Welcome to the Universe of HyperText

Home

Access to this information is provided as part of the WorldWideWeb project. The WWW project does not take responsibility for the accuracy of information provided by others.

How to proceed

References to other information are represented like this. Double-click on it to jump to related information.

General CERN Information sources

Now choose an area in which you would like to start browsing. The system currently has access to three sources of information. With the indexes, you should use the keyword search option on your browser.

CERN Information

A general keyword index of information made available by the computer centre, including CERN, Cray and IBM help files, "Writeups", and the Computer Newsletter (CNL). (This is the same data on CERNVM which is also available on CERNVM with the VM FIND command).

Yellow Pages

A keyword index to the CERN telephone book by function.

You can access the internet news scheme (See information for new users). News articles are distributed typically CERN-wide or worldwide, and have a finite lifetime.

They may be of general interest at CERN include

A European Technology Interest Group) news.

You can find more news on this machine, see also the following topics:

1990,91, CERN. Distribution restricted: ask for terms. TEST VERSION ONLY

Text: Text which is not constrained to be linear.

Media: Information which is not constrained linear... or to be text.

This is the first version of the NextStep WorldWideWeb application with the libWWW library. Bug reports to www-bug@info.cern.ch. Check the list of known bugs in the web too.

This was the original prototype for the World-Wide Web. Many others for other platforms now exist. Read the web for details.

You should configure the newsreader code in this application to know where your local news (NNTP) server is. Type in a terminal window
Welcome to Amazon.com Books!

One million titles, consistently low prices.

(If you explore just one thing, make it our personal notification service. We think it's very cool!)

Spotlight! -- August 16th
These are the books we love, offered at Amazon.com low prices. The spotlight moves EVERY day so please come often.

One Million Titles
Search Amazon.com's million title catalog by author, subject, title, keyword, and more... Or take a look at the books we recommend in over 20 categories... Check out our customer reviews and the award winners from the Hugo and Nebula to the Pulitzer and Nobel... and bestsellers are 30% off the publishers list...

Eyes & Editors, a Personal Notification Service
Like to know when that book you want comes out in paperback or when your favorite author releases a new title? Eyes, our tireless, automated search agent, will send you mail. Meanwhile, our human editors are busy previewing galleys and reading advance reviews. They can let you know when especially wonderful works are published in particular genres or subject areas. Come in, meet Eyes, and have it all explained.

Your Account
Check the status of your orders or change the email address and password you have on file with us. Please note that you do not need an account to use the store. The first time you place an order, you will be given the opportunity to create an account.
Registration

New to Amazon.com? Register Below.

My name is: [Text Input]

My e-mail address is: [Text Input]

Type it again: [Text Input]

My mobile phone number is: [Text Input] (Optional)

Learn more

Protect your information with a password
This will be your only Amazon.com password.

Enter a new password: [Text Input]

Type it again: [Text Input]

Create account

Conditions of Use Privacy Notice © 1996-2012, Amazon.com, Inc. or its affiliates
Registration

New to Amazon.com? Register Below.

My name is: Joe User

My e-mail address is: user@user.com

Type it again: user@user.com

My mobile phone number is: 111-222-3333 (Optional)

Learn more

Protect your information with a password
This will be your only Amazon.com password.

Enter a new password: ••••••

Type it again: ••••••

Create account
<form action="/some/uri" method="post">
    Name: <input type="text" name="name"/>
    Email: <input type="text" name="email"/>
    <input type="submit" value="click me"/>
</form>

POST /some/uri
...headers...
Name=Joe+User&Email=user@user.com
click button

GET

wait

HTML form

POST field data

wait

fill in fields;
click button

wait

new page
<table>
<thead>
<tr>
<th>How to</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to rap</td>
<td>81,300,000</td>
</tr>
<tr>
<td>How to raise a dragon</td>
<td>4,790,000</td>
</tr>
<tr>
<td>How to raise your IQ by eating gifted children</td>
<td>14,200</td>
</tr>
<tr>
<td>How to raise money</td>
<td>38,000,000</td>
</tr>
<tr>
<td>How to raise chickens</td>
<td>13,000,000</td>
</tr>
<tr>
<td>How to raise a dragon walkthrough</td>
<td>1,740,000</td>
</tr>
<tr>
<td>How to raise your metabolism</td>
<td>326,000</td>
</tr>
<tr>
<td>How to raise blood pressure</td>
<td>86,100,000</td>
</tr>
<tr>
<td>How to rationalize the denominator</td>
<td>67,400</td>
</tr>
<tr>
<td>How to raise a kitten</td>
<td>1,500,000</td>
</tr>
</tbody>
</table>
AJAX
Asynchronous JavaScript And XML

- Based on XMLHTTP in Microsoft Internet Explorer (1999): async HTTP requests
- Jesse James Garrett, 2005
  http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications
GET

initiate async request

response

update page

click, type...

click, type...

click, type...
Why AJAX works

• JavaScript functions (callbacks) called on events
• UI freezes during callbacks, so they must be fast: never wait for I/O
• Browser runs multiple HTTP requests asyncly
  – Returns immediately on initiating request
  – Callback later on completion
• Callbacks alter the displayed page by modifying its in-memory rep (the DOM)
• Browser refreshes display without flickering
DOM

Document Object Model

• In-memory representation of HTML page
• Matches tree structure of HTML
• DOM is JavaScript API to this tree data structure
  – Common usage refers to data structure itself
<p id="test">This is the <b>HTML</b> of a paragraph element.</p>
```javascript
var testPara = document.getElementById("test");
var boldElement = testPara.childNodes[1];

var boldElement = testPara.childNodes[1];
```

```
"This is the " B " of a paragraph element."
```

```
"HTML"
```
```javascript
var testPara = document.getElementById("test");
var boldElement = testPara.childNodes[1];
```

**NodeList is not an array!**
var testPara = document.getElementById("test");
var boldElement = testPara.childNodes[1];
var testPara = document.getElementById("test");
var boldElement = testPara.childNodes[1];

JQuery

JQuery children ≠ DOM children
:eq(0) = :nth-child(1)
<p>Another paragraph.</p>

This is the " of a paragraph element."

id="test"
function appendParagraph() {
    var newPara = document.createElement("p");
    var newText = document.createTextNode("Another paragraph.");
    newPara.appendChild(newText);
    var testPara = document.getElementById("test");
    testPara.parentNode.insertBefore(newPara, testPara.nextSibling);
}

demo

$('<p>Another paragraph.</p>').insertAfter($('#test'))
Object Relational Mapping

<table>
<thead>
<tr>
<th>sites</th>
<th>ORM</th>
<th>sites</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id: &quot;somesite&quot;</td>
<td>count: 1</td>
<td>id</td>
</tr>
<tr>
<td>id: &quot;anothersite&quot;</td>
<td>count: 2</td>
<td>&quot;somesite&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;anothersite&quot;</td>
</tr>
</tbody>
</table>

Object HTML Mapping

Element

- **tagName**: "P"
- **childNodes**: NodeList

<p>This is the <b>HTML</b> of a paragraph element.</p>
SQL → ORM
DOM → JQuery
DOM

**enqueue**

**update**

**response**

**initiate**

*render events*

*events*

*JavaScript event queue*

*XML HTTP sessions*
JavaScript events

• JavaScript functions (callbacks) are registered in the DOM
  – `<button onClick="myCallback()">`
  – also in XMLHttpRequest call
• Events queue up callbacks to run later
• Queue is single-threaded: one at a time
• Callbacks must execute quickly to avoid hanging the UI. No crunching or waiting!
Auto-capitalized field

<input id='test' type='text'>

<script>
    document.getElementById('test').onkeypress = function(event) {
        this.value = this.value.toUpperCase();
    }
</script>

<demo>

http://javascript.info/tutorial/events-and-timing-depth
<input id='test' type="text">

<script>
document.getElementById('test').onkeyup = function(event) {
    this.value = this.value.toUpperCase();
};
</script>

demo
<input id='test' type='text'>

<script>
document.getElementById('test').onkeypress = function(event) {
    var elem = this;
    var after = function() {
        elem.value = elem.value.toUpperCase();
    };
    setTimeout(after, 0);
};
</script>
What's the story?

Event queue

// Read keyboard scan code
// Trigger keypress event
// Insert character into DOM element
keypress event is synchronous!
synchronous events not queued

Event queue

```javascript
// Read keyboard scan code
// Trigger keypress event
this.value = this.value.toUpperCase();

// Insert character into DOM element
```
// Read keyboard scan code
// Trigger keypress event

setTimeout(after, 0)

// Insert character into DOM element

this.value = this.value.toUpperCase()
Welcome to Callback Hell

• All the horror of multi-threaded programming in a single thread!
• Async callbacks execute single-threaded
  – Not executed inside another async callback
  – Order undefined
• Sync callbacks execute anywhere anytime
  – DOM change events are all synchronous!
• Docs must give order and state guarantees
  – AS IF!
1.6.3. Key events

The DOM Level 2 Event specification does not provide a key event module. An event module designed for use with keyboard input devices will be included in a later version of the DOM specification.
Processing of key events

There are keydown, keypress, and keyup events. For most keys, Gecko dispatches a sequence of key events like this:

1. When the key is first depressed, the keydown event is sent.
2. If the key is not a modifier key, the keypress event is sent.
3. When the user releases the key, the keyup event is sent.

5.2.5.1 Keyboard Event Order

The keyboard events defined in this specification occur in a set order relative to one another, for any given key:

1. **keydown**
2. **keypress** (only for keys which produce a **character value**)
3. If the key is depressed for a sustained period, the following events may repeat at an environment-dependent rate:
   - **keydown** (with **repeat** attribute set to true)
   - **keypress** (with **repeat** attribute set to true; only for keys which produce a **character value**)
4. Any **default actions** related to this key
5. **keyup**
Search Results

Want better search results? See our search tips!

JavaScript is a dynamically-typed language commonly used for client-side scripting. It is NOT the same as Java. Use this tag for questions regarding ECMAScript and its dialects/implementations (excluding ActionScript). If a framework or library, such as jQuery, is used, include that tag as well.

jQuery keypress() event not firing?

18 votes
7 answers
37k views

Pause after first keypress

1 vote
1 answer
279 views

Javascript function on keypress?

1 vote
1 answer
input

The DOM input event is fired synchronously when the value of an `<input>` or `<textarea>` element is changed. Additionally, it's also fired on contenteditable editors when its contents are changed. In this case, the event target is the editing host element. If there are two or more elements which have contenteditable as true, "editing host" is the nearest ancestor element whose parent isn't editable. Similarly, it's also fired on root element of designMode editors.

**Interface:** Event
- Synchronicity: synchronous
- Bubbles: yes
- Target: Element
- Cancelable: no
- Default action: none

**Specification**

**HTML5**

**Browser compatibility**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Chrome</th>
<th>Firefox (Gecko)</th>
<th>Internet Explorer</th>
<th>Opera</th>
<th>Safari</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic support</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>(Yes)</td>
<td>(Yes)</td>
</tr>
<tr>
<td>immediately after compositionupdate</td>
<td>(Yes)</td>
<td>12.0 (12)</td>
<td>(Yes)</td>
<td>--</td>
<td>(Yes)</td>
</tr>
<tr>
<td>on contenteditable element</td>
<td>(Yes)</td>
<td>14.0 (14)</td>
<td>(Yes)</td>
<td>--</td>
<td>(Yes)</td>
</tr>
<tr>
<td>when designMode is &quot;on&quot;</td>
<td>(Yes)</td>
<td>14.0 (14)</td>
<td>--</td>
<td>--</td>
<td>(Yes)</td>
</tr>
</tbody>
</table>

Opera does not fire an input event after dropping text in an input field.
IE 9 does not fire an input event when the user removes characters from input filled by keyboard, cut, or drag operations.
Cocoa Event Handling Guide

Handling Mouse Events  51
  Overview of Mouse Events  51
  Handling Mouse Clicks  52
  Handling Mouse Dragging Operations  56
    The Three-Method Approach  57
    The Mouse-Tracking Loop Approach  59
    Filtering Out Key Events During Mouse-Tracking Operations  62

Handling Key Events  66
  Overview of Key Events  66
    Overriding the keyDown: Method  67
    Handling Keyboard Actions and Inserting Text  68
    Specially Interpreting Keystrokes  71
  Handling Key Equivalents  74
  Keyboard Interface Control  75

Using Tracking-Area Objects  78
  Creating an NSTrackingArea Object  78
  Managing a Tracking-Area Object  80
  Responding to Mouse-Tracking Events  81
  Managing Cursor-Update Events  82
  Compatibility Issues  83
Cocoa is Apple's application-development framework for Mac OS X, consisting of Foundation, Application Kit, and Core Data. For iOS questions, use [cocoa-touch].
• 1/3 of the code in Adobe’s desktop applications is devoted to event handling logic
• 1/2 of the bugs reported during a product cycle exist in this code

— Sean Parent, Adobe Software Technology Lab 2006
Web app trends

• JavaScript on server: Node
• Desktop-inspired frameworks
  – Spine, Backbone
• Async data replication instead of requests
  – Meteor
• Fix/replace JavaScript
  – CoffeeScript, Harmony, Google Closure, Dart
Apps vs. Web
<table>
<thead>
<tr>
<th>Apps</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branded devices</td>
<td>Everywhere</td>
</tr>
<tr>
<td>Evolving from desktop</td>
<td>Evolving from forms</td>
</tr>
<tr>
<td>Single vendor control</td>
<td>Open standards</td>
</tr>
<tr>
<td>Comprehensive features</td>
<td>Gaps and incompatibilities</td>
</tr>
<tr>
<td>Complete docs</td>
<td>Sketchy docs</td>
</tr>
<tr>
<td>Complete dev tool chain</td>
<td>Basic dev tools</td>
</tr>
<tr>
<td>Gilded cage</td>
<td>Laissez-faire</td>
</tr>
</tbody>
</table>
A Generation Lost in the Bazaar

- [http://queue.acm.org/detail.cfm?id=2349257](http://queue.acm.org/detail.cfm?id=2349257)
- Firefox has 122 package dependencies
- FreeBSD ports contains 24,400 patch files
- 1,342 pasted copies of same crypto code
- *Quality happens only when someone is responsible for it*

—Fred Brooks
<table>
<thead>
<tr>
<th><strong>Apps</strong></th>
<th><strong>Web</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Branded devices</td>
<td>Everywhere</td>
</tr>
<tr>
<td>Evolving from desktop</td>
<td>Evolving from forms</td>
</tr>
<tr>
<td>Single vendor control</td>
<td>Open standards</td>
</tr>
<tr>
<td>Comprehensive features</td>
<td>Gaps and incompatibilities</td>
</tr>
<tr>
<td>Complete docs</td>
<td>Sketchy docs</td>
</tr>
<tr>
<td>Complete dev tool chain</td>
<td>Basic dev tools</td>
</tr>
<tr>
<td>Gilded cage</td>
<td>Laissez-faire</td>
</tr>
<tr>
<td>Cathedral</td>
<td>Bazaar</td>
</tr>
<tr>
<td>Evil?</td>
<td>Good?</td>
</tr>
</tbody>
</table>