DECIPHERING THE CSS PROPERTY SYNTAX

By Chen Hui Jing / @hj_chen

HAVE YOU EVER SEEN A CSS SPECIFICATION?

Meet CSS Grid Layout Module Level 1.

BACKUS NAUR FORM (BNF)

- Introduced by **John Backus** and **Peter Naur**
- A context-free notation method to describe the syntax of a language.
- The CSS property value syntax is loosely based on BNF notation.

```
<symbol> ::= __expression__
```

The stuff on the left can be replaced by the stuff on the right.

A BNF SANDWICH

A sandwich consists of a *lower slice of bread*, *mustard* or *mayonnaise*; optional *lettuce*, an optional slice of *tomato*; two to four slices of either *bologna*, *salami*, or *ham* (in any combination); one or more slices of *cheese*, and a *top slice of bread*.

```
sandwich ::= lower_slice [ mustard | mayonnaise ] lettuce? tomato? [ bologna |
salami | ham ]{2,4} cheese+ top_slice
```

Analogy from How to Read W3C Specs.

COMPONENT VALUE TYPES

Value type	Description	Example
Keyword values	Actual value used; No quotation marks or angle brackets	`auto` or `none`
Basic data types	To be replaced with actual values; Angle brackets	`<length>` or `<percentage>`
Property data type	Uses same set of values as defined property, usually used for shorthand property definitions; Quotation marks within angle brackets	`<'grid-template-rows'>` or `<'flex-basis'>`
Non-property data type	Set of values is defined somewhere else in the specification, usually near its first appearance; Angle brackets only	`<line-names>` or `<track-repeat>`

COMPONENT VALUE COMBINATORS

SPACE-SEPARATED LIST OF VALUES

All values must occur in specified order

```
<'property-name'> = value1 value2 value3
```

↓

```
.selector { property: value1 value2 value3; }
```

&&

All values must occur, order doesn't matter

```
<'property-name'> = value1 && value2
```

↓

```
.selector { property: value1 value2; }
.selector { property: value2 value1; }
```

|

Only 1 value must occur

```
<'property-name'> = value1 | value2 | value3
```

↓

```
.selector { property: value1; }
.selector { property: value2; }
.selector { property: value3; }
```

||

1 or more values must occur, order doesn't matter

`<'property-name'> = value1 || value2 || value3`

↓

```
.selector { property: value3; }
.selector { property: value2 value3; }
.selector { property: value1 value2 value3; }
 and so on...
```

[]

Components belong to a single grouping

```
<'property-name'> = [ value1 | value2 ] value3
```

↓

```
.selector { property: value1 value3; }
.selector { property: value2 value3; }
```

COMPONENT VALUE MULTIPLIERS

?

Optional value, can occur 0 or 1 time

```
<'property-name'> = value1 [, value2 ]?
```

↓

```
.selector { property: value1; }
.selector { property: value1, value2; }
```

*

Optional value, can occur 0 or many times, multiple values are comma-separated

```
<'property-name'> = value1 [, <value2>]*
```

↓

```
.selector { property: value1; }
.selector { property: value1, <value2>; }
.selector { property: value1, <value2>, <value2>, <value2>; }
 and so on...
```

+

Can occur 1 or many times, multiple values are space-separated

`<'property-name'> = <value>+`

↓

```
.selector { property: <value>; }
.selector { property: <value> <value>; }
.selector { property: <value> <value> <value> <value>; }
 and so on...
```

{ }

Value occurs times, multiple values are space-separated

```
<'property-name'> = <value>{2}
```

↓

```css
.selector { property: <value> <value>; }
```

{ , }

Value occurs at least times, at most times, multiple values are space-separated

```
<'property-name'> = <value>{1,3}
```

↓

```
.selector { property: <value>; }
.selector { property: <value> <value>; }
.selector { property: <value> <value> <value>; }
```

{ ,}

Value occurs at least times, no maximum limit, multiple values are space-separated

```
<'property-name'> = <value>{1,}
```

↓

```
.selector { property: <value>; }
.selector { property: <value> <value> <value>; }
 and so on...
```

# #

Value occurs 1 or many times, multiple values are comma-separated

```
<'property-name'> = <value>#
```

↓

```
.selector { property: <value>; }
.selector { property: <value>, <value>; }
.selector { property: <value>, <value>, <value>, <value>; }
 and so on...
```

[ ]!

Values in grouping are required, at least 1 value must occur

```
<'property-name'> = <value1> [ <value2> | <value3> ]!
```

↓

```
.selector { property: <value1> <value2>; }
.selector { property: <value1> <value3>; }
```

CONVOLUSION MAX

BOX-SHADOW

```
none | <shadow>#
```

where

```
<shadow> = inset? && <length>{2,4} && <color>?
```

- all 3 values must occur, in any order
- inset is optional
- at least 2 length values, at most 4
- color value is optional
- entire set can occur multiple times, comma-separated

BACKGROUND

```
<bg-layer># , <final-bg-layer>
```

where

```
<bg-layer> = <bg-image> || <position> [ / <bg-size> ]? || <repeat-style> || <attachment> || <box> || <box>
```

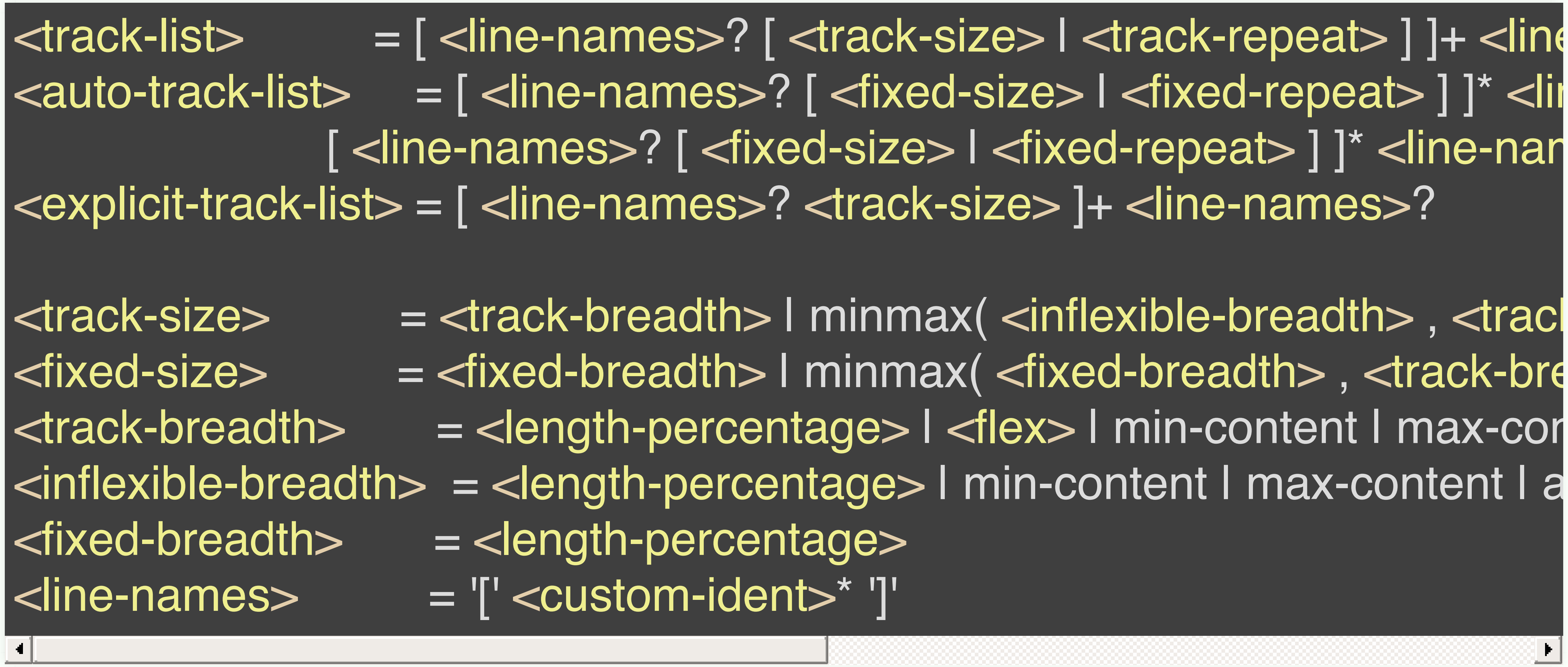
```
<final-bg-layer> = <'background-color'> || <bg-image> || <position> [ / <bg-size> ]? || <repeat-style> || <attachment> || <box> || <box>
```

- at least 1 value must occur, the rest can OTOT
- for <position>, can optionally include <bg-size>
- entire set for <bg-layer> can occur multiple times, comma-separated
- only <final-bg-layer> can have <'background-color'>

GRID-TEMPLATE-COLUMNS / GRID-TEMPLATE-ROWS

`none | <track-list> | <auto-track-list>`

where



```
<track-list>          = [ <line-names>? [ <track-size> | <track-repeat> ] ]+ <line
<auto-track-list>     = [ <line-names>? [ <fixed-size> | <fixed-repeat> ] ]* <li
                        [ <line-names>? [ <fixed-size> | <fixed-repeat> ] ]* <line-na
<explicit-track-list> = [ <line-names>? <track-size> ]+ <line-names>?

<track-size>          = <track-breadth> | minmax( <inflexible-breadth> , <trac
<fixed-size>          = <fixed-breadth> | minmax( <fixed-breadth> , <track-bre
<track-breadth>       = <length-percentage> | <flex> | min-content | max-co
<inflexible-breadth>  = <length-percentage> | min-content | max-content | a
<fixed-breadth>       = <length-percentage>
<line-names>          = '[' <custom-ident>* ']'
```

I is troll

HERE'S A CHEATSHEET, YOU CAN PRINT IT.

FURTHER READING

- CSS Value Definition Syntax
- Understanding the CSS Specifications by Elika Etemad
- How to Read W3C Specs by J. David Eisenberg
- CSS reference by MDN
- Understanding The CSS Property Value Syntax by Russ Weakley

THE END

http://www.chenhuijing.com

@hj_chen

@hj_chen

@huijing