An Introduction to JavaScript

IF3110 – Web-based Application Development
School of Electical Engineering and Informatics
Institut Teknologi Bandung

Introduction

- Interpreter based, used to be only running on the browser – run on server-side (node.js) or in a JVM (rhino)
- JavaScript has similar syntax to Java, but it is **not** based on it
- Dynamic typing and supports duck typing
- Objects as general containers
- Function is the first-class (Lambda/closure)
- Linkage via global variables
- Prototypes over class-inheritance (class keyword is syntactic sugar to prototypes)
- Formalized & standardised as **ECMAScript** (**ECMA-262** and ISO/IEC 16262)

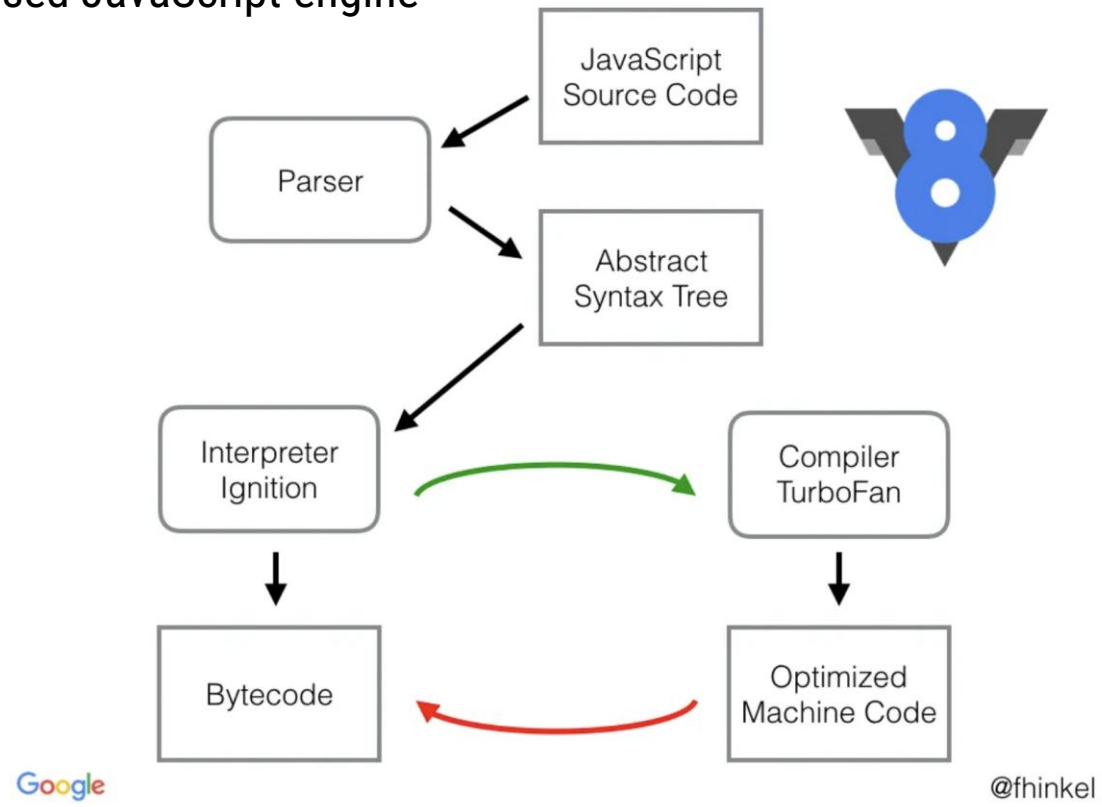
Introduction

- Javascript runs in browser:
  - Core syntax + DOM
- Has various implementation variant (the JavaScript Engine)
  - JavaScript Engine -> a special program embeded in a browser to execute JavaScript code. This program implements ECMAScript standard.
    - V8 -> Chrome, Edge, Opera, Node.js (not a browser)
    - SpiderMonkey -> Mozila Firefox
    - JavaScriptCore -> WebKit/Safari
  - support provided for DOM
    - The DOM is not, however, typically provided by the JavaScript engine but instead by a browser

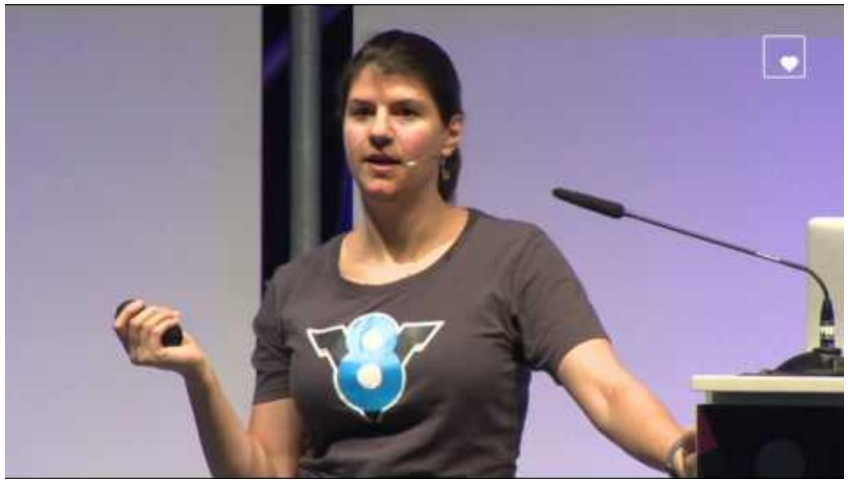
Inside the JavaScript Engine

V8 from Google is the most used JavaScript engine

written in C++



Courtesy: Franziska Hinkelmann's talk on V8



https://www.youtube.com/watch?v=p-iiEDtpy6I

V8 has had an interpreter as its first execution tier. This interpreter "compiles" JavaScript "just in time/JIT" to bytecode (which is then interpreted).

JavaScript in HTML

- Locating JavaScript
  - internal HTML document
    - Standar HTML 4.01

    <script type="text/javascript">
    …statement…
    </script>
    - Old Tag

    <script language="JavaScript">
    …statement…
    </script>
  - file external
    - Script JavaScript dituliskan pada file tersendiri (ekstensi file .js)
    - Pemanggilan file JavaScript:

    <script src="namafile.js"></script>

- Tips: place js script at the end of a html, to improve the performance

Simple Example

```
<HTML>
<HEAD>
<TITLE>Hello javascript</TITLE>
</HEAD>
<BODY onload="createDoc();">
<H1>Hello javascript...</H1>
<HR>
<SCRIPT type="text/javascript">
  function createDoc() {
    document.write(“Hello…”)
  }
</SCRIPT>
</BODY>
</HTML>
```

Basic Syntax

- Alike to Java or C
- Case-sensitive
- One line represents a statement
- “;” is optional to end a statment
- Comment: // and /* */

Basic Syntax: Type

- Data Type
  - Number
  - String
  - Boolean
  - Object: e.g.:
    - Function
    - Array
    - Date
    - Regexp
  - Null
  - undefined

Number

All numbers are represented in floating-point 64-bit IEEE-754 (double), but can be operated in integer

```
0, 3,  10000000, 0xF3, 031

3.14,  1.23e10, 3.14E-14
```

Be careful with **rounding errors**

Special values

```
NaN, Infinity

Number.MAX_VALUE, Number.MIN_VALUE

Number.POSITIVE_INFINITY

Number.NEGATIVE_INFINITY
```

Number

NaN: Not a Number, result from a wrong operation

Every operation that has NaN, will result in NaN

NaN != NaN

```
>>> var a = 0/0;
>>> a
NaN
>>> typeof a
"number"
```

Number function

```
Number(value)
```

Cast some value into number

Result in NaN if there is some problem,

Alternative: + prefix & parseInt

```
+value
```

```
parseInt(value, radix)
```

Math

Math object provides a collection of standard function in arithmetic

`abs`		absolute value
`floor`	integer	
`log`		logarithm
`max`		maximum
`pow`		raise to a power
`random`	random number	
`round`	nearest integer	
`sin`		sine
`sqrt`	square root	

String

- Series of character written with some delimiter “ or ‘

  ‘this is string’, “this also a string”, “ini bisa ‘kan”, “nama=‘amir’”, ‘This string\n consists of 2 lines’

- Immutable
- Some basic operations:

  s = “hello”

  s.length, s.charAt(2), sub = s.substring(2, 3)

  i = s.indexOf(‘e’)

  "hello, world".replace("hello", "goodbye")

  "hello".toUpperCase()

String Methods

```
charAt
concat
indexOf
lastIndexOf
match
replace
search
slice
split
substring
toLowerCase
toUpperCase
```

boolean

- Value: true atau false
- Boolean(value): cast some value into boolean. Alternative: use !!
- Falsy: 0, **`false, null, undefined,""`** (empty string), **`NaN`**
- Truthy: other than falsy (including “null”, “0”, “false”)

`null`

A value that isn't anything

`undefined`

A value that isn't even that

The default value for variables and parameters

The value of missing members in objects

Loosely and Dynamically Typed

Loosely typed

Any of these types can be stored in a variable, or passed as a parameter to any function.

The language is not "untyped”.

Conversion of types happen automatically.

Dynamically typed

Types are not required in advance.

Types are not checked at compile-time.

Type Conversion

10 + " objects" // => "10 objects". Number 10 converts to a string

"7" * "4" // => 28: both strings convert to numbers

var n = 1 - "x"; // => NaN: string "x" can't convert to a number

n + " objects" // => "NaN objects": NaN converts to string "NaN"

Variable

- Case sensitive
- Declare by var, const, or let (ES6, 2015)

  var i; var x = 2;

  let y = 3; // having a block scope, and cannot be redeclared

  const z = 4; // having a block scope, and cannot be redeclared & reassigned
- Variable with out a declaration, will be declared automatically as a global variable

  a = 5;
- Scope: global dan local. local used when variable declared with in a function with var
- Convention: variable name, parameter, member, function starts with lowercase. Constructor starts with uppercase

Operator Aritmatika

Operator	Usage	Example	Result
+	addition	3+4	7
-	subtraction	4-3	1
*	multiplication	4*3	12
/	division	4/3	1.33333333
%	modulus	4 % 3	1
++	increment	x=5 x++	x=6
--	decrement	x=5 x--	x=4

Assignment Operator

operator	example	explanation
=	a = b	assignment b to a
+=	a += b	a = a+b
-=	a -= b	a = a-b
*=	a *= b	a = a*b
/=	a /= b	a = a/b
%=	a %= b	a = a%b

Comparative Operator

operator	example	explanation
==	a == b	a equal to b
!=	a != b	a not equal to b
<	a < b	a lesser than b
>	a > b	a bigger than b
<=	a <= b	a lesser or equal to b
>=	a >= b	a bigger or equal to b
===	‘10’ === 10	comparation w/o automatic type conversion

Logic Operator

operator	usage	example
&&	and	x = 6 y = 3 (x < 7 && y < 4)
||	or	x = 6 y = 3 (x < 7 || y < 2)
!	not	x = 6 y = 3 x != y

== !=

Equal and not equal operators

These operators can do type correction

It is better to use === (strict equality operator) and !==, which do not do type correction.

Examples:

"5" == 5 // true, because of type coercion

"5" === 5 // false, no type coercion occurs

&&

- The guard operator, aka *logical and*
- If first operand is truthy
  - then result is second operand
  - else result is first operand
- It can be used to avoid null references

```
if (a) {
   return a.member;
} else {
   return a;
}
```

- can be written as

```
return a && a.member;
```

||

The default operator, aka *logical or*

If first operand is truthy

then result is first operand

else result is second operand

It can be used to fill in default values.

```
var last = input || nr_items;
```

(If **`input`** is truthy, then **`last`** is input, otherwise set **`last`** to **`nr_items`**.)

!

Prefix *logical not* operator.

If the operand is truthy, the result is **`false`**. Otherwise, the result is **`true`**.

`!!` produces booleans.

Bitwise

`&` `|` `^` `>>` `>>>` `<<`

The bitwise operators convert the operand to a 32-bit signed integer, and turn the result back into 64-bit floating point.

string operator

“ini “+”buku”

“ini buku”

“jumlah adalah “+ 5

“jumlah adalah 5”

“jumlah adalah “ + 5 + 3

“jumlah adalah 53”

5 + 3 + “ adalah bilangan”

“8 adalah bilangan”

Break statement

Statements can have labels.

Break statements can refer to those labels.

```
loop: for (;;) {
        ...
            if (...) {
                break loop;
            }
        ...
}
```

For statement

Iterate through all of the elements of an array:

```
for (var i = 0; i < array.length; i += 1) {

                    // within the loop,

                    // i is the index of the current member

                    // array[i] is the current element


}
```

For statement

Iterate through all of the members of an object:

```
for (var name in object) {
    if (object.hasOwnProperty(name)) {


        // within the loop,
        // name is the key of current member
        // object[name] is the current value


    }
}
```

Switch statement

Multiway branch

The switch value does not need to a number. It can be a string.

The case values can be expressions.

Switch statement

```
switch (expression) {

case ';':

case ',':

case '.':

    punctuation();

    break;

default:

    noneOfTheAbove();

}
```

Throw statement

```
throw new Error(reason);


throw {
    name: exceptionName,
    message: reason
};
```

Try statement

```
try {
    ...
} catch (e) {
    switch (e.name) {
    case 'Error':
        ...
        break;
    default:
        throw e;
    }
}
```

Try statement

The JavaScript implementation can produce these exception names:

```
'Error'

'EvalError'

'RangeError'

'SyntaxError'

'TypeError'

'URIError'
```

With statement

- Intended as a short-hand
- Ambiguous
- Error-prone
- Don't use it

```
with (o) {
    foo = null;
}
```

- `o.foo = null;`
- `foo = null;`

Function

- mechanism to structure a program
  - modular
  - reusable
- kind
  - built-in
  - user defined
- function can be
  - have parameters
  - return a value

Function

syntax

```
function namaFungsi ([parameter]){
    … statements
    [return value]
}
```

function can be defined within another function

Many ways to define a function

- typical definition

```
function f(x, y) { return x*y; }
```

- constructor Function()

```
var f = new Function("x", "y", "return x*y;");
```

- function literal

```
var f = function(x, y) { return x*y; }
```

- arrow function

```
var f = (x, y) => return x*y;
```

function as data

function can be treated as data, stored in variable

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

var b = square;

var c = b( 5 );
```

function: parameter

function can have parameter/argument

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

argument can be access from the function w/ name or object arguments

```
function square(x) { return arguments[0]*x; }
```

argument checking/verification in Javascript done at runtime

Return statement

`return` *expression* **`;`**

or

`return;`

If there is no *expression*, then the return value is **`undefined`**.

Except for constructors, whose default return value is **`this`**.

Object

Object – entity that composed of states represented as attributes (properties) value, and behavior represented in term of methods

JavaScript Object

- In JavaScript, almost everything is object
  - Unless the 6 primitive types (string, number, boolean, null, undefined, symbol in ES6, bigint in ES11)
- Object in Javascript is a collection
  - similar to hashtable – can be accessed by a key (member name)
- Member is accessable by dot or subscript (a.b or a[‘b’])
- **`new Object()`** produces an empty container of name/value pairs
- A name can be any string, a value can be any value except **`undefined`**
- A function is also an object

Object

- Java Script is prototype-based object language
- Object created by constructor

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

- Object can be created from literal

```
var circle = { x:0, y:0,
radius:2 }
```

- `class` keyword is introduced in ES6 (2015)

- Attributed & methods access by . (dot)

```
var book = new Object();

book.title = "Javascript: The
Definitive Guide"

book.author = "David
Flanagan"
```

Object

- property (attribute) can be enumerated w/ for loop

```
for( var name in obj ) {

  document.write( name + "<BR>");

}
```

- property of an object can be removed

```
delete book.title;
```

property access

- using dot

```
object.property
```

- as associative array

```
object["property"]
```

Constructor

- special function to create/initiate an object
- called with “new” command

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

prototype

- prototype: mechanism to share properties and methods of objects Javascript
- every object has prototype
- properties & methods, to be share with other objects, place in prototype.

```
Circle.PI = 3.14;

Circle_area() { return Circle.PI * this.r * this.r; }

Circle.prototype.area = Circle_area;


c = new Circle();

document.write( c.area() );
```

prototype

- properties & method, in object instance, allocated ONLY for that particular instance

  'this' refers to the instance

- properties & method, in constructor, allocated ONLY for that constructor

  can be used from other object

  'this' refers to constructor

- properties & method, in prototype, can be used by other object

  'this' refers to the instance

Circle class prototype

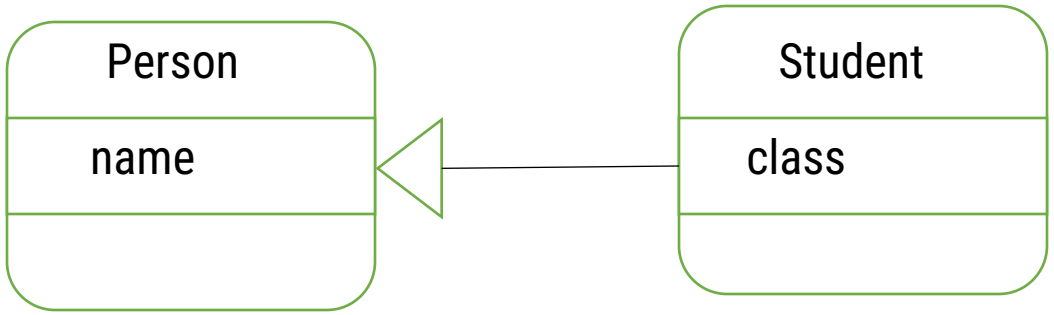
```
function Circle(radius) {
  this.r = radius;
  this.min = function() {   }
}

Circle.PI = 3.14159;

Circle.prototype.area = function() {
   return Circle.PI * this.r * this.r;
}

Circle.max = function(a,b) {
    if (a.r > b.r) return a;
    else return b;
}
```

```
var c = new Circle(1.0);
var d = Circle(2.0}
Circle.prototype.name  = "a";
c.r = 2.2;
var a = c.area();
var x = Math.exp(Circle.PI);
var d = new Circle(1.2);
var bigger = Circle.max(c,d);
```

Inheritance

Inheritance: OO concept for reusability

using an existing class to define a new class



Inheritance through prototype

JavaScript provides inheritance through prototype:

```
function Person(n) { this.name = n; }
function Student(n, c) {
   this.name = n;
   this.class = c;
}
Student.prototype = new Person();
```

Class and Inheritance

Since ES6 (2015) :

```
class Person {
  constructor(n) {
    this.name = n;
  }
}
class Student extends Person {
  constructor(n, c) {
    super(n);
    this.class = c;
  }
}
```

Object

constructor

o.constructor consists of function constructor used to initialize o

toString()

represents string of an object. Called automatic when an object is casted to string

valueOf()

represents object other than string. Called automatic when an object is casted to other than string

isPrototypeOf(x)

verify whether an object is prototype from another object

Array

- by Array constructor

```
var a = new Array();
var a = new Array( 1, 2, "tiga" );
```

- by literal

```
var a = [ 1, 2, "tiga" ];
```

- array access

```
a[0] = 1;
a[7] = 8;
```

- length of array

```
a.length
```

Array methods

join()

convert array values into string

reverse()

revers the value of array

sort()

sort the array values alphabetically

concat()

combing array values with the value of parameter

slice()

slice the value of array array

splice()

remove and add the value of array

push(), pop(), shift(), unshift()

Array Example

```
<html>
<script type="text/javascript">
//cara 1 pendefinisian array
mhs = new Array();
mhs[0] = "Bevin";
mhs[1] = "Andini";
mhs[2] = "Citra";
//cara 2 pendefinisian array
mhs = new Array("Bevin", "Andini", "Citra");
//cara 3 pendefinisian array
mhs = ["Bevin", "Andini", "Citra"];
//cara pengaksesan array
document.write("Mahasiswa pertama adalah "+mhs[0]+".<br>"); //Bevin
document.write("Mahasiswa terakhir adalah "+mhs[2]+".<br>"); //Citra
//cara pengaksesan array menggunakan loop
for (i=0; i<mhs.length; i++) {
  document.write(mhs[i] +"<br>");
}
//mengurutkan array
mhs.sort();
//menggabungkan array
document.write(mhs.join("-")); //Andini-Bevin-Citra
</script>
</html>
```

Regular expression

- used to process text, search and convert text with particular pattern
- Literal:

```
var pattern = /s$/
```

- RegExp object

```
var pattern = new RegExp("s$");
```

Browser object

window

navigator

screen

history

location

Not standard, but provided in every browser

Browser object

Window object: global execution context,

```
var x = 5;
```

sama dengan

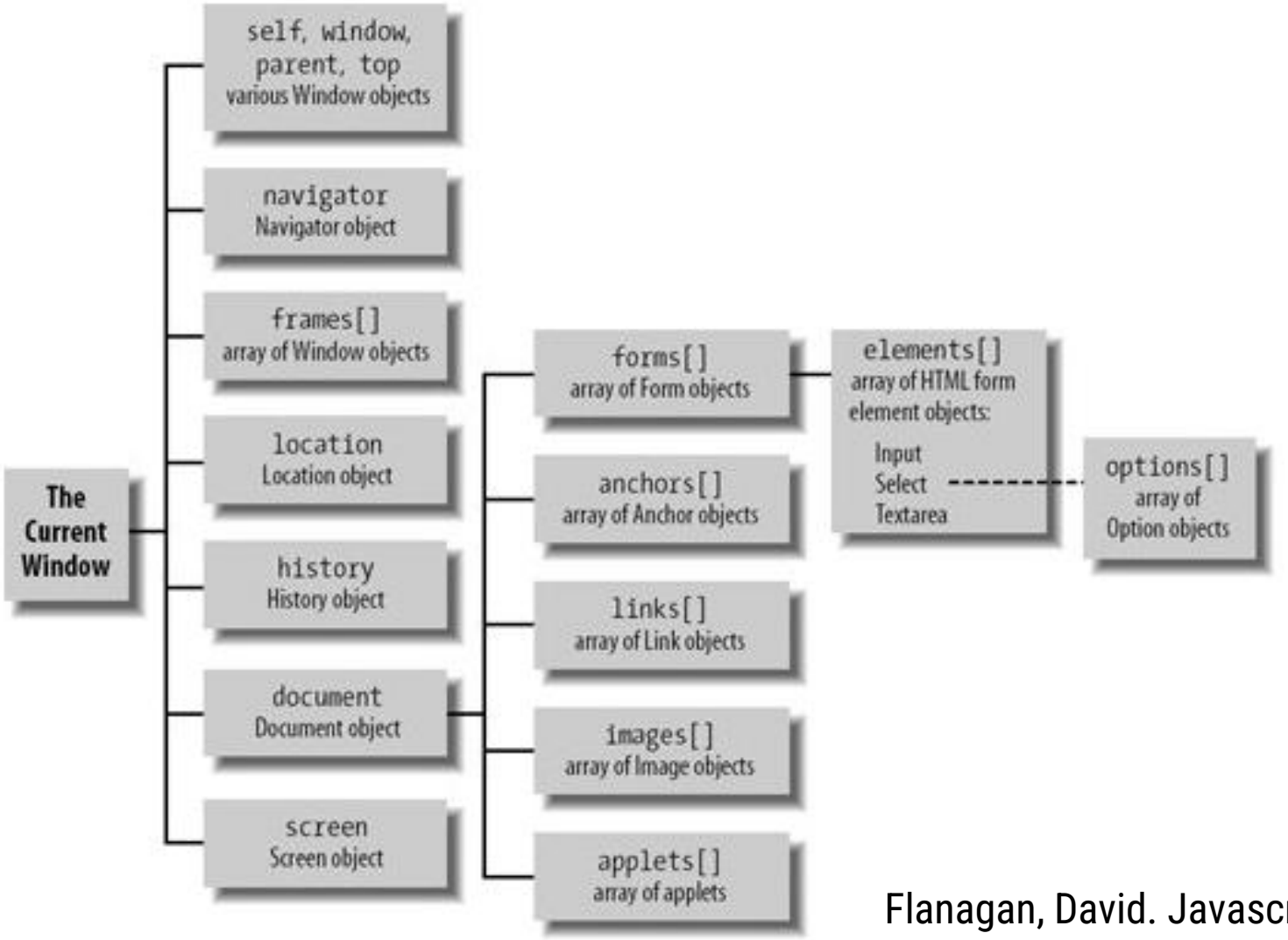
```
window.x = 5;
```

It refers to window browser (or frame) that renders the page

Document object: page that is being rendered

```
window.document
```

DOM



Flanagan, David. Javascript: The Definitive Guide, 5th Ed.

Document object

getElementById(id)

getElementsByTagName(tag)

createElement(tag)

HTMLDocument, derived from Document

- body, forms, anchors, images, links
- title, lastModified, referrers
- etc
- close(), open(), write()

Javascript as event handler

HTML Element HTML can define to run script if an event occurs

onclick, onmousedown, onmouseup, onchange, onload

In-line HTML:

```
<input type="checkbox" name="options” value="giftwrap"
onclick="giftwrap = this.checked;" >
```

Javascript as event handler

Script:

```
<script>
// Define a function to display function displayTime() {
// A script of js code the current time
var elt = document.getElementById("clock");
// Find element with id="clock"
}
window.onload = displayTime;
// Start displaying the time when document loads.
</script>
```

Javascript as URL and Bookmarklet

Javascript can be executed as a URL link

```
javascript:alert("Hello World!")
javascript:var now = new Date( ); "<h1>The time is:</h1>" + now;
```

URL containing javascript can be saved as a bookmark

```
<a href='javascript:var e="", r="";
do{e=prompt("Expression: "+e+"\n"+r+"\n",e);
try{r="Result: "+eval(e); }catch(ex){r=ex; }}
while(e);void 0;'> JS Evaluator</a>
```

JavaScript URL often used by hackers to run malicious codes, e.g., Cross Site Scripting (XSS)

Javascript Execution in HTML

Javascript is executed as soon as the element, containing script, has been processed by the browser.

To postponed the execution: use `defer` attributed,

OR

write a script as a function that is called by onload attributes from the body element

Javascript threading model

Javascript runs sequentially; with a single thread model.

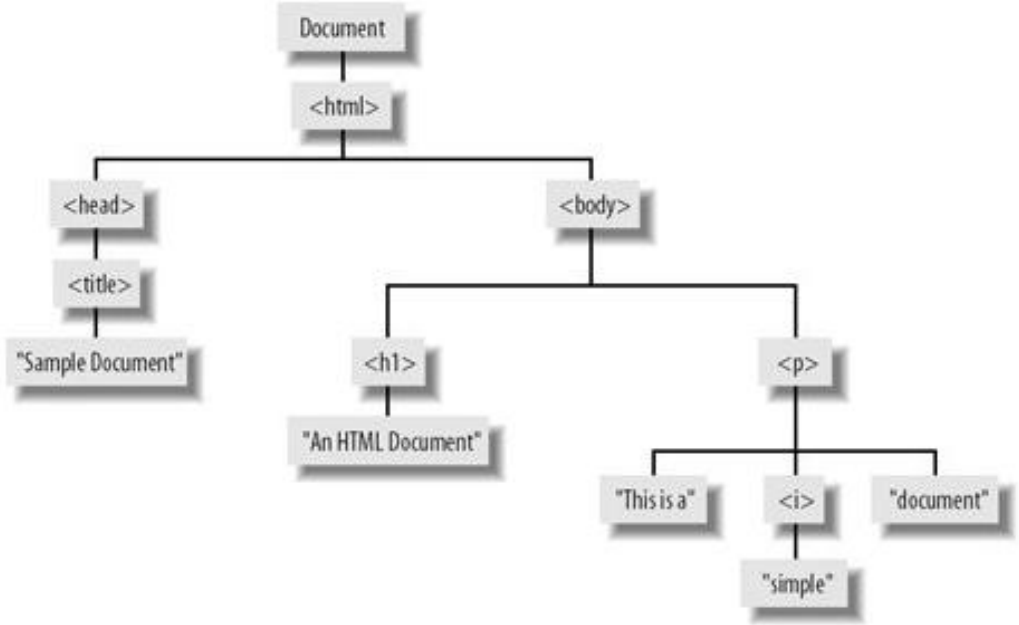
On loading HTML document HTML, executing a script will suspend the loading process until the execution finishes

event-handler javascript SHOULD NOT be long, it creates a browser freeze.

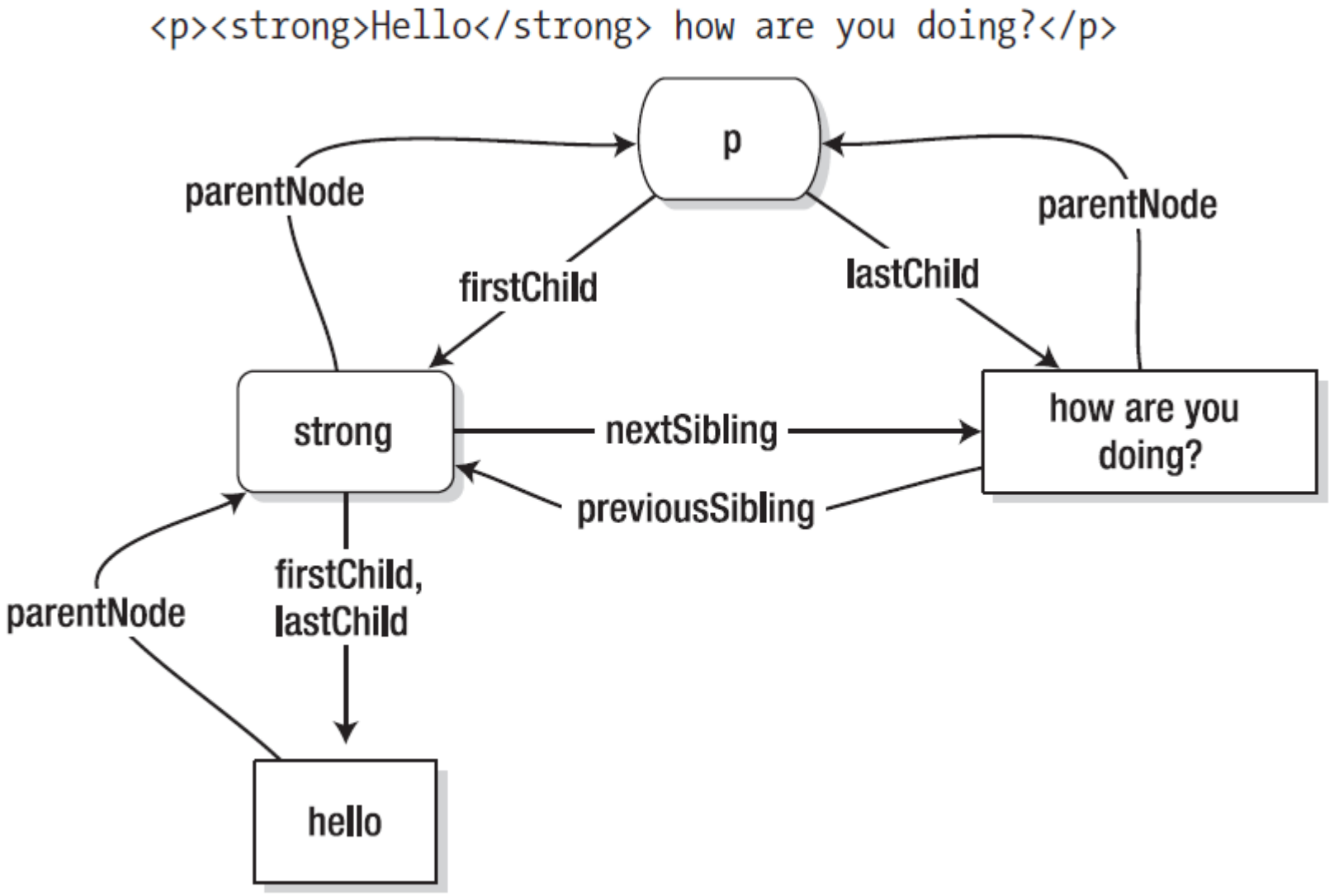
W3C DOM

HTML Document is accessed by javascript via DOM

```
<html> <head> <title>Sample Document</title> </head> <body> <h1>An
HTML Document</h1> <p>This is a <i>simple</i> document.</p> </html>
```

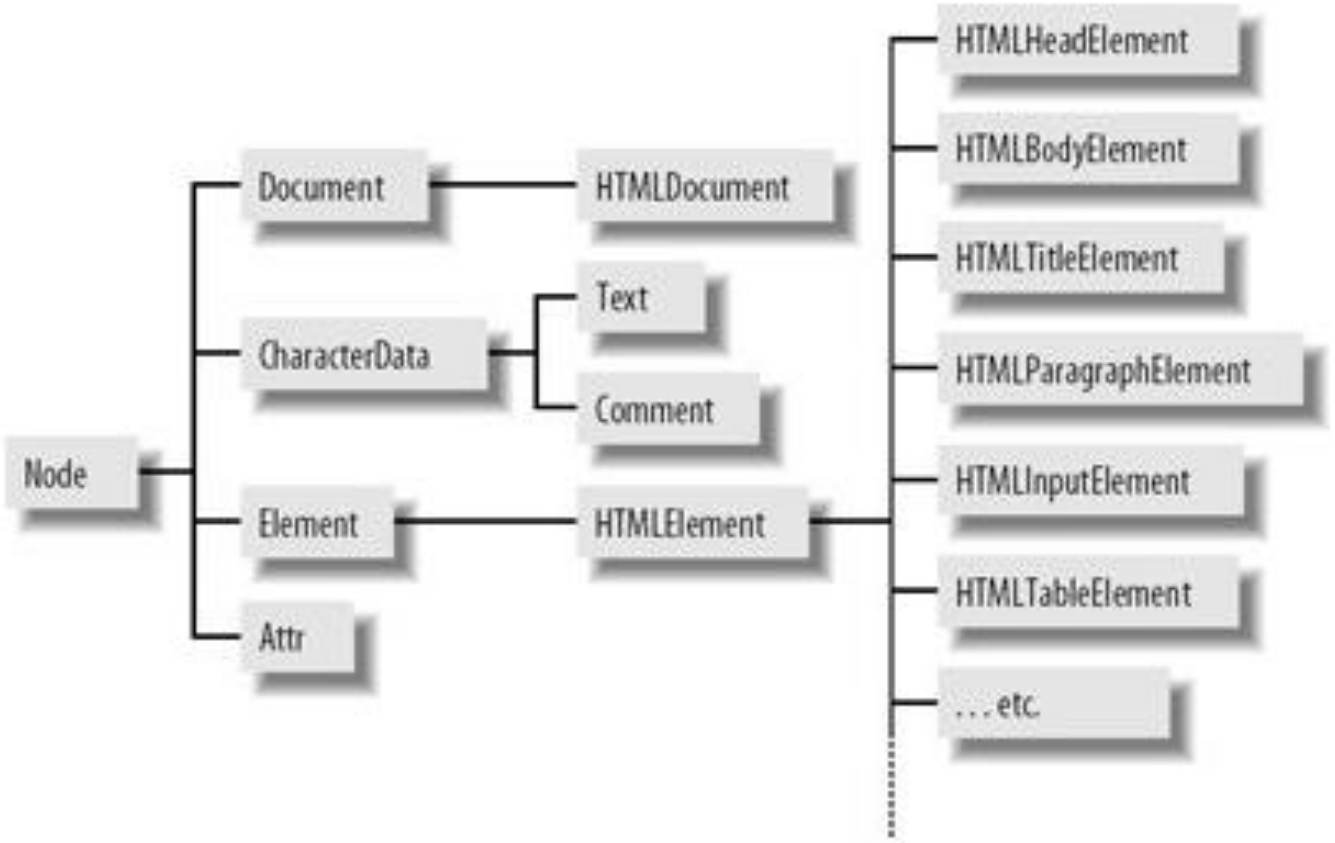


W3C DOM


Hello how are you doing?
p
parentNode
parentNode
firstChild
lastChild
strong
nextSibling
how are you doing?
previousSibling
parentNode
firstChild, lastChild
hello

W3C DOM

HTML element in DOM is descendant of Node



W3C DOM

DOM Level: standardisation DOM by W3C in 2 version (level)

- DOM Level 1, 1998
  - Definition of Node, Element, Attr, Document etc
- DOM Level 2, 2000
  - Modular. DOM Level 1 becomes Core module, there are optional moduls, such as Event, CSS
- DOM Level 3

W3C DOM

Accessing element:

document.getElementById*(id_name);*

return object Node object, if existed

Document.getElementByTagName(t*ag_name*);

return NodeList, can be accessed as array

```
document.getElementsByTagName("h1")[0]
var li = document.getElementsByTagName("li");
for ( var j = 0; j < li.length; j++ ) {
  li[j].style.border = "1px solid #000";
}
```

Encode/Decode URIs

- In a URL (Uniform Resource Locator) or a URI (Uniform Resource Identifier), some characters have special meanings.
- If you want to "escape" those characters, you can use the functions encodeURI() or encodeURIComponent().
  - The first one will return a usable URL,
  - The second one assumes you're only passing a part of the URL, like a query string for example, and will encode all applicable characters.

```
>>> var url = 'http://www.packtpub.com/script.php?q=this and that';
>>> encodeURI(url);
"http://www.packtpub.com/scr%20ipt.php?q=this%20and%20that"
>>> encodeURIComponent(url);
"http%3A%2F%2Fwww.packtpub.com%2Fscr%20ipt.php%3Fq%3Dthis%
20and%20that"
```

- The opposites of encodeURI() and encodeURIComponent() are decodeURI() and decodeURIComponent() respectively.
- Sometimes, in older code, you might see the similar functions escape() and unescape() but these functions have been deprecated and should not be used.

Timers

Timer berguna untuk membuat animasi atau efek tampilan

```
<script>
  var WastedTime = {
    start: new Date( ),
    displayElapsedTime: function( ) {
      var now = new Date( );
      var elapsed = Math.round((now - WastedTime.start)/60000);
      window.defaultStatus = "You have wasted " + elapsed + "
         minutes."; }
  }
  setInterval(WastedTime.displayElapsedTime, 60000);
</script>
```

Forms

- Form can be accessed via javascript using document.forms[]
- Form object has submit() and reset() method
- Javascript can be called from a form via event:
  - onsubmit, onreset
  - onchange, onblur, onfocus

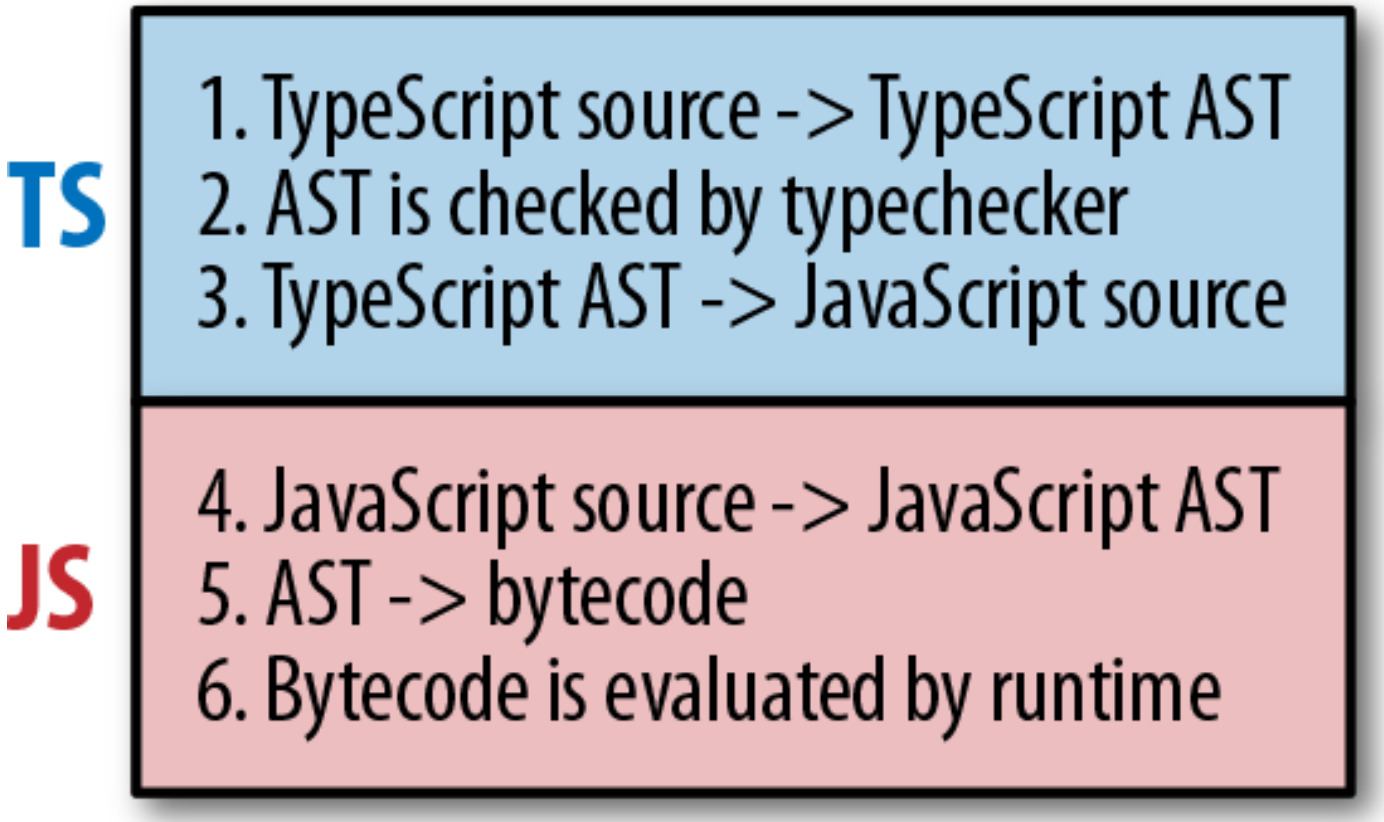
Form Element

Object	HTML tag	type property	Description and events
Button	`<input type="button">` or `<button type="button">`	"button"	A push button; `onclick`.
Checkbox	`<input type="checkbox">`	"checkbox"	A toggle button without radio-button behavior; `onclick`.
File	`<input type="file">`	"file"	An input field for entering the name of a file to upload to the web server; `onchange`.
Hidden	`<input type="hidden">`	"hidden"	Data submitted with the form but not visible to the user; no event handlers.
Option	`<option>`	none	A single item within a Select object; event handlers are on the Select object, not on individual Option objects.
Password	`<input type="password">`	"password"	An input field for password entrytyped characters are not visible; `onchange`.
Radio	`<input type="radio">`	"radio"	A toggle button with radio-button behavioronly one selected at a time; `onclick`.
Reset	`<input type="reset">` or `<button type="reset">`	"reset"	A push button that resets a form; `onclick`.
Select	`<select>`	"select-one"	A list or drop-down menu from which one item may be selected; `onchange`. (See also Option object.)
Select	`<select multiple>`	"select-multiple"	A list from which multiple items may be selected; `onchange`. (See also Option object.)
Submit	`<input type="submit">` or `<button type="submit">`	"submit"	A push button that submits a form; `onclick`.
Text	`<input type="text">`	"text"	A single-line text entry field; `onchange`.
Textarea	`<textarea>`	"textarea"	A multiline text entry field; `onchange`.

TypeScript

- What is TypeScript?:
  - Programming Language: A language that includes all the existing JavaScript syntax, plus new TypeScript-specific syntax for defining and using types
  - Type Checker: A program that takes in a set of files written in JavaScript and/or TypeScript, checks for incorrect syntaxes
  - Compiler (Transpiler) : A program that runs the type checker, reports any issues, then outputs the equivalent JavaScript code.
  - Language Service: A program that uses the type checker to tell editors such as VS Code how to provide helpful utilities to developers

TypeScript


TS
1. TypeScript source -> TypeScript AST
2. AST is checked by typechecker
3. TypeScript AST -> JavaScript source
JS
4. JavaScript source -> JavaScript AST
5. AST -> bytecode
6. Bytecode is evaluated by runtime

TypeScript

- JS
  - Unconstrained
  - Checked at runtime
  - Errors surfaced at runtime (mostly)

```
function squareOf(n) {
  return n * n
}
squareOf(2)     // evaluates to 4
squareOf('z')   // evaluates to NaN
```

- TS
  - Constrained or bound statically
  - Checked at compile time
  - Errors surfaced at compile time (mostly)

```
function squareOf(n: number) {
  return n * n
}
squareOf(2)     // evaluates to 4
squareOf('z')   // Error TS2345: Argument of type '"z"' is not
                // parameter of type 'number'.
```