



CHAPTER 9 – INTRODUCTION TO JAVASCRIPT



Contents of the slides is taken from WD3304 IWEF by HSL, 2019

TOPICS TO BE COVERED

» **Introduction**

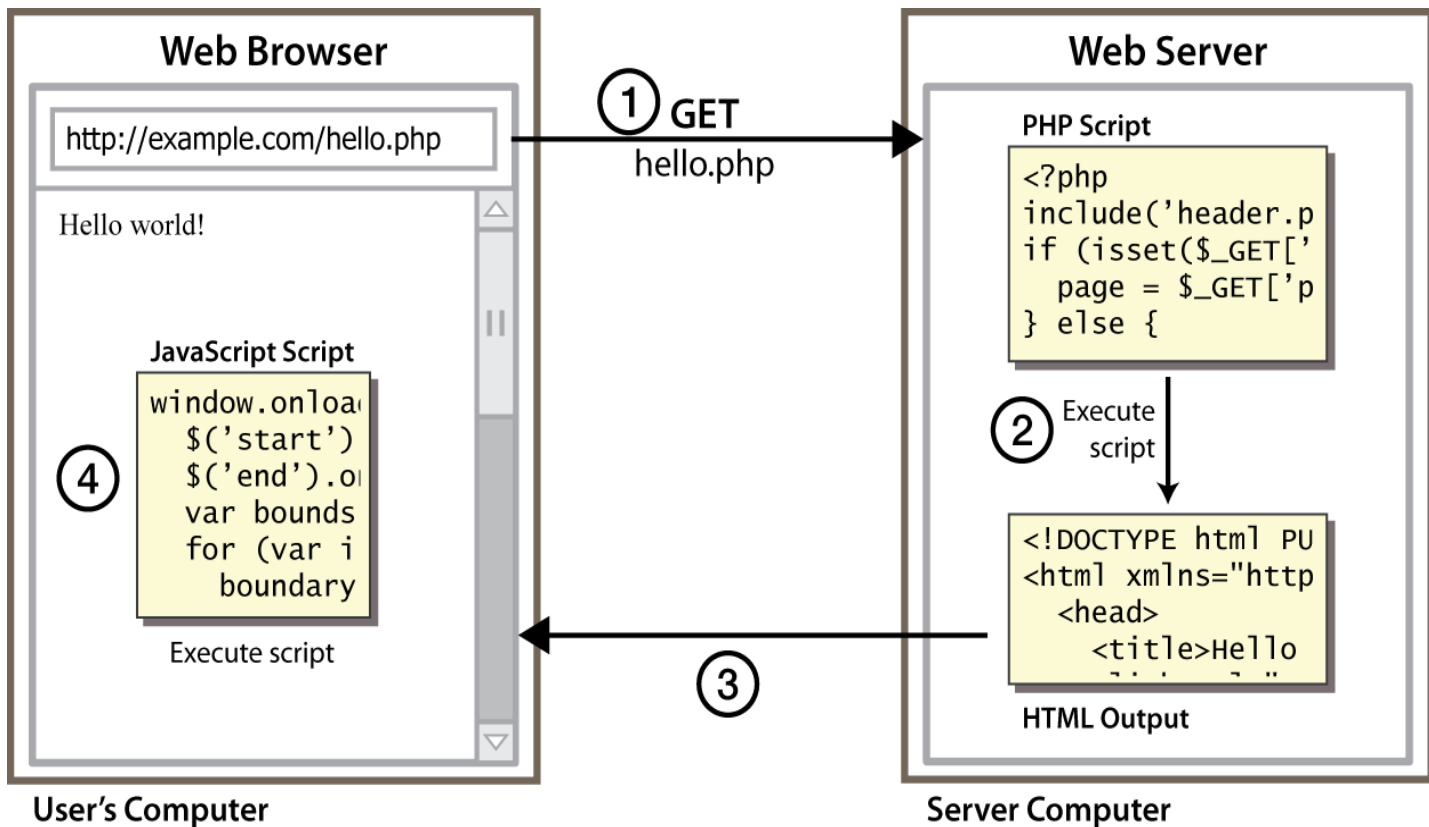
What is JS and what isn't

Inserting the JS

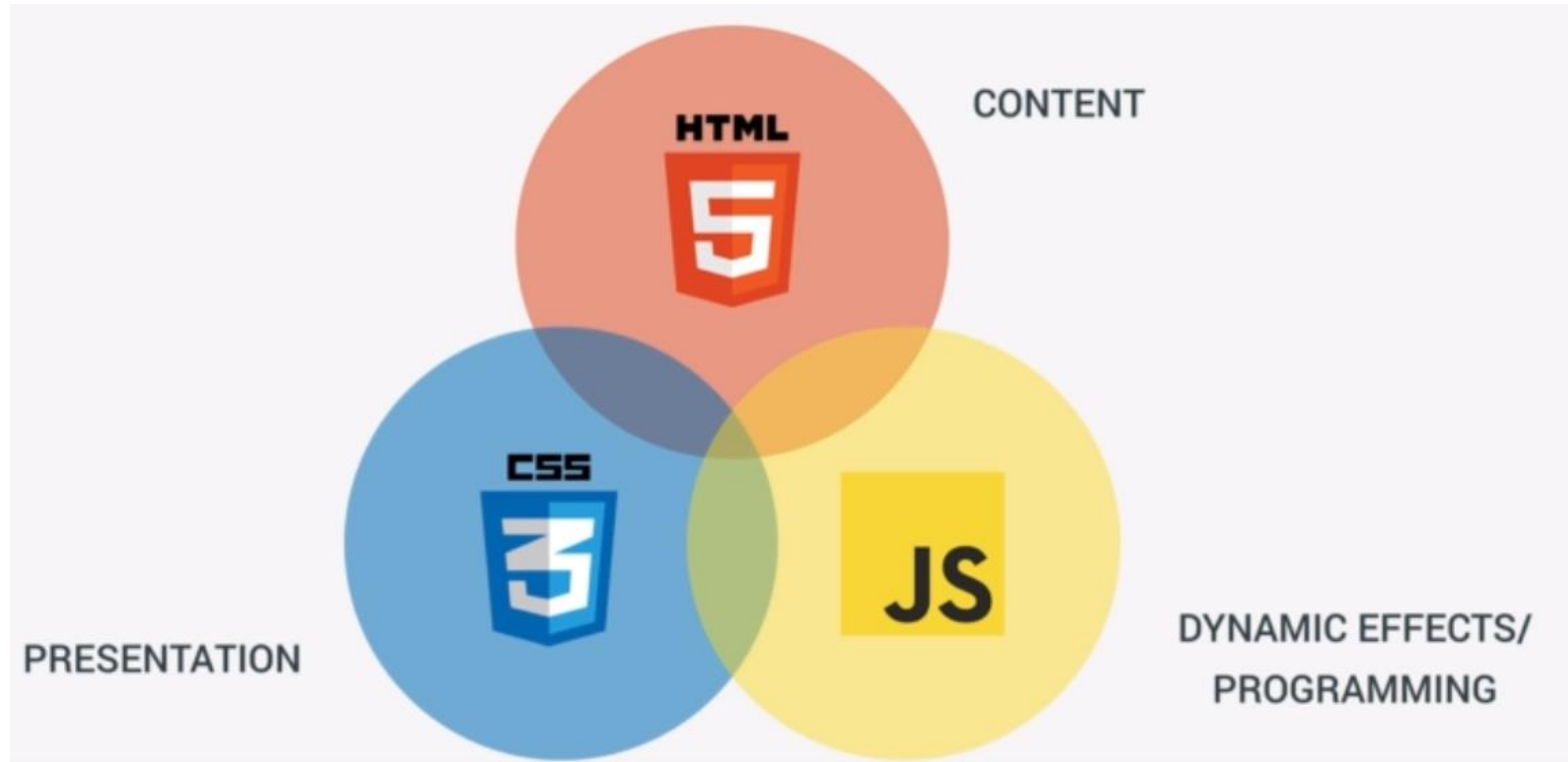
Variables



Client-Side Scripting



Separation of Concerns



In Conclusion



CONTENT

NOUNS

```
<p></p>
```

means "paragraph"



PRESENTATION

ADJECTIVES

```
p {color: red;}
```

means "the paragraph
text is red"



DYNAMIC EFFECTS/
PROGRAMMING

VERBS

```
p.hide();
```

means "hide the
paragraph"

So...JavaScript?

- 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)

So...JavaScript?

- JavaScript is what made modern web development possible:
 - Dynamic effects and interactivity
 - Modern web application that we can interact with
- Frameworks/libraries like jQuery, React and Angular are 100% based on JavaScript, you need to master JavaScript in order to use them! We will be covering Angular Basics in this module too.

Inserting JavaScript

- Three common ways of inserting JavaScript code in a webpage:
 - Inside an HTML tag script
 - In an external .JS file
 - As a value of some HTML attributes

JavaScript in a Tag Script

- Can be embedded between the <body></body> tag.
- Need to inform browser if it's a JavaScript code.
- If embedding code in web page, the code must be enclosed between the <script></script> tag.

```
<script type="text/javascript">  
    alert("JavaScript is working in your browser!");  
</script>
```

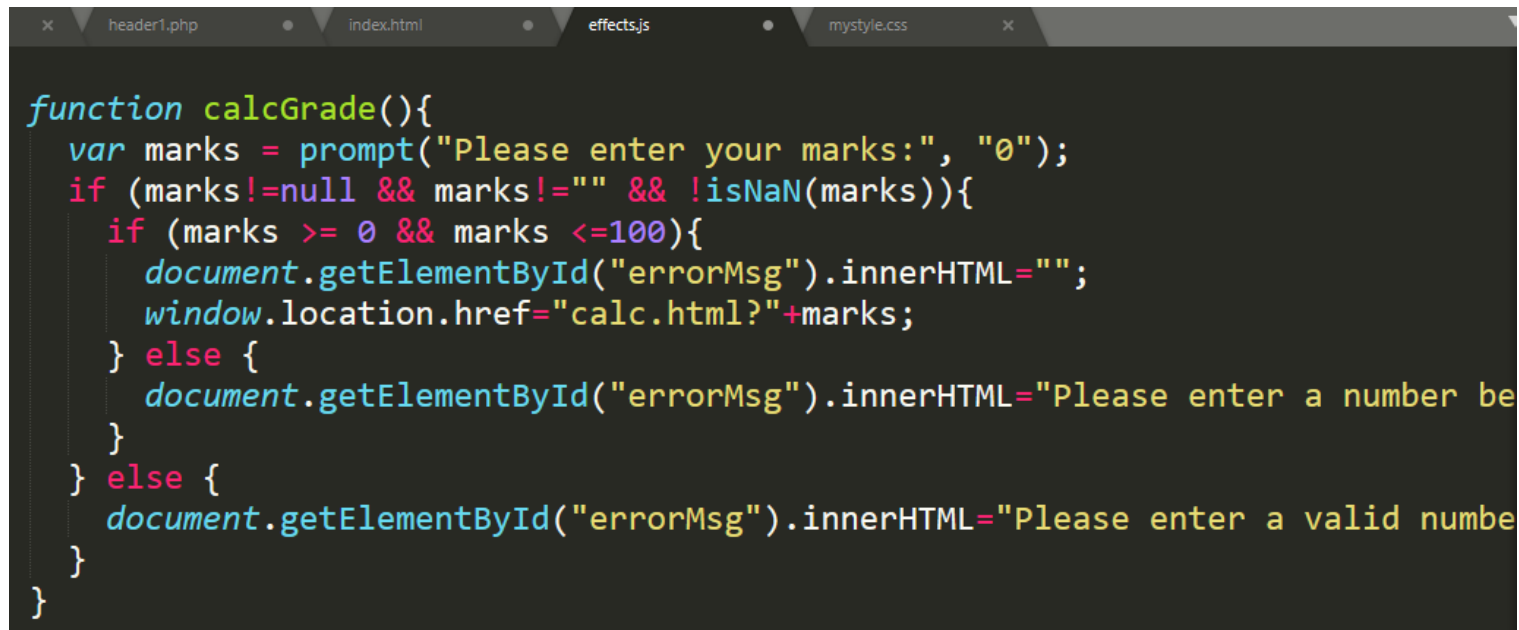
JavaScript in an External file

- Inside an .js file, linked in an HTML page.
- The external file will just include JS code. NO HTML!!!

```
<head>
  <title>Simplicity: A simple image gallery.</title>
  <link rel="stylesheet" href="layout/style.css" type="text/css" />
  <link rel="stylesheet" href="css/lightbox.css" type="text/css" media="
screen"/>
  <script type="text/javascript" src="js/prototype.js"></script>
  <script type="text/javascript" src="js/
scriptaculous.js?load=effects,builder"></script>
  <script type="text/javascript" src="js/lightbox.js"></script>
</head>
```

JavaScript in an External file

- How the insides of a .js looks like (without script tag):

A screenshot of a code editor with a dark background. The editor has several tabs at the top: 'header1.php', 'index.html', 'effects.js' (which is the active tab), and 'mystyle.css'. The code in the 'effects.js' tab is a JavaScript function named 'calcGrade'. The code is color-coded: 'function' is green, 'calcGrade()' is green, 'var' is blue, 'marks' is blue, '=' is blue, 'prompt' is blue, 'Please enter your marks:' is yellow, '0' is yellow, ';' is blue, 'if' is blue, 'marks!=null' is blue, '&&' is blue, 'marks!=""' is blue, '&&' is blue, '!isNaN' is blue, '(marks)' is blue, '{' is blue, 'if' is blue, 'marks >= 0' is blue, '&&' is blue, 'marks <=100' is blue, '{' is blue, 'document.getElementById' is blue, 'errorMsg' is blue, 'innerHTML=""' is blue, ';' is blue, 'window.location.href' is blue, 'calc.html?' is blue, 'marks' is blue, ';' is blue, '}' is blue, 'else {' is blue, 'document.getElementById' is blue, 'errorMsg' is blue, 'innerHTML="Please enter a number be' is blue, '}' is blue, '}' is blue, 'else {' is blue, 'document.getElementById' is blue, 'errorMsg' is blue, 'innerHTML="Please enter a valid numbe' is blue, '}' is blue, and the final closing brace '}' is blue. The code is as follows:

```
function calcGrade(){
  var marks = prompt("Please enter your marks:", "0");
  if (marks!=null && marks!="" && !isNaN(marks)){
    if (marks >= 0 && marks <=100){
      document.getElementById("errorMsg").innerHTML="";
      window.location.href="calc.html?" + marks;
    } else {
      document.getElementById("errorMsg").innerHTML="Please enter a number be
    }
  } else {
    document.getElementById("errorMsg").innerHTML="Please enter a valid numbe
  }
}
```

As a value of HTML attribute

```
<p>Click <a href="javascript:alert('JavaScript is working in your browser!')">here</a>  
to alert a message.</p>
```

Hello World!

- Create a new helloworld.html file.
- Include the following code in the file:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Hello World</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("<h1>Hello World! This is written using JS!</h1>");
    </script>
  </body>
</html>
```

Now for some fun JS structure!

JS Document Structure

- JavaScript statement is terminated by a semicolon (;)

```
var x = 1;  
var y = 2;  
document.write("<h1>Hello World! This is written using JS!</h1>");
```

- JavaScript is **CASE-SENSITIVE**, meaning:
 - Name of keywords + other language identifiers
 - Must be typed with the correct **CAPITALISM**.
 - Add != ADD != add != aDD (or w/e)

Variables

- are handy because they allow you to carry things in them which we call as data.
- Acts as a container that stores a value which can be used again and again instead of writing it repetitively.
- For example, a program created to calculate and print a credit card bill would deal with:
 - Amount that is charged to the card:
 - `var amount_charged = 1000;`
 - `amount_charged` is the variable and 1000 is the data.

Variables

- Here are a few rules you have to follow when it comes to naming variables:
 - The first character must be a letter or an underscore (_). You can't use a number as the first character.
 - As with the rest of JavaScript, variable names are case sensitive. That is, a variable named myName is treated as an entirely different variable than one named my_Name or myname.
 - You can't use one of JavaScript's reserved words as a variable name. All programming languages have a supply of words that are used internally by the language and that can't be used for variable names because doing so would cause confusion (or worse). Note, too, that JavaScript also has many keywords that should be avoided as well, like **this**.

Variables

```
<script type="text/javascript">  
  var my_var = "Susan";  
  var another_var, yet_another_var;  
  
  another_var = "Jason";  
  yet_another_var = "Tyson";  
  
  document.write(another_var);  
</script>
```

Variables

```
<script type="text/javascript">  
    var MYVAR,  
        myvar,  
        myVar,  
        MyVar,  
        MyVaR;  
  
    MYVAR = "Lisa";  
    MyVaR = "Joseph";  
  
    document.write(MyVaR);  
</script>
```

Variable Scopes

- There are two types of scope:
 - Local

```
<script type="text/javascript">
  function myFunc(){
    var my_var = "Bruce";

  }

  document.write("my_var= " + my_var); //undefined
</script>
```

- A variable declared locally inside a method cannot be called globally.

Variable Scopes

- There are two types of scope:
 - Local
 - Instead, this should be done in this manner as the variable can only be called locally:

```
<script type="text/javascript">
    function myFunc(){
        var my_var = "Bruce";
        document.write("my_var= " + my_var);
    }

    myFunc();
</script>
```

Variable Scopes

- There are two types of scope:
 - Global

```
<body>
  <script type="text/javascript">

    var x = 2;

    function myFunc(){
      var y = 3;
      var total = x + y;
      document.write("Total= " + total);
    }

    myFunc();

  </script>
</body>
```

Variable Keywords

break	else	new	var
case	finally	return	void
catch	for	switch	while
continue	function	this	with
default	if	throw	
delete	in	try	
do	instanceof	typeof	

Variable Reserved Words

abstract	enum	int	short
boolean	export	interface	static
byte	extends	long	super
char	final	native	synchronized
class	float	package	throws
const	goto	private	transient
debugger	implements	protected	volatile
double	import	public	

Data Types

- JavaScript is loosely typed language.
 - This means unlike Java, you don't need to specify variable data types
 - The data type will be determined automatically during run-time by JavaScript

Data Types

- The main ones are:
 - **Number**
 - `var x = 0.01; var y = 1; var z = -0.1;`
 - Just one Number type for integers and floats.
 - FYI, NaN is considered of type Number as well.
 - **String**
 - `var x = "Hello World!"; var y = 'Yes mam.';`
 - Strings of text in quotation marks. Values can be either enclosed in double quotes or single quotes.
 - Use backslash to escape special characters:
 - "Hi, this is \"special\" word here."

Data Types

- The main ones are:
 - **Boolean**
 - `var x = true; var y = false;`
 - True (1) or false (0)
 - **Null**
 - Special value which exists that indicates a null (empty) value has been defined.
 - **Undefined**
 - Value has not been declared and it does not exist.

Data Types

- Other data types include:

- **Object**

- `const dog = { name: "Haru", age = "3", food: 'egg' };`

- **Array**

- `var array1 = [1, 2, 3]; var array2 = ["pear", "apple"];`

Type Coercion

- JavaScript automatically converts our variables into appropriate data types.

```
var firstName = "John";  
var age = 23;  
  
document.write(firstName + " " + age);
```

- Age was originally of type Number but when concatenated with two other strings, it gets converted into a string so they can form a larger string of characters.

Type Coercion

- Let's try something different now.

```
var firstName = "John";  
var age = 23;  
  
var job, isMarried;  
job = "teacher";  
isMarried = false;  
  
document.write(firstName + " is a " + age + " year old " + job + ". Is he married? " +  
    isMarried);
```

- Even a Boolean gets converted into a string, so JS also does type coercions with Booleans.

Variable Mutation

- Changing the values of a variable.
- If the var keyword has been declared for a variable, there is no need to declare it again. For e.g. let's try changing the age:

```
var firstName = "John";  
var age = 23;  
  
var job, isMarried;  
job = "teacher";  
isMarried = false;  
  
age = "twenty eight";  
job = "driver";  
  
document.write(firstName + " is a " + age + " year old " + job + ". Is he married? " +  
    isMarried);
```

Variable Mutation

- Instead of writing it to the browser, we can also display it using an alert box.

```
var firstName = "John";  
var age = 23;  
  
var job, isMarried;  
job = "teacher";  
isMarried = false;  
  
age = "twenty eight";  
job = "driver";  
  
alert(firstName + " is a " + age + " year old " + job + ". Is he married? " + isMarried)
```


Variable Mutation

- A prompt is a open dialog in the form of an alert box that accepts user input.
- A prompt can also be triggered and the answer inputted by the user can then be stored in a variable.

```
var firstName = "John";  
var age = 23;  
  
var job, isMarried;  
job = "teacher";  
isMarried = false;  
  
age = "twenty eight";  
job = "driver";  
  
alert(firstName + " is a " + age + " year old " + job + ". Is he married? " + isMarried);  
  
var lastName = prompt("What is his last name?");  
document.write(firstName + " " + lastName);
```

Using Quotes

- `a = "3 + 4"` is not the same as `a = 3 + 4`
- Enclosing it in speech marks makes it a string – it's not evaluated.

Comparison Operators

- A set of special characters that evaluate and compare values in different ways.

Operator	Meaning
==	Is equal to
!=	Is NOT equal to
===	Is identical to (equal to and of the same data type)
!==	Is NOT identical to
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

Comparison Operators

- The goal of comparing values is to obtain the result of either **true** or **false**.
- JavaScript evaluates the statement and gives us back a Boolean value, depending on the statement whether it is true or not.

Example:

```
alert (5 = 5); //This will alert "true"
```

```
alert("5" = 5);
```

```
alert ("5" != 5);
```

If/else Statements

- If/else statements are how we get JavaScript to ask itself a true/false question.

```
if(true){  
    //Do something  
}else {  
    //Do something  
}
```

- It tells the browser, is this condition is met, then execute the commands listed between the curly braces ({ }).

```
if( 1 = 2){  
    alert("There is something wrong with the equation");  
else {  
    alert("You are right. It is not the same");  
}
```

Functions

- A code that doesn't run until it is referenced or called.
- There are TWO types:
 - Native JavaScript functions (out of the box functions)
 - Custom functions (those you make up yourself).

Native Functions

- **Alert(), confirm(), prompt()**
 - These functions trigger browser-level dialog boxes.
- **Date()**
 - Returns the current date and time.
- **parseInt("123")**
 - Take a string data type containing numbers and turn it into a number data type.
- **setTimeout(functionName, 5000)**
 - Will execute a function after a delay.

Custom Functions

- To create a custom function, we type the *function* keyword followed by a name for the function, like so:

```
function name(){  
    //What you want to execute.  
}
```


Create a script asking for the first and the last name of a person when the page is loaded.

Use prompt and store both first and last name in two variables.

Then, welcome the user with an alert that says “Welcome to my site, Frank Miller!”, where Frank and Miller are the values you entered in the prompts.

**DO
TOGETHER!**