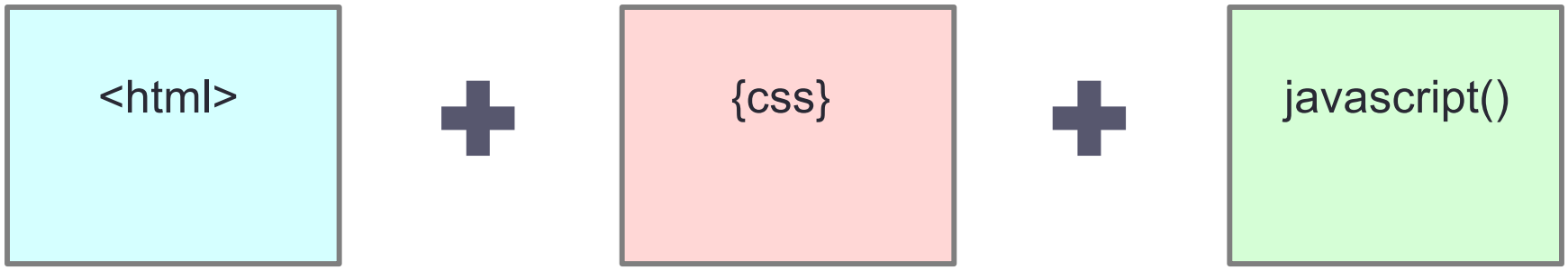
JavaScript

CS 4640
Programming Languages for Web Applications

[Robert W. Sebesta, "Programming the World Wide Web
Jon Duckett, Interactive Frontend Web Development]

How HTML, CSS, and JS Fit Together



Content layer

The HTML gives the page structure and adds semantics

Presentation layer

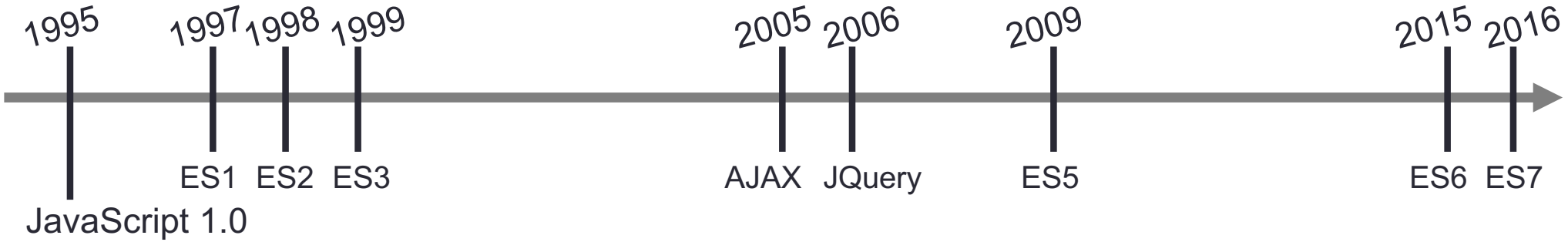
The CSS enhances the HTML page with rules that state how the HTML content is presented

Behavior layer

The JS controls how the page behaves, adding interactivity

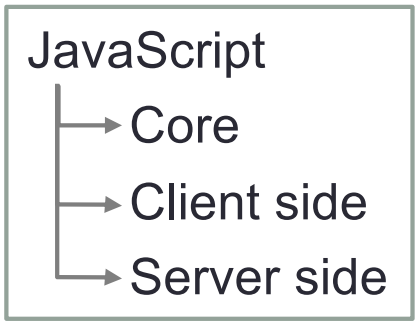
JavaScript: Some History

- JavaScript was introduced as part of the Netscape 2.0 browser
- Microsoft soon released its own version called Jscript
- European Computer Manufacturers Association (ECMA) developed a standard language known as ECMAScript
- ECMAScript Edition 6 is widely supported and is what is commonly called "JavaScript"
- JavaScript = ECMAScript + DOM API
  - ECMAScript: Language specification
  - DOM API: Communication with HTML



JavaScript

- JavaScript is **not** Java
- JavaScript is a scripting language:
  - Embedded in HTML
  - Interpreted as the page is loaded
  - Can manipulate the HTML page



- Primary purpose is for client-end processing of HTML documents
  - Netscape's Livewire, first server-side JS engine, allows JS to be used for form processing on the server
  - We will not be discussing server-side JS
- Dynamically typed – no type checking

JavaScript Characteristics

- JavaScript does not need to be compiled
  - JS is an interpreted language
  - A JS interpreter is software that runs inside a browser that reads and executes JavaScript

- Interpreted vs. compiled languages:
  - Advantage: simplicity
  - Disadvantages: efficiency, maintainability, scalability, reliability

Why and Why Not JavaScript?

- What can be done with JS on the client and cannot be done with other techniques on the server?
  - Monitor user events and take actions
  - Some dynamic effects
- What can be done on both client and server, but are better with JS?
  - Build HTML dynamically when page is loaded
  - Interactive web pages
  - Communicate with the server asynchronously (Ajax)
- What are the drawbacks of JS?
  - Platform dependent
  - Can be turned off by users
  - Performance depends on the user's hardware, not the server
  - Security – JS code is visible in the browser
  - Hard to write reliable and maintainable JS

Embedding JS in HTML

Use the <script> tag to embed JS code in <head> or <body>

```
<script type="text/javascript">
   // code goes here
</script>
```

- Functions and code that may execute multiple times are typically placed in the <head> section
  - These are only interpreted when the relevant function or event-handler are called
- Code that needs to be executed only once, when the document is first loaded is placed in the <body> section

Embedding JS in HTML

- JS code may be written in a separate file. The external file can be included in an HTML file using the src attribute of a <script> tag

```
<script type="text/javascript" src="path/to/file.js"></script>
```

- JS code is visible in the client browser
  - Do not "hardcode" anything that you don't want the client to see

Script Calls

Some script calls may be embedded in the HTML tags

```
<select name="country" onchange="jmp(url)">
```

or

```
<a href="javascript:newWindow('resources/JSPWebResources.html')">JSP resources</a>
```

```
function newWindow(url)
{
   hWnd = window.open (url,"HelpWindow","width=410,height=180,resizable=yes,scrollbars=yes");
}
```

Script is evaluated once encountered by browser

[Will revisit this when we discuss functions]

First Example

```
<!DOCTYPE html>
<html>
<head>
<title>First JavaScript Example</title>
</head>
<body>
<h2>This line is straight HTML</h2>
<h3>
<script type = "text/javascript">
    document.write("These lines are produced by<br/>");
    document.write("the JavaScript program<br/>");
    alert("Hey, JavaScript is fun!");
</script>
</h3>
<h2>More straight HTML</h2>
<script type = "text/javascript" src="file.js"></script>
</body>
</html>
```

[see jex1.html]

Variables

- Variables are loosely typed
- Type is determined dynamically based on the value stored
  - The typeof operator can be used to check type of a variable
- Declarations are made using the var keyword
  - Variables declared but not initialized have the value undefined
  - Variables used and not declared or initialized have value Null
- Names start with letters, $, or _ followed by any sequence of letters, $, _, or digits
- Case sensitive

Variables

Variable is a container that hold things for later use

- String
- Number
- Undefined
- Null
- Objects (including arrays)
- Symbols
- Functions

```
var strVar = 'Hello';
var num = 10;
var undefinedVar;
var nulled = null;
var intArray = [1, 2, 3];
var sym = Symbol('Decscription of the symbol');
function setFocus(ele) {
    document.getElementById(ele).focus();
}
var focusVar = setFocus('firstName');
```

Variables

- Loose typing means that JS figures out the type based on the value

```
var x;      // type: Undefined
x = 2;      // type: Number
x = 'Hi';   // type: String
```

- Variables have block scope.
  - Declarations outside of any function are global
  - Declarations within a function are local to that function

Expressions

- If operator is + and an operand is string, it treats the + as a string concatenation operator and coerce other operand to string

```
var x = 'Hello';
var y = 4;
var result = x + y;    // 'Hello4'
```

- If operator is arithmetic, and string value can be coerced to a number, it will do so
- If string is non-numeric, result is `NaN` (NotaNumber)
- String can be explicitly converted to a number using parseInt and parseFloat

[see jex2.html]

Using Arithmetic Operators

Let's initialize, y = 4

Operator	Description	Example	Resulting x
+	Addition	x = y + 5	9
		x = y + "5"	"45"
		x = "Four" + y + "4"	"Four44"
-	Subtraction	x = y - 2	2
++	Increment	x = y++	4
		x = ++y	5
--	Decrement	x = y--	4
		x = --y	3
*	Multiplication	x = y * 4	16
/	Division	x = 10 / y	2.5
%	Modulo	x = y % 3	1

Using Assignment Operators

Let's initialize, x = 10

Operator	Example	Equivalent arithmetic operators	Resulting x
=	x = 5	x = 5	5
+=	x += 5	x = x + 5	15
-=	x -= 5	x = x - 5	5
*=	x *= 5	x = x * 5	50
/=	x /= 5	x = x / 5	2
%=	x %= 5	x = x % 5	0

Applying Comparison and Conditional operators

Let's initialize, x = 10

Operator	Description	Example	Result
==	Equal to (value only)	x == 8	false
		x == "10"	true
===	Equal to (value and type)	x === 10	true
		x === "10"	false
!=	Not equal (value only)	x != 5	true
!==	Not equal (value and type)	x !== "10"	true
		x !== 10	false
>	Greater than	x > 5	true
>=	Greater than or equal to	x >= 10	true
<	Less than	x < 5	false
<=	Less than or equal to	x <= 10	true

Chaining Multiple Comparisons with Logical Operators

Let's initialize, x = 10 and y = 5

Operator	Description	Example	Result
&&	And	(x == 10 && y == 5)	true
		(x == 10 && y > x)	false
||	Or	(x >= 10 || y > x)	true
		(x < 10 && y > x)	false
!	Not	!(x == y)	true
		!(x > y)	false
	Mix	(x >= 10 && y < x || x == y)	true
		((x < y || x >= 10) && y >= 5)	true
		(!(x == y) && y >= 10)	false

Control Structures (if Statement)

```
if (x == 5) {
    do_something();
}

if (x == 5) {
    do_something();
} else {
    do-something_else();
}

if (x < 5) {
    do_something();
} else if (x < 10) {
    do_something_else();
} else {
    do_nothing();
}
```

Control Structures (switch Statement)

```
switch (expression) {
    case value1:
        // code to execute
        break;
    case value2:
        // code to execute
        break;
    default:
        // code to execute if not value1 or value2
}
```

Looping (while Loop)

```
while (condition) {
    // code to execute
    // update to end the loop
}

var i = 1;
while (i < 5) {
    // code to execute
    i++;
}
```

Looping (do-while Loop)

```
do {
    // code to execute
} while (condition);

var days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
var i = 0;
do {
    var day = days[i++];
    console.log("It's " + day);
} while (day != "Wednesday");
```

Looping (for Loop)

```
for (assignment; condition; update;) {
    // code to execute
}
```

```
var cars = ["BMW", "Volvo", "Saab", "Ford", "Fiat", "Audi"];
var text = "";
var i;
for (i = 0; i < cars.length; i++) {
    text += cars[i] + "<br>";
}
```

Looping (for-in Loop)

```
var days = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday"];
for (var idx in days) {
   console.log("It's " + days[idx] + "<br />");
}
```

Array Objects

- More relaxed version of Java arrays
  - Size can be changed and data can be mixed
  - Cannot use arbitrary keys as with PHP arrays - index

- Creating arrays
  - Using the new operator and a constructor (Array) with multiple arguments

```
var A = new Array("hello", 2, "you");
```

  - Using the new operator and a constructor (Array) with a single numeric argument, assigned the size of array

```
var B = new Array(50);
```

  - Using square brackets to make a literal

```
var C = ["we", "can", 50, "mix", 3.5, "types"];

var D = [50];
```

Array Predefined Operations

- Concat two arrays into one – copies to end
- Join array items into a single string (commas between)
- Push, pop, shift, unshift – easy implementation of stacks and queues
  - Push and pop are a "right stack" – add/remove from end
  - Shift and unshift are a "left stack" – add/remove from beginning queue
- Sort
  - Sort by default compares using alphabetical order
  - To sort using numbers we pass in a comparison function defining how the numbers will be compared
- Reverse the items in an array

[see jex4.html]