StuQueue
斯达克学院

从前端小工到中级工程师的必备技能

前端模版 & DSL

想当年没有模版的时候

不尽流出了眼泪

比惨大会

```
container.innerHTML = '<div>' + word + '</div>'
```

```
var div = document.createElement('div')
var p = document.createElement('p')
p.innerText = 'hello world'
div.appendChild(p)
document.body.appendChild(div)
```

后来大家觉得好麻烦

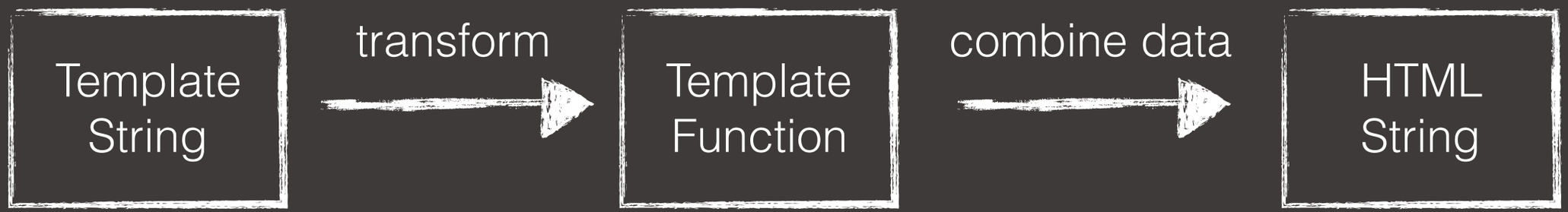
所以 jQuery 的作者就做了个小东西

JavaScript Micro-Templating

```
// Simple JavaScript Templating
// John Resig - https://johnresig.com/ - MIT Licensed
(function(){
  var cache = {};

  this.tmpl = function tmpl(str, data){
    // Figure out if we're getting a template, or if we need to
    // load the template - and be sure to cache the result.
    var fn = !/\W/.test(str) ?
      cache[str] = cache[str] ||
        tmpl(document.getElementById(str).innerHTML) :

      // Generate a reusable function that will serve as a temp
      // generator (and which will be cached).
      new Function("obj",
        "var p=[],print=function(){p.push.apply(p,arguments);};

        // Introduce the data as local variables using with(){}
        "with(obj){p.push('" +

        // Convert the template into pure JavaScript
        str
          .replace(/[\r\t\n]/g, " ")
          .split("<%").join("\t")
          .replace(/((^|%>)[^\t]*)'/g, "$1\r")
          .replace(/\t=(.*?)%>/g, "',$1,'")
          .split("\t").join("');")
          .split("%>").join("p.push('")
          .split("\r").join("\\'")
      + "');}return p.join('');");

    // Provide some basic currying to the user
    return data ? fn( data ) : fn;
  };
})();
```

```
<script type="text/html" id="item_tmpl">
  <div id="<%=id%>" class="<%=(i % 2 == 1 ? " even" : "")%>">
    <div class="grid_1 alpha right">
      <img class="righted" src="<%=profile_image_url%>"/>
    </div>
    <div class="grid_6 omega contents">
      <p><b><a href="/<%=from_user%>"><%=from_user%></a>:</b> <
    </div>
  </div>
</script>
```

```
var results = document.getElementById("results");
results.innerHTML = tmpl("item_tmpl", dataObject);
```


Template
String
transform
Template
Function
combine data
HTML
String

模版语言的出现，体现了神马?

DOM API 强大但不好维护，
前端做了那么多 View 相关的库本质上就是为了屏蔽底层 DOM 操作

功能 VS 性能

功能：handlebars 强的的 Blocks 扩展能力

```
<div class="entry">
  <h1>{{title}}</h1>
  <div class="body">
    {{#bold}}{{body}}{{/bold}}
  </div>
</div>
```

```
Handlebars.registerHelper('bold', function(options) {
  return new Handlebars.SafeString(
      '<div class="mybold">'
      + options.fn(this)
      + '</div>');
});
```

http://handlebarsjs.com/

性能：以 doT 为代表

所以怎么做到性能比较好呢?

性能：doT 为代表

- 首先你得小
- 其次，你可以线下预编译，成为静态的 js 文件

```
var render = (function () {
    var cache =
    "var $out = [];\
    with ($data) {\
        $out.push('<h3>');\
        if (typeof content === 'string') {\
            $out.push(content);\
        }\
        $out.push('</h3>');\
    }\
    return $out.join('');"

    return function (data) {
        var fn = new Function('$data', cache);
        return fn(data);
    }
})();
```

- 还有你可以不走正规的词法分析，走正则
- 没有什么扩展能力
- 再则你可以不使用with

性能：doT 为代表

高性能JavaScript模板引擎原理解析

doT 源码

想一想🤔，只用正则替换有什么问题?

MVVM们 和 React 的冲击

DOM Template & JSX

DOM Template: 反正就是没有任何东西不能在DOM上被表征的

```
1    <div id="app-3">
2      <p v-if="seen">现在你看到我了</p>
3      <ol>
4        <li v-for="todo in todos">
5          {{ todo.text }}
6        </li>
7      </ol>
8    </div>
```

DOM Template

- 比较好静态分析
- 可以把模版直接插入DOM，不会有太奇葩的事情发生
- 不需要写词法分析，一般可以直接用现成的HTML parser

JSX: JS上可以写标签

```
var MyComponent=React.createClass({
  getInitialState: function() {
       return {clickNum: 0};
  },
  handleClick:function(){
    var num=this.state.clickNum;
    num++;
    this.setState({clickNum:num});
  },
  render: function() {
   return (
      <div>
        <h1 onClick={this.handleClick}>Hello {this.props.name}!</h1>
        <h2 style={{color:'red'}}>点击{this.props.name}次数: {this.state.clickNum}</h2>
      </div>
    );
  }
});
ReactDOM.render(
  <div>
     <MyComponent name="张三" />
     <hr/>
     <MyComponent name="李四" />
  </div>,
  document.getElementById('example')
);
```

JSX

- JSX 打破了内容和逻辑分离这个 web 传统的分离方式
- 尽量减少了概念的引入
- 通过 component 化来拆解应用

Web Component

样式（CSS）

内容（HTML）

逻辑（Javascript）

样式（CSS）

内容（HTML）

逻辑（Javascript）

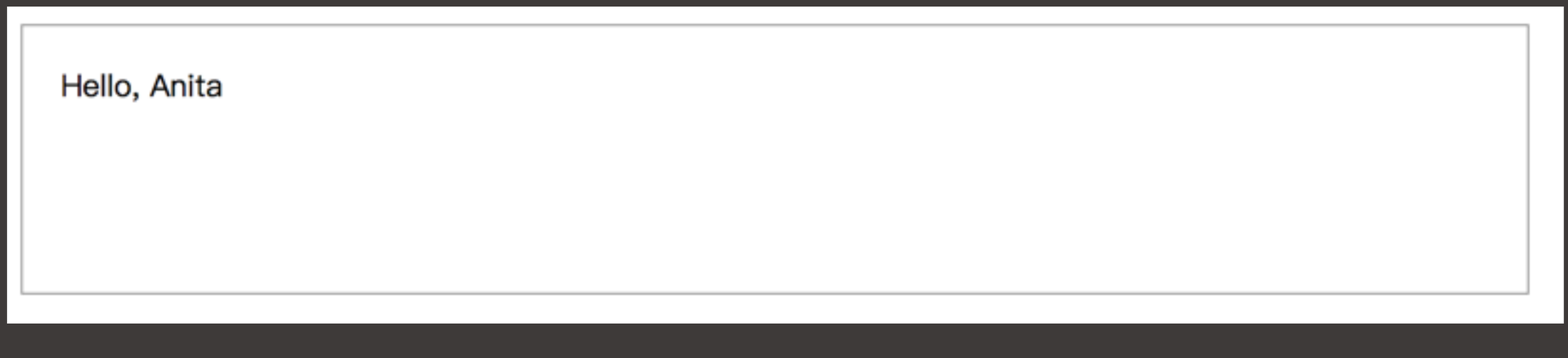
Component

custom element

```
class HelloElement extends HTMLElement {
  // Monitor the 'name' attribute for changes.
  static get observedAttributes() {return ['name']; }

  // Respond to attribute changes.
  attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == 'name') {
      this.textContent = `Hello, ${newValue}`;
    }
  }
}

// Define the new element
customElements.define('hello-element', HelloElement);
```

```
<hello-element name="Anita"></hello-element>
```

custom element



但我们依然要关注 DOM 实现细节

Polymer = MVVM + Web Component

面向未来的探索

反正 Polymer 我不准备讲，
因为目前还没有什么地方可以用

https://www.polymer-project.org/

大家可以自行体验下

但我们需要一个兼容性足够好的类Polymer方案

DSL

或者说，我要给你们找个轮子练习下……

案例分析：找个理由，为啥要弄 DSL

我们先想象一般会有什么阻力吧

推行新技术一般会遇到的阻力

- 旧的技术能完成目前的需求，为什么要换，增加翻新成本?
- 我们已经熟悉了现有的开发方式，为什么要换，增加学习成本?

再挖掘核心价值！

找价值

1 业务

2 人员

3 技术

寻找业务热点

我们要一个好旗帜🚩

planA：支付宝 DSL 事件

planA：支付宝 DSL 事件

- 我们应不应当有自己的面向开发者的 DSL?
- 或者……

planB：Facebook Licence 事件

Apache Foundation bans use of Facebook BSD+Patents licensed libraries like React.js

简单说就是 React.js 有不对等防御性协议，
我们要不要规避其风险?

人员原因

planC：从动态运营到动态开发资源

动态化运营故事

planC：从动态运营到动态开发资源

则，开发资源调配会变得方便

planC：从动态运营到动态开发资源

但全栈肯定不是什么都精通，
而是什么都能做，
所以要降低开发难度！

DSL 是降低前端开发难度的良药！

planD：从分裂走向统一

前端A团队技术栈

前端B团队技术栈

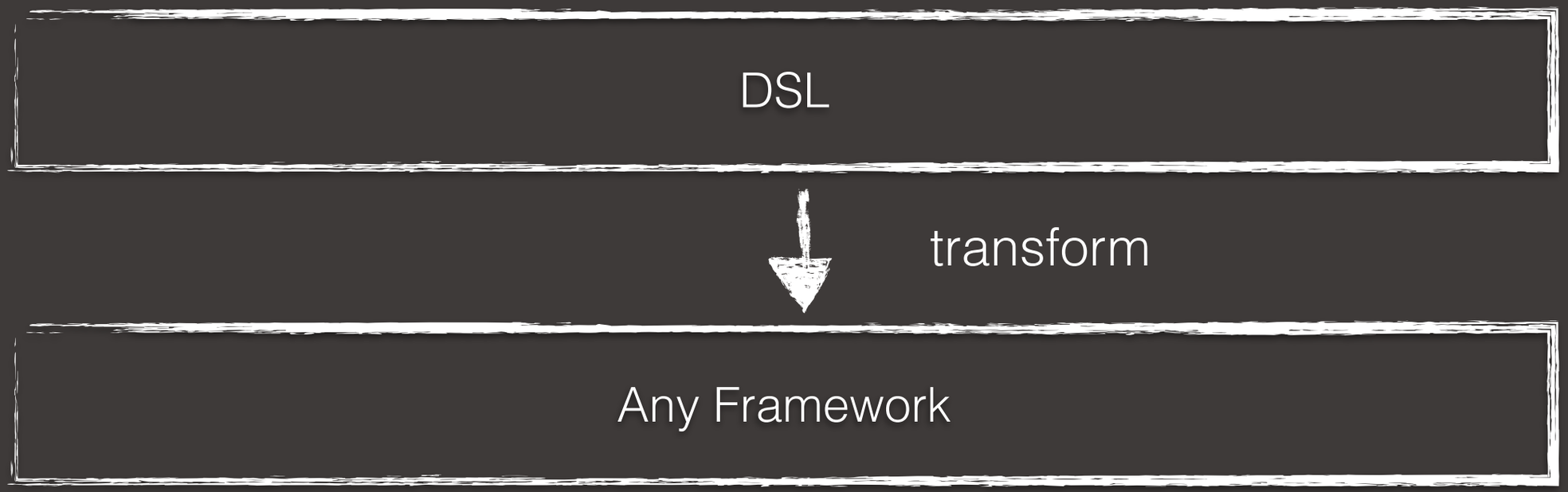
统一技术展

技术原因

planE：上层稳定，底层变动不影响上层

- 2014，jQuery 很差，Angular 多好～
- 2015，Angular很差，React很好～
- 2016，React全家桶太复杂，Vue很简单～
- 2017?

planE：上层稳定，底层变动不影响上层



基于这套架构可以做到换底！

根据不同的人，我们可以摆出不同的 Plan

价值在何处?

- 说出故事，找到价值，聚合资源，一件事是否能做的出色跟资源有很大关系
- 一件事情，说不清楚价值，没人认可，做的再好也白做
- 很多事情要跨团队配合才能完成，必须有这样的人来推进事情发生

OK，我们案例分析完了

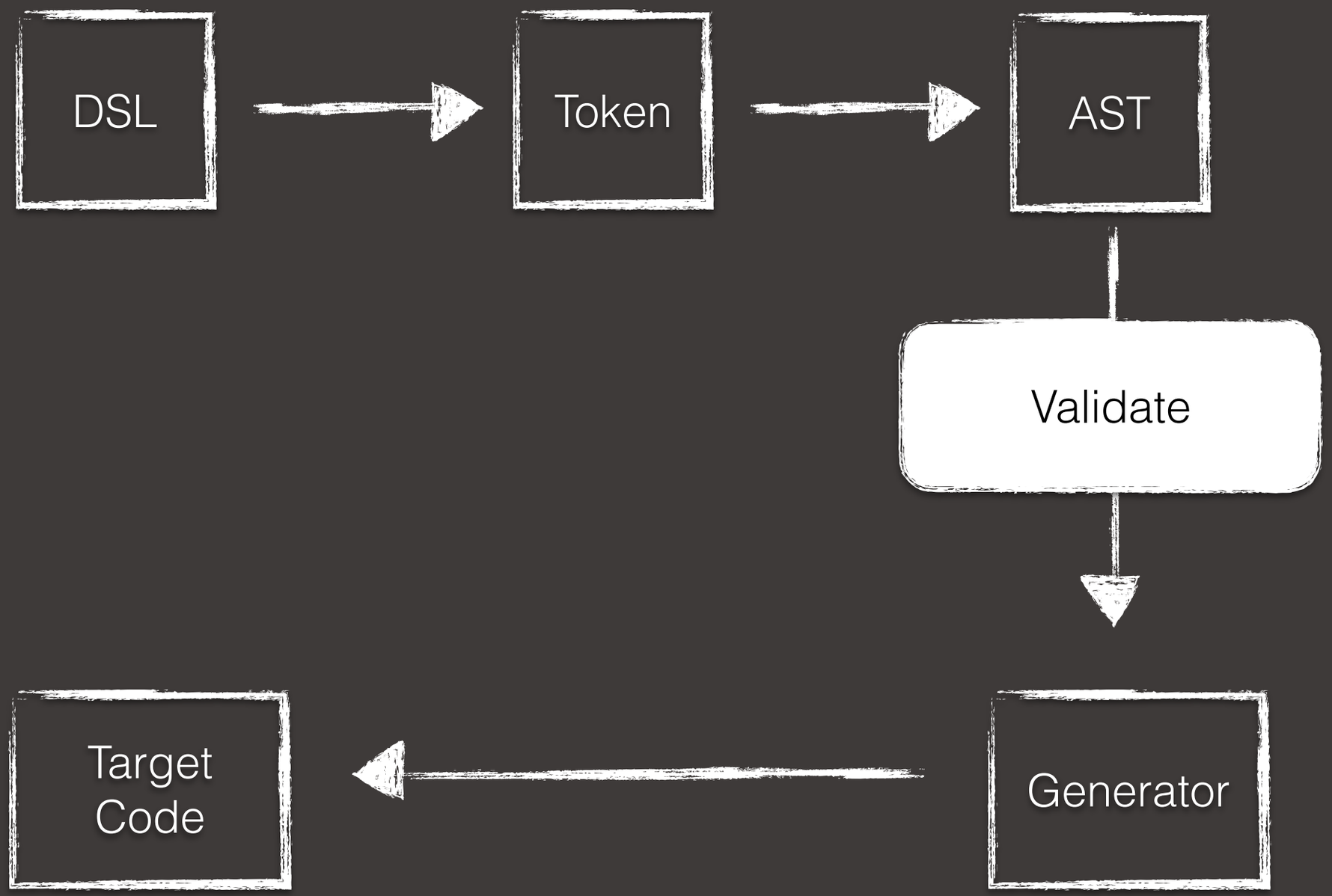
可能有同学觉得很虚，可以在工作中遇到同样问题时候在回来看录播

DSL 设计与实现是我们九月的大作业

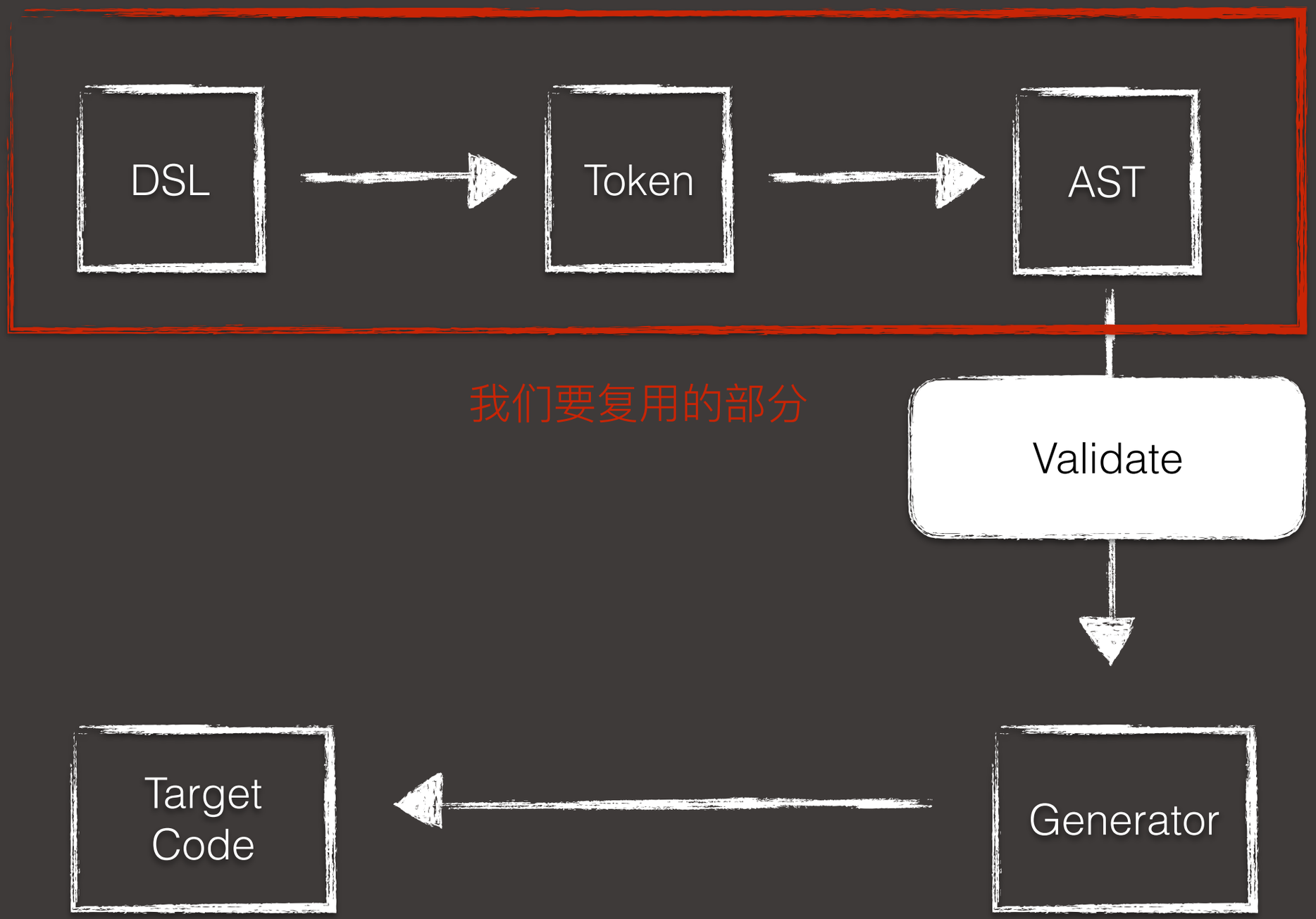
我们先从 svelte 开始

https://svelte.technology/

Svelte 基本过程



Svelte 基本过程



StuQ 斯达克学院

THANKS!

扫码了解更多