

JavaScript Introduction

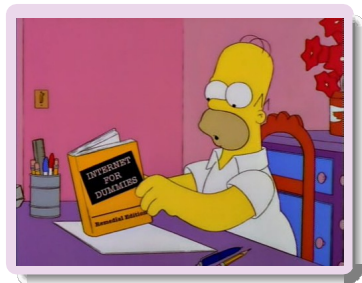
Topics discussed this presentation

- Brief introduction to and history of language
- Roles of the language
- Its data types
- JavaScript Object Notation (JSON)
- Simple program employing JavaScript

Javascript

Overview

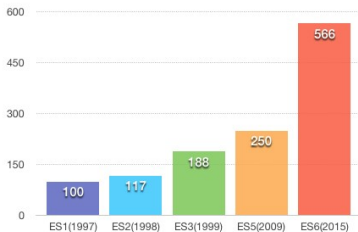
- Originally a small language
- Not anymore - now enormous
- Flawed but powerful
- Not **Java**
- Not a subset of Java
 - Very different languages
- Shares C-family syntax
- Similarities Scheme & Self
- Scores of *badly written books aimed at the dummies and amateur market*



Javascript

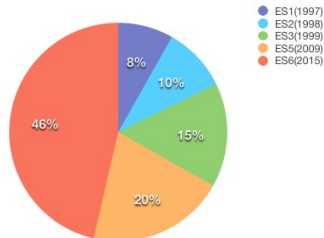
Language specification growth

ECMAScript, Growth in language complexity as measured by increase in successive specification versions.



Standard ECMA-262

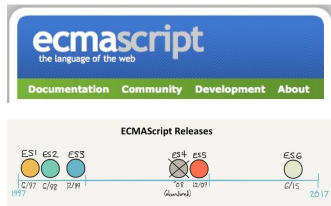
VERSION	SPECIFICATION PAGES
ES1(1997)	100
ES2(1998)	117
ES3(1999)	188
ES5(2009)	250
ES6(2015)	566



Ecma International

ECMAScript - the language of the web

- ECMAScript: standardization body
- Several popular implementations:
 - JavaScript
 - JScript
 - ActionScript
- Edition 6 (ES6) published June 2015
 - Course applies ES6



JavaScript

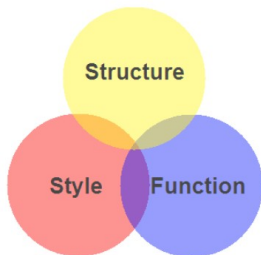
Several frameworks available

- Client-side
 - Angular
 - Backbone
 - Ember
 - Aurelia
 - React
- Server-side (node)
 - hapi
 - express
 - koa
 - sails
- MEAN stack collection:
 - MongoDB
 - Express.js
 - Angular
 - Node.js



Nature of JavaScript

Structure Client-Side Web



- Markup (HTML)
 - Structure
 - Content
- Style (CSS)
 - Style
 - Presentation
 - Appearance
- Function (Javascript)
 - Actions
 - Manipulations

Nature of JavaScript

The Language

Although not Java, has:

- Similar syntax & keywords
 - Similar standard library naming conventions

Object oriented but does not have classes in classical sense.

- uses *syntactic sugar* to simulate classes
- prototypal: objects inherit from objects

Dynamic typing

- Variable may be reference to object of any type

Javascript Styling

Our choice from several available Style Guides and IDEs

Airbnb JavaScript Style Guide() {

A mostly reasonable approach to JavaScript

downloads 649k/month downloads 563k/month glitter join chat

Other Style Guides

- ES5 (Deprecated)
- React
- CSS-in-JavaScript
- CSS & Sass
- Ruby

Table of Contents

1. Types
2. References
3. Objects
4. Arrays
5. Destructuring
6. Strings
7. Functions
8. Arrow Functions
9. Classes & Constructors
10. Modules
11. Iterators and Generators
12. Properties
13. Variables

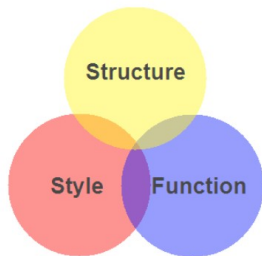


- *Airbnb JavaScript Style Guide*
- *WebStorm JavaScript IDE*

Nature of JavaScript

The Language

- Provides access to main components web page:
 - Cascade Style Sheet (CSS) properties
 - Markup content (e.g.: div, img, p)
 - Forms (Communication to server)
- Most often used client-side
- Growing use server-side (node.js)
- Weakly typed with first-class functions
 - function: block reusable code (more on this later)
 - functions are objects
 - may be passed as parameters



Javascript

Primitive Data Types

- Six primitive types
 - **boolean**
 - **number**
 - **string**
 - **null**
 - **undefined**
 - **symbol** (ES6)
- All other types are **objects**

```
console.log('This is a string');  
console.log(true is a boolean);  
console.log(10.5 is a number);
```

Javascript

Primitive Wrapper Data Types

- Four wrapper types
 - **Boolean**
 - **Number**
 - **String**
 - **Symbol**

```
// Wrapper's valueOf returns primitive value.  
const b = Boolean(true); // b => true.
```

var, const and let

var, **const** and **let** used to store values and object references:

- **var** exists since ES1.
- **const** & **let** introduced ES6.
- Significant behavioural differences.
- Preference given henceforth to use of **const**, then **let**.
- **var** usage should be avoided.

```
var x = 10; // Avoid future use
let y = 20; // Use where reassign likely
const z = 30; // cannot be reassigned
```

Data types

boolean

- boolean can be
 - true
 - false

```
// Output: b is true
const b = true;
if (b) {
  console.log('b is true');
};
```

```
// Output: b is true
const b = true;
if (b) {
  console.log('b is ', Boolean(b));
};
```

Data types

number

- *number* 64-bit floating point
 - Similar to Java's *double*
 - No integer type
 - *number* type includes
 - NaN
 - Infinity
 - Problematic in finance
 - $0.1 + 0.2 = 0.3$
 - This expression *false*

```
// Output is 3.3333333333333335
const val = 10 / 3;
console.log(val);
```

```
// Output: true. val is not a number
const val = '2005/12/12';
console.log(isNaN(val)); // true
// Output: string
console.log(typeof val);
```

```
const val = 10 / 0;
console.log(val); // Infinity
console.log(typeof val); // number
```

Data types

string

- *string* sequence of zero or more Unicode characters.
 - Similar to Java *String*.
 - No *char* type as in Java.
 - Literals use ' or " to enclose characters
 - Either quote type may be used in pairs.
 - Illegal to mix.
 - **Important:** Use only single quotes to comply with style guide.

'This is a string.'

"This is also a string but we will not use double quotes."

"This is not a legal string'

Data types

string

- Internal quotes
- Can use escape sequence \

```
const s = 'What\'s a \"celeb\" famous for?';  
  
// What's a "celeb" famous for?  
console.log(s);
```


Data types

null & undefined

- Variable not assigned a value is of type **undefined**
- **null** indicates the absence of a value
- Some experienced developers no longer use *null*.

```
var planes; // => undefined
// A language error in ES5, fixed ES6
console.log(typeof planes); // => object in ES5
```

```
const planes;
// SyntaxError: Missing initializer in const declaration
```



Data types

symbol

Associated wrapper class **Symbol**

- Introduced in ES6
- Can generate unique property keys
- Eliminates risk collision

```
let uniqueKey = Symbol();
obj = {};
obj[uniqueKey] = 'unique';
console.log(obj[uniqueKey]); // => unique
console.log(uniqueKey); // => Symbol()
console.log(typeof(uniqueKey)); // => symbol
```

Data types

Object

Object literal

- comma-separated list of colon-separated name:value pairs in curly braces.

```
const book = {  
  title: 'Java',  
  author: 'Chapman',  
  ISBN: 'ISBN-10 03219804333',  
  edition: 4,  
  isInPrint: true,  
};
```

```
book.isInPrint // => true
```

Data types

Object

Container comprising

- name-value pairs
- value may be object
- may add new properties anytime

```
▼ Watch Expressions + ↻
▼ book: Object
  ▼ author: Object
    name: "Simpson"
    ► __proto__: Object
  isbn: "ISBN-10 03219804333"
  title: "Java"
  ► __proto__: Object
```

```
const book = {
  title: 'Java',
  author: {
    name: 'Simpson',
  }
};
console.log(book.title); // => Java
```

```
// Add new property (name-value pair)
book.isbn = 'ISBN-10 03219804333';
```

Semicolon insertion

Example where positioning of curly brace matters

```
function myFunction() {  
  return  
  {  
    status: true  
  };  
};
```

Semi-colon silently inserted following **return** keyword has unintended consequences: returned value is **undefined**.

```
console.log(myFunction()); // undefined
```

Semicolon insertion

Example where positioning of curly brace matters

```
function myFunction() {  
  return {  
    status: true  
  };  
};
```

K&R style, put the { at the end of a line instead of the front, because *it avoids a horrible design blunder in JavaScript's return statement. (Crockford)*

```
console.log(myFunction()); // Object{status:true}
```

JavaScript

Run Program - Simple Example

```
/**
 * A Web Page with HTML & reference to external JavaScript file
 */
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1 id="hello">Hello ICTSkills</h1>
    <script src="js/foo.js"></script>
  </body>
</html>
```

JavaScript

Run Program - Simple Example

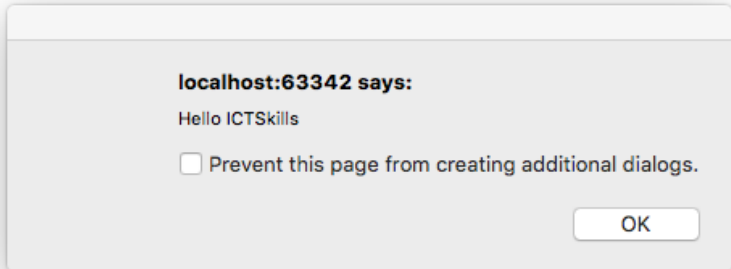
```
/**
 * Demo JavaScript code
 */
alert('Hello ICTSkills');
function foo() {
  const size = 3;
  for (let i = 0; i < size; i += 1) {
    console.log(i);
  }
}

foo();
```


JavaScript

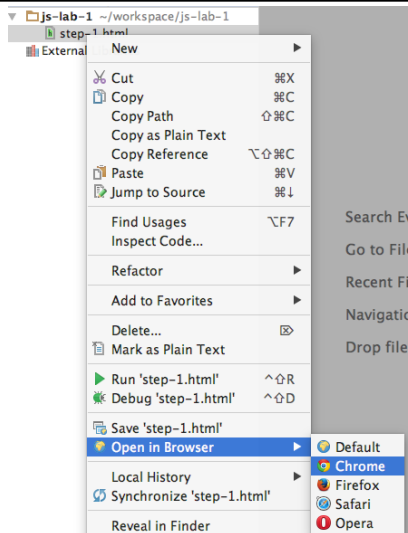
Run Program - Simple Example

Hello ICTSkills



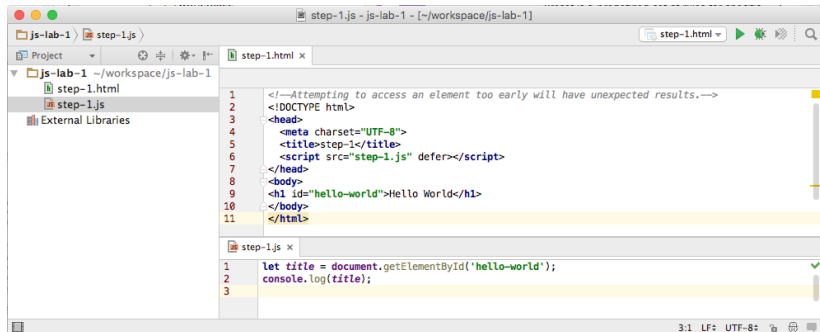
JavaScript

Run Program - Simple Example



JavaScript

Run Program - Simple Example



```
step-1.html x
1 <!--Attempting to access an element too early will have unexpected results.-->
2 <!DOCTYPE html>
3 <head>
4   <meta charset="UTF-8">
5   <title>step-1</title>
6   <script src="step-1.js" defer></script>
7 </head>
8 <body>
9   <h1 id="hello-world">Hello World</h1>
10 </body>
11 </html>

step-1.js x
1 let title = document.getElementById('hello-world');
2 console.log(title);
3
```

3:1 LF UTF-8