DOM Events

The Complete Web Developer in 2018

The Complete Web Developer in 2018
Zero to Mastery
Andrei Neagoie
Lecture Notes by Stephanie

DOM Events

- Use javascript to listen to events (ex: hover, click) and react to them

Add <button> in html file.
Add link to script.js in html file (<script> at end of <body>)

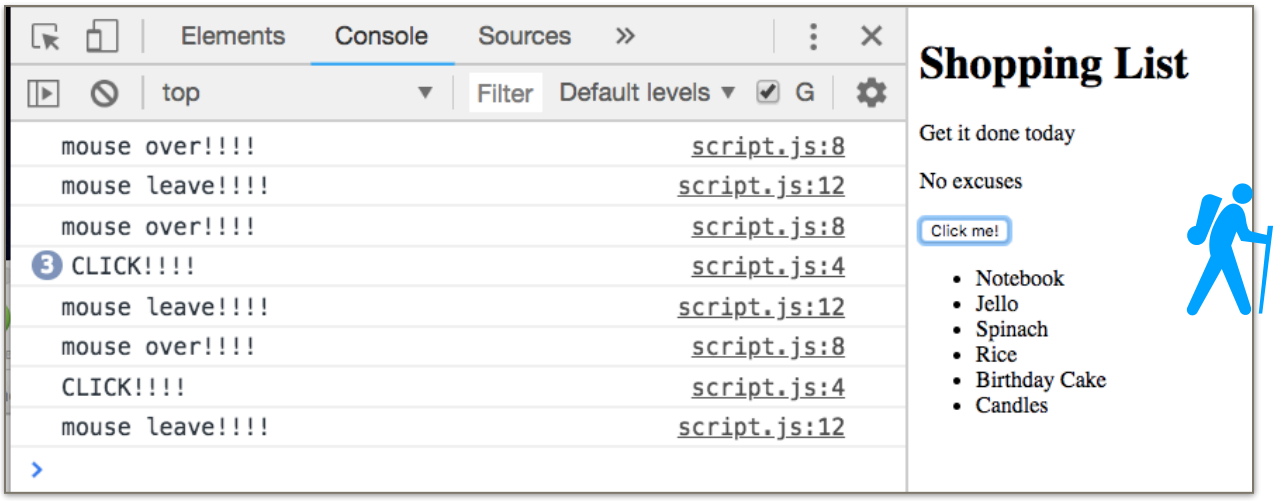
index.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Javascript + DOM</title>
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
        <h1>Shopping List</h1>
        <p id="first">Get it done today</p>
        <p class = "second">No excuses</p>
        <button>Click me!</button>
        <ul>
                <li random="23">Notebook</li>
                <li>Jello</li>
                <li>Spinach</li>
                <li>Rice</li>
                <li>Birthday Cake</li>
                <li>Candles</li>
        </ul>
        <script type="text/Javascript" src="script.js"></script>
</body>
</html>
```

script.js

```
var button = document.getElementsByTagName("button")[0];

button.addEventListener("click", function() {
        console.log("CLICK!!!!");
})

button.addEventListener("mouseenter", function() {
        console.log("mouse over!!!!");
})

button.addEventListener("mouseleave", function() {
        console.log("mouse leave!!!!");
})
```

Based on previous html and javascript code, see console log for every time button is clicked, mouse is over button, or mouse leaves button:



Aside

```
⊗ ▶Uncaught TypeError:                        script.js:3
  button.addEventListener is not a function
      at script.js:3
```

Common error: Get this if we do not specify which element of the array we are using the method on. (Try to use method on array instead of method on element >> error)

Example:

```
var button = document.getElementsByTagName(“button");

button.addEventListener("click", function() {
        console.log("CLICK!!!!");
})
```

* * *

Lets update code to allow user to add items to the list:

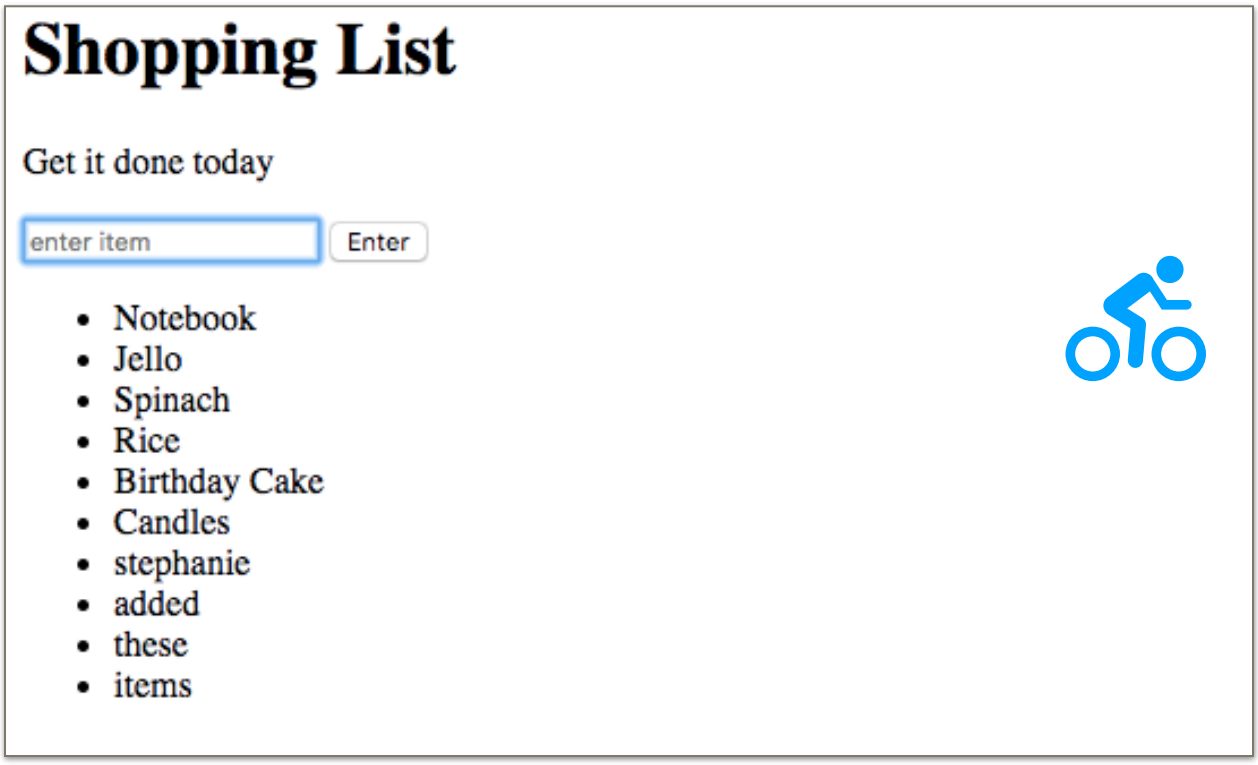
index.html

```
<!DOCTYPE html>
<html>
<head>
        <title>Javascript + DOM</title>
        <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
        <h1>Shopping List</h1>
        <p id="first">Get it done today</p>
        <input id="userinput" type="text" placeholder="enter item">
        <!-- use ID instead of class for speed -->
        <button id="enter">Enter</button>
        <!-- use ID instead of class for speed -->
        <ul>
                <li random="23">Notebook</li>
                <li>Jello</li>
                <li>Spinach</li>
                <li>Rice</li>
                <li>Birthday Cake</li>
                <li>Candles</li>
        </ul>
        <script type="text/Javascript" src="script.js"></script>
</body>
</html>
```

script.js

```
// we want to select user input, button, list

var button = document.getElementById("enter");
var input = document.getElementById("userinput");
var ul = document.querySelector("ul");

// click button
button.addEventListener("click", function() {
	if (input.value.length > 0){ // user input not blank

		// create list element
		var li = document.createElement(“li");

		// add text to list element (user input)
		li.appendChild(document.createTextNode(input.value));

		// append li to unordered list
		ul.appendChild(li);

		// reset form to blank value
		input.value="";
	}
})


// press enter
input.addEventListener("keypress", function(event) {
	//console.log(event);

	// user input not blank AND press enter key
	if (input.value.length > 0 && event.keyCode ===13){

		// create list element
		var li = document.createElement(“li");

		// add text to list element (user input)
		li.appendChild(document.createTextNode(input.value));

		// append li to unordered list
		ul.appendChild(li);

		// reset form to blank value
		input.value="";
	}


})
```

Now, every time user types in item and presses enter or clicks the button, the item is added to the bottom of the list.



Interesting features of code we used for this (above):

- Need to look at keyCode 13 to know when user pressed “enter” during input
- Input field is cleared after every new item
- We repeat code throughout this javascript file, therefore it is not very extensible

DRY - do not repeat yourself

If you copy and paste a lot of code, the code is not very *extensible*
We can *refactor* the code (make the code look better, clean it up):

script.js

```
// select elements we are interested in, cache them
// we want to select user input, button, list
var button = document.getElementById("enter");
var input =
document.getElementById("userinput");
var ul = document.querySelector("ul");


// refactored code (cleaned up)
// function declarations
function inputLength (){
	return input.value.length;
}

function createListElement(){
	// create list element
	var li = document.createElement("li");
	// add text to list element (user input)
	li.appendChild(document.createTextNode(input.value));
	// append li to unordered list
	ul.appendChild(li);
	// reset form to blank value
	input.value="";
}

function addListAfterClick(){
	if (inputLength() > 0){ // user input not blank
		createListElement();
	}
}

function addListAfterKeypress(event){ // still need event param here
	//console.log(event);

	// user input not blank AND press enter key
	if (inputLength() > 0 && event.keyCode ===13){
		createListElement();
	}
}

// click button >> if anyone clicks btn, run this fnxn
button.addEventListener("click", addListAfterClick);

// press enter >> if press enter, run this fnxn
input.addEventListener("keypress", addListAfterKeypress);
```

- code does not repeat itself
- Queries are cached at the beginning (doesnt use too much browser power)
- Everything broken down into simple functions
- Run functions if event takes place

Event listener with anonymous function

Here, we'll take a look at how to use an anonymous function to pass parameters into the event listener.

HTML

```
<table id="outside">
    <tr><td id="t1">one</td></tr>
    <tr><td id="t2">two</td></tr>
</table>
```

JavaScript

```
// Function to change the content of t2
function modifyText(new_text) {
  var t2 = document.getElementById("t2");
  t2.firstChild.nodeValue = new_text;
}

// Function to add event listener to table
var el = document.getElementById("outside");
el.addEventListener("click", function(){modifyText("four")}, false);
```

Notice that the listener is an anonymous function that encapsulates code that is then, in turn, able to send parameters to the `modifyText()` function, which is responsible for actually responding to the event.

+ + + Background Generator Exercise + + +

<input type="color">

`<input>` elements of type `"color"` provide a user interface element that lets a user specify a color, either by using a visual color picker interface or by entering the color into a text field in `"#rrggbb"` hexadecimal format. Only simple colors (with no alpha channel) are allowed. The values are compatible with CSS.

CSS gradients

Syntax

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

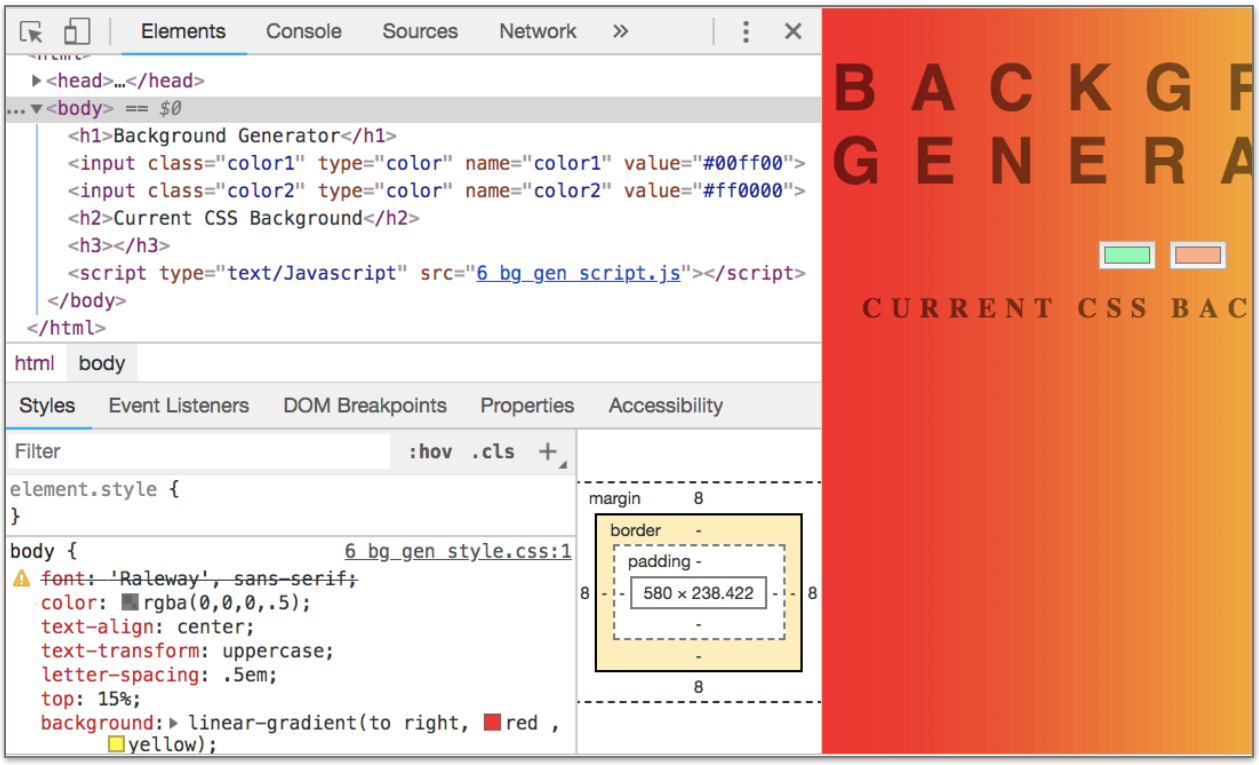
Example

```
#grad {
  background-image: linear-gradient(red, yellow);
}
```

color1.addEventListener("input",.....

The DOM `input` event fires synchronously when the `value` of an `<input>`, `<select>`, or `<textarea>` element is altered. The event also applies to elements with `contenteditable` enabled, and to any element when `designMode` is turned on.

To find where background is in HTML, use developer tools >> right click page >> Inspect, and find it in the <body>



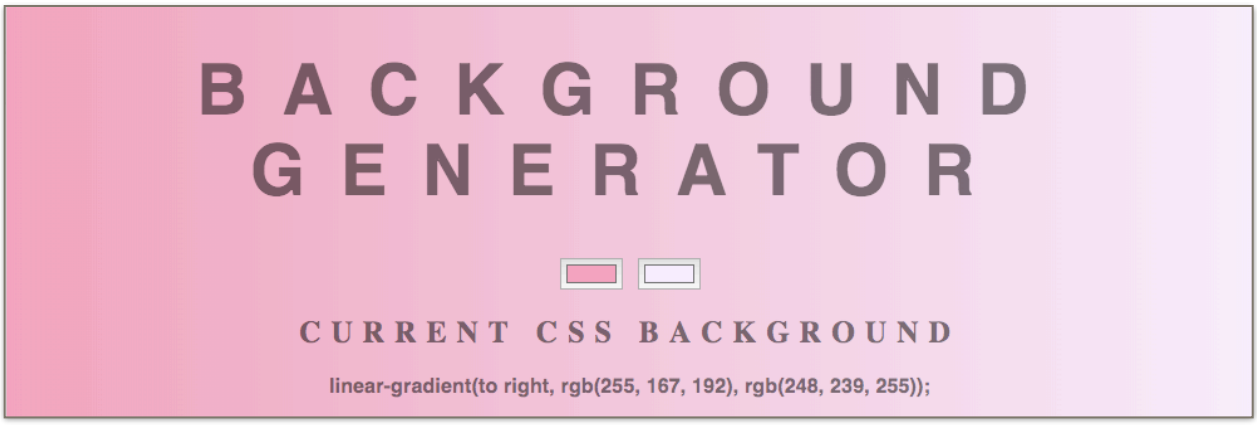
Lets look at code and result after getting it to work….

6 bg gen index.html

```
<!DOCTYPE html>
<html>
<head>
	<title>Gradient Background</title>
	<link rel="stylesheet" type="text/css" href="6 bg gen style.css">
</head>
<body id="gradient">
	<h1>Background Generator</h1>
	<input class = "color1" type="color" name="color1" value="#FFA7C0">
	<input class = "color2" type="color" name="color2" value="#F8EFFF">

	<h2>Current CSS Background</h2>

	<h3></h3>

	<script type="text/Javascript" src="6 bg gen script.js"></script>
</body>
```

6 bg gen style.css

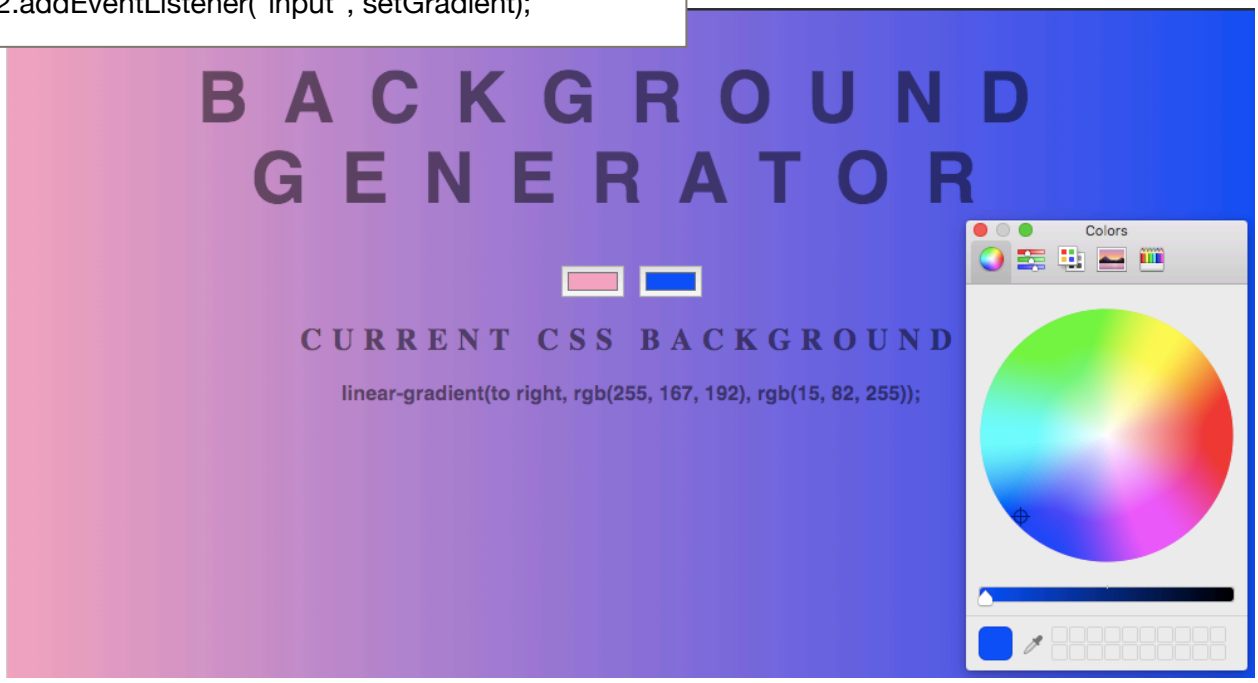
```
body {
	font: 'Raleway', sans-serif;
   color: rgba(0,0,0,.5);
   text-align: center;
   text-transform: uppercase;
   letter-spacing: .5em;
   top: 15%;
	background: linear-gradient(to right, red , yellow); /* Standard syntax */
}

h1 {
   font: 600 3.5em 'Raleway', sans-serif;
   color: rgba(0,0,0,.5);
   text-align: center;
   text-transform: uppercase;
   letter-spacing: .5em;
   width: 100%;
}

h3 {
	font: 900 1em 'Raleway', sans-serif;
   color: rgba(0,0,0,.5);
   text-align: center;
   text-transform: none;
   letter-spacing: 0.01em;

}
```



6 bg gen script.js

```
// cache selectors

var css = document.querySelector("h3");
var color1 = document.querySelector(".color1");
var color2 = document.querySelector(".color2");
var body = document.getElementById("gradient");

// function declaration

function setGradient() {
        body.style.background =
        "linear-gradient(to right, "
        + color1.value
        + ", "
        + color2.value
        + ")";

        css.textContent = body.style.background + ";";
}

// event listeners & function calls
setGradient();
color1.addEventListener("input", setGradient);
color2.addEventListener("input", setGradient);
```

Click to select colors…
changes background!



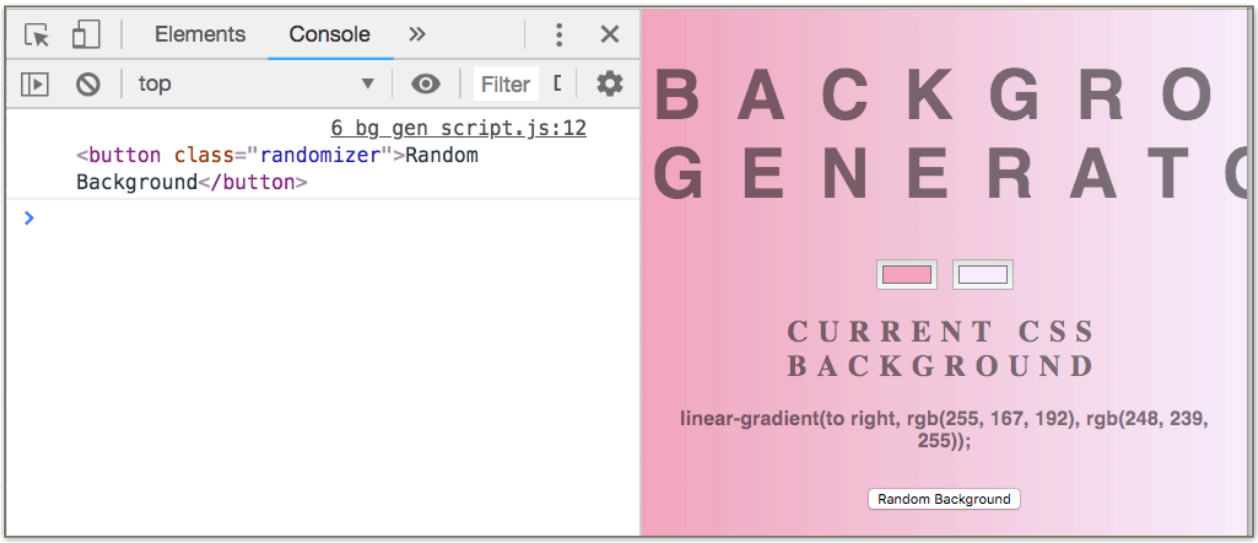
Something interesting I found when selecting classes…..

```
<!DOCTYPE html>
<html>
<head>
        <title>Gradient Background</title>
        <link rel="stylesheet" type="text/css" href="6 bg gen style.css">
</head>
<body id="gradient">
        <h1>Background Generator</h1>
        <input class = "color1" type="color" name="color1" value="#FFA7C0">
        <input class = "color2" type="color" name="color2" value="#F8EFFF">

        <h2>Current CSS Background</h2>

        <h3></h3>

        <h2><button class="randomizer">Random Background</button></h2>

        <script type="text/Javascript" src="6 bg gen script.js"></script>
</body>
```

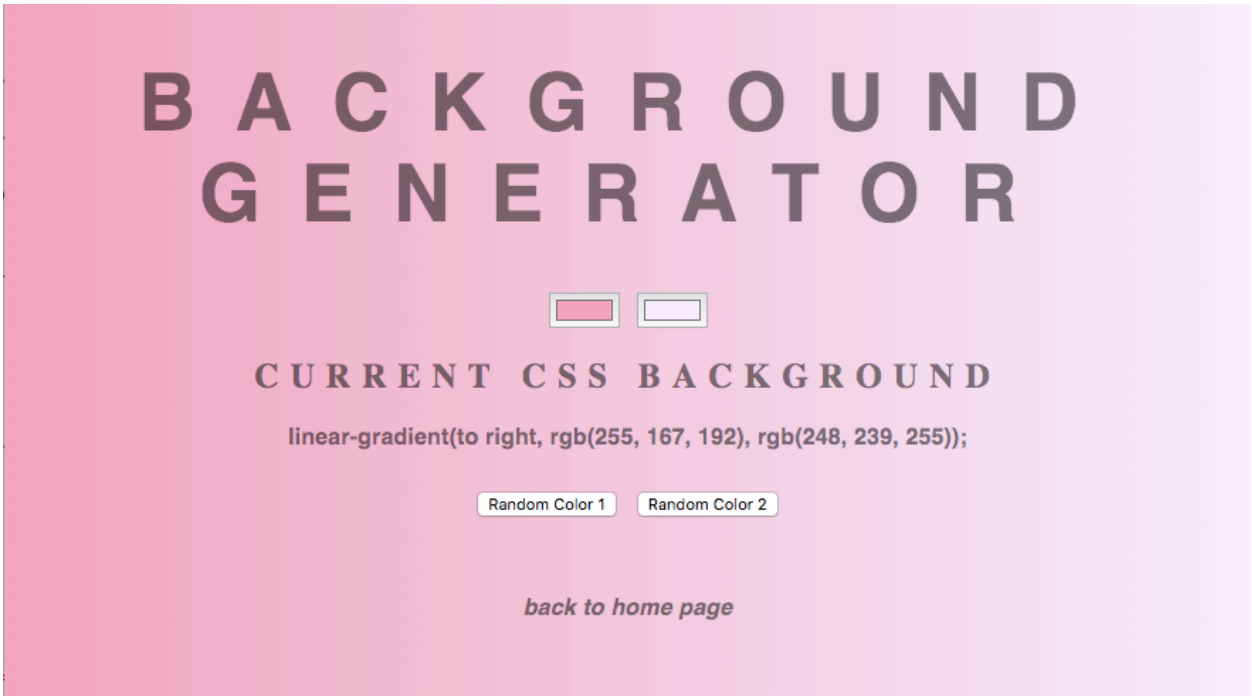
```
// cache selectors

var randomizer1 = document.getElementsByClassName("randomizer")[0];

console.log(randomizer1);
```

is the same selector as…

```
// cache selectors

var randomizer2 = document.querySelector(".randomizer");

console.log(randomizer2);
```



Final product, posted to GitHub:



choose color with color selector OR use random color buttons