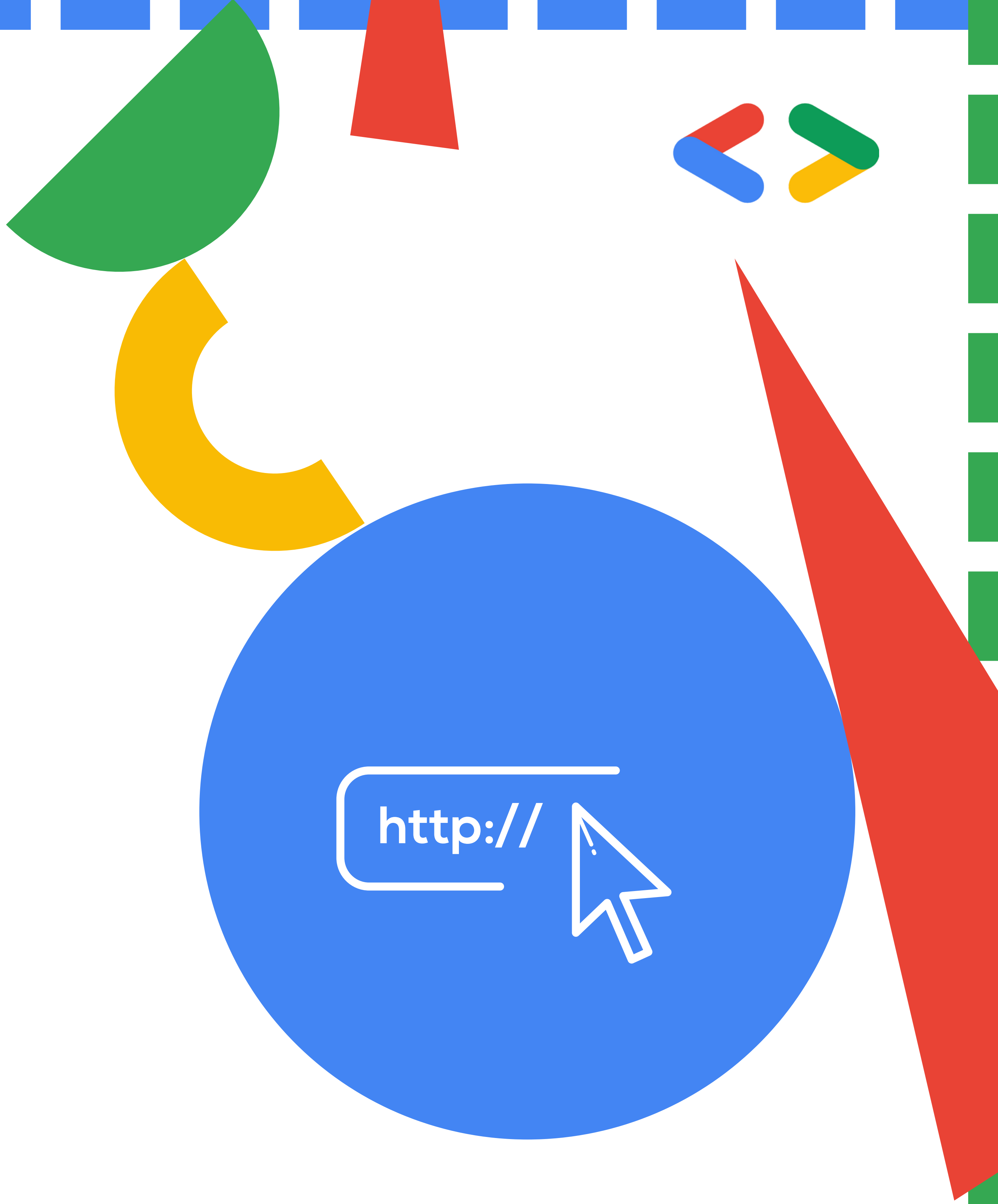
WELCOME
WEB DEVELOPMENT
WORKSHOP



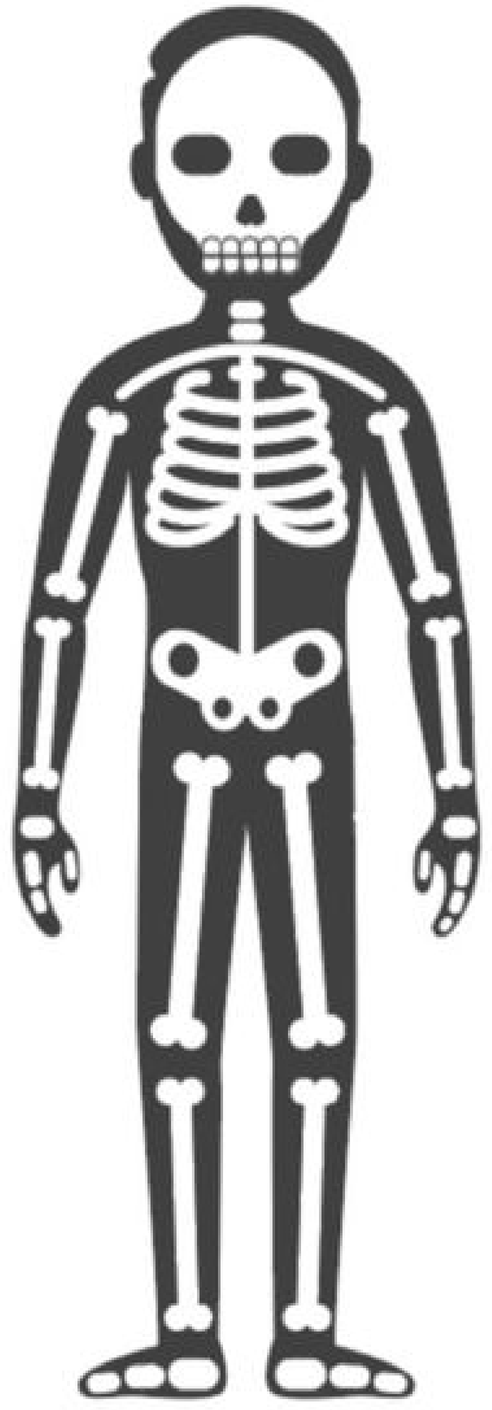
Google Developer Student Clubs
Maharishi Markandeshwar (Deemed to be University)

Our Learning Checklist

What we'll cover in this session

- Recap Session 1
- JAVASCRIPT
- ANGULARJS

HTML THE SKELETON



HTML Provides name (tags) to describe different types of content (elements) on your website.
for example <header>,<div>,<button>.

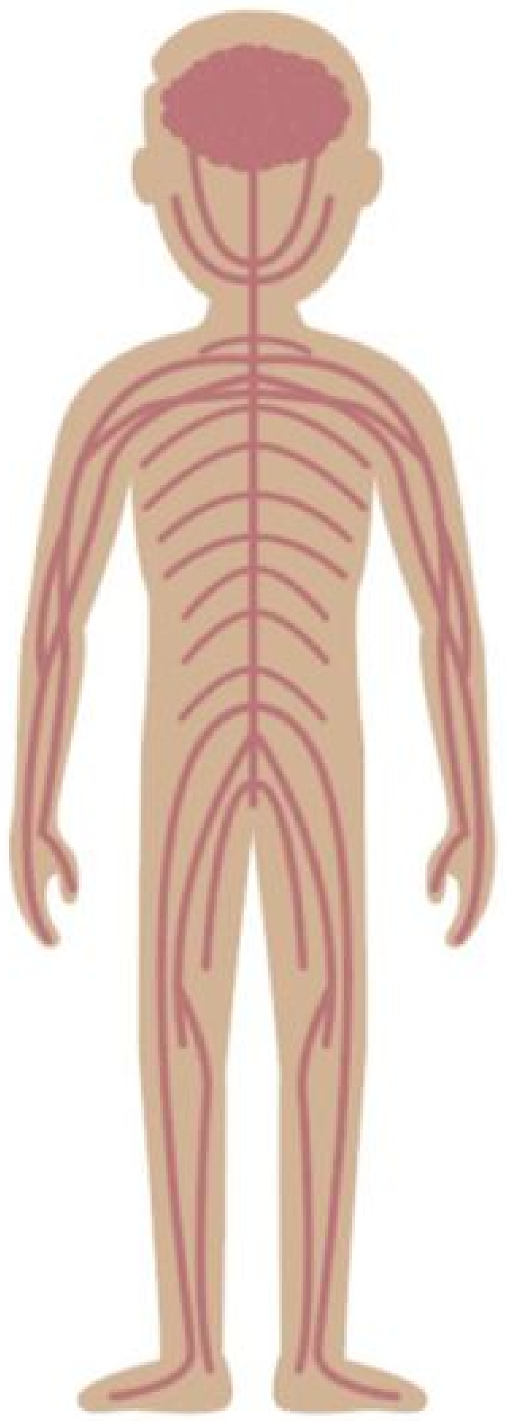
This allows your browser understand what it is reading and how to render it.While your browser can render HTML by itself,you can make it dynamic and beatiful using JS & CSS.
So think of HTML as a skeleton that comes to life with JS & CSS

CSS THE SKIN

CSS will make our site presentable and enhance the user experience for our site visitors.

CSS allows you to add colour, modify sizes, embed background images, and make your design responsive for desktop & mobile viewing alike.

JAVASCRIPT THE BRAIN



JS will provide the brain and muscles of the website allowing it to interact and re-render HTML content based on user input or data from a server.

This makes your HTML dynamic rather than static. Games and interactive application can be built upon HTML thanks to the power of JS.

JavaScript

Google Developer Student Clubs
Maharishi Markandeshwar (Deemed to be University)

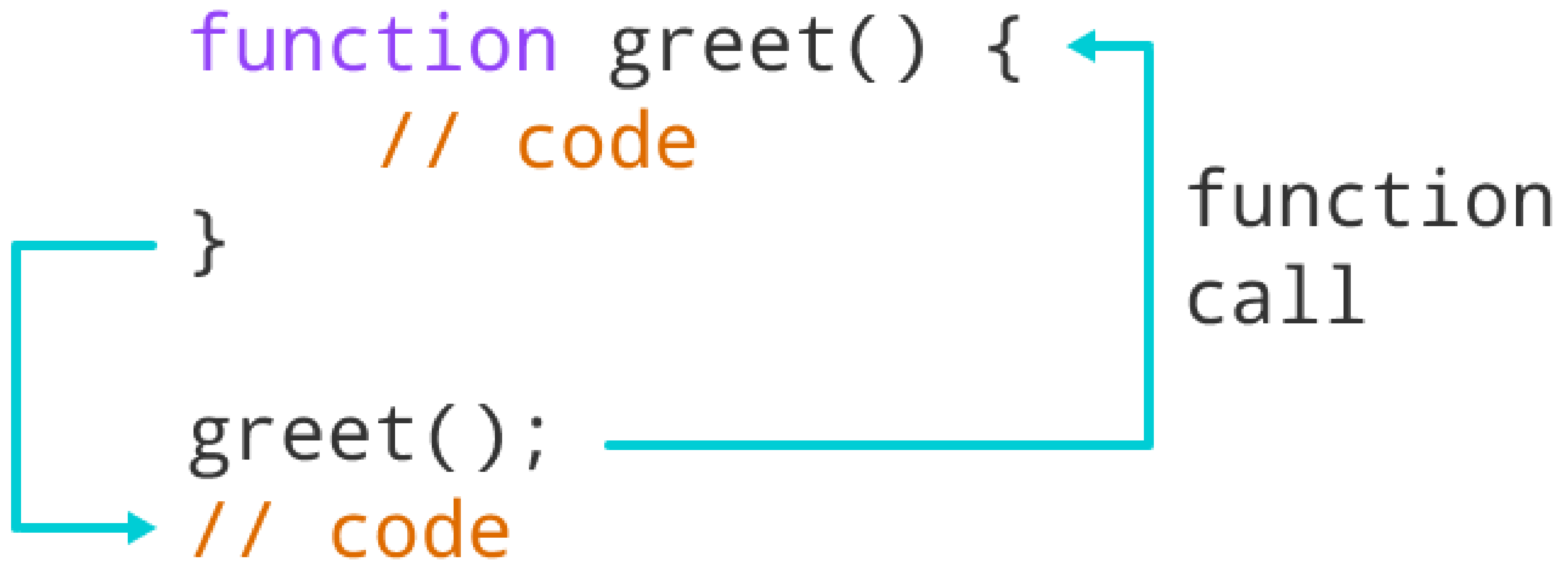
What is JS

JS(Java Script)

It is a lightweight programming language ("scripting language")

- used to make web pages interactive
- insert dynamic text into HTML (ex: user name)
- react to events (ex: page load user click)
- get information about a user's computer (ex: browser type)
- perform calculations on user's computer (ex: form validation)

Basic JS Sytax structure



```
function greet() {
    // code
}

greet();
// code
```

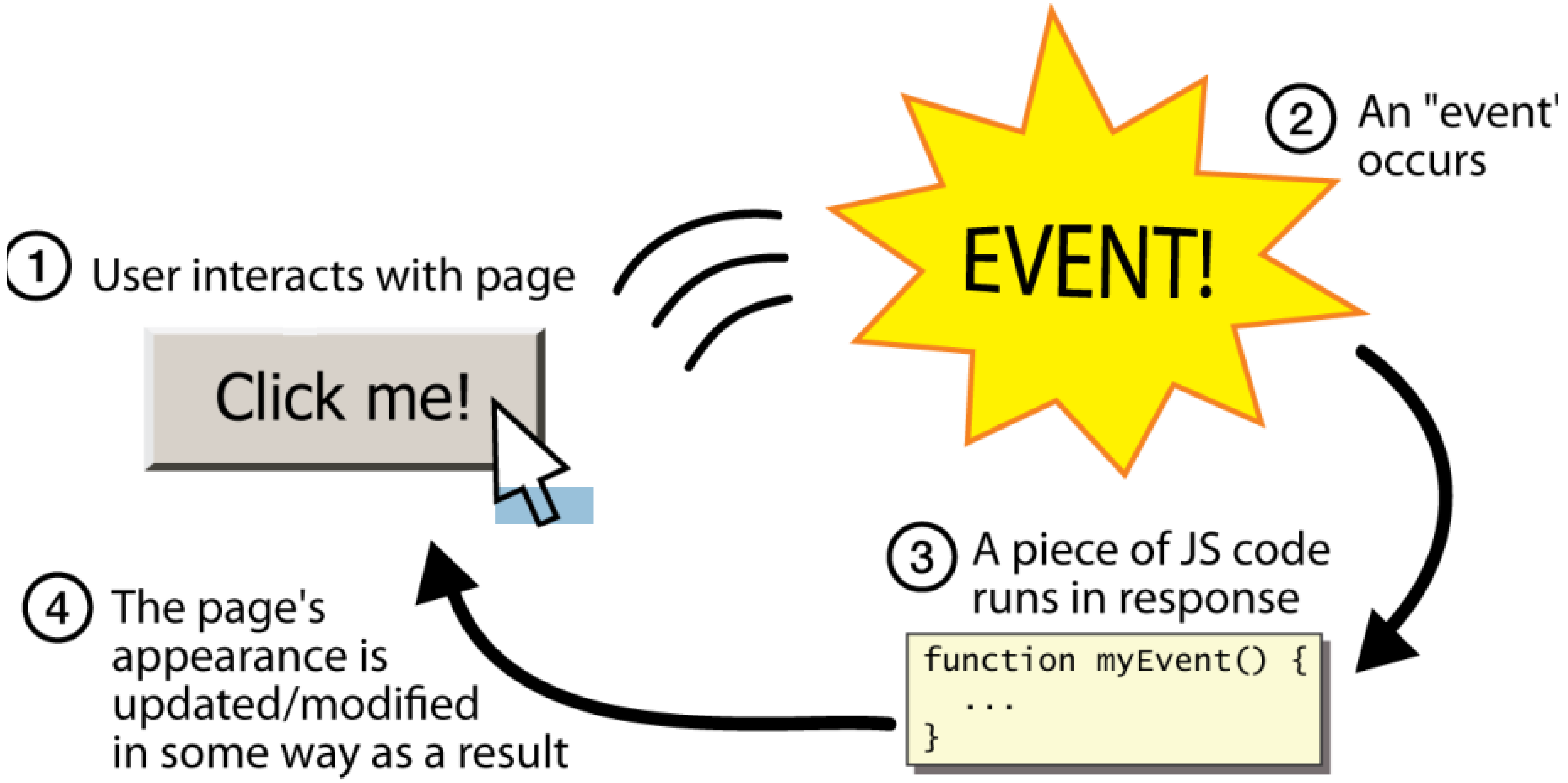
Saved in a index.**js** file

Linking to a JavaScript file

```
<script src="filename" type="text/javascript"></script>
```

- script tag should be placed in HTML page's head
- script code is stored in a separate .js file
- JS code can be placed directly in the HTML file's body or head (like CSS) but this is bad style (should separate content, presentation, and behavior

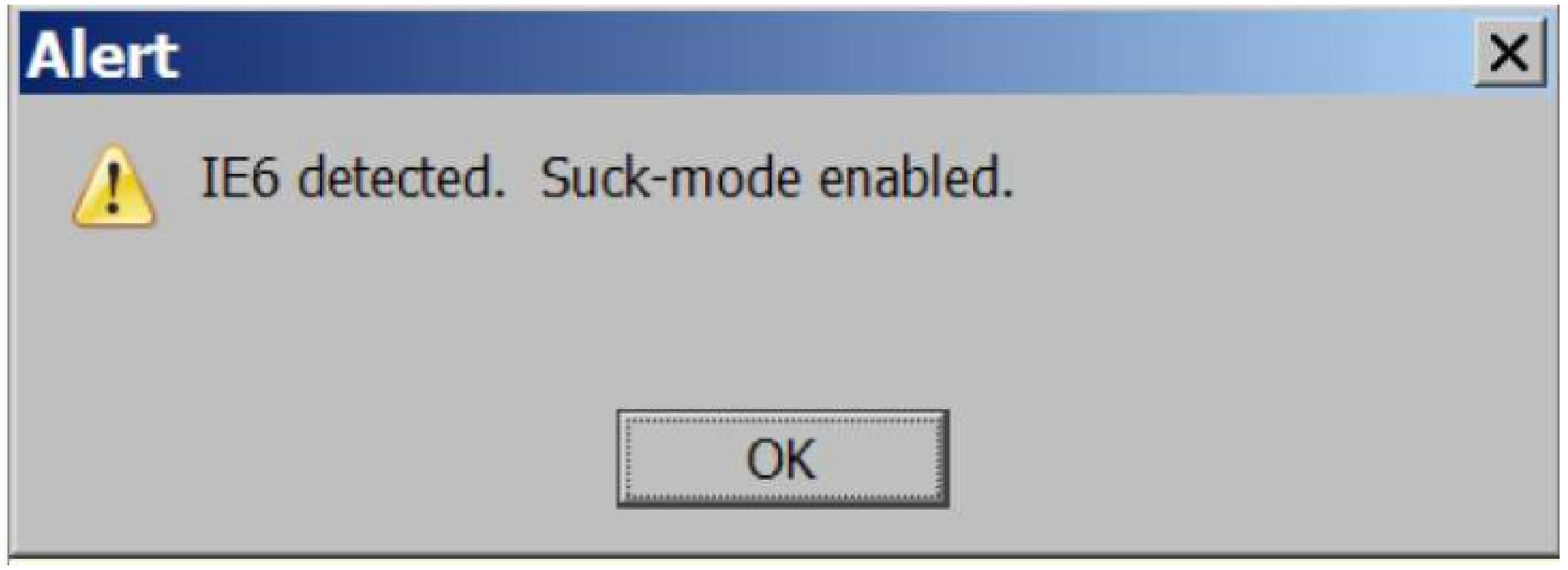
Event-driven programming



1 User interacts with page
Click me!
EVENT!
2 An "event' occurs
3 A piece of JS code runs in response
function myEvent() {
...
}
4 The page's appearance is updated/modified in some way as a result

A JavaScript statement:alert

```
alert("IE6 detected. Suck-mode enabled.");
```



IE6 detected. Suck-mode enabled.

It is a JS command that pops up a dialog box with a message

JavaScript functions

```
function name()
{ statement ;
statement ;
...
statement ;
}
```

```
function myFunction() {
alert("Hello!");
alert("How are you?");
}
```

statements placed into functions can be evaluated in response to user events

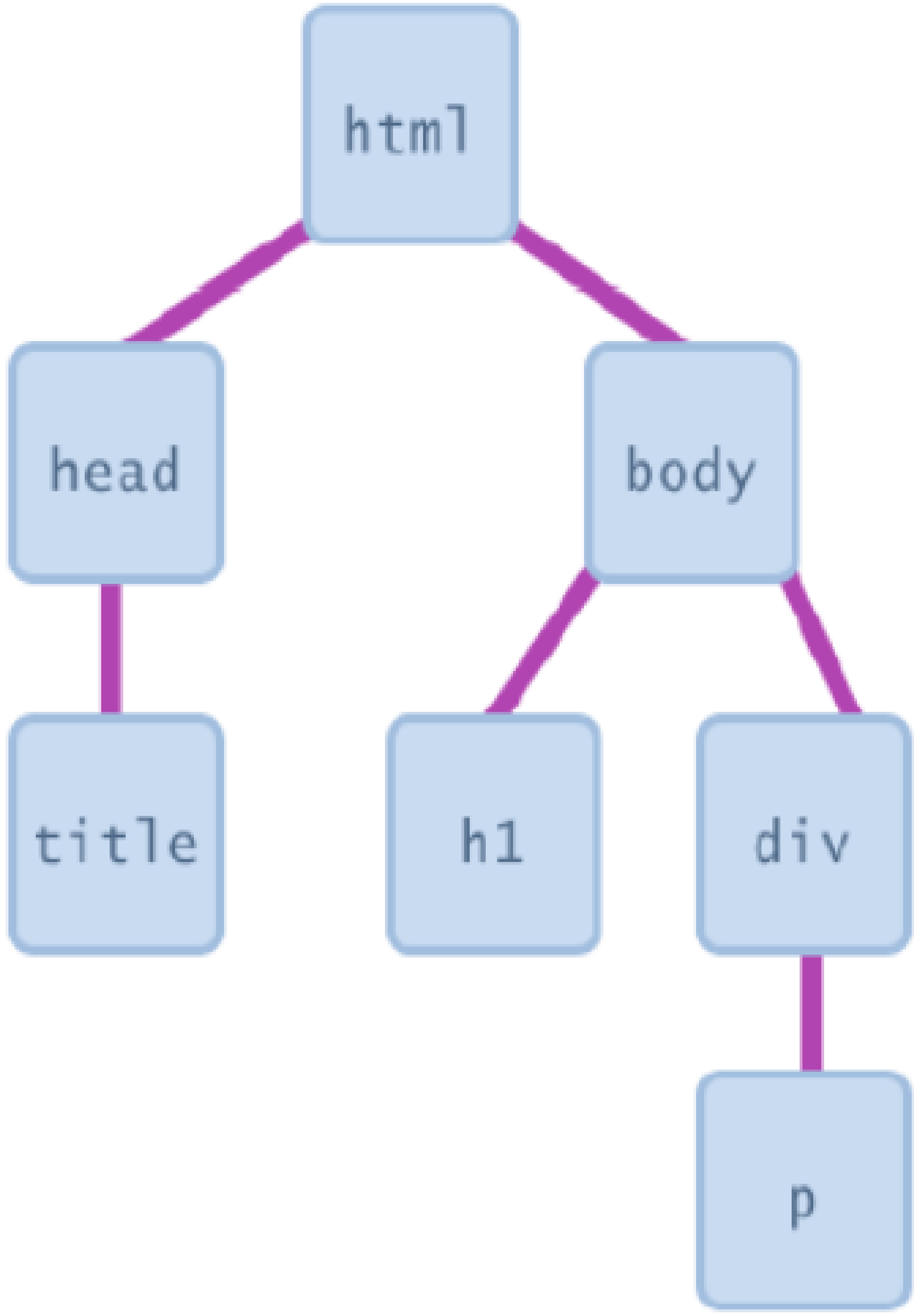
Event handlers

```
<button>Click me !</button>

<button onclick="myfunction();">Click me !</button>
```

- JavaScript functions can be set as event handlers
- when you interact with the element, the function will execute
- onclick is just one of many event HTML attributes we'll use

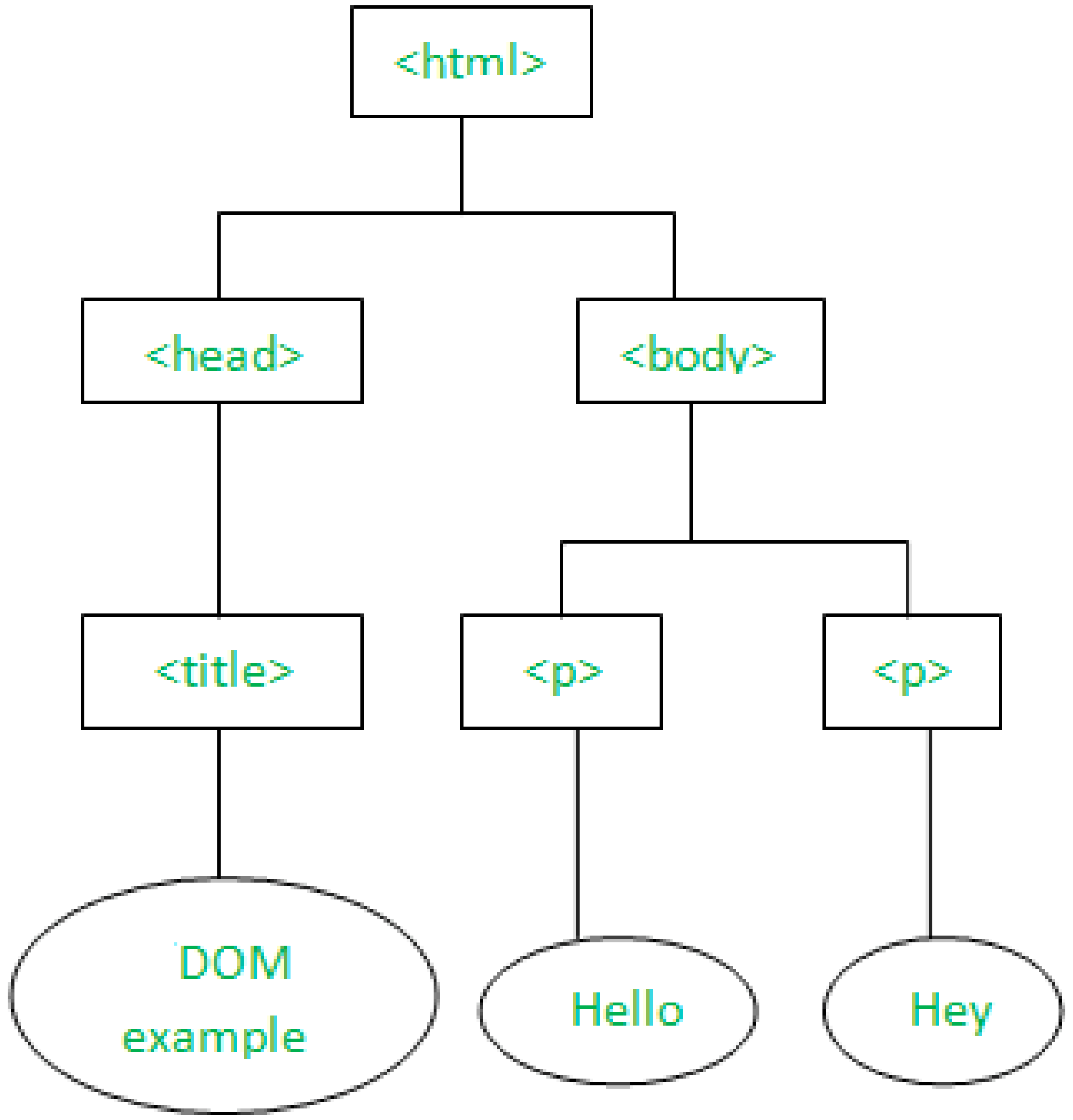
Document Object Model (DOM)

- most JS code manipulates elements on an HTML page
- we can examine elements' state
  - e.g. see whether a box is checked
- we can change state
  - e.g. insert some new text into a div
- we can change styles
  - e.g. make a paragraph red



Document Object Model (DOM)

```
<html>
<head>
<title>
 DOM example
</title>
</head>
<body>
<p id = "para1">Hello</p>
<p id =  “para2”>Hey</p>
</body>
</html>
```



Accessing elements: document.getElementById

```
var name = document.getElementById("id");
                                                    JS
```

```
<button onclick="changeText();">Click me!</button>
<span id="output">replace me</span>
<input id="textbox" type="text" />                HTML
```

```
function changeText() {
       var span = document.getElementById("output");
       var textBox = document.getElementById("textbox");

        textbox.style.color = "red";

}                                                   JS
```

Accessing elements: document.getElementById

- **document.getElementById** returns the DOM object for an element with a given id
- can change the text inside most elements by setting the innerHTML property
- can change the text in form controls by setting the value property

Changing element style: element.style

Attribute	Property or style object
color	color
padding	padding
background-color	backgroundColor
border-top-width	borderTopWidth
Font size	fontSize
Font famiy	fontFamily

AngularJS

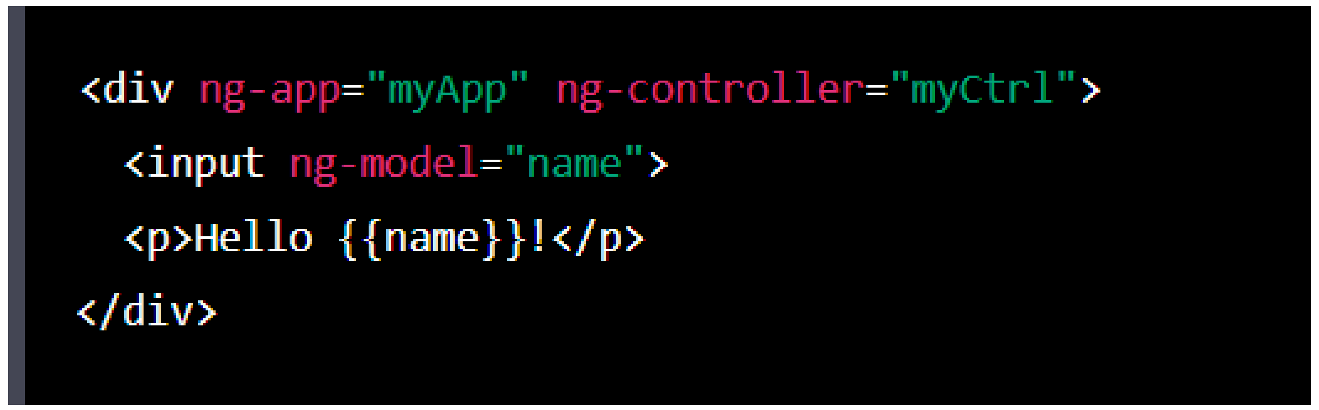
ANGULAR

Google Developer Student Clubs
Maharishi Markandeshwar (Deemed to be University)

What is AngularJS

AngularJS

AngularJS is a front-end web application framework that allows developers to create dynamic web apps with reusable components and two-way data binding. Developed by Google, it makes web development more efficient and structured.



```
<div ng-app="myApp" ng-controller="myCtrl">
  <input ng-model="name">
  <p>Hello {{name}}!</p>
</div>
```

Why AngularJS?

- Write less Javascript Code
- In place DOM Manipulation
- In place Service Providers
- Good Unit testing support
- Supports better routing
- Good Support for Animation

Data binding

Binding in AngularJS (1.x) is a way to link the data in the model with the view (HTML) in such a way that changes in the model are automatically reflected in the view, and changes in the view are automatically reflected in the model.

Two-way data binding and expressions

Two-way data binding is a feature in AngularJS that allows changes made in the view to automatically update the model, and vice versa. This is achieved by using a combination of AngularJS expressions and directives, such as **ng-model** and **ng-bind.**

AngularJS Modules

Modules in AngularJS are used to organize the code and define reusable components, such as services, directives, and controllers. A module can be defined using the **angular.module** function, and it can contain one or more components.

```
var app = angular.module("myApp", []);
```

AngularJS Controllers

Controllers are JavaScript functions that are used to control the behavior of the view. They are defined using the **.controller** method on a module, and they can manipulate the scope, which is an object that represents the model in the view. The scope can be accessed and manipulated using AngularJS expressions, directives, and filters.

```
app.controller("myCtrl", function($scope)
```

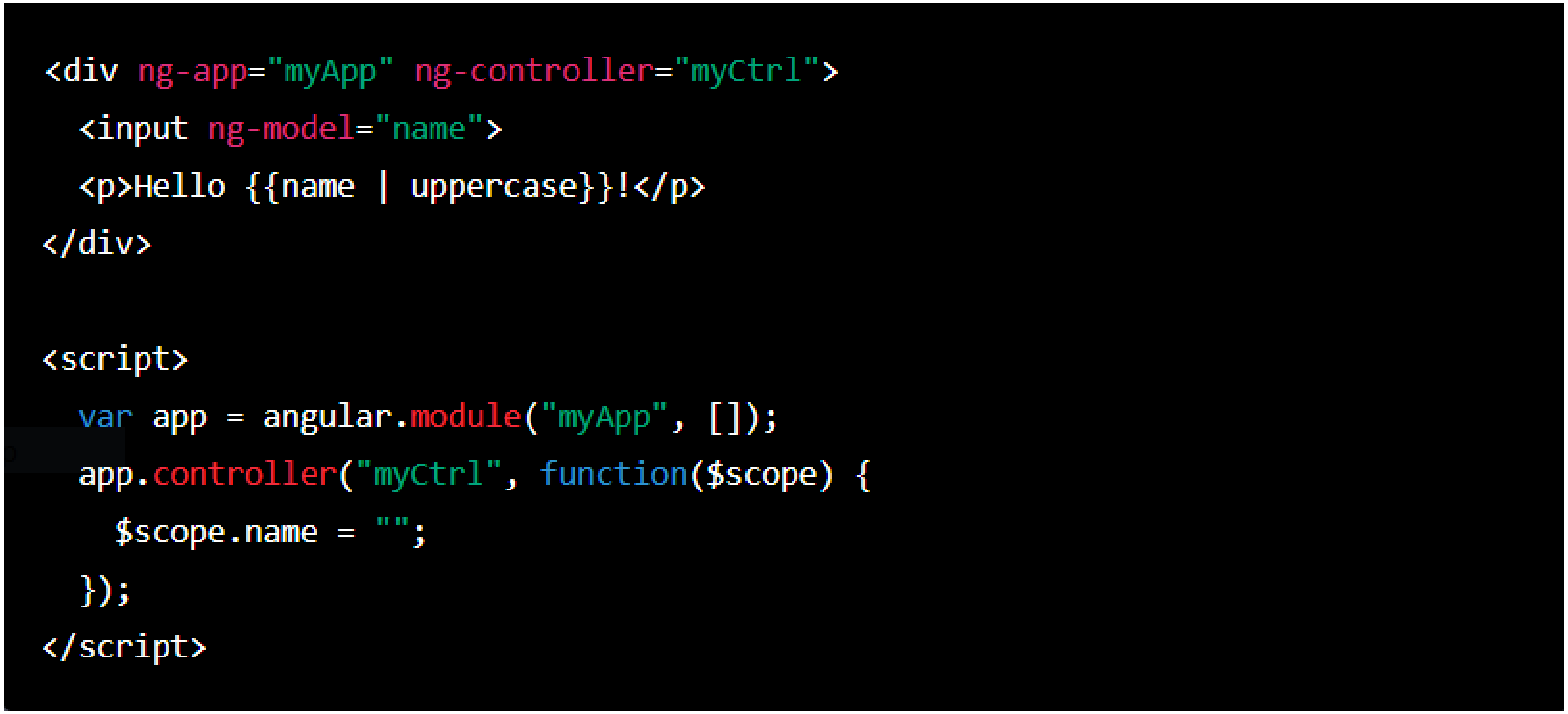
Two-way data binding and expressions

```
<div ng-app="myApp" ng-controller="myCtrl">
  <input ng-model="name">
  <p>Hello {{name}}!</p>
</div>


<script>
  var app = angular.module("myApp", []);
  app.controller("myCtrl", function($scope) {
    $scope.name = "";
  });
</script>
```

AngularJS Filters

Filters are used to format the value of an expression before it is displayed to the user. Filters can be used to format data in AngularJS expressions, as well as in directive bindings. Some common filters in AngularJS include **uppercase**, **lowercase**, **currency**, **number**, and **date**.

AngularJS Filters



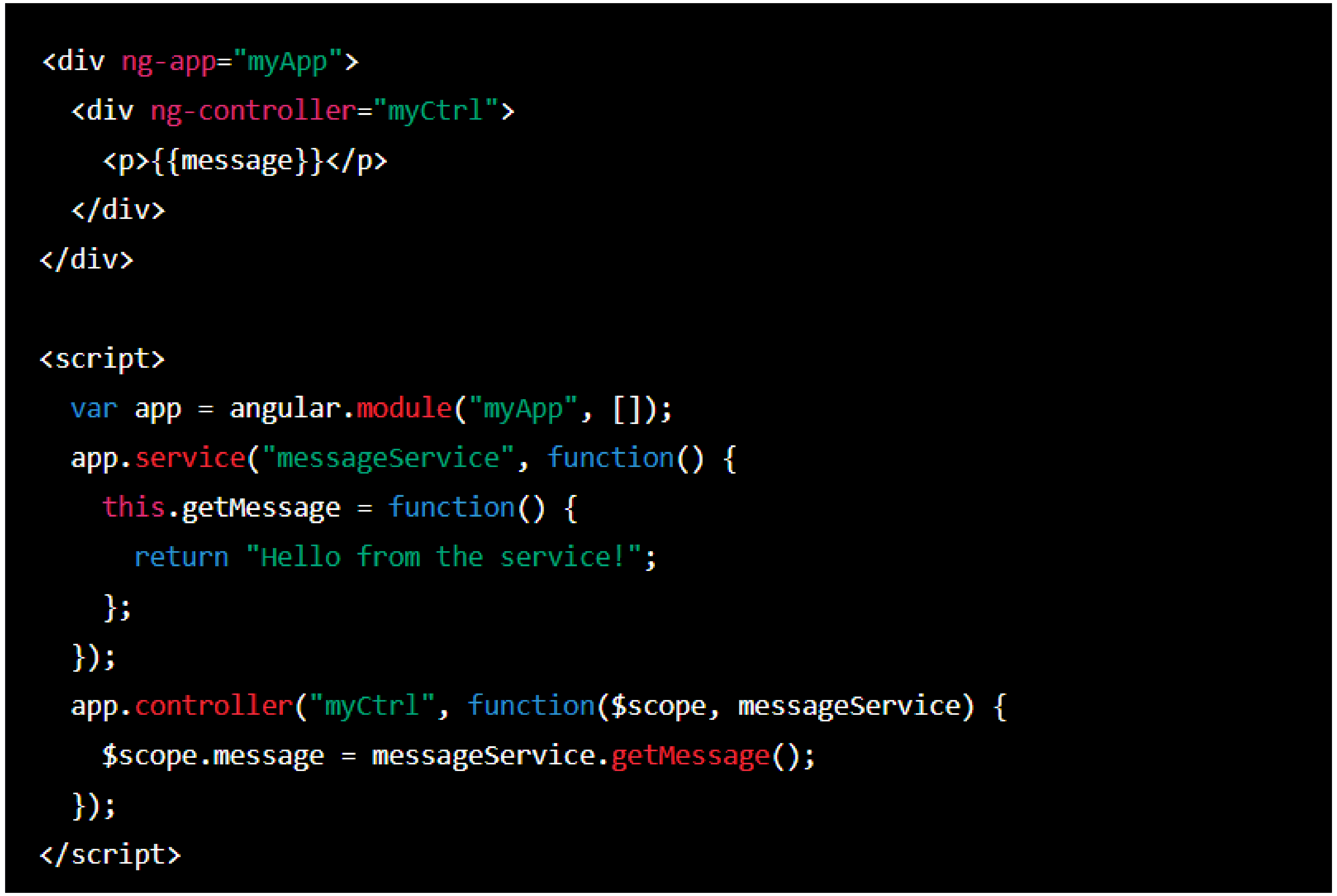
```
<div ng-app="myApp" ng-controller="myCtrl">
  <input ng-model="name">
  <p>Hello {{name | uppercase}}!</p>
</div>

<script>
  var app = angular.module("myApp", []);
  app.controller("myCtrl", function($scope) {
    $scope.name = "";
  });
</script>
```

Dependency Injection and Services

Dependency Injection (DI) is a technique in AngularJS to manage component dependencies by injecting them at runtime.

Services are singleton objects that provide data and functionality to components and can be easily injected using DI. This allows for writing modular and testable code.

Dependency Injection and Services



```
<div ng-app="myApp">
  <div ng-controller="myCtrl">
    <p>{{message}}</p>
  </div>
</div>

<script>
  var app = angular.module("myApp", []);
  app.service("messageService", function() {
    this.getMessage = function() {
      return "Hello from the service!";
    };
  });
  app.controller("myCtrl", function($scope, messageService) {
    $scope.message = messageService.getMessage();
  });
</script>
```

Routing and navigation

Routing is a feature in AngularJS that enables navigation between different views in a single-page application. The **ngRoute** module provides the routing functionality, and it allows you to define different routes, each with its own URL and view.

```
var app = angular.module("myApp", ["ngRoute"]);
```

Routing and navigation

```
<div ng-app="myApp">
  <p><a href="#/view1">View 1</a></p>
  <p><a href="#/view2">View 2</a></p>
  <div ng-view></div>
</div>

<script>
  var app = angular.module("myApp", ["ngRoute"]);
  app.config(function($routeProvider) {
    $routeProvider
      .when("/view1", {
        template: "<h1>View 1</h1>"
      })
      .when("/view2", {
        template: "<h1>View 2</h1>"
      });
  });
</script>
```

Any doubt Regarding Web Development Workshop

Thank you for attending our workshop. Your participation is appreciated.