April 4, 2018

Javascript language fundamentals

Variables and constants//pic

For() {

Var localVar2;

Let blockVar;

//in java, we know that the variable will only be available for that loop and not for the code outside that block

Primitive data types = numbers, strings, Booleans,null and undefined

Number //none pic

//in Boolean, the only false string is an empty string

//there are no imaginary numbers in js

//implicit type conversions- it shows great flexibility. Unlike in java that has a compiler which tells you of mistakes, this doesn’t work for javascript because of this flexibility

//in java void it doesn’t return anything, in js functions, there will always be a return

April 6, 2018

* JavaScript Event Handling
* Event Propagation Order(DOM-compliant browsers)

- capturing phase=//from outside going inwards

- at target

- bubbling phase=inside going outwards

=//starts at the window object, then trigger at the document object, then the document itselt, then the root of the html, followed by the body

\* Event Handler Registration

<button id=’button’>Click Me</button>

Function handler(){

}

* Method #1 (inline HTML event attribute)

Non-standard, universal support, not recommended

<button id=’button’ onclick=’handler()’>ClickMe</button>

* Method#2 (pre-DOM L2)

Non-standard, universal support

var button = document.getElementById(‘button’);

Button.onclick =handler;

//can register only one handler

* Method #3 (DOM L2 event registration)

Standard, well-supported

Var button = document.querySelector(‘#button’);

button.addEvenListener(‘click’, handler); //bubbling phase

button.addEvenListener(‘click’, handler,true); //capturing phase

//can register more than one event handler

//can register handlers on the bubble or capture phase

//use removeEventListener() to remove event handler(same argument)

//use dispatchEvent() to trigger event programmatically

Button.addEventListener(‘click’, function() {});

Button.removeEventListener(‘click’, function() {});

From the 2, the function wont be removed because every function is unique, there can no be the same instance of it.

* Method#4(Microsoft-specific, non-standard, IEv6-v10)

var button = document.getElementById(‘button’);

button.attachEvent(‘onclick’, handler);

//can register more than one event handler

//can register handlers only on the bubble phase(old MS browsers)

//use detachEvent() to remove event handler

//use fireEvent() to trigger event programmatically

//cross-browser event registration

Var button = document.getElementById(‘button’);

If(button.addEventListener){

Button.addEventListener(‘click’, handler);

}else if(button.attachEvent){

Button.attachEvent(‘onclick’, handler);

}else{

Button.onclick = handler;

}

//determining the identity of the element handling the event

//for method 1 registration

Function handler(obj){

//’obj’ is the element that is handling the event

}

//for methods #2 and #3 registration

Function handler(){

Var element =this;

}

\*determining associated information about the event

//whenever an event is triggered, information about the event is made available

//via the ‘event’ object property of the ‘window’ object(i.e., window.event)

//for method #1 registration, the event object may be passed explicitly in the call to the handler

//for method#2 and #3 registration, the event object

* Event Object Properties and Methods:

- bubbles

- eventPhase //simply tells you in what phase you are in. There are 3 values AT\_TARGET, BUBBLING\_PHASE, CAPTURING\_PHASE

- target (or srcElement for non-standard)

- currentTarget //which particular element targeted the event, and which particular element targeted the handling

- preventDefault() //allows to prevent the default behaviour when the event occurs e.g clicking a link, submitting information

- stopPropagation() //stops propagation, stop handling etc…that becomes the last element

- stopImmediatePropagation() //that becomes the last handler

- other event-specific information //control-c

* Creating and Dispatching Events

var clickEvent = new Event(‘click’);

var button = document.querySelector(‘#button’);

button.dispatchEvent(clickEvent);

April 7, 2018

* AJAX JavaScript

XMTLHTTPRequest, open, send

* JS is inherently single-threaded = means it can only do 1 thing at a time

Synchronous fetch

Asynchronous fetch

* //Same-origin request- can only fetch request from the same or original javascript
* callback =a function specified such that when an event occurs, that function will be invoked automatically
* JS Promise =

-var promise = new Promise(function(resolve, reject)){

//do operation(asynchronously)

//success – calls resolve(sum) function

//failure – calls reject(err) function

}

fetch().then(calculate).then(print);

promise().then(success).catch( failure);

function success(result){}

function failure(err){}

They are inherently asynchronous. You don’t have to wait since the promise function automatically progresses.