Week 0 Unit 2: **Do You Really Understand JavaScript?**

Do You Really Understand JavaScript?

Recommended JavaScript tutorials

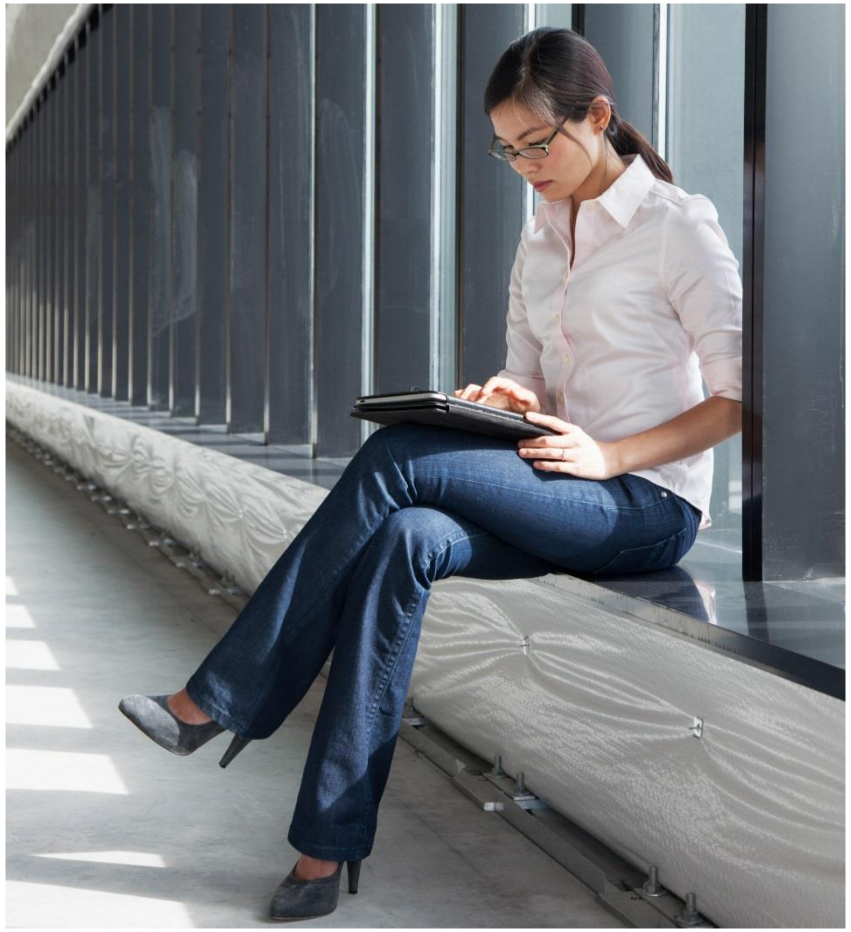
If you are fairly new to JavaScript, you might want to read one of the following free tutorials during the preparation week of this course:

Codecademy: JavaScript Fundamentals

The JavaScript Tutorial

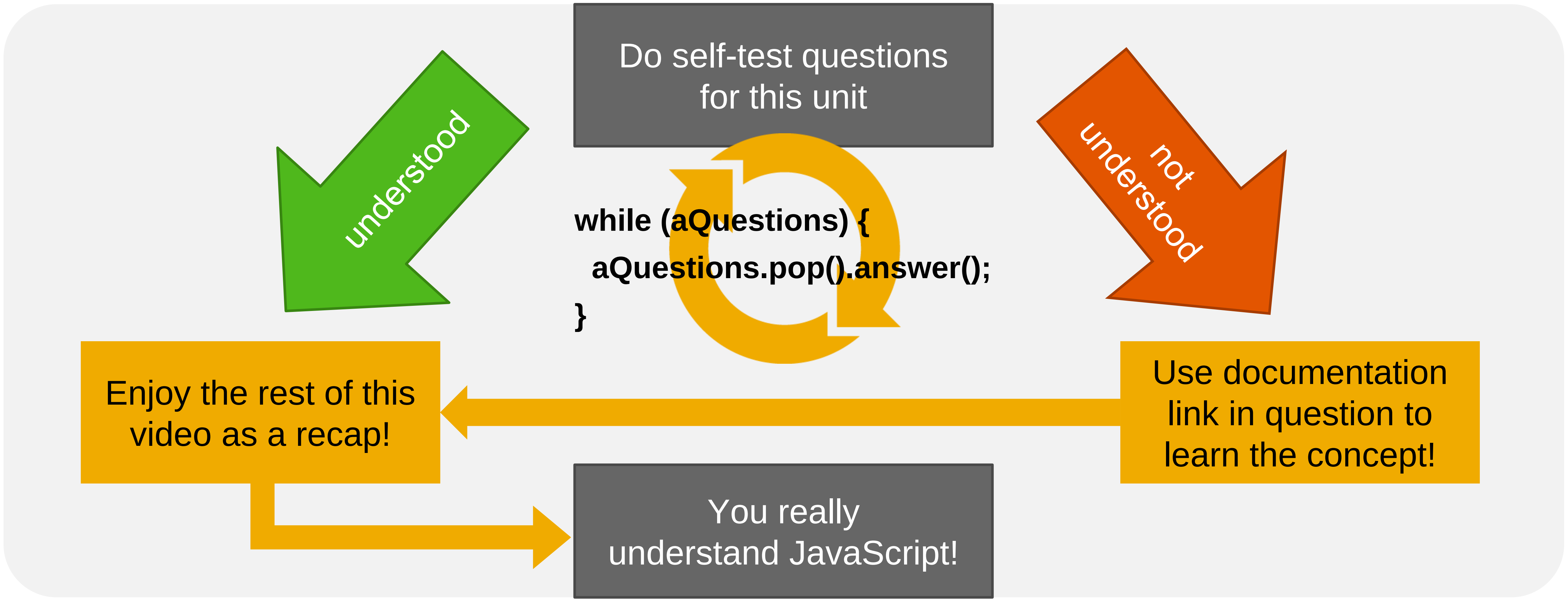
JavaScript Garden

JavaScript Developer Documentation



Do You Really Understand JavaScript?

How to test your knowledge


Do self-test questions for this unit
understood
not understood
Enjoy the rest of this video as a recap!
Use documentation link in question to learn the concept!
You really understand JavaScript!

```
while (aQuestions) {
  aQuestions.pop().answer();
}
```

Do You Really Understand JavaScript?

JavaScript & browsers

JavaScript code is interpreted at runtime

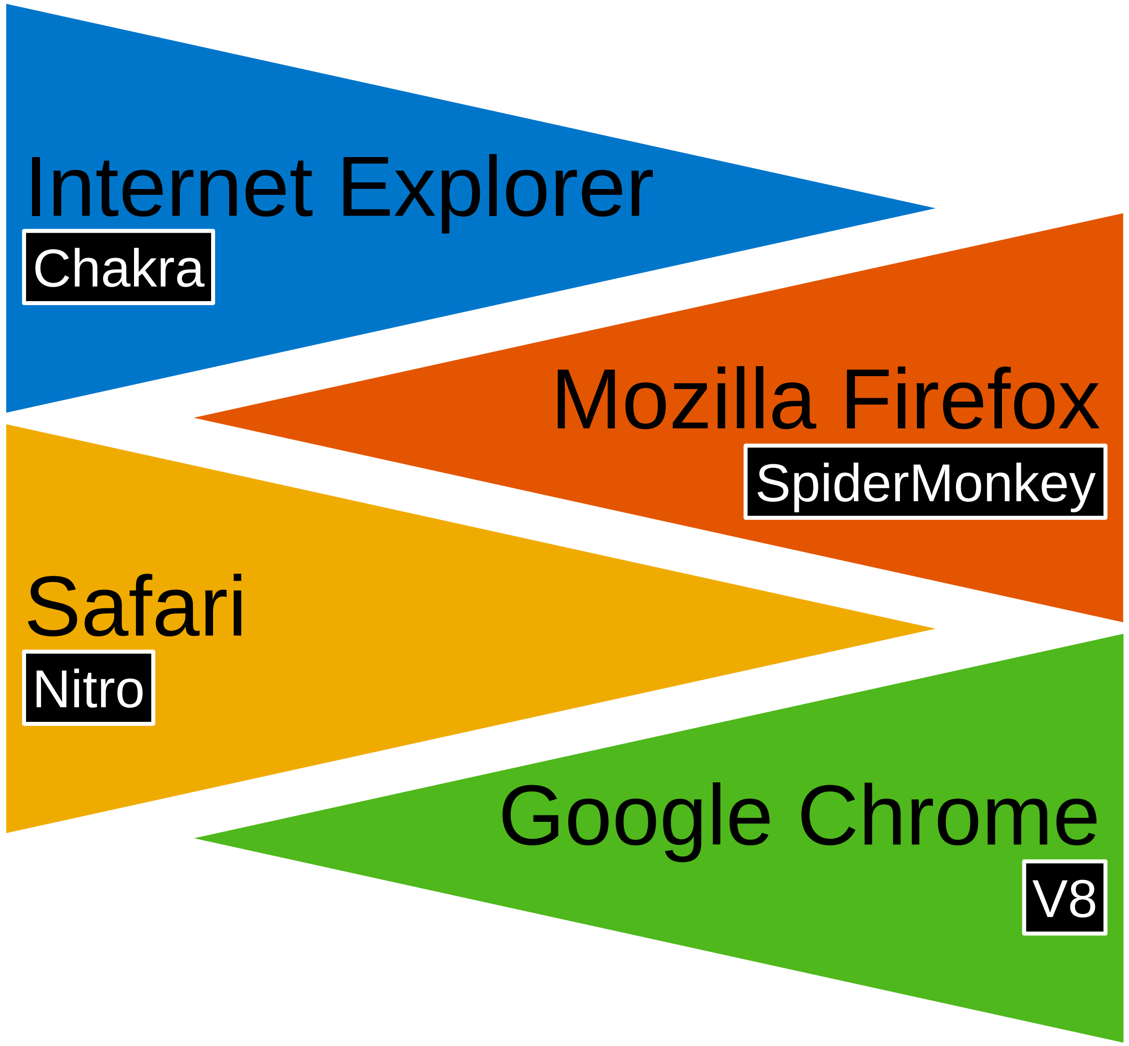
All modern desktop and mobile browsers are highly optimized for processing JavaScript code with their JavaScript engines.

JavaScript is properly known as ECMAScript

Browser engines implement ECMAScript, but with slight variations in the range of supported features and implementation details (e.g. the exact details of how XMLHttpRequest, XML API, and the DOM API have been implemented vary between browsers).

Client-side JavaScript is executed in a sandbox

For security reasons, you cannot access the local (file) system from a Web page.



Do You Really Understand JavaScript?

Client-side scripting

All presentation logic is done on the client

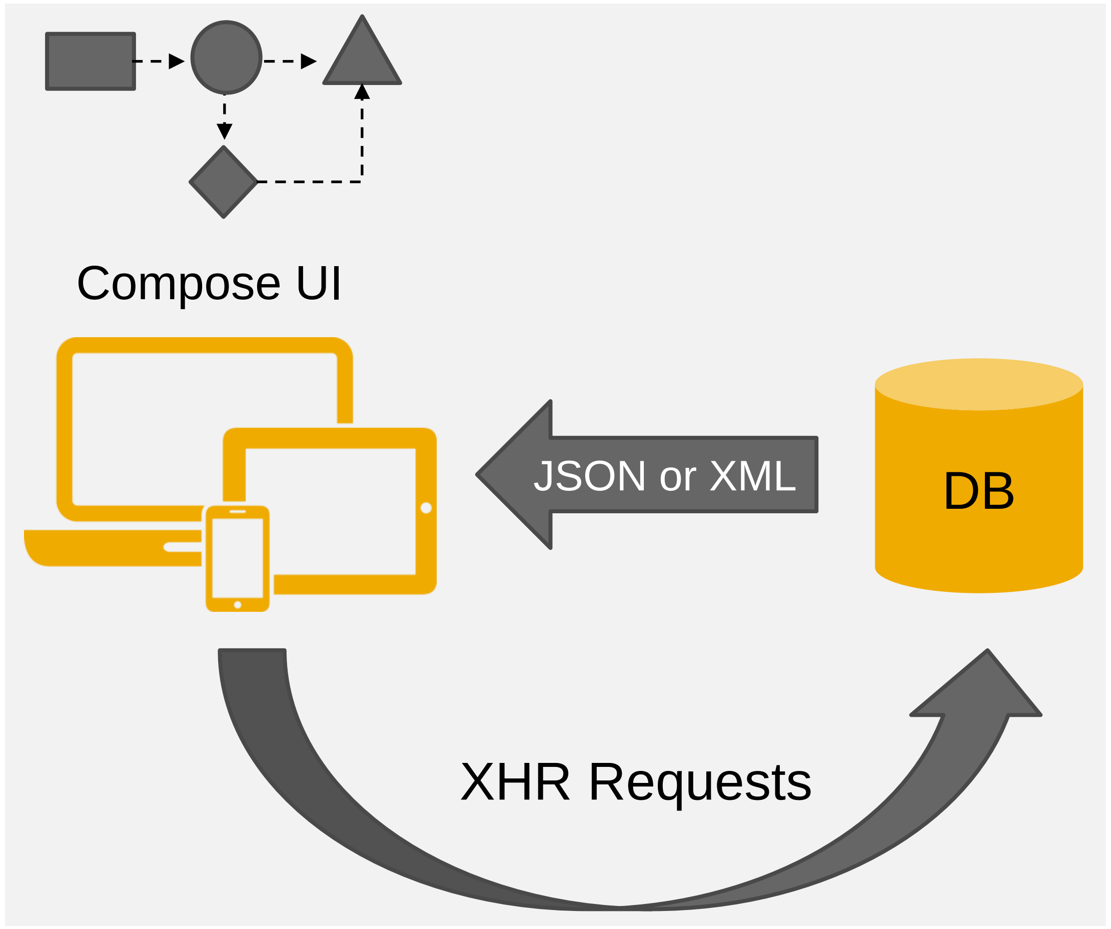
Only resources (HTML file, code, data) are loaded from the Web server. The code is then processed on the client to create DOM elements on the screen.

JavaScript is single-threaded

There is only one JavaScript thread per window. Other activities like rendering or downloading resources may be managed by separate threads but can be blocked by scripts.

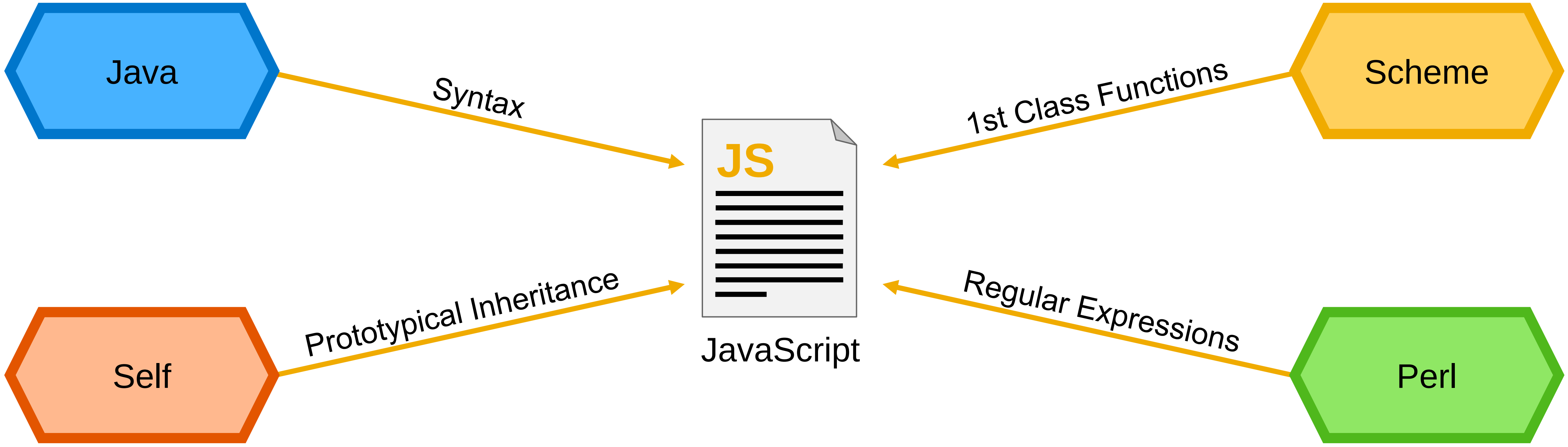
Data is loaded with XHR requests

During execution, data is often requested from a back-end system such as an OData or a REST-based service. The data is usually supplied in JSON or XML format.



Do You Really Understand JavaScript?

Linguistic inheritance of JavaScript



Do You Really Understand JavaScript?

Types and implicit conversions

There are only 6 data types in JavaScript:

- **Number**
  double-precision 64-bit format (IEEE 754)
- **String**
  Sequences of Unicode characters
- **Boolean**
  True or false
- **Object**
  Function / Array / Date / RegExp
- **Null**
  Deliberate non-value
- **Undefined**
  Indicates an uninitialized value

```
"5" * "2"
➢ 10

Typeof ("Hello" + 1)
➢ String
```

Implicit type conversions can be nice or dangerous; try to avoid them

```
1 == true
➢ True

1 === true
➢ False
```

Beware of truthy/falsy values, always use "===" for strict checks

```
var iAmount = 23;
var sString = "Hello";
```

Use Hungarian notation

Do You Really Understand JavaScript?

Objects

Objects are unordered collections of name-value pairs
Names are called “properties”, values can be of any type.
If it is a function, it becomes an object “method”.

Objects may have a constructor function
Attributes can then be stored and accessed with the “this” pointer in the current context

Everything except for core types is an object
Even functions are just objects “with an executable part”

Inheritance is not based on classes but prototypes
Properties and/or methods can be added to the object itself or to the prototype and deleted at runtime.

```
// object literal
var oObjLiteral = {};

// an Object object
var oObject = new Object();

// properties referenced using dot notation
oObject.property;
oObject.method([parameter]);
// properties referenced using array notation
oObject["property"];

// adding a property to an object
oObject.newProperty = "Property Value";
// deleting properties or methods
delete oObject.myMethod;
```

Do You Really Understand JavaScript?
Functions

Functions are objects with an executable part
They can be created and destroyed dynamically. Since a function is just an object, in addition to its executable code you can also assign your own properties (name, arguments, …) to it.

Functions can be passed as arguments to other functions
Functions are frequently passed as parameters to other functions in JavaScript. This is the basis upon which “asynchronous callbacks” work.

JavaScript variables exist within the scope of a function
All variables declared with a function are visible to all coding within that function. This is known as “function scope”. There is no block scope in JavaScript.

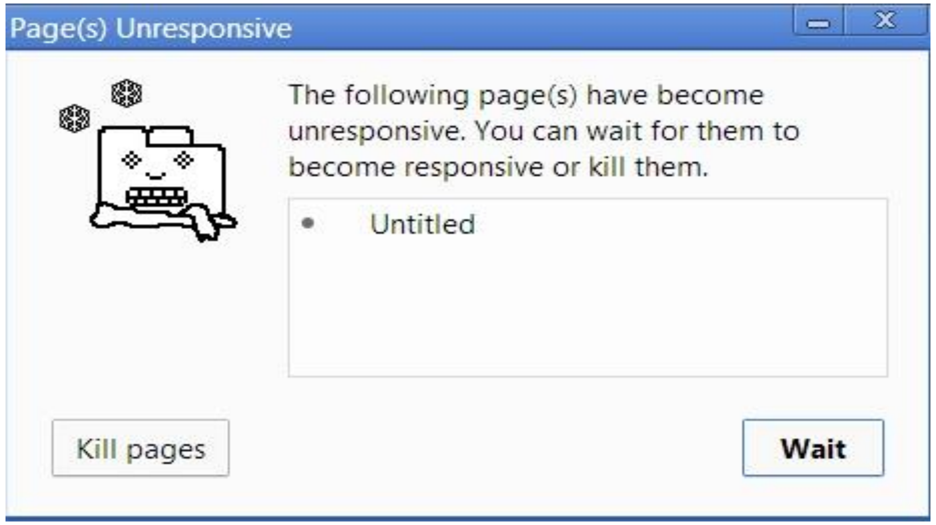
```
// function expression (anonymous)
var fnAdd = function(a, b) {
  return a + b;
};

// function declaration (named)
function add(a, b) {
  return a + b;
}

// function calls
fnAdd(2,3); // 5
add(2,3); // 5
```

Do You Really Understand JavaScript?

Asynchronous processing

Be careful! JavaScript is single-threaded
Long-running or resource-intensive tasks should be performed asynchronously, otherwise the UI might become unresponsive and your users might see a message like this.

- **Asynchronous module definition (AMD)**
  Helper tools for module loading: requireJS, sap.ui.define
- **Divide long-running tasks with setTimeout(…, 0)**
  It will continue with the execution immediately after all other tasks are processed
- **Use callback functions, event listeners, promises, and framework hooks**
  These patterns help you to efficiently structure and process application logic
- **Use asynchronous XHR calls**
  Avoid synchronous server requests because script execution will pause until the resource is loaded


Page(s) Unresponsive

The following page(s) have become unresponsive. You can wait for them to become responsive or kill them.

- Untitled

Do You Really Understand JavaScript?

Method chaining (cascading)

If a function returns a reference to the current context then a programming technique called method chaining can be used

Method chaining is widely used in many JavaScript frameworks including jQuery and SAPUI5

```
jQuery("#myButton")
  .text("Click me")
  .css("color", "#c00")
  .bind("click", function(e) {
    alert("Thanks for clicking");
  });
```

- ⊕ Chaining is a nice time-saver (the element in the example has to be looked up only once)
- ⊖ Debugging long method chains is more difficult

Do You Really Understand JavaScript?

Closures

Nested functions inherit the scope of their parent function

Closures adapt to variable changes, even if the changes happen a long time after the function was created. So you have to think of a closure as of a “live” thing.

“A closure is a special kind of object that combines two things: a function, and the environment in which that function was created. The environment consists of any local variables that were in-scope at the time that the closure was created.”

Source: MDN

```
function outer(param) {
    var attr1 = “One”;
    inner();

    // the nested function inherits all
    // the outer variables and parameters
    function inner() {
        var attr2 = “Two”;
        alert(attr1);          // “One”
        alert(attr2);          // “Two”
        alert(param);          // “Three”
    }
}
outer(“Three”);
```

Do You Really Understand JavaScript?

Scope: this or that?

this is a reference to the current execution context and depends on the scope

- Global scope: window
- Object scope: current object instance
- Function scope: depends on us!

When using callbacks, the context may be lost

this in an asynchronous callback function is by default the global window object!

Often, the “that” or “bind” construct is used to build a closure and ensure that the value of this is set correctly.

```
var myObj = {
whatsThis : function(that) {
    setTimeout(function () {
      // this is the global window object
      console.log(this);
    }, 0);
  }
};
var that = this;
setTimeout(function () {
          // access “that” closure
          // for working with the context
}, 0);

setTimeout(function () {
          // this is still the context
}.bind(this), 0);
```