HTML & CSS 預處理器

Preprocessor 介紹

HTML -> PUG 轉換

```
<div id="app">文字內容</div>

<div class="content">文字內容</div>

<a class="link" href="##">連結文字</a>

<div id="parent">
  <div class="child child1">
    大兒子
    <div class="grandchild">孫子</div>
  </div>
  <div class="child child2">二兒子</div>
  <div class="child child3">三兒子</div>
</div>
<div id="parent2" class="nice"></div>
```

->

```
div(id="app") 文字內容

div(class="content") 文字內容

a(class="link" href="##") 連結文字

div(id="parent")
  div(class="child child1")
    | 大兒子
    div(class="grandchild") 孫子
  div(class="child child2") 二兒子
  divclass="child child3") 三兒子

div(id="parent2" class="nice")
```

->

```
#app 文字內容

.content 文字內容

a.link(href="##") 連結文字

#parent
  .child.child1
    | 大兒子
    .grandchild 孫子
  .child.child2 二兒子
  .child.child3 三兒子

#parent2.nice
```

CSS -> SCSS -> SASS 轉換

```
.parent {
  display: block;
  color: #fff;
}
.parent .child {
  width: 300px;
}
.parent .child.child1 {
  background: #000;
}
.parent .child.child2 {
  background: #eee;
}
```

->

```
.parent {
  display: block;
  color: #fff;
  .child {
    width: 300px;
    &.child1 {
      background: #000;
    }
    &.child2 {
      background: #eee;
    }
  }
}
```

->

```
.parent
  display: block
  color: #fff
  .child
    width: 300px
    &.child1
      background: #000
    &.child2
      background: #eee
```

PUG 定義變數

```
- const num = 2
- const name = 'Jarvis'

div(class=`person person${num}`) #{name}

// 有加 - 表示在 PUG 使用 JS 語法
```

PUG 迴圈

```
each val in [1, 2, 3, 4, 5]
  div(class=`ball ball${val}`)

- const num = 50
- for (let i = 1; i < num; i++)
  div(class=`ball ball${i}`)
```

SCSS 定義變數

```scss
$color-main: #333;
$color-sub: #eee;

.class-name {
  background-color: $color-main;
}
```

SCSS 模組設置

```scss
@mixin size($w: 100px, $h: 100px) {
  width: $w;
  height: $h;
}
.class-name {
  @include size(20px, 20px);
}
```

SCSS 迴圈

```scss
@for $i from 1 through 30 {
  $size: random(30)+30;
  .class-name#{$i} {
    width: $size;
    height: $size;
  }
}
```

SCSS 判斷式

```scss
@if (color1 == color2) {
  $color1: nth($colors, random(5));
}
.class-name#{$i} {
  color: $color1;
}
```

CSS 滑鼠互動與動畫基礎

透過 hover 及 active 製作互動效果

CSS 滑鼠互動

```
// 滑鼠進入
.class-name:hover {
  opacity: 1;
}
// 滑鼠進入 .class-name 時，後代 .child-class 作用
.class-name:hover .child-class {
  opacity: 1;
}
```

```
// 滑鼠點住
.class-name:active {
  color: #333;
}
// 滑鼠點住 .class-name 時，後代 .child-class 作用
.class-name:active .child-class {
  color: #333;
}
```

詳細說明（以 hover 為例）：

1. 滑鼠進入哪個元素後作用，就將 :hover 加在此元素上（滑鼠進入誰，就將誰加上 :hover）
2. transition 是加在樣式有變化的元素上
3. 如果想要滑鼠進入、移出皆有過渡效果，須將 transition 設在原始狀態上
4. 如果想要滑鼠進入、移出時的過度時間或延遲等不同，個別設定 transition 在對應選擇器上即可，如下：

```
.class-name:hover .child-class
  opacity: 1
  transition: 0.5s 0.3s  // 滑鼠進入時，將延遲 0.3 秒才顯示，並花 0.5 秒過渡
.child-class
  opacity: 0
  transition: 0.3s
```

過渡設定（transition）

transition-property:

指定屬性

transition-duration:

過渡時間

transition-delay:

延遲過渡

transition-timing-function:

定義加速度

// ease（預設）
// linear（線性）
// ease-in（平滑進入）
// ease-out（平滑結束）
// ease-in-out（平滑進出）
// cubic-bezier(n,n,n,n)（自定義）

```
transition: width 4s 1s ease-in;
```

屬性縮寫

CSS 影格動畫製作

透過 keyframes 設定元素動畫

CSS 影格設置

```
@keyframes aniName {
  from {
    width: 0;
  }
  to
    width: 100px;
  }
}
```

```
@keyframes aniName {
  0% {
    width: 0;
  }
  50% {
    width: 100px;
  }
  100% {
    width: 50px;
  }
}
```

動畫屬性設定（animation）

animation-name:

動畫名稱

animation-duration:

動畫時間長度

animation-delay:

時間延遲

animation-timing-function:

定義加速度
// ease（預設）
// linear（線性）
// ease-in（平滑進入）
// ease-out（平滑結束）
// ease-in-out（平滑進出）
// cubic-bezier(n,n,n,n)（自定義）

animation-iteration-count:

循環次數
// infinite（無限）

animation-direction:

時間方向性
// alternate（來回播放）
// reverse（反向播放）

animation-fill-mode:

播放前、後位置定義
// backwards（動畫從0%開始）
// forwards（動畫結束後停在100%）
// both（結合backwards及forwards）

animation-play-state:

動畫狀態
// running（預設）
// paused（暫停）

```
animation: name 4s 1s ease-in infinite alternate both;
```

屬性縮寫

專案實作

製作煙火動畫