ual: creative computing institute

Peckham DAZ

Accessible Web Development

Session 2: JavaScript

Date

Session Structure

- Lecture (1 hour)
- Break (15 mins)
- Labs Exercises (2 hours 30 mins)
- Debrief (15 mins)

Recap...

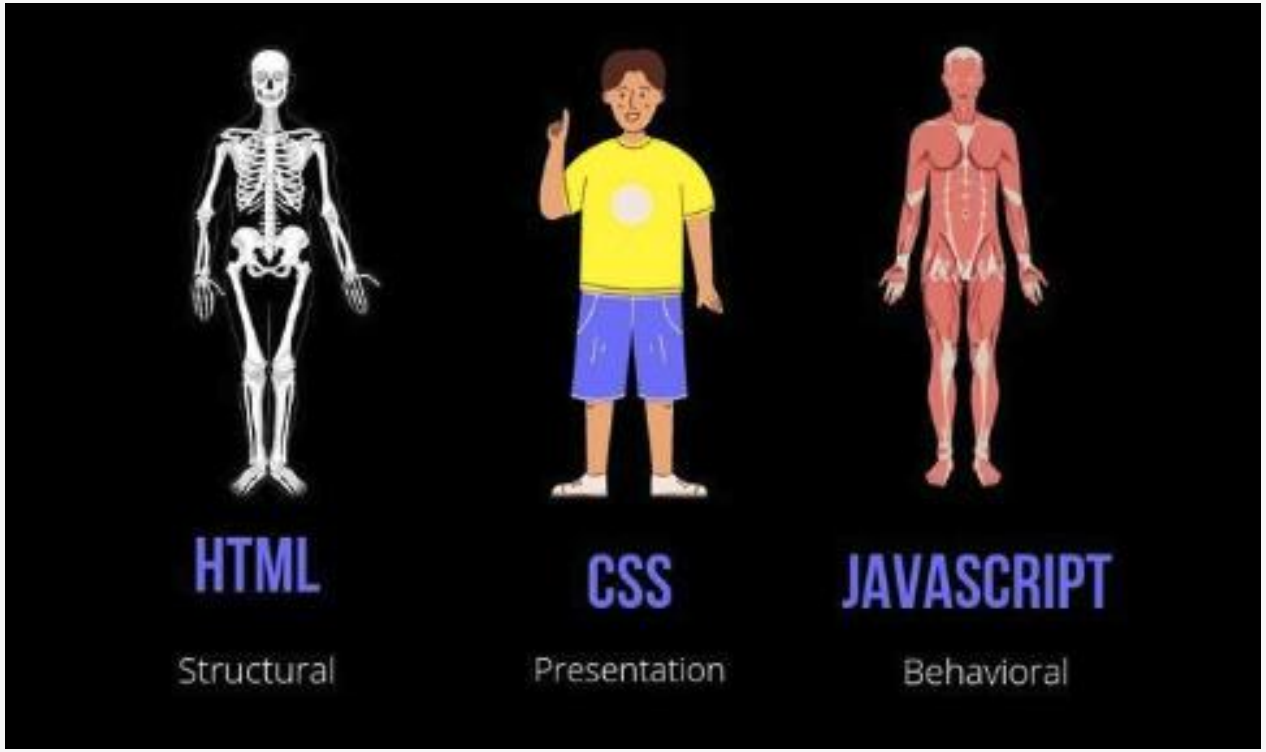
Front-end refers to the visible user interface that a user interacts with.

It’s made up of 3 languages:

HTML: The structure and layout of a page

CSS: The styling and look of a page

JAVASCRIPT: The functionality of a page



ual: creative computing institute

Background

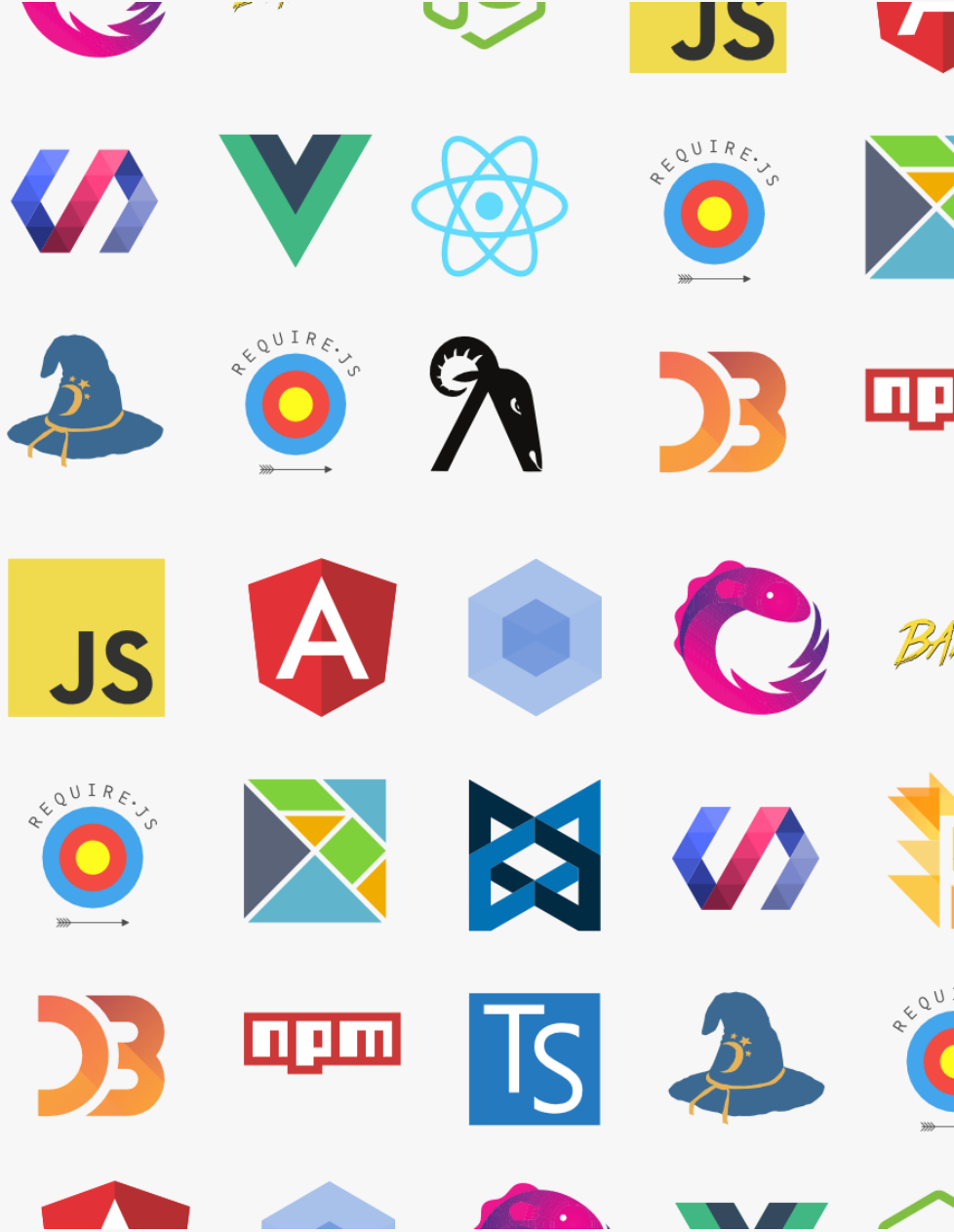
JavaScript

Origins

- Created in 1995 by Brendan Eich at Netscape.
- Originally named Mocha, then LiveScript, before becoming JavaScript.
- Designed to add interactivity to webpages in just 10 days.

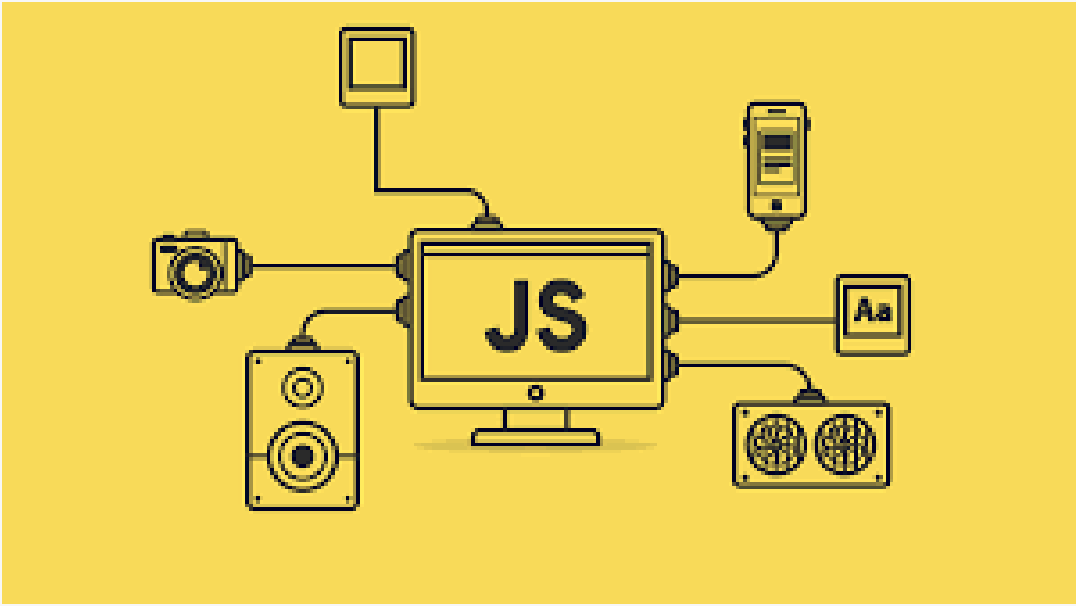
Evolution

- Standardized as ECMAScript (ES) in 1997.
- ES6 (current version) introduced modern features.
- Now used in frameworks and libraries such as React, Vue, and Node.js.



Why is it Important?

- Runs in all browsers without extra software.
- Enables interactivity, animations, and real-time updates.
- Works on both front-end and back-end (Node.js).
- Powers modern web applications, mobile apps, and even AI projects.



JavaScript can be used for anything...

https://p5js.org/examples/ - Make web-based art

https://threejs.org/examples/ - Make 3D websites/games

https://ml5js.org/ - Machine Learning in your browser

Note: JavaScript is not Java

ual: creative computing institute

JavaScript

How to write JavaScript code

Case Sensitivity

- JavaScript pays attention to whether letters are uppercase or lowercase in variable and function names.
- For instance, "myvar" and "Myvar" are treated as different names.

Variables

- Variables are containers for pieces of data - numbers, strings etc
- Must start with a letter, underscore (_) , or dollar sign ($)
- Declared using **var**, **let**, and **const.**
- **Const** cannot be edited once declared, but **var** and **let** can change throughout your program.

```
let name = "John"; // let keyword
const age = 30; // const keyword (constant value)
var isMarried = false; // var keyword
(global/function scope)
```

Scope

- **Global scope:** variables that can be accessed from anywhere in the program
- **Function scope**: variables that can be used within a function
- **Block variables:** variables that can be used within a code block

```
// Global Scope
var globalVar = "I am global";

function testScope() {
  // Function Scope
  var functionVar = "I am inside a function";

  if (true) {
    // Block Scope
    let blockVar = "I am inside a block";
    console.log(blockVar); // Accessible here
  }
// console.log(blockVar); // Error: blockVar is not defined
}

testScope();

// console.log(functionVar); // Error: functionVar is not defined
console.log(globalVar); // Accessible here
```

Core Datatypes

- **Boolean**
- **String**
- **Number (Integers & floating points)**
- **Array**
- **Object**
- Undefined
- Null

```
let stringVar = "This is a string"; // String
let numberVar = 42; // Number
let booleanVar = true; // Boolean
let nullVar = null; // Null
let undefinedVar; // Undefined
let objectVar = { key: "value" }; // Object
let arrayVar = [1, 2, 3, 4, 5]; // Array
```

Operators

Arithmetic

Addition(+), Subraction(-), Multiplication(*), Division(/)

Comparison

Equal to (==), Not equal to(!=), Strict equal to(===), Greater than(>), Less than (<)

Logical

AND(&&), OR(||), NOT (!)

```
let a = 10;
let b = 5;
let add = a + b; // Addition
let subtract = a - b; // Subtraction
let multiply = a * b; // Multiplication
let divide = a / b; // Division
let remainder = a % b; // Remainder
let exponent = a ** b; // Exponentiation
let isEqual = a == b; // Equality
let isStrictEqual = a === b; // Strict Equality
let isNotEqual = a != b; // Inequality
let isStrictNotEqual = a !== b; // Strict Inequality
let greaterThan = a > b; // Greater than
let lessThan = a < b; // Less than
```

Expressions

Expressions are code snippets that **evaluate to a value**, such as a combination of variables, operators, and function calls, which can be used to **perform calculations, assign values, or produce outputs**.

```
let sum = 5 + 3; // Addition
let product = 4 * 2; // Multiplication
let greeting = "Hello, " + name + "!"; // String concatenation
```

Conditional Statements

Allows your program to make decisions and perform actions based on whether a given condition is **true** or **false.**

```
if (a > b) {
  console.log("a is greater than b");
} else if (a < b) {
  console.log("a is less than b");
} else {
  console.log("a is equal to b");
}
```

Functions

Functions are **resuable blocks of code** that perform a specific task. Like a machine that takes an input, does something, and produces an output.

```
// Function to add two numbers
function addNumbers(a, b) {
  return a + b;
}
// Calling the addNumbers function
addNumbers(5, 10);
```

Arrays & Loops

- Arrays are data structures that allow you to store **multiple pieces of data** within a single variable.
- Array items are **indexed starting from 0** and are separated by a comma.
- Loops are constructs that allow you to **execute a block of code repeatedly** until a specific condition is met.
- For loops have a **counter variable** with an intial value, **a condition**, and **a counter** incrementer/decrementer.

```
// Array
let fruits = ["apple", "banana", "cherry"];
console.log(fruits[0]); // Accessing 0 array element

// For loop
for (let i = 0; i < 3; i++) {
  console.log(fruits[i]);
}
```

ual: creative computing institute

Accessible JS Examples

JS can make websites more inclusive and usable

Accessible Navigation

Some users rely on keyboards instead of a mouse to navigate through websites

How can we include these users?

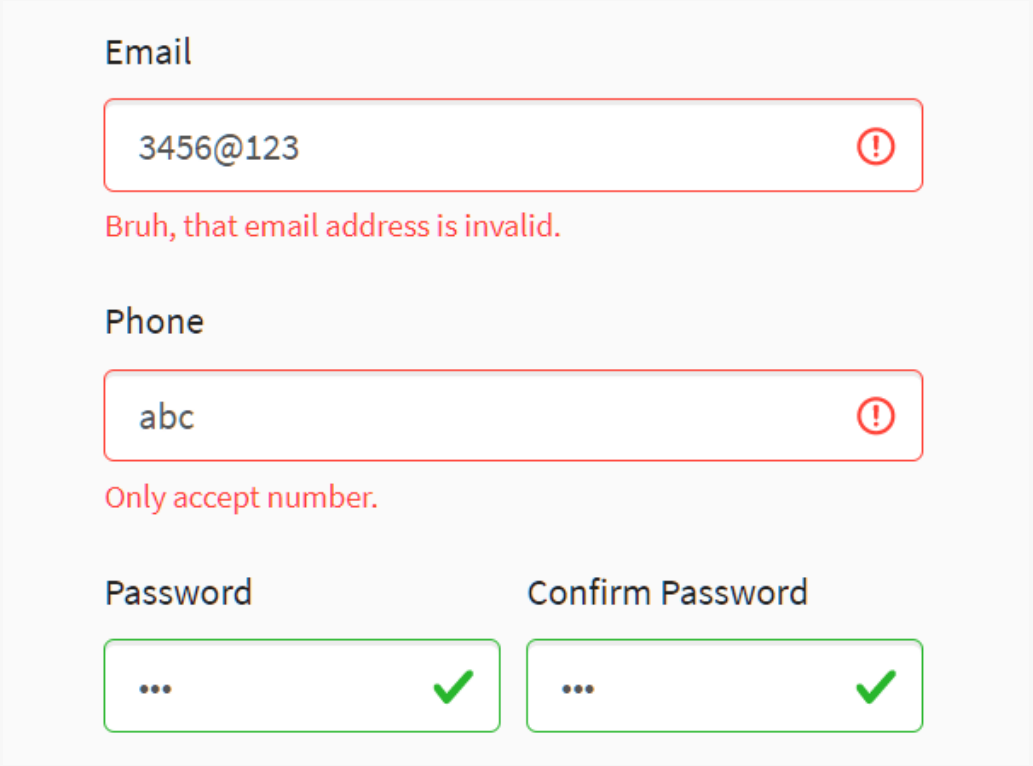
- Make sure buttons, links, and form fields can be accessed using the Tab key.

Accessible Forms

Users should be able to clearly understand when forms have been filled out incorrectly.

How can we make accessible forms?

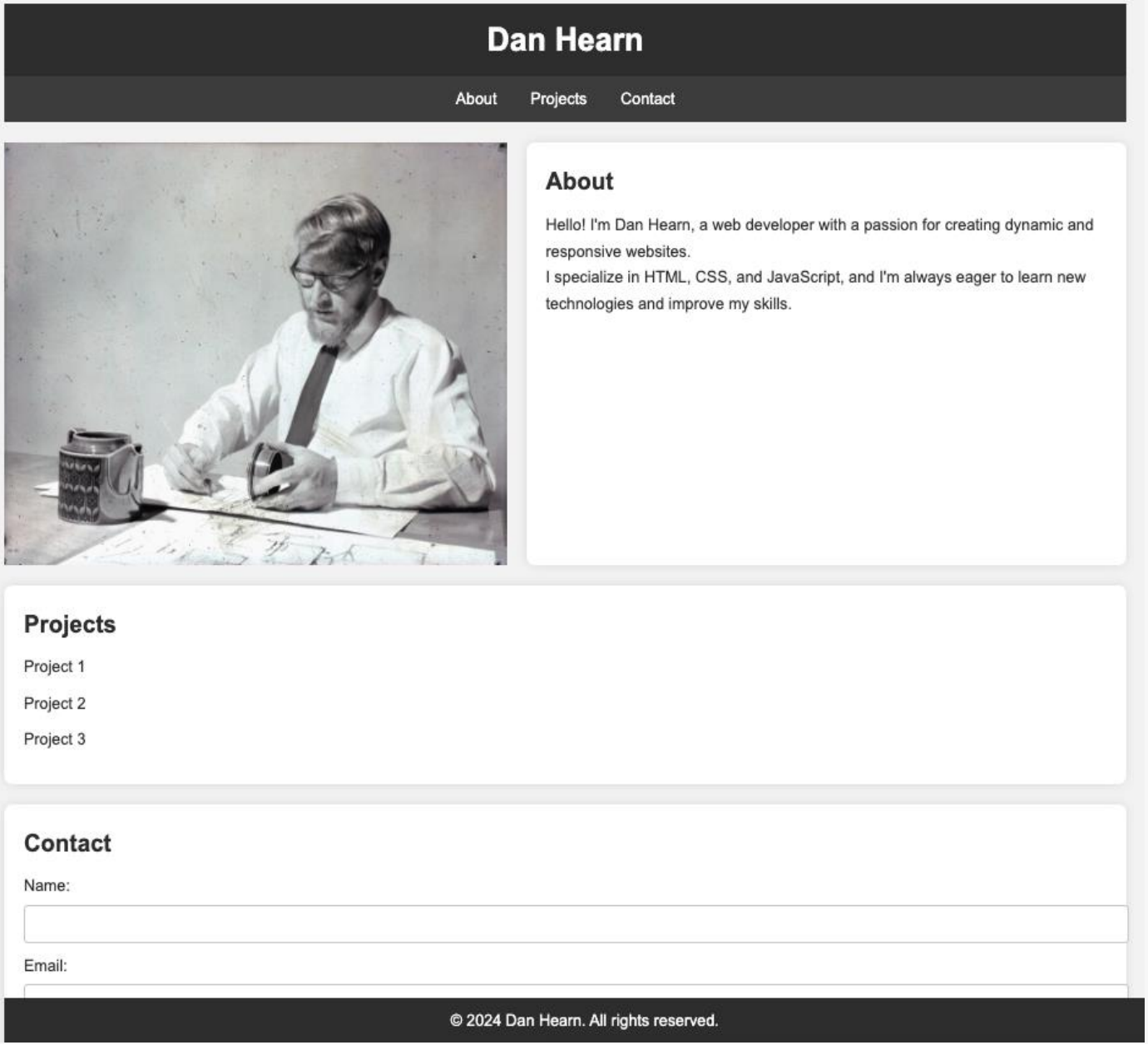
- Use JavaScript to display error messages near the input field.
- Use JavaScript to dynamically apply CSS styling to fields that are missing or incorrect



JavaScript Portfolio

Let's add some functionality to the portfolio we had a look at last session...

[JavaScript Portfolio]



Tips for JavaScript Coding

Finding Information

- Use **MDN Web Docs** for official JavaScript documentation.
- Check **W3Schools** for quick examples and explanations.
- Search **Stack Overflow** for answers to common issues.
- Read **error messages carefully** and search for solutions online.

Writing & Testing Code

- Break down problems into **small, testable parts**.
- Write clear, **well-structured code** with meaningful variable names.
- Keep functions focused on **one task** for easier debugging.

Tips for Debugging

Using Console & DevTools

- Use `console.log()` to check variable values and test logic.
- Open **DevTools (F12)** to inspect errors in the **Console** tab.
- Set breakpoints in the **Sources** tab to step through code.

Avoid Common Mistakes

- Check for **missing brackets, parentheses, or typos**.
- Use a **code editor with linting** (e.g., VS Code) to spot errors.
- Test code in **small sections** before running everything.

Using ChatGPT...

Avoid using it at the beginning of your coding journey! You must learn the basics before it becomes a useful tool.

If you do use it:

- **Do** get it to explain code or concepts you don't understand
- **Do** get it to debug or explain your code
- **Don't** get it to write code for you!!
- **Don't copy and paste.** Writing it out will help you understand.

Lab Exercises

Can be found on the Peckham DAZ github

ual: creative computing institute

Great work! ☺

@ual_cci

@ual_cci

arts.ac.uk/cci

Date