



CSS in JS

ВАЛЕРИЙ Сорокобатько
v.sorokobatko@gmail.com



ВАЛЕРИЙ СОРОКОБАТЬКО

- **Create-react-app** контрибьютор
- автор **Awesome-create-react-app**
- автор **React Storybook README Addon**



CSS in JS



ПЛАН

- **В чем польза?**
- **Существующие решения**
- **Требования к библиотеке**
- **Что используем мы**
- **Нюансы в реализации CSS in JS подхода**



В чем польза?



Удобство разработки

Удобство разработки



Andrey Okonetchnikov
@okonetchnikov

*“Investing into **#DX** you’ll get a better **#UX** in the result”*



Удобство разработки

Не нужно думать о подключении CSS файлов

Достаточно подключить только JS модуль





Удобство разработки

Особенно удобно при шаринге коде





Удобство разработки

The screenshot shows a web browser window with the URL `https://tuchk4.github.io/storybook-readme/?selectedKind=Button&selectedStory=Default&full=0&down=1&left=1&panelRight=1&downPan...`. The browser title is "React Storybook" and the user name "Valeriy" is visible in the top right corner.

The main content area displays a "Hello Button" component. To the left, there is a "README ADDON" sidebar with a "Filter" input field and a list of categories: "Button", "Default", "Colored", and "Header".

To the right, there is a "README" sidebar with the following sections:

- Button component**
- Usage**

```
import Button from 'components/button';
```
- Properties**
 - `onClick` - click callback
 - `label` - button text
- | propName | propType | defaultValue | isRequired |
|----------|----------|--------------|------------|
| onClick | func | - | |
| label | string | - | + |
- Roadmap**
- Icons**

```
import Button from 'components/button';  
  
render() {
```



Удобство разработки

Более мощный CSS



Удобство разработки

Более мощный CSS

```
import color from 'color'  
const red = Color('red')  
  
const styles = {  
  color: red.lighten(10).toHex()  
}
```



Удобство разработки

Более мощный CSS

```
const styles = {  
  width: window.innerWidth  
}
```



Удобство разработки

Более мощный CSS

`.col-1 .col-2 .col-3 .col-4 ...`

`.col-md-1 .col-md-2 .col-md-3 .col-md-4 ...`

`.col-xs-1 .col-xs-2 .col-xs-3 .col-xs-4 ...`



Удобство разработки

Более мощный CSS

```
.col-1 .col-2 .col-3 .col-4 ...  
.col-md-1 .col-md-2 .col-md-3 .col-md-4 ...  
.col-xs-1 .col-xs-2 .col-xs-3 .col-xs-4 ...
```

```
const css = props => ({  
  width: `12 / ${props.size}%`  
})
```



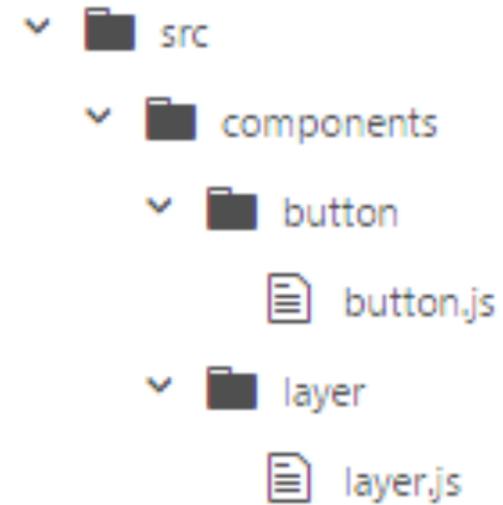
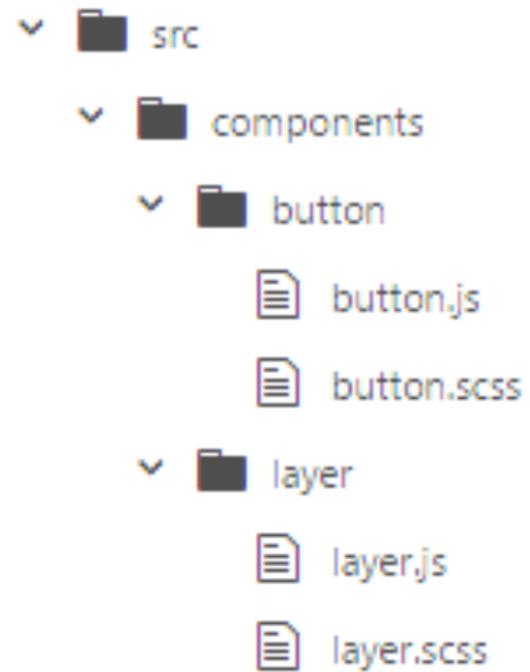
Удобство разработки

Меньше код и файлов



Удобство разработки

Меньше код и файлов





Удобство разработки

Не повторяйте себя
(DRY)



Удобство разработки

Не повторяйте себя

```
const ANIMATION_DURATION = 2.5

const styles = {
  animation: `${ANIMATION_DURATION}s pulse`
};
```



Удобство разработки

Не повторяйте себя

```
const ANIMATION_DURATION = 2.5
```

```
const styles = {  
  animation: `${ANIMATION_DURATION}s pulse`  
};
```

```
mountElement()  
setTimeout(() => {  
  unmountElement()  
}, ANIMATION_DURATION * 1000)
```



Удобство разработки

ЧИСТЫЙ КОД



Удобство разработки

~~Чистый код~~

```
import styles from './button.css'
import Button from './button'

const MyButton = ({ primary, raised, large, children, ...props }) => {
  const classList = classNames(styles.button, {
    [styles.primary]: primary,
    [styles.large]: large,
    [styles.raised]: raised,
  })

  return <Button className={classList} {...props}>{children}</Button>
}
```



Удобство разработки

ЧИСТЫЙ КОД

```
import Button from './button'  
  
const MyButton = injectStyles({  
  color: props => props.primary ? 'blue' : 'black',  
  fontSize: props => props.large ? '2em' : '1em',  
  boxShadow: props => props.raised ? '1px 1px #ccc' : 'none',  
})(Button)
```



Удобство разработки

Без конфигов



Удобство разработки

Без конфигов

CSS in JS – это нативный JS



Удобство разработки

Без конфигов

CSS in JS – это нативный JS

- Не нужны лоадеры и плагины
- Не нужно настраивать webpack и babel
- Проще запускать тесты

Удобство разработки

lodash для CSS in JS





Inline styles или CSS?

Проблемы inline styles

```
<div style="padding-bottom: 8px;">
  ▼ <div style="padding: 16px; font-weight: 500; box-sizing: border-box; position: relative; white-space: nowrap;
  cursor: pointer;">
    ▼ <div style="display: inline-block; vertical-align: top; white-space: normal; padding-right: 90px;">
      ▶ <span style="color: rgba(0, 0, 0, 0.87); display: block; font-size: 15px;">...</span>
      ▶ <span style="color: rgba(0, 0, 0, 0.54); display: block; font-size: 14px;">...</span>
    </div>
    ▼ <button tabindex="0" type="button" style="border: 10px; box-sizing: border-box; display: inline-block; font-
    family: Roboto, sans-serif; -webkit-tap-highlight-color: rgba(0, 0, 0, 0); cursor: pointer; text-decoration:
    none; margin: auto; padding: 12px; outline: none; font-size: 0px; font-weight: inherit; position: absolute; z-
    index: 1; overflow: visible; transition: all 450ms cubic-bezier(0.23, 1, 0.32, 1) 0ms; width: 48px; height:
    48px; top: 0px; bottom: 0px; right: 4px; background: none;">
      ▶ <div>_</div>
    </button>
  </div>
  ▼ <div style="padding: 16px; font-size: 14px; color: rgba(0, 0, 0, 0.87);">
    ▶ <div>...</div>
    ▼ <button tabindex="0" type="button" style="border: 10px; box-sizing: border-box; display: inline-block; font-
    family: Roboto, sans-serif; -webkit-tap-highlight-color: rgba(0, 0, 0, 0); cursor: pointer; text-decoration:
    none; margin: 0px; padding: 0px; outline: none; font-size: inherit; font-weight: inherit; position: relative;
    z-index: 1; height: 36px; line-height: 36px; min-width: 88px; color: rgba(0, 0, 0, 0.87); transition: all
    450ms cubic-bezier(0.23, 1, 0.32, 1) 0ms; border-radius: 2px; user-select: none; overflow: hidden; background-
    color: rgba(64, 168, 45, 0.21); text-align: center;">
```



Проблемы inline styles

- Невозможно менять стили для всех
КОМПОНЕНТОВ ОДНОГО ТИПА



Проблемы inline styles

- Невозможно менять стили для всех компонентов одного типа
- Нет стандартной поддержки псевдоклассов



Проблемы inline styles

- Невозможно менять стили для всех компонентов одного типа
- Нет стандартной поддержки псевдоклассов
- Много «лишнего» кода

Поговорим о CSS in JS



Проблемы CSS in JS



Проблемы CSS in JS

- Дополнительные операции в рантайме



Проблемы CSS in JS

- Дополнительные операции в рантайме
- Плохая интеграция с IDE



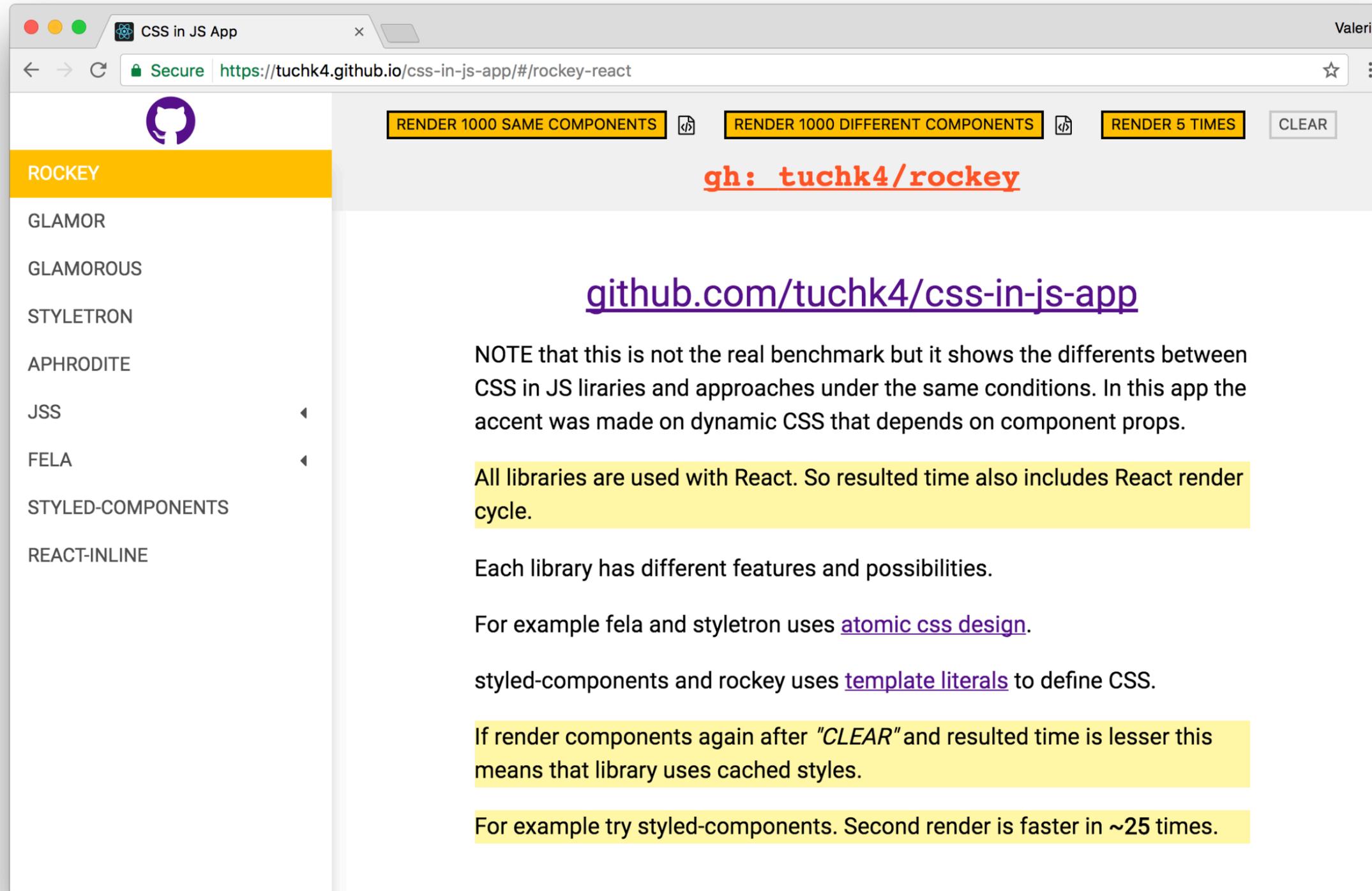
Проблемы CSS in JS

- Дополнительные операции в рантайме
- Плохая интеграция с IDE
- Дополнительная работа с DOM



Существующие решения

Пример



The screenshot shows a web browser window with the URL `https://tuchk4.github.io/css-in-js-app/#/rockey-react`. The page features a sidebar on the left with a list of CSS-in-JS libraries: ROCKEY (highlighted in orange), GLAMOR, GLAMOROUS, STYLETRON, APHRODITE, JSS, FELA, STYLED-COMPONENTS, and REACT-INLINE. The main content area has three buttons: "RENDER 1000 SAME COMPONENTS", "RENDER 1000 DIFFERENT COMPONENTS", and "RENDER 5 TIMES", along with a "CLEAR" button. Below the buttons, the text "gh: [tuchk4/rockey](https://github.com/tuchk4/rockey)" is displayed. A link to github.com/tuchk4/css-in-js-app is also present. The main text includes a note about the benchmark's purpose and several highlighted statements: "All libraries are used with React. So resulted time also includes React render cycle.", "Each library has different features and possibilities.", "For example fela and styletron uses [atomic css design](#).", "styled-components and rockey uses [template literals](#) to define CSS.", "If render components again after 'CLEAR' and resulted time is lesser this means that library uses cached styles.", and "For example try styled-components. Second render is faster in ~25 times."

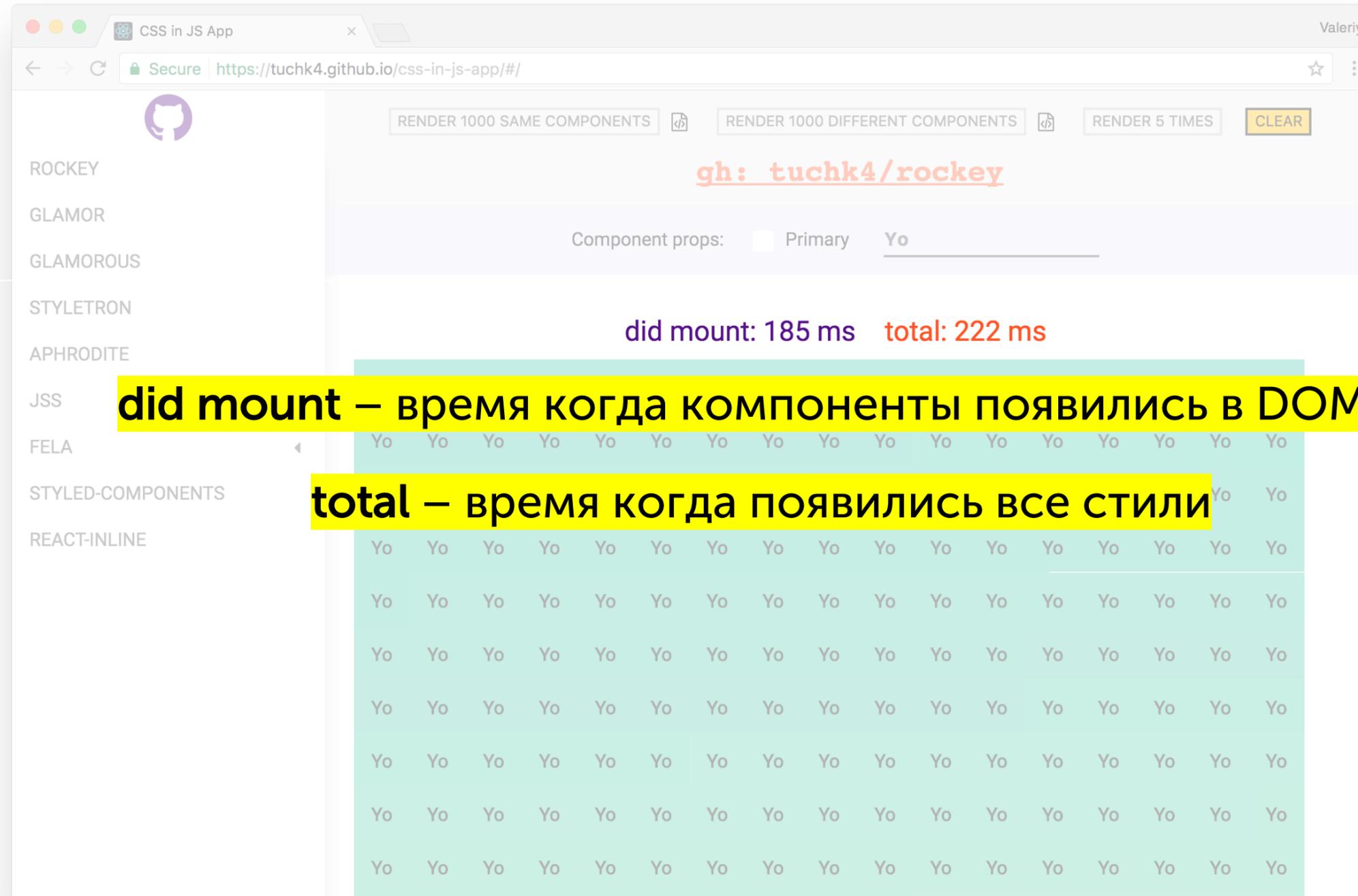


Пример

- Рендерит 4000 компонентов
- С акцентом на динамический CSS

```
{  
  minWidth: '16px',  
  height: '16px',  
  display: 'inline-block',  
  backgroundColor: props => colors[props.i]  
}
```

Пример



RENDR 1000 SAME COMPONENTS | RENDR 1000 DIFFERENT COMPONENTS | RENDR 5 TIMES | CLEAR

gh: tuchk4/rockey

Component props: Primary Yo

did mount: 185 ms total: 222 ms

did mount – время когда компоненты появились в DOM

total – время когда появились все стили

Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

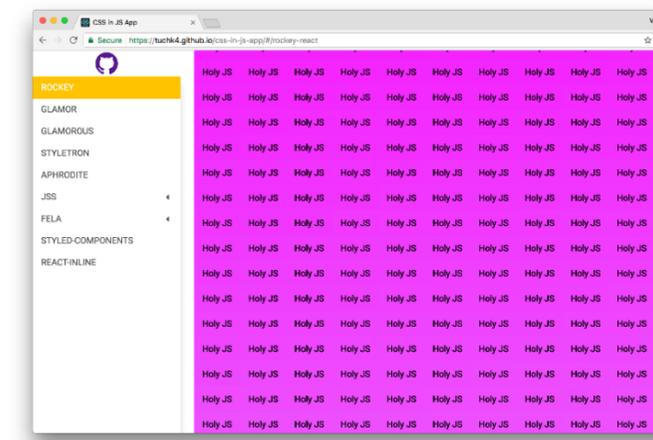
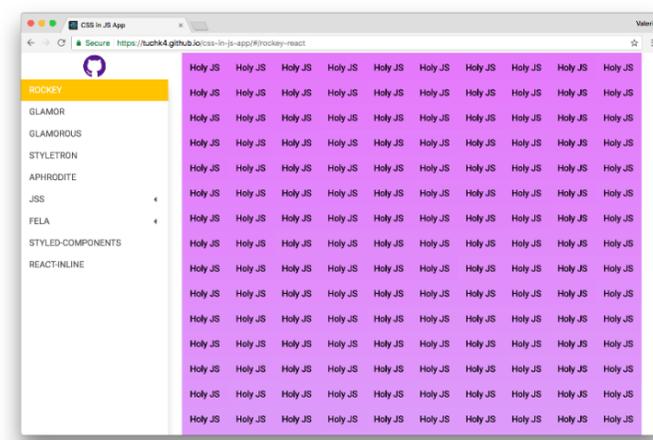
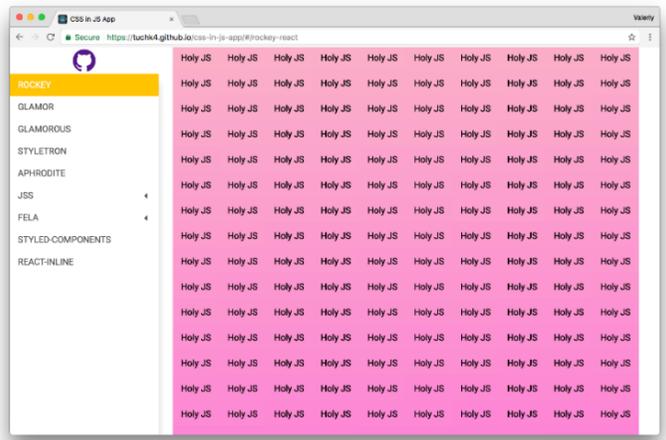
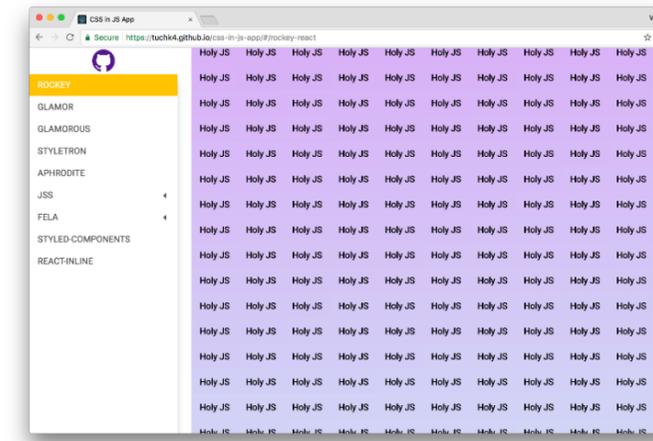
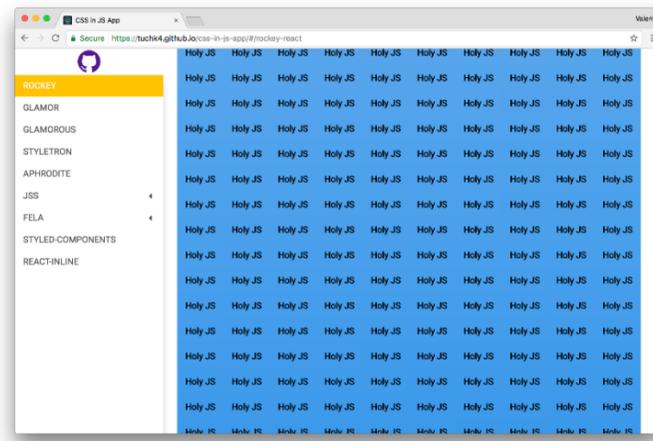
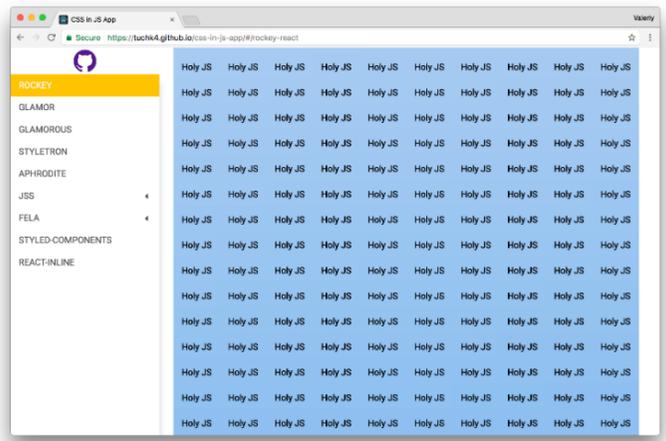
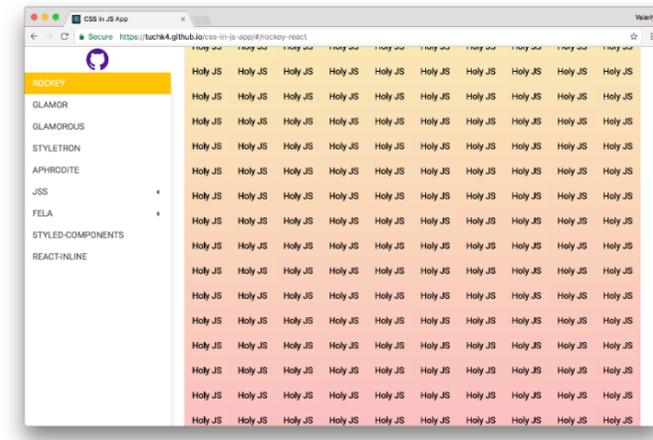
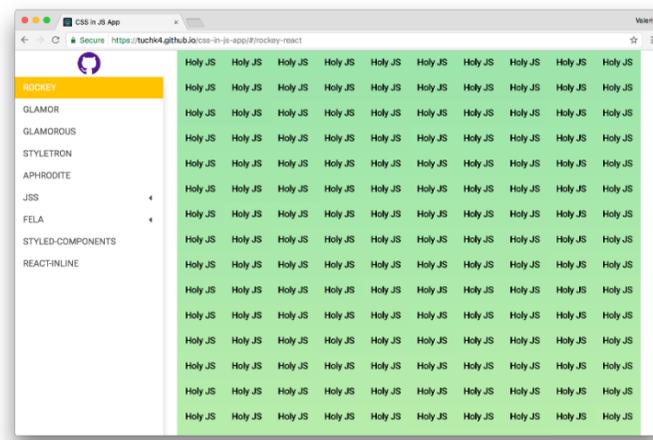
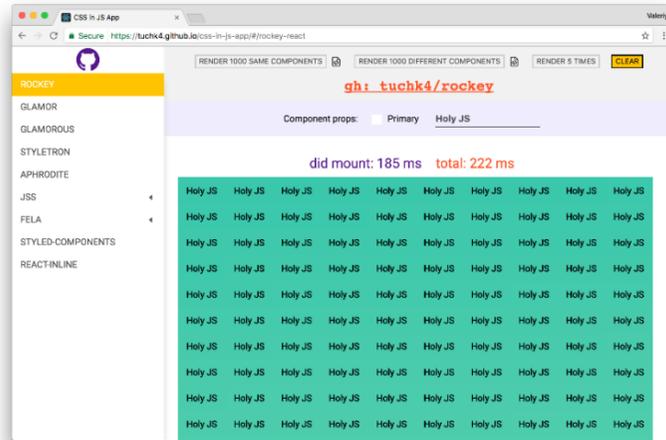
Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo

Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo Yo



Пример





JSS

```
const styles = {  
  block: {  
    minWidth: '16px',  
    height: '16px',  
    display: 'inline-block',  
    backgroundColor: props => colors[props.i]  
  }  
}
```

```
const styles = {  
  block: {  
    minWidth: '16px',  
    height: '16px',  
    display: 'inline-block',  
    backgroundColor: props => colors[props.i]  
  }  
}
```

```
const styles = {  
  block: {  
    minWidth: '16px',  
    height: '16px',  
    display: 'inline-block',  
    backgroundColor: props => colors[props.i]  
  }  
}
```

```
.block-0-1 {  
  background-color: ■ #dc2430;  
}
```

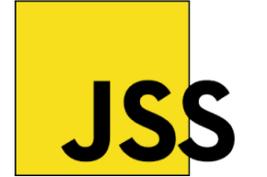
```
.block-0-0 {  
  height: 16px;  
  display: inline-block;  
  min-width: 16px;  
}
```

```
.block-0-1 {  
  background-color: ■ #dc2430;  
}
```

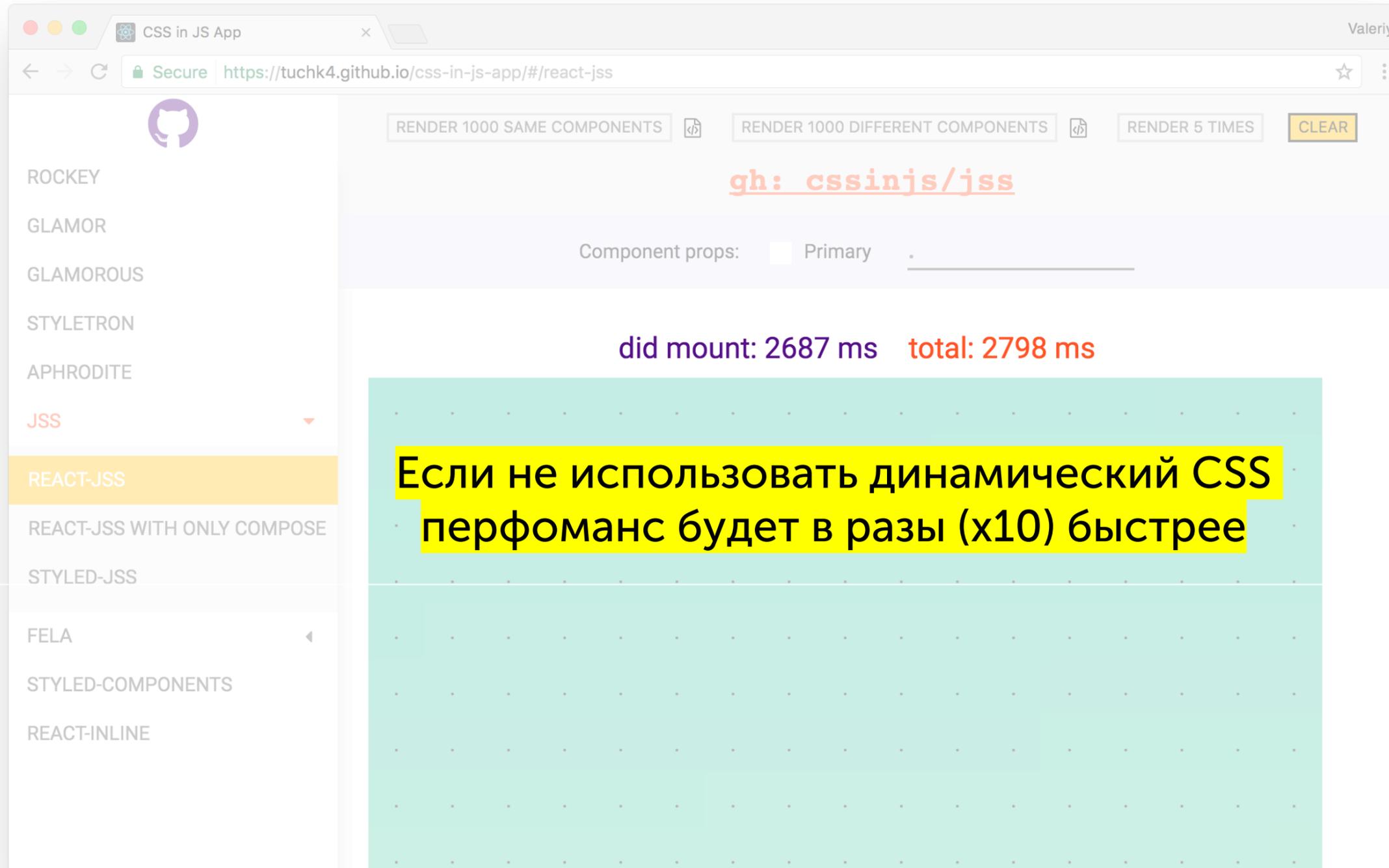
```
.block-0-0 {  
  height: 16px;  
  display: inline-block;  
  min-width: 16px;  
}
```



JSS



The screenshot shows a web browser window titled "CSS in JS App" with the URL `https://tuchk4.github.io/css-in-js-app/#/react-jss`. The interface includes a sidebar with a list of CSS-in-JS libraries: ROCKEY, GLAMOR, GLAMOROUS, STYLETRON, APHRODITE, JSS, REACT-JSS (highlighted), REACT-JSS WITH ONLY COMPOSE, STYLED-JSS, FELA, STYLED-COMPONENTS, and REACT-INLINE. The main content area features three buttons: "RENDER 1000 SAME COMPONENTS", "RENDER 1000 DIFFERENT COMPONENTS", and "RENDER 5 TIMES", along with a "CLEAR" button. Below these buttons, the text `gh: cssinjs/jss` is displayed. A "Component props:" section shows a "Primary" checkbox and a text input field. The performance results are shown as `did mount: 2687 ms` and `total: 2798 ms`. The main content area is filled with a grid of teal squares, each containing a small black dot.



RENDER 1000 SAME COMPONENTS RENDER 1000 DIFFERENT COMPONENTS RENDER 5 TIMES CLEAR

gh: [cssinjs/jss](https://github.com/cssinjs/jss)

Component props: Primary

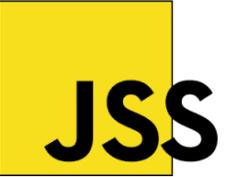
did mount: 2687 ms total: 2798 ms

Если не использовать динамический CSS перфоманс будет в разы (x10) быстрее

ROCKEY
GLAMOR
GLAMOROUS
STYLETRON
APHRODITE
JSS
REACT-JSS
REACT-JSS WITH ONLY COMPOSE
STYLED-JSS
FELA
STYLED-COMPONENTS
REACT-INLINE



JSS



Мы не используем:

- Низкий перфоманс для динамических правил
- Не хотим описывать стили объектом



FELA



FELA

“If the view is a function of state, your CSS should be too”

Style = **Function**(props)



FELA

```
const rule = props => ({  
  minWidth: '16px',  
  height: '16px',  
  display: 'inline-block',  
  backgroundColor: colors[props.i]  
})
```



FELA

```
const rule = props => ({  
  minWidth: '16px',  
  height: '16px',  
  display: 'inline-block',  
  backgroundColor: colors[props.i]  
})
```



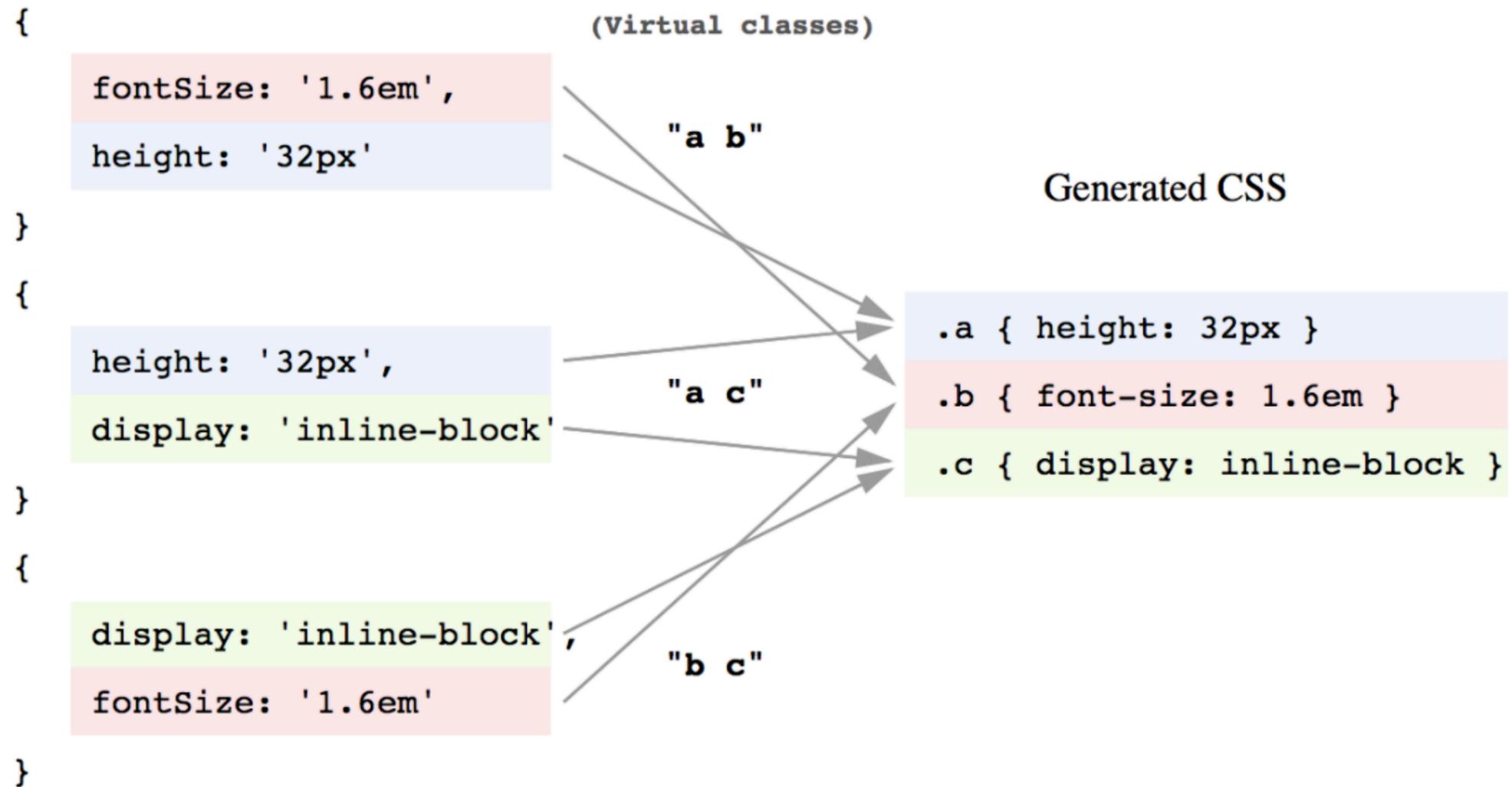
FELA

```
const rule = props => ({  
  minWidth: '16px',  
  height: '16px',  
  display: 'inline-block',  
  backgroundColor: colors[props.i]  
})
```



FELA

Atomic CSS design



FELA

```
.a {  
  background-color: #7b4397;  
}  
  
.b {  
  display: inline-block;  
}  
  
.c {  
  height: 16px;  
}  
  
.d {  
  min-width: 16px;  
}
```



FELA

```
<div class="a b c d e f g h anx j k">Yo</div>  
<div class="a b c d e f g h any j k">Yo</div>  
<div class="a b c d e f g h any j k">Yo</div>  
<div class="a b c d e f g h any j k">Yo</div>  
<div class="a b c d e f g h anz j k">Yo</div>  
<div class="a b c d e f g h anz j k">Yo</div>  
<div class="a b c d e f g h anz j k">Yo</div>  
<div class="a b c d e f g h aoa j k">Yo</div>  
<div class="a b c d e f g h aoa j k">Yo</div>  
<div class="a b c d e f g h aob j k">Yo</div>  
<div class="a b c d e f g h aob j k">Yo</div>  
<div class="a b c d e f g h aob j k">Yo</div>  
<div class="a b c d e f g h aob j k">Yo</div>  
<div class="a b c d e f g h aoc j k">Yo</div>  
<div class="a b c d e f g h aoc j k">Yo</div>  
<div class="a b c d e f g h aoc j k">Yo</div>  
<div class="a b c d e f g h aoc j k">Yo</div>  
<div class="a b c d e f g h aoc j k">Yo</div>
```



FELA



FELA

Мы не используем:

- Не хотим описывать стили объектом
- Не удобно работать в devtools (Atomic CSS design)



GLAM



GLAM

```
<div css={`  
  color: ${color};  
  font-size: 128px;  
  font-family: sans-serif;  
  margin: 0;  
`} >Hello World!</div>
```



GLAM

babel →

```
<div className="css-19w4ohx">Hello World!</div>
```



GLAM

```
.vars-l37dqe {  
  --css-19w4ohx-0: #4263eb;  
}  
  
.css-19w4ohx {  
  color: var(--css-19w4ohx-0);  
  font-size: 128px;  
  font-family: sans-serif;  
  margin: 0;  
}
```

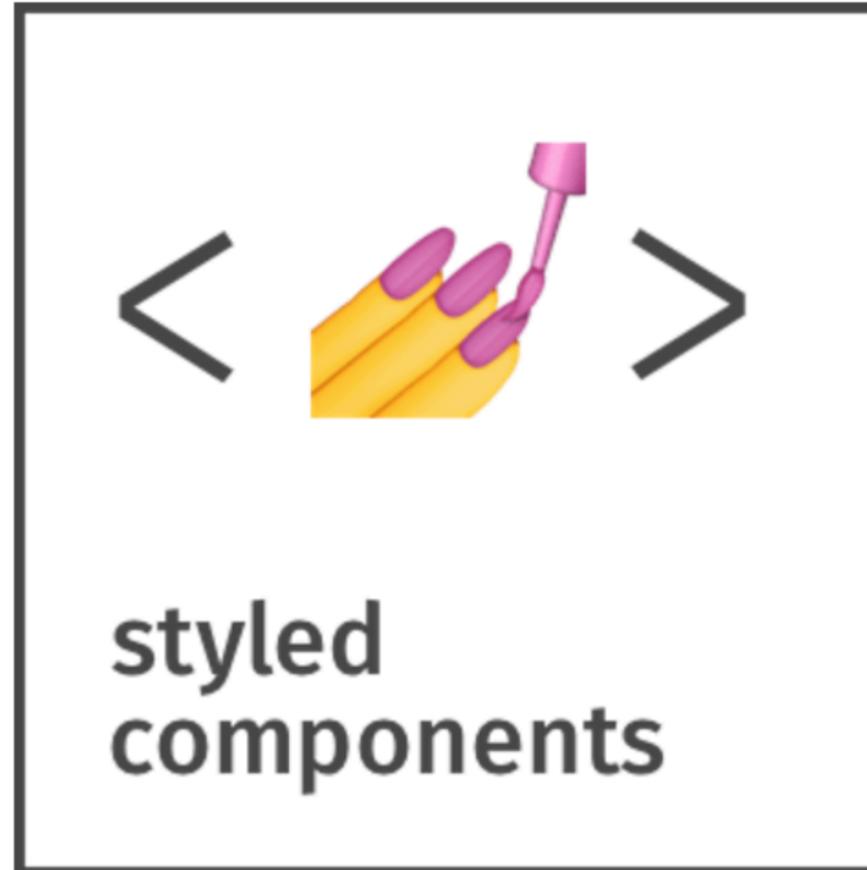


GLAM

Мы не используем:

- Нужен babel plugin и webpack loader
- Много чего еще не реализовано

 **Create React App**



styled-components



```
const Block = styled.div`  
  min-width: 16px;  
  height: 16px;  
  display: inline-block';  
  background-color: ${props => colors[props.i]}
```

styled-components



```
const Block = styled.div`  
  min-width: 16px;  
  height: 16px;  
  display: inline-block';  
  background-color: ${props => colors[props.i]}
```



styled-components



James Kyle ✓
@thejameskyle

To me CSS-in-JS is a logical next step to simply throwing out selectors in favor of targeting elements directly

styled-components



```
.cJoRIh {  
  min-width: 16px;  
  height: 16px;  
  display: inline-block;  
  background-color:  #00dbde;  
}  


---

  
.sc-bdVaJa {  
}
```



styled-components



CSS in JS App x Valeriy

Secure https://tuchk4.github.io/css-in-js-app/#/styled-components

RENDER 1000 SAME COMPONENTS RENDER 1000 DIFFERENT COMPONENTS RENDER 5 TIMES CLEAR

[gh: styled-components/styled-components](https://github.com/styled-components/styled-components)

Component props: Primary Yo

did mount: 5898 ms total: 5991 ms

Yo																
Yo																
Yo																
Yo																
Yo																
Yo																
Yo																
Yo																
Yo																



styled-components



Мы не используем:

- Очень низкий перфоманс
- Не удобно работать в devtools

Требования к библиотеке



Требования к библиотеке

- Должна быть быстрой!



Требования к библиотеке

- Должна быть быстрой!
- Небольшой размер



Требования к библиотеке

- Должна быть быстрой!
- Небольшой размер
- Максимально совместим с devtools



Требования к библиотеке

- Должна быть быстрой!
- Небольшой размер
- Максимально совместим с devtools
- Читаемые имена CSS классов



Требования к библиотеке

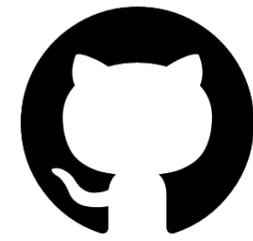
- Должна быть быстрой!
- Небольшой размер
- Максимально совместим с devtools
- Читаемые имена CSS классов
- Валидация правил и autoprefixer



Требования к библиотеке

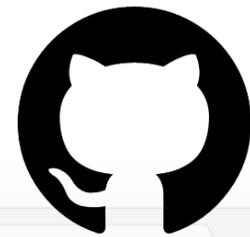
- Должна быть быстрой!
- Небольшой размер
- Максимально совместим с devtools
- Читаемые имена CSS классов
- Валидация правил и autoprefixer
- Динамический CSS

ЧТО и ПОЧЕМУ ИСПОЛЬЗУЕМ МЫ?



tuchk4/rockey





tuchk4/rockey



CSS in JS App x Valeriy

Secure https://tuchk4.github.io/css-in-js-app/#/rockey-react

RENDER 1000 SAME COMPONENTS RENDER 1000 DIFFERENT COMPONENTS RENDER 5 TIMES CLEAR

gh: [tuchk4/rockey](#)

Component props: Primary Yo

did mount: 175 ms total: 203 ms

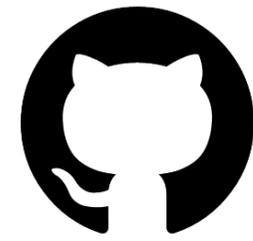
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	
Yo																	



rockey + recompose



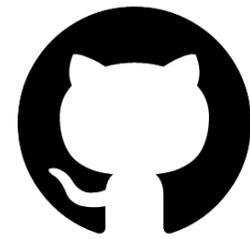
```
compose(  
  css`  
    font-size: 20px;  
    padding: 5px;  
    margin: 5px;  
    background: white;  
    border: 1px solid #ccc;  
  `,  
  
  css`  
    ${props => props.isRaised && `  
      box-shadow: 1px 1px #ccc;  
    `}  
  `,  
  
  )(Button)
```



tuchk4/rockey



Компоненты как селекторы

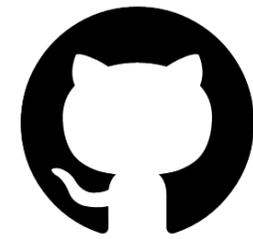


tuchk4/rockey



Компоненты как селекторы

```
const Card = (props) => (  
  <div>  
    <CardHeader>  
      <Title>I am CardHeader</Title>  
      <CloseIcon/>  
    </CardHeader>  
    <CardBody>I am Body</CardBody>  
    <CardActions>  
      <Button>Click me</Button>  
    </CardActions>  
  </div>  
)
```

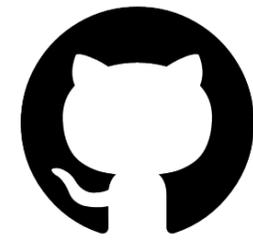


tuchk4/rockey



Компоненты как селекторы

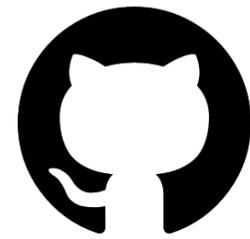
```
rockey(Card)`  
  Card {  
    width: 100px;  
  
    CardHeader {  
      background: #fc3;  
      CloseIcon {  
        float: right;  
      }  
    }  
  
    CardActions {  
      Button {  
        float: right;  
      }  
    }  
  }  
`
```



tuchk4/rockey



«Читабельные» имена CSS классов



tuchk4/rockey

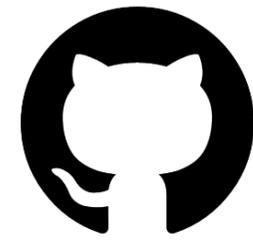


«Читабельные» имена CSS классов

- ▼ `<div class="Card-66f5">`
 - `<div class="CardHeader-be1e">I'm CardHeader</div>`
 - `<div class="CardBody-8ad4">I'am Body</div>``</div>`
- ▼ `<div class="PrimaryCard-2da2 Card-66f5">`
 - `<div class="CardHeader-be1e">I'm CardHeader</div>`
 - `<div class="CardBody-8ad4">I'am Body</div>``</div>`



Правильное наследование



tuchk4/rockey



Правильное наследование

```
const Button = rockey.button`  
  font-size: 20px;  
  padding: 5px;  
  margin: 5px;  
  background: white;  
  border: 1px solid #ccc;  
`
```

```
const PrimaryButton = Button`  
  color: blue;  
`
```

```
const SuperButton = PrimaryButton`  
  color: red;  
`
```



Правильное наследование

```
<button class="Button-b95b">Button</button>
```

```
<button class="PrimaryButton-ae2 PrimaryButton-ae2 Button-b95b">PrimaryButton</button>
```

```
<button class="SuperButton-baa8 PrimaryButton-ae2 Button-b95b">SuperButton</button>
```



tuchk4/rockey



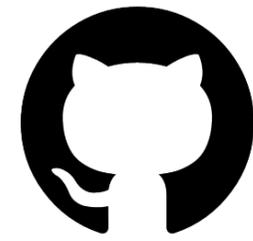
Правильное наследование

```
<button class="Button-b95b">Button</button>  
<button class="PrimaryButton-ae2 Button-b95b">PrimaryButton</button>  
<button class="SuperButton-baa8 PrimaryButton-ae2 Button-b95b">SuperButton</button>
```



Правильное наследование

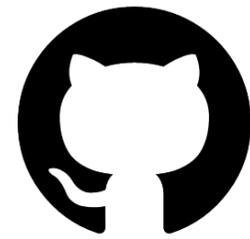
```
<button class="Button-b95b">Button</button>  
<button class="PrimaryButton-ae2 Button-b95b">PrimaryButton</button>  
<button class="SuperButton-baa8 PrimaryButton-ae2 Button-b95b">SuperButton</button>
```



tuchk4/rockey



Looks



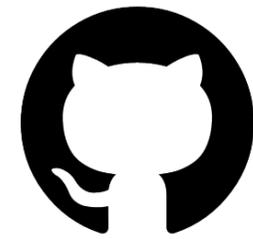
tuchk4/rockey



Looks

```
<Button primary={true}>I am m Button</Button>
```

```
<PrimaryButton>I am PrimaryButton</PrimaryButton>
```



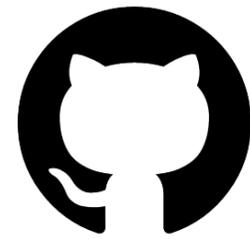
tuchk4/rockey



Looks

```
<Button primary={true}>I am m Button</Button>
```

```
<PrimaryButton>I am PrimaryButton</PrimaryButton>
```

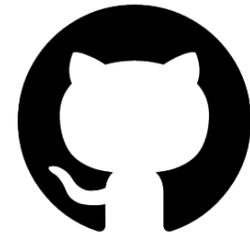


tuchk4/rockey



Looks

Button	raised	disabled	success	warning	primary	ripple
raised	-					
disabled		-				
success			-			
warning				-		
primary					-	
ripple						-

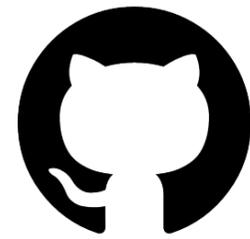


tuchk4/rockey



Looks

Button	raised	disabled	success	warning	primary	ripple
raised	-	✓	✓	✓	✓	✓
disabled	✓	-	✓	✓	✓	✓
success	✓	✓	-	✗	✗	✓
warning	✓	✓	✗	-	✗	✓
primary	✓	✓	✗	✗	-	✓
ripple	✓	✓	✓	✓	✓	-



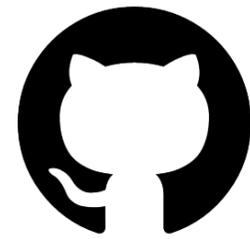
tuchk4/rockey



Looks

```
const Button = rockey.button`  
  padding: 5px;  
  color: ${props => props.disabled ? '#ccc' : '#000'};  
`
```

```
const { PrimaryButton, SuccessButton } = Button.look`  
  PrimaryButton {  
    color: blue;  
  }  
  
  SuccessButton {  
    color: green;  
  }  
`
```



tuchk4/rockey



Looks

```
const Button = rockey.button`  
  padding: 5px;  
  color: ${props => props.disabled ? '#ccc' : '#000'};  
`
```

```
const { PrimaryButton, SuccessButton } = Button.look`  
  PrimaryButton {  
    color: blue;  
  }  
  
  SuccessButton {  
    color: green;  
  }  
`
```



Реализация



Реализация

С использованием шаблонных строк

```
const Block = styled.div`  
  min-width: 16px;  
  height: 16px;  
  display: inline-block';  
  background-color: ${props => colors[props.i]}
```



Реализация

С использованием шаблонных строк

- Быстрее писать код



Реализация

С использованием шаблонных строк

- Быстрее писать код
- `ctrl+c` / `ctrl+v` из CSS файлов и примеров



Реализация

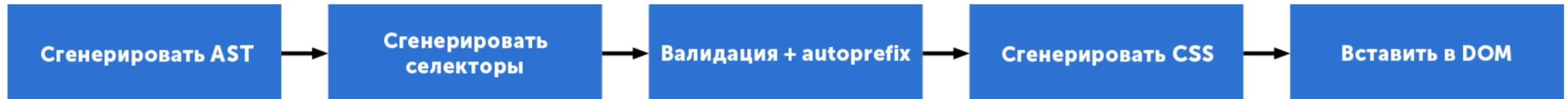
С использованием шаблонных строк

- Быстрее писать код
- `ctrl+c` / `ctrl+v` из CSS файлов и примеров
- Более привычный синтаксис



Этапы

Этапы



Нужен ли настоящий АСТ?



Нужен ли настоящий AST?

post**CSS**

Нужен ли настоящий AST?

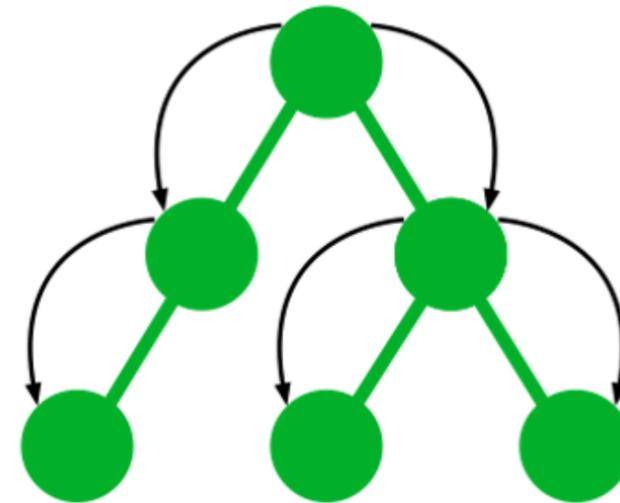
```
Dialog {  
  DialogHeader {  
    color: purple;  
  }  
}
```



Разбор CSS
строки



Разбор AST



CSS

- Вложенные селекторы
- Уникальные имена классов (cssmodules)
- Плагины
- Генерируем финальный CSS



Нужен ли настоящий AST?

Заранее знаем задачи

- Вложенный CSS
- Autoprefix
- Уникальные имена CSS классов

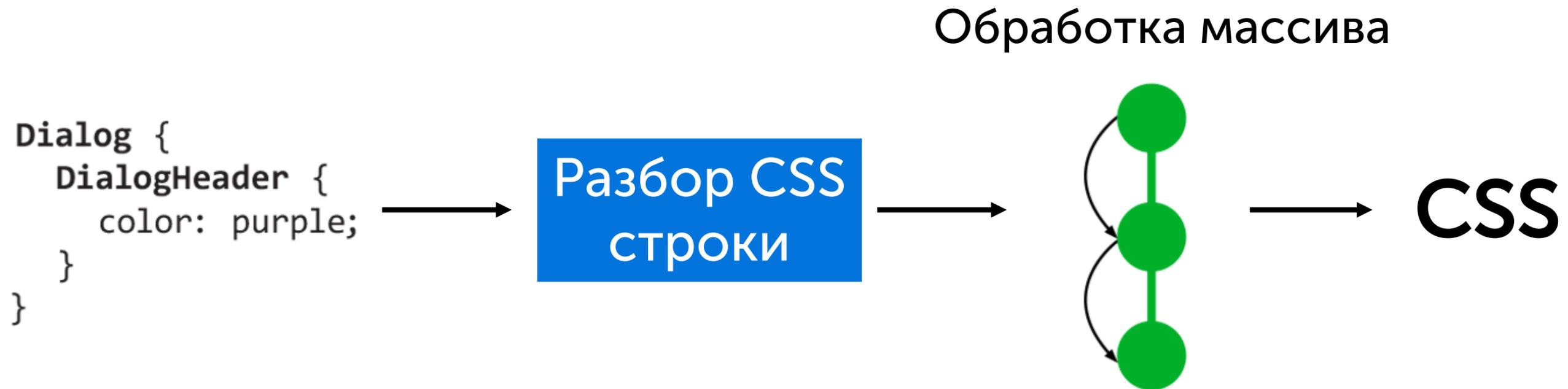


Нужен ли настоящий AST?

Задаем функцию определения имени CSS класса
при инициализации парсера

```
const parser = createParser({  
  getClassName: name => `${name}-${hash()}`  
})
```

Нужен ли настоящий AST?



- Генерируем финальный CSS



Парсинг

Парсинг

```
for (let symbol of raw) {  
  if (symbol === '{') {  
  
  } else if (symbol === '}') {  
  
  } else if (symbol === ':') {  
  
  }  
}
```

Парсинг

```
for (let symbol of raw) {  
    if (isOpenBracket(symbol)) {  
        openBlock();  
    } else if (isCloseBracket(symbol)){  
        closeBlock();  
        processBlock();  
    }  
    ...  
}
```



Парсинг

```
function render(raw) {  
  const isOpenBracket = symbol => {...}  
  const isCloseBracket = symbol => {...}  
  const openBlock = () => {...}  
  const closeBlock = () => {...}  
  const processBlock = () => {...}  
  
  for (let symbol of raw) {  
    if (isOpenBracket(symbol)) {  
      openBlock();  
    } else if (isCloseBracket(symbol)) {  
      closeBlock();  
      processBlock();  
    }  
    ...  
  }  
}
```

Парсинг

```
function render(raw) {  
  const isOpenBracket = symbol => {...}  
  const isCloseBracket = symbol => {...}  
  const openBlock = () => {...}  
  const closeBlock = () => {...}  
  const processBlock = () => {...}  
  
  for (let symbol of raw) {  
    if (isOpenBracket(symbol)) {  
      openBlock();  
    } else if (isCloseBracket(symbol)) {  
      closeBlock();  
      processBlock();  
    }  
    ...  
  }  
}
```



Парсинг

Автоматически заменяем вызов функции на её тело



Парсинг

```
if (symbol === '{') {
  if (!!components.length) {
    openedBrackets++;
    if (openedBrackets !== 1) {
      current += symbol;
    }
  } else if (!!modifier) {
    openedModifierBrackets++;
    if (openedModifierBrackets !== 1) {
      current += symbol;
    }
  }
} else if (symbol === '}') {
  if (!!components.length) {
    openedBrackets--;
  } else if (!!modifier) {
    openedModifierBrackets--;
  } else {
    throw new RockeySyntaxError(`${raw.slice(i - 1, i + 1)}\`);
  }
  if (!!components.length) {
    if (openedBrackets === 0) {
      const selector = [];
    }
  }
}
```



Парсинг

+40% перфоманса



tuchk4.github.io/rockey-css-parse-repl

The screenshot shows a web browser window with the URL `https://tuchk4.github.io/rockey-css-parse-repl/`. The page title is "Rockey CSS Parse REPL" and the user is logged in as "Valeriy".

The main content area displays the following code:

```
const getClassName = name => `__${name}_hash__`
```

```
1 Button {
2   color: ${props => props.color};
3   border: 1px solid #000;
4
5   :hover {
6     color: red;
7
8     Icon {
9       margin: 5px;
10    }
11  }
12 }
13
14 Layer {
15   padding: 15px;
16
17   Button {
18     margin: 0;
19   }
20 }
21
```

The "Props:" panel shows a single property:

color	: purple	REMOVE
-------	----------	--------

Below the props panel is an "ADD PROPERTY" button.

The generated CSS output is as follows:

```
.__Button_hash__:hover .__Icon_hash__ {
  margin: 5px;
}

.__Button_hash__:hover {
  color: red;
}

.__Button_hash__ {
  border: 1px solid #000;
}

.__Layer_hash__ .__Button_hash__ {
  margin: 0;
}

.__Layer_hash__ {
  padding: 15px;
}

.__mixin_color-1.__Button_hash__ {
  color: purple;
}
```

Как вставить CSS в DOM в рантайме



Как вставить CSS в DOM в рантайме

- Создать `<style/>` и вставить в `<head/>`
- `CSSStyleSheet` Api



CSSStyleSheet Api



CSSStyleSheet Api

```
const node = document.createElement('style')  
document.head.appendChild(style)
```

```
const sheet = node.sheet
```

```
sheet.insertRule(css, sheet.cssRules.length)
```



CSSStyleSheet Api

```
const node = document.createElement('style')  
document.head.appendChild(style)
```

```
const sheet = node.sheet
```

```
sheet.insertRule(css, sheet.cssRules.length)
```



CSSStyleSheet Api

Невозможно редактировать значения

```
Styles Computed Event Listeners DOM Breakpoints Properties
Filter
element.style {
}
.m-backgroundColor-962f-1.ShortcutDiv1-0a28 {
  background-color: ■ rgb(67, 198, 172);
}
.ShortcutDiv1-0a28 {
  min-width: 16px;
  height: 16px;
  display: inline-block;
  text-align: center;
  font-weight: bold;
  padding: ▶ 15px;
  border-width: ▶ initial;
  border-style: ▶ none;
  border-color: ▶ initial;
  border-image: ▶ initial;
}
```



<style>

<style>

<style>

```
.button {  
  color: red;  
}
```

```
.button .icon {  
  color: red;  
}
```

```
.icon {  
  color: black;  
}
```

</style>

```
insert(css)  
  const textNode = document.createTextNode(css);  
  styleNode.appendChild(textNode);  
}
```



<style>

**Вставлять в момент рендера
компонента**

<style>

insert(css)

#1



<style>

```
.button {  
  color: red;  
}
```

```
.button .icon {  
  color: red;  
}
```

```
.icon {  
  color: black;  
}
```

insert(css)

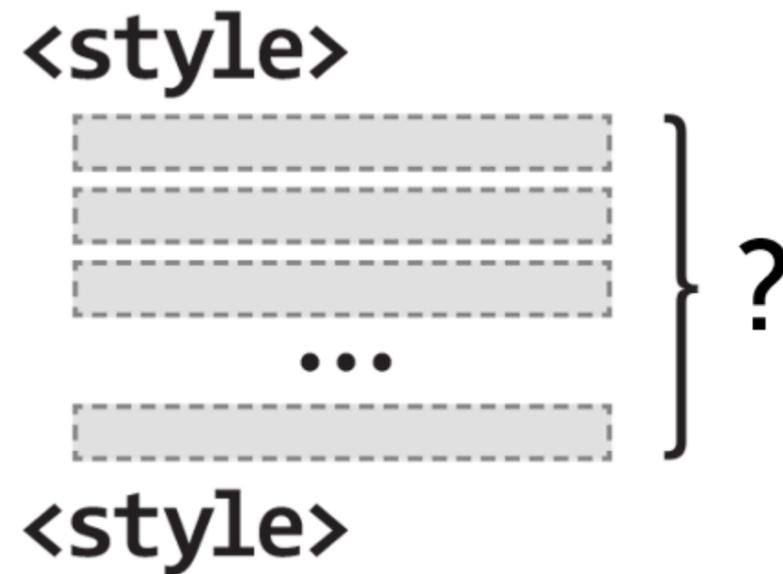
#2



</style>

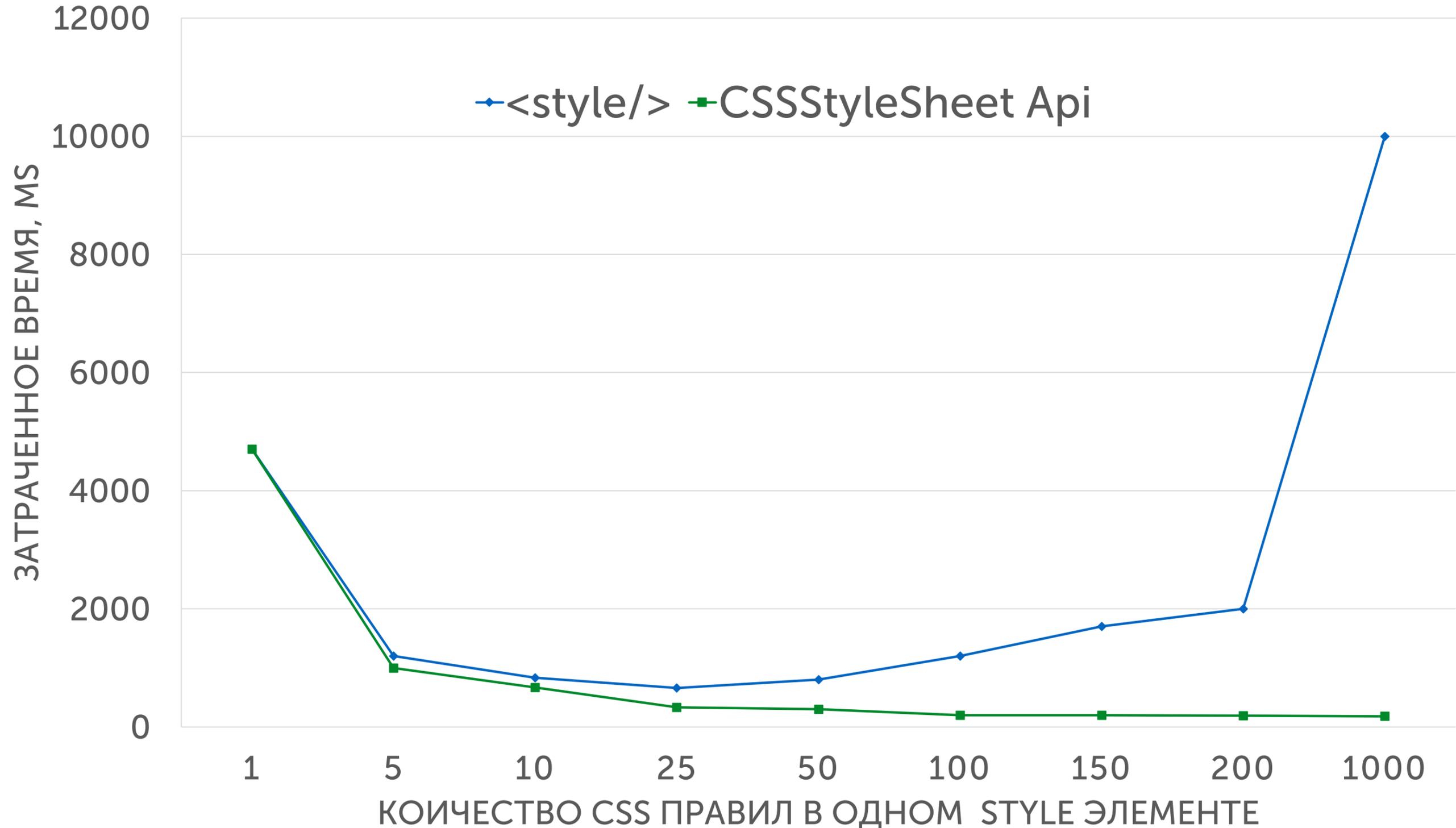
<style>

Количество стилей в одном <style> элементе





Вставляем 5000 CSS правил



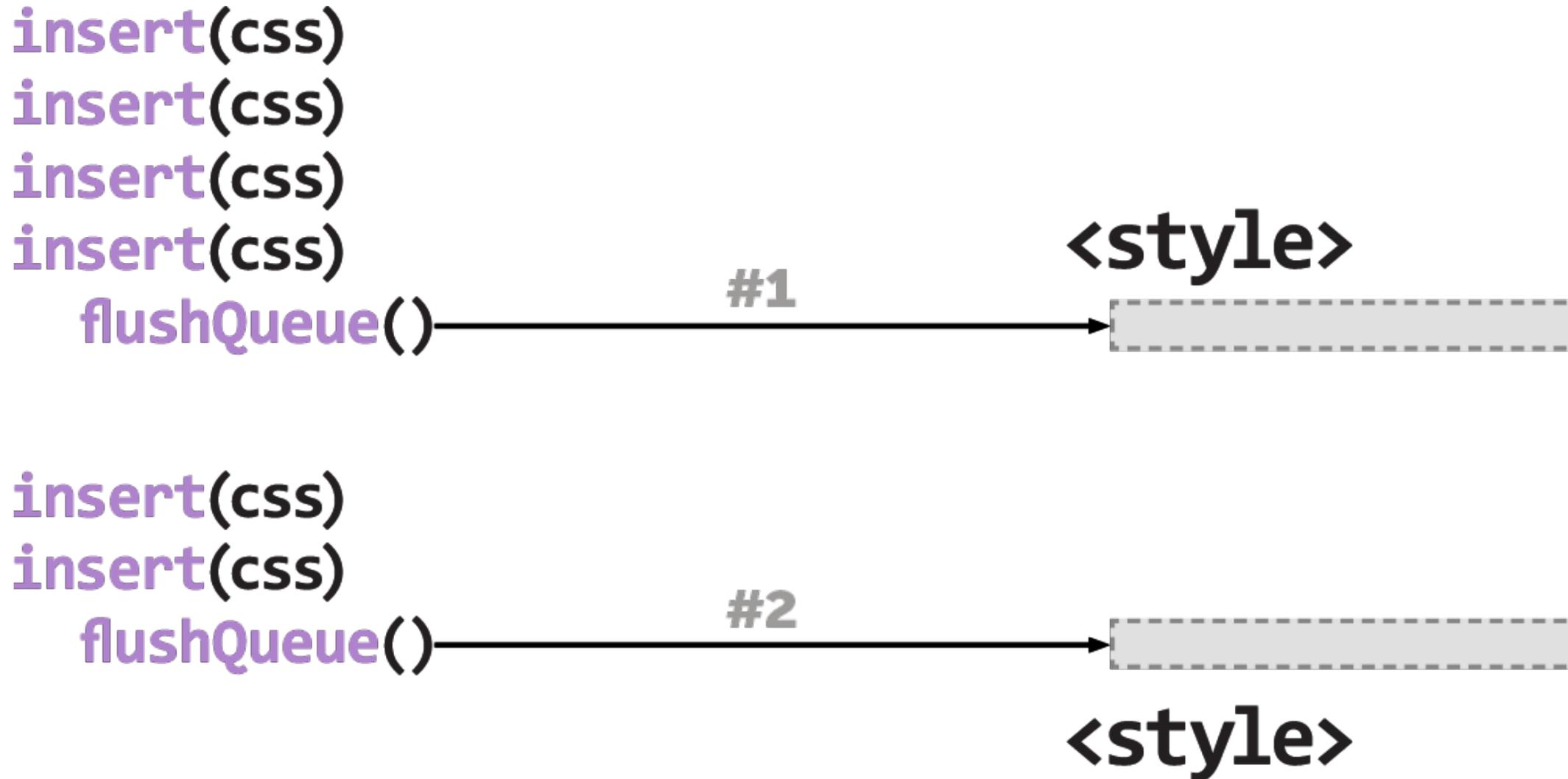


<style>

Собирать очередь?

<style>

Собираем очередь





<style>

Ограничивать размер очереди?

<style>

MAX_QUEUE_SIZE = 2





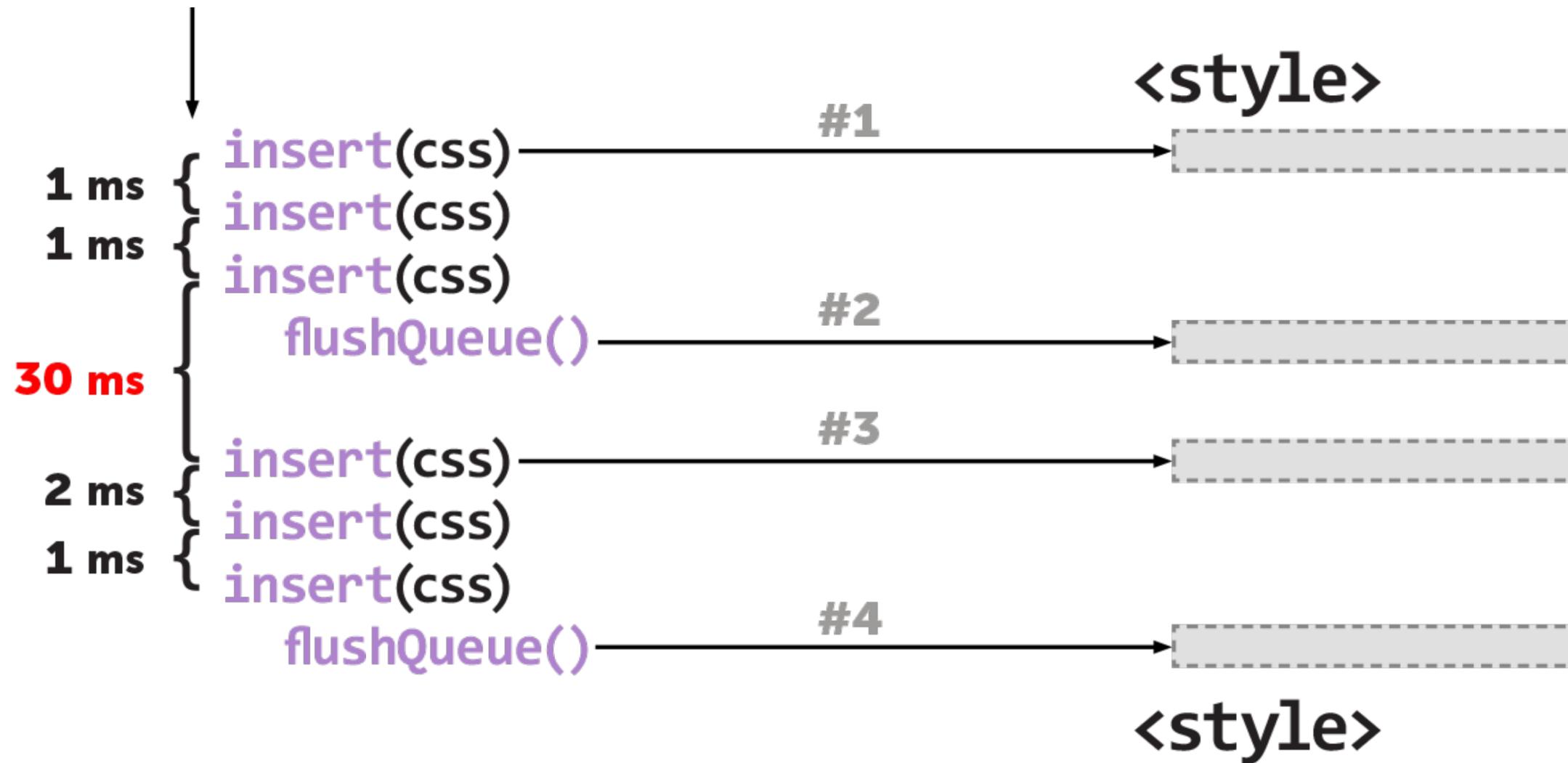
<style>

Замерять частоту вызова `insert()`?



<style> CALL_FREQUENCY

Собираем очередь если между вызовами менее 5ms

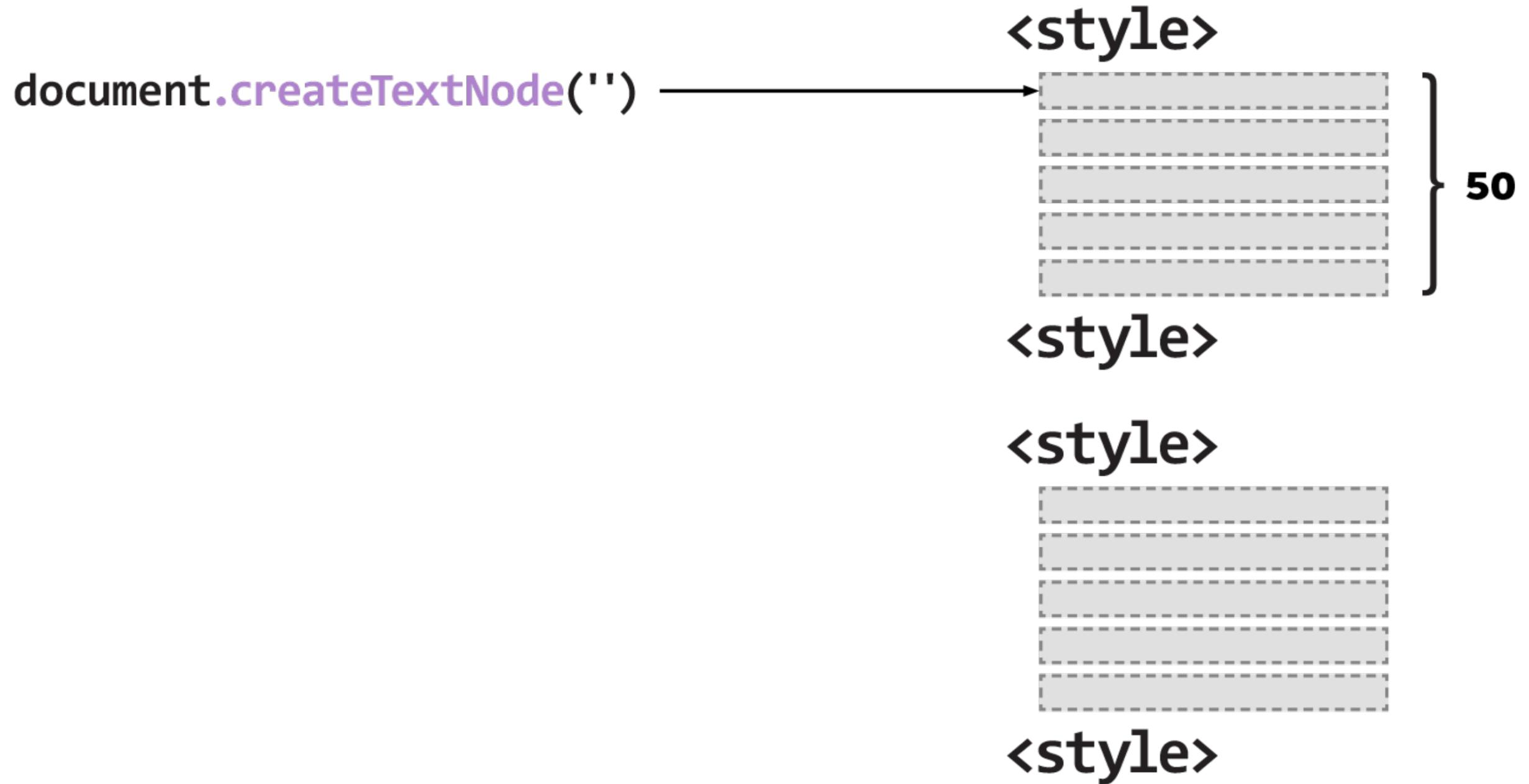




<style>

При старте приложения создаем **N textNode**
и сразу вставляем их в **<style/>**

<style>





<style>

```
insert(css) {  
  const textNode = requestFreeTextNode(css)  
  textNode.textContent = css  
}
```



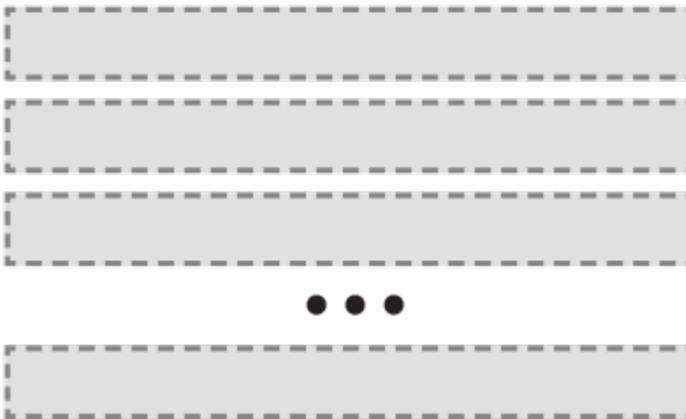
<style>

```
insert(css) {  
  const textNode = requestFreeTextNode(css)  
  textNode.textContent = css  
}
```

<style>

<style>

```
.button {  
  color: red;  
}
```



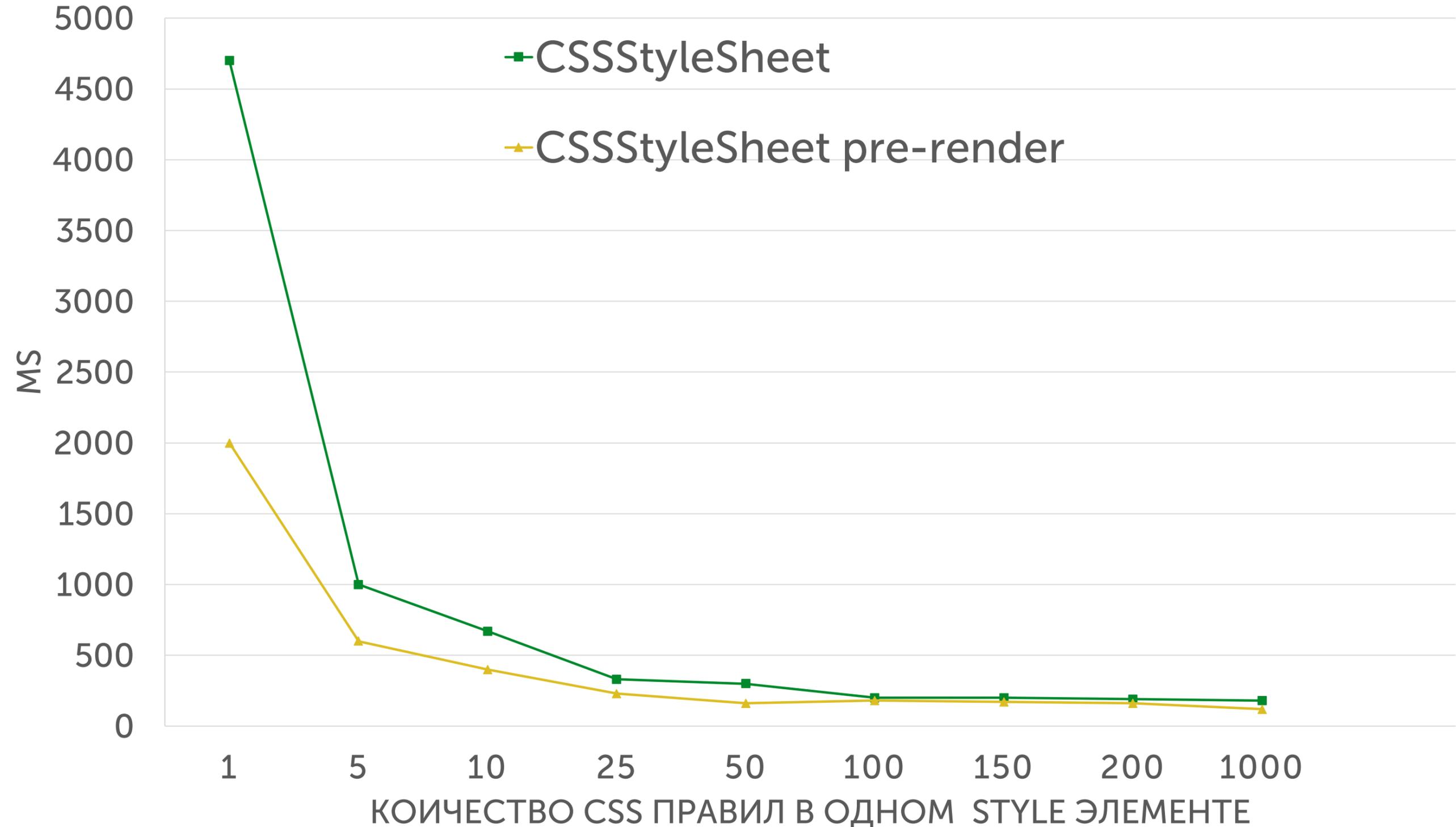
requestFreeTextNode()

textNode.textContent = css

</style>



Вставляем 5000 CSS правил

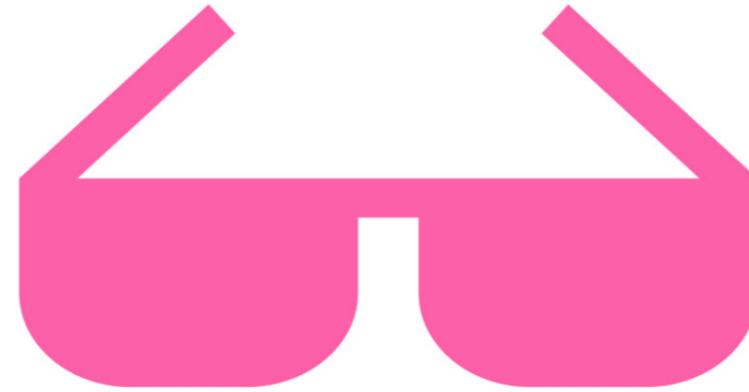


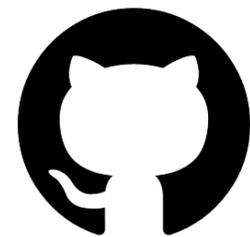


ИТОГ

production – CSSStyleSheetApi
1000 CSS правил в одном <style/>

development – работа как с обычным DOM
25 CSS правил в одном <style/>



 **tuchk4/awesome-css-in-js**



Спасибо

ВАЛЕРИЙ Сорокобатько