JavaScript Basics

Building Modern Web Applications - VSP2019

Karthik Pattabiraman
Kumseok Jung

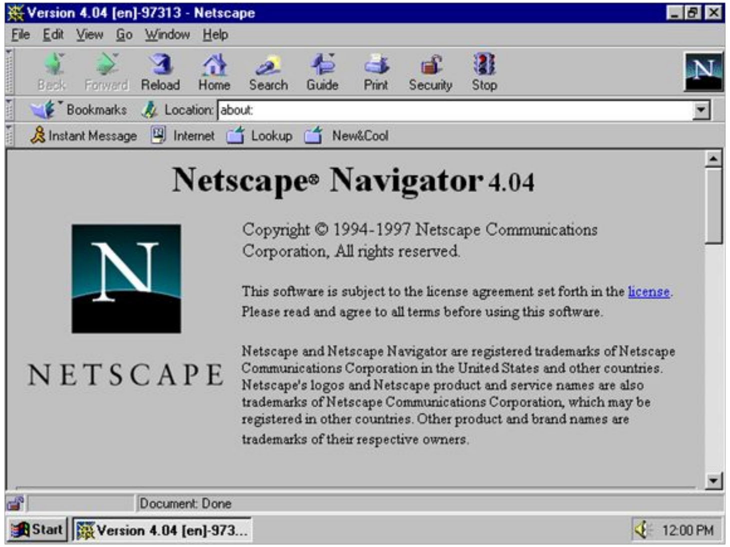
Introduction to JavaScript

1. **Introduction to JavaScript**
2. Data Types
3. Statements and Expressions
4. Class Activity

History: Browser War I (1995 ~ 2000)

- Web browser market was getting hot
    - Netscape Navigator vs Internet Explorer
    - Netscape had more features and was more robust
    - IE had the marketing power of Microsoft and was bundled with Windows as a free download !
    - JavaScript 1.0 was developed and integrated with Netscape by Brendan Eich (more on this later)
    - In 1997, Microsoft released IE4 which had feature parity with Netscape and integrated with Windows
        - Netscape was never the same again. They were sold to AOL and essentially lost the browser wars by the early 2000s
        - This started a first era of “browser monoculture”

History: Netscape Navigator vs Internet Explorer


Version 4.04 [en]-97313 - Netscape
File Edit View Go Window Help
Back Forward Reload Home Search Guide Print Security Stop
Bookmarks Location: about:
Instant Message Internet Lookup New&Cool
Document: Done
Start Version 4.04 [en]-973...
12:00 PM

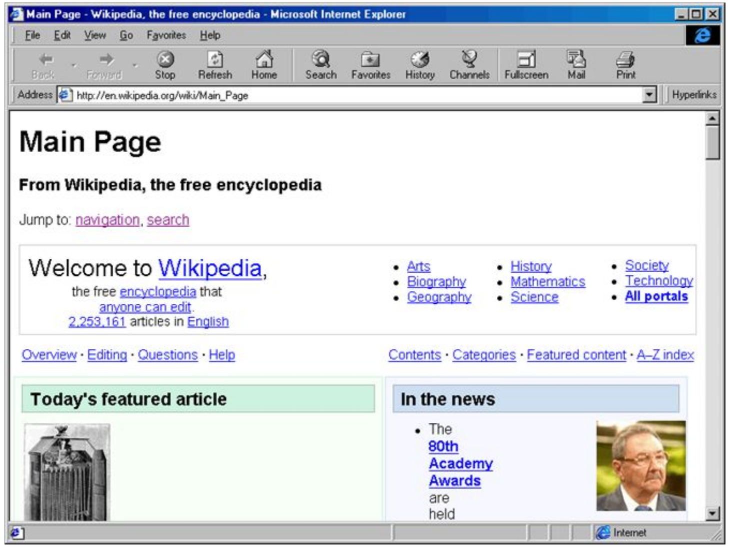
Netscape® Navigator 4.04

NETSCAPE

Copyright © 1994-1997 Netscape Communications Corporation, All rights reserved.

This software is subject to the license agreement set forth in the license. Please read and agree to all terms before using this software.

Netscape and Netscape Navigator are registered trademarks of Netscape Communications Corporation in the United States and other countries. Netscape's logos and Netscape product and service names are also trademarks of Netscape Communications Corporation, which may be registered in other countries. Other product and brand names are trademarks of their respective owners.


Main Page - Wikipedia, the free encyclopedia - Microsoft Internet Explorer
File Edit View Go Favorites Help
Back Forward Stop Refresh Home Search Favorites History Channels Fullscreen Mail Print
Address http://en.wikipedia.org/wiki/Main_Page
Hyperlinks
Internet

Main Page

From Wikipedia, the free encyclopedia

Jump to: navigation, search

Welcome to Wikipedia, the free encyclopedia that anyone can edit. 2,253,161 articles in English

- Arts
- Biography
- Geography
- History
- Mathematics
- Science
- Society
- Technology
- All portals

Overview · Editing · Questions · Help

Contents · Categories · Featured content · A–Z index

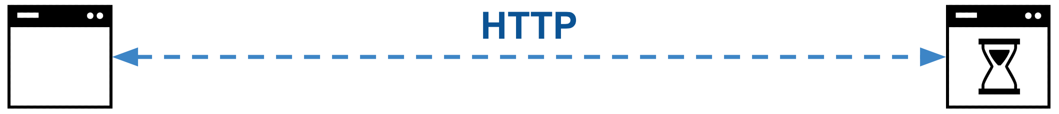
Today's featured article

In the news

- The 80th Academy Awards are held

History: Browser War I (1995 ~ 2000)

Web Ecosystem (~2000)



Web Browser

Examples:
WorldWideWeb (1990)
Mosaid (1993)
Netscape Navigator (1994)
Internet Explorer (1995)
Opera (1996)

Web Server

Examples:
CERN httpd (1990)
IIS (1995)
Apache (1995)

History: Browser War II (2000 ~ 2005)

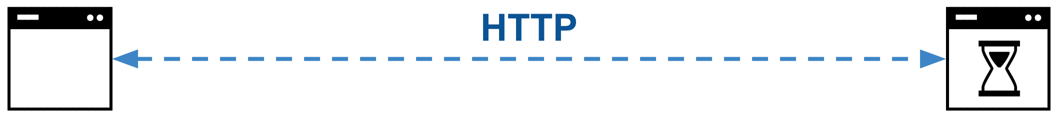
- But, while Netscape was finished, they released their code to the Mozilla foundation
  - Team of dedicated volunteers that rebuilt Navigator from scratch and ironed out its quirks
  - First release in 2004. Rapid releases in 2005,2006
  - First browser that was standards compatible
  - Microsoft became complacent. Removed most core staff from the IE team and didn’t develop it.
  - By 2006, Mozilla (Firefox) was back in the game

History: The Rise of AJAX (2005 ~ 2009)

- AJAX (**A**synchronous **Ja**vaScript and **X**ML)
  - Feature introduced by Microsoft IE in their Outlook Web Access (OWA) client in the early 2000s
  - Google used it for Google maps and Gmail (2004), and suddenly it was all the rage (2005)
  - JavaScript became the new popular kid on the block, and JavaScript performance started to become increasingly important with Mozilla taking the lead
  - Google introduced Chrome in 2008 which was primarily about faster JavaScript execution and support
    - ... which was itself based on WebKit, the engine powering the Safari browser (Mac)
    - ... which was itself based on KHTML and KJS, used in the Konqueror browser, from the KDE team (a Linux desktop environment)

History: Browser War II (2000 ~ 2005)

Web Ecosystem (2000~)



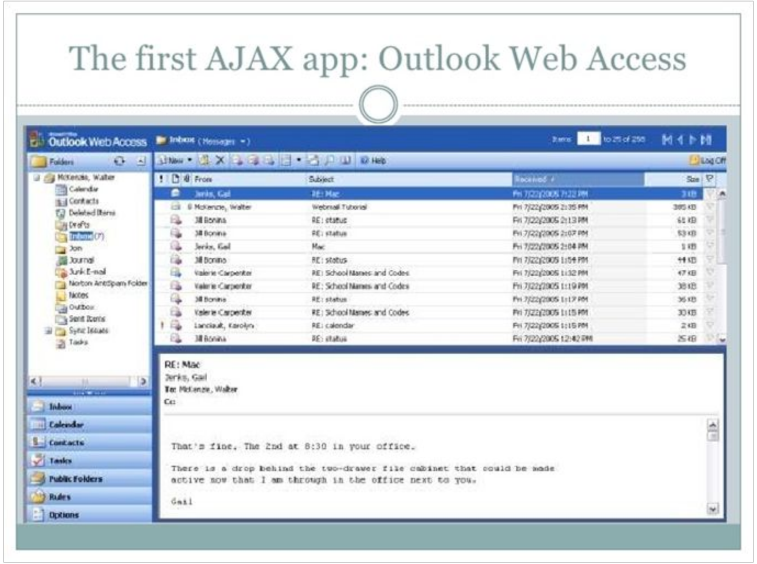
Web Browser

Examples:
WorldWideWeb (1990)
Mosaid (1993)
Netscape Navigator (1994)
Internet Explorer (1995)
Opera (1996)
Safari (2003)
Firefox (2004)
Chrome (2008)
Edge (2015)

Web Server

Examples:
CERN httpd (1990)
IIS (1995)
Apache (1995)
Nginx (2004)

History: The Rise of AJAX (2005 ~ 2009)



The first AJAX app: Outlook Web Access

Inbox (Messages) Items 1 to 25 of 258

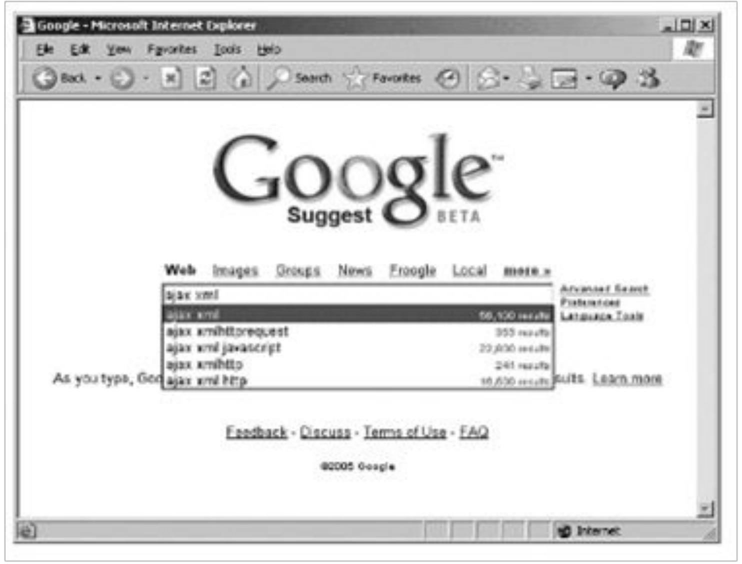
From	Subject	Received	Size
Jenks, Gail	RE: Mac	Fri 7/22/2005 7:22 PM	3 KB
McKenzie, Walter	Webmail Tutorial	Fri 7/22/2005 2:35 PM	385 KB
Jill Bonina	RE: status	Fri 7/22/2005 2:13 PM	61 KB
Jill Bonina	RE: status	Fri 7/22/2005 2:07 PM	53 KB
Jenks, Gail	Mac	Fri 7/22/2005 2:04 PM	1 KB
Jill Bonina	RE: status	Fri 7/22/2005 1:54 PM	44 KB
Valerie Carpenter	RE: School Names and Codes	Fri 7/22/2005 1:32 PM	47 KB
Valerie Carpenter	RE: School Names and Codes	Fri 7/22/2005 1:19 PM	38 KB
Jill Bonina	RE: status	Fri 7/22/2005 1:17 PM	36 KB
Valerie Carpenter	RE: School Names and Codes	Fri 7/22/2005 1:15 PM	30 KB
Lanciault, Karolyn	RE: calendar	Fri 7/22/2005 1:15 PM	2 KB
Jill Bonina	RE: status	Fri 7/22/2005 12:42 PM	25 KB

RE: Mac
Jenks, Gail
To: McKenzie, Walter
Cc:

```
That's fine. The 2nd at 8:30 in your office.

There is a drop behind the two-drawer file cabinet that could be made
active now that I am through in the office next to you.

Gail
```



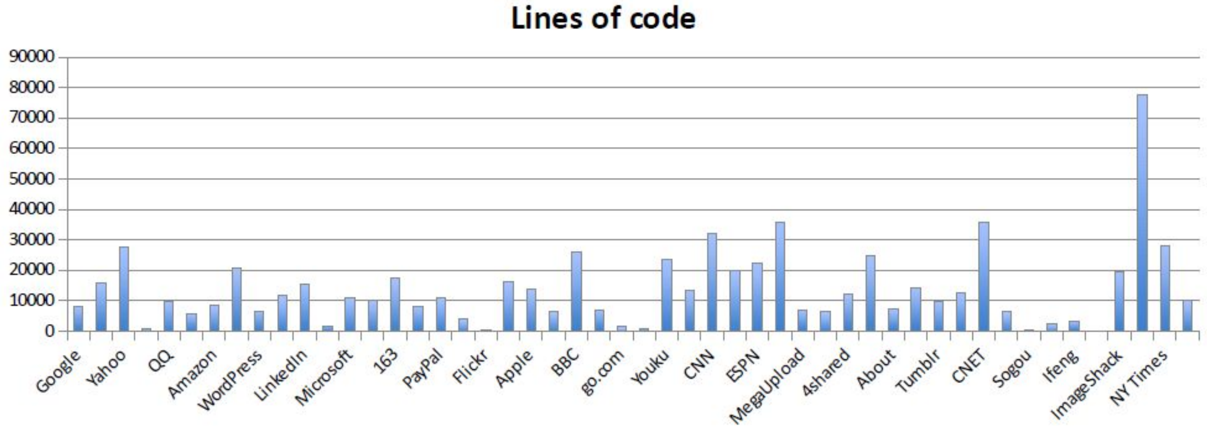
Google - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Google Suggest BETA
Web Images Groups News Froogle Local more »
ajax xml
ajax xml 59,100 results
ajax xmlhttprequest 353 results
ajax xml javascript 23,000 results
ajax xmlhttp 241 results
ajax xml http 10,600 results
Advanced Search
Preferences
Language Tools
As you type, Goo... ...sults. Learn more
Feedback - Discuss - Terms of Use - FAQ
©2005 Google
Internet

History: It’s a JavaScript World (2010 ~ 2015)

- JavaScript becomes mainstream
  - More and more websites heavily using JS, with thousands of lines of minified code
  - New applications (e.g., Google docs, Office live etc.) and frameworks (e.g., jQuery)
  - Academic papers are written about JavaScript in terms of its performance, reliability, security
  - JavaScript is the most popular language on Github, Stackoverflow and is in the top 5 on the Tiobe Index
  - JavaScript is also used to teach introductory CS by Khan Academy . often taught as the first language
  - Many variants of JavaScript: TypeScript (MS), DART (Google) and Flow (Facebook). Also, HTML5
  - Language for programming IoT devices (Samsung)

History: Prevalence of JavaScript (~2012)

- 97 of Alexa top 100 websites use JavaScript
- Many of them have thousands of lines of code



History: Browsers of Today

Desktop:

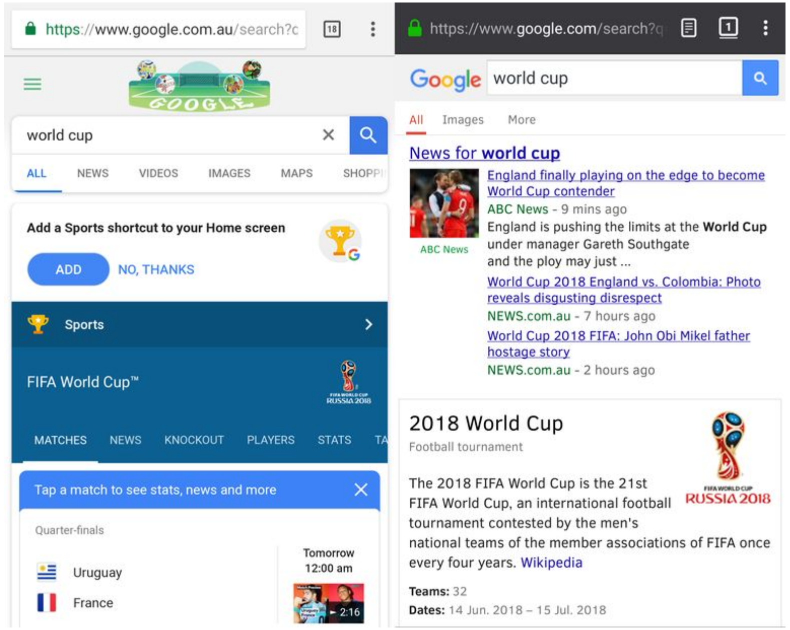
- Chrome: Blink (derived from WebKit)
- Safari: WebKit
- Opera: now derived from WebKit (they use to have their own
- engine)
- Firefox: Gecko (being progressively rewritten and optimized)
- MS Internet Explorer (IE, MSIE): MSHTML
- Edge: EdgeHTML (derived from MSHTML)

History: Browsers of Today

Mobile:

- Android:
- Chrome: Blink (derived from WebKit)
- Edge: WebKit
- Opera: WebKit
- Firefox: Gecko
- Safari: WebKit
- All other browsers are required to use the IoS WebKit engine

History: Browsers of Today



Add a Sports shortcut to your Home screen

FIFA World Cup™

Tap a match to see stats, news and more

Quarter-finals

Uruguay
France

Tomorrow
12:00 am

News for world cup

England finally playing on the edge to become World Cup contender
ABC News - 9 mins ago
England is pushing the limits at the **World Cup** under manager Gareth Southgate and the ploy may just ...

World Cup 2018 England vs. Colombia: Photo reveals disgusting disrespect
NEWS.com.au - 7 hours ago

World Cup 2018 FIFA: John Obi Mikel father hostage story
NEWS.com.au - 2 hours ago

2018 World Cup
Football tournament

The 2018 FIFA World Cup is the 21st FIFA World Cup, an international football tournament contested by the men's national teams of the member associations of FIFA once every four years. Wikipedia

Teams: 32
Dates: 14 Jun. 2018 – 15 Jul. 2018

Modern Web Application

- Client - a lot more interactive and actively renders content within the browser
- Application logic split between client and server – client can execute (JavaScript) code
- No need to reload the web page for updating the state of the page being displayed (DOM)
- Rich message passing interface with the server through AJAX messages

Modern Web Application

- Client-side components
  - HTML/DOM
  - CSS
  - JavaScript
- AJAX messages (client-server interactions)
- Server code (Node.js or any other platform)

Modern Web Application: Client - HTML/DOM

- Hyper-text markup language to describe the structure and contents of the initial page
  - Also has pointers to the JavaScript code (e.g., <script>)
- Is retrieved by the browser and parsed into a tree called the Document Object Model (DOM)
  - Common way for elements to interact with the page
  - Can be read and modified by the JavaScript code
  - Modifications to the DOM are rendered by browser

Modern Web Application: Client - CSS

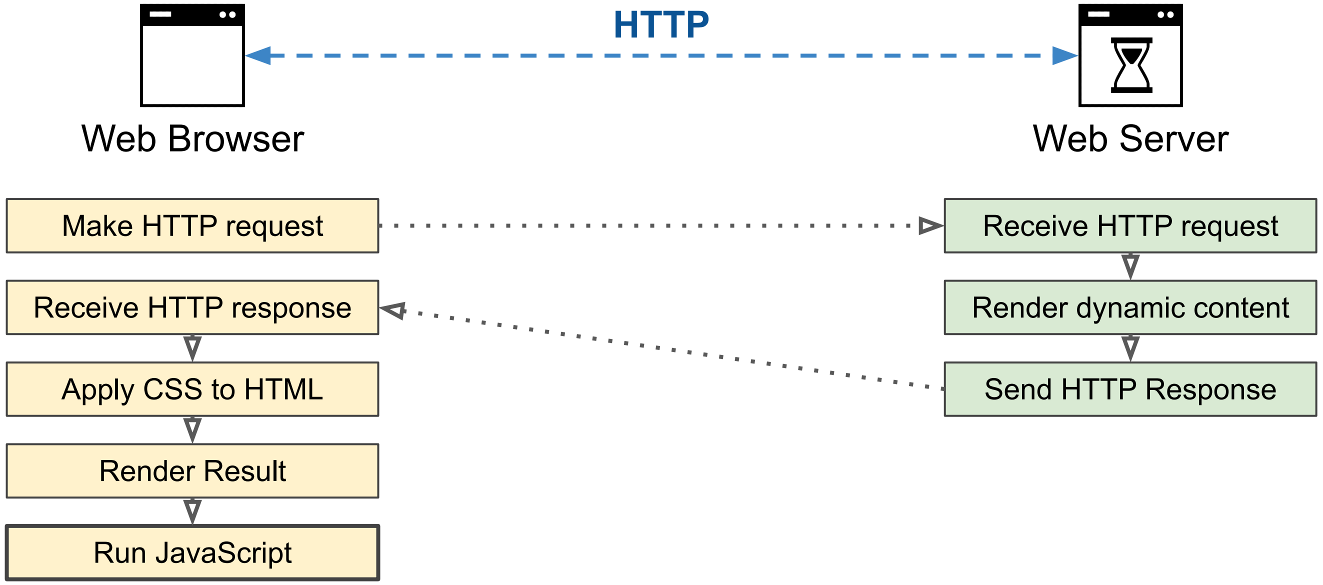
- CSS (Cascading style sheets) separate the content of the page from its presentation
- Written in a **declarative** fashion through **selectors and rules**
- Ensure uniformity by applying the rule to all elements of the webpage in the DOM

Modern Web Application: Client - JavaScript

- Unique to modern web applications and provides active functionality
- Executed when `<script>` tag is encountered or when events are triggered on DOM elements
- Is a full-fledged programming language with many advanced features (and some bad ones)

JavaScript: Browsers become Runtime Systems

Anatomy of a Web Application (2000s~)



Execution Model: Single-threaded Event loop

At the heart of the JavaScript engine (“interpreter”) is the **event loop**

- Functions are invoked **in response to events**
- The event loop processes functions **one by one**
- Unconventional execution model compared to existing languages

We will discuss more on this later in the course

Introduction to JavaScript: HTML `<script>` tag

1. Inline JavaScript - directly as part of the HTML document

```html
<html>
     <head>
          <title>My JavaScript Page</title>
     </head>
     <body>
          Hello
          <script type="text/javascript">
          var i = 2+2;
          document.writeln(i);
          </script>
          World
     </body>
</html>
```

Introduction to JavaScript: HTML `<script>` tag

2. External JavaScript - asynchronously loaded via HTTP

```html
<html>
     <head>
          <title>My JavaScript Page</title>
     </head>
     <body>
          Hello
          <script type="text/javascript" src="app.js"></script>
          World
     </body>
</html>
```

Introduction to JavaScript: HTML `<script>` tag

2. External JavaScript - asynchronously loaded via HTTP

```
// app.js
alert("Hello World!");
```

Introduction to JavaScript: HTML `<script>` tag

2. External JavaScript - asynchronously loaded via HTTP

```html
<html>
    <head>
        <title>My JavaScript Page</title>
    </head>
    <body>
        Hello
        <script type="text/javascript" src="app.js"></script>
        World
    </body>
</html>
```

Clear separation of
document structure (HTML)
from style (CSS) and
from application logic (JavaScript).

RECOMMENDED

Class Activity: Including JavaScript in HTML

Create **index.html** and write the following HTML code

```
<html>
     <head>
          <title>My JavaScript Page</title>
     </head>
     <body>
          Hello
          <script type="text/javascript">
          document.writeln(“World!”);
          </script>
     </body>
</html>
```

lecture-2/activity1.html

Class Activity: Including JavaScript in HTML

Create **app.js** and write the following JavaScript code

```
document.writeln(“World!”);
```

lecture-2/activity1.js

Class Activity: Including JavaScript in HTML

Change **index.html** to load script from **app.js** source file

```
<html>
     <head>
          <title>My JavaScript Page</title>
     </head>
     <body>
          Hello
          <script type="text/javascript" src="app.js"></script>
     </body>
</html>
```

lecture-2/activity1.html

Introduction to JavaScript: Comments

- Useful to document your JavaScript code!
    - Information for **other team members** and **yourself**
    - Single-line comments: `// example comment`
    - Multi-line comments: `/* example
      comment */`

```
// This line will be ignored by the Javascript engine

greeting = “Hello”; // This is also ignored

/* <- Starting here,
multiple lines will be ignored
up to this token right here -> */
```

Introduction to JavaScript: The `window` Object

JavaScript VMs have many **built-in** objects

- `window` object represents the enclosing window of a browser
    - It holds the **global context** of a JavaScript program
    - In non-strict mode, **any undeclared variables** are attached to the `window` object

```
greeting = “Hello”;

console.log(window.greeting);
```

TRY IT!

Introduction to JavaScript: The `window` Object

JavaScript VMs have many **built-in** objects

- `window` object represents the enclosing window of a browser
  - It holds the **global context** of a JavaScript program
  - In non-strict mode, **any undeclared variables** are attached to the `window` object

```
greeting = “Hello”;

console.log(window.greeting);


window.greeting = “Hello”;

console.log(greeting);
```

TRY IT!

Data Types

1. Introduction to JavaScript
2. **Data Types**
3. Statements and Expressions
4. Class Activity

Data Types: Primitive Objects

- Boolean: `true` or `false`
- Number: `1, 3.1412, 1.6e3, 01011001`
  - There is no distinction between Integers and Floating Point Numbers
- String: `“Hello”, ‘World’`

```
b = false;
n = 42;
s = “Hello World!”;

console.log(typeof b);   // prints: boolean
console.log(typeof n);    // prints: number
console.log(typeof s);    // prints: string
```

Data Types: Array

- Used to hold multiple objects in a sequence
- Arrays in JavaScript are **not typed** and **dynamic**
    - Can hold **any object** regardless of type
    - Can **add** or **remove** items **anytime**

```
my_list = [ “Hello World!”, 42, true ];

console.log(my_list[0]);   // prints: Hello World!
console.log(my_list[1]);   // prints: 42
console.log(my_list[2]);   // prints: true

my_list.push(“JavaScript is great!”);

console.log(my_list[3]);   // prints: JavaScript is great!
```

Data Types: Array

- Used to hold multiple objects in a sequence
- Arrays in JavaScript are **not typed** and **dynamic**
    - Can hold **any object** regardless of type
    - Can **add** or **remove** items **anytime**
- Arrays can store Arrays

```
my_2d_list = [
    [ “A”, “B”, “C” ],
    [ “D”, “E”, “F” ],
    [ “G”, “H”, “I” ]
];

console.log(my_2d_list[1][2]);    // prints: F
```

Data Types: Associative Array

- key-value data structure, similar to Python dictionary

```
dictionary = {
   ab: “Alberta”,
   bc: “British Columbia”,
   on: “Ontario”,
   qc: “Quebec”
};

console.log(dictionary.bc);      // prints: British Columbia

code = “qc”;
console.log(dictionary[code]);  // prints: Quebec


```

Data Types: Associative Array

- Can be arbitrarily nested

```
member = {
   name: “Alice”,
   age: 25,
   address: {
      province: “BC”,
      city: “Vancouver”,
      street: “123 Main Street”
   }
};

console.log(member.address.city);  // prints: Vancouver
member.phone = “012-345-6789”;

```

Data Types: Associative Array

- Objects have **properties**
    - Properties point to other Objects in the heap
    - Properties can be dynamically added, removed, or re-assigned a value

```
member = {};

member.name = “Alice”;
member.phone = “012-345-6789”;

console.log(member.name);  // prints: Alice

delete member.name
console.log(member.name);  // prints: undefined
```

TRY IT!

Data Types: Function

- Function is also an Object

```
select_max = function (number_list){
    /* do something */
};

console.log(select_max); // prints: [Function: select_max]
                         // *output may differ between browsers
```

Data Types: Function

- Function is also an Object

```
select_max = function (number_list){
   /* do something */
};

console.log(select_max); // prints: [Function: select_max]
                         // *output may differ between browsers

select_max.description
  = “Returns the maximum value from an Array of numbers”;

console.log(select_max.description);
  // prints: Returns the maximum value from an Array of numbers

```

Data Types: `null` and `undefined`

- `null` is actually something
  - It indicates the **absence** of a value
  - `null` itself is an object
  - Big **source of confusion**; dubbed as a major **BUG**
- `undefined` is when there is actually nothing

```
null_data = null;
undefined_data = undefined;

console.log(typeof null_data);      // prints: object
console.log(typeof undefined_data); // prints: undefined

console.log(window.foo); // prints: undefined
```

TRY IT!

Data Types: Summary

Primitive Types:

- boolean
- number
- string
- undefined

Complex Types:

- function
- object

Important Notes:

- null vs undefined

Statements and Expressions

1. Introduction to JavaScript
2. Data Types
3. **Statements and Expressions**
4. Class Activity

Statements and Expressions: Variable Declaration

- `var` keyword used to declare variables
- No **types** - JS is “duck-typed”

```
“use strict”;

var width;
var height;

var width, height, length;

var width = 10;
var width = 20, height = 5, length = 10;
var volume = width * height * length;

console.log(volume);
```

Statements and Expressions: Assignment Statement

- = operator used to assign a new value to a reference
    - In strict mode, assignment is allowed only for declared variables

```
“use strict”;

var width = 20, height = 5, length = 10;
var volume1 = width * height * length;

console.log(volume1);

width = 10;
height = 15;
var volume2 = width * height * length;

console.log(volume1, volume2);
```

Statements and Expressions: Binary/Unary Expression

Arithmetic

```
a + b;
a - b;
a * b;
a / b;
a % b;
```

Bitwise

```
~b;

a & b;
a | b;
a ^ b;
a ~ b;
a << b;
a >> b;
a >>> b;
```

Logical

```
!b;

a == b;
a === b;
a != b;
a !== b;
a > b;
a >= b;
a < b;
a <= b;

a && b;
a || b;
```

Statements and Expressions: Binary/Unary Expression

- 2 Different notions of **equality**
  - `a == b` : a and b are “equivalent”
    - Loose equality
    - Equal if the values are equivalent
    - **Type coercion** performed implicitly
  - `a === b` : a and b are “identical”
    - Strict equality
    - **Type** and **value** are both equal
    - For an Object, its **value** is its location in the heap (“pointer”)

Class Activity:

What would be the output of the following code?

```
var x = 5;
console.log(x == 5);        // prints?
console.log(x != 5);        // prints?
console.log(x >= 5);        // prints?
console.log(x < 5);         // prints?

console.log(x == “5”);      // prints?
console.log(x === “5”);     // prints?
console.log(x != “5”);      // prints?
console.log(x !== “5”);     // prints?
console.log(x !== 5);       // prints?
```

Class Activity:

What would be the output of the following code?

```
var x = 5;
console.log(x == 5);        // prints: true
console.log(x != 5);        // prints: false
console.log(x >= 5);        // prints: true
console.log(x < 5);         // prints: false

console.log(x == “5”);      // prints: true
console.log(x === “5”);     // prints: false
console.log(x != “5”);      // prints: false
console.log(x !== “5”);     // prints: true
console.log(x !== 5);       // prints: false


```

Class Activity:

What would be the output of the following code?

```
var x = { name: “Foo”, value: 5 };
var a = { name: “Foo”, value: 5 };
var b = x;

console.log(a.name === x.name);   // prints?
console.log(a.value === x.value); // prints?
console.log(a === x);             // prints?
console.log(b === x);             // prints?
```

Class Activity:

What would be the output of the following code?

```
var x = { name: “Foo”, value: 5 };
var a = { name: “Foo”, value: 5 };
var b = x;

console.log(a.name === x.name);   // prints: true
console.log(a.value === x.value); // prints: true
console.log(a === x);             // prints: false
console.log(b === x);             // prints: true
```

Statements and Expressions: Call Expression

- Function calls have the form:
  - `functionName (argument1, argument2, argument3, ...)`
  - Invokes function referred by `functionName` with the given *arguments*
  - Same as many other languages

```
console.log(“Foo”);
alert(“Foo”);
setTimeout(alert, 1000, “Foo”);
setInterval(alert, 1000, “Foo”);
```

Statements and Expressions: Function Declaration

- Functions can be declared with the `function` keyword
  - Can accept arbitrary arguments
  - No need to specify the return type
  - Lexical scoping - functions can have local variables that inherit the local context at the time of declaration *(we will cover this in more detail later)*

```
function density(mass, width, height, length){
   var volume = width * height * length;
   return mass / volume;
};

density(10, 20, 5, 10);
```

Variable and Function Declaration: Hoisting

Variable and Function Declarations are **hoisted**

- **Processed before** other expressions in the program
- To avoid confusion, **best to put** Variable Declarations and Function Declarations **at the top** of the program

```
console.log(density); // prints: [Function: density]

function density(mass, width, height, length){
   var volume = width * height * length;
   return mass / volume;
};
```

Class Activity: Boolean operators

Implement the following function

```
/* returns true if value is between lower and upper */
function isBetween(value, lower, upper){
   return ( /* ... */ );
};
```

Statements and Expressions: If, Else, Else If

- `if` statements are used to conditionally execute code
  - Has the form `if (condition) {expression} else {expression}`
  - `else` block is optional

```
if (temperature < 0) freezes = true;
else freezes = false;
```

Statements and Expressions: If, Else, Else If

- Blocks of code can be grouped with {}

```
if (temperature < 0){
   freezes = true;
   boils = false;
}
else {
   freezes = false;
   boils = true;
}
```

Statements and Expressions: If, Else, Else If

- `if` statements can be followed by multiple `else if` blocks

```
if (temperature < 0){
    freezes = true;
    boils = false;
}
else if (temperature < 100){
    freezes = false;
    boils = false;
}
else {
    freezes = false;
    boils = true;
}
```

Statements and Expressions: Switch

- `switch` statements can be used to handle many conditions

```
switch (group){
   case “child”:
       fee = 0;
       break;
   case “adult”:
       fee = 10;
       break;
   case “senior”:
       fee = 4;
       break;
   default:
       break;
}
```

Statements and Expressions: For

- `for` statements are used to repeat a block of code
- Similar to other languages, the `for` statement accepts 3 expressions:
  - Initial condition - run at the beginning of the loop
  - Termination condition - if this evaluates to true, the loop will exit
  - Increment expression - run after each iteration of the loop

```
for (          ;        ;    ){
   console.log(i);
}
```

Statements and Expressions: For

- `for` statements are used to repeat a block of code
- Similar to other languages, the `for` statement accepts 3 expressions:
  - Initial condition - run at the beginning of the loop
  - Termination condition - if this evaluates to true, the loop will exit
  - Increment expression - run after each iteration of the loop

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

Statements and Expressions: For

- `for … in` statements can be used to easily iterate through an object

```
var item = {
   a: 3,
   b: 1,
   c: 4
}
for (var key in item){
   console.log(key + “ : ” + item[key]);
}
```

Statements and Expressions: While, Do While

- `while` statements are similar to a `for` statement
  - Only the termination condition is specified
  - No initial condition or increment expression

```
while (i < 10){
   console.log(i);
   i++;
}

do {
   console.log(i);
   i++;
} while (i < 10);
```

Statements and Expressions: Try, Catch, Throw

- `try - catch` statements are used to handle errors
  - `try` block to enclose code that *might* throw an error
  - `catch` block to handle the error if thrown

```
try {
   myFunction(i);
}
catch (err){
   if (err.code === “FooError”){
      handleError(err);
   }
   else {
      throw err
   }
}
```

Statements and Expressions: New Expression

We will come back to this!

Class Activity

1. Introduction to JavaScript
2. Data Types
3. Statements and Expressions
4. **Class Activity**

Class Activity:

Consider the function `getRandomInt(min, max)` below:

```
function getRandomInt(min, max){
   min = Math.ceil(min);
   max = Math.floor(max);
   return Math.floor(Math.random() * (max - min)) + min;
};
```

Write the following functions:

1. `randomArray(n, min, max)`: returns an array of `n` random values generated between `min` and `max`
2. `sortArray(arr)`: returns an array containing all the values of `arr` sorted in ascending order