@property Rule

CSS Properties and Values API Level 1 provides a CSS equivalent solution to `CSS.registerProperty`, the `@property` at-rule.

Unfortunately it is not implemented by any browser yet.

The reason why I'm writing about it now is that Chrome filed an Intent to Ship the feature.

What problem are Houdini Custom Properties trying to solve?

Custom properties are treated as strings, so developers have to jump through hoops to make them work and have to do things that are not necessarily intuitive to create the effects they need.

```
:root {
  --base-font-size: 16;
  --base-text-color: '#aaa'#aaa'body {
  font-size: calc(var(--base-font-size) * 1px);
  color: var(--base-text-color);
}
```

Another thing to consider is that, because all custom properties are treated as strings they not animatable and cannot be validated since the validator has no way of knowing what is the actual value of the property it's parsing.

Houdini Custom Properties

Houdini makes it easier to deal with the drawbacks of custom properties and provides, in my opinion, a better developer experience.

The first version of Houdini Custom Properties is written in JavaScript and it provides additional information about the properties that makes them easier to work with.

The items that we must have on each custom property declaration:

- **name**: the name of the custom property, including the two dashes (–)
- **syntax**: one or more of the valid syntax names available
    - An optional multiplier (either +, #) immediately after the syntax name
    - The separator character (|) between the values. This is only required if there is more than one value on the syntax
- **initial value**: The default value for the property. This includes the type of the value we're using
- **inherits**: Whether the property will propagate its value down the tree

The first example will be used for font sizing and can take either a length or percentage value using the `<length-percentage>` shorthand syntax. The default value is 16px and it will inherit down the tree unless the CSS author overrides it.

```
CSS.registerProperty({
  name: "--base-font-size",
  syntax: "<length-percentage>",
  initialValue: "16px",
  inherits: true
});
```

The second example is a color custom property that uses a three-digit RGB color as the default value but can use any color syntax allowed in CSS Color Module Level 3 and CSS Color Module Level 4 (currently a working draft). This property will not inherit down the tree.

```
CSS.registerProperty({
  name: "--my-color",
  syntax: "<color>",
  initialValue: "#333",
  inherits: false
});
```

Houdini Custom Properties in CSS

One of the main reasons why I've refrained from using Houdini Custom Properties

is that, until now, they must be defined in Javascript and they add an additional script, either inline or external, to the each page of the site you use them on.

The new CSS `@property` at-rule takes the same values as the JavaScript version and needs to be placed in one stylesheet linked to the pages.

The equivalent CSS @property version of the JavaScript examples are shown below.

```
@property --base-font-size {
  syntax: "<length-percentage>";
  i"<length-percentage>"  inherits: true;
}

@property --base-text-color {
  syntax: "<color>";
  initialValue: "#333";
  inherits: false;
}
"<color>"
```

The advantages of using Houdini-style custom properties is we can animate and validate them without having to jump through the hoops we have to today.

The downside is that we have to write more code (either CSS or JavaScript) and still provide a fallback for browsers that don't support custom properties or Houdini APIs yet.