

# CI6206

# Internet Programming



Lecture

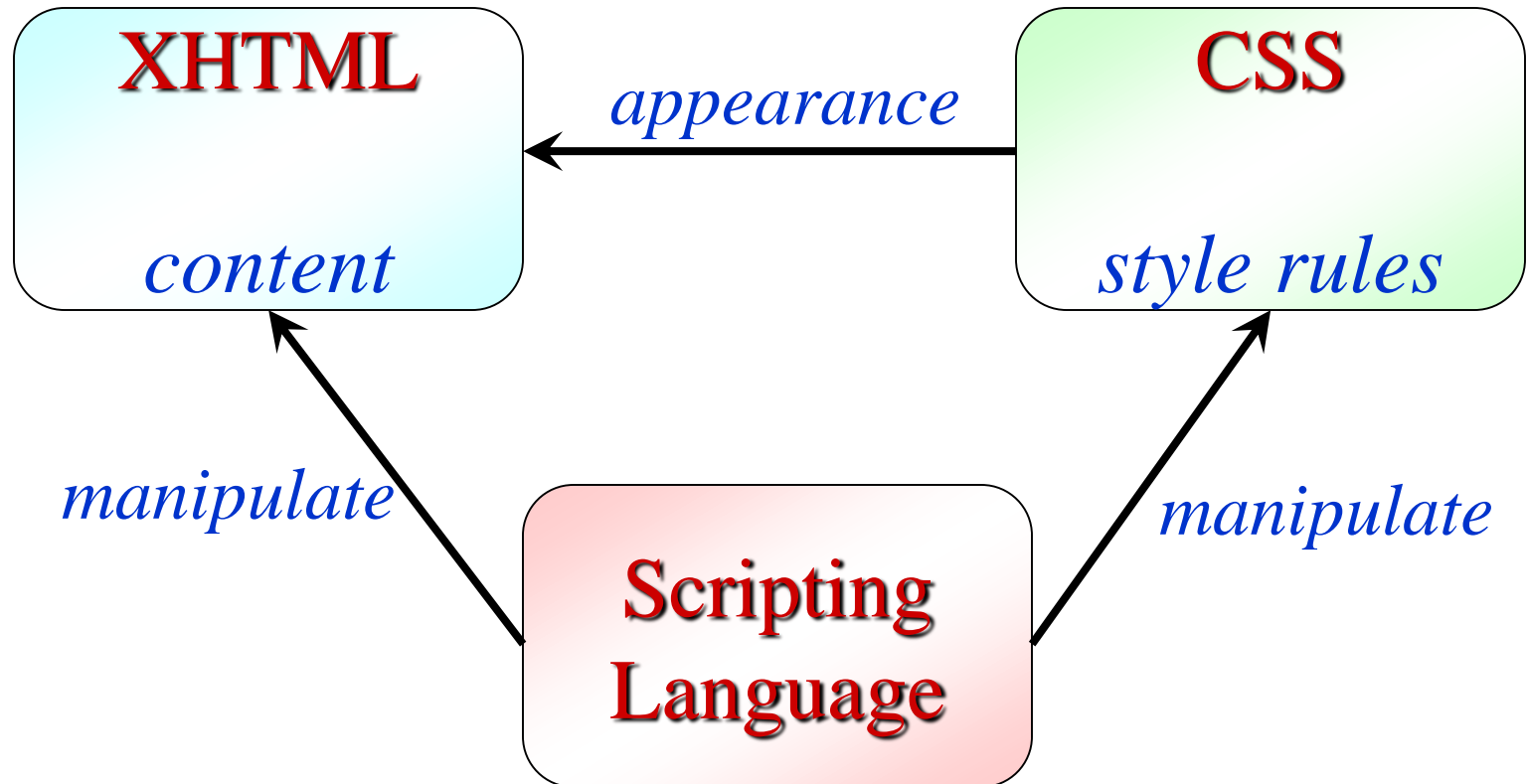
Client-side Web Development:  
Javascript

# Learning Objective

---

- ❑ HTML vs DOM
- ❑ What is JavaScript ?
- ❑ How to use JavaScript ?
  - Where to place it?
  - Syntax
  - What are available in JavaScript?
- ❑ Focus at
  - ready made boxes: alert, prompt, confirm
  - variable
  - string

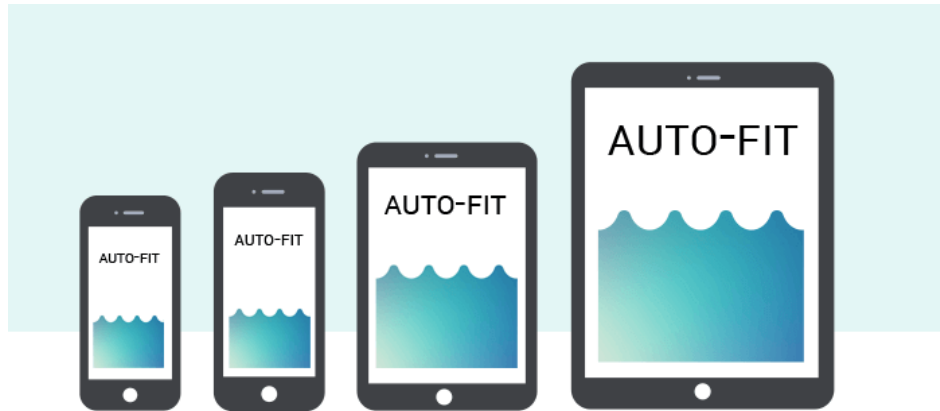
# Dynamic HTML



# Dynamic HTML (DHTML)

---

- ❑ Manipulating the web page's structure is essential for creating a highly responsive UI



- ❑ Example : <https://www.illy.com/en-us/home>
- ❑ Two main approaches
  - Manipulate page via plain Javascript
  - Manipulate page using Javascript + library (e.g., jQuery)

# HTML – INPUT Object

[https://www.w3schools.com/jsref/dom\\_obj\\_text.asp](https://www.w3schools.com/jsref/dom_obj_text.asp)

← → ↻ 🔒 w3schools.com/jsref/dom\_obj\_text.asp

🏠 HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP MORE ▾

API History  
API Storage  
  
HTML Objects  
 <input> button  
 <input> checkbox  
 <input> color  
 <input> date  
 <input> datetime  
 <input> datetime-local  
 <input> email  
 <input> file  
 <input> hidden  
 <input> image  
 <input> month

**Tip:** You can also access `<input type="text">` by searching through

## Create an Input Text Object

You can create an `<input>` element with `type="text"` by using the `createElement()` method:

### Example

```
var x = document.createElement("INPUT");  
x.setAttribute("type", "text");
```

Try it Yourself »

# HTML – INPUT Object

---

## Input Text Object Properties

Property	Description
<u><a href="#">autocomplete</a></u>	Sets or returns the value of the autocomplete attribute of a text field
<u><a href="#">autofocus</a></u>	Sets or returns whether a text field should automatically get focus when the page loads
<u><a href="#">defaultValue</a></u>	Sets or returns the default value of a text field
<u><a href="#">disabled</a></u>	Sets or returns whether the text field is disabled, or not
<u><a href="#">form</a></u>	Returns a reference to the form that contains the text field
<u><a href="#">list</a></u>	Returns a reference to the datalist that contains the text field
<u><a href="#">maxLength</a></u>	Sets or returns the value of the maxlength attribute of a text field
<u><a href="#">name</a></u>	Sets or returns the value of the name attribute of a text field
<u><a href="#">pattern</a></u>	Sets or returns the value of the pattern attribute of a text field
<u><a href="#">placeholder</a></u>	Sets or returns the value of the placeholder attribute of a text field
<u><a href="#">readOnly</a></u>	Sets or returns whether a text field is read-only, or not
<u><a href="#">required</a></u>	Sets or returns whether the text field must be filled out before submitting a form
<u><a href="#">size</a></u>	Sets or returns the value of the size attribute of a text field
<u><a href="#">type</a></u>	Returns which type of form element a text field is
<u><a href="#">value</a></u>	Sets or returns the value of the value attribute of the text field

# Example <Input>

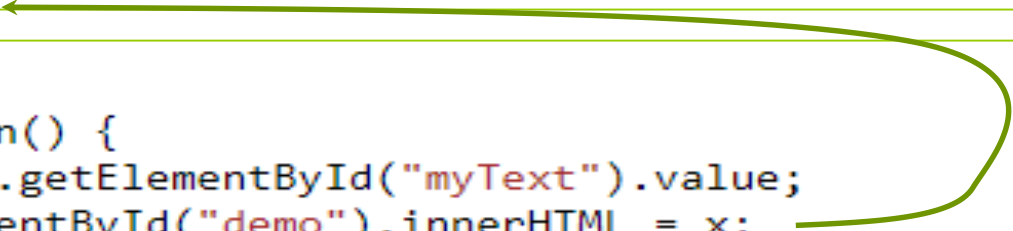
```
<input type="text" id="myText" value="The Text value">

<p>Click the "Try it" button to get the text in the text field.</p>

<button onclick="myFunction()">Click Me!</button>

<p id="demo"></p>
```

```
<script>
function myFunction() {
    var x = document.getElementById("myText").value;
    document.getElementById("demo").innerHTML = x;
}
</script>
```



## A demonstration of how to access a Text Field

Click the "Try it" button to get the text in the text field.

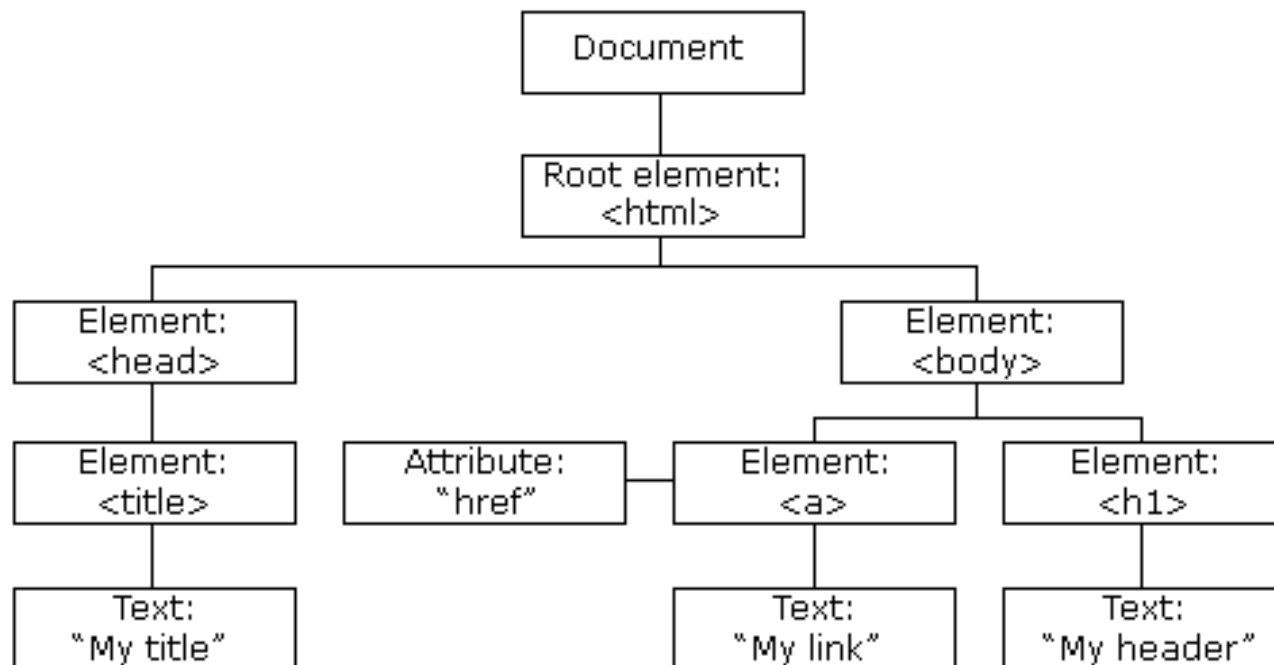
The Text value

[Try here](#)

HTML  
DOM

# Document Object Model (DOM) Tree

- When a web page is loaded, the browser creates a DOM of the page - a programming interface for **HTML** and XML documents
- A parent/child relationship with a tree of Objects





# DOM vs HTML

---

- ❑ The Document Object Model (**DOM**) is a programming interface for **HTML** and XML documents - it can be manipulated.
- ❑ This document can be either displayed in the browser window or as the **HTML** source.

# Finding Objects

---

- ❑ Objects can be referenced
  - by their id or name (this is the easiest way, but you need to make sure a name is unique in the hierarchy)
  - by their numerical position in the hierarchy, by walking the array that contains them
  - by their relation to parent, child, or sibling (*parentNode, previousSibling, nextSibling, firstChild, lastChild or the childNodes as array*)

# What is JavaScript ?

---

## □ Definition

- A **Programming Language** for the Web.
- Capable of updating and changing both **HTML** and **CSS**.
- Capable of **calculating, manipulating** and **validating** data.
- Adds interactivity to HTML pages.

## □ Examples

- validates forms
- react to events
- creates visual effects
- change the content of an HTML element dynamically
- control Web browser window

# How to use JavaScript ?

---

- How do we let the browser know in advance that it is not normal text but JavaScript ?

## Example:

```
<html>
<head>
<title>My JavaScript Page</title>
</head>

<body>
<script type="text/javascript">
    alert("Welcome to my world!!!");
</script>
</body>
</html>
```

Using the `<script>` tag

# How to use JavaScript ?

---

## □ Where to place it ?

You can enter JavaScript directly in both the `<head>` and `<body>` sections of the document.

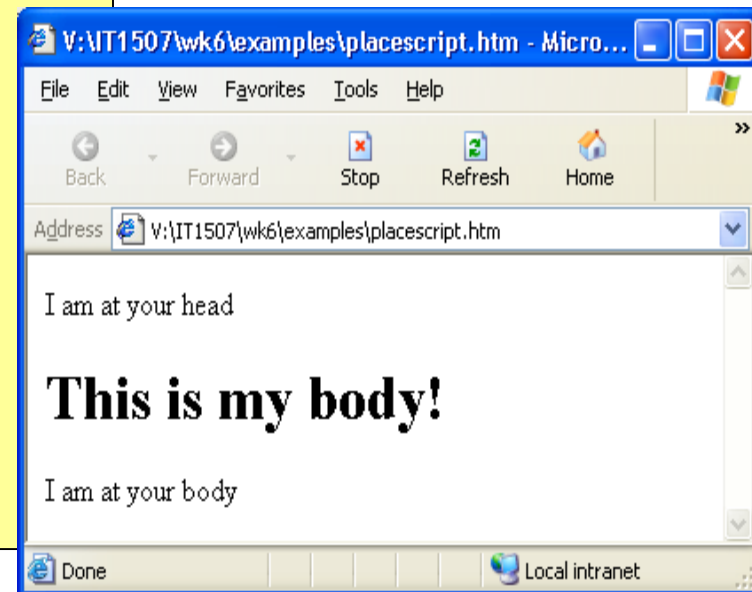
JavaScript code can also be stored in a `separate file` and be included to the document.

As a general rule, place as much of your JavaScript In the document head.

Reason being JavaScript is interpreted in the order in which it appears in the document. It is especially important when your document body need to perform tasks that depend on the scripts.

# Placement of JavaScript

```
<html>
<head>
<script type="text/javascript">
document.write("I am at your head");
</script>
</head>
<body>
<h1>This is my body!</h1>
<script type="text/javascript">
document.write("I am at your body");
</script>
</body>
</html>
```



# JavaScript as an external file

---

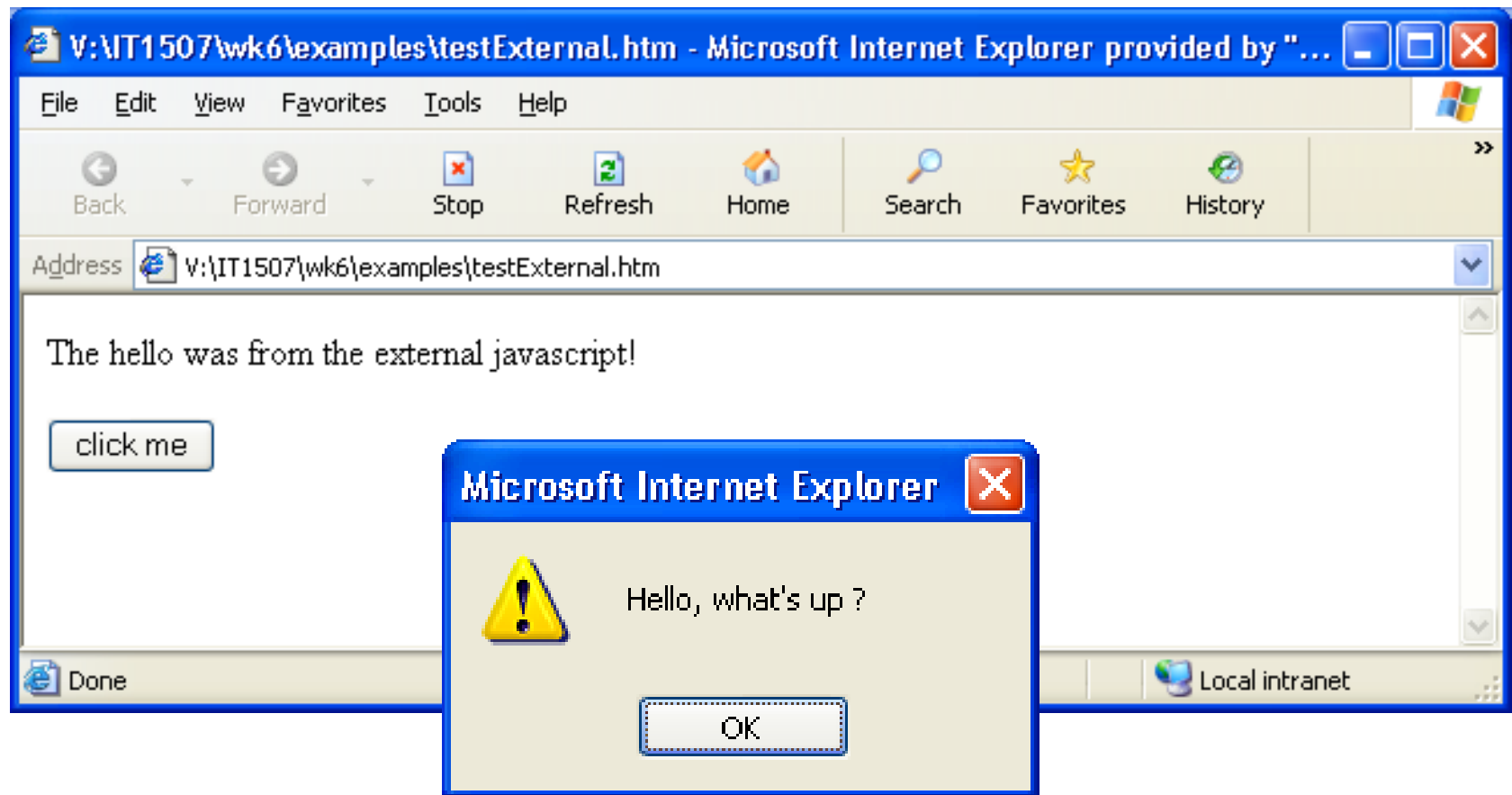
```
<html>
<head>
<script type="text/javascript" src="hello.js">
</script>
</head>
<body>
<p>The hello was from the external javascript!</p>
<form>
<input type="button" value="click me" onClick="hello();"
/>
</form>
</body>
</html>
```

```
function hello() {
    document.write("Hello, what's up?");
}
```



hello.js

# JavaScript as an external file





# Script loading & execution

---

- ❑ In modern websites, scripts are often “heavier” than HTML: their download size is larger, and processing time is also longer.

```
<p>...content before script...</p>  
  
<script src="https://javascript.info/article/script-async-defer/long.js?speed=1"></script>  
  
<!-- This isn't visible until the script loads -->  
  
<p>...content after script...</p>
```

- ❑ HTML will pause when script is loading
- ❑ Resulting in webpage loading delay

# Script loading & execution

---

- 3 types of script execution
  - **Normal execution** *<script>*
  - **Deferred execution** *<script defer>*
  - **Asynchronous execution** *<script async>*

# Script loading & execution

---

## ■ Normal execution `<script>`

- This is the default behavior of the `<script>` element.
- Parsing of the HTML code pauses while the script is executing.

## ■ Deferred execution `<script defer>`

- This `defer` attribute informs the browser not to wait for the script.
- the browser will continue to process the HTML, build DOM.
- The script loads “in the background”, and then runs when the DOM is fully built.
- However, not every browser supports `defer`.
- The `defer` attribute is only referencing external scripts.

# Script loading & execution

## Defer attribute

---

```
<p>...content before script...</p>
```

```
<script defer src="https://javascript.info/article/script-async-defer/long.js?speed=1"></script>
```

```
<!-- visible immediately -->
```

```
<p>...content after script...</p>
```

### □ Output

- *...content before script ...*
- *...content after script ...*
- *Script completed (this is a heavy script)*

### □ Summary

- Scripts with defer never block the page.
- Scripts with defer always execute when the DOM is ready

# Script loading & execution

## Defer attribute

---

- ❑ Does defer script keep their relative order ?

```
<script defer src="https://javascript.info/article/script-async-defer/OneHeavy.js"></script>
```

```
<script defer src="https://javascript.info/article/script-async-defer/TwoLow.js"></script>
```

- ❑ both scripts download in together – “not to block” rule
- ❑ If TwoLow.js loads first, it will wait and runs after OneHeavy.js executes.

# Script loading & execution

---

## ■ **Asynchronous execution** *<script async>*

- Somewhat behave like defer. Script non-blocking as well.
- Other scripts don't wait for async scripts, and async scripts don't wait for them.
- Either scripts will run first as long as it is loaded.
- "Load-first" order

```
<script async src="https://javascript.info/article/script-async-defer/OneHeavy.js"></script>
```

```
<script async src="https://javascript.info/article/script-async-defer/TwoLow.js"></script>
```

# Script loading & execution

## Async Attribute

---

- Async scripts are great when we integrate an independent third-party script into the page: counters, ads and so on.
- They don't depend on developer's scripts, and vice-versa

<!-- Google Analytics is usually added like this -->

```
<script async src="https://google-analytics.com/analytics.js"></script>
```


# Syntax of JavaScript

---

- ❑ JavaScript lines end with a semicolon.
- ❑ Always put the text within " ".
- ❑ Capital letters are different from lowercase letters.

```
document.write("Hello World!");
```

```
Document.write(Hello World!)
```




- ❑ There are keywords that carry special meaning for JavaScript. Programmers are not allowed to use them for different purpose.



# What does JavaScript have?

---

- ❑ Variable - a place to hold information
  - ❑ Statement
    - Assign value to a variable
    - Checking conditions for decision making
      - ❑ If....else
      - ❑ switch case
    - Repeating a task
      - ❑ do ... while
      - ❑ for , for.. In
- 
- A yellow rectangular box labeled "Control statements" has two red arrows pointing from it. One arrow points to the "If....else" and "switch case" sub-items, and the other points to the "do ... while" and "for , for.. In" sub-items.

# What does JavaScript have?

---

- ❑ Operators - for example: + - \* / = ==
- ❑ Function - statements can be grouped to accomplished certain task(s)
- ❑ Event Handlers - will be triggered when certain events happen. For example:
  - onclick, onmouseover, onmouseout

# What does JavaScript have?

---

- Objects - there some objects which have been defined and can be used readily. For example:
  - window, document, form, history, image, location, math, string, array, date .....

# Elements of the JavaScript Language

---

## □ Arrays

- Creation: `var arrayName = new Array();`
- Access: `arrayName[index]`
- Size of array: `arrayName.length`

## □ The **if**, **for** and **while** statements

- Syntax similar to Java

# Array

---

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var studNames = ['Alan', 'Bervyn', 'Candy'];
```

```
document.getElementById("demo").innerHTML = studNames;
```

```
</script>
```

```
</body>
```

```
</html>
```

Try examples in <https://jsfiddle.net>

# Array

---

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var studNames = ['Alan', 'Bervyn', 'Candy'];
```

```
document.getElementById("demo").innerHTML = studNames[0];
```

```
</script>
```

```
</body>
```

```
</html>
```

# Variable Name

---

- ❑ Rules and conventions when naming a variable:
  - First character of an Identifier may begin with an uppercase or lowercase ASCII letter, dollar sign (\$), or underscore ( \_ )
  - You can use numbers in an identifier, but not as the first character
  - You **cannot include spaces** in an identifier
  - You **cannot use reserved words** for identifiers

# Variable Names (Cont.)

- ❑ Reserved words (keywords):
  - Special words part of the JavaScript

**Table 2-1** JavaScript reserved words

<code>abstract</code>	<code>else</code>	<code>instanceof</code>	<code>switch</code>
<code>boolean</code>	<code>enum</code>	<code>int</code>	<code>synchronized</code>
<code>break</code>	<code>export</code>	<code>interface</code>	<code>this</code>
<code>byte</code>	<code>extends</code>	<code>long</code>	<code>throw</code>
<code>case</code>	<code>false</code>	<code>native</code>	<code>throws</code>
<code>catch</code>	<code>final</code>	<code>new</code>	<code>transient</code>
<code>char</code>	<code>finally</code>	<code>null</code>	<code>true</code>
<code>class</code>	<code>float</code>	<code>package</code>	<code>try</code>
<code>const</code>	<code>for</code>	<code>private</code>	<code>typeof</code>
<code>continue</code>	<code>function</code>	<code>protected</code>	<code>var</code>
<code>debugger</code>	<code>goto</code>	<code>public</code>	<code>void</code>
<code>default</code>	<code>if</code>	<code>return</code>	<code>volatile</code>
<code>delete</code>	<code>implements</code>	<code>short</code>	<code>while</code>
<code>do</code>	<code>import</code>	<code>static</code>	<code>with</code>
<code>double</code>	<code>in</code>	<code>super</code>	



# Declaring Variables

---

- ❑ To use a variable, you have to **declare** it first.

**var** *variable\_name* = *value*;

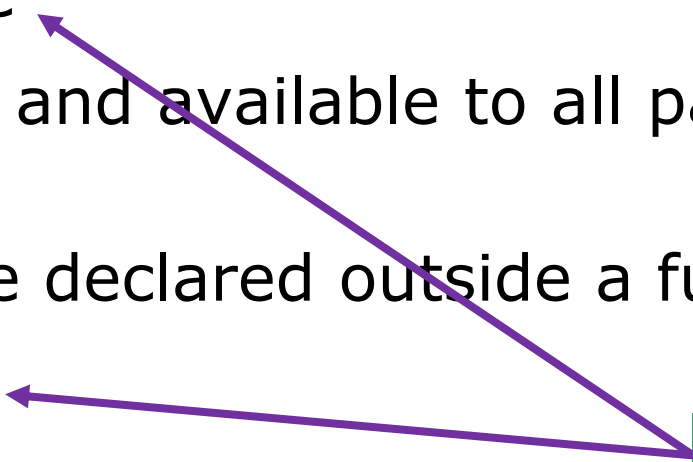
(without the var it is still ok, but it is a good practice)

- ❑ Global variable

- it is known and available to all parts of your program.
- it should be declared outside a function.

- ❑ Local variable

- it is known within a function.
- it is declared inside a function.



Variable scope

# Scope

## JavaScript Scope

A GLOBAL variable can be accessed from anywhere

I can display Volvo

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>JavaScript Scope</h2>
```

```
<p>A GLOBAL variable can be accessed from anywhere.</p>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var carName = "Volvo";
```

```
myFunction();
```

```
function myFunction() {
```

```
    document.getElementById("demo").innerHTML =
```

```
    "I can display " + carName;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

# Declaring Variables (Cont.)

---

Examples:

```
var myVar ="Hello World";
```

```
var myVar =8;
```

```
var myVar =4.56;
```

```
var myVar =true;
```

```
var myVar =null;
```

# String

---

- ❑ To assign a string to a variable
  - ❖ `myFeeling="sleepy"`
  - ❖ `degree="extremely"`
  
- ❑ To join (concatenate) two strings
  - ❖ `mode = degree + " " + myFeeling`  
(the value of mode is "extremely sleepy")

# Dialog boxes

---

## Example 1:

Experience with 3 different kinds of popup dialog boxes.

```
<html>
<body>
<script type="text/javascript">
<!--
var name=prompt("What is your name ?");
var answer=confirm("Are you sure ?");
if (answer==true) {
    alert("Hi "+name+", nice to meet you");
} else {
    alert("Can't remember your name?! That's ok! Relax");
}
window.close();
-->
</script>
</body>
</html>
```

# Convert String to Integer

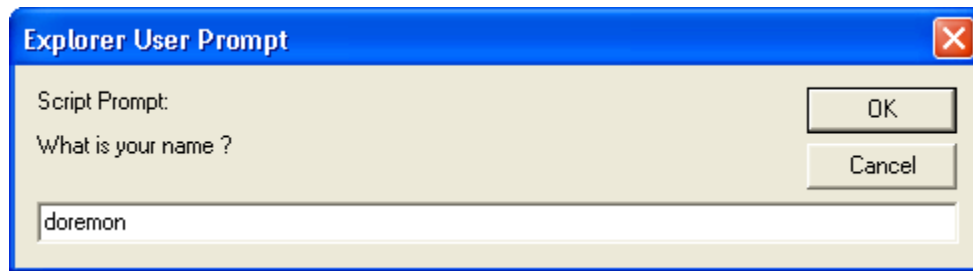
---

```
<html>
<head>
<script type="text/javascript">
function add(num1, num2) {
    total=parseInt(num1)+parseInt(num2);
    return total;
}
</script>
</head>
<body>
<script type="text/javascript">
var firstNumber    = prompt("Please enter the first number ");
var secondNumber   = prompt("Please enter the second number ");
var total          = add(firstNumber, secondNumber);
alert("The result is : "+total);
</script>
</body>
</html>
```

# Let's start with simple examples!

## Example 1:

Experience with 3 different kinds of popup dialog boxes.



prompt



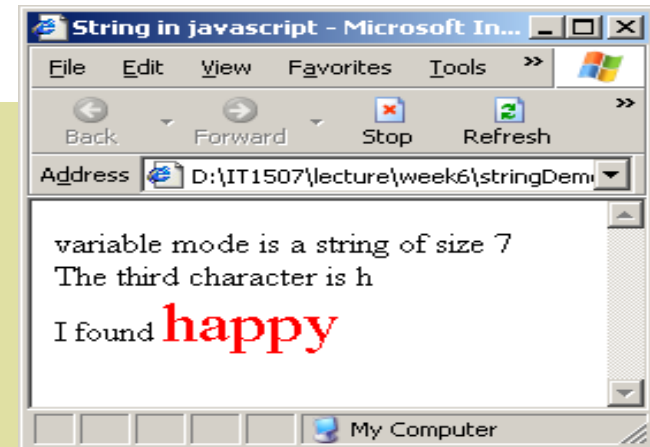
confirm



alert

# Usage of String

- What other ways can we manipulate String ?
  - Built-in function (*not limited to these*)
    - charAt(i) // get the character at position i-1
    - substr(startIndex,length) // extract the substring
    - length // the size of the String



```
<html>
<head>
  <title>String in javascript</title>
</head>
<body>
  <script type="text/javascript">
    var mode="unhappy";
    document.write("variable mode is a string of size "+mode.length+"<br/>");
    document.write("The third character is "+mode.charAt(2)+"<br/>");
    document.write("I found <span style='color:red; font size:+2em'>" +mode.substr(2,5)+"</span>");
  </script>
</body>
</html>
```



# Functions

---

- Function is formed by group of statements

keyword

function name

parameter

```
function add(num1, num2) {  
    var total;  
    total=parseInt(num1)+parseInt(num2);  
    return total;  
}
```

return a value (*optional*)

# Functions

---

- User defined functions

- Defined by the user as previous example

- Built-in functions

For example:

- `eval()` – evaluate the value
  - **`parseInt('62')`** – converts string to integer
  - **`parseFloat('62.3')`** – converts string to float
  - `Boolean(x)` – returns true or false
  - `isNaN()` – is not a number ?

# Functions - Example

---

```
<html>
<head>
<script type="text/javascript">
function add(num1, num2) {
    total=parseInt(num1)+parseInt(num2);
    return total;
}
</script>
</head>
<body>
<b><script type="text/javascript">
    var firstNumber=prompt("Please enter the first number ");
    var secondNumber=prompt("Please enter the second number ");
    var total=add(firstNumber, secondNumber);
    alert("The result is : "+total);
</script>
</body>
</html>
```

# Boolean example

---

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p>Display the Boolean value of false:</p>
```

```
<button onclick="myFunction()">Try it</button>
```

```
<p id="demo"></p>
```

```
<script>
```

```
function myFunction() {
```

```
    var x = 0;
```

```
    document.getElementById("demo").innerHTML = Boolean(x);
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

# Objects

---

```
<!DOCTYPE html>
<html>
<body>
<p>Creating a JavaScript Object.</p>
<p id="here">?</p>
<script>
var student = {
    firstName : "Peter",
    lastName  : "Anderson",
    age       : 26,
    school    : "wkwsci"
};

document.getElementById("here").innerHTML =
student.firstName + " is " + student.age + " years old.";
</script>

</body>
</html>
```

# Events

---

- Something happen
  - Actions that users perform ([DOM events](#))
    - Move the mouse over
    - Click on a button/word/image
  - Web browser operation, e.g.
    - Loading a document

# Handling an event - onmouseover

```
<html>
<head>
<script type="text/javascript">
function show() {
    alert("Nobody!!");
}
</script>
</head>
<body>
<p>Do you know
    <span style="font-size:+2em; color:red" onmouseover="show();">who</span>
    am I ?</p>
</body>
</html>
```

Calling the function



Detect mouse over event



show

# Handling an event – onclick

---

## □ Tag

- `<p onclick="doSomething()">Click me</p>`  
`<h1 onclick="doSomething()">Click me</h1>`

## □ Button

- `<input type="button" onclick="changeLink()" value="Change link">`

## □ Img

- ``



# Handling an event – onclick

---

```
<html>
<head>
<script>
    function changecolor()
    {
        document.body.style.background = "blue";
    }
</script>
</head>
<body>
    <p onclick="changecolor()">change background color to blue</p>
</body>
</html>
```

[changeBgColor.htm](#)

# Handling an event – onclick+Button

---

```
<html>
<head>
<script>
function changeLink()
{
document.getElementById('myAnchor').innerHTML="WKWSCI";
document.getElementById('myAnchor').href="http://www.wkwsci.ntu.edu.sg";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>
<body>

<a id="myAnchor" href="http://www.ntu.edu.sg">NTU</a>
<input type="button" onclick="changeLink()" value="Change link">

</body>
</html>
```

<eventButton.htm>

# Handling an event – onclick+img+search

---

[changeImage.htm](#)

```
<html>
<body>
<script>
function changeImage()
{
  element=document.getElementById('myimage')
  if (element.src.match("bulbon"))
  {
    element.src="pic_bulboff.gif";
  }
  else
  {
    element.src="pic_bulbon.gif";
  }
}
</script>

<p>Click to turn on/off the light</p>
</body>
</html>
```

# Document Object Model (Search)

---

- Direct access of document objects
  - getElementById
  - getElementsByTagName
  
- Example:
  - To retrieve the color of an element with id="p1" in the document:
    - color= **document.getElementById("p1").style.color**
  - To retrieve all images in the document:
    - Images= **document.getElementsByTagName("img")**

# Handling an event – onclick+innerHTML

---

```
<html>
<head>
<script>
  function change(here)
  {
    here.innerHTML="to there";
  }
</script>
</head>
<body>
  <p onclick="change(this)">From here ...</p>
</body>
</html>
```

[changeText.htm](#)

# White spaces

---

JavaScript ignores multiple spaces.  
You can add white space to your script to make it more readable.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>JavaScript Whitespaces</h2>
```

```
<p id="demo"></p>
```

```
<script>
```

```
document.getElementById("demo").innerHTML =
```

```
"Hello          Dolly!";
```

```
</script>
```

```
</body>
```

```
</html>
```

# Javascript Comments

Single line comments start with `//`  
Multi-line comments start with `/*` and end with `*/`.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>JavaScript Comments</h2>
```

```
<p id="demo"></p>
```

```
<script>
```

```
var x;
```

```
x = 5;
```

```
// x = 6; I will not be executed
```

```
document.getElementById("demo").innerHTML = x;
```

```
/*
```

```
Multiple
```

```
Lines
```

```
*/
```

```
</script>
```

```
</body>
```

```
</html>
```

**JavaScript Comments**

# Case-sensitive

## JavaScript is Case Sensitive

Try change lastName to lastname.

Peterson

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript is Case Sensitive</h2>
<p>Try change lastName to lastname.</p>

<p id="demo"></p>

<script>
var lastname, lastName;
lastName = "Doe";
lastname = "Peterson";
document.getElementById("demo").innerHTML = lastname;
</script>

</body>
</html>
```



# Error Handling

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p id="demo"></p>
```

```
<script>
```

```
try {
```

```
    aaalert("Welcome guest!");
```

```
}
```

```
catch(err) {
```

```
    document.getElementById("demo").innerHTML = err.message;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

The **try** statement allows you to define a block of code to be tested for errors while it is being executed.

The **catch** statement allows you to define a block of code to be executed, if an error occurs in the try block.

# Error handling

aaalert is not defined

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<p id="demo"></p>
```

```
<script>
```

```
try {
```

```
    aaalert("Welcome guest!");
```

```
}
```

```
catch(err) {
```

```
    document.getElementById("demo").innerHTML = err.message;
```

```
}
```

```
</script>
```

```
</body>
```

```
</html>
```

# Error handling

The **throw** statement allows you to create a custom error.

Technically you can **throw an exception (throw an error)**.

```
<!DOCTYPE html>
<html>
<body>
<p>Please input a number between 5 and 10:</p>

<input id="demo" type="text">
<button type="button" onclick="myFunction()">Test Input</button>
<p id="p01"></p>

<script>
function myFunction() {
  var message, x;
  message = document.getElementById("p01");
  message.innerHTML = "";
  x = document.getElementById("demo").value;
  try {
    if(x == "") throw "empty";
    if(isNaN(x)) throw "not a number";
    x = Number(x);
    if(x < 5) throw "too low";
    if(x > 10) throw "too high";
  }
  catch(err) {
    message.innerHTML = "Input is " + err;
  }
}
</script>
</body>
</html>
```

# Error handling

---

```
<script>
function myFunction() {
    var message, x;
    message = document.getElementById("p01");
    message.innerHTML = "";
    x = document.getElementById("demo").value;
    try {
        if(x == "") throw "empty";
        if(isNaN(x)) throw "not a number";
        x = Number(x);
        if(x < 5)    throw "too low";
        if(x > 10)   throw "too high";
    }
    catch(err) {
        message.innerHTML = "Input is " + err;
    }
}
</script>
```

# Creating nodes + Appending

create element node first, and then append it to an existing element.

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<div id="div1">
```

```
<p id="p1">This is a paragraph.</p>
```

```
<p id="p2">This is another paragraph.</p>
```

```
</div>
```

```
<script>
```

```
var para = document.createElement("p");
```

```
var node = document.createTextNode("This is new.");
```

```
para.appendChild(node);
```

```
var element = document.getElementById("div1");
```

```
element.appendChild(para);
```

```
</script>
```

```
</body>
```

```
</html>
```

This is a paragraph.

This is another paragraph.

This is new.

This code creates a new <p> element:

```
var para = document.createElement("p");
```

To add text to the <p> element, you must create a text node first. This code creates a text node:

```
var node = document.createTextNode("This is a new paragraph.");
```

Then you must append the text node to the <p> element:

```
para.appendChild(node);
```

Finally you must append the new element to an existing element.

This code finds an existing element:

```
var element = document.getElementById("div1");
```

This code appends the new element to the existing element:

```
element.appendChild(para);
```

# More Examples

---

## □ Form Validation

- `isNaN()` – test for not-a-number
  - `alert("You must enter a numeric value!")`
  - `checkbox.checked==true` – verify if checkbox is checked
  - `Onchange` – update to form values
  - `OnClick` – clicking on button
- [W3schools example](#)

# More Examples

---

## □ Reacting to Events

- `document.images[0].src="happyface.jpg"`
- `document.bgColor="#0000ff"`
- `onmouseover`
- `onmouseout`
- `onclick`



# More Examples

---

## □ Change HTML element dynamically

- `x = document.getElementById('myTable').insertRow(2);`
- `y = x.insertCell(0)`
- `y = x.insertCell(1)`
- `y.innerHTML="cell value"`
- `y.style.background="yellow"`
- `onClick`

---

The end

# More Examples

---

## □ Web browser window

```
■ window.open(  
  "http://www.w3schools.com",  
  "_blank",  
  "toolbar=yes,  
  location=yes,  
  directories=no,  
  status=no,  
  menubar=yes,  
  scrollbars=yes,  
  resizable=no,  
  copyhistory=yes,  
  width=400,  
  height=400"  
)
```