

INFO2180 LECTURE 4

JAVASCRIPT

**JAVASCRIPT IS A PROGRAMMING
LANGUAGE THAT ADDS
INTERACTIVITY TO YOUR WEBSITE**

[https://developer.mozilla.org/en-US/docs/Learn/
Getting started with the web/JavaScript basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics)

**JAVASCRIPT (JS) IS A LIGHTWEIGHT,
INTERPRETED, PROGRAMMING
LANGUAGE WITH FIRST-CLASS
FUNCTIONS.**

<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

OVERVIEW

OVERVIEW

- ▶ Introduced 1995 in Netscape Navigator Web Browser
- ▶ JavaScript has nothing to do with Java.
- ▶ It was called that as a marketing ploy since Java was popular.
- ▶ Primarily runs in your browser
- ▶ It can now run on the server-side using Node.js

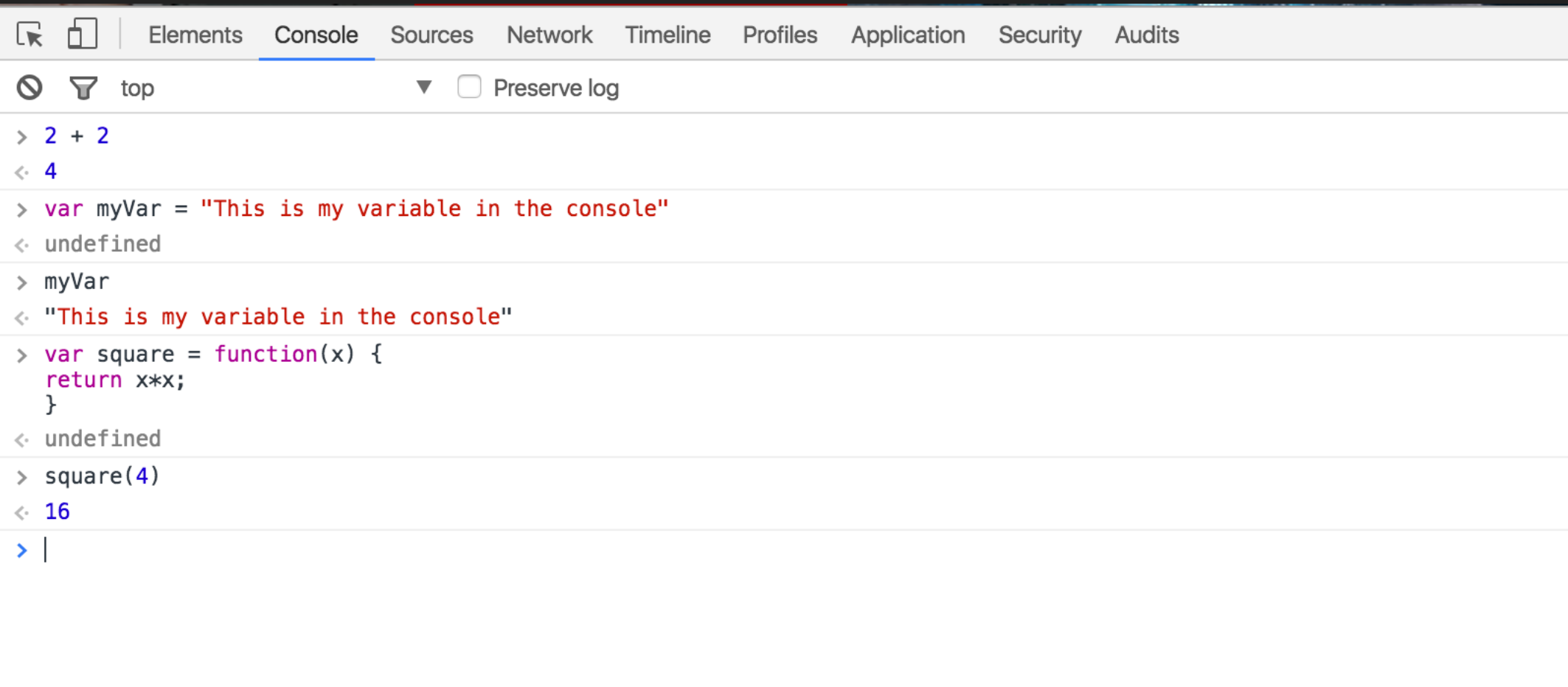
OVERVIEW

- ▶ ECMAScript is the standard. Current version is ECMAScript 2017 or ECMAScript 8 (ES8) as at June 2017.
- ▶ While not all browsers currently support all the new features, you can take advantage of them through a transpiler (e.g. BabelJS)

THE CONSOLE

THE CONSOLE LETS YOU USE STANDARD JAVASCRIPT STATEMENTS AND CONSOLE-SPECIFIC COMMANDS WHILE A PAGE IS LIVE IN THE BROWSER TO HELP YOU DEBUG THE PAGE.

<https://developers.google.com/web/tools/chrome-devtools/debug/console/>



CONSOLE.LOG

You can use `console.log()` instead of `alert()` to help you with debugging your code.

VALUES, TYPES AND OPERATORS

THERE ARE 6 BASIC TYPES OF VALUES

- ▶ Numbers
- ▶ Strings
- ▶ Booleans
- ▶ Objects
- ▶ Functions
- ▶ Undefined values

OPERATORS

- ▶ Unary Operator (e.g. `typeof`)
- ▶ Binary Operators (e.g. `+`, `-`, `*`, `/`, and `%`)
- ▶ Ternary Operator (e.g. `?:` used as short hand if/else)
- ▶ Logical Operators (e.g. `&&`, `||`)
- ▶ Comparison Operators (e.g. `==`, `!=`, `===`, `!==`, `<`, `>`, `<=`, `>=`)

EXAMPLES OF NUMBERS

14

(Integer)

9.81

(Fractional Number)

2.998e8

(Scientific Notation)

EXAMPLES OF ARITHMETIC

$$100 + 4 * 11$$

$$10 / 2$$

$$314 \% 100$$

EXAMPLES OF NUMBERS

What is NaN?

EXAMPLES OF NUMBERS

NaN stands for “not a number”, even though it is a value of the number type. You’ll get this result when you, for example, try to calculate $0 / 0$ (zero divided by zero), **Infinity** - **Infinity**, or any number of other numeric operations that don’t yield a precise, meaningful result.

EXAMPLES OF STRINGS

**“The Quick brown fox jumps over
the lazy dog.”**

**“This is one line\nAnd this is
another”**

EXAMPLES OF STRING CONCATENATION

`"Hello" + " " + "World!"`

EXAMPLES OF BOOLEAN VALUES

true and **false**

EXAMPLES OF COMPARISONS

10 < 12

// → true

15 > 18

// → false

"Aardvark" < "Zoroaster"

// → true

EXAMPLES OF LOGICAL OPERATORS

false || true

// → true

false && true

// → false

EXAMPLES OF TERNARY OPERATOR

```
(3 < 4) ? "It's less than 4" :  
        "It's greater than 4";
```

UNDEFINED VALUES

`null` and `undefined`

UNDEFINED VALUES

A variable that has not been assigned a value is of type undefined. A method or statement also returns undefined if the variable that is being evaluated does not have an assigned value. A function returns undefined if a value was not returned.

EXPRESSIONS AND STATEMENTS

EXPRESSIONS

AN EXPRESSION IS A COMBINATION OF VALUES, VARIABLES, AND OPERATORS, WHICH COMPUTES TO A VALUE.

http://www.w3schools.com/js/js_syntax.asp

SIMPLE EXAMPLE OF AN EXPRESSION

10

"my string"

5 - 2

x * 10

STATEMENTS

STATEMENTS ARE INSTRUCTIONS TO BE EXECUTED AND ARE USUALLY SEPARATED BY SEMICOLONS.

http://www.w3schools.com/js/js_syntax.asp

VARIABLES

EXAMPLES

```
let x = 3;
```

```
let myvar = "my value";
```

```
var someVar = "some val";
```

```
const SOME_NUM = 7;
```

KEYWORDS AND RESERVED WORDS

**break case catch class const continue debugger
default delete do else enum export extends
false finally for function if implements
import in instanceof interface let new null
package private protected public return static
super switch this throw true try typeof var
void while with yield**

These cannot be used as variable or function names.

FUNCTIONS

**FUNCTIONS ARE THE BREAD
AND BUTTER OF JAVASCRIPT
PROGRAMMING.**

http://eloquentjavascript.net/03_functions.html

IT IS A TOOL TO STRUCTURE LARGER PROGRAMS, TO REDUCE REPETITION, TO ASSOCIATE NAMES WITH SUBPROGRAMS, AND TO ISOLATE THESE SUBPROGRAMS FROM EACH OTHER.

EXAMPLE OF A FUNCTION DEFINITION

```
var makeNoise = function() {  
    console.log("Pling!");  
};
```

```
makeNoise();  
// -> Pling!
```

ANOTHER EXAMPLE

```
var power = function(base, exponent) {  
    var result = 1;  
    for (var count = 0; count < exponent; count++)  
        result *= base;  
    return result;  
};
```

```
console.log(power(2, 10));
```

```
// → 1024
```

SHORTHAND EXAMPLE

```
function square(x) {  
    return x * x;  
}
```

ARROW FUNCTIONS EXAMPLE

```
(x, y) => {  
    return x * y;  
}
```

ARROW FUNCTIONS EXAMPLE

$(x) \Rightarrow \{ x * x; \}$

$(x) \Rightarrow x * x;$

CONTROL STRUCTURES

IF/ELSE

```
var num = Number(prompt("Pick a number", "0"));

if (num < 10) {
    alert("Small");
} else if (num < 100) {
    alert("Medium");
} else {
    alert("Large");
}
```

WHILE LOOP

```
var number = 0;  
while (number <= 12) {  
    console.log(number);  
    number = number + 2;  
}
```

FOR LOOP

```
for (var number = 0; number <= 12; number++) {  
    console.log(number);  
}
```

```
// → 0
```

```
// → 1
```

```
// ... etcetera
```

SWITCH

```
switch (prompt("What is the weather like?")) {  
    case "rainy":  
        console.log("Remember to bring an umbrella.");  
        break;  
    case "sunny":  
        console.log("Dress lightly.");  
        break;  
    default:  
        console.log("Unknown weather type!");  
        break;  
}
```

COMMENTS

EXAMPLE OF JAVASCRIPT COMMENTS

```
// This is a single line comment
```

```
/*
```

```
This is a multi-line  
comment
```

```
*/
```

OBJECTS AND ARRAYS

ARRAYS

Arrays allow you to store a list/collection of values.

```
var fruits = ["Apple", "Banana"];
```

```
console.log(fruits[0]);
```

```
// → Apple
```

ARRAY OPERATIONS

```
var newList = fruits.push("Orange");
```

```
// → ["Apple", "Banana", "Orange"]
```

Add to the end of an Array

ARRAY OPERATIONS

```
var newList = fruits.pop();
```

```
// → ["Apple", "Banana"]
```

Remove from the end of an
Array.

ARRAY OPERATIONS

```
var fruits = ["Apple", "Banana"]
```

```
fruits.length;
```

```
// → 2
```

Returns the number of
items in an Array.

OBJECTS

Objects allow you to store a list/collection of values too but with named keys (instead of numbers). It's like an associative array or hash in other languages.

EXAMPLE OF AN OBJECT

```
var myCar = {  
  wheels: 4,  
  "brand name": "Toyota",  
  model: "Corolla",  
  features: ["air condition",  
    "radio"]  
};
```

EXAMPLE OF RETRIEVING A VALUE FROM AN OBJECT

```
myCar.wheels
```

```
// → 4
```

```
myCar['wheels']
```

```
// → 4
```

EXAMPLE OF SETTING A VALUE OF AN OBJECT

```
myCar.year = "2005";
```

```
myCar['year'] = "2005";
```


There is a lot more to Objects and you can mimic OOP principles with them. For more information read http://eloquentjavascript.net/06_object.html.

STRICT MODE

STRICT MODE

- ▶ JavaScript can be made a little more "strict" by enabling strict mode.
- ▶ To do this you would place `"use strict";` at the top of your JS file or the top of a function body.
- ▶ Strict mode for instance will require you to use `var`. (e.g. `var myVariable = "foo";` as opposed to just saying `myVariable = "foo"`).
- ▶ It also disallows giving a function multiple parameters with the same name. (e.g. `function getAnswer(x, x) { //some code }`)
- ▶ Strict mode does a few more things that might be helpful in spotting problems in your code.

ADDING JAVASCRIPT TO YOUR WEBPAGES

EXTERNAL FILES

```
<script src=" ../myscript.js"></script>
```

Usually this is put in the `<head></head>` tags.

INTERNAL/EMBEDDED FILES

```
<script type="text/javascript">
```

```
// My code here
```

```
</script>
```

EXAMPLE USING ATTRIBUTES

```
<button  
onclick="alert( ' Boom! ' );">DO NOT  
PRESS</button>
```

THE DOCUMENT OBJECT MODEL (DOM)

IT IS A PROGRAMMING INTERFACE FOR HTML, XML AND SVG DOCUMENTS. IT PROVIDES A STRUCTURED REPRESENTATION OF THE DOCUMENT AS A TREE.

https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model

```
<!doctype html>
<html>
  <head>
    <title>My home page</title>
  </head>
  <body>
    <h1>My home page</h1>
    <p>Hello, I am Marijn and this is my home
page.</p>
    <p>I also wrote a book! Read it
    <a href="http://
eloquentjavascript.net">here</a>.</p>
  </body>
</html>
```

html

head

title

My home page

body

h1

My home page

p

Hello, I am Marijn and this is...

p

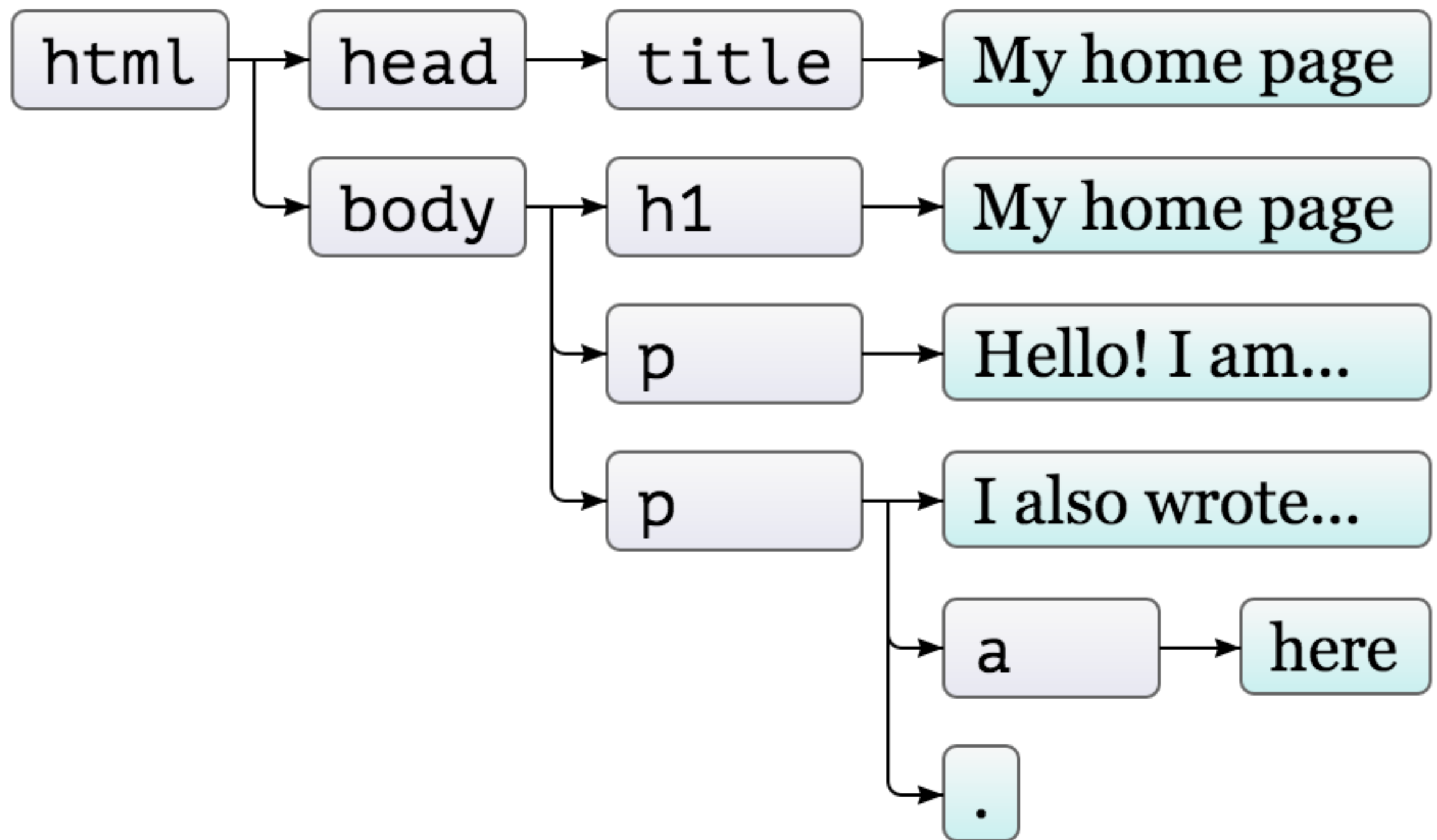
I also wrote a book! Read it

a

here

.

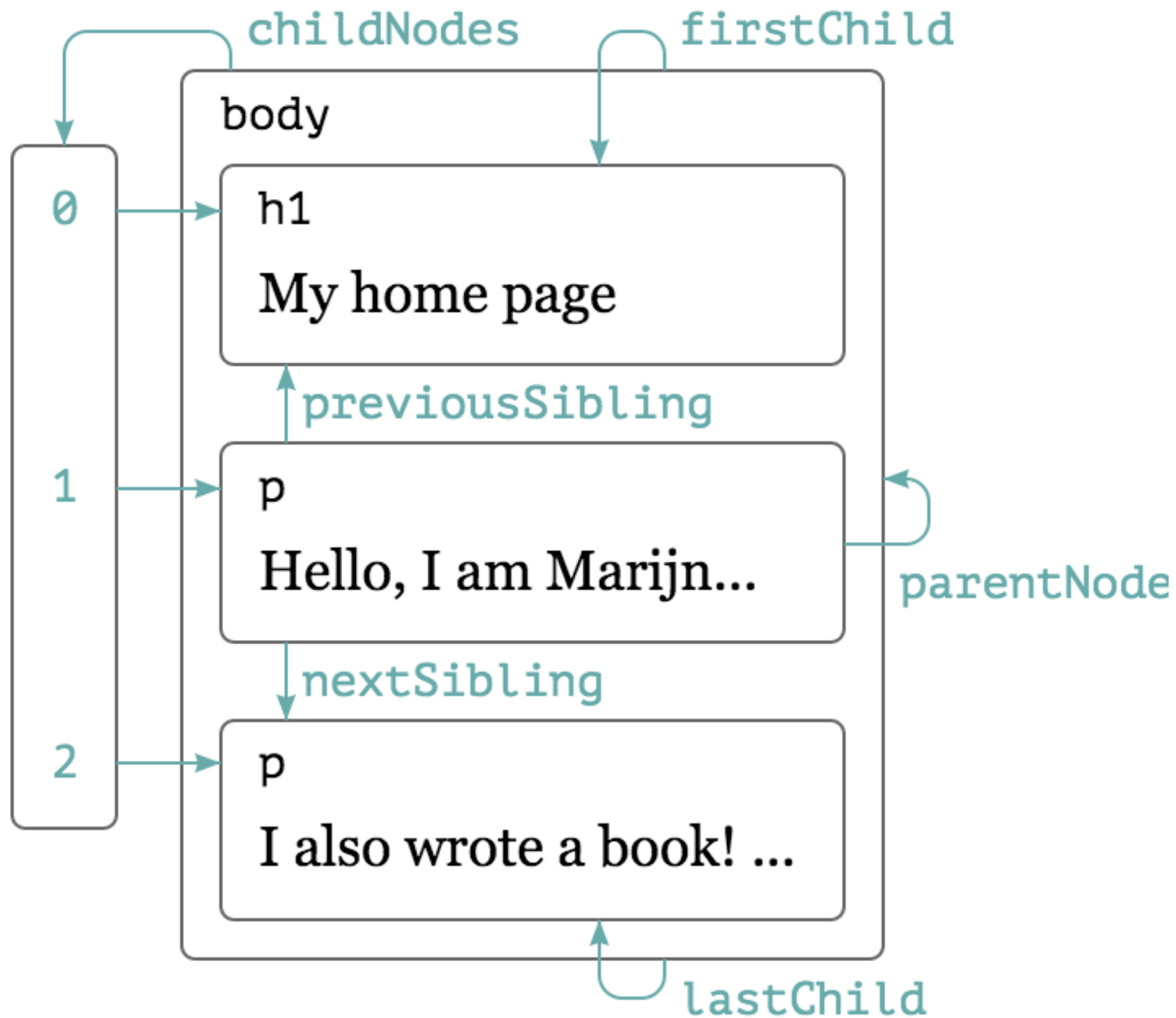
http://eloquentjavascript.net/13_dom.html



document is the global variable that gives us access to all the elements in our HTML document.

How can we traverse the DOM?

The DOM has various properties and methods for traversing the tree. We can use these properties and methods in JavaScript to find elements and manipulate the DOM.



Each node in the tree has a
parentNode and **childNodes**
property.

In addition to those you also have **firstChild**, **lastChild**, **previousSibling** and **nextSibling** properties.

While that is good, sometimes you want to access elements by their tag names, class names or id attributes. How could we do that?

getElementsByTagName()

getElementById()

getElementsByClassName()

EXAMPLES

```
var link =  
document.body.getElementsByTagName("a")[0];  
  
console.log(link.href);  
// → http://eloquentjavascript.net
```

EXAMPLES

```
<p>My profile photo:</p>
```

```
<p></p>
```

```
<script>
```

```
  var photo =
```

```
document.getElementById("profile-image");
```

```
  console.log(photo.src);
```

```
</script>
```

What if you want to change the
DOM?

EXAMPLES OF CHANGING THE DOM

```
var target =  
document.getElementById("one");
```

```
target.removeChild(node)
```

```
target.appendChild(newNode)
```

```
document.body.insertBefore(newNode,  
target)
```


EXAMPLES OF CHANGING THE DOM

```
let someNode = document.getElementById("one");
```

```
let newNode = document.createElement("div");
```

```
someNode.append(newNode);
```

```
someNode.prepend(newNode)
```

```
someNode.before(newNode);
```

```
someNode.after(newNode);
```

```
newNode.remove();
```

Note: These methods are still not yet available in all browsers.

You can also create nodes.

createTextNode()

createElement()

EXAMPLES OF CREATING NODES

```
let myDiv = document.createElement("div");  
let myText = document.createTextNode("my  
text");  
myDiv.append(myText);  
document.body.append(myDiv);
```

We could then change the text in a node by using **innerHTML** or **textContent** properties.

EXAMPLES OF CHANGING THE TEXT IN AN ELEMENT

```
let myDiv =  
document.getElementById("#myDiv");  
myDiv.innerHTML = "<strong>Hello!</strong>";  
// or  
myDiv.textContent = "Hello!";
```

What about attributes?

EXAMPLE OF GETTING THE VALUE OF AN ATTRIBUTE

```
<a href="page.html">Link</a>
```

```
<script>
```

```
document.getElementsByTagName("a")
```

```
[0].href
```

```
</script>
```

```
// → page.html
```


getAttribute()

setAttribute()

EXAMPLE OF GETTING THE VALUE OF AN ATTRIBUTE

```
<a href="page.html">Link</a>
```

```
<script>
```

```
document.getElementsByTagName("a")
```

```
[0].getAttribute("href");
```

```
</script>
```

```
// → page.html
```

What if we want to *add*
or *remove* a **class** on
an element?

We could use the **setAttribute()** method or we can use the **className** property.

EXAMPLE OF ADDING A CLASS TO AN ELEMENT

```
<div class="">Some text</div>
```

```
<script>
```

```
document.querySelector("div").className
```

```
me = "bar";
```

```
</script>
```

We can also use the **classList** property and its associated **add()** and **remove()** methods.

EXAMPLE OF ADDING A CLASS TO AN ELEMENT

```
<div class="foo">Some text</div>
```

```
<script>
```

```
document.querySelector("div").classList.add("bar");
```

```
</script>
```

What if we want to
change the style of an
element?

EXAMPLES OF CHANGING A STYLE

```
var elem = document.getElementById("my-id");
```

```
elem.style.fontSize = "12px";
```

CSS properties that contain a dash (–) have their dashes removed and the letter that follows capitalized. So **font-family** would be **fontFamily**.

Can I access elements like I do
in CSS using selectors?

querySelectorAll()

querySelector()

EXAMPLES OF QUERY SELECTORS

```
document.querySelectorAll(".class-name");
```

```
document.querySelectorAll("div > p");
```

```
document.querySelector("ul li");
```

`querySelector()` will return only the first matching element.

EVENT HANDLING

**EVENT HANDLERS MAKE IT POSSIBLE
TO DETECT AND REACT TO EVENTS WE
HAVE NO DIRECT CONTROL OVER.**

THERE ARE MANY DIFFERENT EVENT TYPES

- ▶ mouse events (e.g. **click**, **mouseover**, **mousedown**, **mouseup**, **mousemove**)
- ▶ key events (e.g. **keypress**, **keyup**, **keydown**)
- ▶ load events (e.g. **load**, **unload**, **beforeunload**, **DOMContentLoaded**)
- ▶ focus events (e.g. **focus**, **blur**)
- ▶ etc...

ATTACHING EVENTS

- ▶ You can attach events directly in HTML (inline) as an attribute

```
<a href="#" onclick="myFunction();">Link</a>
```

- ▶ Or using the DOM property with JavaScript
`document.getElementById("my-id").onclick = myFunction();`

- ▶ Or using an event listener

```
target.addEventListener("click", function() { //  
some code here});
```


EXAMPLE

```
<button>Click me</button>
```

```
<p>No handler here.</p>
```

```
<script>
```

```
  var button = document.querySelector("button");
```

```
  button.addEventListener("click", function() {
```

```
    console.log("Button clicked.");
```

```
  });
```

```
</script>
```

Mouse events also allow access to properties that tell you the mouse position.

clientX/clientY

(in relation to the viewport)

pageX/pageY

(in relation to the top left of document)

offsetX/offsetY

(in relation to an element)

screenX/screenY

(in relation to the screen)

Many events have default actions associated with them. How do we prevent the default action from occurring?

A link for example when clicked will go to the URL specified in the href attribute.

To prevent the default action
we can use the **preventDefault**
method.

EXAMPLE

```
<a href="https://developer.mozilla.org/">MDN</a>
<script>
  var link = document.querySelector("a");
  link.addEventListener("click", function(event) {
    console.log("Nope.");
    event.preventDefault();
  });
</script>
```

RESOURCES

- ▶ Eloquent JavaScript - <http://eloquentjavascript.net/>
- ▶ Mozilla Developer Network - [https://
developer.mozilla.org/en-US/docs/Web/JavaScript/Guide](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)
- ▶ W3Schools - <http://www.w3schools.com/js/>
- ▶ The Modern JavaScript Tutorial - <http://javascript.info/>
- ▶ JSLint - <http://jshint.com/>
- ▶ Learn ES2015 - <https://babeljs.io/docs/learn-es2015/>