LEaner Style Sheets

LESS (CSS)

More from LESS - Kyle Lin

Period 8 - Softdev

Introduction

Introduction

Cascading Stylesheets (CSS) and web design in general aren't considered 'programming' by the vast majority of the Computer Science field. You don't define functions, you don't have variables, and you they're simply displaying elements.

Introduction

Cascading Stylesheets (CSS) and web design in general aren't considered 'programming' by the vast majority of the Computer Science field. You don't define functions, you don't have variables, and you they're simply displaying elements.

LESS (Leaner Style Sheets) lets you write cleaner and more consistent CSS.

Introduction

Cascading Stylesheets (CSS) and web design in general aren't considered 'programming' by the vast majority of the Computer Science field. You don't define functions, you don't have variables, and you they're simply displaying elements.

LESS (Leaner Style Sheets) lets you write cleaner and more consistent CSS.

Vanilla CSS is pretty hard to write and keep track of. Having multiple elements maintain the same color scheme can be pretty tedious, most if not all of us have been there before. That is where CSS Pre-Processors come in.

Introduction

Although you need to compile your stylesheets to plain CSS beforehand, they add a variety of benefits.

Introduction

Although you need to compile your stylesheets to plain CSS beforehand, they add a variety of benefits.

- Variables to keep constants throughout your stylesheet

Introduction

Although you need to compile your stylesheets to plain CSS beforehand, they add a variety of benefits.

- Variables to keep constants throughout your stylesheet
- Calculation of values for more precision

Introduction

Although you need to compile your stylesheets to plain CSS beforehand, they add a variety of benefits.

- Variables to keep constants throughout your stylesheet
- Calculation of values for more precision
- Reuse styles from other elements

Introduction

Although you need to compile your stylesheets to plain CSS beforehand, they add a variety of benefits.

- Variables to keep constants throughout your stylesheet
- Calculation of values for more precision
- Reuse styles from other elements
- Bundling styles

Variables

Variables

LESS allows you to declare variables and use them in your CSS to keep consistency.

Variables

LESS allows you to declare variables and use them in your CSS to keep consistency.

To declare a variable, Simply do

@variable: value.

Variables

LESS allows you to declare variables and use them in your CSS to keep consistency.

To declare a variable, Simply do

@variable: value.

Example:

@myblue: #00aeef;

Variables

Example:

```
h1 {
    color: #00aeef;
    background-color: #00aeef;
}

.mybg {
    background-color: #00aeef;
    stroke: solid 1px #00aeef;
}
```

Variables

This can be written as:

```
@myblue: #00aeef;

h1 {
    color: @myblue;
    background-color: @myblue;
}

.mybg {
    background-color: @myblue;
    stroke: solid 1px @myblue;
}
```

Calculations

Calculations

Calculations can be used in tandem with variables to provide a consistent experience throughout your page.

Calculations

Calculations can be used in tandem with variables to provide a consistent experience throughout your page.

For example, say you want to place the element 30% + 15px from the left.

Calculations

Calculations can be used in tandem with variables to provide a consistent experience throughout your page.

For example, say you want to place the element 30% + 15px from the left.

This can be done by:

```
@myleft: 30% + 15px;

.exampledivv {
    left: @myleft;
}
```

Calculations

You can also use calculations to compute hexcodes!

Calculations

You can also use calculations to compute hexcodes!

Example:

```
@bluecolor: #00aeef;
@newbluecolor: @bluecolor + #111;
```

Calculations

You can also use calculations to compute hexcodes!

Example:

```
@bluecolor: #00aeef;
@newbluecolor: @bluecolor + #111;
```

@newbluecolor is equivalent to: #00b00

Mixins

Mixins

A **mixin** is the LESS equivalent of transgenesis.

Mixins

A **mixin** is the LESS equivalent of transgenesis.

You can take the best parts of one style and transplant it into another

Mixins

A **mixin** is the LESS equivalent of transgenesis.

You can take the best parts of one style and transplant it into another

Of course this transplants **ALL** of the styling associated with it so be careful!

Calculations

For example:

```
.bluesmush {
    color: blue;
    width: 5px;
}

h1 {
    font-family: Arial;
    .bluesmush;
}
```

Calculations

Is equivalent to:

```
h1 {
    font-family: Arial;
    color: blue;
    width: 5px;
}
```

Nesting

Nesting

Nesting of styles can be accomplished using LESS.

Nesting

Nesting of styles can be accomplished using LESS.

Nesting is another feature of LESS that allows you to style quickly and in an organized manner.

Nesting

Nesting of styles can be accomplished using LESS.

Nesting is another feature of LESS that allows you to style quickly and in an organized manner.

Nesting is useful when you have an element and need to style it along with its children.

Nesting

For example, say you wanted to nest this:

```
#bigdiv {
    width: 900px;
}

#bigdiv > .container {
    width: 300px;
}

#bigdiv > .container > div {
    width: 50px;
}

#bigdiv > .container > div > h3 {
    color: blue;
}
```

Nesting

This can be done easily in LESS like so:

```
#bigdiv {
    width: 900px;
    .container {
        width: 300px;
        div {
            width: 50px;
            h3 {
                color: blue;
            }
        }
    }
}
```

Importing

Importing

LESS stylesheets can be separated easily with the use of importing.

Importing

LESS stylesheets can be separated easily with the use of importing.

If you're familiar with C, this is similar to the inclusion of headers at the beginning of your code.

Importing

LESS stylesheets can be separated easily with the use of importing.

If you're familiar with C, this is similar to the inclusion of headers at the beginning of your code.

Inclusion includes variables and style definitions.

Importing

Here's an example:

```
toimport.less:
    @myblue: #00AEEF;
    @mywhite: #FEFEFE;
style.less:
    @import "toimport.less";
```

Import References

Import References

@import is nice and all, but using it imports the entire stylesheet.

Import References

@import is nice and all, but using it imports the entire stylesheet.

LESS includes a way to import only what you're using.

Import References

@import is nice and all, but using it imports the entire stylesheet.

LESS includes a way to import only what you're using.

This can by done by:

```
@import (reference) "stylesheet.less";
```

Import References

Here's an example:

```
toimport.less:
    @myblue: #00AEEF;
    @mywhite: #FEFEFE;
style.less:
    @import (reference) "toimport.less";
    h1 {
        color: @myblue;
    }
```

Import References

Here's an example:

```
toimport.less:
    @myblue: #00AEEF;
    @mywhite: #FEFEFE;
style.less:
    @import (reference) "toimport.less";
    h1 {
        color: @myblue;
    }
```

This only imports the @myblue variable and not @mywhite.

Import References

Here's an example:

```
toimport.less:
    @myblue: #00AEEF;
    @mywhite: #FEFEFE;
style.less:
    @import (reference) "toimport.less";
    h1 {
        color: @myblue;
    }
```

This only imports the @myblue variable and not @mywhite.

This is useful in keeping filesizes small. You don't want to import all 35kb of a stylesheet when you're only using one line of it.

How the heck do I actually use LESS?

How the heck do I actually use LESS?

Knowing all of this is great and all, but how do you actually turn .less files into css that you can use?

How the heck do I actually use LESS?

Knowing all of this is great and all, but how do you actually turn .less files into css that you can use?

With LESS, you can either use node.js or import a script into your html page.

How the heck do I actually use LESS?

Knowing all of this is great and all, but how do you actually turn .less files into css that you can use?

With LESS, you can either use node.js or import a script into your html page.

node.js:

How the heck do I actually use LESS?

Knowing all of this is great and all, but how do you actually turn .less files into css that you can use?

With LESS, you can either use node.js or import a script into your html page.

node.js:

Install less by running

```
npm install -g less;
```

How the heck do I actually use LESS?

Knowing all of this is great and all, but how do you actually turn .less files into css that you can use?

With LESS, you can either use node.js or import a script into your html page.

node.js:

Install less by running

```
npm install -g less;
```

Then you can compile your code by running lessc like so:

```
lessc styles.less styles.css
```

How the heck do I actually use LESS?

Alternatively, you can add a javascript script to your html page after including the less file.

How the heck do I actually use LESS?

Alternatively, you can add a javascript script to your html page after including the less file.

```
<link rel="stylesheet/less" type="text/css" href="style.less"/>
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.0.0/less.min.js"></script>
```

How the heck do I actually use LESS?

Alternatively, you can add a javascript script to your html page after including the less file.

```
<link rel="stylesheet/less" type="text/css" href="style.less"/>
<script src="//cdnjs.cloudflare.com/ajax/libs/less.js/3.0.0/less.min.js"></script>
```

* You can also download it and include the local file.

Fin.