

JAVASCRIPT – LECTURE 2

Outline — Part B

- JavaScript Functions and Events
 - ▣ Events Handlers
- Using Object in JavaScript
 - ▣ Object-Oriented Programming
 - ▣ JavaScript Object Model
 - ▣ Using Built-In objects (Predefined Object)
 - ▣ Custom Object Types
- Error in JavaScript
- Exception Handling in JavaScript

Functions

- A function is a block of organized reusable code (a set of statements) for performing a single or related action.
- Begins with keyword “function” and the function name and “(...)”
- Inside the parentheses
 - ▣ We can pass parameters to the function
 - ▣ E.g. `function myfuc(arg1, arg2) {...}`
 - ▣ Built-in and user-defined functions

Built-In Functions

- Functions provided by the language and you cannot change them to suit your needs.
- Some of the built-in functions in JavaScript are shown here:
 - ▣ `eval` - `eval(expr)`
 - `eval` evaluates the expression or statements
 - ▣ `isFinite`
 - Determines if a number is finite
 - ▣ `isNaN`
 - Determines whether a value is “Not a Number”
 - ▣ `parseInt`
 - Converts string literals to integers, no number → NaN.
 - ▣ `parseFloat`
 - Finds a floating-point value at the beginning of a string.

User-Defined Functions

- ❑ For some functionality, you cannot achieve by only using the built-in functions.
- ❑ You can define a function as follows

```
function <function_name> (parameters)
{
    // code segments;
}
```

Function Declarations

- Declaration Syntax

- ▣ Functions are declared using the function reserved word
- ▣ The return value is not declared, nor are the types of the arguments
- ▣ Examples:

```
function square(x) { return(x * x); }
```

```
function factorial(n) {  
    if (n <= 0) {  
        return(1);  
    } else {  
        return(n * factorial(n - 1));  
    }  
}
```

Events

- Events are the actions that occur as a result of browser activities or user interactions with the web pages.
- Such as the user performs an action (mouse click or enters data)
- We can validate the data entered by a user in a web form
- Communicate with Java applets and browser plug-ins

Event Categories

- Keyboard and mouse events
 - ▣ Capturing a mouse event is simple
- Load events
 - ▣ The page first appears on the screen: “Loads”, leaves: “Unloads”, ...
- Form-related events
 - ▣ onFocus() refers to placing the cursor into the text input in the form.
- Others
 - ▣ Errors, window resizing.

Events defined by JavaScript

HTML elements	HTML tags	JavaScript defined events	Description
Link	<a>	click dblClick mouseDown mouseUp mouseOver	Mouse is clicked on a link Mouse is double-clicked on a link Mouse button is pressed Mouse button is released Mouse is moved over a link
Image		load abort error	Image is loaded into a browser Image loading is abandoned An error occurs during the image loading
Area	<area>	mouseover mouseout dblClick	The mouse is moved over an image map area The mouse is moved from image map to outside The mouse is double-clicked on an image map
Form	<form>	submit Reset	The user submits a form The user refreshes a form
...

Event Handlers

- When an event occurs, a code segment is executed in response to a specific event is called “event handler”.
- Event handler names are quite similar to the name of events they handle.
- E.g the event handler for the “click” event is “onClick”.
- `<HTMLtag eventhandler="JavaScript Code">`

Event Handlers

Event Handlers	Triggered when
onChange	The value of the text field, textarea, or a drop down list is modified
onClick	A link, an image or a form element is clicked once
onDbClick	The element is double-clicked
onMouseDown	The user presses the mouse button
onLoad	A document or an image is loaded
onSubmit	A user submits a form
onReset	The form is reset
onUnLoad	The user closes a document or a frame
onResize	A form is resized by the user

onClick event Handler

```
<html>
<head>
<title>onClick Event Handler Example</title>
<script language="JavaScript">
function warnUser(){ return confirm("Web students?"); }
</script>
</head>
<body>
<a href="http://alphapeeler.sf.net", onClick="return
warnUser();">Students access only</a>
</body>
</html>
```

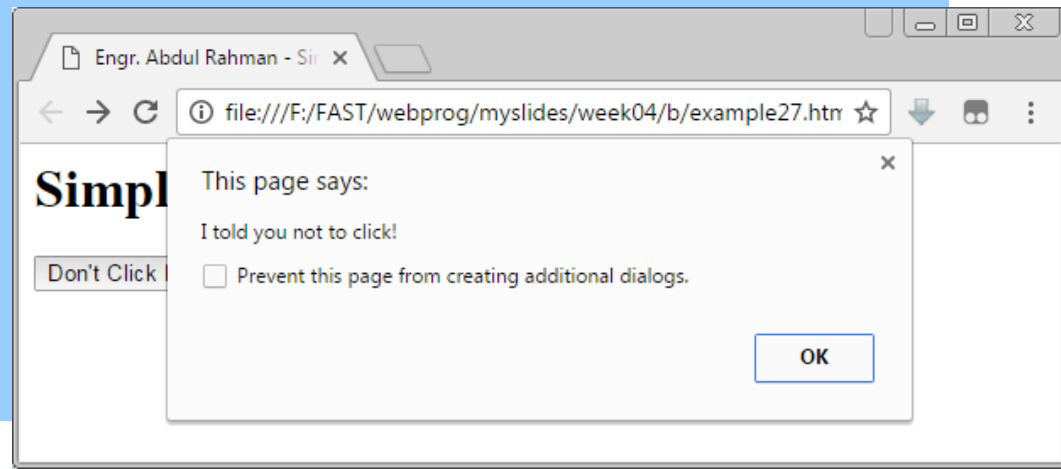
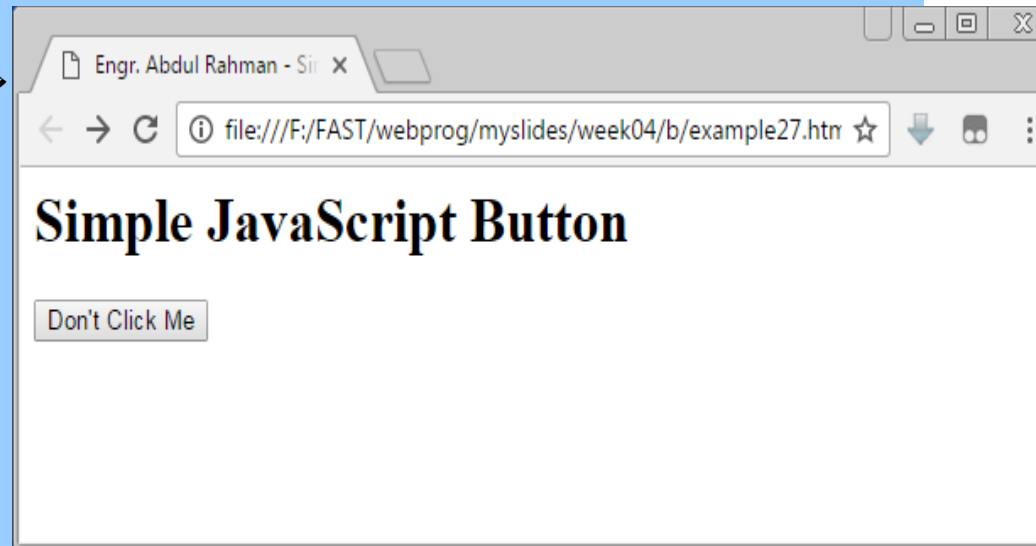
onLoad event Handler

```
<html>
<head>
<title>onLoad and onUnload Event Handler Example</title>
</head>
<body onLoad="alert('Welcome User');"
onUnload="alert('Thanks for visiting the page');">
Load and UnLoad event test.
</body>
</html>
```

User Events, Form Example

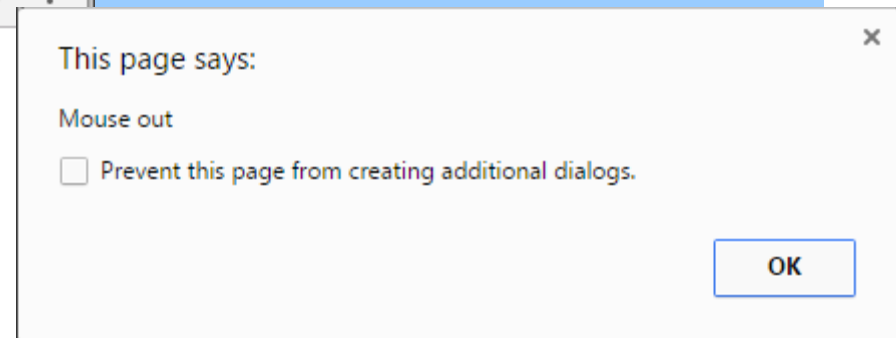
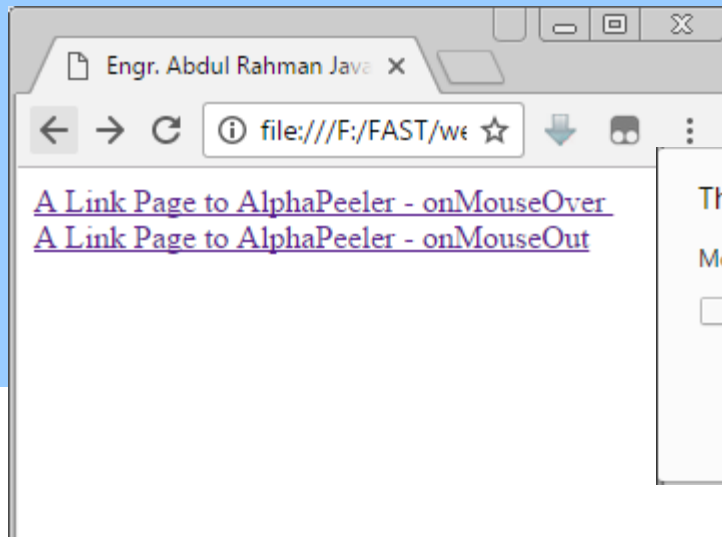
```
<html><head>
  <title>Simple JavaScript Button</title>
  <script language="JavaScript">!--
  function dontClick() {
    alert("I told you not to click!");
  }
  // --></script>
</head>

<body>
<h1>Simple JavaScript Button</h1>
<form>
  <input type="button"
    value="Don't Click Me"
    onClick="dontClick()">
</form>
</body></html>
```



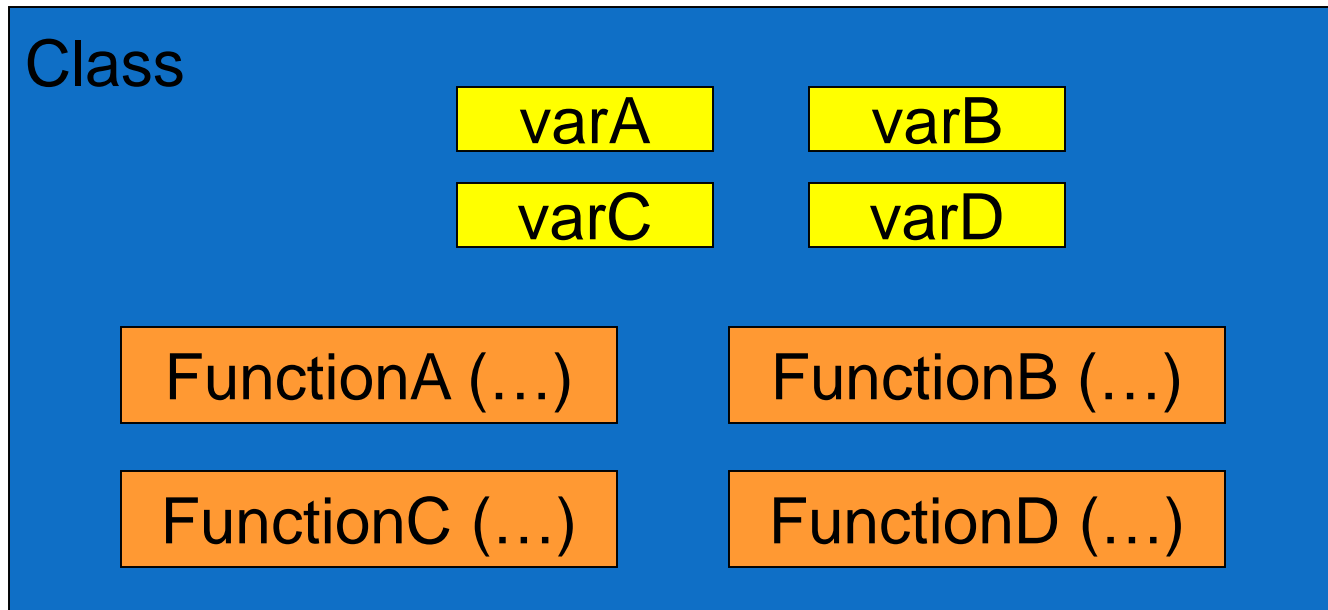
onMouseOver and onMouseOut Event Handlers

```
<html>
<body>
<a href="http://alphapeeler.sf.net" onMouseOver = "alert('mouse over');">
A Link Page to AlphaPeeler - onMouseOver
<BR>
<a href="http://alphapeeler.sf.net" onMouseOut = "alert('Mouse out');">
A Link Page to AlphaPeeler - onMouseOut
</a>
</body>
</html>
```



Understanding JavaScript Objects

- One of the most important features of JavaScript, enables modular programs.
- Objects are based on Classes, variables, functions, statements are contained in a structure called class.



Class and Object

- ❑ You can instantiate an object from a class by using the constructor function.
- ❑ JavaScript is said to be an Object-based programming language.
- ❑ What is property?
 - ▣ A variable belongs to the object.
- ❑ What is method?
 - ▣ It is a function belongs to the object.

Creating Instances of Objects

- You can use the “**new**” operator to create instances of objects of a particular class or object type.
 - ▣ Variable = new objectType(parameters)
- This objectType() is called constructor.
- E.g. Date is a predefined object type.

```
var objA = new Date();  
objB = new Date(2017, 5, 23);  
document.write("objA = " + objA + "<BR>");  
document.write("objB = " + objB);
```



Objects and Classes

- Fields Can Be Added On-the-Fly
 - ▣ Adding a new property (field) is a simple matter of assigning a value to one
 - ▣ If the field doesn't already exist when you try to assign to it, JavaScript will create it automatically.
 - ▣ For instance:

```
var test = new Object();  
test.field1 = "Value 1"; // Create field1 property  
test.field2 = 7;         // Create field2 property
```

Objects and Classes – Literal Notation

- You Can Use Literal Notation
 - ▣ You can create objects using a shorthand “literal” notation of the form

`{ field1:val1, field2:val2, ... , fieldN:valN }`

- ▣ For example, the following gives equivalent values to `object1` and `object2`

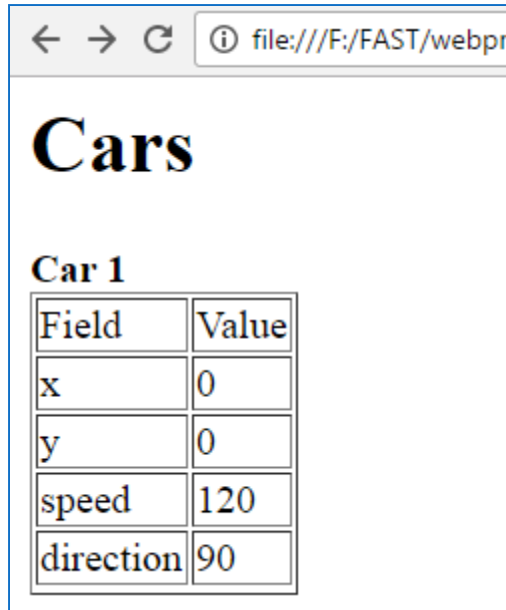
```
var object1 = new Object();  
var object2 = new Object();  
object1.x = 3;  
object1.y = 4;  
object1.z = 5;  
object2 = { x:3, y:4, z:5 };
```

Objects and Classes - Constructor

- A “Constructor” is Just a Function that Assigns to “this”
 - JavaScript does not have an exact equivalent to Java’s class definition
 - The closest you get is when you define a function that assigns values to properties in the `this` reference
 - Calling this function using `new` binds `this` to a new Object
 - For example, following is a simple constructor for a `Car` class

```
function Car(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
}
```

Constructor, Example 33.html



A screenshot of a web browser window. The address bar shows 'file:///F:/FAST/webpr'. The page title is 'Cars'. Below the title, there is a section labeled 'Car 1' containing a table with 2 columns: 'Field' and 'Value'. The table has 5 rows: the first row is the header, and the following four rows contain the data for 'Car 1'.

Field	Value
x	0
y	0
speed	120
direction	90

Q: Implement the HTML shown on the screen.

Hints:

```
document.getElementById("x").innerHTML=car1.x;
```

```
var car1 = new Car(0, 0, 1, 90);
```

Class Methods, Example

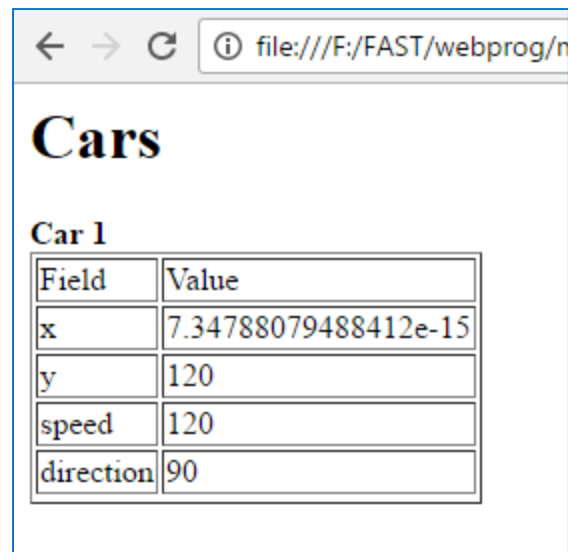
```
function degreesToRadians(degrees) {  
    return(degrees * Math.PI / 180.0);  
}
```

```
function move() {  
    var angle = degreesToRadians(this.direction);  
    this.x = this.x + this.speed * Math.cos(angle);  
    this.y = this.y + this.speed * Math.sin(angle);  
}
```

```
function Car(x, y, speed, direction) {  
    this.x = x;  
    this.y = y;  
    this.speed = speed;  
    this.direction = direction;  
    this.move = move;  
}
```

Class Methods, Result

```
var car1= new Car(0, 0, 1, 90);  
car1.move();
```



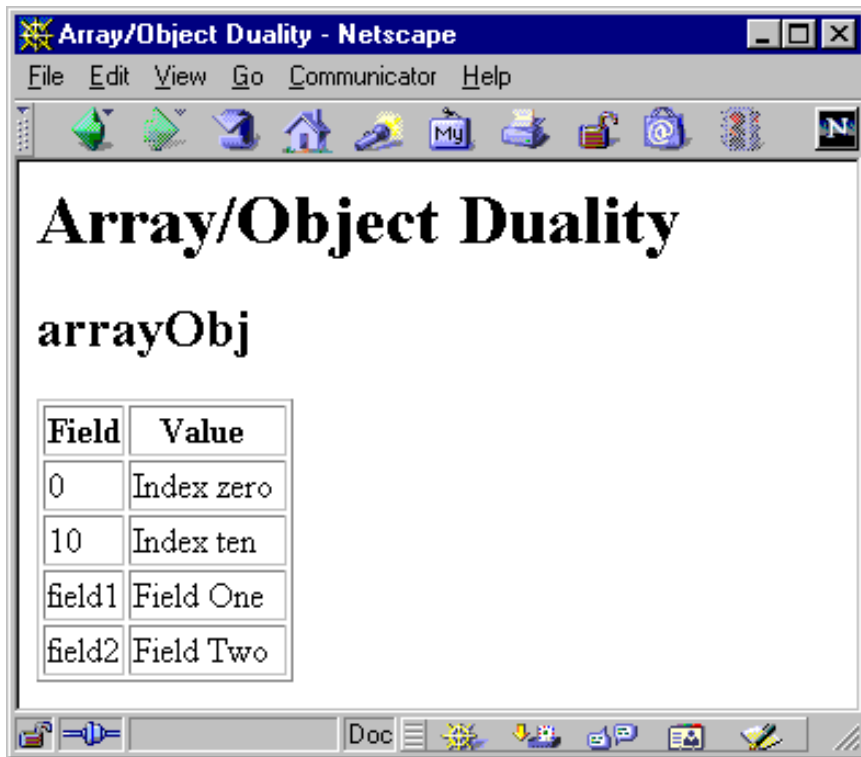
A screenshot of a web browser window. The address bar shows a file path: file:///F:/FAST/webprog/n. The page title is "Cars". Below the title, there is a section labeled "Car 1". Under "Car 1", there is a table with two columns: "Field" and "Value". The table contains five rows of data.

Field	Value
x	7.34788079488412e-15
y	120
speed	120
direction	90

Objects and Classes - Arrays

- Arrays
 - For the most part, you can use arrays in JavaScript a lot like Java arrays.
 - Here are a few examples:
var squares = new Array(5);
for(var i=0; i<squares.length; i++) {
 vals[i] = i * i;
}
// Or, in one fell swoop:
var squares = new Array(0, 1, 4, 9, 16);
var array1 = new Array("fee", "fie", "fo", "fum");
// Literal Array notation for creating an array.
var array2 = ["fee", "fie", "fo", "fum"];
 - Behind the scenes, however, JavaScript simply represents arrays as objects with numbered fields
 - You can access named fields using either `object.field` or `object["field"]`, but numbered fields only via `object[fieldNumber]`

Array, Example



```
var arrayObj = new Object();  
arrayObj[0] = "Index zero";  
arrayObj[10] = "Index ten";  
arrayObj.field1 = "Field One";  
arrayObj["field2"] = "Field Two";
```

Build-In JavaScript Objects

Object	Description
Array	Creates new array objects
Boolean	Creates new Boolean objects
Date	Retrieves and manipulates dates and times
Error	Returns run-time error information
Function	Creates new function objects
Math	Contains methods and properties for performing mathematical calculations
Number	Contains methods and properties for manipulating numbers.
String	Contains methods and properties for manipulating text strings

“onerror” event handler

```
<html><head>
<script language="JavaScript">
function errorHandler(errorMsg, url, lineNumber, column, errorObj) {
    alert('Error: ' + errorMsg + '\n'
        + ' url: ' + url + '\n'
        + ' Line: ' + lineNumber + '\n'
        + ' Column: ' + column + '\n'
        + ' StackTrace: ' + errorObj);
}
window.onerror = errorHandler;
</script>
</head>
<body>
<script language="JavaScript">
    document.write("Hello there;
</script>
</body>
</html>
```

This page says:

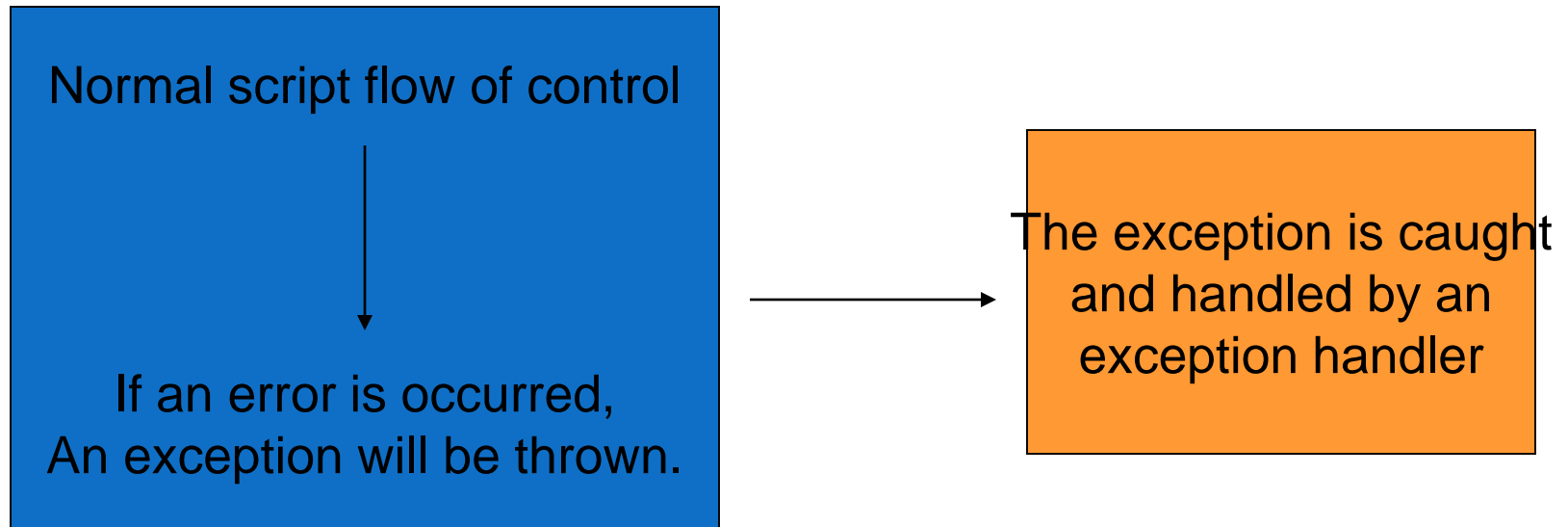
Error: Uncaught SyntaxError: Invalid or unexpected token
url: file:///F:/FAST/webprog/myslides/week04/b/example35.html
Line: 17
Column: 17
StackTrace: SyntaxError: Invalid or unexpected token

☐ Prevent this page from creating additional dialogs.

OK

Exception Handling in JavaScript

- ❑ An exception is an error generated by the script.
- ❑ The code that handles an exception is called an exception handler that will catch exceptions.



try ... catch ... finally

```
try
{
    // normal statements
    // might result in an error, throw exceptions
}
catch(errorVariable)
{
    // statements that execute in the exception event
}
finally
{
    // After the execution in the catch or try block,
    // the statements in the finally block are executed.
}
```

Try ... catch ... finally example

```
<script language="JavaScript">
try{
    document.write("Try block begins" + "<br>");
    // create a syntax error
    eval ("10 + * 5");
}catch(errVar){
    document.write("Exception caught" + "<br>");
    document.write("Error name: " + errVar.name + "<br>");
    document.write("Error message: " + errVar.message + "<br>");
}finally{
    document.write("Finally block reached!");
}
</script>
```