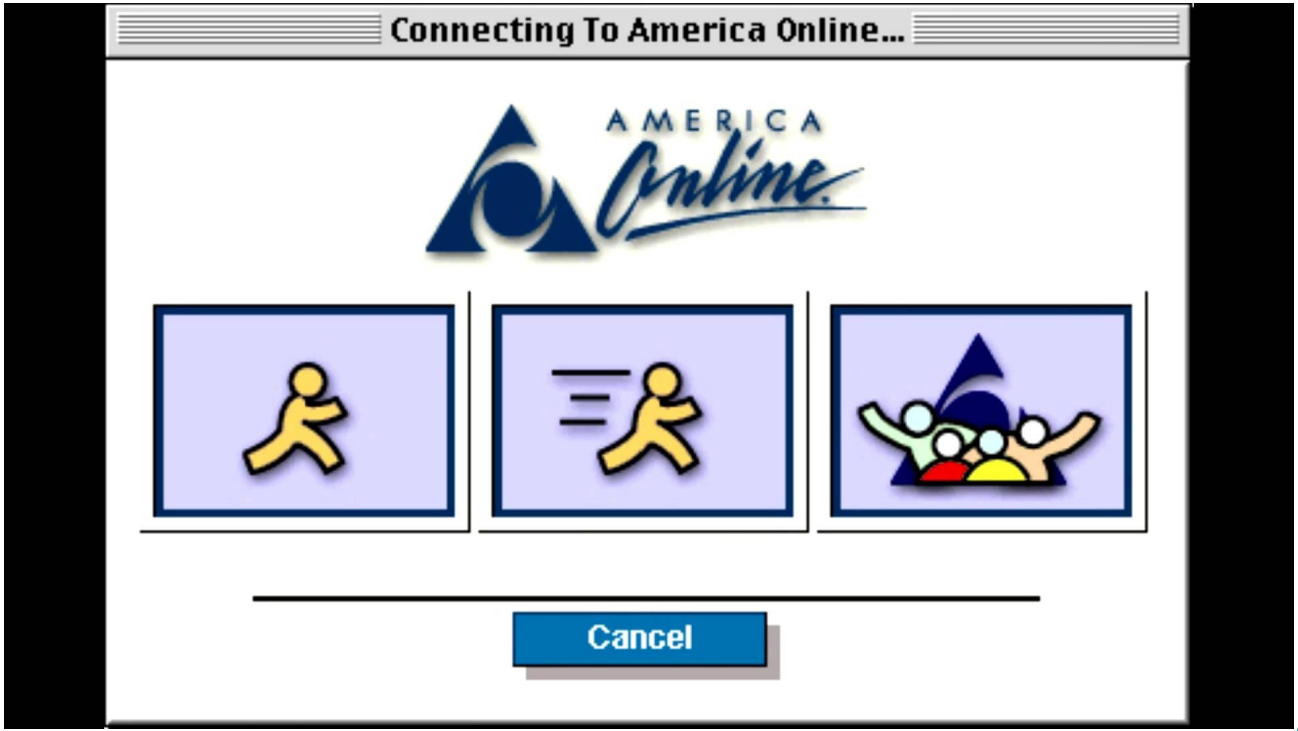
Module 3 - Lecture 5

Introduction to JavaScript

Client-Side Scripting

The Internet revolution



Static HTML pages are boring

JavaScript enables dynamic behavior by executing code in the user's browser.

Use cases

- Responding to events (clicks, typing, scrolling, resizing window).
- Interact with web services (APIs).
- Manipulate HTML without refreshing!

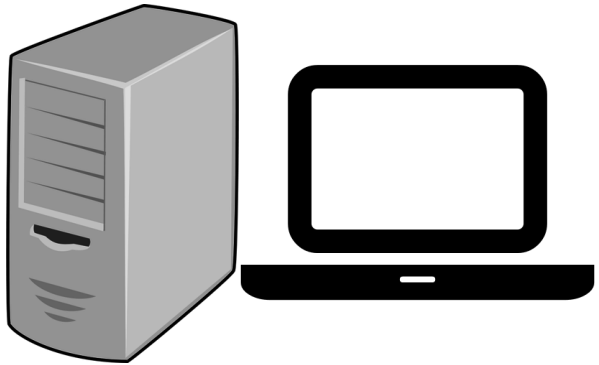
Benefits:

- More responsive and interactive experience for the user.
- Can reduce stress on your web server.

Web Application Architecture

CLIENT

DATABASE



WEB SERVER

JavaScript

Java / C# , SQL

Separation of Concerns

- **HTML** - Content
- **CSS** - Design
- **JavaScript** - Interactivity

JavaScript

JavaScript vs. Java / C#

- Java and C# require a **runtime** to execute.
- JavaScript requires a **browser** to execute.

- Java and C# code is **compiled**.
- JavaScript is **interpreted**.

- Java and C# are **statically and strongly typed** languages.
- JavaScript is a **dynamically and loosely typed** language.

Static/Strong vs. Dynamic/Loose typing

- Static typing requires that the user declare variables with a type.
  - Compile-time type checking.

- Strong typing means the variable doesn't change types at run-time.

- Dynamic typing doesn't require users to declare a type.
  - Type checking occurs at run-time.

- Loosely/Weakly typed languages allow for a program to infer the data type of a variable based on what a variable holds at run-time.

Declaring Variables

- var, let, const
  - Don’t use var. var allows redeclaration and var is function scoped via hoisting.
  - let and const are block-scoped, just like Java and C#.

```
{
    var x = 2; // Don’t use var
    let dayOfWeek = “Sunday”;
    const PI = 3.14;
}
```

Strict vs. Loose Equality

Strict Equality compares two operands for **both type and value.** Strict equality uses ===

```
{
    let x = 1;
    let y = 1;
    let z = x === y; // z is true
}
```

Loose Equality compares two operands for **value only after converting to a common type.** Strict equality uses ==

```
{
    let x = “1”;
    let y = 1;
    let z = x == y; // z is true
}
```

Arrays

- Not fixed in size.
- Zero-based indexing.
- Have a length property to determine current size.
- Unlike Java and C#, arrays in JavaScript can contain different data types.

Arrays

Declaration

```
let myArray = [];            // Declare empty array
let myArray = [1, 2, 5, 20]; // Declare and Initialize
```

Push/Pop - to use an array like a Stack.

```
myArray.push(1);      // add 1 to the end
myArray.pop();        // get last item in array
```

Accessing items or Assigning

```
let thirdItem = myArray[2];   // access third item in array
myArray[2] = 5;               // reassign third item
```

Concat - to combine two arrays.

```
let myArray = [1, 2];
let myOtherArray = [3, 4];
let finalArray = myArray.concat(myOtherArray); // [1, 2, 3, 4]
```

Objects

```
let myObject = {};      // Declare an empty object


// Declare an object with data
let myObject = {
      firstName: “John”,
      lastName: “Smith”,
      age: 40
};


// Access data from an object
let theFirstName = myObject.firstName;

// Re-assign data in an object
myObject.firstName = “Jane”;

// Assign data that doesn’t exist yet
myObject.occupation = “Developer”;
```

QUESTIONS?