Different strategies for using @support

The @supports at-rule allow us to test if something is supported in CSS.

The version of `@supports` that we see most often is testing for a rule inside a selector.

```
@supports (display: grid) {
  /*
    Styles for when grid is supported
  */
}
```

We also have three logical operators: `and`, `or` and `not` that enable more complex queries.

We can use `and` to test for more than one property. For example, we can test for grid and subgrid support.

```
@supports
  (display: grid) and
  (display: subgrid) {
    /* code goes here */
  }
```

The `or` operator comes in handy when dealing with vendor prefixes. As long as they work the same way we can put them in a `@supports` at-rule.

```
@supports (
  (display: table) and
  (display: table-cell)) {
    /*
      Do something when the browser
      supports both properties
```

```
    */
  }
```

The `not` operator negates the parameter that follows. In essence we're telling the browser to do something if the property **is not** supported.

```
@supports not (display: table) {
  /*
    code for browsers that don't
    support display: table
  */
}
```

Selector list argument of `:nth-child` pseudo-classes

Note:

As of this writing, the feature is only supported in Safari

All our examples work with properties and values like `(perspective: 10px)`. Most of the time this is all that we will need.

But there are times when wed want to test for support of a selector like `:nth-child(-n+3 of li.important)`

Chris Coyier wrote @supports selector() to explain how the `selector()` functions works inside a `@support` at-rule. The idea is that you check if a selector is supported and add rules if they are:

```
@supports selector(:nth-child(1 of .foo)) {
  li:nth-child(odd of .list-item) {
    background: lightgoldenrodyellow;
```

```
  }
}
```

The selector we test for doesn't have to be the same selector that we want to use, they just have to be valid CSS selectors.

As with all feature queries using `@supports`, you can have style multiple elements when the query matches.

You can also use the `not` operator to test if a feature is not supported.

```
/* test if the selector is supported */
@supports selector(:nth-child(1 of .foo)) {
  li {
    padding: 0.25em;
  }

  li:nth-child(odd of .list-item) {
    background: lightgoldenrodyellow;
  }

  li.separator {
    list-style: none;
    margin: 0.25em 0;
  }
}

/* test if the selector is not supported */
@supports not selector(:nth-child(1 of .foo)) {
  li.separator {
    height: 1px;
    list-style: none;
    border-top: 1px dashed purple;
    margin: 0.25em 0;
  }
}
```