Class 4:
Interaction I

Class 1

Course Description

Learn how to weave a range of online technologies into engaging interactive experiences. This course will teach the basics of web technologies that are fundamental to building an online presence for any design project. Students will learn how to identify the current standards and protocols for web browsers, mobile applications that rely on browsers and apps. You will also gain an understanding of the fundamentals of markup languages (HTML, XML) as well as formatting (CSS) and scripting (JavaScript). These basic skills will be contextualized within a basic overview of interface design. With the knowledge built in this course students will begin to understand how to create responsive web-based projects that adapt to different devices and develop strategies for creating screen-based interfaces.

Hypertext Markup Language

Extensible Markup Language

Cascading Style Sheets

JavaScript

Hypertext Markup Language

```
<!doctype html>
<html lang="en">
  <head>
    <title>What Screens Want</title>
    <link rel="stylesheet" href="styles/style.css">
  </head>
  <body>
    <div class="container">

      <!-- Start of navigation element -->
      <header class="site-header">
        <nav>
          <ul class="navigation">
            <li class="navigation-item"><a href="design.html"
```

Cascading Style Sheets

```
body {
  color: #555;
  font-family: sans-serif;
  margin: 0;
}

a {
  color: #999;
  text-decoration: none;
}

.logo-text {
  font-size: 1em;
  margin: 0;
}
```

JavaScript

```
console.log('Welcome to Developer Tools. JavaScript loaded from main.js!');
```

JavaScript

```
console.log('Welcome to Developer Tools.
JavaScript loaded from main.js!');
```

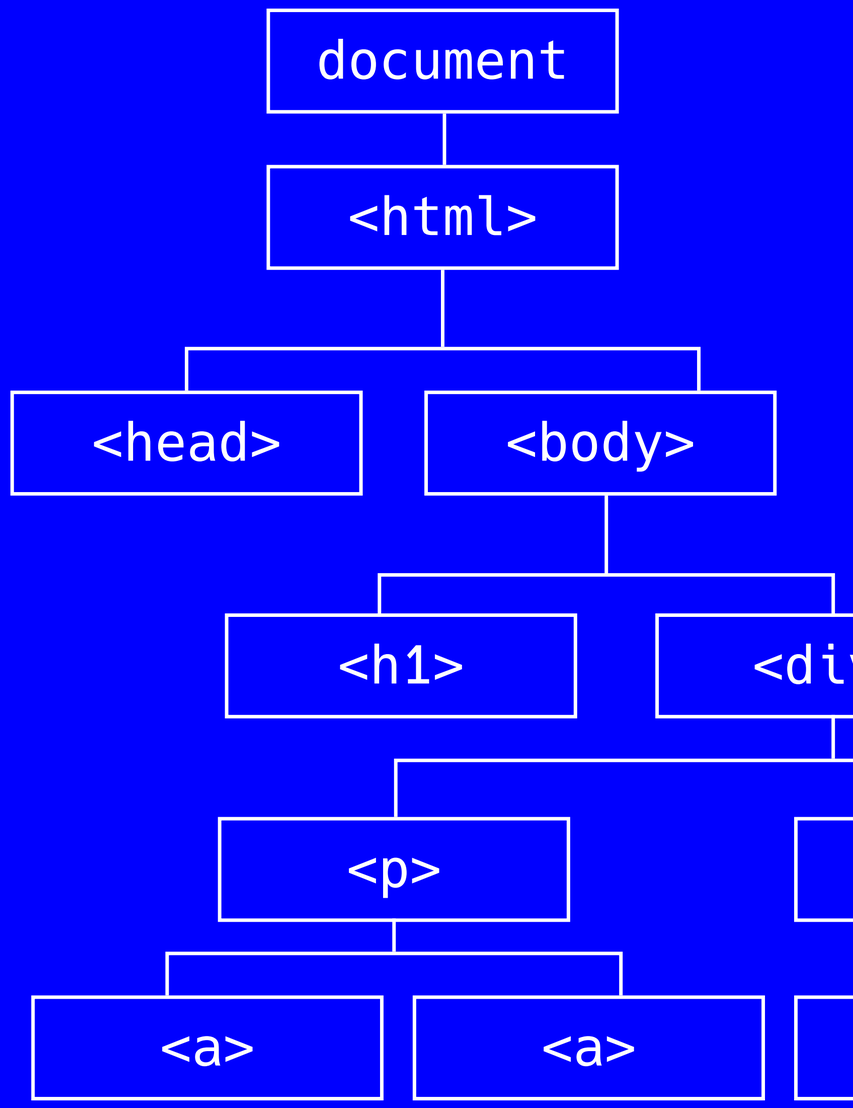
→ AHH!! Nothing is happening!!
→ Chrome: ⌘ + ⌥ + P
→ Safari: ⌘ + ⌥ + C

Document Object Model

A cross-platform programming interface that treats an HTML document as a tree structure where in each node is an object representing a part of the document.

The DOM model represents a document with a logical tree.

Each branch of the tree ends in a node, and each node contains objects. DOM methods allow programmatic access to the tree; with them you can change the document's structure, style or content.



JavaScript

JavaScript is a programming language that allows you to implement complex things on web pages. Every time a web page does more than just sit there and display static information for you to look at—displaying timely content updates, or interactive maps, or animated 2D/3D graphics, or scrolling video jukeboxes, and so on—you can bet that JavaScript is probably involved.

→ Client-side and Back-end
→ Java ≠ JavaScript

Skipping History (Again)

→ December, 1995
→ Netscape
→ DHTML
→ …
→ Libraries (jQuery, React, Vue.js)
→ Node.js
→ ES5, ES6

Skipping History (Again)

→ Canvas, WebGL, Service Workers, Video, Audio, Storage
→ …
→ JavaScript is Eating the World

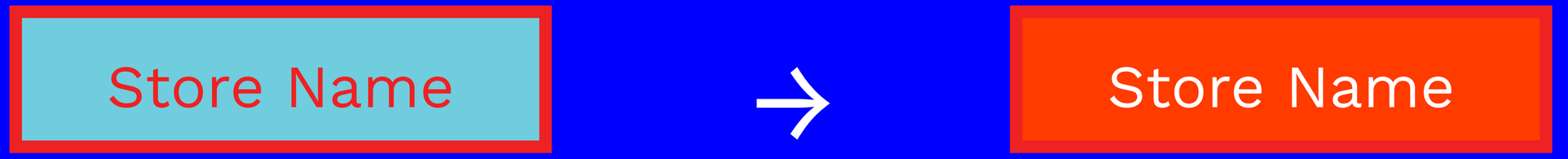
Interaction: CSS

```
body {
  color: #555;
  font-family: sans-serif;
  margin: 0;
}

a {
  color: #999;
  text-decoration: none;
}

.logo-text {
  font-size: 1em;
  margin: 0;
}
```

Pseudo-class

```
:hover
:active
:visited
:focus
...
```

Interaction: CSS



→


```
.store-item-button {
  font-size: 100px;
  color: red;
  padding: 1em;
  background: blue;
}
```

```
.store-item-button:hover {
  color: white;
  background: red;
}
```

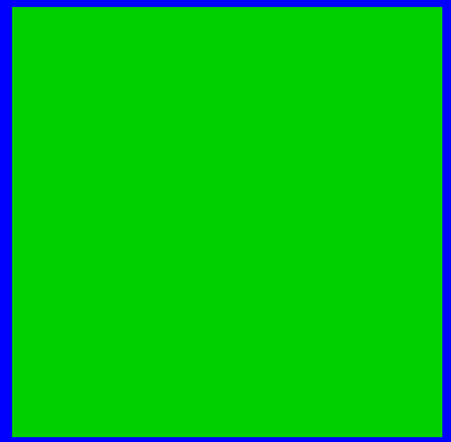
Store Link →

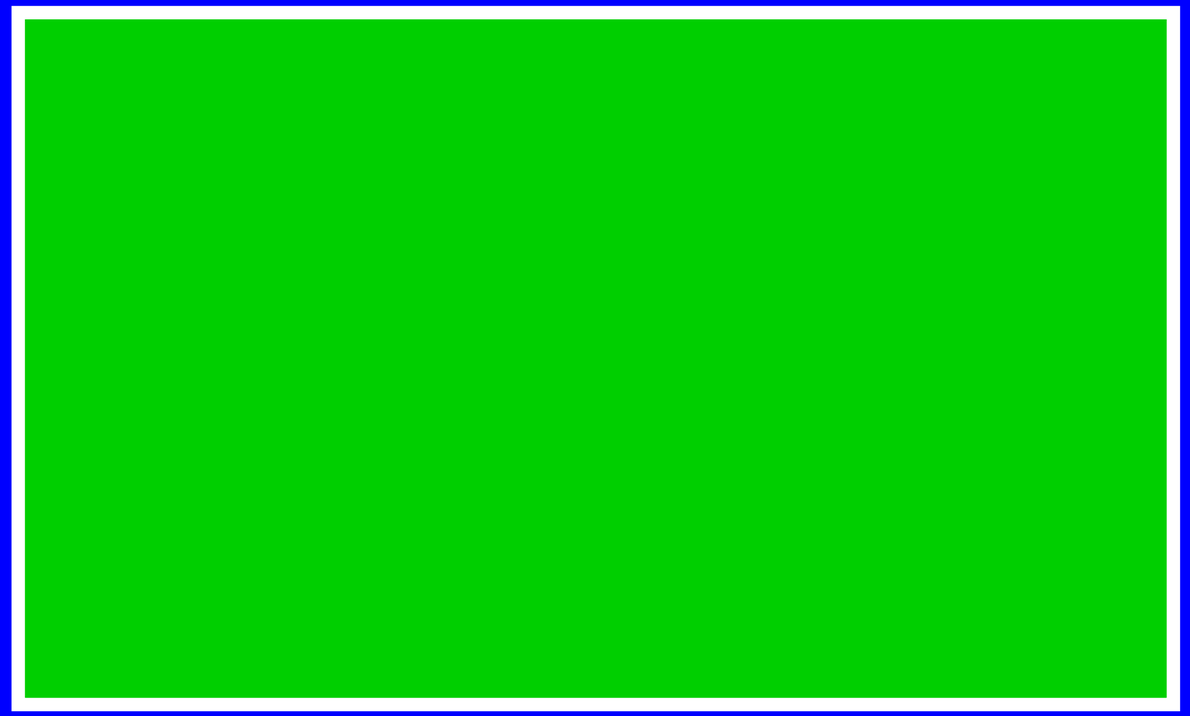
Store Link

```
.store-link {
  text-decoration: none;
  color: #ffffff;
}
```

```
.store-link:hover {
  text-decoration: underline;
  color: orange;
}
```

Interaction: CSS





```
.item-block {
  width: 100px;
  height: 100px;
  background: green;
}
```

```
.item-block:hover {
  width: 600px;
  height: 400px;
  border: 10px solid white;
}
```

Interaction: JavaScript

→ More complex interactivity
→ Calculations
→ Conditions
→ Extensibility
→ …
→ Much more!

JavaScript: Terms

Variable

Variables are containers that you can store values in. A **variable** is a value that is subject to change, depending on conditions or on information passed to the program.

Can be a:

- **String**
- **Number**
- **Boolean**
- **Array**
- **Object**

JavaScript: Terms

→ String
Number
Boolean
Array
Object

```
var myVariable = 'Hello';
```

JavaScript: Terms

String
→ Number
Boolean
Array
Object

→

```
var myVariable = 'Hello';
var myNumber = 10;
```

JavaScript: Terms

String
Number
→ Boolean
Array
Object

→

```
var myVariable = 'Hello';
var myNumber = 10;
var myBoolean = true;
```

JavaScript: Terms

String
Number
Boolean
→ Array
Object

→

```
var myVariable = 'Hello';
var myNumber = 10;
var myBoolean = true;
var myArray =
   ['mango','cheetos',
    'apple'];
```

JavaScript: Terms

String
Number
Boolean
Array
→ Object

```
var myVariable = 'Hello';
var myNumber = 10;
var myBoolean = true;
var myArray =
  ['mango','cheetos',
   'apple'];
→ var myObj =
  document.
  querySelector('h1');
```

JavaScript: Terms

Operators

An operator is a mathematical symbol which produces a result based on two values (or variables).

Can be a:

Addition
Subtraction
Multiplication
Division
Assignment
Equality
Not
Does-not-equal

JavaScript: Terms

Addition	`+`	`2 + 5 = 7`
Subtraction	`-`	`5 - 2 = 3`
Multiplication	`*`	`3 * 4 = 12`
Division	`/`	`4 / 2 = 2`

Assignment	`=`	`variable = 'Hello';`
Equality	`===`	`var1 === var2`
Not	`!`	`!(var1 === var2)`
Does-not-equal	`!==`	`var1 !== 10`

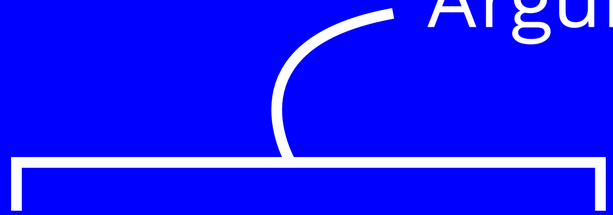
JavaScript: Terms

Function

Functions are a way of packaging functionality that you wish to reuse. When you need the procedure you can call a function, with the function name, instead of rewriting the entire code each time.

No functions in HTML or CSS.

JavaScript: Terms




```
function multiply(num1, num2) {
  var result = num1 * num2;
  return result;
}

multiply(4, 7);
```

JavaScript: Terms

Conditionals

Conditionals are code structures which allow you to test if an expression returns true or not, running alternative code revealed by its result.

A very common form of conditionals is the if...else statement.

JavaScript: Terms

```
var city = 'toronto';

if (city === 'toronto') {
  alert('The city that we are in!');
} else {
  alert('We are somewhere else?');
}
```

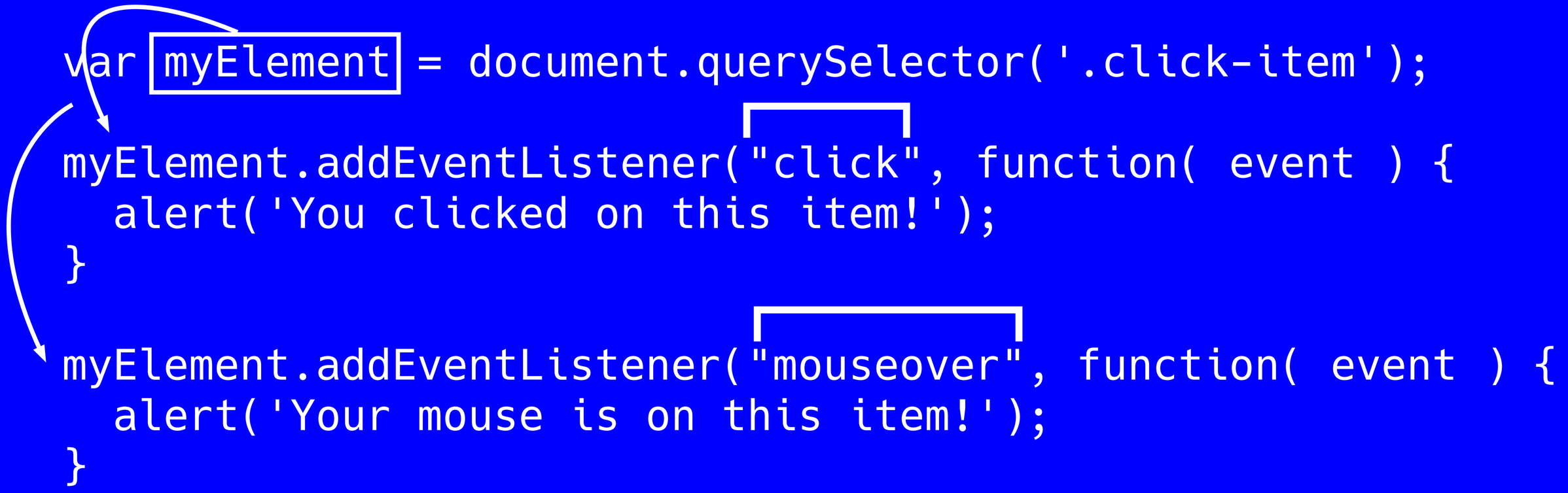
JavaScript: Terms

Events

These are code structures which **listen for things happening in browser, running code in response.**

We'll be seeing these often!

JavaScript: Terms



```
var myElement = document.querySelector('.click-item');

myElement.addEventListener("click", function( event ) {
  alert('You clicked on this item!');
}

myElement.addEventListener("mouseover", function( event ) {
  alert('Your mouse is on this item!');
}
```

Interaction: JavaScript

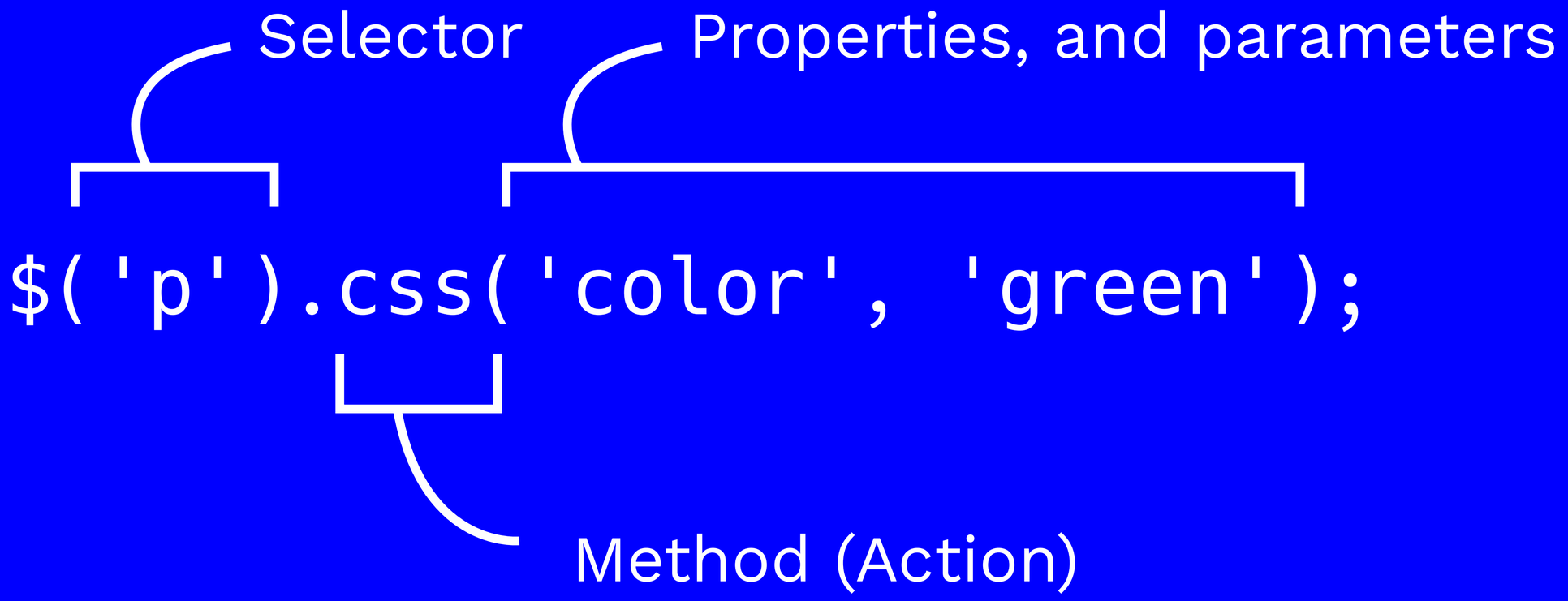
→ { and (should always closed!!
→ Objects everywhere!
→ Random
→ It’s all text
→ Test, run, debug in browser
→ Many ways of doing one thing
→ Rabbit hole

Interaction: JavaScript

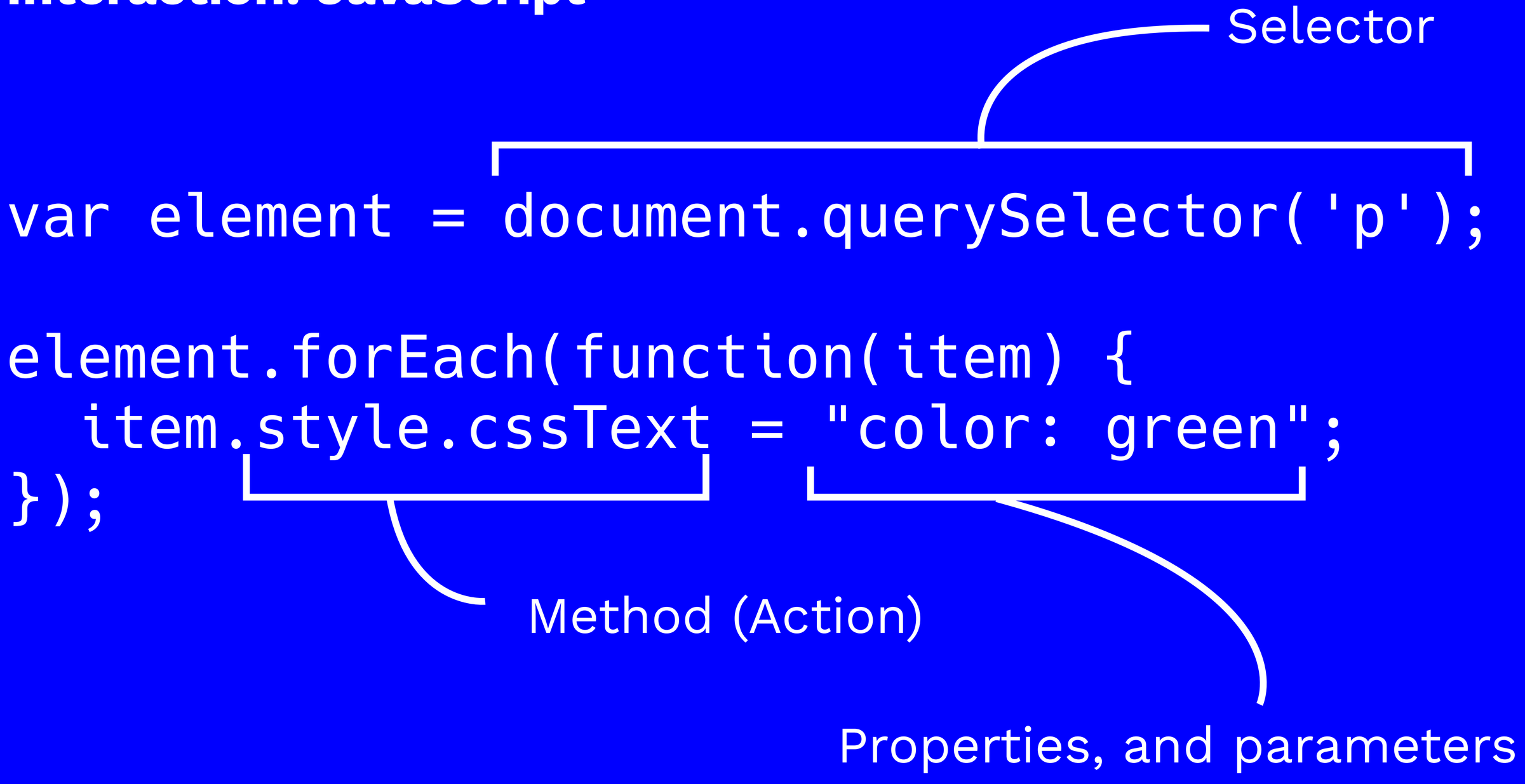
→ jQuery
→ jQuery Libraries
→ January-ish, 2006
→ John Resig

→ 2010’s: React, Angular, Vue.js
→ Web Components, Typescript

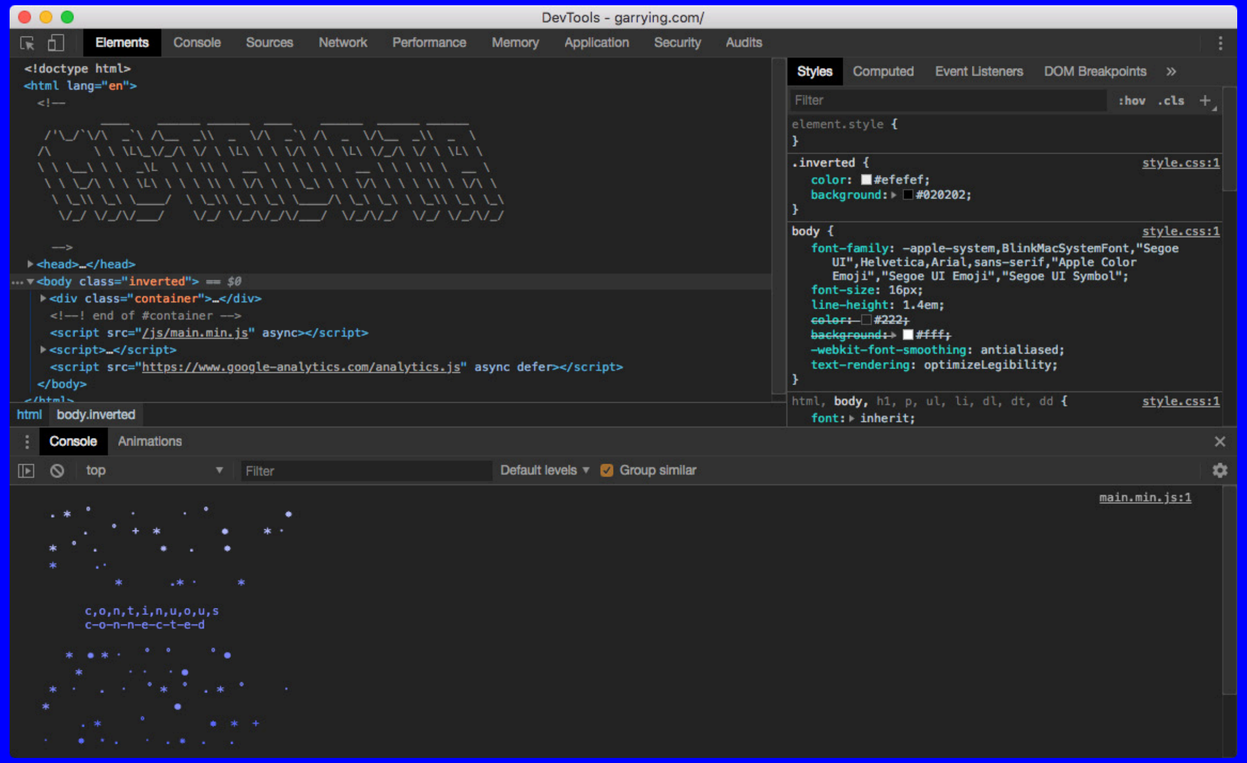
Interaction: JavaScript



Interaction: JavaScript



JavaScript: Console



→ Chrome: ⌘ + ⌥ + P
→ Safari: ⌘ + ⌥ + C

Interaction: JavaScript

Five tasks to try

→ Click on the page to change the background color

→ Click on an element to make it fade out

→ Create a balloon in HTML/CSS & pop it with a click

→ Hover on an image to replace it with another

→ Click on the page to add a background image

References

JavaScript:
https://jquery.com/
https://developer.mozilla.org/bm/docs/Web/JavaScript
https://www.codecademy.com/learn/introduction-to-javascript
https://javascript30.com/
http://youmightnotneedjquery.com/
https://developer.mozilla.org/en-US/docs/Web/Events
https://github.com/bevacqua/es6

CSS:
https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes
https://developer.mozilla.org/en-US/docs/Web/CSS/transition
https://guide.freecodecamp.org/html/responsive-web-design