



baseballyama

Today's goal

Svelte の最新情報をキャッチアップできる

自己紹介

- 山下 裕一郎 (baseballyama)
- 株式会社フライル
- Svelte コアチームメンバー



The screenshot displays the Flyle website. The top navigation bar includes links for 'プラン' (Plans), '導入事例' (Use Cases), 'ブログ' (Blog), 'イベント' (Events), and 'お役立ち資料' (Helpful Materials), along with 'ログイン' (Login), 'お問い合わせ' (Contact Us), and a '資料ダウンロード' (Download Materials) button. The main headline reads '数千件のVOC分析・分類作業を最先端のAIで不要に' (Eliminate thousands of VOC analysis and classification tasks with cutting-edge AI). Below this, a description states: 'Flyleは、製品開発・CXのためのニーズ分析からアイデア管理までを大きく効率化するVOC管理・製品企画クラウドです' (Flyle is a VOC management and product planning cloud that significantly improves efficiency from needs analysis to idea management for product development and CX). Three bullet points highlight features: '最先端のAI技術でVOC・フィードバック・アンケートを自動分析' (Automated analysis of VOC, feedback, and surveys using cutting-edge AI technology), '課題・アイデアの優先度づけ、共有を効率化' (Efficient prioritization and sharing of issues and ideas), and '先進的な企業が導入するプロダクトマネジメント体制を構築' (Building a product management system adopted by advanced companies). Two call-to-action buttons are present: '3分で分かる“Flyle” 資料ダウンロード' (Download materials to understand "Flyle" in 3 minutes) and 'どんな効率化ができるか トライアルで体感する' (Experience the efficiency improvements with a trial). The dashboard preview on the right shows a 'フィードバック' (Feedback) section with a 'AI分析' (AI Analysis) button, a 'クリティカル' (Critical) filter, and a list of feedback items with status indicators like 'ネガティブ' (Negative), '要望' (Request), and 'ポジティブ' (Positive).

ソフトウェア業界だけでなく、メーカー・製造業・化学・旅行・小売など多様な業界での実績

Logos of partner companies: 株式会社JALブランドコミュニケーション, BOÖK・OFF, RICOH, GMO, DeNA, commune, Money Forward, UZABASE, kaonavi, SEIKO, HRBrain, oRo co.,Ltd., CLOUDSIGN, BrainPad, LPIXEL, estie, enpay, SUPER STUDIO, パーソル キャリア, infomart, TOKIUM, RComClip, basic, フロスマート.

Agenda

- Svelte とは
- 最新情報の共有
 - リアクティビティ (Runes)
 - Slot / Snippets
 - イベントハンドラー
 - その他

Svelteとは

- フロントエンドを構築するためのフレームワーク
- HTMLのスーパーセットを謳っている
- コンパイラを使用する点が他のフレームワークと大きく異なる
- コンパイラを使用することで軽量のランタイムをエンドユーザーに配信できる

強化されたチーム

■ チームの変更

- Dominic Gannaway が Vercel に入社
 - 元 React コア チーム メンバー
 - lexical と inferno の作者
- Vercel の Svelte チームは3名に



Rich Harris



Simon Holthausen

The screenshot shows the GitHub profile of Dominic Gannaway (trueadm). It includes a profile picture, a bio stating he is a software engineer at Vercel and author of @sveltejs, and a list of pinned repositories: sveltejs/svelte, facebook/lexical, infernojs/inferno, facebook/react, sveltejs/kite, and facebookarchive/prepack. It also features a contribution graph for the last year, showing activity from April 2024 to April 2023, and a section for contribution activity in April 2024, highlighting 33 commits in 3 repositories and a pull request in sveltejs/svelte that received 30 comments.

Svelte 5

Svelte 5 は
これまでの反省やユーザーからの要望を踏まえ
0から書き直した

Svelte 5

Svelte 5 での変更点を見ていきましょう

Runes

- 最も大きな変更の1つ
- Svelte 4 まではリアクティビティには `$` を使用していた
- Svelte 5 では Runes と呼ばれるマクロを使用する

Runes

基本的なステート

Svelte 4

Reactivity



App.svelte +

```
1 <script>
2   let count = 0;
3
4   function increment() {
5     count += 1;
6   }
7 </script>
8
9 <button on:click={increment}>
10   clicks: {count}
11 </button>
12
13
```

input ☒ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3
4   function increment() {
5     count += 1;
6   }
7 </script>
8
9 <button onclick={increment}>
10   clicks: {count}
11 </button>
```

input ☒ output



PREVIEW



```
<script>
- let count = 0;
+ let count = $state(1);

  function increment() {
    count += 1;
  }
</script>

<!-- この差分については後述します -->
- <button on:click={increment}>
+ <button onclick={increment}>
  clicks: {count}
</button>
```

Runes

算出フィールド

Svelte 4

Computed



App.svelte* +

```
1 <script>
2   let count = 0;
3   $: doubled = count * 2;
4
5   function increment() {
6     count += 1;
7   }
8 </script>
9
10 <button on:click={increment}>
11   clicks: {count}
12 </button>
13 doubled: {doubled}
14
```

input ☐ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3   let doubled = $derived(count * 2);
4
5   function increment() {
6     count += 1;
7   }
8 </script>
9
10 <button onclick={increment}>
11   clicks: {count}
12 </button>
13 doubled: {doubled}
```

input ☐ output



PREVIEW



```
<!-- 一部抜粋 -->
```

```
<script>
```

```
  let count = 0;
```

```
- $: doubled = count * 2;
```

```
+ let doubled = $derived(count * 2);
```

```
  function increment() {
```

```
    count += 1;
```

```
  }
```

```
</script>
```

Runes

これによりこんなことができるようになります

Svelte 4

Computed (Not Working)



App.svelte +

```
1 <script>
2   let count = 0;
3   const getDoubled = () => count * 2;
4   $: doubled = getDoubled();
5
6   function increment() {
7     count += 1;
8   }
9 </script>
10
11 <button on:click={increment}>
12   clicks: {count}
13 </button>
14 doubled: {doubled}
```

input ☐ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3   const getDoubled = () => count * 2;
4   const doubled = $derived(getDoubled());
5
6   function increment() {
7     count += 1;
8   }
9 </script>
10
11 <button onclick={increment}>
12   clicks: {count}
13 </button>
14 doubled: {doubled}
```

input ☐ output



PREVIEW



```
<!-- 一部抜粋 -->
```

```
<script>
```

```
  let count = 0;
```

```
  const getDoubled = () => count * 2;
```

```
- $: doubled = getDoubled();
```

```
+ const doubled = $derived(getDoubled());
```

```
  function increment() {
```

```
    count += 1;
```

```
  }
```

```
</script>
```

Runes

- Svelte 4 では依存関係の追跡をコンパイルタイムで実施していました
 - 推移的に呼び出される関数内でのステートの書き換えに追従することができませんでした
- Svelte 5 では依存関係の追跡をランタイムで実施します
 - これにより推移的に呼び出される関数内でのステートの書き換えにも追従できるようになりました
 - これにより `.svelte.(js|ts)` ファイルでもリアクティビティを使用できるようになりました
 - これにより `composables` によるロジックの共有化がより柔軟になりました

Runes

エフェクト関数

Svelte 4

Effect



App.svelte +

```
1 <script>
2   let count = 0;
3   $: {
4     console.log({ count });
5   }
6
7   function increment() {
8     count += 1;
9   }
10 </script>
11
12 <button on:click={increment}>
13   clicks: {count}
14 </button>
```

input ☐ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3
4   $effect(() => {
5     console.log({ count });
6   });
7
8   function increment() {
9     count += 1;
10  }
11 </script>
12
13 <button onclick={increment}>
14   clicks: {count}
15 </button>
```

input ☐ output



PREVIEW



```
<!-- 一部抜粋 -->
<script>
  let count = 0;
-  $: {
-   console.log({ count });
- }
+ $effect(() => {
+   console.log({ count });
+ });

  function increment() {
    count += 1;
  }
</script>
```

Runes

オブジェクト

Runes (オブジェクト)

- また、リアクティビティがよりきめ細かになりました
- Svelte 4 までは、リアクティブを発火する方法は再代入のみでした
- Svelte 5 では、push 関数などの破壊的関数の呼び出しでも発火します

Svelte 4

Reactivity



App.svelte +

```
1 <script>
2   let count = 0;
3   let items = [];
4
5   function onClick() {
6     count += 1;
7     items.push(count);
8   }
9 </script>
10
11 <button on:click={onClick}>
12   clicks: {count}
13 </button>
14 [JSON.stringify(items)]
```

input ☒ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3   let items = $state([]);
4
5   function onClick() {
6     count += 1;
7     items.push(count);
8   }
9 </script>
10
11 <button onclick={onClick}>
12   clicks: {count}
13 </button>
14 [JSON.stringify(items)]
```

input ☒ output



PREVIEW



Runes (その他)

- `beforeUpdate` / `afterUpdate` は廃止されます。
- `$effect.pre` / `$effect` を使用してください。

Runes

- 全 Runes API は以下でご確認ください。

<https://svelte-5-preview.vercel.app/docs/runes>

Slot / Snippets

- JSX が `<template>` よりも優れている点の1つはファイル内で要素を簡単に使い回すことができる点です。

```
const data = ["Alice", "Bob", "Charlie", "David"];
const renderItemElement = (name) => <div key={name}>{name}</div>;

return (
  <div>
    <div>{data.map(renderItemElement)}</div>
  </div>
);
```

- Svelte 5 では、このJSXの利点を取り入れることに成功しました。

```
<script>
  const data = ["Alice", "Bob", "Charlie", "David"];
</script>

{#Snippet item(name)}
  <div>{name}</div>
{/Snippet}

{#each data as name}
  {@render item(name)}
{/each}
```

Slot / Snippets

- この Snippet は子コンポーネントに渡すことができます



PREVIEW

WORK IN PROGRESS!

Docs Status • Svelte •



App.svelte Child.svelte +

RUNES

Result

JS output

CSS output

AST output

```
1  ✓ <script>
2    import Child from './Child.svelte';
3  </script>
4
5  {#snippet item(name)}
6    <div>{name}</div>
7  {/snippet}
8
9  <Child {item} />
```

Message from child component!

Console

CLEAR

Slot / Snippets

- 詳細は、以下をご確認ください。

<https://svelte-5-preview.vercel.app/docs/Snippets>

イベントハンドラー

- イベントハンドラーの書き方が若干変わりました

Svelte 4

Reactivity



App.svelte +

```
1 <script>
2   let count = 0;
3
4   function increment() {
5     count += 1;
6   }
7 </script>
8
9 <button on:click={increment}>
10   clicks: {count}
11 </button>
12
13
```

input ☒ output



SVELTE | REPL



Svelte 5

App.svelte +

 RUNES

```
1 <script>
2   let count = $state(0);
3
4   function increment() {
5     count += 1;
6   }
7 </script>
8
9 <button onclick={increment}>
10   clicks: {count}
11 </button>
```

input ☒ output



PREVIEW



```
<!-- 一部抜粋 -->  
- <button on:click={increment}>  
+ <button onclick={increment}>  
  clicks: {count}  
</button>
```

イベントハンドラー

- この変更により、イベントは `props` と同様に記述できるようになりました
- また定型的な `createEventDispatcher` の使用は不要になりました🎉



PREVIEW

WORK IN PROGRESS!

Docs Status • Svelte •



App.svelte



RUNES

Result

JS output

CSS output

AST output

```
1 <script>
2   let count = $state(0);
3
4   function onclick() {
5     count += 1;
6   }
7 </script>
8
9 <button {onclick}>
10   clicks: {count}
11 </button>
```

clicks: 0

Console

CLEAR

```
<!-- 一部抜粋 -->  
- <button onclick={increment}>  
+ <button {onclick}>  
  clicks: {count}  
</button>
```

イベントハンドラー

- また、子コンポーネントが任意のイベントを受け取れるようになりました
 - これは特にUIライブラリの作者にとって待望の機能でした



PREVIEW

WORK IN PROGRESS!

Docs Status • Svelte •



App.svelte Child.svelte +

RUNES

Result

JS output

CSS output

AST output

```
1 <script>
2   import Child from './Child.svelte';
3   let clickCount = $state(0);
4   let dblclickCount = $state(0);
5
6   function onclick() {
7     clickCount += 1;
8   }
9
10  function ondblclick() {
11    dblclickCount += 1;
12  }
13 </script>
14
15 <Child {onclick} {ondblclick} />
16 <div>
17   clickCount: {clickCount}
18 </div>
19 <div>
```

Click Child Component

clickCount: 0

dblclickCount: 0

Console

CLEAR

イベントハンドラー

- Svelte 4 では、各DOMにイベントがアタッチされていました
- Svelte 5 では、アプリケーションルートにのみイベントをアタッチします
 - 一般的に Event delegation と呼ばれる手法です
 - これは、React や Solid で用いられている手法です

イベントのアタッチは時間のかかる処理であるため、特に大規模なDOMでレンダリング速度やハイドレーション速度に違いが出るはずです。

また、メモリ使用量にも良い影響があるはずです。

テンプレート部での TypeScript

- テンプレート部で TypeScript が使用できるようになりました🎉
- `acorn-typescript` を使用しています



PREVIEW

WORK IN PROGRESS!

Docs Status • Svelte •



App.svelte



RUNES

Result

JS output

CSS output

AST output

```
1 ✓ <script lang="ts">
2   const data = [{ name: "Alice" }, { name: "Bob" }, { name: "Charlie" }, { name: "David" }]
3 </script>
4
5 ✓ {#each data as user}
6   {@const name: string = user.name }
7   <li>{name}</li>
8 {/each}
```

- Alice
- Bob
- Charlie
- David

Console

CLEAR

CSS パーサー

- Svelte 4 までは、`css-tree` を使用してスタイル部を解析していました
- Svelte 5 では、独自のコードで CSS を解析しています
 - Svelte コンパイラは、CSS セレクタ部は重要ですが、宣言部は基本的に何もする必要がありません
 - よって、一般的な CSS パーサーよりも簡易なパーサーで充分でした
 - これにより将来の CSS の新仕様に対して簡単に対応できる可能性が高まりました
 - また、当時コンテナクエリをサポートする CSS パーサーはありませんでした
 - 更に、このメリットを活かして、子コンポーネントにスタイルを渡す方法を検討しています (遂に! 🎉)

型定義ファイル

- Svelte 5 では `dtb-buddy` というライブラリを使用して型を自動生成しています
- これにより、1.1MB あった型情報が 25KB になりました
- また、関数から宣言先に飛ぶ際、型宣言ではなく関数自体に飛べるようになりました🎉

参照: <https://github.com/sveltejs/svelte/pull/8702>

HTMLのスーパーセットとしての Svelte

このコードをブラウザで表示すると...

```
<div />  
hello!
```

このようにレンダリングされます...

```
<div>hello!</div>
```

しかし Svelte 4 はこうレンダリングしていました...

```
<div></div>  
hello!
```

- `<div>` などの 非 void HTML タグは自己終了できません
- Svelte 5 は `<div />` のようなコードを警告するようになりました

参考: <https://github.com/sveltejs/svelte/pull/11114>

Svelte 4 からの移行

- Svelte 5 は Svelte 4 までの機能をサポートしています
- 1コンポーネントずつ漸進的に移行可能です
- 実際に私も個人的に管理している SvelteKit アプリを移行しましたが、ライブラリバージョンを Svelte 5 に上げただけでも正しく動作しました
- Svelte 5 では Svelte 4 までのテストが通ることを確認しています
- 詳細は [ステータスページ](#) から確認可能です



PREVIEW

WORK IN PROGRESS!

Docs Status • Svelte •



App.svelte



RUNES

Result

JS output

CSS output

AST output

```
1 <script>
2   let count = 0;
3
4   function increment() {
5     count += 1;
6   }
7 </script>
8
9 <button on:click={increment}>
10   clicks: {count}
11 </button>
12
13
```

clicks: 0

Console

CLEAR

ベンチマーク

Svelte は、ベンチマークを追求するのではなく、実際的な開発者体験とユーザー体験を追求しています。よって、ベンチマークを重要視していません。

この考え方に基づき、この発表でもベンチマークは紹介しません

但し `js-framework-benchmark` の結果はかなり良いです。

(ベンチマーク用の関数を一切用意していないにも関わらず!)

興味のある方は一度ご確認ください。

まとめ (今日ご紹介したもの)

- (チーム体制) 元 React コアチームのメンバーが参加しフルタイム3名体制になった
- (リアクティビティ) 依存関係の追跡がランタイムになったことで柔軟性が向上した
 - js/ts ファイルでもリアクティビティを使用できるようになった
 - Array / Map の破壊メソッドの利用でもリアクティブになった
 - 高度な機能も登場し、必要に応じてより細かな制御も可能になった
- (新しい slot) Snippets により Svelte コンポーネント内で再利用可能な部品を定義できるようになった
- (イベントハンドラー) 記述が単純になり柔軟性が向上した
 - `createEventDispatcher` は不要になった
 - イベントのアタッチがトップ要素に移譲されることで性能・メモリ効率が向上した
 - Uコンポーネントは任意のイベントを受け付け可能になった
- テンプレート部の TypeScript 対応
- 自作のCSSパーサーを採用
- 型定義ファイルの改善
- Svelte 4 から漸進的に移行可能

まとめ

これ以外にも沢山の改善が導入されています。
ぜひ公式ドキュメントを一読ください。

公式ドキュメント: <https://svelte-5-preview.vercel.app/docs/introduction>

まとめ

Svelte 4 と Svelte 5 の書き方の対比表は
以下のサイトがよくまとまっています。

Component party: <https://component-party.dev/?f=svelte4,svelte5>

その他

今日紹介できませんでしたが
Next.js / Nuxt の Svelte 版である SvelteKit も
昨年末にメジャーバージョンアップをしました。
こちらも一度ドキュメントを参照ください。

また Svelte 用の ESLint Plugin も提供しているので
是非ご活用ください

SvelteKit: <https://kit.svelte.dev/>

SvelteKit (日本語版): <https://kit.svelte.jp/>

ESLint Plugin Svelte: <https://github.com/sveltejs/eslint-plugin-svelte>