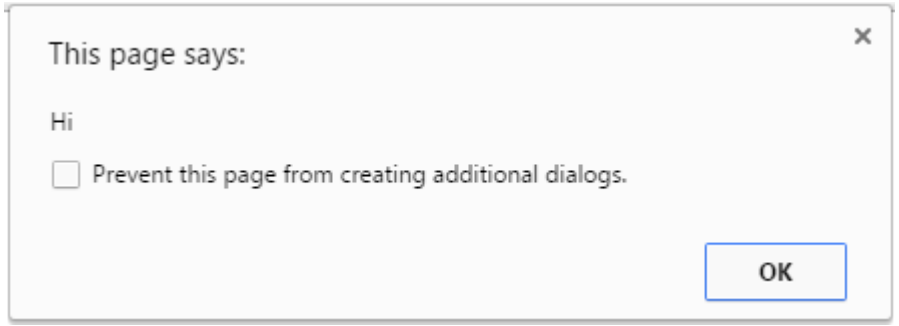
CSIT128 / CSIT828

The Basics of JavaScript

Joseph Tonien

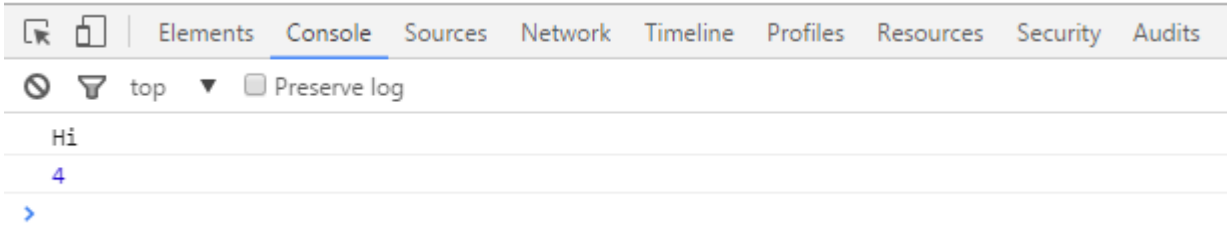
My First JavaScript

```
<button type="button" onclick="alert('Hi');">
Click me
</button>

<button type="button" onclick="alert(1+1);">
Click me
</button>
```



My First JavaScript

```
<button type="button" onclick="console.log('Hi');">
Click me
</button>

<button type="button" onclick="console.log(2+2);">
Click me
</button>
```



My First JavaScript

```
<button type="button" onclick="alert('Hi'); console.log(2+2);">
Click me
</button>
```

My First JavaScript

```
<button type="button" onclick="sayHi();">
Click me
</button>

<script>
function sayHi(){
  alert("Hi");
}
</script>
```

Where to include JavaScript

You can put you JavaScript anywhere in your HTML file.

Common practice:

- In the head
- At the end of body

```
<script>

function sayHi(){
  alert("Hi");
}

</script>
```

Where to include JavaScript

In the head

```
<head>
<title>JavaScript Example</title>
<script>
function sayHi(){
  alert("Hi");
}
</script>
</head>
```

Where to include JavaScript

At the end of body (just before the closing body tag)

```
...
<script>
function sayHi(){
  alert("Hi");
}
</script>
</body>
</html>
```

External JavaScript

Instead of putting javascript code inside your html file

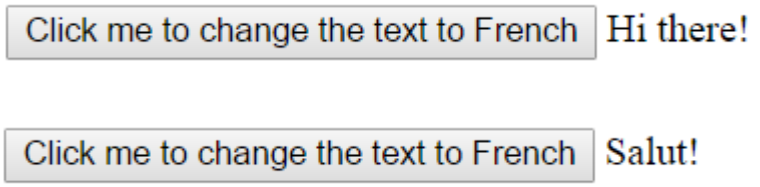
```
<script>

function sayHi(){
  alert("Hi");
}

</script>
```

you can specify an external javascript file:

```
<script type="text/javascript" src="js/myscript.js"></script>
```

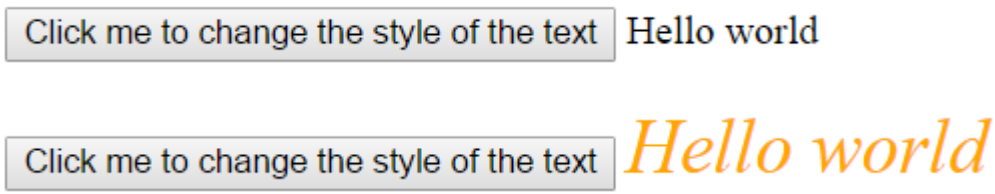
Change content by JavaScript

```
<button type="button" onclick="changeToFrench();">
Click me to change the text to French
</button>
<span id="french">Hi there!</span>


<script>

function changeToFrench(){
  document.getElementById("french").innerHTML = "Salut!";
}
</script>
```



Change style by JavaScript

```
<button type="button" onclick="changeHelloWorldStyle();">
Click me to change the style of the text
</button>
<span id="hello">Hello world</span>

<script>

function changeHelloWorldStyle(){
  var e = document.getElementById("hello");
  e.style.color = "orange";
  e.style.fontSize = "30px";
  e.style.fontStyle = "italic";
}

</script>
```



Basic JavaScript syntax

JavaScript statements are separated by semicolons

```
function changeHelloWorldStyle(){
  var e = document.getElementById("hello");
  e.style.color = "orange";
  e.style.fontSize = "30px";
  e.style.fontStyle = "italic";
}
```

Basic JavaScript syntax

JavaScript Comments

Code after double slashes // or between /* and */ is treated as a comment.

Comments are ignored, and will not be executed.

```
/*
this function change the style of the text
 */
function changeHelloWorldStyle(){
  //get the html element
  var e = document.getElementById("hello");
  //change the style
  e.style.color = "orange";
  e.style.fontSize = "30px";
  e.style.fontStyle = "italic";
}
```

Basic JavaScript syntax

JavaScript uses the `var` keyword to declare variables.

```
var studentName = "John";
var x, y;
x = 5;
y = x + 2;
```

All JavaScript identifiers are **case sensitive**.

- The variables studentName and StudentName are two different variables.
- The variables x and X are two different variables.

Basic JavaScript syntax

Variable naming: two good conventions

underscore:

student_name, student_id, first_name, last_name

camel case:

studentName, studentId, firstName, lastName

Basic JavaScript syntax

JavaScript has dynamic types.

!!!

This means that the same variable can be used as **different types**:

```
var x;                    // x is undefined
alert(x);

var x = 2016;             // x is a number
alert(x);

var x = "Wollongong";     // x is a string
alert(x);
```

A variable declared without a value will have the value **undefined**.

Basic JavaScript syntax

JavaScript data type: number

```
var age = 19;
var pi = 3.14;
var x;

alert(typeof age);    //number
alert(typeof pi);     //number
alert(x);             //undefined
```

Basic JavaScript syntax

Arithmetic operators are used to perform arithmetic on numbers

```
+       Addition
-       Subtraction
*       Multiplication
/       Division
%       Modulus
++      Increment
--      Decrement
```

Basic JavaScript syntax

Assignment operators

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Basic JavaScript syntax

JavaScript data type: string

```
var age = "19";
var name = "John";
var x;

alert(typeof age);     //string
alert(typeof name);    //string
alert(x);              //undefined
```

Basic JavaScript syntax

Strings are text, written within double or single quotes:

```
var firstName, lastName, fullName;

firstName = "John";       // using double quotes
lastName = 'Lee';         // using single quotes

fullName = firstName + " " + lastName;
alert(fullName);
```

Use + for string concatenation

Basic JavaScript syntax

Mixing between double or single quotes:

```
var x;

x = "I'm John";                 //single quote inside double quotes
alert(x);


x = "My name is 'John'"; //single quotes inside double quotes
alert(x);


x = 'My name is "John"'; //double quotes inside single quotes
alert(x);
```

Basic JavaScript syntax

JavaScript evaluates expressions from left to right

```
var x;

x = 2016 + "Wollongong";        //2016Wollongong
alert(x);

x = 2016 + 1 + "Wollongong";    //2017Wollongong
alert(x);

x = "Wollongong" + 2016;        //Wollongong2016
alert(x);

x = "Wollongong" + 2016 + 1;    //Wollongong20161
alert(x);
```

Basic JavaScript syntax

JavaScript data type: boolean

```
var authenticated = false;
var isReturningUser = true;
var x;

alert(typeof authenticated);      //boolean
alert(typeof isReturningUser);    //boolean
alert(x);                         //undefined
```

Basic JavaScript syntax

JavaScript data type: boolean

```
var x = 5;
var positive = (x > 0);

alert(typeof positive);  //boolean

if(positive){
  alert("x is positive");
}
```

Basic JavaScript syntax

Comparison and Logical Operators

```
==        equal to
===       equal value and equal type
!=        not equal
!==       not equal value or not equal type
>         greater than
<         less than
>=        greater than or equal to
<=        less than or equal to
?         ternary operator
```

Basic JavaScript syntax

```
==        equal to
===       equal value and equal type


var x = 5;
var y = "5";

if(x == y){
  alert("yes");
}else{
  alert("no");
}
```

Basic JavaScript syntax

```
==        equal to
===       equal value and equal type


var x = 5;
var y = "5";

if(x === y){
  alert("yes");
}else{
  alert("no");
}
```

Basic JavaScript syntax

```
!=        not equal
!==       not equal value or not equal type


var x = 5;
var y = "5";

if(x != y){
  alert("yes");
}else{
  alert("no");
}
```

Basic JavaScript syntax

```
!=        not equal
!==       not equal value or not equal type


var x = 5;
var y = "5";

if(x !== y){
  alert("yes");
}else{
  alert("no");
}
```

Basic JavaScript syntax

```
?         ternary operator
                  test ? expression1 : expression2


var x = 5;
var y = "x is " + ( x%2==0 ? "even" : "odd" );


alert(y);
```

Date

```
var now = new Date();  //current date & time

alert(now);
alert(typeof now);     //object
```

Date

There are several ways to create a Date object.

```
var d = new Date();
var d = new Date(millisec);
var d = new Date(dateString);
var d = new Date(year, month, day, hour, min, sec, millisec);
```

Date

```
var d = new Date(millisec);
```

Dates are calculated in milliseconds from 01 January, 1970 00:00:00 Universal Time (UTC). One day contains 86,400,000 millisecond.

```
var d = new Date(86400000);
alert(d);     //02 Jan 1970 00:00:00 UTC
```

Date

```
var d = new Date(dateString);

//using YYYY-MM-DD format
var d = new Date("2000-01-30");
alert(d);

//using YYYY-MM-DDTHH:MI:SS
var d = new Date("2000-01-30T10:00:00");
alert(d);
```

Date

```
var d = new Date(year, month, day, hour, min, sec, millisec);
```

The last 4 parameters can be omitted.

Months count from 0 to 11. January is 0. December is 11.

```
var d = new Date(2000, 0, 1);   // 01 Jan 2000
alert(d);
```

Date

`getDate()`	Get the day as a number (1-31)
`getDay()`	Get the weekday as a number (**0-6**) *Sunday is 0, Saturday is 6*
`getFullYear()`	Get the four digit year (yyyy)
`getHours()`	Get the hour (0-23)
`getMilliseconds()`	Get the milliseconds (0-999)
`getMinutes()`	Get the minutes (0-59)
`getMonth()`	Get the month (**0-11**) *January is 0, December is 11*
`getSeconds()`	Get the seconds (0-59)
`getTime()`	Get the milliseconds since 01/Jan/1970

Date

```
var now = new Date();
alert("now is " + now);
alert("getDate returns " + now.getDate());
alert("getDay returns " + now.getDay());
alert("getFullYear returns " + now.getFullYear());
alert("getHours returns " + now.getHours());
alert("getMilliseconds returns " + now.getMilliseconds());
alert("getMinutes returns " + now.getMinutes());
alert("getMonth returns " + now.getMonth());
alert("getSeconds returns " + now.getSeconds());
alert("getTime returns " + now.getTime());
```

Date

`setDate()`	Set the day as a number (1-31)
`setFullYear()`	Set the year (optionally month and day)
`setHours()`	Set the hour (0-23)
`setMilliseconds()`	Set the milliseconds (0-999)
`setMinutes()`	Set the minutes (0-59)
`setMonth()`	Set the month (0-11)
`setSeconds()`	Set the seconds (0-59)
`setTime()`	Set the milliseconds since 01/Jan/1970

Date

```
var now = new Date();
alert(now);

var tomorrow = new Date();
tomorrow.setDate(now.getDate() + 1);
alert(tomorrow);

var hundredDayAgo = new Date();
hundredDayAgo.setDate(now.getDate() - 100);
alert(hundredDayAgo);
```

String

```
var text = "One Fish, Two Fish, Red Fish, Blue Fish";

var textLength = text.length;
                    → 39


var upper = text.toUpperCase();
                    → ONE FISH, TWO FISH, RED FISH, BLUE FISH


var lower = text.toLowerCase();
                    → one fish, two fish, red fish, blue fish
```

String

```
var text = "One Fish, Two Fish, Red Fish, Blue Fish";

var fishIndex = text.indexOf("Fish");          → 4
var catIndex = text.indexOf("cat");            → -1

var redFound = text.includes("Red");           → true
var greenFound = text.includes("Green");       → false
```

String

```
var text = "One Fish, Two Fish, Red Fish, Blue Fish";

var s1 = text.slice(10, 12);   → Tw

var s2 = text.slice(10);       → Two Fish, Red Fish, Blue Fish

var s3 = text.slice(-9, -6);   → Blu

var s4 = text.slice(-9);       → Blue Fish
```

Array

```
var arrayName = [item0, item1, ...];
```

```
var subjects = ["ISIT206", "MATH121", "CSCI301"];

alert(subjects);          //ISIT206,MATH121,CSCI301

alert(typeof subjects);   //object
```

Array

```
var arrayName = [item0, item1, ...];
```

Array can contain items of different types !!!

```
var info = ["John", new Date("1996-01-20"), "CSCI204", 85];

alert(info);                    //John,Sat Jan 20 1996...,CSCI204,85

alert(typeof info);             //object
```

Array

```
var subjects = ["ISIT206", "MATH121", "CSCI301"];

subjects[1] = "LOGIC101";  //change the content of item 1
subjects[3] = "LAW201";    //add new item 3

alert(subjects[0]);      //ISIT206
alert(subjects[1]);      //LOGIC101
alert(subjects[2]);      //CSCI301
alert(subjects[3]);      //LAW201
```

Array

Length of array

```
var subjects = ["ISIT206", "MATH121", "CSCI301"];

subjects[1] = "LOGIC101";
subjects[3] = "LAW201";

// loop through an array
for(var i = 0; i < subjects.length; i++){
  alert(subjects[i]);
}
```

Array

```
var square = [];  //empty array

for(var i = 0; i < 10; i++) {
    square[i] = i*i;
}

for(var i = 0; i < square.length; i++) {
    alert(square[i]);
}
```

Array

The `push()` method adds a new element to the end of an array

```
var square = [];  //empty array

for(var i = 0; i < 10; i++) {
    square.push(i*i);
}

for(var i = 0; i < square.length; i++) {
    alert(square[i]);
}
```

Object

```
var info = {name: "John", dob: new Date("1996-01-20"), year: 2};

alert(info);            //[object Object]

alert(typeof info);     //object
```

Object is defined by a list of `property:value`

```
var objectName = {property1:value1, property2:value2, ...};
```

Object

Access the values of an object

```
var info = {
  name: "John",
  dob: new Date("1996-01-20"),
  year: 2
};  //it is better to write this way

alert(info.name);       //John
alert(info["name"]);    //John
```

Object values can be obtained by **two ways**:

```
obj.property

obj["property"]
```

Object

Change the values of an object

```
var info = {
  name: "John",
  dob: new Date("1996-01-20"),
  year: 2
};

// two ways:
info.year = 1;
info["year"] = 1;
```

Object

Delete object properties

```
var info = {
  name: "John",
  dob: new Date("1996-01-20"),
  year: 2
};

// two ways:
delete info.year;
delete info["year"];
```

Object

Create an empty object

```
var info = { };


info.firstName = "John";
info.lastName = "Lee";


alert(info["firstName"]);
alert(info.lastName);
```

Array vs Object

```
var arrayName = [item0, item1, ...];

var objectName = {property1:value1, property2:value2, ...};
```

Arrays use numbered index:

```
arrayName[0] = "LOGIC101";
arrayName[1] = "CSCI111";
```

Objects use named index:

```
objectName["firstName"] = "John";
objectName.lastName = "Lee";
```

Array Sorting

```
var subjects = ["ISIT206", "MATH121", "CSCI301"];
subjects.sort();
```

Now subjects is ["CSCI301", "ISIT206", "MATH121"]

```
var numbers = [1, 20, -3, 4];
numbers.sort();
```

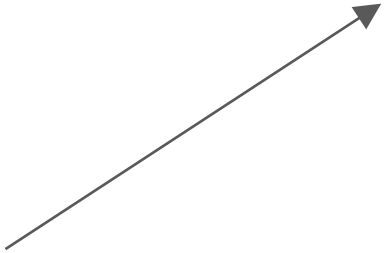
Now numbers is [-3, 1, 20, 4] !!!

```
numbers.sort(function (a, b) {  return a - b;  });
```

Now numbers is [-3, 1, 4, 20]

Array Sorting

```
var numbers = [1, 20, -3, 4];
numbers.sort(function (a, b) {  return a - b;  });
Now numbers is [-3, 1, 4, 20]

In general:
the_array_to_be_sorted.sort(the_sorting_function ...);
```



The sorting function `function (a, b)` must

- return a **positive value** to indicate `a > b`
- return a **negative value** to indicate `a < b`
- return **zero** to indicate `a = b`

Array Sorting

```
ninja_results = [
  {name: "John", level: 4, seconds: 85},
  {name: "Peter", level: 2, seconds: 35},
  {name: "Kate", level: 4, seconds: 80},
  {name: "Luke", level: 5, seconds: 120}
];
```

We want to sort the ninja results based on the level first, if two persons achieved the same level, then we compare the number of seconds.

http://www.uow.edu.au/~dong/w3/example/js/ninja.html

Array Sorting

```
ninja_results.sort(
  function (p1, p2) {
    if (p1["level"] > p2["level"]){
          return 1; // sort
        }
        if (p1["level"] < p2["level"]){
          return -1; // don't sort
        }
    //at this point the two persons have the same level
        if (p1["seconds"] < p2["seconds"]){
          return 1; // sort
        }
        if (p1["seconds"] > p2["seconds"]){
          return -1; // don't sort
        }
        return 0;
  }
);
```

Before sorting

```
ninja_results = [
  {name: "John", level: 4, seconds: 85},
  {name: "Peter", level: 2, seconds: 35},
  {name: "Kate", level: 4, seconds: 80},
  {name: "Luke", level: 5, seconds: 120}
];
```

After sorting

```
ninja_results = [
  {name: "Peter", level: 2, seconds: 35},
  {name: "John", level: 4, seconds: 85},
  {name: "Kate", level: 4, seconds: 80},
  {name: "Luke", level: 5, seconds: 120}
];
```

Confirm box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel".

If the user clicks "OK", the box returns true.

If the user clicks "Cancel", the box returns false.

```
var ok = confirm("Do you want to proceed with the order?");
if(ok){
  alert("User clicked OK");
}else{
  alert("User clicked Cancel.");
}
```

Prompt box

When a prompt box pops up, the user will have to click either "OK" or "Cancel".

If the user clicks "OK" the box returns the input value.

If the user clicks "Cancel" the box returns null.

We can also specify the default text in the input box:

```
prompt("sometext","defaultText");
```

```
var name = prompt("Please enter your name", "cat in the hat");
if(name != null){
  alert("Hello " + name);
}
```

References

`http://www.w3schools.com/js`

Robert W. Sebesta, *Programming the World Wide Web*, Pearson.