JavaScript quick start

Maktab Sharif Front-End Bootcamp
Winter - 2018
Alireza Riahi

JavaScript Introduction

JavaScript Programs

Mostly, JavaScript runs in your web browser alongside HTML and CSS, and can be added to any web page using a script tag. The script element can either contain JavaScript directly (internal) or link to an external resource via a src attribute (external).

A browser then runs JavaScript line-by-line, starting at the top of the file or script element and finishing at the bottom (unless you tell it to go elsewhere).

Internal Scripts

You can just put the JavaScript inside a script element:

```
<script>
    alert("Hello, world.");
</script>
```

External Scripts

An external JavaScript resource is a text file with a .js extension. To add a JavaScript file to your page, you just need to use a script tag with a src attribute pointing to the file. So, if your file was called script.js and sat in the same directory as your HTML file, your script element would look like this:

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

JavaScript Variables

Variables and Data

Storing data so we can use it later is one of the most important things when writing code. Fortunately, JavaScript can do this!

Variables is declaring for store some data! its meaning variables have data! So in programming language variables datas called value!

Each variable has a name. We call and use variables with their names.

So now we know a variable has a name and a value!

You can name a variable anything you like, but its best to name it in a way that tells what you’ ve stored there. For example, surname is better than a or b and …

There are two parts to creating a variable; declaration and initialization.

JavaScript Variables

Declaration Variable

Declaration is declaring a variable to exist.

As above, to declare a variable, use the var keyword followed by the variable name, like this:

```
var surname;
var age;
```

Notice those semicolons (;)!!! Almost every line in JavaScript ends in a semicolon - you'll be using them a lot.

Initialization Variable

Initialization is giving a variable its value for the first time. The value can change later, but it is only initialized once. You initialize a variable using the equals sign (=). You can read it as "the value of the variable on the left should be the data on the right":

JavaScript Variables

```
var name = “Tom";
```

“Tom” is a string - a collection of letters. A string is surrounded by single or double quote marks.

```
var age = 20;
```

20 is just a number - and numbers don’t go in quotes.

Assignment

As mentioned, you can set a variable’s value as many times as you like.

It’s called assignment and it looks very similar to initialization. You again use the equals sign, but there’s no need for the var keyword because we’ve already declared the variable.

```
name = "Andy";
age = 43;
```

Only do this if you’ve declared the variable using the var keyword!

JavaScript Operators

Mathematical symbols are called operators; We can use math operators in javascript like this:

```
var firstNum = 20;
var secNum = 10;

var sum = firstNum + secNum; // 20 + 10 = 30
var sub = firstNum - secNum; // 20 - 10 = 10
var mult = firstNum * secNum; // 20 * 10 = 200
var div = firstNum / secNum; // 20 / 10 = 2
var rem = firstNum % secNum; // 20 % 10 = 0
var rem = firstNum % 11; // 20 % 11 = ?

var increment = ++firstNum; // 20 + 1 = 21
var decrement = --firstNum; // 20 - 1 = 19
```

Notice operators are used like math:
(10 + 2) / 2 + 4 * 2; // 12 / 2 + 4 * 2
The part in brackets is worked out first!

JavaScript Comparisons

logic

A really important part of programming is being able to compare values in order to make decisions in code. When a comparison is made the outcome is either true or false; a special kind a of data called a boolean. This is logic.

Equality

To find out when two values are equal, use the **triple equals** operator (“===”).

```
15.234 === 15.234 // true
```

We can also determine if two values are not equal using the **triple not equal** operator (“!==”).

```
15.234 !== 18.4545 // true
```

It’s important to know that strings containing a number and an actual number are **not equal**.

```
'10' === 10 // false
```

JavaScript Comparisons

Greater than and less than

A Comparing two numbers is useful, for example, to determine which of two is larger or smaller. This first example is a comparison of 10 and 5 to see if 10 is larger, using the **greater than** operator (“>”).

```
10 > 5 // true
```

Next we use the **less than** operator (“<”) to determine if the left value is smaller.

```
20.4 < 20.2 // false
```

Combining a comparison of equality and size can be done with the **greater than or equal to** and **less than or equal to** operators (“>=” and “<=” respectively).

```
10 >= 10 // true
10 <= 5 // false
```

JavaScript Conditional

Logic is used to make decisions in code; choosing to run one piece of code or another depending on the comparisons made. This requires use of something called a **conditional**. There are a few different conditionals that you might want to use, but we'll just focus the one used most commonly: `if`.

It's very simple: if some logic (the **condition**) is true, run a **block** of code. You can also supply more code to be run if the condition is not true, and supply additional conditions and blocks of code to optionally be run. These forms are called **if-else**, as you'll see below.

```
if (10 > 5) {
    // Run the code in here
}
```

The code between the braces - "{" and "}" - is called a **block**, and this one is linked to the `if` statement. It's only run if the **conditional** (between the parentheses) is **true**.

JavaScript Conditional

The if-else form of an `if` statement is used to run an alternative piece of code if the conditional is not true. The code in the `if` block below will be ignored, for example - only the code in the `else` block will be run.

```
if (43 < 2) {
    // Run the code in here
} else {
    // Run a different bit of code
}
```

JavaScript Loops

Loops are a way of repeating the same block of code over and over. A `for` loop is similar to an `if` statement, but it combines three semicolon separated pieces information between the parentheses: **initialization**, **condition** and a final **expression.** The **initialization** part is for creating a variable to let you track how far through the loop you are; the **condition** is where the looping **logic** goes; and the final expression is run at the end of every loop.

```
for (var i = 1; i < 10; i++) {
    alert(i);
}
```

This gives us alert boxes containing the numbers 1 to 10 in order.

JavaScript Functions

Functions are reusable blocks of code that carry out a specific task. To **execute** the code in a function you **call** it. A function can be passed **arguments** to use, and a function may **return** a value to whatever called it. To create a function, use the `function` keyword before function name. You then list the arguments in parentheses, and then supply a block that contains the function's code. Here's a function that adds two numbers:

```
function add (a, b) {
    return a + b;
};
```

a and **b** are the function's parameters, and the value it returns is signified by the `return` keyword. The `return` keyword also stops execution of the code in the function; nothing after it will be run.

```
var result = add(1, 2); // result is now 3
```

JavaScript Arrays

Arrays are lists of any kind of data, including other arrays. Each item in the array has an **index** — a number — which can be used to retrieve an **element** from the array.

The indices start at 0; that is, the first element in the array has the index `0`, and subsequent elements have incrementally increasing indices, so the last element in the array has an index one less than the length of the array.

You can find the number of elements in the array using its `length` property:

```
var emptyArray = [];
var shoppingList = ['Milk', 'Bread', ‘Beans'];
shoppingList[0]; // result is Milk
shoppingList.length; // result is 3
```

You can use `push` and `pop` methods to add and remove elements from the end of the array:

```
shoppingList.push('A new car’); // ['Milk', 'Bread', 'Beans',
'A new car']
shoppingList.pop(); // ['Milk', 'Bread', 'Beans']
```

JavaScript Arrays

JavaScript Array simple example:

```
function salam (personName) {
    return "salam " + personName;
}

var people = ['Mohammad', 'Fatemeh', 'Ali'];

people.push('Hasan');
people.push(‘Hosein');
people.push('Ahmad');

people.pop();

for (var i=0; i < people.length; i++) {
    var greeting = salam(people[i]);
    console.log(greeting);
}
```

Finished

for deep information you can visit

www.tutorialspoint.com

www.w3schools.com

Alireza Riahi
ali.r.riahi@gmail.com

Maktab Sharif - Winter 2018