Școala
informală
de IT

JavaScript Basics

Agenda

- What Javascript is and why it’s useful
- How and where to include JavaScript in a web page
- Values & Variables
- Primitive Data Types & Operators
- Arrays and Objects
- Flow control: selection and repetition structures

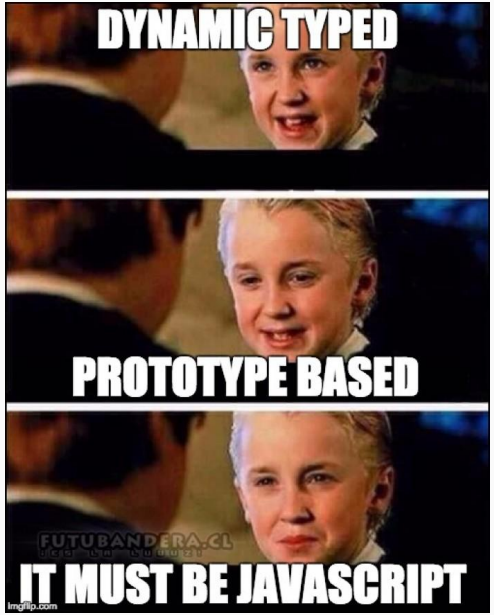
JavaScript history & characteristics

A brief history of JavaScript

- Programming language developed in 1995 by Brendan Eich
- Main goal: language for the browser, useful to make the page dynamic
- It's not Java!
- ECMAScript = the specification for JavaScript

Main characteristics

- **Scripting language:** designed specifically for acting on an existing entity or system
- **Dynamic-typed:** types are associated with values not variables; let's say x can be a number and later on a string
- **Object-based:** almost everything is an object; it is an object-oriented language
- **Prototype-based:** uses prototype where other languages use classes as inheritance
- **Functional:** JavaScript functions are “first class citizens”: they can be composed, sent as parameters ...



Including JavaScript in a Web Page

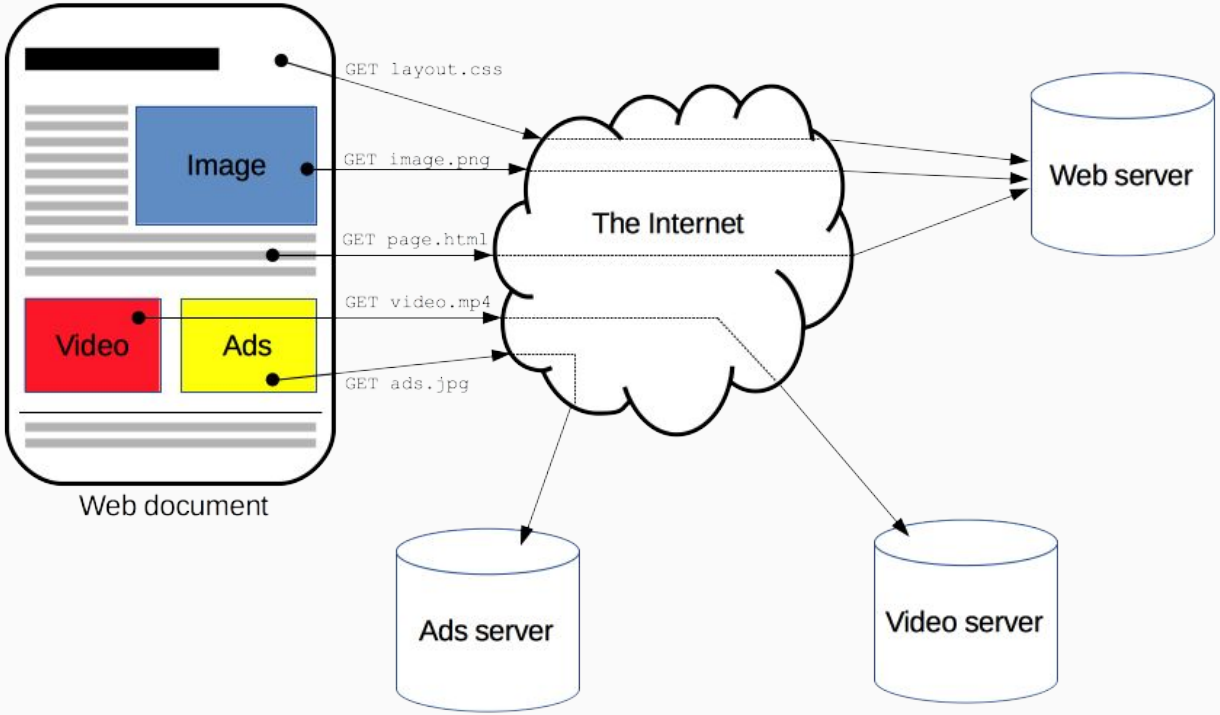
Including JavaScript in a Web Page

1. <script> tag inside the HTML file
2. <script src='relative-path/external-file.js'> tag to reference an external file (using a relative path to that file)
3. <script src='https://web-path/external-file.js'> tag to reference a file on the Internet (using the link to that file)

Question: Where do we place JavaScript files?

- Inside the head tag
- Inside the body tag
- window.onload

How The Web Browsers Work



A complete document is reconstructed from the different sub-documents fetched: text, layout description, images, videos, scripts

How The Web Browsers Work

1. Get HTML
2. Parse head
   - Make requests to get the referenced files *in the order of their appearance*
   - => place .css and font files first
3. Parse body
   - Create DOM and CSSOM, merge them and render the page
   - Make requests to get the referenced files
   - => for most use cases, place .js files as the last tags in the body

Values & Variables

Values & Variables

- Values = the simplest components in JavaScript
  - Examples: 17, “JavaScript”, true, function() {}
  - As seen above, values have different types
- Variables are used to store values - several ways of doing this:
  - Declare, then initialize with a value (2 statements)
  - Declare and initialize on the same line
  - Re-assign values later

```
var x;
x = 2;
console.log(x);

var x = 2;
console.log(x);

var x = 2;
console.log(x);
x = 5;
console.log(x);
```

Naming Variables

- **Rules:**
  - **A variable name must begin with a letter, $ or _**
  - **It must contain only letters, numbers, $ or _**
  - **They are case sensitive: *var hello* is different than *var HELLO***
  - **Avoid reserved words**

```
// good
var name;
var fullName;
var $body;
var _sum;
var Car;

// bad
var 4Sparta;
var imSoHappyClapAlong!
```

Primitive Data Types & Operators

Primitive Data Types

- **Number**
  - Stored on 64 bits
  - Special numbers: Infinity, -Infinity, NaN
- **String**
  - Use single quotes, double quotes, or backticks to mark strings (the quotes at the start and the end of the string match)
  - Character escaping
- **Boolean**
- **Null** - an explicitly empty value
- **Undefined** - a value that hasn't been defined

```
var age = 25;
var price = 3.99;

var name = 'John Doe';
var restaurant = "John's pizzeria";
var bar = `Irish Pub`;
var paragraph = 'Line 1 \nLine 2';

var isTrue = true;
var isFalse = false;

var thisIsNothing = null;

var thisIsNotDefined;
```

Operators

- **Arithmetic**
  - Addition: +
  - Subtraction: -
  - Multiplication: *
  - Division: /
  - Modulus: %
- **Comparison**
  - >, <, >=, <=
  - Equality operator: == & ===

```
5 + 5           // 10
10 - 7          // 3
2 * 3           // 6
12 / 4          // 3
33 % 10         // 3
(5 + 5) * 4     // 40

3 < 4   // true
3 > 10  // false
5 <= 5  // true
8 >= 7  // true

2 == 2   // true
2 == '2' // true

2 === 2   // true
2 === '2' // false
```

Operators

- **Logical**
  - **AND &&**
  - **OR ||**
  - **NOT !**

In JS evaluation stops the logical operator as soon as it knows the answer

- **Unary: typeof, -**
- **Ternary: ?:**

```
2 > 0 && 0 > 2  // false
2 > 0 || 0 > 2  // true
!(0 < 2)        // false

typeof 10       // number
typeof "text"   // string
typeof true     // boolean
typeof x        // undefined

var thisIsNothing = null;
typeof thisIsNothing  // object

var age = condition ? trueVal : falseVal;
```

Expressions

Beside primitive data types variables can store the result of expressions

```
var x = 1 + 1; // 2
var y = x * 2; // 4
var firstName = "Chuck";
var lastName = "Norris";
var fullName = firstName + ' ' + lastName;

var isOdd = x % 2 === 1  // false (x = 2)
...
```

(Introduction to) Arrays & Objects

Arrays

- Arrays are list-like objects that hold ordered list of values.
- In JS the length of an array is not fixed.
- Elements type is not fixed also. Anything can be in an array.
- How to create an array?
  - Using an array literal (recommended)
  - Using the JavaScript new keyword

```
// array literal
var arr = ['first element', 'second element'];

// new keyword
var arr = new Array('first element', 'second element');
```

Arrays

- **How do we access the data stored in Array?**
  - **You refer to an array element by referring to the index number.**
- **Array properties and methods**
  - **Length** - The length property of an array returns the number of array elements
  - **Sort** - The sort() method sorts the elements of an array in place and returns the array.The default sort order is according to string *Unicode code points*.
  - There are plenty of array methods!

```
console.log(arr[0]);
console.log(arr[1]);
console.log(arr[arr.length - 1]);

// flexible element's type
var mix = [1, 'two', ['apple', 'orange']];
console.log(mix[2]);

// changing arrays
mix[2] = 3;
mix[3] = [1, 2, 3];
console.log(mix);

arr.sort()
arr.sort(compareFunction)
```

Objects

- Key-value pairs
- Comma-delimited
- Wrapped in curly braces

```
var robot = {
    model: 'TRX1000',
    weight: 180,
    color: 'red'
}
```

Key

Value

Objects

- **Objects can have properties that are functions.**
- **A property that is a function is called method.**

```
var robot = {
  model: 'TRX1000',
  color: 'red',
  walk: function() {
    console.log('robot is walking');
  },
  run: function() {
    console.log('running');
  },
  attack: function() {
    if (window.confirm('are you sure?')) {
      alert('attacking!');
    }
  }
};

console.log(robot.model);
robot.run();
robot.attack();
```

This is an object method.
And it represents a dynamic action

WAT?

JavaScript WAT? video

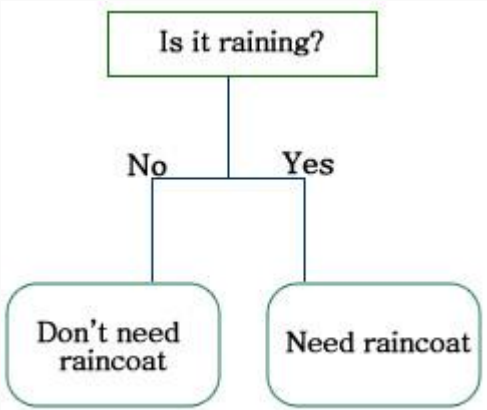
Control Structures

IF statement

```
if (condition) {
  // statement
}

// example
var x = 10;
if (x > 0) {
  console.log('it is a positive number');
}
```

IF/ELSE statement




```
if (condition) {
  // statement
} else {
  // alternative condition
}
```

```
if (condition1) {
  // statement
} else if (condition2) {
  // alternative condition
} else if (condition3) {
  // alternative condition
} else {
  // final alternative
}
```

SWITCH statement

```
if (weather === 'rainy') {
  console.log("Bring an umbrella.");
} else if (weather === 'sunny') {
  console.log("Dress lightly.");
} else if (weather === 'sunny') {
  console.log("Go outside.");
} else {
  console.log("Unknown weather type!");
}
```

```
switch (weather) {
  case 'rainy':
    console.log("Bring an umbrella.");
    break;
  case 'sunny':
    console.log("Dress lightly.");
    break;
  case 'cloudy':
    console.log("Go outside.");
    break;
  case:
    console.log("Unknown weather type!");
    break;
}
```

Repetitive structures

```
var scoops = 5;
while (scoops > 0) {
   document.write("Another scoop!<br>");
   scoops = scoops - 1;
}
document.write("Life without ice cream isn't the same");
```

Is scoops greater than zero? Looks like it to us!

WHILE loop

- **The while loop will repeat a statement (or set of statements) as long as the condition is met (true)**

```
while (expression) {
  // statement(s) to repeat
}

// example
var x = 0;
while (x < 10) {
  console.log(x);
  x++;
}
```

DO… WHILE loop

- The do… while loop will repeat a statement (or set of statements) as long as the condition is met (true)
  - similar to the while loop
  - It executes the statement(s) at least once!

```
do {
  // statement(s) to repeat
} while (expression)

// example
var x = 0;
do {
  console.log(x);
  x++;
} while (x < 10)
```

FOR loop

- a counter/index is created to track the progress of the loop (in the initialization part)
- the statement(s) to repeat are executed only if condition passes
- at the end of the loop body, the counter/index is updated to track progress

```
for (initialize; condition; update) {
  // statement(s) to repeat
}

// example
for (var i = 0; i < 10; i++) {
  console.log(i);
}

// the same as
var i = 0;
while (i < 10) {
  console.log(i);
  i++;
}
```

Resources

https://medium.freecodecamp.org/whats-the-difference-between-javascript-and-ecmascript-cba48c73a2b5

Very gentle introduction to JavaScript: http://jsforcats.com

Another (not so gentle) introduction to JS: https://eloquentjavascript.net