

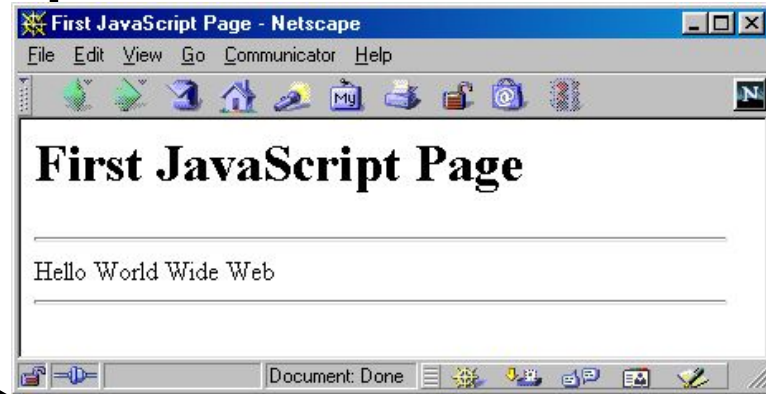
Lecture 7

JavaScript

Senior- Lecturer: Sarsenova Zh.N.

A simple Script

```
<html>
<head><title>First JavaScript
  Page</title></head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
  document.write("<hr>");
  document.write("Hello World Wide
  Web");
  document.write("<hr>");
</script>
</body>
</html>
```



Embedding JavaScript

```
<html>
<head><title>First JavaScript Program</title></head>
<body>
<script type="text/javascript"
        src="your_source_file.js"></script>
</body>
</html>
```

[Inside your_source_file.js](#)

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

- Use the **src** attribute to include JavaScript codes from an external file.
- The included code is inserted in place.

Popup Boxes

- Alert box
- Confirm box
- Prompts box

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

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



The typeof operator

- **typeof** operator used to find the type of a JavaScript variable.

JavaScript typeof

The typeof operator returns the type of a variable or an expression.

```
string  
string  
string
```

```
<!DOCTYPE html>  
<html>  
<body>  
  
<h2>JavaScript typeof</h2>  
<p>The typeof operator returns the type of  
a variable or an expression.</p>  
  
<p id="demo"></p>  
  
<script>  
document.getElementById("demo").innerHTML =  
typeof "" + "<br>" +  
typeof "John" + "<br>" +  
typeof "John Doe";  
</script>  
</body>  
</html>
```

Example

```
<script>
document.getElementById("demo").innerHTML =
typeof 0 + "<br>" +
typeof 314 + "<br>" +
typeof 3.14 + "<br>" +
typeof (3) + "<br>" +
typeof (3 + 4);
</script>
```

JavaScript typeof

The typeof operator returns the type of a variable or an expression.

number
number
number
number
number

Undefined data type

- In JavaScript, a variable without a value, has the value **undefined**. The typeof is also **undefined**.

```
<script>
var car;
document.getElementById("demo").innerHTML =
car + "<br>" + typeof car;
</script>
```

```
undefined
undefined
```


Null

- In JavaScript null is "nothing". It is supposed to be something that doesn't exist.
- Unfortunately, in JavaScript, the data type of null is an object.

```
<script>
var person = {firstName:"John",
lastName:"Doe", age:50, eyeColor:"blue"};
person = null;
document.getElementById("demo").innerHTML =
typeof person;
</script>
```

object

Primitive Data

```
<script>
document.getElementById("demo").innerHTML =
typeof "john" + "<br>" +
typeof 3.14 + "<br>" +
typeof true + "<br>" +
typeof false + "<br>" +
typeof x;
</script>
```

string
number
boolean
boolean
undefined

Complex Data

```
<script>
document.getElementById("demo").innerHTML =
typeof {name:'john', age:34} + "<br>" +
typeof [1,2,3,4] + "<br>" +
typeof null + "<br>" +
typeof function myFunc(){};
</script>
```

object
object
object
function

Line Breaks

- To display line breaks inside a popup box, use a back –slash followed by the character n.
- Example: `alert(“Hello\n How are you?”);`

JavaScript Objects

- Objects in real life is for instance Students
- Properties: name, ID, weigh, height etc.
- Methods: eat, speak, walk, read, do something and etc.

JavaScript Objects

- You have already learned that JavaScript variables are containers for data values.

```
<!DOCTYPE html>
<html>
<body>

<p>Creating a JavaScript Variable.</p>

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

<script>
var car = "Fiat";
document.getElementById("demo").innerHTML =
car;
</script>

</body>
</html>
```

Creating a JavaScript Variable.

Fiat

Objects

- Objects are variables too.
- Objects can contain many values

Example

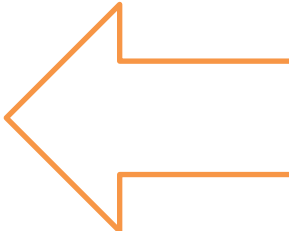
```
<!DOCTYPE html>
<html>
<body>

<p>Creating a JavaScript Object.</p>

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

<script>
var car = {type:"Fiat", model:"500", color:"white"};
document.getElementById("demo").innerHTML = car.type;
</script>

</body>
</html>
```



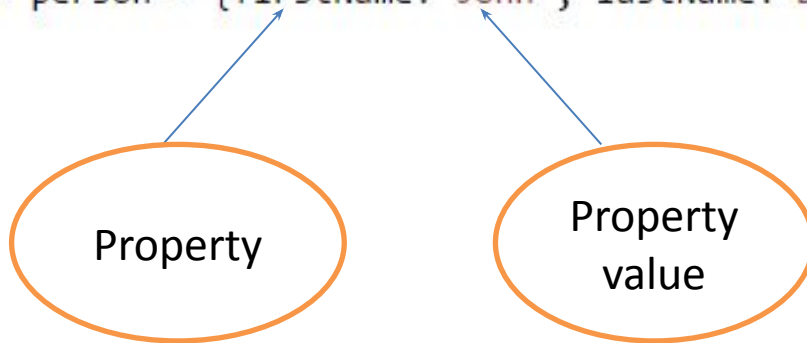
This code assigns **many values** (Fiat, 500, white) to a variable named **car**

The values are written as **name: value** pairs (name and value separated by a colon)

Object Properties

- The name: values pairs are called properties.

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```



Object Methods

- Methods are actions that can be performed on objects.
- Methods are stored in properties as function definitions

Example

```
<!DOCTYPE html>
<html>
<body>

<p>Creating a JavaScript Object.</p>

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

<script>
var person = {firstName:"John", lastName:"Doe",
age:50, eyeColor:"blue"};

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

</body>
</html>
```

Creating a JavaScript Object.

John is 50 years old.

Accessing Object Properties

- 2 ways

`objectName.propertyName`

```
document.getElementById("demo").innerHTML =  
person.firstName + " is " + person.age + " years  
old.";
```

Or

`objectName["propertyName"]`

```
document.getElementById("demo").innerHTML =  
person["firstName"] + " " + person["id"];  
</script>
```

Accessing Object methods

- `ObjectName.methodName()`

```
<script>
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

```
document.getElementById("demo").innerHTML =
person.fullName();
```

Creating and using an object method.

An object method is a function definition, stored as a property value.

John Doe

Example

An object method is a function definition, stored as a property value.

If you access it without (), it will return the function definition:

```
function () { return this.firstName + " " + this.lastName; }
```

```
<script>
var person = {
  firstName: "John",
  lastName : "Doe",
  id       : 5566,
  fullName : function() {
    return this.firstName + " " + this.lastName;
  }
};
```

```
document.getElementById("demo").innerHTML =
person.fullName;
```

Do Not Declare Strings, Numbers, and Booleans as Objects!

- When a JavaScript variable is declared with the keyword "new", the variable is created as an object:

```
var x = new String();      // Declares x as a String object
var y = new Number();     // Declares y as a Number object
var z = new Boolean();    // Declares z as a Boolean object
```

Conditional Statements

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

Types of conditional statements

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

If statement Syntax and example

- If (expression) {

Statements to be executed if expression is true

}

```
<script type="text/javascript">
  <!--
    var age = 20;

    if( age > 18 ){
      document.write("<b>Qualifies for driving</b>");
    }
  //-->
</script>
```

If .. Else statement

```
if (expression){  
    Statement(s) to be executed if expression is true  
}  
  
else{  
    Statement(s) to be executed if expression is false  
}
```

```
<script type="text/javascript">  
    <!--  
        var age = 15;  
  
        if( age > 18 ){  
            document.write("<b>Qualifies for driving</b>");  
        }  
  
        else{  
            document.write("<b>Does not qualify for driving</b>");  
        }  
        //-->  
</script>
```

Does not qualify for driving

Set the variable to different value and then try...

If...else if... statement syntax

```
if (expression 1){  
    Statement(s) to be executed if expression 1 is true  
}  
  
else if (expression 2){  
    Statement(s) to be executed if expression 2 is true  
}  
  
else if (expression 3){  
    Statement(s) to be executed if expression 3 is true  
}  
  
else{  
    Statement(s) to be executed if no expression is true  
}
```

If...else if... statement example

```
<script type="text/javascript">
  <!--
    var book = "maths";
    if( book == "history" ){
      document.write("<b>History Book</b>");
    }

    else if( book == "maths" ){
      document.write("<b>Maths Book</b>");
    }

    else if( book == "economics" ){
      document.write("<b>Economics Book</b>");
    }

    else{
      document.write("<b>Unknown Book</b>");
    }
  //-->
</script>
```

Maths Book

Switch Statement

- You should use the Switch statement if you want to select one of many blocks of code to be executed.

Switch statement's syntax

```
switch(expression) {  
    case n:  
        code block  
        break;  
    case n:  
        code block  
        break;  
    default:  
        default code block  
}
```

```
<!DOCTYPE html>
<html>
<body>

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

<script>
var day;
switch (new Date().getDay()) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
  case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case 6:
    day = "Saturday";
}
document.getElementById("demo").innerHTML = "Today is " + day;
</script>

</body>
</html>
```

Today is Monday

The break Keyword

- When JavaScript reaches a **break** keyword, it breaks out of the switch block.
- This will stop the execution of more code and case testing inside the block.
- When a match is found, and the job is done, it's time for a break. There is no need for more testing

The default Keyword

- The **default** keyword specifies the code to run if there is no case match:

```
<script>
var text;
switch (new Date().getDay()) {
  case 6:
    text = "Today is Saturday";
    break;
  case 0:
    text = "Today is Sunday";
    break;
  default:
    text = "Looking forward to the Weekend";
}
document.getElementById("demo").innerHTML = text;
</script>
```

Looking forward to the Weekend

JavaScript Math Objects

- allows you to perform mathematical tasks on numbers
- **Math.round(x)** returns the value of x rounded to its nearest integer
- `Math.round(4.7); // returns 5`
`Math.round(4.4); // returns 4:`
- **Math.pow(x, y)** returns the value of x to the power of y:
- `Math.pow(8, 2); // returns 64`
- **Math.sqrt(x)** returns the square root of x:
- `Math.sqrt(64); // returns 8`
- **Math.abs(x)** returns the absolute (positive) value of x:
- `Math.abs(-4.7); // returns 4.7`
- **Math.ceil(x)** returns the value of x rounded **up** to its nearest integer:
- `Math.ceil(4.4); // returns 5`
- **Math.floor(x)** returns the value of x rounded **down** to its nearest integer:
- `Math.floor(4.7); // returns 4`
- **Math.min()** and **Math.max()** can be used to find the lowest or highest value in a list of arguments:
- `Math.min(0, 150, 30, 20, -8, -200); // returns -200`
- `Math.max(0, 150, 30, 20, -8, -200); // returns 150`
- **Math.random()** returns a random number between 0 (inclusive), and 1 (exclusive):
- `Math.random();`

Math Properties (Constants)

- `Math.E` // returns Euler's number
- `Math.PI` // returns PI
- `Math.SQRT2` // returns the square root of 2
- `Math.SQRT1_2` // returns the square root of 1/2
- `Math.LN2` // returns the natural logarithm of 2
- `Math.LN10` // returns the natural logarithm of 10
- `Math.LOG2E` // returns base 2 logarithm of E
- `Math.LOG10E` // returns base 10 logarithm of E

JavaScript Random Integers

- `Math.random()` used with `Math.floor()` can be used to return random integers.
- `Math.floor(Math.random() * 10);` //returns a number between 0 and 9

Home work: read Chapter 15