Internet Computing
JavaScript

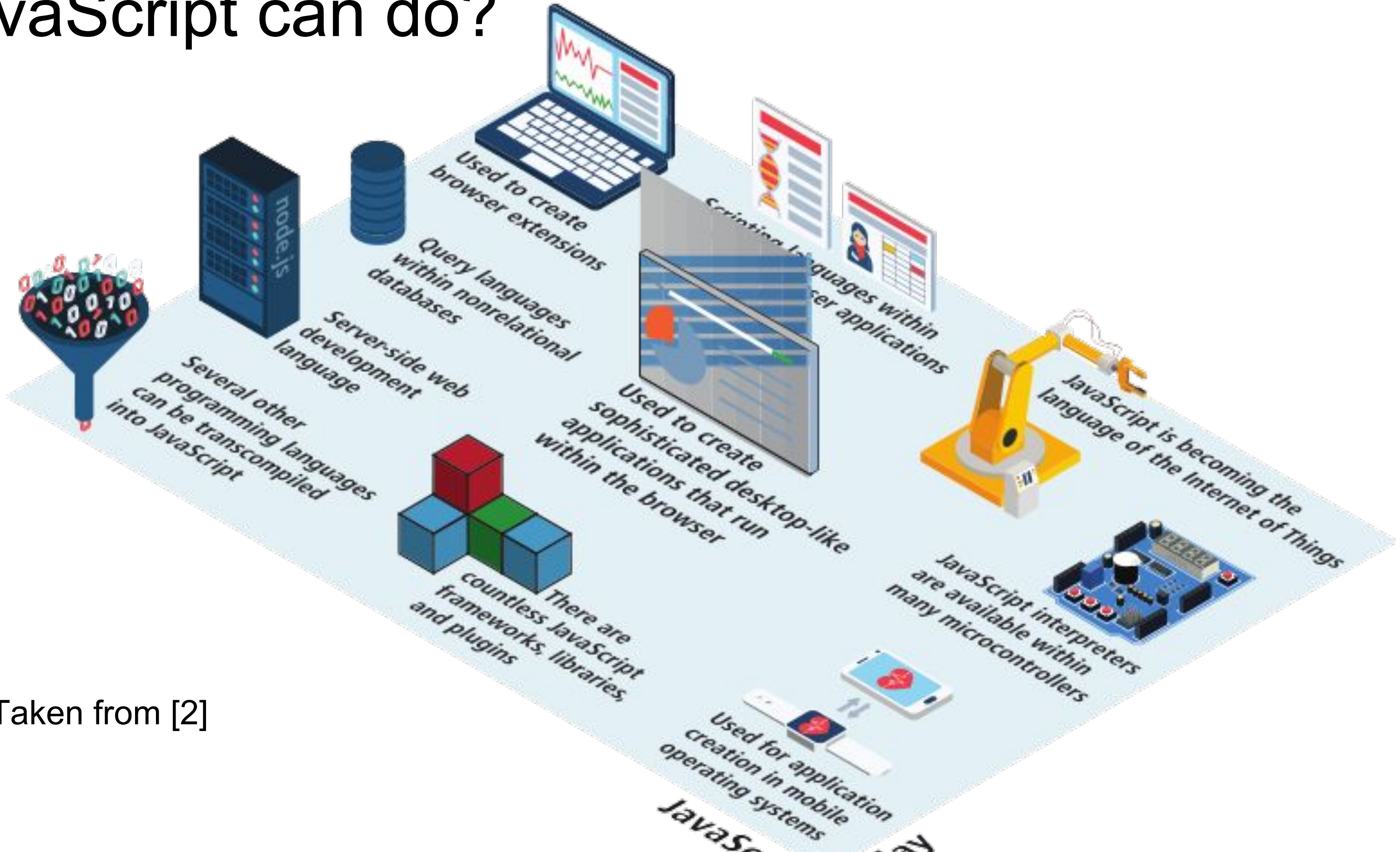
What is JavaScript

- Originally developed at Netscape by Brendan Eich:
  - original prototype created in 10 days
  - Named Mocha at first, then was renamed to livescript
- In 1995, later the branch joined with SUN microsystems
  - renamed it to JavaScript
- JavaScript versions
  - A language standard was developed in the latte 1990 by ECMA (European Computer Manufactures Association
    - ECMA-262 (https://www.ecma-international.org/publications/standards/Ecma-262.htm)
  - https://www.w3schools.com/js/js_versions.asp
  - Most browsers implement languages that conform to ECMA-262
  - Latest version of ECMAScript is the sixth edition

What is JavaScript

- Three Categories of JavaScript
- Core
  - Operators, expressions, statements
- Client-side
  - Supports the control of a browser and interactions with users
  - Code runs in browser after page is sent back from server.
- Server-side
  - Support communication with a DBMS

What JavaScript can do?


node.js
Used to create browser extensions
Query languages within nonrelational databases
Server-side web development language
Several other programming languages can be transcompiled into JavaScript
Scripting languages within
applications
Used to create sophisticated desktop-like applications that run within the browser
JavaScript is becoming the language of the Internet of Things
JavaScript interpreters are available within many microcontrollers
There are countless JavaScript frameworks, libraries and plugins
Used for application creation in mobile operating systems

Taken from [2]

How to link JavaScript and HTML

- Inline
  - Including JavaScript code directly within an HTML element

```
<input type="button" onClick="alert('Are you sure?');" />
```

- Explicit Embedding
  - JavaScript is placed within <script> element
  - poor code quality though
    - separate content, presentation, and behavior

```
<script type="text/javascript">
     /* A JavaScript Comment */
     alert("Hello World!");
</script>
```

- Implicit embedding/external JavaScript
  - In a separate file
  - Referenced from within head

```
<head>
<script type="text/javascript"
src="greeting.js"></script>
</head>
```

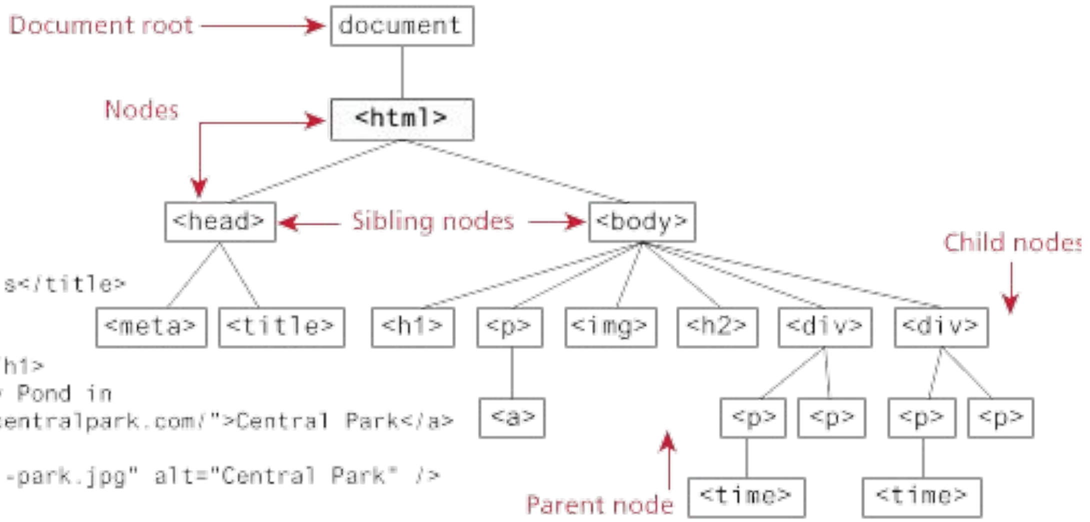
JavaScript vs Java

- Differences
    - Java is a strongly typed language while JavaScript is dynamically typed
        - JavaScript: `var a;`
    - Objects in java are static while they are dynamic in JavaScript
        - Static: the collection of data members and methods is fixed at compile time
        - Dynamic: the number of data members and methods of an object can change during execution
- Similarities: Syntax of expressions, assignment statements, and control statements

JavaScript Uses

- Transfer of load from server to client
  - Benefits other clients
  - User interactions through forms are easy
    - Provide feedback to the user through mouse events
    - Generate new content for an HTML element
    - Generate a new element
- The Document Object Model makes it possible to support dynamic HTML documents with JavaScript
  - Access and modify the style properties and content of the elements
- Much of what we will do with JavaScript is event-driven computation
  - Code are executed in response to user’s action

The Document Object Model (DOM)



```
<html>
<head>
   <meta charset="utf-8">
   <title>Share Your Travels</title>
</head>
<body>
   <h1>Share Your Travels</h1>
   <p>Photo of Conservatory Pond in
      <a href="http://www.centralpark.com/">Central Park</a>
   </p>
   <img src="images/central-park.jpg" alt="Central Park" />
   <h2>Reviews</h2>
   <div id="latestComment">
      <p>By Ricardo on <time>2016-05-23</time</p>
      <p>Easy on the HDR buddy.</p>
   </div>
   <div>
      <p>By Susan on <time>2016-11-18</time></p>
      <p>I love Central Park.</p>
   </div>
</body>
</html>
```

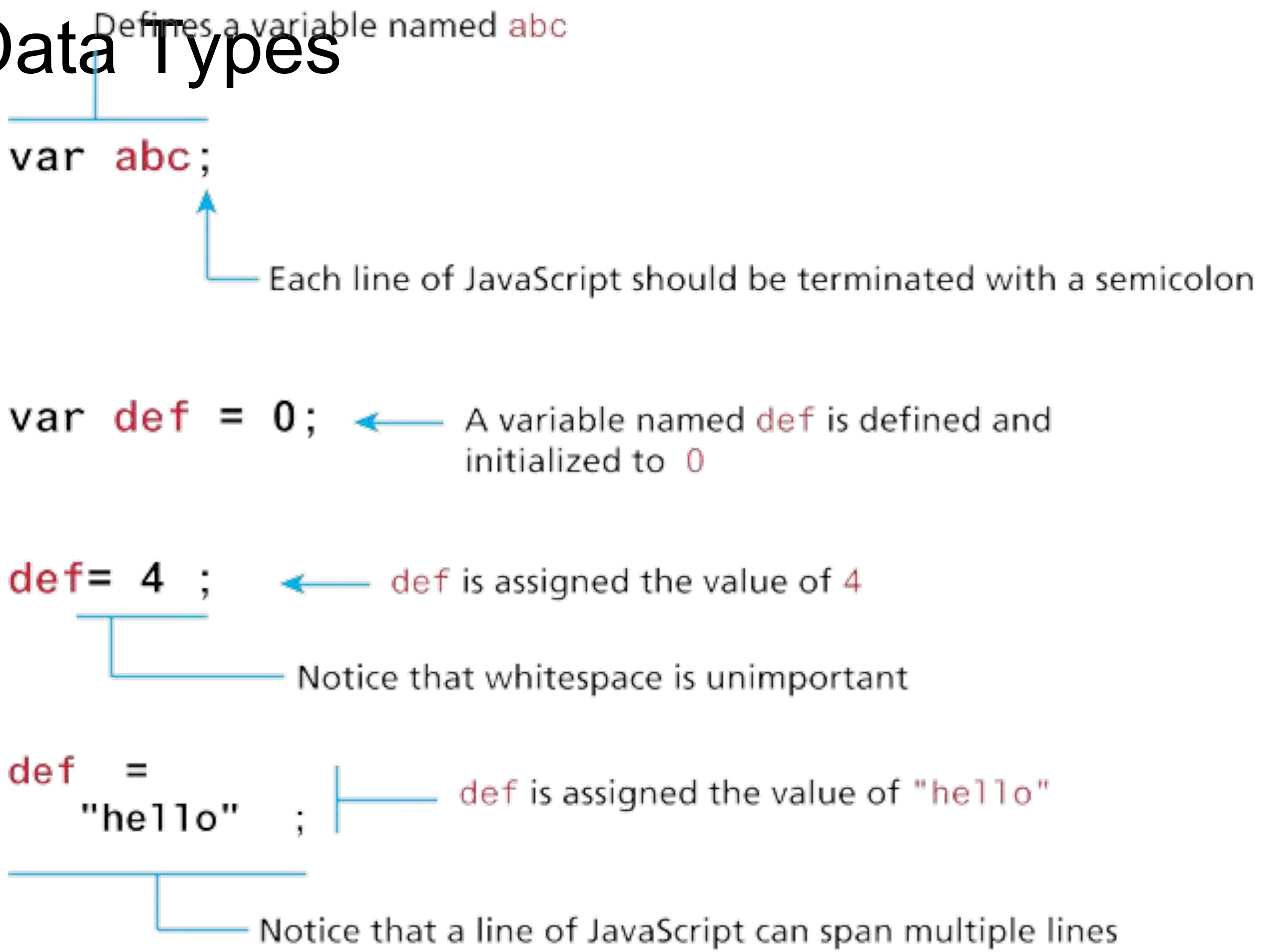
General syntax

- Language Basics:
  - Identifier form: begin with a letter or underscore, or a $ sign, followed by any number of letters, underscores, and digits
  - No length limitations
  - Case sensitive
- reserved words
  - https://www.w3schools.com/js/js_reserved.asp
- Comments: both // and /* … */

Variables and Data Types

- Variables in JavaScript are dynamically typed
  - simply use the var keyword to declare a variable
  - Or just assign a value to a variable name

Variables and Data Types



Defines a variable named abc

```
var abc;
```

Each line of JavaScript should be terminated with a semicolon

```
var def = 0;
```

A variable named def is defined and initialized to 0

```
def= 4  ;
```

def is assigned the value of 4

Notice that whitespace is unimportant

```
def   =
    "hello"   ;
```

def is assigned the value of "hello"

Notice that a line of JavaScript can span multiple lines

Taken from [2]

Variables and Data Types

- Two basic data types:
  - reference types
    - usually referred to as objects)
  - primitive types
    - Represent simple forms of data
    - Number, String, Boolean, Undefined, or Null
      - Number, String, and Boolean have wrapper objects caleed `Number, String, and Boolean`
        - In the cases of Number and String, primitive values and objects are coerced back and forth so that primitive values can be treated essentially as if they were objects

```
Var price = 427, str_price;
Str_price = price.toString();
```

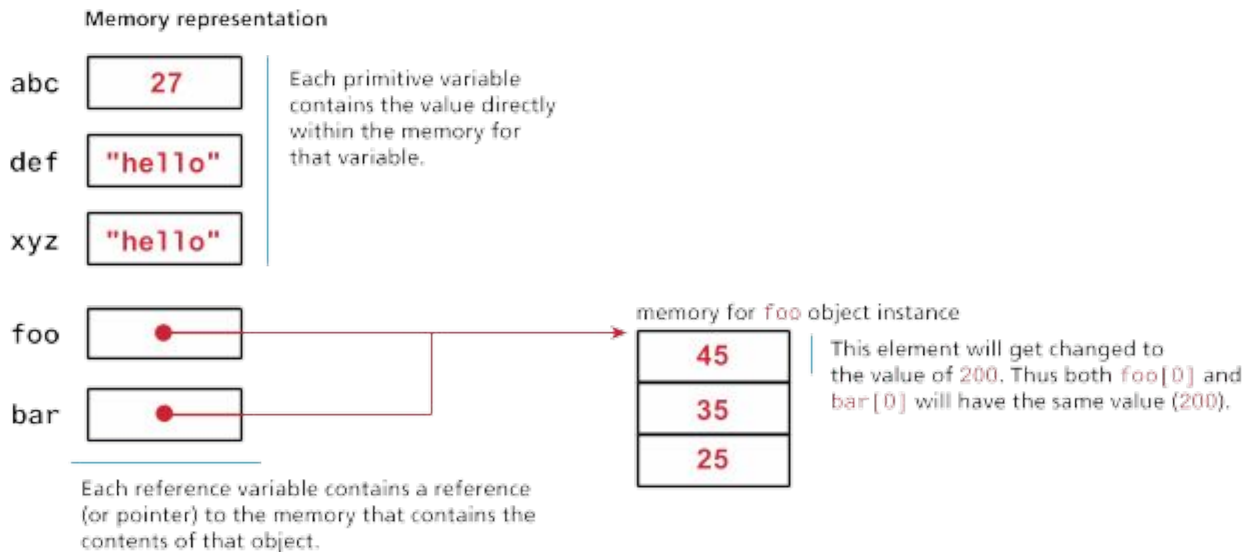
Variables and Data Tyes

```
var abc = 27;
var def = "hello";
```
variables with primitive types

```
var foo = [45, 35, 25];
```
variable with reference type (i.e., array object)

```
var xyz = def;
var bar = foo;
```
these new variables differ in important ways (see below)

```
bar[0] = 200;
```
changes value of the first element of array



Each primitive variable contains the value directly within the memory for that variable.

This element will get changed to the value of 200. Thus both foo[0] and bar[0] will have the same value (200).

Each reference variable contains a reference (or pointer) to the memory that contains the contents of that object.

Taken from [2]

Type conversion

- Implicit type conversions in JavaScript: coercion
  - Catenation coerces numbers to strings
    - if either operand of + is a string, it is assumed to be concatenation)
  - Numeric operators (other than +) coerce strings to numbers
    - Conversions from strings to numbers that do not work return NaN
  - In the cases of Number and String, primitive values and objects are coerced back and forth, so that primitive values can be treated essentially as if they were objects

“August” + 1997

7 * “3”
7* “August”

```
Var price = 427, str_price;
Str_price = price.toString();
```

- Explicit type conversion
  - `Var number = Number (astring)`

JavaScript Output

- The JavaScript model for the HTML document is the Document Object Model (DOM)
- The model for the browser display window is the Window object
  - The Window object has two properties, document and window, which refer to the Document and Window objects, respectively
- The Document object has a method, write, which dynamically creates content
  - The parameter is a string, often catenated from parts, some of which are variables
  - `document.write("Answer: " + result +  "<br />");`
  - The parameter is sent to the browser, so it can be anything that can appear in an HTML document (<br />, but not \n)
- The Window object has three methods for creating dialog boxes
  - alert, confirm, and prompt

JavaScript Output

- alert("Hej! \n");
  - Parameter is plain text, not HTML
  - Opens a dialog box which displays the parameter string and an OK button
  - It waits for the user to press the OK button
- confirm("Do you want to continue?");
  - Opens a dialog box and displays the parameter and two buttons, OK and Cancel
  - Returns a Boolean value, depending on which button was pressed (it waits for one)
- prompt("What is your name?", "");
  - Opens a dialog box and displays its string parameter, along with a text box and two buttons, OK and Cancel
  - The second parameter is for a default response,if the user presses OK without typing a response in the text box (waits for OK)
- console.log(“Hello World”);
  - Appears in the browser console (use chrome developer tools to see browser console)

Control Statements

- Similar to C, Java, and C++
  - The variables declared within a block are not local to the block
- Control Expression
  - Primitive values
    - If it is a string, it is true unless it is empty (“) or zero ("0")
    - If it is a number, it is true unless it is zero
    - NAN, undefined, null, "", '' are false when interpreted as boolean
  - Relational Expressions
    - The usual six: ==, !=, <, >, <=, >=
      - Operands are coerced if two operands are not of the same type
        - If one is a string and one is a number, it attempts to convert the string to a number
        - If one is Boolean and the other is not, the Boolean operand is coerced to a number (1 or 0)
  - The unusual two: === and !==
    - Same as == and !=, except that no coercions are done (operands must be identical)
  - Comparisons of references to objects are not useful (addresses are compared, not values)

```
“3” === 3   false
“3” == 3      true
```

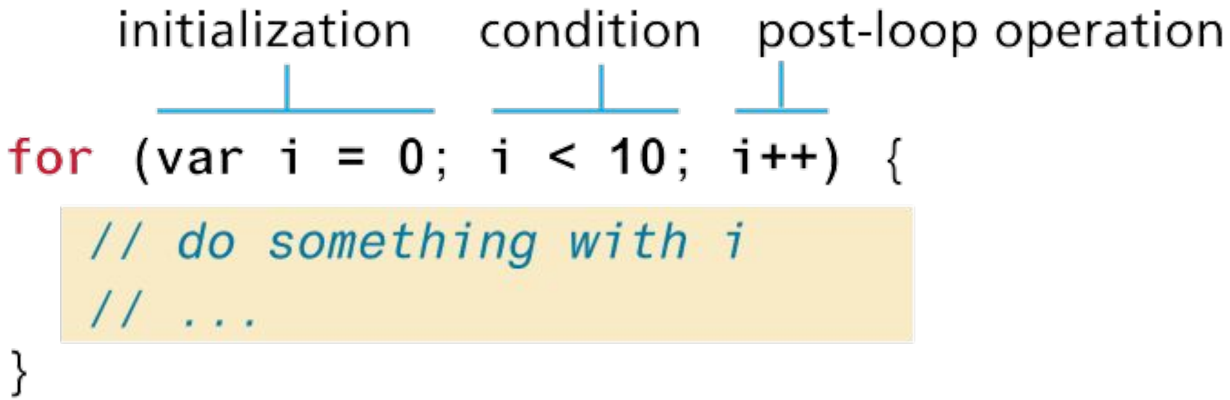
Control Statements

- The selection statements are similar to
  - If-then and if-then-else
  - Switch statement:
    - The control expression can be a number, a string, or a Boolean
    - Different cases can have values of different types

```
switch (expression) {
    case value_1:
      // value_1 statements
    case value_2:
      // value_2 statements
    …
    [default:
      // default statements]
  }
```

Loops

```
var count = 0;
while (count < 10) {
        // do something
        // ...
        count++;
}
count = 0;
do {
        // do something
        // ...
        count++;
} while (count < 10);
```



```
for (var i = 0; i < 10; i++) {
    // do something with i
    // ...
}
```

Object Orientation and JavaScript

- JavaScript is NOT an object-oriented programming language
  - No support for class-based inheritance or polymorphism
  - Has prototype-based inheritance
    - Simulate inheritance with the prototype object
- JavaScript objects are collections of properties,
- which are like the members of classes in Java and C++
- Its objects serve both as objects and as models of objects (classes)
- The root object in JavaScript is Object – all objects are derived from Object
- All JavaScript objects are accessed through references
- JavaScript's key construct is the function rather than the object/class.
  - "first-class" functions are used in many situations

Object Creation

- Object Literal Notation
- Using constructor (new keyword)

Var my_car = {make: "Ford", model: "Fusion”}

```
var my_car = new Object(); //create an empty object
my_car.make = "Ford";   //create and initialize properties
my_car.model = "Fusion";
var property1 = my_car["model"];
delete my_car.model;
for (var prop in my_car) {
}
```

- Properties can be accessed by **dot notation** or in **array notation,** as in

References

1. Programming the World Wide Web, 8th edition
2. Fundamentals of Web Development, 2nd edition
3. https://www.w3schools.com/