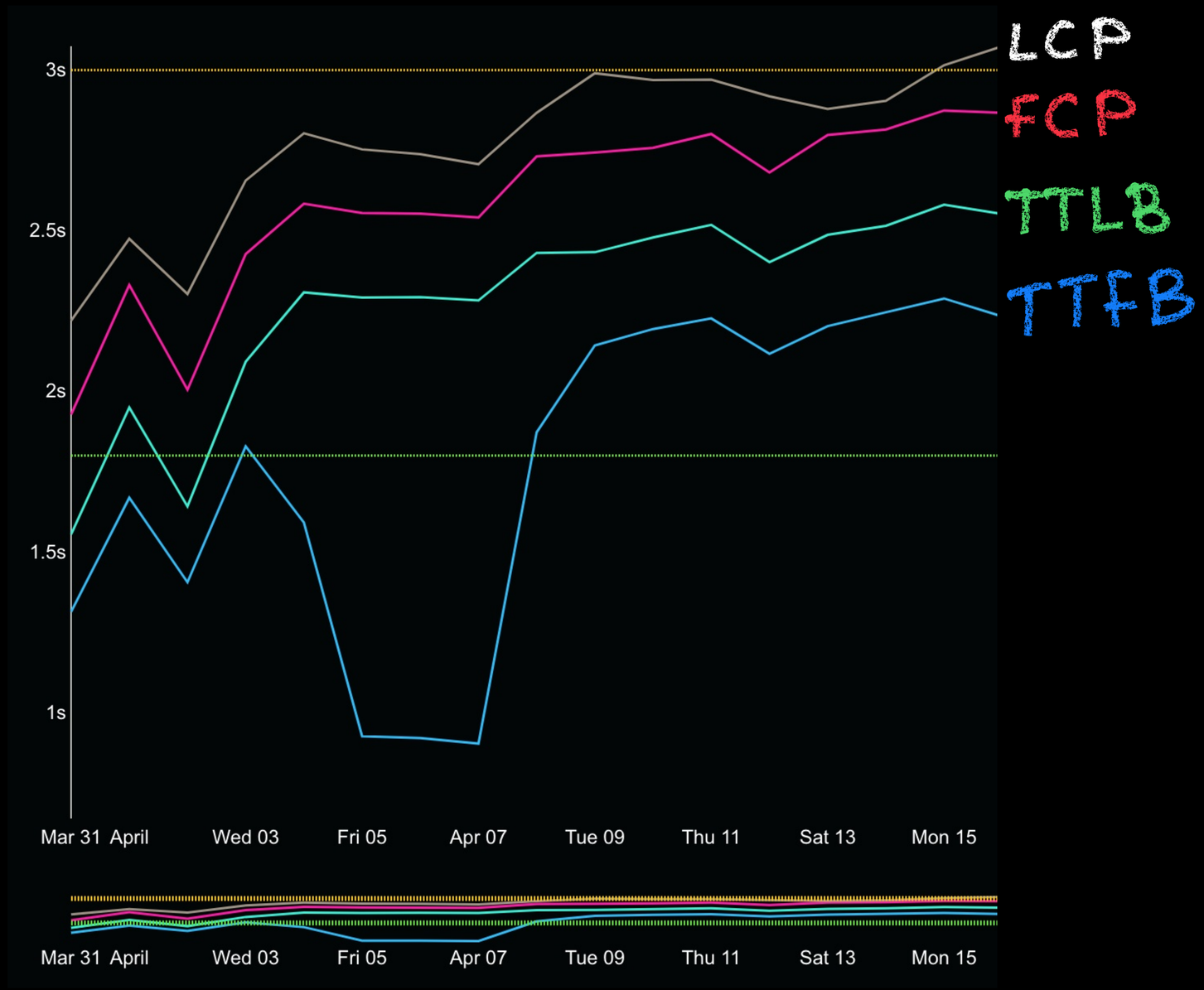



# Первое включение

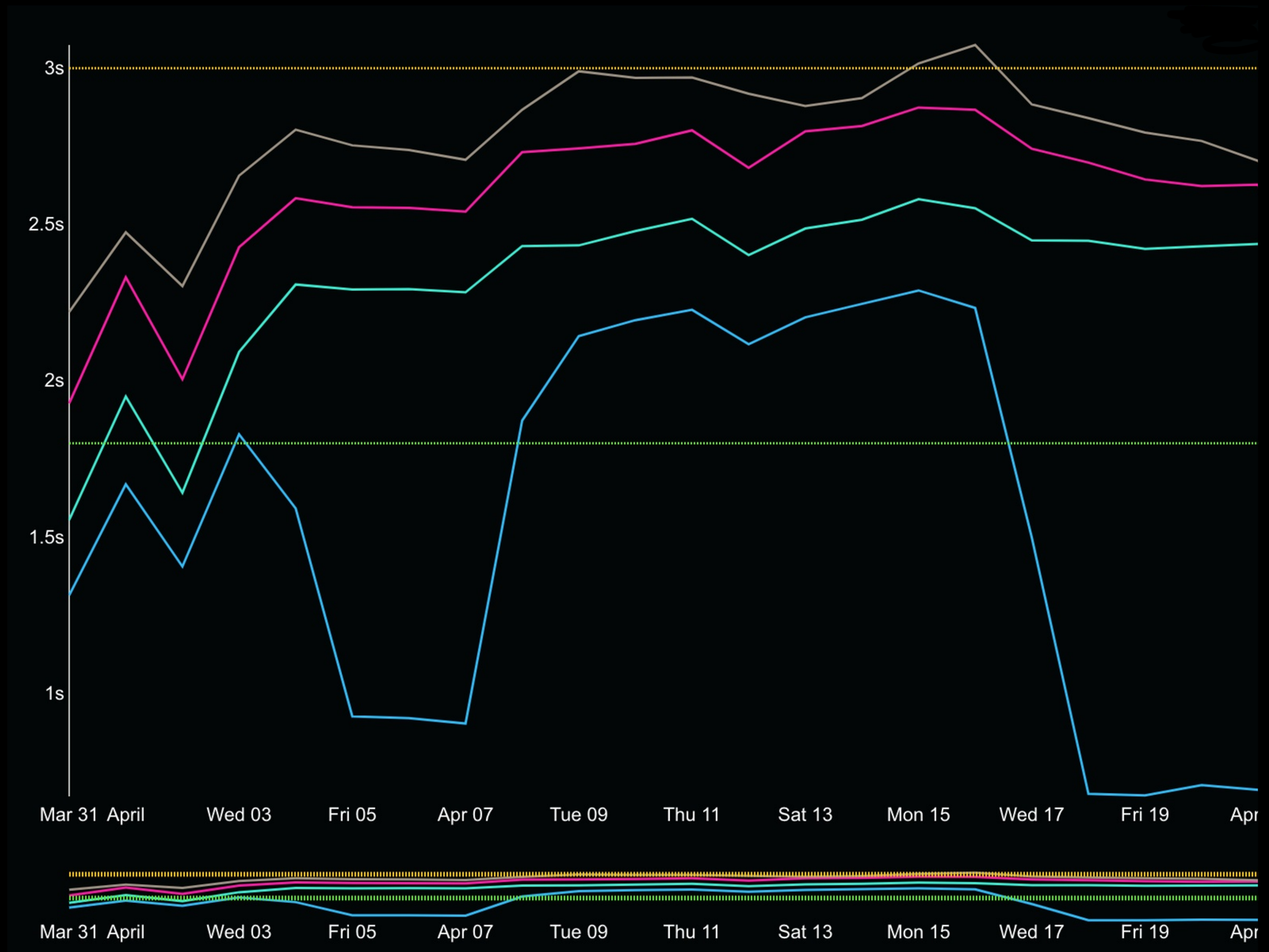



# Первое включение

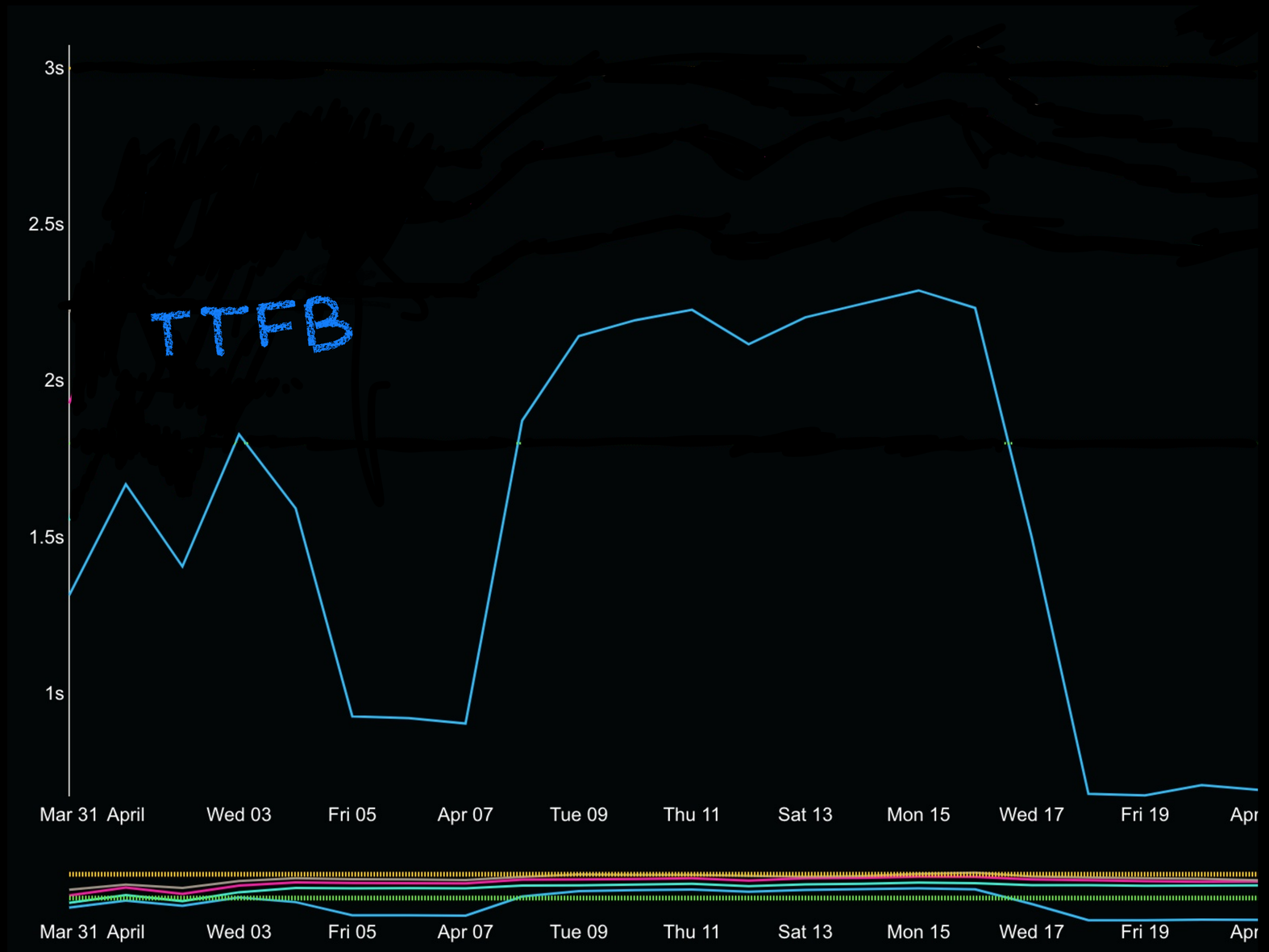

# Первое включение


LCP
FCP
TTLB
TTFB
3s
2.5s
2s
1.5s
1s
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15

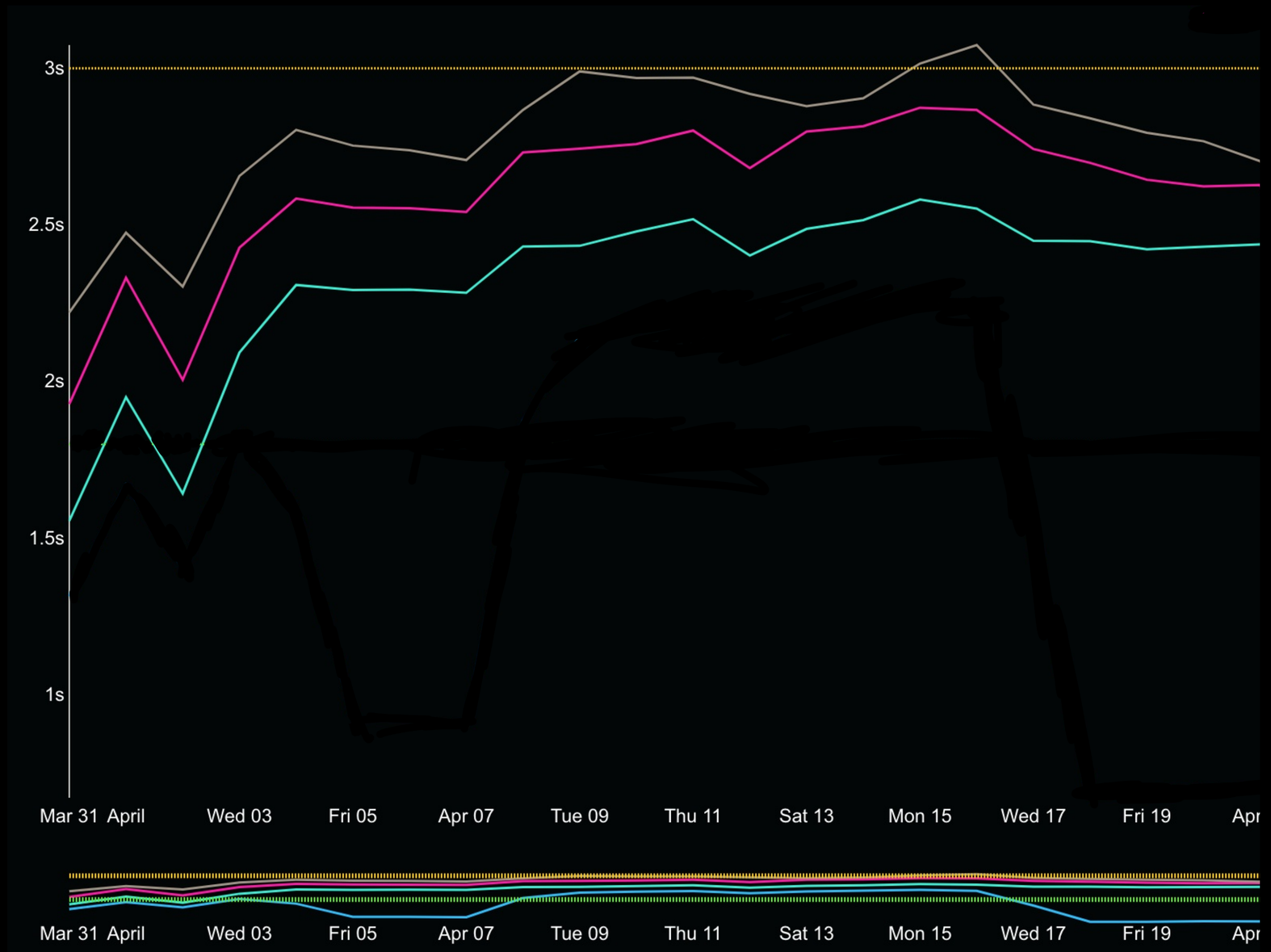
# Чиним


3s
2.5s
2s
1.5s
1s
Mar 31 April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr
Mar 31 April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr

# Итог

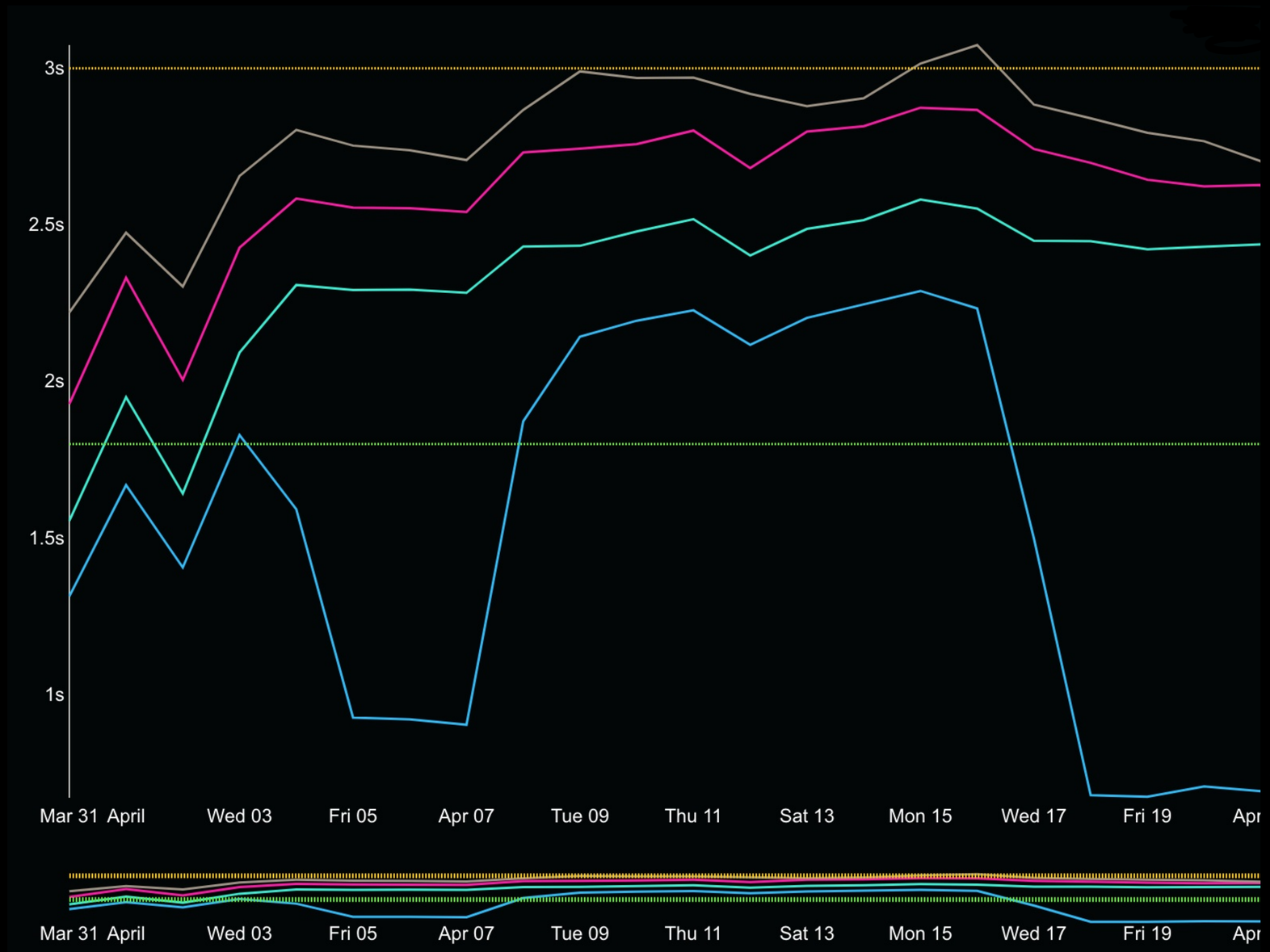

3s
2.5s
TTFB
2s
1.5s
1s
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr

# Итог




LCP
FCP
TTLB

# Итог


3s
2.5s
2s
1.5s
1s
Mar 31 April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
LCP
FCP
TTLB
TTLB

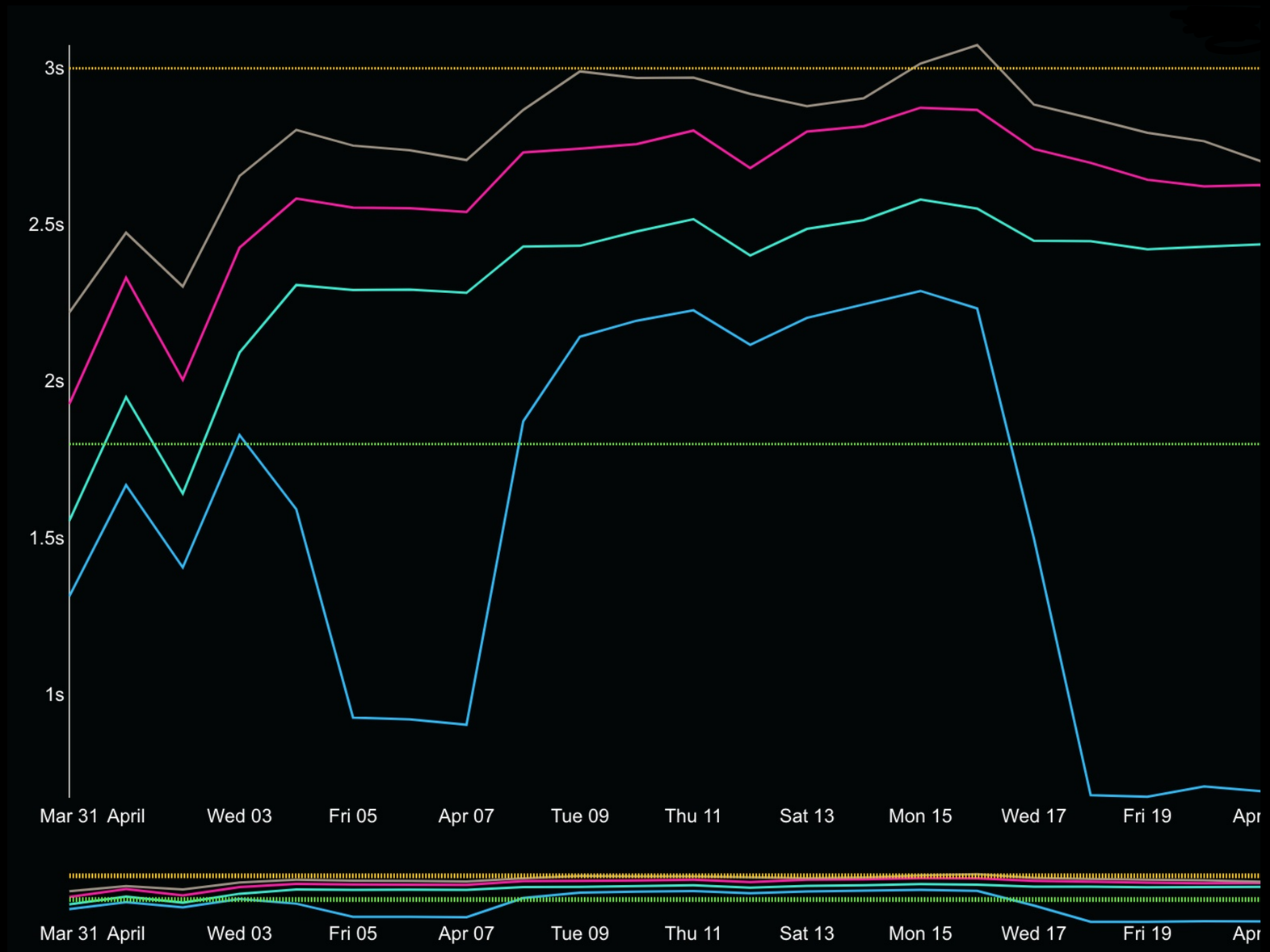
# Итог

мы получили положительный
результат, но есть куда стремиться

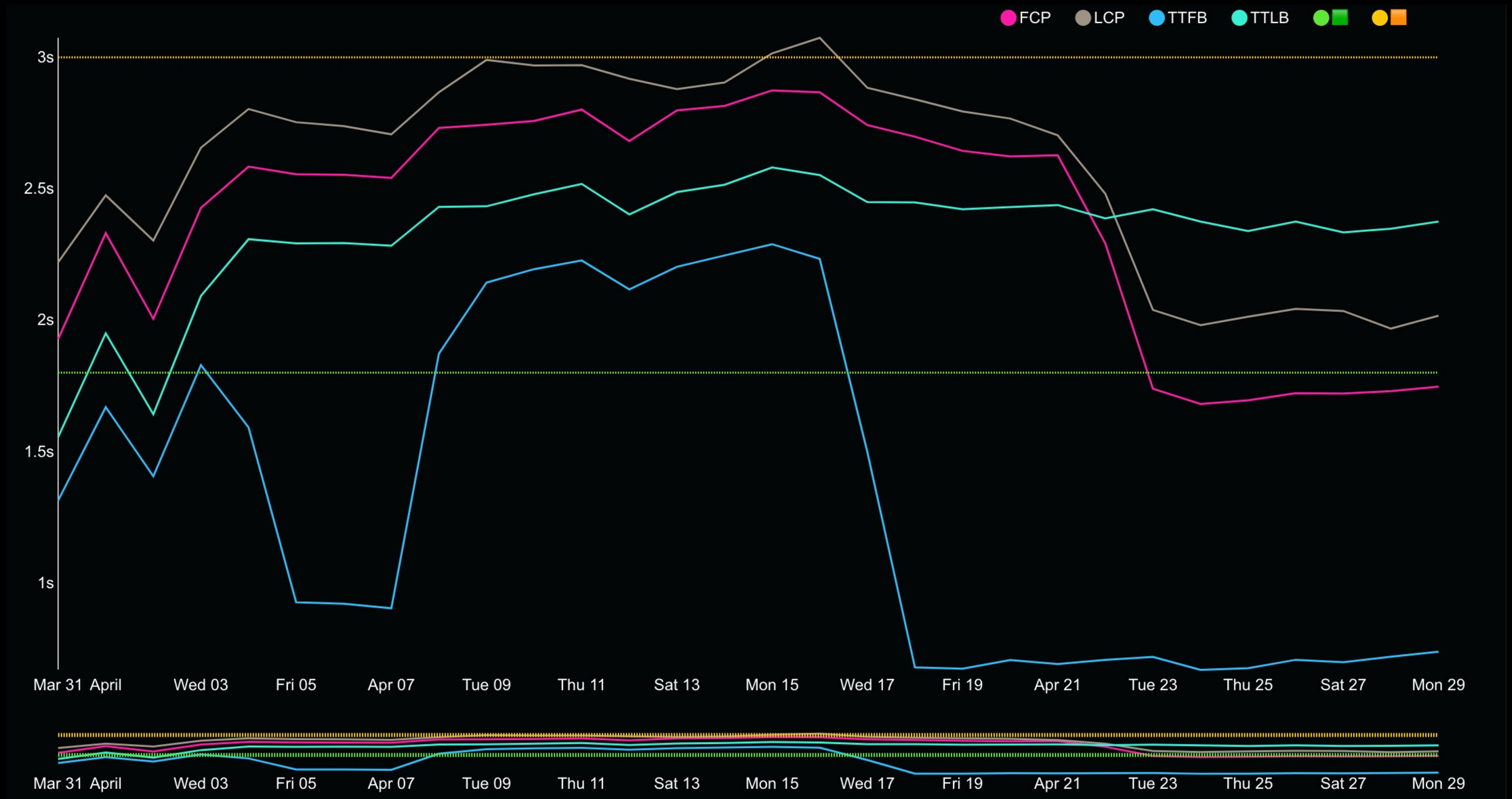
# идеи

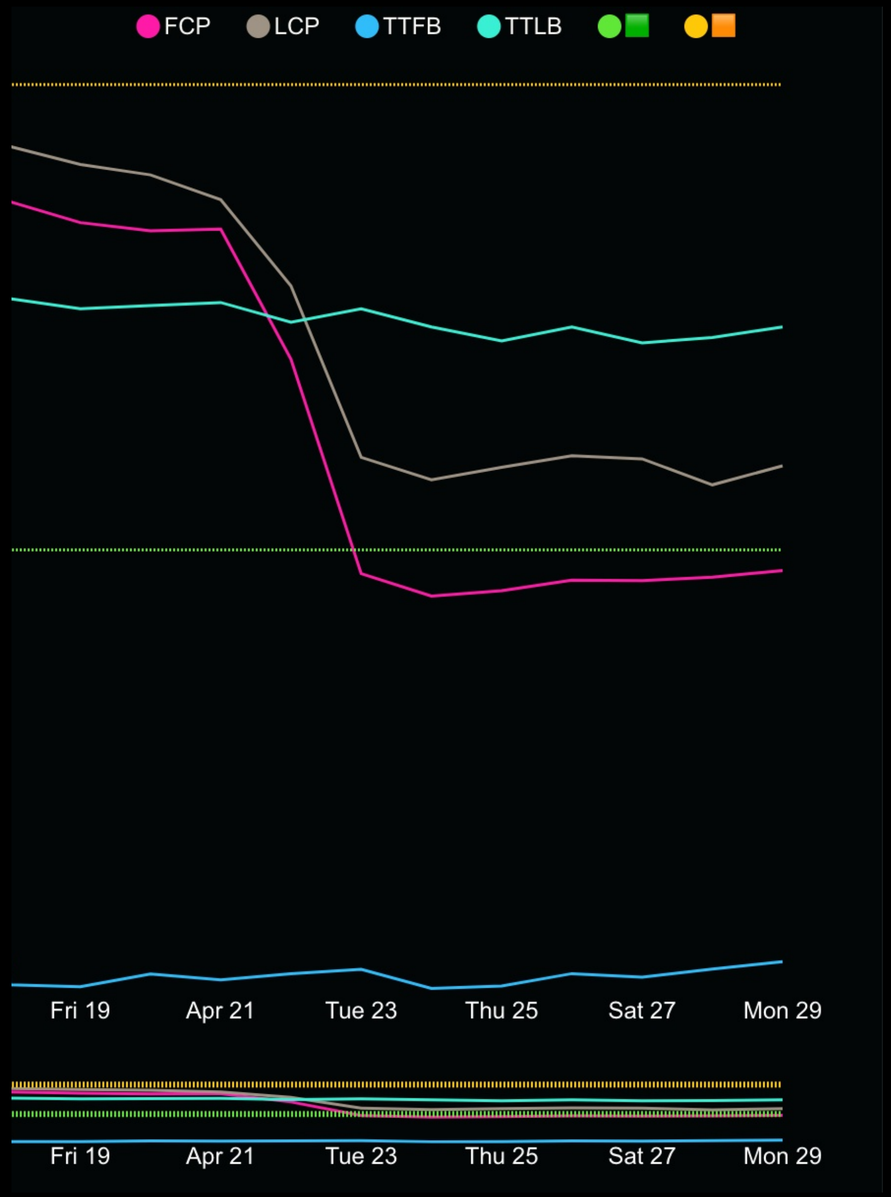
# Идеи

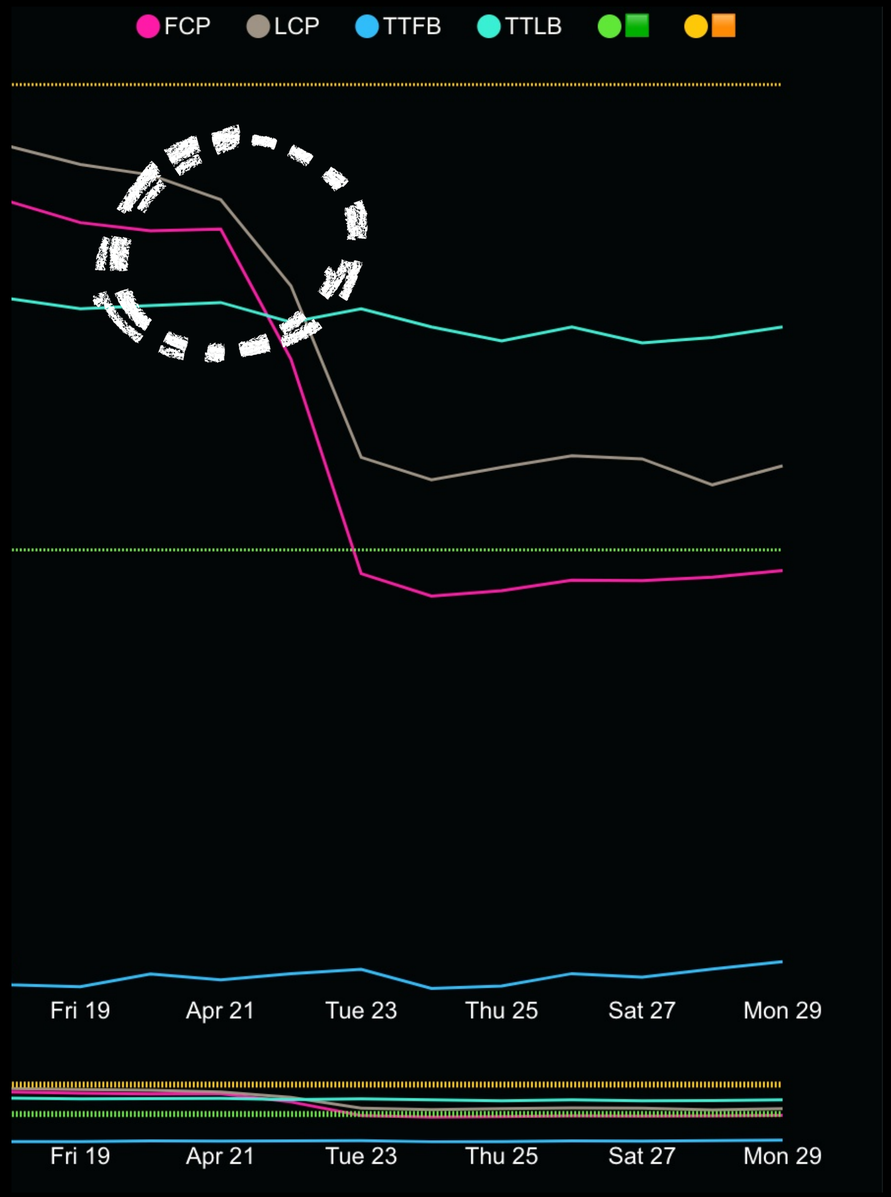
- Оторвать еще кусочки :)


3s
2.5s
2s
1.5s
1s
Mar 31 April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
LCP
FCP
TTLB
TTLB

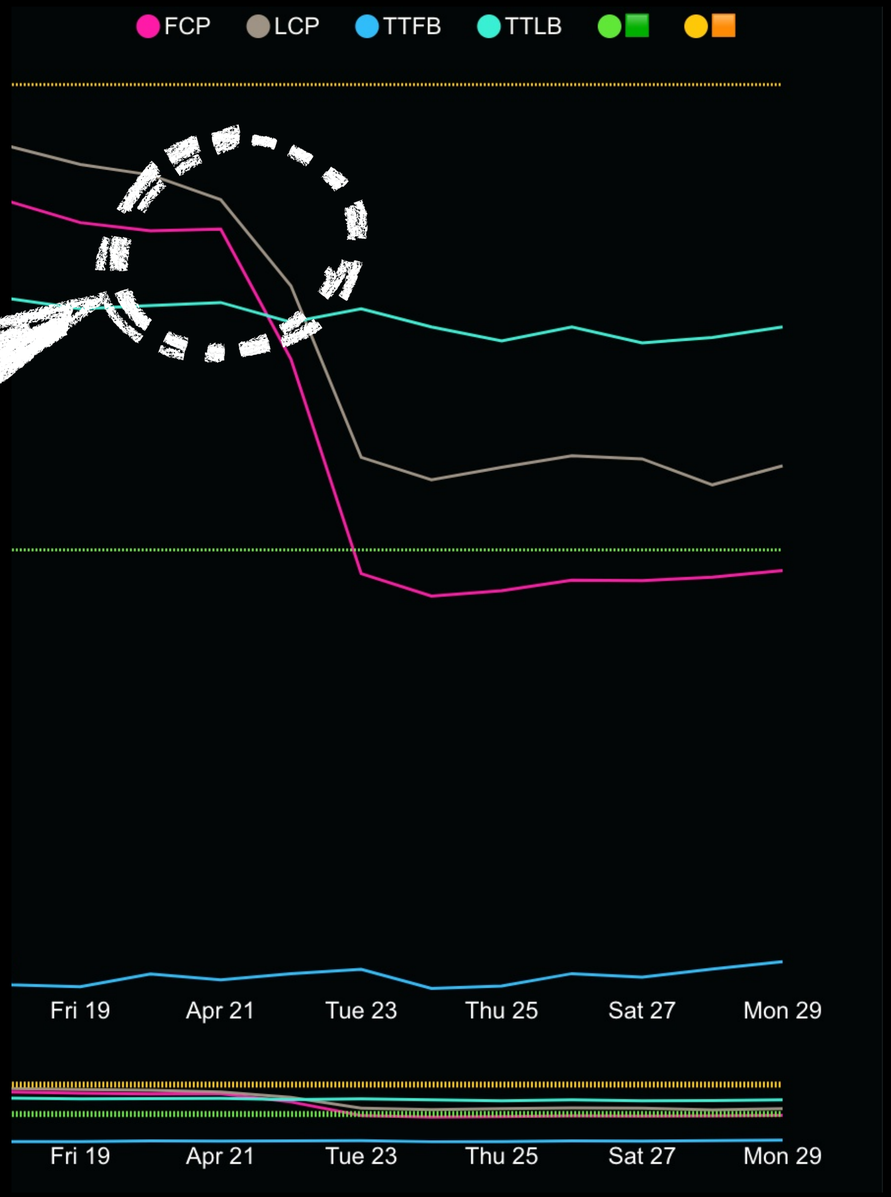
# Итог

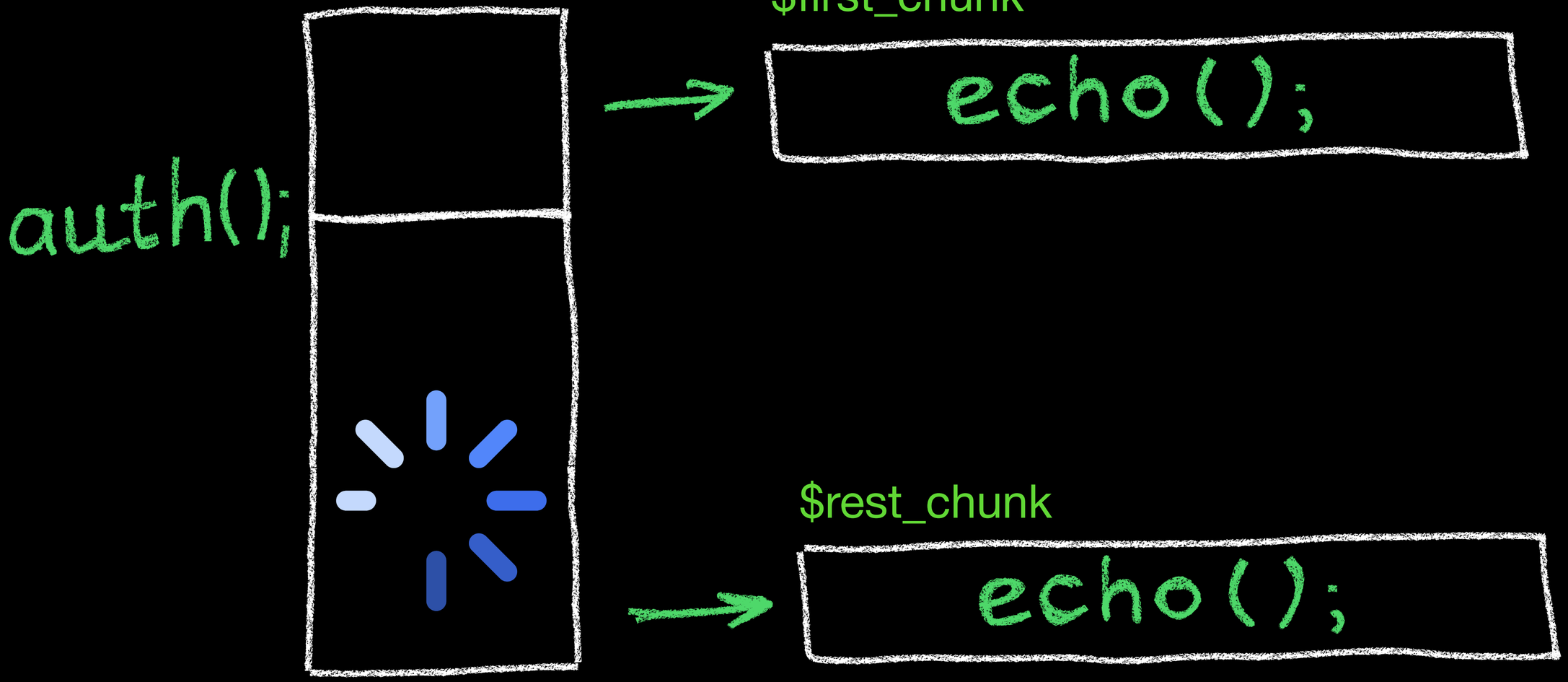

FCP
LCP
TTFB
TTLB
3s
2.5s
2s
1.5s
1s
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29
Mar 31
April
Wed 03
Fri 05
Apr 07
Tue 09
Thu 11
Sat 13
Mon 15
Wed 17
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29

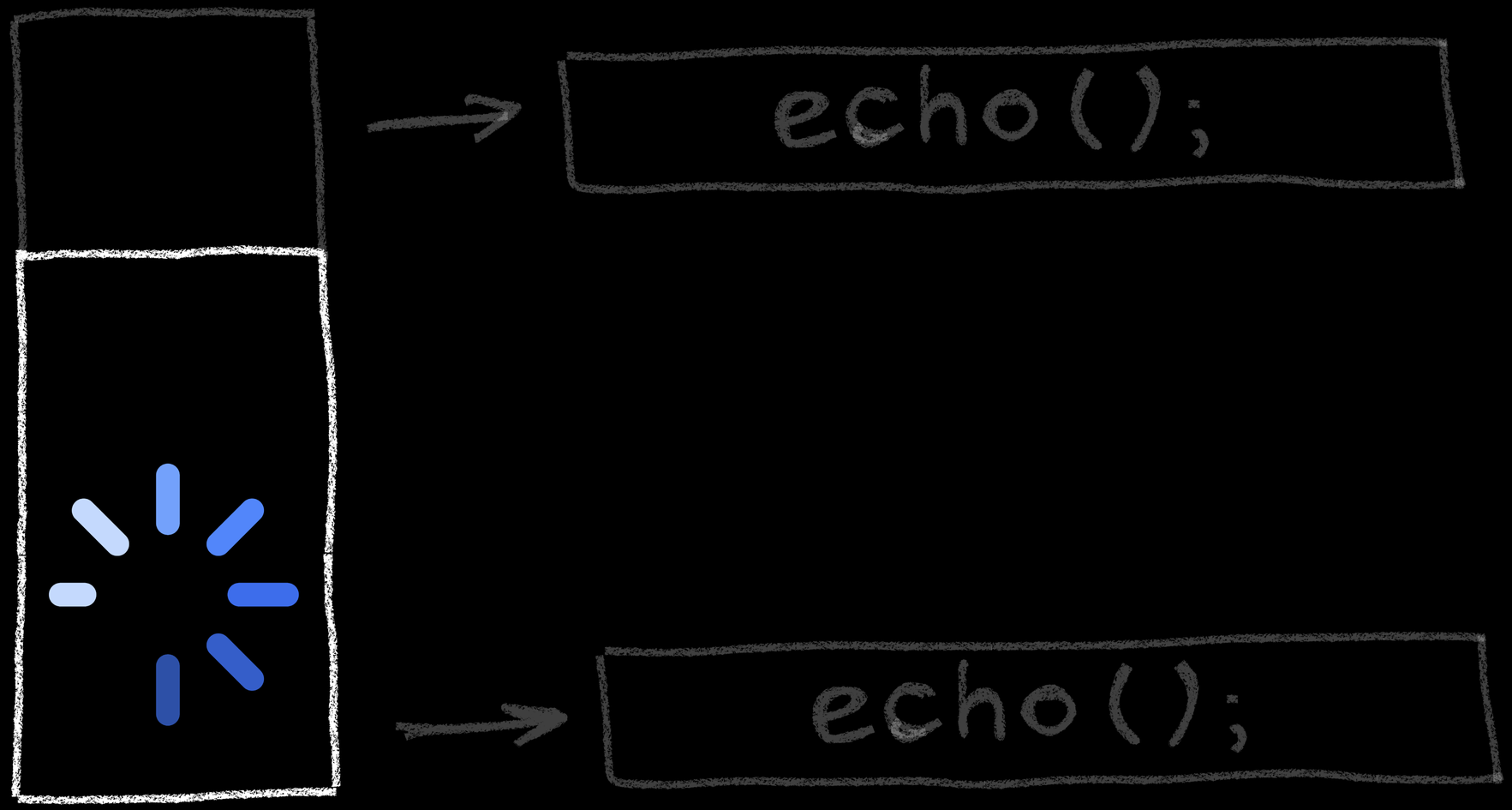

FCP
LCP
TTFB
TTLB
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29

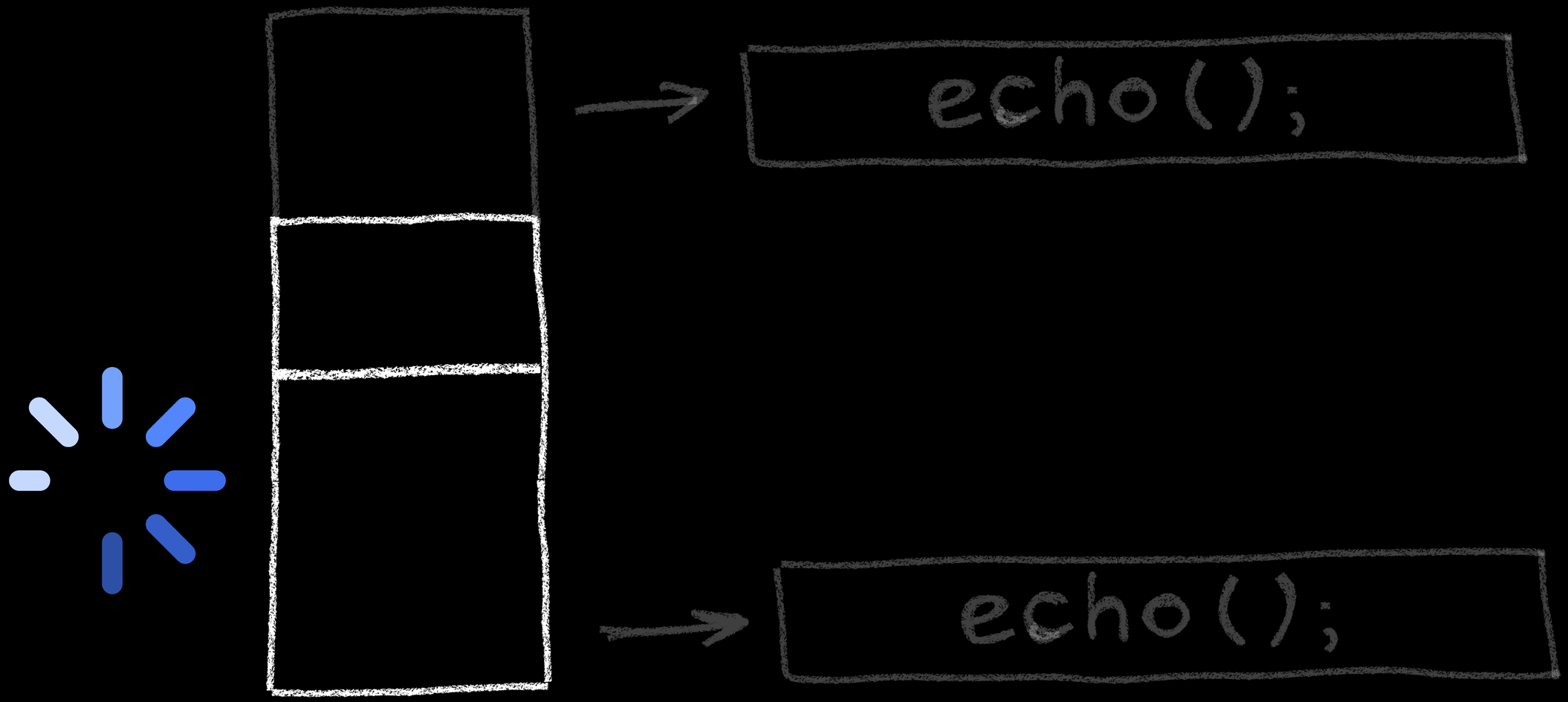

FCP
LCP
TTFB
TTLB
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29
Fri 19
Apr 21
Tue 23
Thu 25
Sat 27
Mon 29

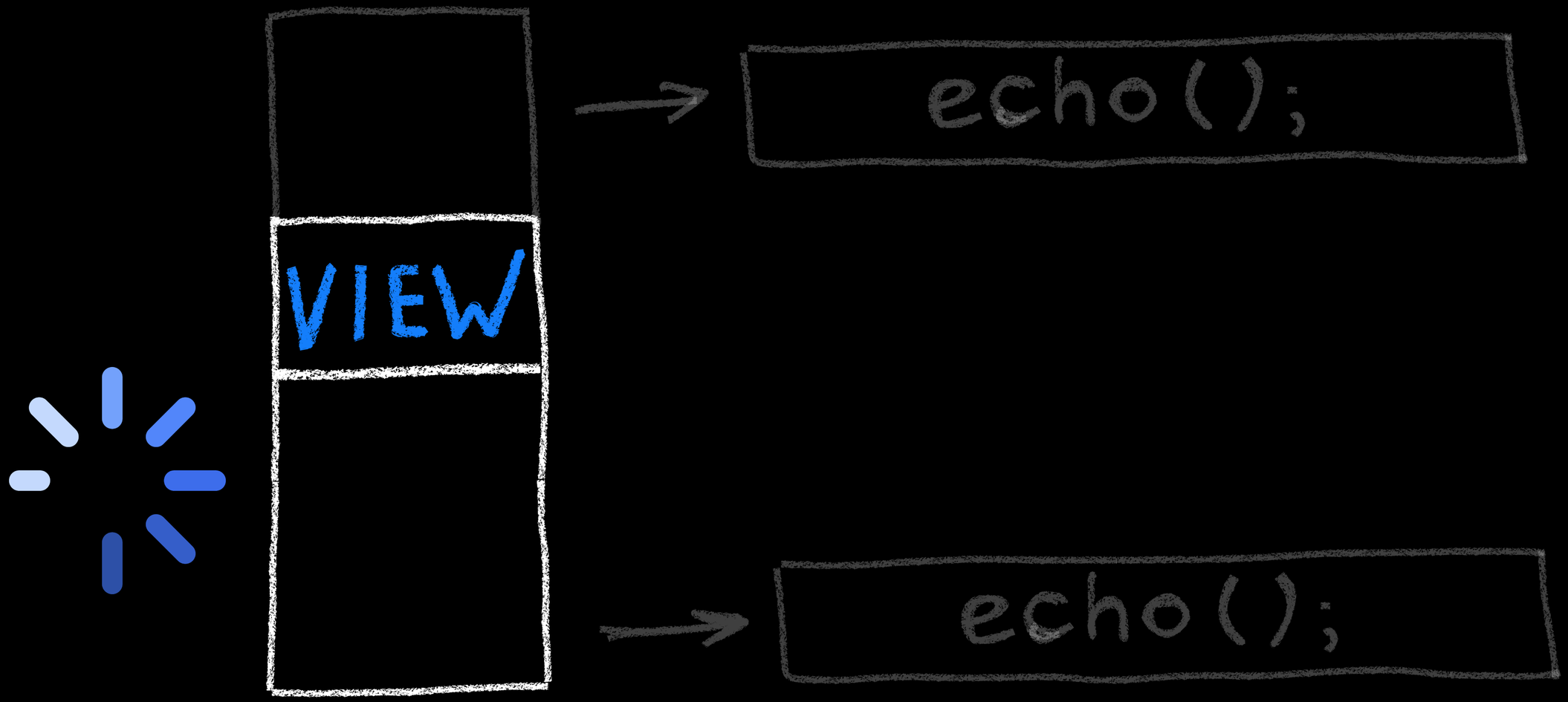
Что произошло 21 апреля?

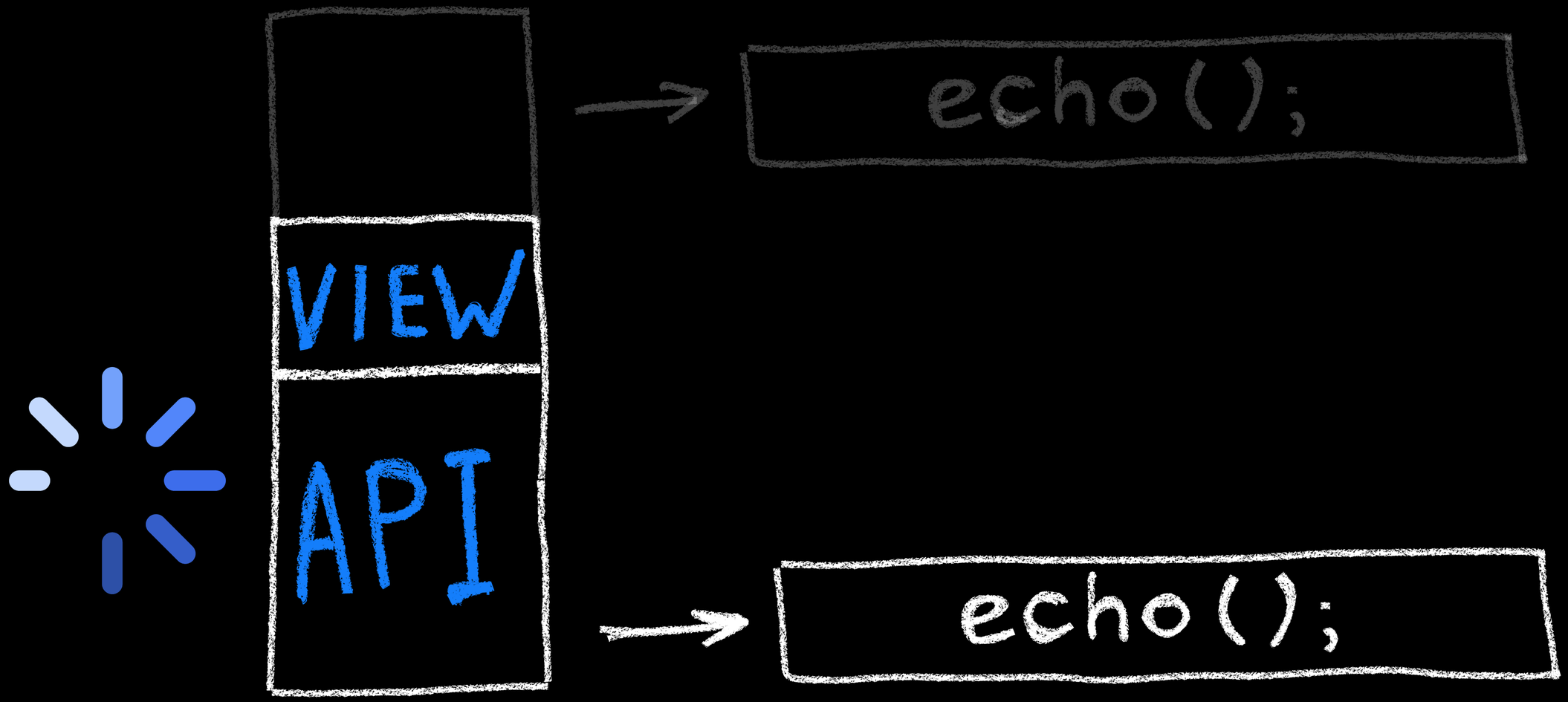


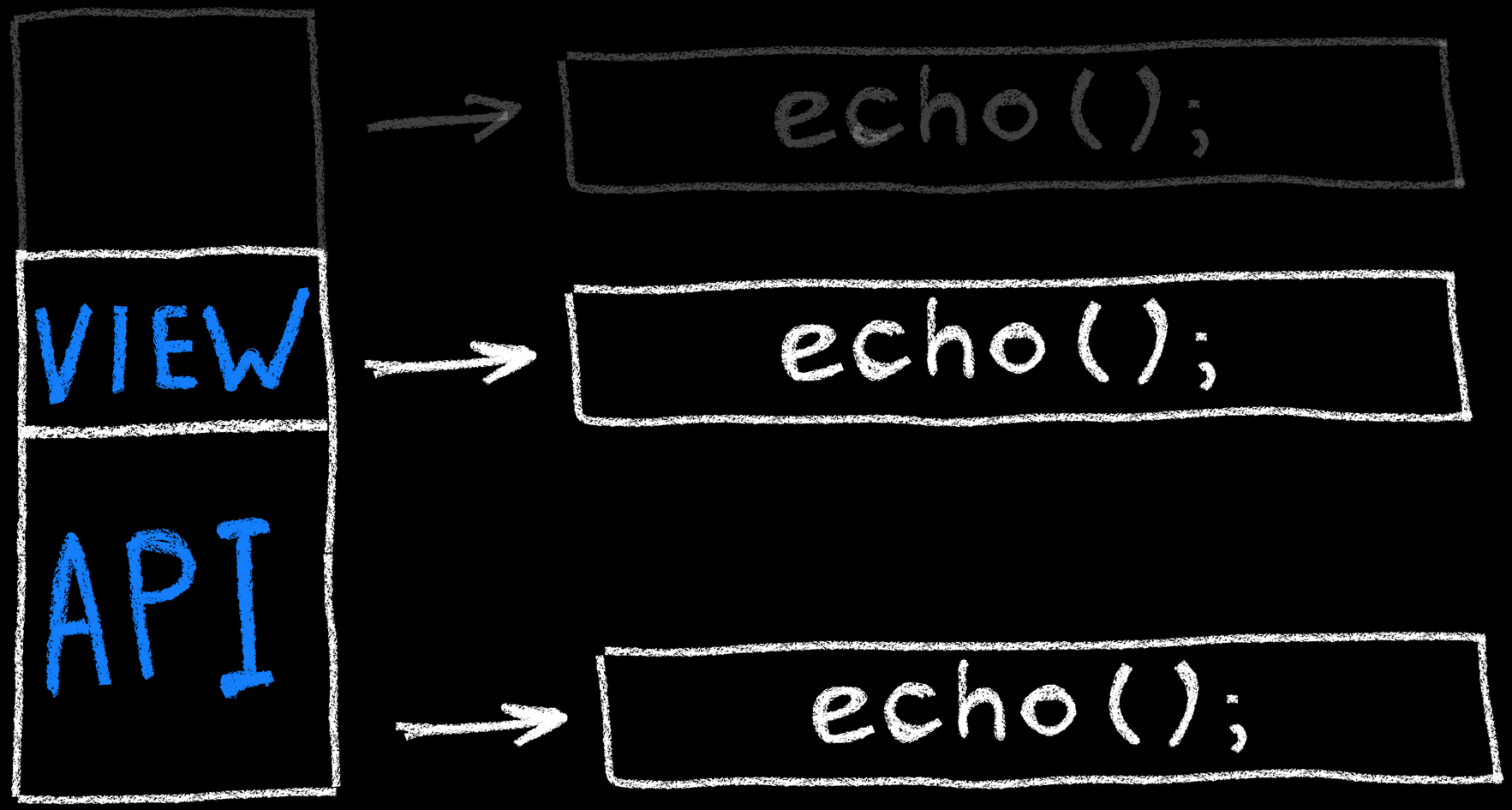

$first_chunk
auth();
echo();
$rest_chunk
echo();


echo();
echo();

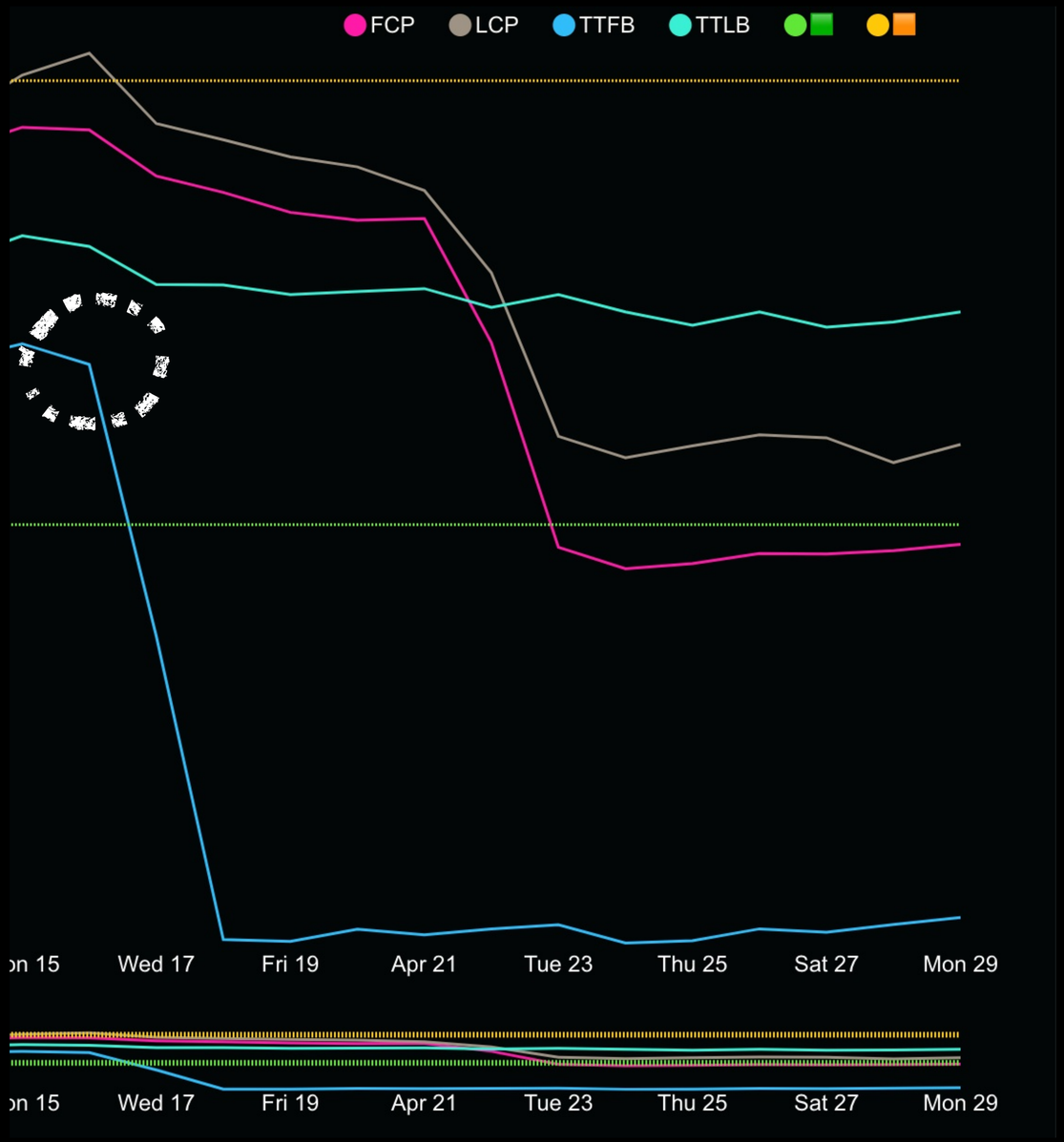

echo();
echo();


VIEW
echo();
echo();


echo();
VIEW
API
echo();

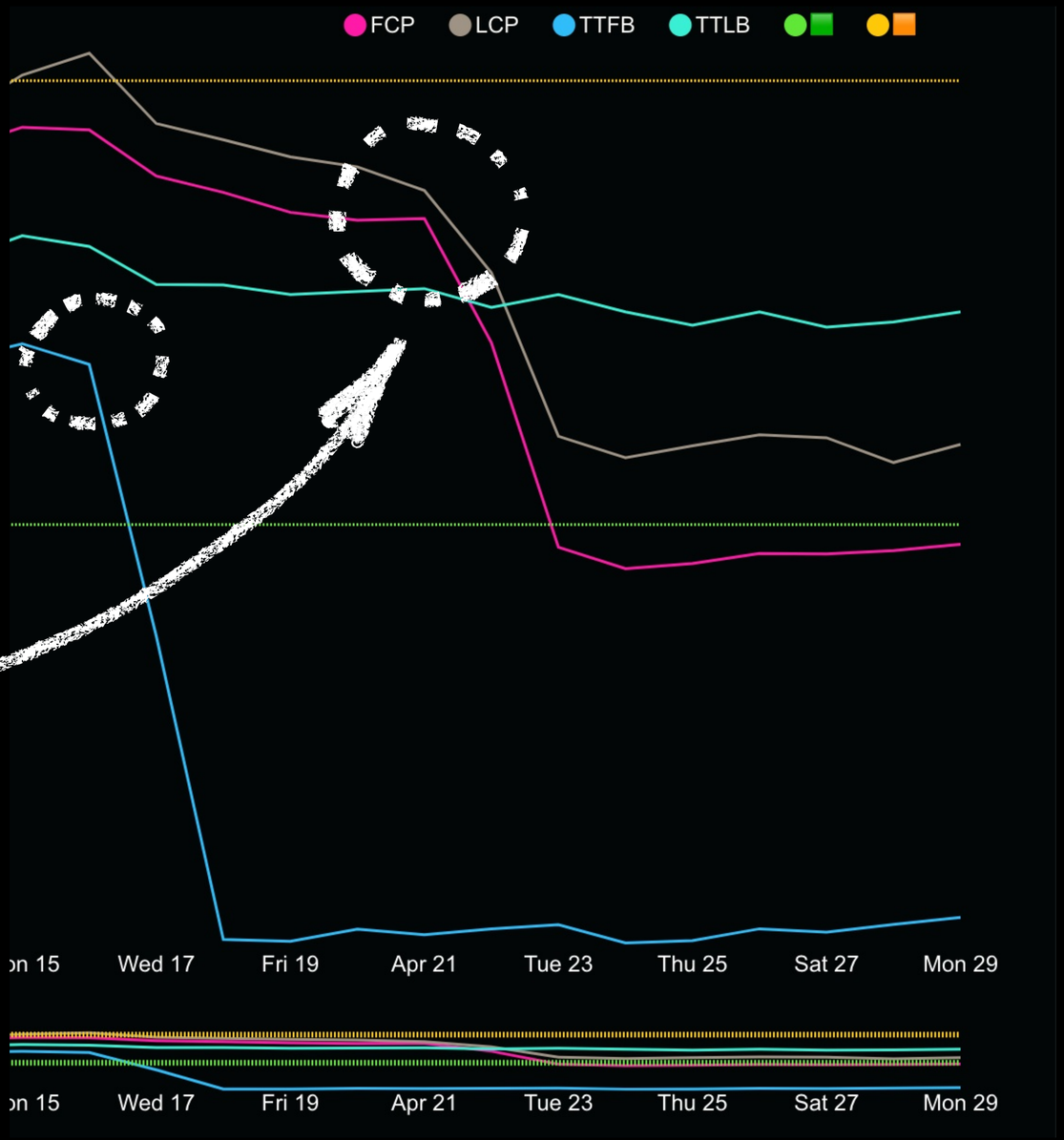

echo();
VIEW
echo();
API
echo();

Здесь начали качать
служебные скрипты



Здесь начали качать
служебные скрипты

Здесь перестали ждать
API



# Идеи

- Оторвать еще кусочки :)
- Отдавать API частями;

# Стриминговый execute

# Идеи

- Оторвать еще кусочки :)
- Отдавать API частями;
- Попробовать GraphQL + Relay.

# итог?

# Итог

- Transfer-encoding: chunked
  помогает показать
  пользователю контент быстрее,

# Итог

- Transfer-encoding: chunked
  помогает показать
  пользователю контент быстрее,
  если у вас HTTP/1.1;

# Итог

- Transfer-encoding: chunked
  помогает показать
  пользователю контент быстрее,
  если у вас HTTP/1.1;
- Свежие протоколы
  НЕ поддерживают chunked;

# Итог

- Transfer-encoding: chunked
  помогает показать
  пользователю контент быстрее,
  если у вас HTTP/1.1;
- Свежие протоколы
  НЕ поддерживают chunked;
- На чанки можно разбить
  и запросы, и страницу, и видео.

# Итог

- Transfer-encoding: chunked помогает показать пользователю контент быстрее, если у вас HTTP/1.1;
- Свежие протоколы НЕ поддерживают chunked;
- На чанки можно разбить и запросы, и страницу, и видео.
- не делайте преждевременных оптимизаций.

# спасибо за внимание!

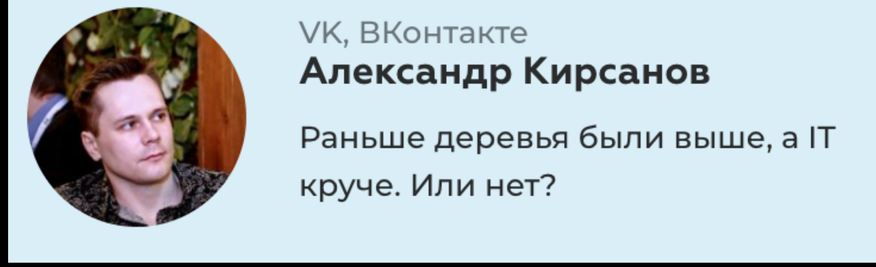
Фидбек и оценка доклада

Текст и ссылки!

# Фидбек и оценка доклада

# Текст и ссылки!

А еще можно сходить сюда:




26 мая, 17:00 (Новосиб)