JavaScript 基础

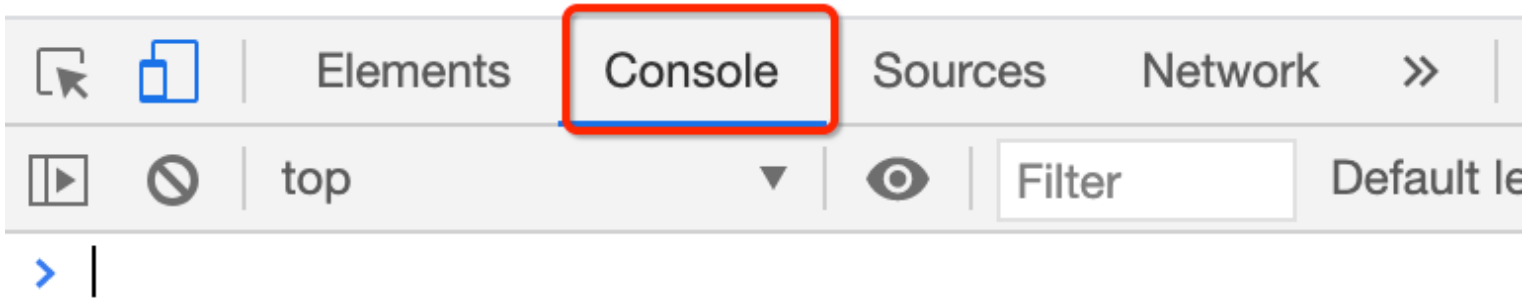
主讲人： 和凌志

JavaScript语言简介

- JavaScript是由Netscape公司开发的， 它是基于对象事件驱动的编程语言
- JavaScript开发环境简单，不需要编译器，而是直接运行在Web浏览器中

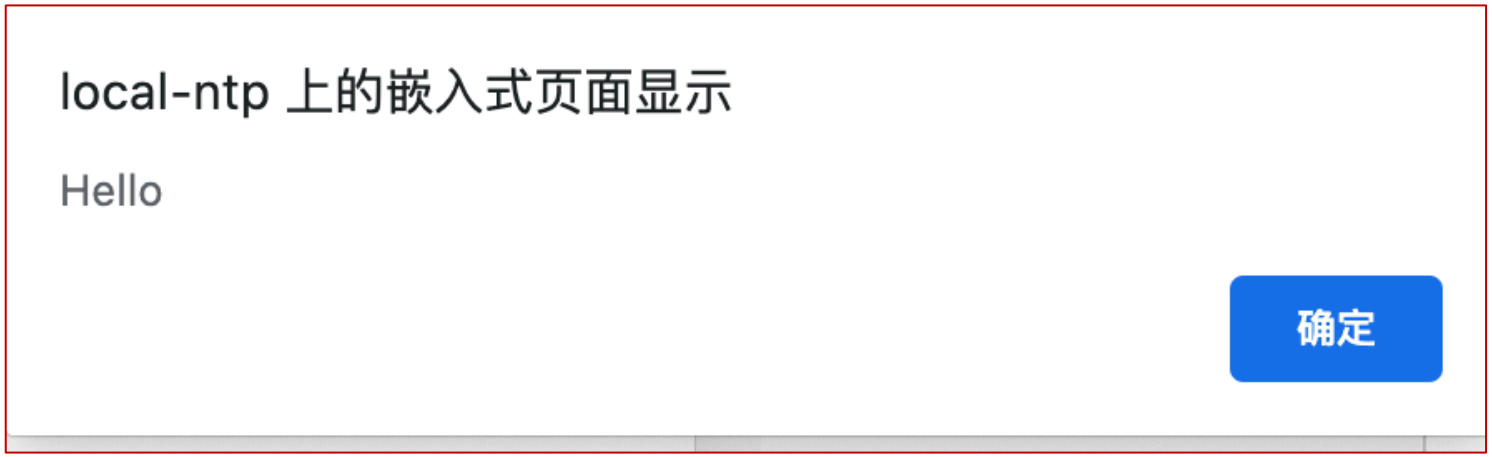
JavaScript 直接在 浏览器中运行

- 进入浏览器开发模式，Console 下



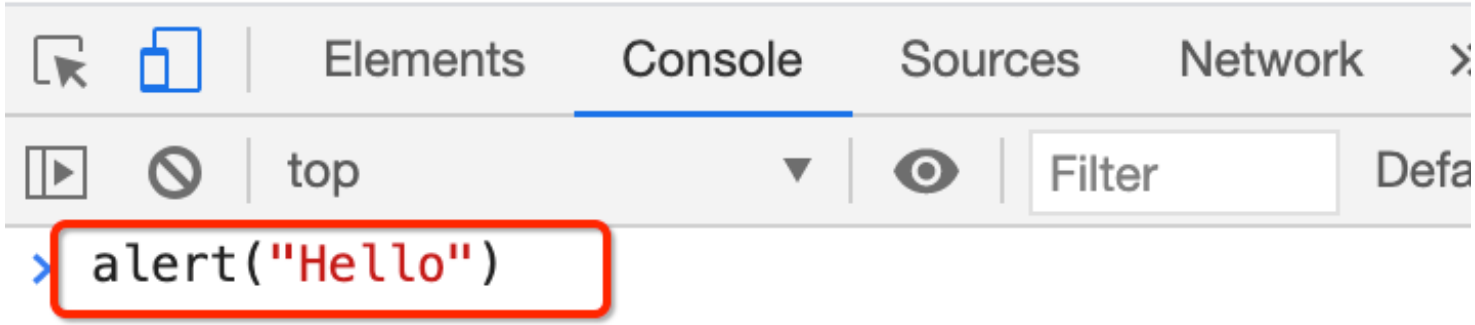
JavaScript 直接在 浏览器中运行

- 弹出提示框，如下



JavaScript 直接在 浏览器中运行

- 编写 javascript 代码



JavaScript语言简介

- JavaScript的出现，它可以使得信息和用户之间不仅只是一种显示和浏览的关系，而是一种互动的关系
- JavaScript 实现 WEB 网页实时的、动态的、可交式的表达能力

基于对象的JavaScript语言

- JavaScript语言是基于对象的（Object-Based），而不是面向对象的（object-oriented）。
- JavaScript 没有提供象抽象、继承、重载等有关面向对象语言的功能。它把复杂对象统一起来，形成了一个非常强大的对象系统。
- 虽然JavaScript语言是一门基于对象的，但它还是具有一些面向对象的基本特征。它可以根据需要创建自己的对象，从而进一步扩大JavaScript的应用范围，增强编写功能强大的Web文档。

Javascript Introduction

- JavaScript is a scripting language most often used for client-side web development.
- JavaScript is an implementation of the ECMAScript standard. The ECMAScript only defines the syntax/characteristics of the language and a basic set of commonly used objects
- JavaScript supported in the browsers typically support additional objects. e.g. Window, Frame, Form, DOM object, etc.

What can we do with JavaScript?

- To create interactive user interface in a web page (e.g., menu, pop-up alert, windows, etc.)
- Manipulating web content dynamically
  - Change the content and style of an element
  - Replace images on a page without page reload
  - Hide/Show contents
- Generate HTML contents on the fly
- Form validation
- AJAX

JavaScript 运行环境

- 内嵌在浏览器运行
- 通过 node.js 运行， 安装node.js

Embedding JavaScript

- The scripts inside an HTML document is interpreted in the order they appear in the document.
- Scripts in a function is interpreted when the function is called.

在浏览器内嵌 JavaScript

JavaScript 动手实验

浏览器 启用 JavaScript (How to enable Javascript ?)

JavaScript Syntax

```
<script>
    document.write("JavaScript is not Java");
</script>
```

疑问： 如果不用 <script> 将会怎样？

JavaScript Popup Boxes

`alert(‘ Hello World’);`　　作为函数，可以单独调用

```
<script>
 alert(' Hello World');
</script>
```

If we place the above JavaScript between the head tags (or body tags), it will run as soon as we load the page.

Case-01:

```
<script>

document.write("<hr>");
document.write("Hello World Wide Web");
document.write("<hr>");

</script>
```

Linking to an external JavaScript file

```
<script src="external_javascript.js"></script>
```

- The code in your **.js** file should be no different to the code you normally have placed in between the script tags.

- Don't need to add the **<script>** tag again — it is already on the HTML page calling the external file!

Case-01: 外部js 文件调用

- 在外部创建一个js文件
- 在 html 中， 通过 **src** 属性，引入外部 js 文件
- 注意所引入的文件路径

```
// case-01.js
document.write("<hr>");
document.write("Hello World Wide Web");
document.write("<hr>");
```

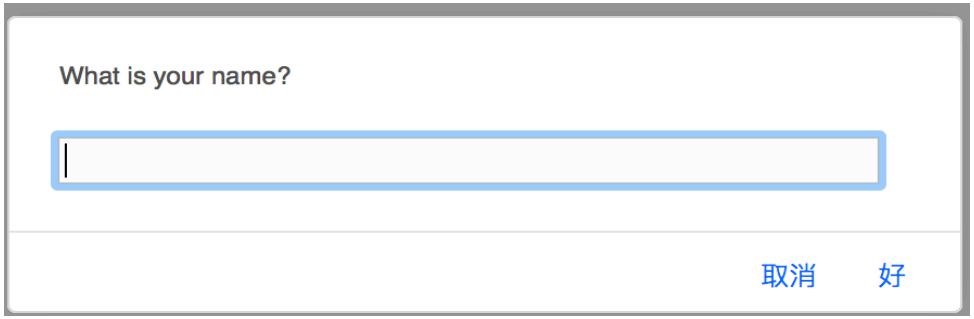
```
<!-- 引入外部js文件，注意路径 -->
<script src = "./js/case-01.js" >   </script>
```

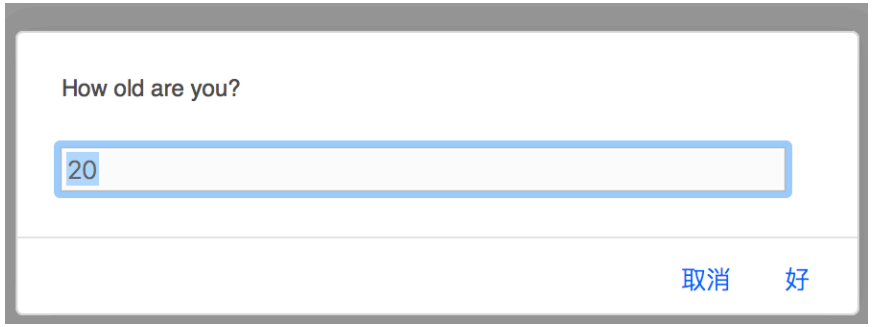
`alert()`, `confirm()`, and `prompt()`

```
<script>
  alert("This is an Alert method");
  confirm("Are you OK?");
  prompt("What is your name?");
  prompt("How old are you?", "20");
</script>
```


This is an Alert method
关闭


Are you OK?
取消
好


What is your name?
取消
好


How old are you?
20
取消
好

`alert()` and `confirm()`

```
alert("Text to be displayed");
```

- Display a message in a dialog box.
- The dialog box will block the browser.

```
var answer = confirm("Are you sure?");
```

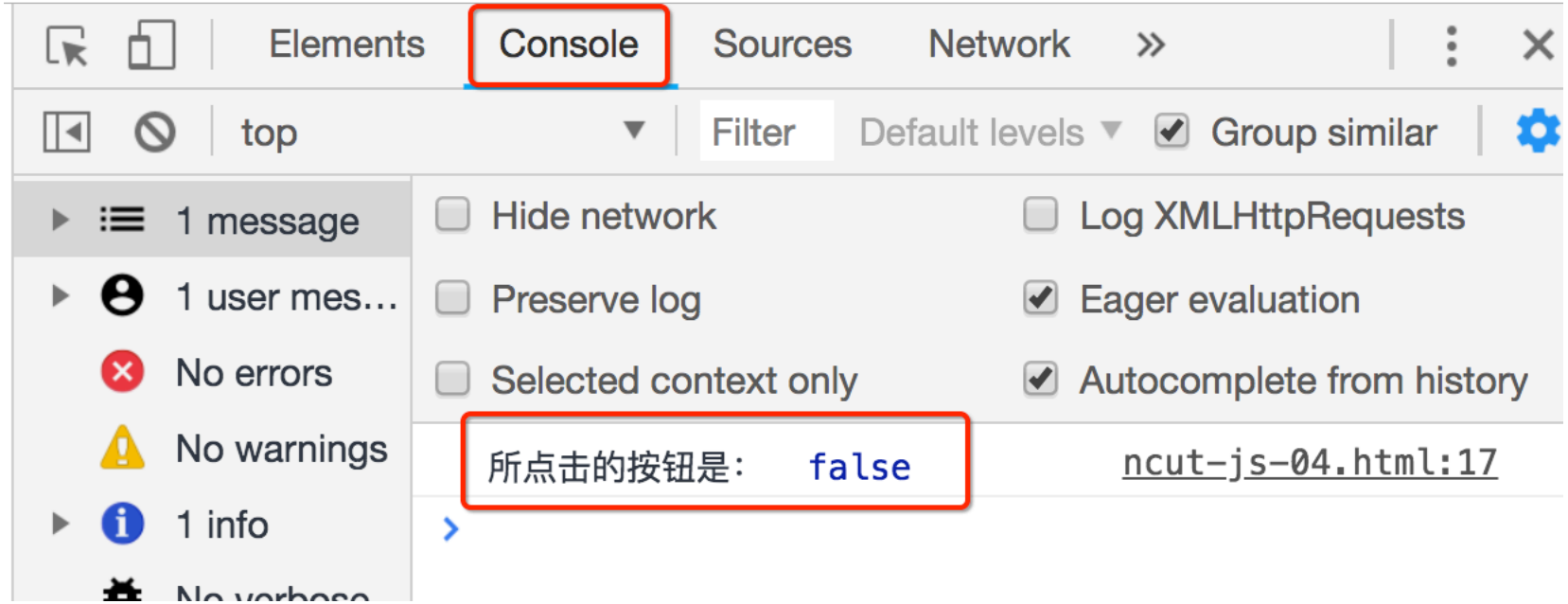
- Display a message in a dialog box with two buttons: "OK" or "Cancel".
- `confirm()` returns `true` if the user click "OK". Otherwise it returns `false`.

如何查看 JavaScript 的输出值？

```
<script>
 var answer = confirm("Are you sure?");
 console.log( "所点击的按钮是:  ", answer);
</script>
```

JavaScript 调试方法—— 基于浏览器

- 在 chrome 浏览器查看
- 右键点击网页 -> 检查



prompt()

```
prompt("What is your student id number?");
prompt("What is your name?", "No name");
```

- Display a message and allow the user to enter a value
- The second argument is the "default value" to be displayed in the input textfield.
- Without the default value, "undefined" is shown in the input textfield.
- If the user click the "OK" button, **`prompt()`** returns the value in the input textfield as a string.
- If the user click the "Cancel" button, **`prompt()`** returns null.

Javascript 语法

- Same as Java/C++ except that it allows an additional character – '$'.
- Contains only 'A' – 'Z', 'a' – 'z', '0' – '9', '_', '$'
- First character cannot be a digit
- Case-sensitive
- 只有 Html 网页代码，不区分大小写(non-case-sensitive)

Variable and Variable Declaration

- Local variable is declared using the keyword 'var'.
- Dynamic binding – a variable can hold any type of value
- If a variable is used without being declared, the variable is created automatically.
- If you misspell a variable name, program will still run (but works incorrectly)

```
<script>
  x = 3;
  var y = 4;
  alert("x=" + x + ", y=" + y);
</script>
```

Data Types

- Primitive data types
  - **Number**: integer & floating-point numbers
  - **Boolean**: true or false
  - **String**: a sequence of alphanumeric characters
- Composite data types (or Complex data types)
  - **Object:** a named collection of data
  - **Array**: a sequence of values (an array is actually a predefined object)
- Special data types
  - **Null**: the only value is "null" – to represent nothing.
  - **Undefined**: the only value is "undefined" – to represent the value of an unintialized variable

Strings

- A string variable can store a sequence of alphanumeric characters, spaces and special characters.
- Each character is represented using 16 bit
  - You can store Chinese characters in a string.
- A string can be enclosed by a pair of single quotes (') or double quote (").
- Use escaped character sequence to represent special character (e.g.: `\", \n, \t`)

typeof operator

```
var x = "hello", y;
alert("Variable x value is " + typeof x );
alert("Variable y value is " + typeof y );
alert("Variable x value is " + typeof z );
```

- Returns a string which can be:
- "number", "string", "boolean", "object", "function", "undefined", and "null"
- An array is internally represented as an object.

Object

- An object is a collection of properties.
- Properties can be variables (Fields) or Functions (Methods)
- There is no "Class" in JavaScript.

Array

- An array is represented by the **`Array`** object. To create an array of N elements, you can write

```
var myArray = new Array(N);
```

- Index of array runs from 0 to N-1.
- Can store values of different types
- Consists of various methods to manipulate its elements. e.g., **`reverse()`**, **`push()`**, **`concat()`**, etc

Array Examples

```
var Car = new Array(3);
Car[0] = "Ford";
Car[1] = "Toyota";
Car[2] = "Honda";

// Create an array of three elements with initial
// values
var Car2 = new Array("Ford", "Toyota", "Honda");

// Create an array of three elements with initial
// values
var Car3 = ["Ford", "Toyota", "Honda"];
```

```
// An array of 3 elements, each element is undefined
var tmp1 = new Array(3);

// An array of 3 elements with initial values
var tmp2 = new Array(10, 100, -3);

// An array of 3 elements with initial values
// of different types
var tmp3 = new Array(1, "a", true);

// Makes tmp3 an array of 10 elements
tmp3.length = 10; // tmp[3] to tmp[9] are undefined.

// Makes tmp3 an array of 100 elements
tmp3[99] = "Something";
// tmp[3] to tmp[98] are undefined.
```

Null & Undefined

- An undefined value is represented by the keyword "**`undefined`**".
  - It represents the value of an uninitialized variable

- The keyword "**`null`**" is used to represent “nothing”
  - Declare and define a variable as “null” if you want the variable to hold nothing.
  - Avoid leaving a variable undefined.

Type Conversion (To Boolean)

- The following values are treated as false
  - null
  - undefined
  - +0, -0
  - "" (empty string)

Type Conversion

- Converting a value to a number

```
var numberVar = someVariable - 0;
```

- Converting a value to a string

```
var stringVar = someVariable + "";
```

- Converting a value to a boolean

```
var boolVar = !!someVariable;
```

Operators

- Arithmetic operators
  - +, -, *, /, %
- Post/pre increment/decrement
  - ++, --
- Comparison operators
  - ==, !=, >, >=, <, <=
  - ===, !== (Strictly equals and strictly not equals)
    - Type and value of operand must match / must not match

== vs ===

```
// Type conversion is performed before comparison
var v1 = ("5" == 5);    // true

// No implicit type conversion.
// True if only if both types and values are equal
var v2 = ("5" === 5);   // false

var v3 = (5 === 5.0); // true

var v4 = (true == 1); // true (true is converted to 1)

var v5 = (true == 2); // false (true is converted to 1)

var v6 = (true == "1") // true
```

Logical Operators

- `!` – Logical NOT

- `&&` – Logical AND
  - `OP1 && OP2`
  - If OP1 is true, expression evaluates to the value of OP2. Otherwise the expression evaluates to the value of OP1.
  - Results may not be a boolean value.

- `||` – Logical OR
  - `OP1 || OP2`
  - If OP1 is true, expression evaluates to the value of OP1. Otherwise the expression evaluates to the value of OP2.

```
var tmp1 = null && 1000;        // tmp1 is null

var tmp2 = 1000 && 500;         // tmp2 is 500

var tmp3 = false || 500;        // tmp3 is 500

var tmp4 = "" || null;          // tmp4 is null

var tmp5 = 1000 || false;       // tmp5 is 1000


// If foo is null, undefined, false, zero, NaN,
// or an empty string, then set foo to 100;
// else = foo

foo = foo || 100;
```

Operators (continue)

- String concatenation operator
  - `+`
  - If one of the operand is a string, the other operand is automatically converted to its equivalent string value.
- Assignment operators
  - `=, +=, -=, *=, /=, %=`
- Bitwise operators
  - `&, |, ^, >>, <<, >>>`

Conditional Statements

- “if” statement
- “if … else” statement
- "? :" ternary conditional statement
- “switch” statement

- The syntax of these statements are similar to those found in C and Java.

Looping Statement

- “for” Loops
- “for/in” Loops
- “while” Loops
- “do … while” Loops
- “break” statement
- “continue” statement

- All except "for/in" loop statements have the same syntax as those found in C and Java.

“**for/in**” statement

```
for (var variable in object) {
  statements;
}
```

- To iterate through all the properties in "object".
- "**`variable`**" takes the name of each property in "object”
- Can be used to iterate all the elements in an Array object.

```
var obj = new Object();  // Creating an object

// Adding three properties to obj
obj.prop1 = 123;
obj.prop2 = "456";
obj["prop3"] = true;    // same as obj.prop3 = true

var keys = "", values = "";
for (var p in obj) {
  keys += p + " ";
  values += obj[p] + " ";
}

alert(keys);
// Show "prop1 prop2 pro3 "

alert(values);
// Show "123 456 true "
```

Example: Using for … in loop with object

Functions (Return Values)

```
// A function can return value of any type
using the keyword "return".
// The same function can possibly return values
// of different types
function foo (p1) {
      if (typeof(p1) == "number")
            return 0;   // Return a number
      else
      if (typeof(p1) == "string")
            return "zero"; // Return a string

      // If no value being explicitly returned
      // "undefined" is returned.
}

foo(1);              // returns 0
foo("abc");          // returns "zero"
foo();               // returns undefined
```

JavaScript 如何创建对象？

Creating objects using `new Object()`

```
var person = new Object();

// Assign fields to object "person"
person.firstName = "John";
person.lastName = "Doe";

// Assign a method to object "person"
person.sayHi = function() {
  alert("Hi! " + this.firstName + " " + this.lastName);
}

person.sayHi();  // Call the method in "person"
```

Creating objects using **Literal Notation (字面量)**

```
var person = {
  // Declare fields
  // (Note: Use comma to separate fields)
  firstName : "John",
  lastName : "Doe",

  // Assign a method to object "person"
  sayHi : function() {
    alert("Hi! " + this.firstName + " " +
          this.lastName);
  }
}

person.sayHi();  // Call the method in "person"
```

Creating objects using Literal Notation (Nested notation is possible)

```
var triangle = {
  // Declare fields (each as an object of two fields)
  p1 : { x : 0, y : 3 },
  p2 : { x : 1, y : 4 },
  p3 : { x : 2, y : 5 }
}

alert(triangle.p1.y);   // Show 3
```

Object Constructor and prototyping

```
function Person(fname, lname) {
  // Define and initialize fields
  this.firstName = fname;
  this.lastName = lname;

  // Define a method
  this.sayHi = function() {
    alert("Hi! " + this.firstName + " " +
          this.lastName);
  }
}

var p1 = new Person("John", "Doe");
var p2 = new Person("Jane", "Dow");

p1.sayHi();  // Show "Hi! John Doe"
p2.sayHi();  // Show "Hi! Jane Dow"
```

Adding methods to objects using prototype

```
// Suppose we have defined the constructor "Person"
// (as in the previous slide).

var p1 = new Person("John", "Doe");
var p2 = new Person("Jane", "Dow");

// Aattaching a new method to all instances of Person
Person.prototype.sayHello = function() {
  alert("Hello! " + this.firstName + " " +
                    this.lastName);
}

// We can also introduce new fields via "prototype"

p1.sayHello();  // Show "Hello! John Doe"
p2.sayHello();  // Show "Hello! Jane Dow"
```

Events （事件）

- An event occurs as a result of some activity
  - A user clicks on a link in a page
  - Page finished loaded
  - Mouse cursor enter an area
  - A preset amount of time elapses
  - A form is being submitted

Event Handlers

- Event Handler – a segment of codes (usually a function) to be executed when an event occurs
- We can specify event handlers as attributes in the HTML tags.
- The attribute names typically take the form "**onXXX**" where **XXX** is the event name.
  - e.g.:

```
<a href="…" onClick="alert('Bye')">Other Website</a>
```

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
onDblClick	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

`onSubmit` Event Handler Example

```
<html><head>
<title>onSubmit Event Handler Example</title>
<script type="text/javascript">
  function validate() {
    // If everything is ok, return true
    // Otherwise return false
  }
</script>
</head>
<body>
<form action="MessageBoard" method="POST"
 onSubmit="return validate();"
>
…
</form></body></html>
```

- If **`onSubmit`** event handler returns false, data is not submitted.
- If **`onReset`** event handler returns false, form is not reset

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

String Object (Some useful methods)

- length
  - A string property that tells the number of character in the string
- charAt(idx)
  - Returns the character at location "idx"
- toUpperCase(), toLowerCase()
  - Returns the same string with all uppercase/lowercase letters
- substring(beginIdx)
  - Returns a substring started at location "beginIdx"
- substring(beginIdx, endIdx)
  - Returns a substring started at "beginIdx" until "endIdx" (but not including "endIdx"
- indexOf(str)
  - Returns the position where "str" first occurs in the string

Error and Exception Handling in JavaScript

- Javascript makes no distinction between Error and Exception (Unlike Java)

- Handling Exceptions
  - The onError event handler
    - A method associated with the window object.
    - It is called whenever an exception occurs
  - The try … catch … finally block
    - Similar to Java try … catch … finally block
    - For handling exceptions in a code segment
  - Use throw statement to throw an exception
    - You can throw value of any type
  - The Error object
    - Default object for representing an exception
    - Each Error object has a name and message properties

try … catch … finally

```
try {
  // Contains normal codes that might throw an exception.

  // If an exception is thrown, immediately go to
  //   catch block.

} catch ( errorVariable ) {
  // Codes here get executed if an exception is thrown
  //   in the try block.

  // The errorVariable is an Error object.

} finally {
  // Executed after the catch or try block finish

  // Codes in finally block are always executed
}
// One or both of catch and finally blocks must accompany the try
block.
```

Javascript 高级

- 函数声明与函数表达式
- 在 node.js 环境运行 JavaScript

函数表达式

```
var sum = function (num1, num2)
{
    return (num1 + num2) ;

  }

console.log (sum (10,10));
```

Q & A