# Simple Svelte

A Frontend Book From A Backend Developer

Fuqiang Presents

2021

# Simple Svelte

A Frontend Book for Backend Developers

**Fuqiang Wang**

**2021-09-13**

**Fuqiang Presents**

# Contents

# Part I

# Svelte Introduction

# Chapter 1

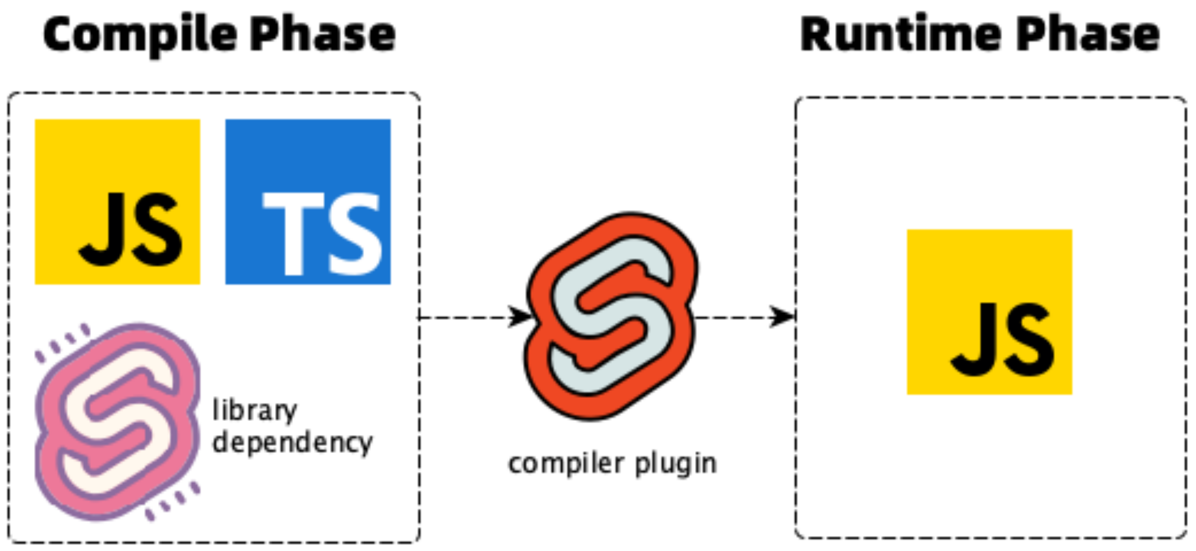
# What's Svelte?

Svelte is a web framework created by Rich Harris, svelte version 3 (in this book when I mention svelte I mean svelte3) is the most popular which was released since 2019.

Svelte is a different kind of web framework to the main-stream React/Vue/Angular SPA [1] frameworks, because it's more a compiler(or transpiler) than a runtime framework.

Most SPA web frameworks has their library/framework dependencies at runtime, but svelte does things in another way, it will compile SPA which is crafted with svelte library and conventions into vanilla javascript, then when the SPA is deployed, it just run as vanilla javascript without any runtime framework dependencies:

[1]SPA: Single Page Application



We can split svelte's work into 2 phases:

1. **dev and compile phase**, in this phase, we write our SPA with svelte libraries and conventions, then compile and debug before deploying to production;
2. **runtime phase**, in this phase, we deploy the compiled result which is just vanilla javascript that can be run by most browsers nowadays;

In this book, we will spend most our time in the first phase, that's, the dev & compile phase, we will learn how a svelte app is developed, how to configure rollup(svelte's module bundler) svelte plugin to compile the app and even how to deploy the app to production.

Before we get started, I would like to talk about why I chose svelte as my SPA web framework, maybe you will choose svelte with same reasons too.

# Part II

# Dive Into Svelte

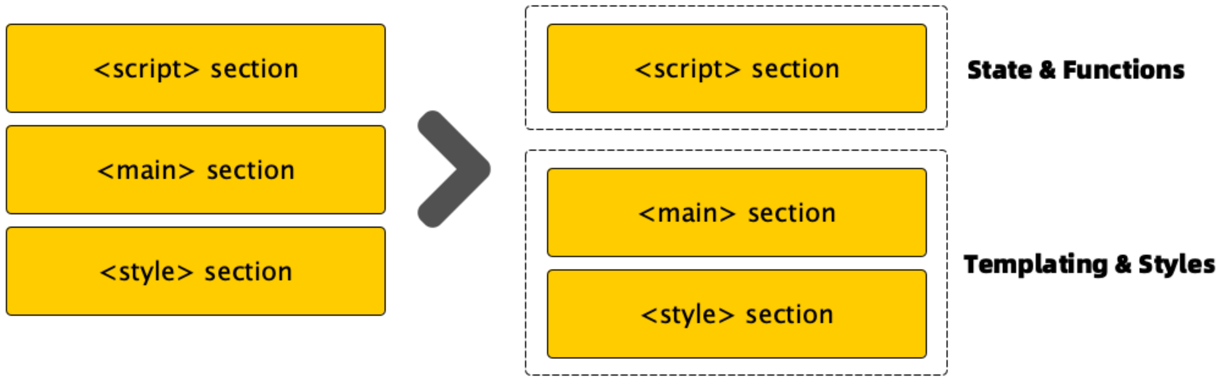
# Chapter 5

# What makes a svelte component

Previously, we have seen that App.svelte has 3 sections/parts:

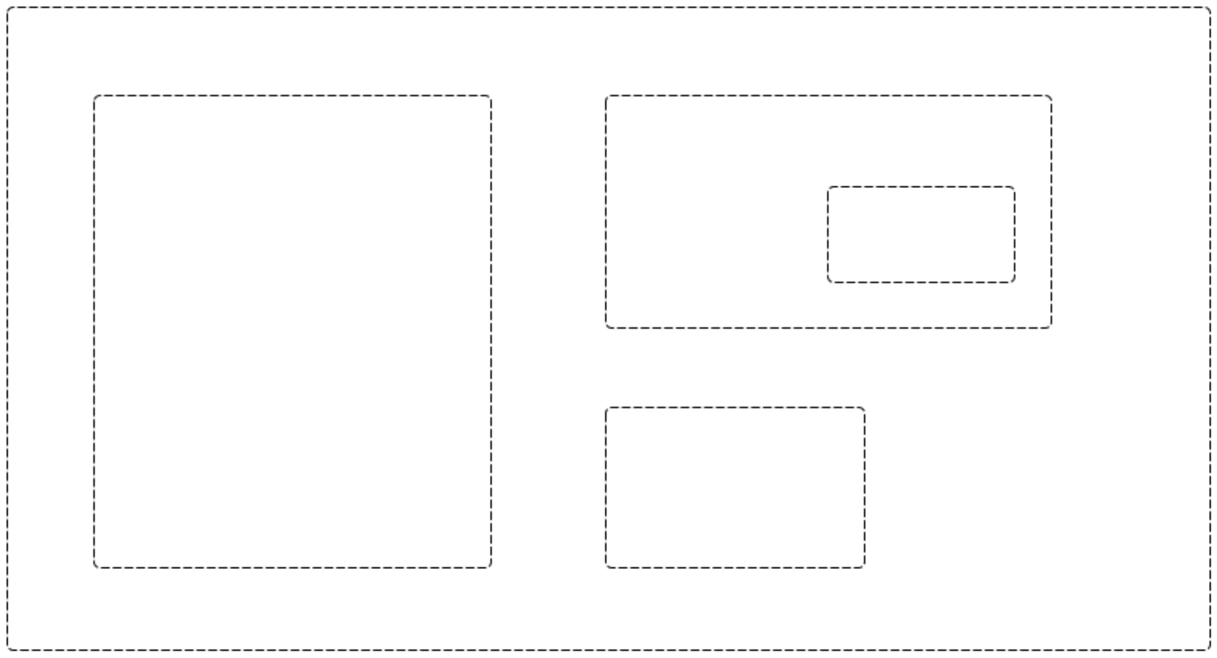
1. `<script>` section
2. `<main>` section
3. `<style>` section

In fact, almost all of svelte components has such 3 sections/parts, we can even further refine them into 2 groups:



- I would like to call the `<script>` section to be **State** section, although functions and logic things are also included in this section;
- as to `<main>` and `<style>` sections, I would like to group them together and call them **Templating** section, of course, we can always apply CSS to templating UI part by adding css to `<style>` section to form scoped css, or use TailwindCSS directly which we will talk about later.

Components can be composited so they may have a hierarchical structure:



we can import chidren components into parent components to form the hierarchy, say, we have A.svelte and B.svelte as children components, and in App.svelte, we can import them in `<script>` section:

```
<script>
import A from './A.svelte';
import B from './B.svelte';
...
```

```
</script>

<A/>
<B/>
...
```

we have been told that each svelte component file ends with .svelte, and their name should be Uppercase (This is a naming convention in svelte, so bear it in mind, please~). After importing, we can use them in Templating section, the previous `<main>` element is not a must, in fact, we can use any HTML elements, even svelte extended ones, component tags after import is one form of the extensions.

> TIP
>
> In fact, a svelte application is just a top svelte component with embedded sub components, usually, the **App.svelte**.

Component in hierarchy can communicate with each via several ways, for example:

1. via component properties (abbr. **props**);
2. via *Context*;
3. via stores (maybe the coolest feature of svelte);

oh, even Slot, with slot, we can customize children components by passing in custom templating content, not only data via properties.

So that's what make a svelte compnent:

- **File name convention**: component file must have a Uppercase file name and end with `.svelte` suffix.
- **Component Structure**: a svelte component has 3 sections(`<script>`, `<main>`, `<style>`) which can be refined into 2 groups(**State** Group and **Templating** Group).

- **Component Hierarchy and Communication**:
  - components can be composited and embedded into each others.
  - components can use props, Content, stores even slot to communicate with each other.

Now we can elaborate all of these things in the follwing chapters…

# Chapter 21

# Who am I

My name is **Fuqiang Wang**, I am the founder of Fuqiang Technology Ltd..

Besides this book, I also wrote *Unveil Spring* and *Unveil SpringBoot* which were published by HuaZhang(China) Press at 2009 and 2018. I am also the founder of **Canal** middleware of Alibaba which is one of the CDC [1] pioneers.

In Cloud Native Era, I am also TVP of Tencent Cloud.

More about me and my books can be found on my blog: https://afoo.me/books.html

The QRCode following is for WeiXin or Wechat:

[1]CDC: Change Data Capture, Debezium are recommended nowadays.

# Chapter 22

# Contact

You are welcome to leave me any comment on my blog: https://afoo.me/books.html, and share the link and books with your friends are appreciated.

- If you would like to follow me on twitter: @fujohnwang
- If you would like to send a mail: `i@afoo.me`