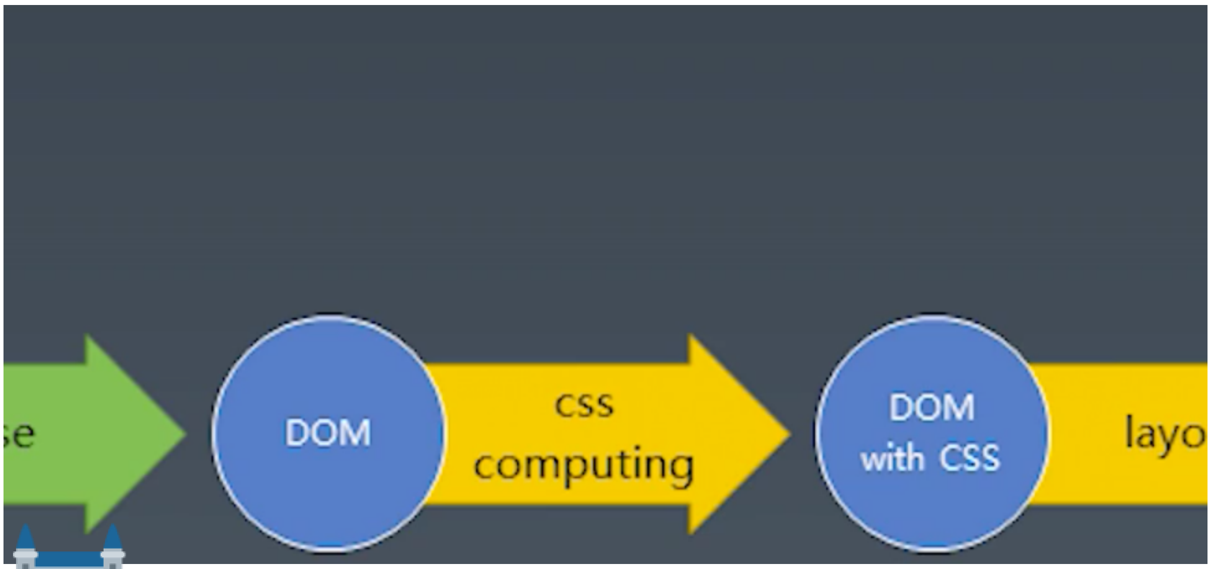


CSS计算

Date	@Aug 1, 2020

收集CSS规则

在style标签结束时处理其中的文本标签：

```
if (token.type === "endtag") {
    currentTextNode = null;
    if (token.tagName != node.name) {
        throw new Error("Tag start end doesn't match!")
    }else {
        if (token.tagName === 'style') {
            addCSSRules(node.children[0].content);
        }
        stack.pop();
    }
}
```

```
function addCSSRules(cssNode) {
    console.log(cssNode);
}
```

安装npm包: css，解析CSS文本

```
const cssParser = require('css');
...
function addCSSRules(cssNode) {
    let ast = cssParser.parse(cssNode);
    console.log(ast);
}
```

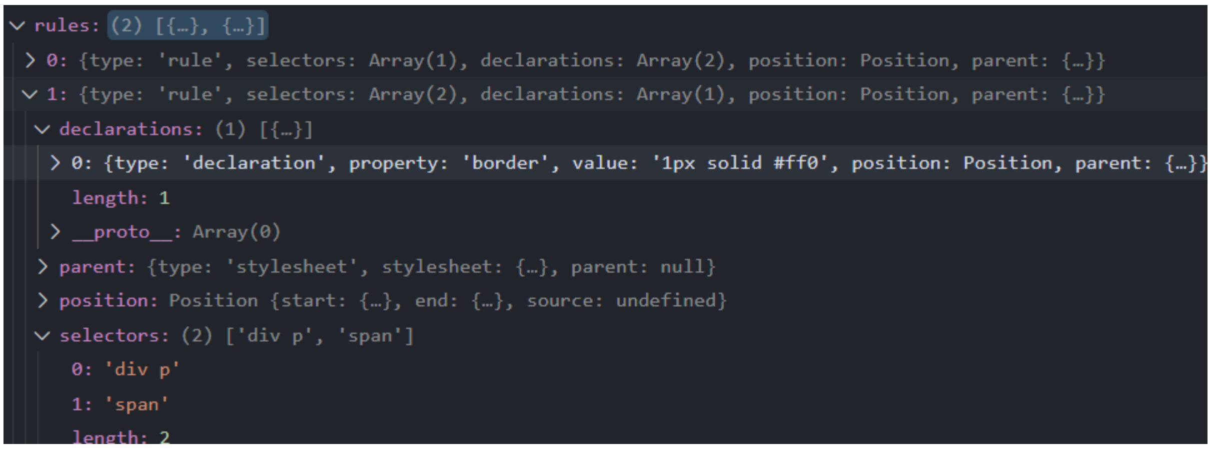
```
{ type: 'stylesheet',
  stylesheet: { source: undefined, rules: [ [Object] ], parsingErrors: [] } }
```

css转换规则

style内容

```
<style>
    p.headline{
        color: "red";
        font-size: 14px;
    }
    div p, span{
        border: 1px solid #ff0;
    }
</style>
```

转换后的`stylesheet`



```
rules: (2) [{…}, {…}]
  0: {type: 'rule', selectors: Array(1), declarations: Array(2), position: Position, parent: {…}}
  1: {type: 'rule', selectors: Array(2), declarations: Array(1), position: Position, parent: {…}}
    declarations: (1) [{…}]
      0: {type: 'declaration', property: 'border', value: '1px solid #ff0', position: Position, parent: {…}}
      length: 1
      __proto__: Array(0)
    parent: {type: 'stylesheet', stylesheet: {…}, parent: null}
    position: Position {start: {…}, end: {…}, source: undefined}
    selectors: (2) ['div p', 'span']
      0: 'div p'
      1: 'span'
      length: 2
```

计算css

- 在检测到每一个startTag时立即计算其css(只能处理head标签中的收集好的css规则)

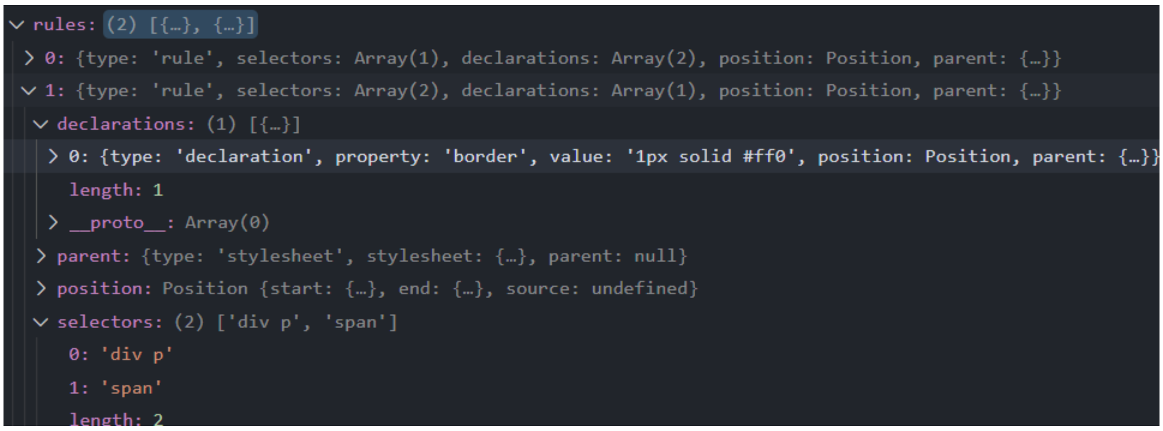
```
if (token.type === "starttag") {
    ...
    let element = {
        name: token.tagName,
        type: 'element',
        children: [],
        attaributes: []
    }
    computeCSS(element);
    ...
}
```

- 计算时必须知道当前元素的所有父元素才能判断css规则是否匹配
- 必须从内向外匹配规则，因此父元素需要逆序判断

```
// 计算css
function computeCSS(element) {
    let parents = stack.slice().reverse();
}
```

选择器与元素的匹配

前面收集css的步骤收集到的 `stylesheet` 包含一个rules数组。每条rule的selectors也是一个数组，对应着css中由逗号分隔的选择器。



```
∨ rules: (2) [{…}, {…}]
  > 0: {type: 'rule', selectors: Array(1), declarations: Array(2), position: Position, parent: {…}}
  ∨ 1: {type: 'rule', selectors: Array(2), declarations: Array(1), position: Position, parent: {…}}
    ∨ declarations: (1) [{…}]
      > 0: {type: 'declaration', property: 'border', value: '1px solid #ff0', position: Position, parent: {…}}
        length: 1
      > __proto__: Array(0)
    > parent: {type: 'stylesheet', stylesheet: {…}, parent: null}
    > position: Position {start: {…}, end: {…}, source: undefined}
    ∨ selectors: (2) ['div p', 'span']
        0: 'div p'
        1: 'span'
        length: 2
```

```
// 计算css
function computeCSS(element) {
    let parents = stack.slice().reverse();
    for (let rule of rules) {
        let selectors = rule.selectors;
        for (let selector of selectors) {
            let selections = selector.split(" ").reverse();
            if (!match(element, selections[0])) continue;
            if (loopCheck(parents, selections.slice(1))){
                console.log("已经匹配上一个选择器，规则是:", rule.declarations);
                break;
            }
        }
    }
}

// 计算选择器是否和元素匹配
function match(element, selector) {
    return element.tagName === selector;
}

function loopCheck(elements, selections) {
    let j = 0;
    for (let i = 0;j < selections.length && i < elements.length;i ++) {
        if (match(elements[i], selections[j])) {
            j ++;
        }
    }
    return j === selections.length;
}
```

计算选择器是否和元素匹配

不考虑复合选择器：

- 类选择器
- id选择器
- 标签选择器

升级：

如何支持复合选择器

e.g. `p.main` , `div#app`

从rule的declarations生成元素的computed属性

```
// 从rule的declarations生成元素的computed属性
function getComputedStyle(declarations, element) {
    let computedStyle = element.computedStyle;
    for(let declear of declarations) {
        computedStyle[declear.property] = declear.value
    }
}
```

> 问题：未处理优先级

处理选择器优先级

css优先级`specificity`四元组：

[inline, id, class, tag]

几个例子：

```
div p#app{
  color: red;    /*不生效 [0, 1, 0, 1]*/
}

div content p#app{
  color: blue;   /*生效 [0, 1, 0, 2]*/
}

div content p.headline {
  color: yellow;   /*不生效  [0, 0, 1, 2]*/
  font-weight: bolder;    /*生效*/
}
```

https://codepen.io/zjlyyq/pen/NWxQMNQ

```
// 计算css优先级
function getSpecificity(selector) {
    let selections = selector.split(" ").reverse();
    let p = [0 ,0, 0, 0];
    for (let selection of selections) {
        if (selection.charAt("0") === "#") {
            p[1] += 1;
        }else if (selection.charAt(0) === ".") {
            p[2] += 2;
        }else {
            p[3] += 1;
        }
    }
    return p;
}
```