Intro to JavaScript

COEN 161

Web Pages vs Web Applications

- A web page is informational
  - Web pages are “static” HTML/CSS
  - Hyperlinks take you to other pages with more information (website)
  - The information is still controlled by the developer

- A web application is *interactive*
  - User can control the page
  - Inputs generate unique outputs
  - The experience can be customized for each user

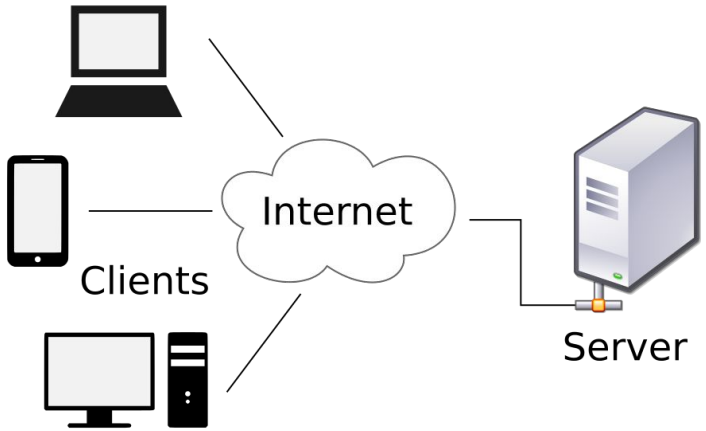
Making a Page Interactive

- First, you need a programming language
  - This language runs in the browser (client side)

- Second, you need a way to access the components on the page
  - The language provides an API that models the HTML document

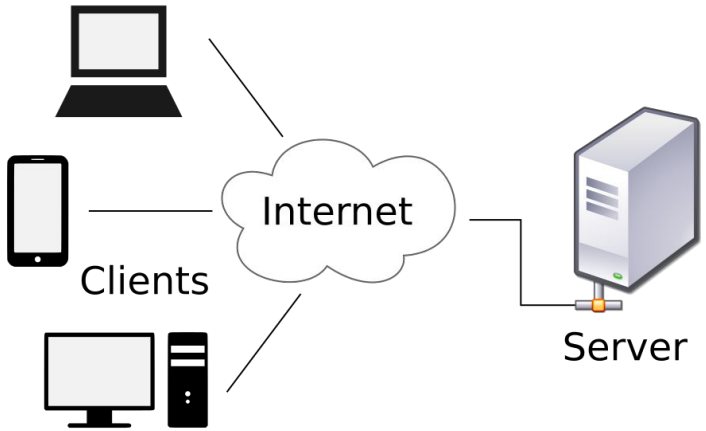
Client-Side Scripting

- In the Web, the client is almost always a browser (desktop or mobile)
- The browser supports a scripting language (JavaScript!)
- The script runs on the user's computer when the page loads (security risk)



Client-Side Scripting

- Client-side scripts are quicker to respond than server-side scripts
- This improves the experience
- Can’t replace the server-side scripts entirely



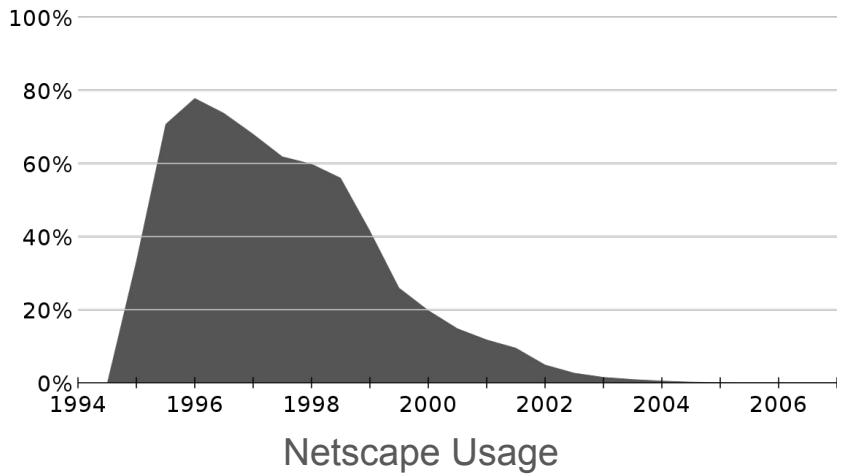
Why JavaScript?

- JavaScript is the default language for all modern web browsers
- It was designed specifically for the web
- JavaScript is one of the most popular programming languages
- Huge developer community supporting and creating additional features

Rank	Language	Share	Trend	Rank	Language	Share	Trend
1	Java	21.3 %	-0.8 %	12	Ruby	1.7 %	-0.3 %
2	Python	19.6 %	+5.2 %	13	VBA	1.4 %	+0.2 %
3	PHP	8.0 %	-1.8 %	14	TypeScript	1.3 %	+0.4 %
4	Javascript	7.9 %	+0.4 %	15	Visual Basic	1.2 %	-0.3 %
5	C#	7.6 %	-0.5 %	16	Scala	1.2 %	+0.1 %
6	C++	6.2 %	-1.0 %	17	Kotlin	0.8 %	+1.0 %
7	C	6.2 %	-1.0 %	18	Perl	0.7 %	-0.1 %
8	R	3.8 %	+0.3 %	19	Go	0.7 %	+0.2 %
9	Objective-C	3.7 %	-1.0 %	20	lua	0.4 %	-0.2 %
10	Swift	3.0 %	-0.2 %	21	Haskell	0.3 %	-0.0 %
11	Matlab	2.2 %	-0.5 %	22	Rust	0.3 %	-0.1 %

History of JavaScript

- Created at Netscape Communications Corp. in 1995
  - Netscape Navigator was one of the most popular browsers in the 90s
  - Developed by Brendan Eich ’83 in 6 months
  - Rushed due to pressure from Sun Microsystems wanted to put Java in the browser
  - This rush resulted in many of the “quirks” that we still deal with today



History of JavaScript

- The goal was to make the web more dynamic
  - Needed a scripting language that was as easy to pick up as HTML
  - Aimed at non-programmers such as
  - Ideally it would be a *dynamic language*
- Dynamic Languages follow these general ideas
  - Faster to write
  - Slower to run (tradeoff)
  - Difficult to debug
- Java was too big to fit these needs
  - Not dynamic
  - It was considered a big “enterprise” language
  - Needed a companion language that was more light-weight
  - This language was developed under the name *Mocha*

JavaScript vs Java

- Java
  - Like C/C++ is a *compiled language*
    - Compiler “checks” the code and decides whether or not it’s ok to run
  - It is *strongly typed*
    - Variable types need to match in order for statements to be valid
    - i.e. integers get assigned to integers, characters to characters, etc.
  - Errors are easily detectable
    - They can be picked up during compilation
    - Exceptions can be thrown during runtime

JavaScript vs Java

- JavaScript
  - Is an *interpreted* language
    - Each line of code is processed one by one during runtime
    - This eliminates the step of compiling before getting to run the code
  - It is *loosely typed*
    - There are some types in JavaScript but they are not enforced
    - All variables are considered the same: *var*
    - They can be assigned to one another regardless of type
  - JavaScript can fail “silently”
    - No clear error messages
    - Exceptions may not be descriptive

JavaScript vs Java

- Both are object-oriented
  - The idea is to represent *things* like shapes, cars, etc.
    - You can also represent abstract things like files, customers, etc.
  - Object-orientation is based on *classes* and *objects*
  - Classes define the data that the class contains and any methods or functions that do work on that data
  - Java is *class-based*
    - Classes define new custom types
    - Classes contain properties about the type defined as variables
    - Classes also contain behaviors related to that type defined as methods/functions
  - In Java, *objects* are instances of a class
    - The keyword *new* is used to *instantiate* a class
    - It creates an *object* of the type *class*

JavaScript vs Java

- JavaScript is also object-oriented, but it is *prototype-based*
  - There are no classes, just objects
  - A special object, called the *prototype*, defines the methods and properties of all the instance objects
  - All the instance objects have **one** link to a prototype

What can JavaScript do?

- Since JavaScript is closely tied to the HTML document, it can…
  - Change HTML content (example)
  - Change HTML attributes (example)
  - Change HTML styles (example)
  - Hide HTML content (example)
  - Show HTML content (example)

How do you write JavaScript?

- JavaScript is inserted using the <script> tag
- The <script> tag can be inserted in either the <head> or the <body>
- JavaScript can be written straight in the <script> tag

```
<script>
    document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

- You can reference external JavaScript using the `src` attribute

```
<script src="myScript.js"></script>
<script src="https://www.w3schools.com/js/myScript1.js"></script>
```

JavaScript Syntax

- JavaScript is *C-style syntax* like C/C++ or even Java

```
var x, y;           // How to declare variables

x = 5; y = 6;       // How to assign values

z = x + y;          // How to compute values
```

- There are two types of values that can be declared in JS
    - Fixed values - which are called **literals**
    - Variable values - which are called **variables**

JavaScript Comments

- JS uses C-style comments
- Code on the same line after a double slash // is treated as a comment
- Code in between /* and */ is treated as a block comment

```
var x = 5;   // I will be executed

// var x = 6;   I will NOT be executed
```

JavaScript Literals

- Numbers can be written with or without decimals

```
10.50
1001
```

- Strings can be written with *single-quotes* or *double-quotes*

```
"John Doe"
'John Doe'
```

JavaScript Variables

- In JS variables are used to store data values
- The `var` keyword is used to *declare* variables
- The equal sign `=` is used to *assign* values to variables

```
var x;

x = 6;
```

JavaScript Operators

- JS uses arithmetic operators to compute values (+ - * /)

```
(5 + 6) * 10
```

JavaScript Operators

- Other supported operators include

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

JavaScript Expressions

- A JS expression is any combination of values, variables, and operators, which computes to a value
- The computation of that value is called an evaluation

```
5 * 10 // 50

x * 10 // you can use variables

"John" + " " + "Doe"  // "John Doe"
```

Other Features

- JavaScript is *case-sensitive*

```
var lastname, lastName;

lastName = "Doe";

lastname = "Peterson";
```

Other Features

- Programmers use different ways to join words into a variable name
  - Hyphens - **these are not allowed in JS since - means subtraction**

```
first-name, last-name, master-card, inter-city
```

  - Underscore

```
first_name, last_name, master_card, inter_city
```

  - Upper Camel Case (Pascal Case) - usually used for class names

```
FirstName, LastName, MasterCard, InterCity
```

  - Lower Camel Case - JS programmers tend to use this syntax for variable names

```
firstName, lastName, masterCard, interCity
```

Resources

https://auth0.com/blog/a-brief-history-of-javascript/

https://medium.com/@benastontweet/lesson-1a-the-history-of-javascript-8c1ce3bffb17

http://pypl.github.io/PYPL.html

https://www.w3schools.com/js/js_syntax.asp