JavaScript

Saac Sornchai

3

languages

all web developers must learn

HTML

1/3

The Content of Web Pages

CSS

2/3

The Layout of Web Pages

JavaScript

3/3

The behavior of Web Pages

What can JavaScript do?

Change HTML Content/Attribute

Change CSS Style

Show/Hide HTML Element

etc.

Where to Insert JavaScript?

Inline JavaScript in HTML Event Attributes

```
<button onclick="/* inline JS codes */">
   Click Me
</button>
```

Inline JavaScript codes in HTML Event Attributes

Internal JavaScript code must be insert between <script> and </script> tags

```
<script>
    /* internal JavaScript codes */
</script>
```

Internal JavaScript

External JavaScript code can be placed in external file

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

External JavaScript codes in .js file

Insert <script> tag in <head> or <body>

```html
<head>
  <!-- other meta tag -->

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

Insert <script> tag inside <head> tag

```
<body>
  <!-- other HTML elements -->

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

Insert <script> tag inside <body> tag

Improves the Display Speed

Placing scripts at the bottom of the <body> element

because script compilation slows down the display

The `type` attribute is not required

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

JavaScript is the default scripting language in HTML

JavaScript Output

4

different ways

to display JavaScript data

1/4 using innerHTML

Writing into an HTML Element

```
<body>

  <p id="output"></p>



  <script>
     document.getElementById("output").innerHTML =
         "Text to display";
  </script>

</body>
```

2/4 using document.write()

Writing into HTML Output (delete all HTML elements)

```
<body>

  <p>Some paragraph</p>

  <button onclick="document.write(5 + 6)">
      Try it
  </button>

</body>
```

3/4 using window.alert()

Writing into an alert box

```
<body>

  <p>Some paragraph</p>



  <script>
     window.alert("Text to display");
  </script>

</body>
```

4/4 using console.log()

Writing into the browser console (for debugging purpose)

```
<body>

  <p>Some paragraph</p>



  <script>
     console.log("Text to display");
  </script>

</body>
```

7

Data types
(ECMAScript Standard)
6 primitive types + object

Boolean

1/7

- `true`
- `false`

null

2/7

- `null` (case-sensitive)

undefined

3/7

- `undefined` (case-sensitive)

Number

4/7

- `123`
- `3.1415926`
- `-3.1E12`
- `-.123456789`
- `.1e-23`

Number

4/7

double-precision 64-bit floating point format IEEE 754

- `123`
- `3.1415926`
- `-3.1E12`
- `-.123456789`
- `.1e-23`

String

5/7

- `"Double quote"`
- `'Single quote'`
- `` `Backtick` ``

Symbol

6/7

- `Symbol()`

Object

7/7

- `{key: value}`

Javascript is a dynamically typed language

Declaration

var

- Declares a variable
- Optional initializing it to a value

let

- Declares a block-scoped local variable
- Optional initializing it to a value

const

- Declares a block-scoped read-only named constant

Variables

The name of variables called **identifiers**

JavaScript identifier

- must start with a letter (A-Z, a-z), underscore (_) or dollar sign ($)
- subsequence character can also be number (0-9)
- case-sensitive
- can use most of ISO 8859-1 or Unicode letters in identifier

3

ways to declare a variable

1/3 with keyword `var`

Declare both local and global variable

```
var a;

if (true) {

   var x = 32

}

// in JavaScript semicolon (;) is optional

console.log(a) // undefined

console.log(x) // 32
```

2/3 with no keyword

Always declare global variable, outside any function

```
gbNumber = 30.1

// generates a strict JavaScript warning

if (true) {

   console.log(gbNumber) // 30.1

}
```

3/3 with keyword `let`

Declare block-scoped local variable

```
var total = 0

for (let i = 0; i < 10; i++) {

   total += i

}

console.log(total) // 45

console.log(i)

// Uncaught ReferenceError: i is not defined
```

undefined v.s. null

Numeric Context

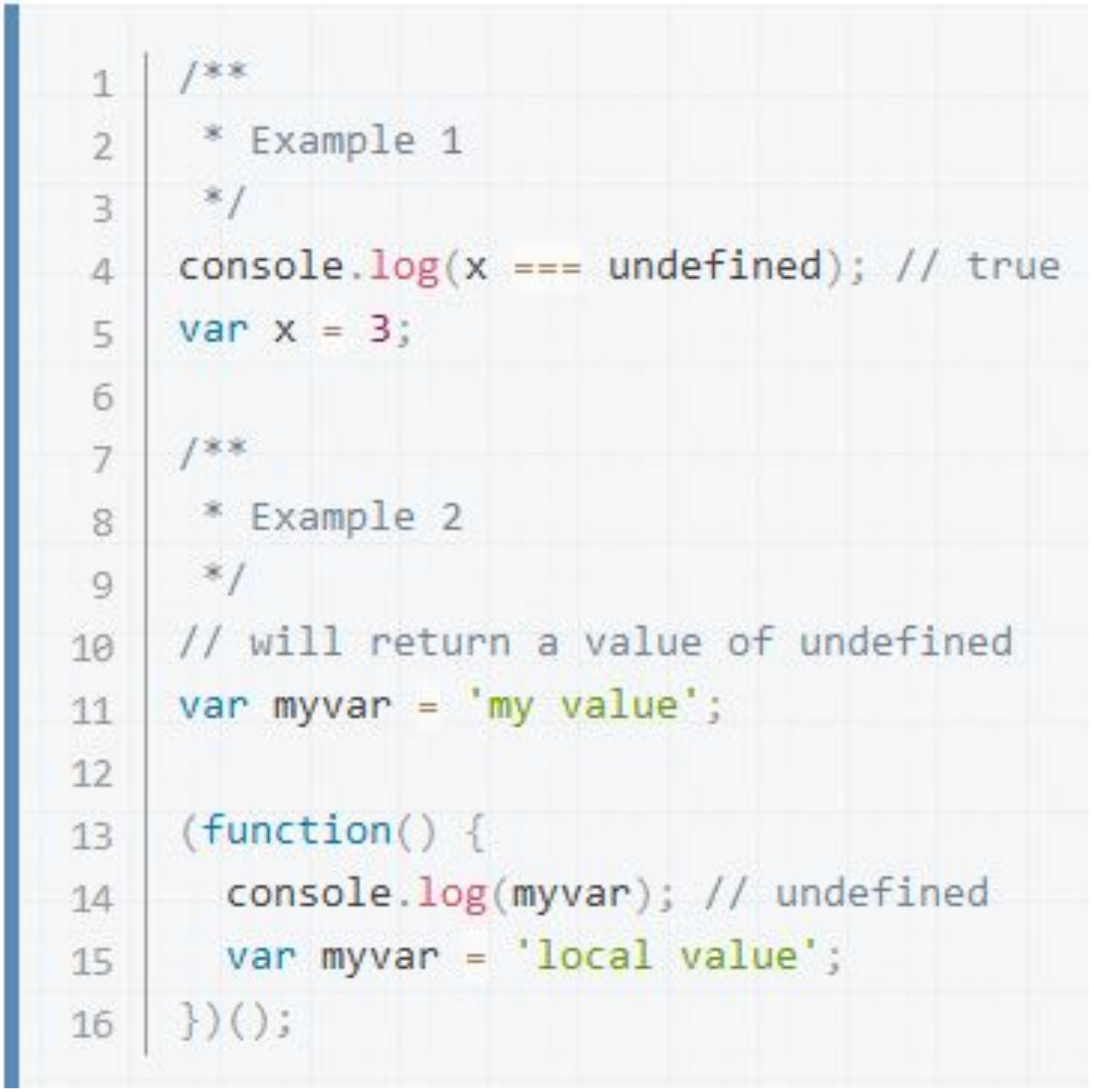
- `undefined` convert to `NaN`
- `null` behaves as `0`

Boolean Context

- `undefined` behaves as `false`
- `null` behaves as `false`

Hoisting

Variable Hoisting

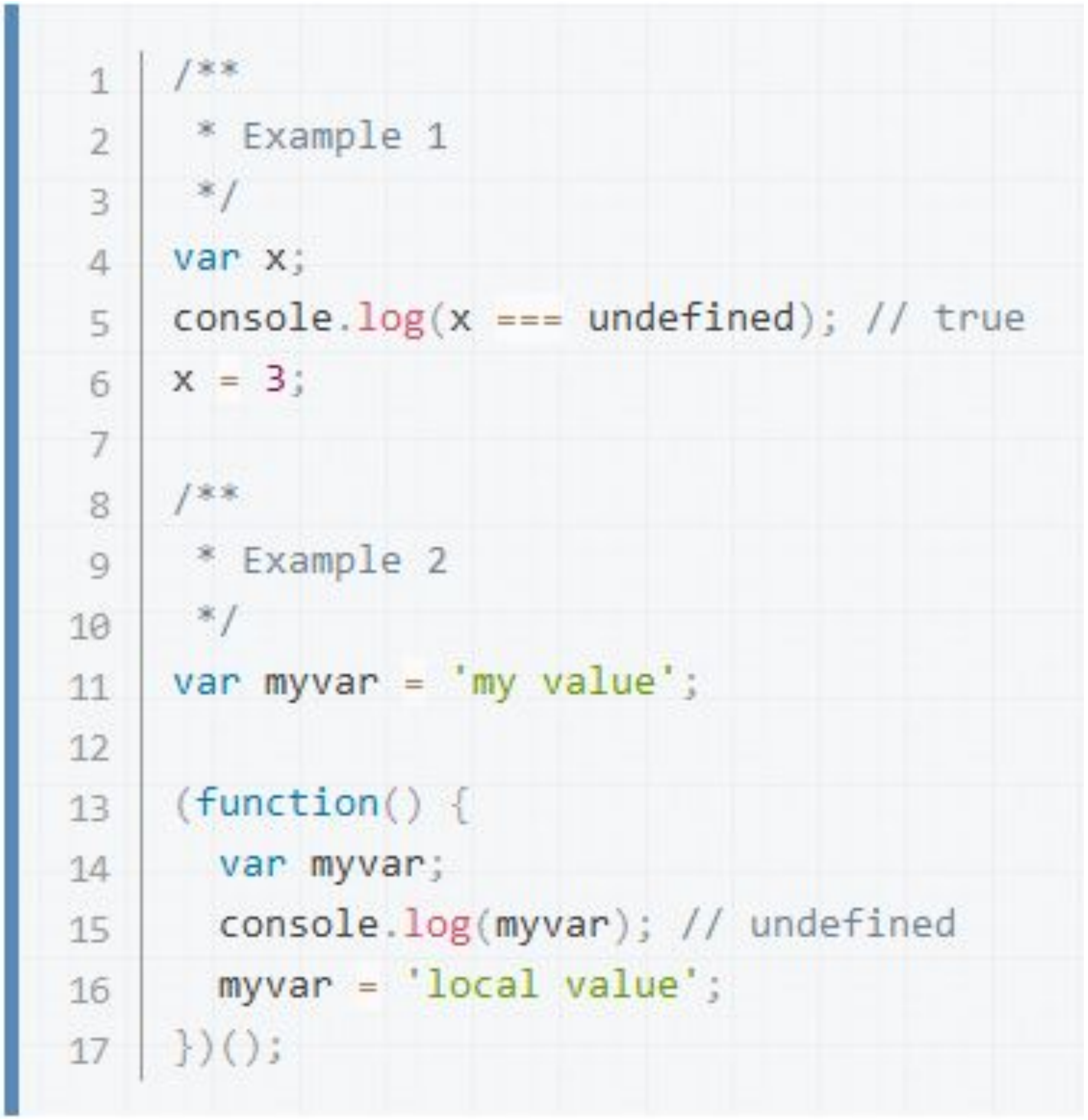


```
/**
 * Example 1
 */
console.log(x === undefined); // true
var x = 3;

/**
 * Example 2
 */
// will return a value of undefined
var myvar = 'my value';

(function() {
  console.log(myvar); // undefined
  var myvar = 'local value';
})();
```

Variable Hoisting



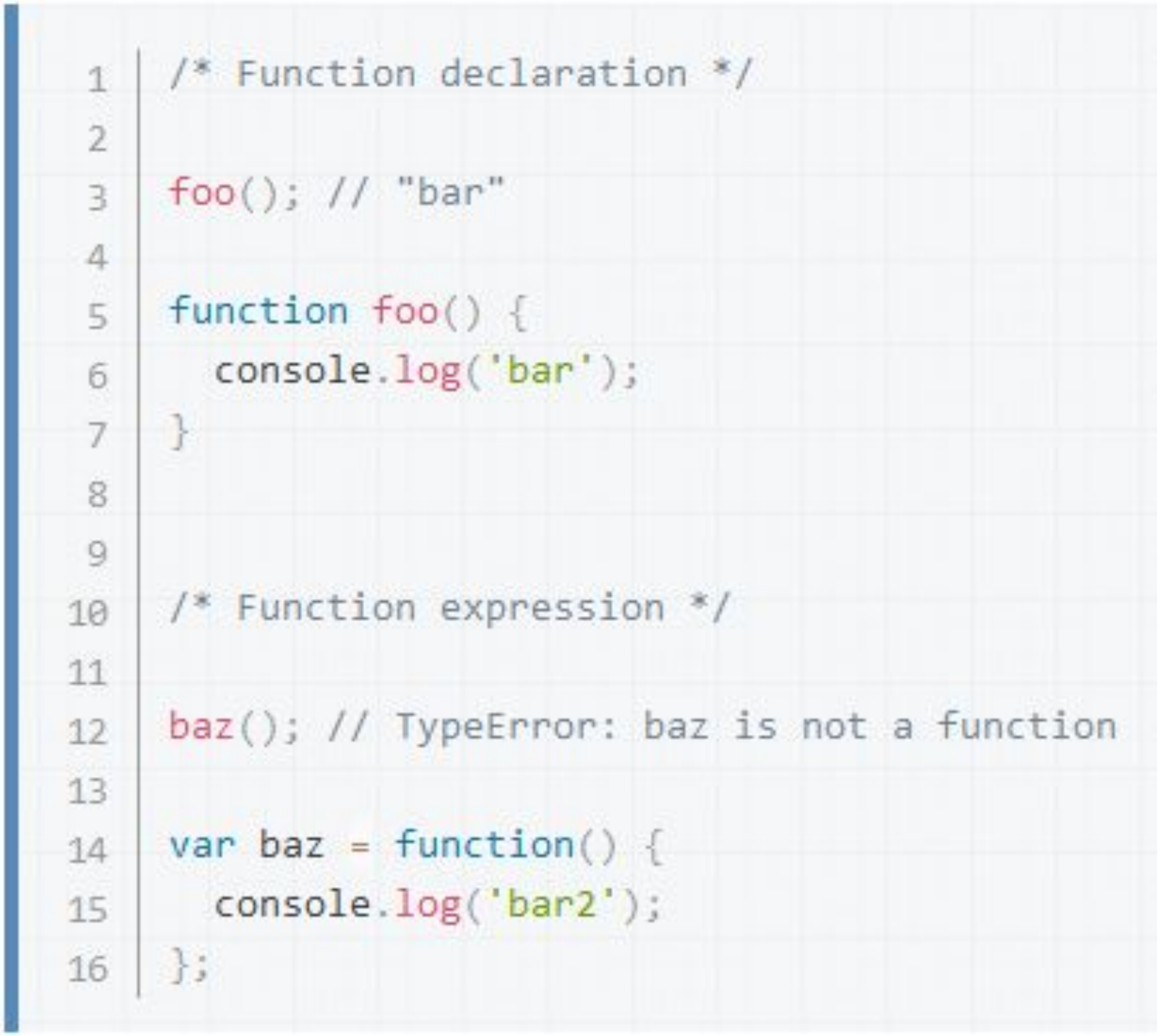
```
/**
 * Example 1
 */
var x;
console.log(x === undefined); // true
x = 3;

/**
 * Example 2
 */
var myvar = 'my value';

(function() {
  var myvar;
  console.log(myvar); // undefined
  myvar = 'local value';
})();
```

`let` and `const`
will not hoist

Function Hoisting



```
/* Function declaration */

foo(); // "bar"

function foo() {
  console.log('bar');
}


/* Function expression */

baz(); // TypeError: baz is not a function

var baz = function() {
  console.log('bar2');
};
```

Constants

JavaScript constant

- Cannot change value through assignment or be re-declared
- Has to be initialized to value
- Scoped rules for constant are the same as `let`
- Cannot declare constant with the same name as a function or a variable in the same scope
- The properties of objects assigned to constants are not protected

Fixed Value Literals

(Not Variables)

Array Literals

Square brackets [ ]

```
var colors = ['red', 'green', 'blue']
```

```
var emptyArray = []
```

An array literal is a type of object initializer.

Boolean Literals

2 literal values

```
const JS_IS_EASY = true
```

```
var isNotTrue = false
```

Integer Literals

Decimal, hexadecimal, octal and binary

```
0, 117 and -345 (decimal, base 10)

015, 0001 and -0o77 (octal, base 8)

0x1123, 0x00111 and -0xF1A7
                 (hexadecimal or "hex",  base 16)

0b11, 0b0011 and -0b11 (binary, base 2)
```

Floating-point Literals

Decimal point, fraction, exponent

```
[(+|-)][digits][.digits][(E|e)[(+|-)]digits]
```

```
3.1415926
-.123456789
-3.1E+12
.1e-23
```

Object Literals

Curly brackets { }

```
var foo = {a: 'alpha', 2: 'two'}

console.log(foo.a) // alpha (dot notation)

console.log(foo.2) // Error: missing ) after argument list

console.log(foo[2]) // two (array-like notation)

console.log(foo[a]) // Error: a is not defined

console.log(foo['a']) // alpha

console.log(foo['2']) // two
```

RegExp Literals

Slashes / /

```
var re = /ab+c/
```

String Literals

Single Quotes ' ', Double Quotes " "

```
'foo'

"bar"

'First line\nAnother line'

"Mike's cat"
```

String Literals

Backtick ` `

```
// Basic literal string creation
`She said: "I'm the 6th adventurer."`


// String interpolation
var name = 'Mike', time = 'today'
`Hello ${name}, how are you ${time}?`
// Hello Mike, how are you today?
```

Control Flow
Conditional Statements

Block Statement

```
var x = 1

{

   var x = 2

}

console.log(x)

// 2
```

```
let x = 1

{

   let x = 2

}

console.log(x)

// 1
```

if...else Statement

```
if (condition_1) {
    statements_1
} else if (condition_2) {
    statements_2
} else {
    statements_3
}
```

Falsy Value

- `false`
- `undefined`
- `null`
- `0`
- `NaN`
- `""` (empty string)

switch Statement

```
switch (expression) {
    case label:
        statements_1
        [break]
    ...
    default:
        statements_def
}
```

throw Statement

```
throw expression
```

```
throw 'Error404' // String Type
throw -1         // Number Type
throw true       // Boolean Type
throw new UserException('Value too high')
```

try...catch Statement

```
try {
    monthName = getMonthName(month)
} catch (e) {
    console.log(e)
} finally {
    closeConnection()
}
```

Control Flow
Loop and Iteration

`for` Statement

```
for ([initExp]; [condition]; [incrementExp])
   statements
```

```
var total = 0, count = 0
for (let i = 0; i < 30; ++i) {
   total += i; count += 1;
}
```

`do...while` Statement

```
do

    statements

while (condition)
```

`while` Statement

```
while (condition)
    statement
```

`labeled` Statement

```
labelName:
   statement
```

`break` Statement

```
break [labelName]
```

break Statement

```
var x = 0;
var z = 0;
labelCancelLoops: while (true) {
  console.log('Outer loops: ' + x);
  x += 1;
  z = 1;
  while (true) {
    console.log('Inner loops: ' + z);
    z += 1;
    if (z === 10 && x === 10) {
      break labelCancelLoops;
    } else if (z === 10) {
      break;
    }
  }
}
```

`continue` Statement

```
continue [labelName]
```

continue Statement

```
checkiandj:
  while (i < 4) {
    console.log(i);
    i += 1;
    checkj:
      while (j > 4) {
        console.log(j);
        j -= 1;
        if ((j % 2) == 0) {
          continue checkj;
        }
        console.log(j + ' is odd.');
      }
      console.log('i = ' + i);
      console.log('j = ' + j);
  }
```

`for...in` Statement

```
for let prop in object

    statements
```

for...of Statement

```
for let value of object

    statements
```

Learning JavaScript

www.codecademy.com/learn/learn-javascript

JavaScript Tutorial

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

jQuery

http://jquery.com

Tutorial

- try.jquery.com
- www.codecademy.com/learn/jquery