Neuron

Release 1.0.0

Aug 20, 2020

Contents

Neuron is a language that compiles directly into HTML, CSS, and JavaScript.

CHAPTER 1

Features

- Object-oriented programming
- Consise and forgiving syntax
- Features the long-awaited unity of HTML, CSS, and JavaScript
- Support in most browsers without third-party software
- Comprehensive standard library

CHAPTER 2

Contents

2.1 Getting Started with Neuron

2.1.1 Installing Neuron with `npm`

With the latest stable version of `npm` installed, run the following in your command line to install the Neuron language globally:

```
npm install -g neuron-lang
```

The command `neuron` will be available for use in the command line globally. The `neuron` command will accept an argument for the directory to the file to compile and execute. `neuron` can also be used to invoke the Neuron language interpereter when used without arguments.

2.1.2 Neuron in the Browser

When using Neuron in the browser, specify the script type as `text/neuron`, including the compiler as a module.

```
<script src = "https://github.com/underpig1/neuron-lang/blob/master/lib/browser/
↪browser.mjs" type = "module"></script>
<script type = "text/neuron">
  // neuron
</script>
```

2.2 Neuron Language Reference

2.2.1 Objects and Functions

Objects and functions can be integrated. Neuronic Objects can hold both properties and methods accessable to itself and to other objects, providing it with a dynamic foundation for script execution. Objects and functions are defined in

the following syntax:

```
myObject {
  property: value;
  method(parameters);
}
```

Objects can hold both properties and methods, accessible to itself and to other objects:

```
myObject {
  width: 1px;
  height: 1px;
  position: 10px 10px;
  log(this.position);
}

log(myObject.position);
```

Objects containing HTML- and CSS-specific properties should be invoked as such:

```
invoke(myObject);
```

Functions can be called as methods:

```
myObject(parameters)
```

Functions with return values can be called as inline functions:

```
[myObject => parameters]
```

Functions can also take parameters, prefixed with # and followed by their parameter number:

```
absoluteValue {
  return([math.abs => #0]); // #0 defines the first parameter passed
}

log([absoluteValue => 0]);
// logs '0'
```

Functions can be executed with the following syntax as well:

```
add {
  log(#0 + #1 is [#0 + #1]);
}

add(0, 1);
// logs '0 + 1 is 1'
```

2.2.2 Bracket Notation

Groupings, literals, and inline functions are all preceded by and followed by brackets. Bracket notation is defined by the following syntax:

```
[function => parameters]
[function parameters]
[literal]
[grouping]
```

Bracket notation can be embedded in parameters or values:

```
log([function => parameters]px, [grouping]px)
```

Groupings

Parameters are seperated by commas or spaces. Bracket notation can escape parameter seperators, and pass a single parameter.

```
log(Hello, World!) // The parameter seperator is not escaped
// logs 'Hello' and 'World!' in 'Hello World!'
log([Hello, World!]) // Bracket notation
// logs 'Hello, World!'
```

Literals

Literals evaluate its contents.

```
log(0 + 1) // Evaluated as a string
// logs '0 + 1'
log([0 + 1]) // Evaluated as a literal
// logs '1'
```

Inline Functions

Inline functions can pass parameters to a function with a return value, seperated by spaces.

```
[function => parameters]
```

```
log([math.sqrt => 1]px);
```

2.2.3 Variables and Assignments

Variables can be defined with the following syntax:

```
#myVar = value
```

Variables can then be referenced or reassigned:

```
log(#myVar);
#myVar = value;
```

Object properties can be referenced and reassigned as well:

```
log(myObject.position);
myObject.position = value;
```

2.2.4 Conditionals

Conditionals can be defined with the following syntax:

```
if (condition) {
  // contents
}
```

The conditional will execute if the condition returns `true`. Conditionals are capable of being embedded in executable objects.

2.2.5 Iterators

Iterators can be defined with the following syntax:

```
for (var in/=> value) {
  // contents
}
```

The iterator will iterate through the value, executing its contents. The variable can then be used with the prefixed `#` notation.

```
for (var => value) {
  log(#var);
}
```

Iterators are capable of being embedded in executable objects.

2.2.6 Operators

Arithmetic Operators

`+` Addition operator

`-` Subtraction operator

`/` Division operator

`*` Multiplication operator

`%` Remainder operator

`**` Exponentiation operator

Relational Operators

`<` Less than operator

`>` Greater than operator

`<=` Less than or equal to operator

`>=` Greater than or equal to operator

`==` Equality operator

`!=` Inequality operator

=== Identity operator

!== Nonidentity operator

Logical Operators

&& Logical AND

|| Logical OR

! Logical NOT operator

Assignment Operators

= Assignment operator

2.2.7 Inheritance

Expressions preceded by a tilde ~ are recognized as inheritance. Inheritance can be defined with the following syntax:

```
~value
```

2.3 Neuron API Reference

2.3.1 Window API

Properties

`window.width` Returns the width of the current window

`window.height` Returns the height of the current window

Static Properties

`window.event` Returns the event of the current window

2.3.2 Math API

Static Properties

`math.infinity` Returns the value of infinity

`math.pi` Returns the value of pi

`math.E` Returns the value of Euler's constant

Static Methods

[math.abs => x] Returns the absolute value of `x`

[math.acos => x] Returns the arccosine of `x`

[math.acosh => x] Returns the hyperbolic arccosine of `x`

[math.asin => x] Returns the arcsine of `x`

[math.asinh => x] Returns the hyperbolic arcsine of `x`

[math.atan => x] Returns the arctangent of `x`

[math.atanh => x] Returns the hyperbolic arctangent of `x`

[math.atan2 => x y] Returns the arctangent of the quotient of `x` and `y`

[math.cbrt => x] Returns the cube root of `x`

[math.ceil => x] Returns the smallest integer greater than or equal to `x`

[math.clz32 => x] Returns the number of leading zeroes of the 32-bit integer `x`

[math.cos => x] Returns the cosine of `x`

[math.cosh => x] Returns the hyperbolic cosine of `x`

[math.fround => x] Returns the nearest single precision float representation of `x`

[math.hypot => ...] Returns the square root of the sum of squares of its arguments

[math.imul => x y] Returns the result of the 32-bit integer multiplication of `x` and `y`

[math.log => x] Returns the natural logarithm of `x`

[math.log10 => x] Returns the base-10 logarithm of `x`

[math.log2 => x] Returns the base-2 logarithm of `x`

[math.max => ...] Returns the largest of its arguments

[math.min => ...] Returns the smallest of its arguments

[math.pow => x y] Returns base `x` to the exponent power `y`

[math.random] Returns a pseudo-random number between `0` and `1`

[math.round => x] Returns `x` rounded to the nearest integer

[math.sign => x] Returns the sign of `x`

[math.sin => x] Returns the sine of `x`

[math.sinh => x] Returns the hyperbolic sine of `x`

[math.sqrt => x] Returns the positive square root of `x`

[math.tan => x] Returns the tangent of `x`

[math.tanh => x] Returns the hyperbolic tangent of `x`

[math.trunc => x] Returns the integer portion of `x`

2.3.3 JSON API

Static Methods

[json.parse => json] Returns the object represented by `json`

[json.string => json] Returns the string value of `json`

2.3.4 RegExp API

Static Methods

[re.match => string regexp] Returns the matching string against the regular expression

[re.replace => string regexp string] Returns the string value, replacing the matching strings against the regular expression

2.3.5 Time API

Static Properties

time.now Returns the numeric value corresponding to the current time

time.day Returns the numeric value corresponding to the current day of the week

time.year Returns the numeric value corresponding to the current year

time.hour Returns the numeric value corresponding to the current hour

time.milliseconds Returns the numeric value corresponding to the current millisecond

time.minutes Returns the numeric value corresponding to the current minute

time.month Returns the numeric value corresponding to the current month

time.seconds Returns the numeric value corresponding to the current second

time.time Returns the numeric value corresponding to the current time

time.timezone Returns the time-zone offset in minutes for the current locale

time.UTC Returns the numeric value corresponding to the current day of the month according to universal time

2.4 Neuron Standard Properties Reference

2.4.1 Static Methods

[colorspaces.hextorgba => hex] Returns the RGBA value of `hex`

[int? => x] Returns a boolean corresponding to the value of `x`, if `x` is an integer

[bool? => x] Returns a boolean corresponding to the value of `x`, if `x` is a boolean

[concatenate => x y] Returns a string with the concatenated value of `x` and `y`

[length? => string] Returns the length of `string`

[includes? => string value] Returns a boolean corresponding to the value of `string`, if `string` includes `value`

[upcase => string] Returns a string corresponding to the uppercased value of `string`

[downcase => string] Returns a string corresponding to the lowercased value of `string`

[input] Returns the value of a prompt, containing the user input

2.4.2 Static Properties

inherited(object) The inheritance property

setProperty(object, property, value) Sets the property of the given object to the passed value

fill(r, g, b, a) Sets the fill property to the values passed

stroke(object, property, value) Sets the stroke property to the values passed

position(x, y) Sets the position property to the values passed

width(x) Sets the width property to the values passed

height(x) Sets the height property to the values passed

polygon([x, y]...) Sets the polygon property containing the passed values; generates a polygon containing the passed values

value(value) The shorthand variable value property

return(value) Returns the given value

gradient([color]...) Sets the gradient property containing the passed values; generates a gradient containing the passed values

image(url) Sets the polygon property containing the passed values; generates an image containing the passed values

alert(value) Produces an alert in the current window containing the given value

window(value) Produces a new window

invoke(object) Invokes the given object

onclick([redirect/log/alert/...], value) Executes the given function once clicked

appendProperty(object, property, value) Appends a property with the passed value to the given object

redirect(url) Redirects the page to the given URL