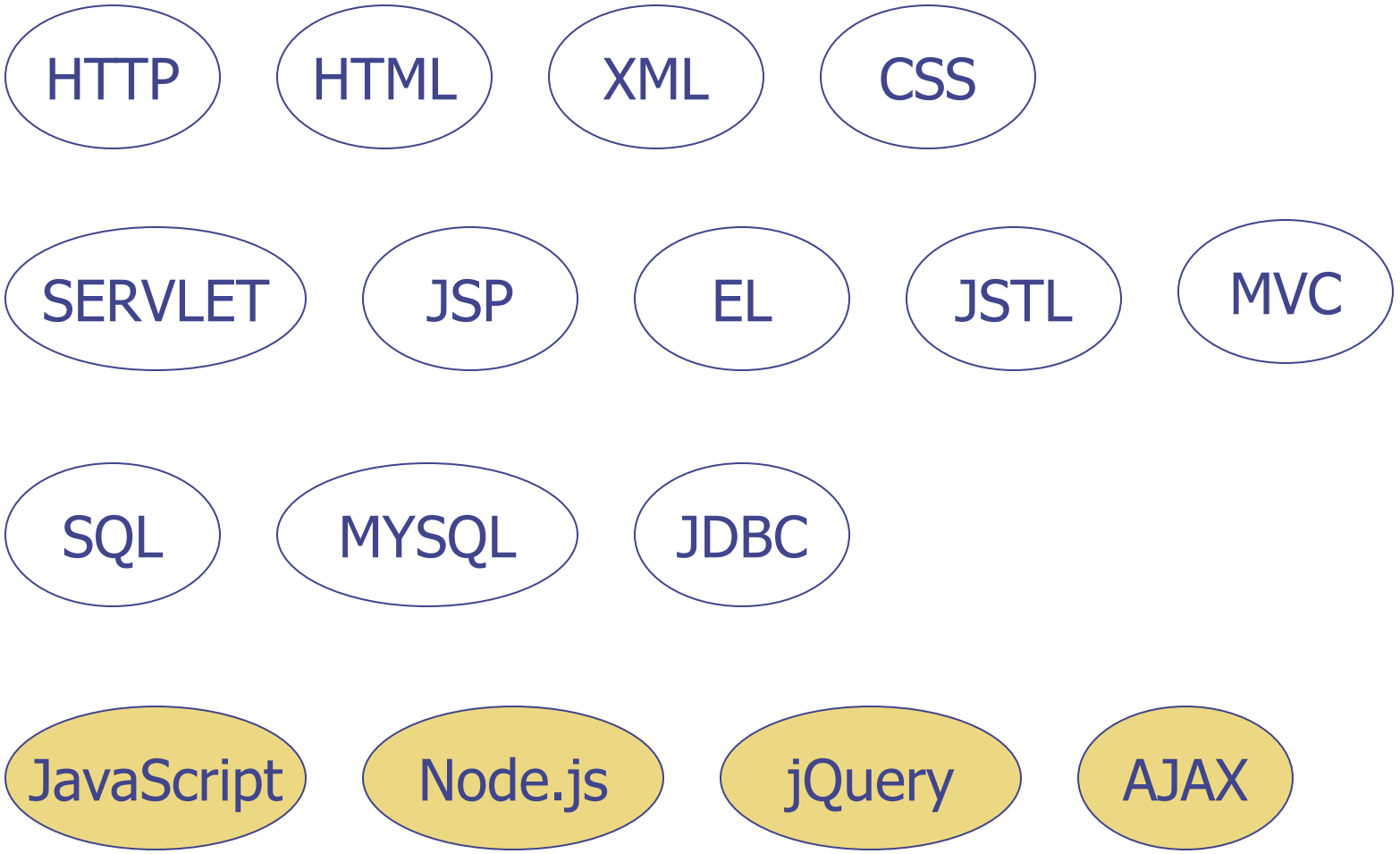
CS3220 Web and Internet Programming

JavaScript Basics

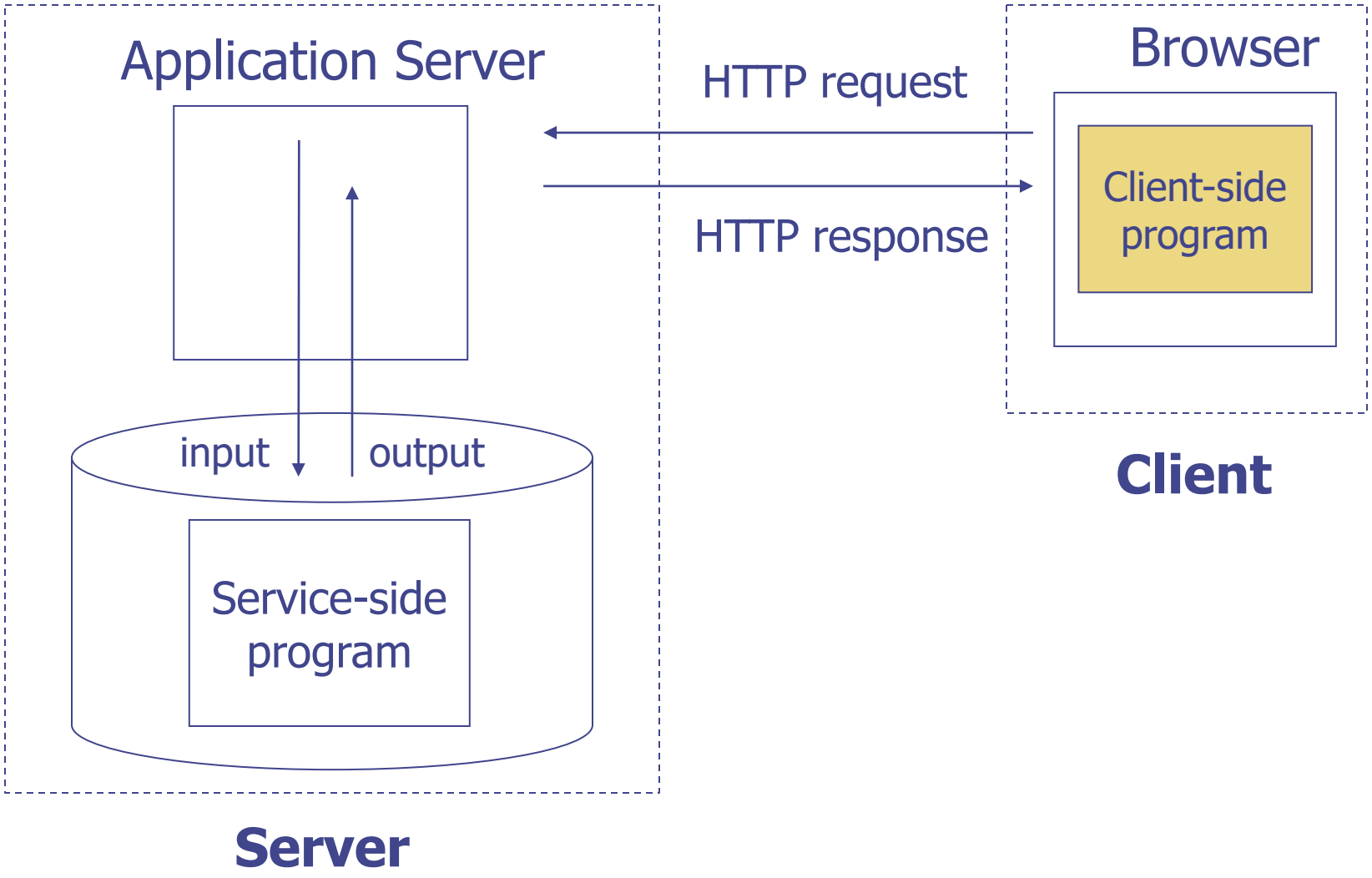
Chengyu Sun

California State University, Los Angeles

The Alphabet Soup of CS3220

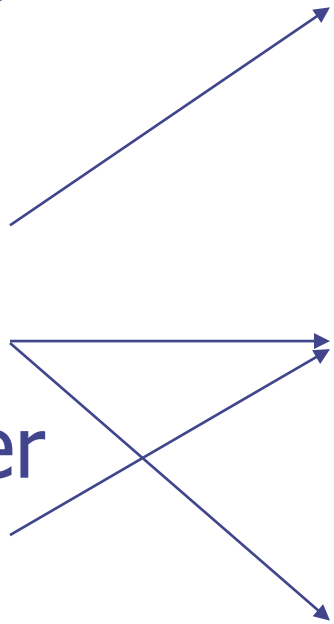


Web Application



Why Client-Side?

- Improve user experience
  - Interactive
  - Responsive
- Reduce server workload



Handle input events

Implement as much functionality on the client-side as possible

Hide the inevitable communication overhead from the user

Client-side Technologies

- HTML, CSS
- JavaScript
- Java Applet
- Rich Internet Application (RIA) technologies
  - JavaFX
  - Adobe Flex and Flash
  - Microsoft Silverlight

About JavaScript …

- Originally developed by Netscape
- Standardized as ECMAScript
- Many variations and languages built on top of JavaScript

Year	Version
1997	1
2009	5 (ES5)
2015	6 (ES6 or ES 2015)
2016	7
2017	8

… About JavaScript

Client-Side JavaScript

JavaScript Inside Browser

Node.js

JavaScript Outside Browser

Core JavaScript Language Features

Node.js Development Tools

- Node.js - https://nodejs.org/
- Text editors for developers
  - Visual Studio Code by Microsoft
  - Sublime Text
  - Atom by GitHub

Basic Usage of Node Shell

- A.K.A. Node REPL (Read-Eval-Print-Loop)
- Interactive mode: `node`
- Execute Node.js code: `node <file>`

Elements of an Imperative Programming Language

- Comments
- Types
- Literals and Variables
- Operators
- Expressions
- Statements
- Functions
- Classes and Objects
- Packages

Elements of JavaScript

- Comments
- Types
- Literals and Variables
- Operators
- Expressions
- Statements
- Functions
- Classes and Objects
- Modules

Comments

- Single-line comment: //
- Block comment: /* */

Types

- Boolean
- Number
- String
- Null
- Undefined
- (Symbol)
- Object

Primitive Types (i.e. types that define *immutable* values)

Literals

- Boolean: `true`, `false`
- Number: `123`, `4.56`
- String: `"hello"`, `'world'`
- Null and Undefined: `null`, `undefined`
- Template literal
- Object literal

Variables and Constants

```
let x;              // declare a variable x
x = 10;             // now x is a number
x = 'abc';          // now x is a string
const y = 20;       // y is a constant
```

- JavaScript variables are dynamically typed (think of them as references instead of storage spaces)

Variable Scope

```
a = 10;          // global scope
var b = 20;      // function scope
let c = 30;      // block scope
const d = 40;    // block scope
```

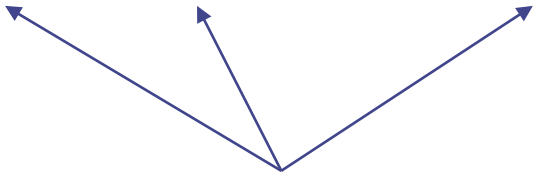
- Scope example
  - Global vs function vs block

Template Literal

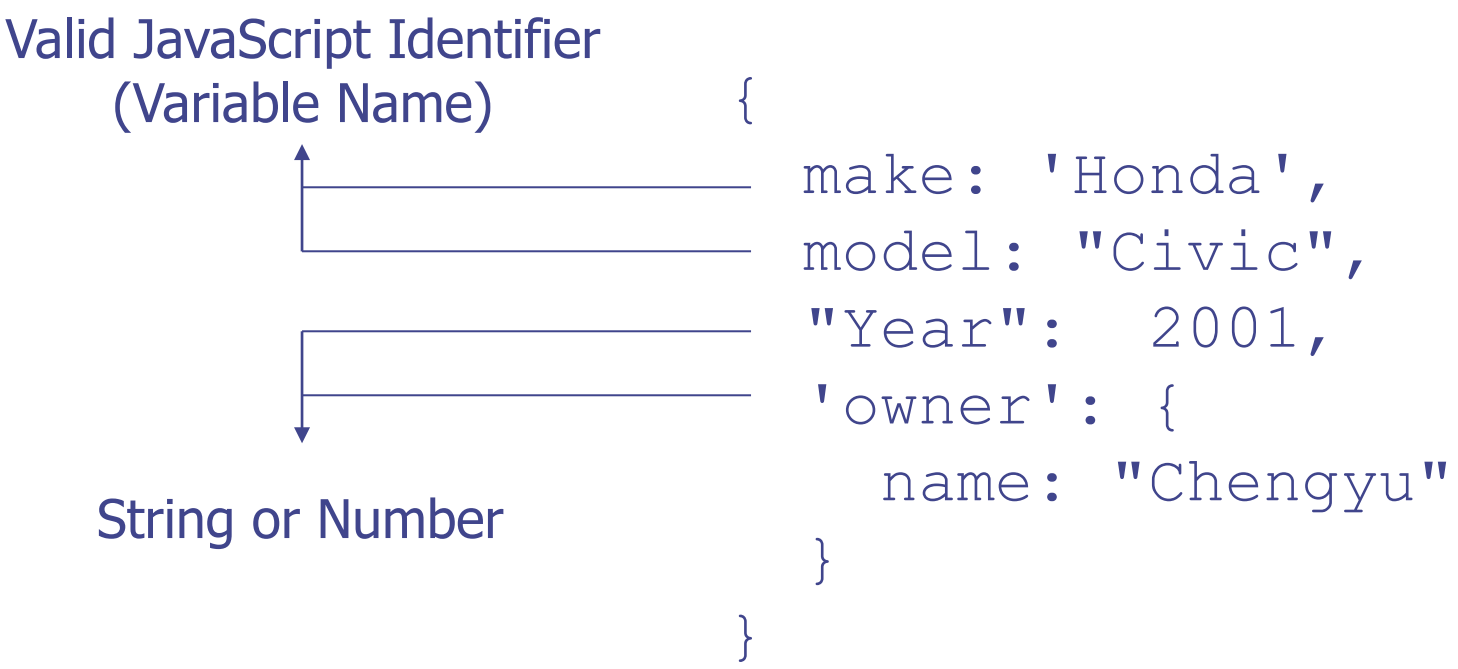
- A.K.A. Template String
- Example:

```
let a = 10;
let b = 20;

console.log( `${a}+${b} is ${a+b}` );
```



Object Literal



- An object literal consists of zero or more `key:value` pairs called *properties*

JSON (JavaScript Object Notation)

- Used as a data exchange format
- Based on a subset of JavaScript syntax
  - Strings are double quoted
  - Property keys are strings

```
{
   "make": "Honda",
   "model": "Civic",
   "year":  2001,
   "owner": {
      "name": "Chengyu"
   }
}
```

Object Property

Valid JavaScript Identifier
(Variable Name)

String or Number

```
console.log( obj.make );

console.log( obj["make"] );

obj[0] = 10;
```

Properties can be dynamically added

- What if there's a variable `make`? ➔
  Computed Property

Array

```
a = ["x", "y", "z"];
a.b = "hello";
a[100] = 10;
```



```
{
  0: "x",
  1: "y",
  2: "z"
}
```

- An array is a special object where array elements are stored as object properties
  - An array may have "holes" (i.e. undefined elements)
- Array has built-in properties like `length`

Operators

- All Java operators, e.g. `+`, `-`, `=`, `&&` ...
- Strict equality/inequality: **`===`**, **`!==`**
  - `===` true if same type and same value
- Type operators: `typeof`, `instanceof`
- Object property operators: `in`, `delete`

Boolean and Equality …

0 == false	
"" == false	
0 == ""	
null == false	
undefined == false	
! null	
! undefined	
! obj	

… Boolean and Equality

- In JavaScript, a *truthy* value is a value that is considered true when encountered in a Boolean context. All values are truthy unless they are defined as *falsy* (i.e., except for `false`, `0`, `""`, `null`, `undefined`, and `NaN`).

Statements ...

- All common Java statements, e.g. `if`, `for`, `while`, `switch`, `break`, `continue` ...
- There's no for-each loop (a.k.a. enhanced for loop in Java) in JavaScript
  - There's a forEach() method in Array
- `for…in` loop iterates over object property keys
- `for…of` loop iterates over object property values

... Statements

- *Strict equality check* is used in `switch` statement
- Semicolon is optional but recommended

Functions as First-class Citizens

- In JavaScript, functions are actually objects
  - Assigned to a variable
  - Assigned as a property of an object
  - *Function literals* (a.k.a. *function expressions, anonymous functions*)
  - Passed as a function argument
  - Returned as a function result

Function Examples

```
function foo() {
  console.log("foo");
}
```

Regular function declaration

```
bar = function() {
 console.log("bar");
};
```

- Function literal
- Function assignment

```
setTimeout( bar, 5000 );
```

Function as parameter

```
setTimeout( function() {
 console.log("foobar");},
  5000 )
```

Function literal as parameter

Function Arguments

```
function add(x,y) {
  return x+y;
}
```

```
add(10,20);
add("10","20");
add(10);
add(10,20,30);
```

- A special variable `arguments` hold all the arguments to a function
- `arguments` is not an array but similar to an array, e.g. `arguments.length`, `arguments[0]`, `arguments[1]`, ...

Arrow Functions

- A.K.A. *lambda expressions, lambdas*
- A more concise way to write function literals

```
function(a) {
  return a*2
}
```

```
(a)=>{return a*2}
```

```
a=>a*2
```

Method

- A *method* is a function that is a property of an object
- `this` in a method refers to the object the method is called on

Method Example

```
let john = {
  firstName: "John",
  lastName: "Doe",
  greeting: function(){
    console.log(`Hi, ${this.firstName}!`);
}

john.greet();
```

Some Important JavaScript Functions

- Array methods
  - Example: sum of array elements using `for` loop, `forEach`, and `reduce`
- JSON methods

Readings

- The Modern JavaScript Tutorial
- MDN JavaScript Reference