HTML

Monday, March 16, 2015 8:39 PM

Key events

I used to use ".which" on key events like this:

```
if (e.which === KEYCODE.RETURN) { foo(); }
```

Then I would manifest a file full of the keycodes, and those would look like this:

```
export const RETURN = 13;
export const A = 'A'.charCodeAt(0); // similar for other letters
export const ONE = '1'.charCodeAt(0); // similar for other numbers
```

That's super tedious, but it's also deprecated. You can now use either "e.key" or "e.code", both of which are supported on modern browsers.

e.key (reference): it's a string, you don't really need constants manifested somewhere since they're already human-readable and will never change, and you can differentiate between locations (e.g. "main keyboard" vs. numpad). Example:

```
if (e.key === 'Enter') { foo(); }
```

e.code: it's a string representing the physical key that you pressed, so if I change keyboard layouts and press '/', it'll still be "Slash" instead of "é" on a [Canadian] French keyboard.

Note that "e.key" will be uppercase if you're holding shift, so if you want to use "e.key" to see if "W" is being held regardless of the state of shift, then you need to check for both "w" and "W" (either with toLowerCase() or just two comparisons). Otherwise, you could just use "e.code" and check against "KeyW" regardless of whether shift is held.

Favicons

This site generates all of the different favicons that you need and also tells you why you need them: https://realfavicongenerator.net/faq

Testing on common screen sizes

http://quirktools.com/screenfly/

onbeforeunload (reference)

If you want to give the user a chance to save their progress before leaving a page, you can use the "beforeunload" event:

```
window.addEventListener('beforeunload', (e) => {
    const dialogText = 'Dialog text here';
    e.returnValue = dialogText;
    return dialogText;
});
```

Placeholder images

If you want placeholder images of particular sizes, here are some helpful services:

- http://placehold.it/ - images are just pictures of text (e.g. "950 x 440")

- http://lorempixel.com/ - images are real images
- https://placekitten.com/ - images are real images, but all pictures are of cats

target="_blank" vulnerability (reference)

Without setting a 'rel' attribute, the page that you open could control its parent page (e.g. with a redirect). To fix it, just set that attribute as shown below:

```
<a href="blah" target="_blank" rel="noopener noreferrer">Click me</a>

You could also use this to safely open from JS:
function safeOpen(url) {
  const w = window.open(undefined, '_blank');
  w.opener = null;
  w.location = url;
}
```

Also, there's a possible performance implication if you don't set the 'rel' tag (reference).

==========

5/23/2015

Jade: CSS style tags (just put a period at the end of the "style" line).

```
doctype html
html
  head
    title= title
    script(type='text/javascript', src='client/misc/loginpage.js')
    style(type='text/css').
       li {
          padding-top: 9px;
          padding-bottom: 3px;
          color: blue;
       }
  body(onload='slotmachines.client.login.main.start()')
     h1= Login
```

==========

4/19/2015

Jade: hrefs inside a list

Note: there are spaces after the pipes; they're very important!

```
  ul
    li.big-list-items: a(href="/login") Login
    li.big-list-items
      a(href="/game") Game
      |
      a(href="/game?user=1") (user1)
      |
      a(href="/game?user=2") (user2)
      |
      a(href="/game?user=3") (user3)
      |
```

```
    a(href="/game?user=4") (user4)
    |
    a(href="/game?user=5") (user5)
  li.big-list-items: a(href="/validate") Validator
  li.big-list-items: a(href="/colorpicker") Color-picker
```

==========

4/12/2015

Suppose you have this:
HTML:

```
<div>
        <button>Click me</button>
</div>
```

JS:

```
$('div').click(function(){console.log('clicked div');});
$('button').click(function(event){console.log('clicked button');});
```

Clicking the button will actually output "clicked div" AND "clicked button", so use event.stopPropagation() to prevent it from going to the parent.

Note: "return false;" at the end of a handler is the same as doing what's below (reference):

```
        event.stopPropagation()
        event.preventDefault()
```

Quirk: could NOT figure out how to make this work with checkboxes (<input> tags). There are apparently some StackOverflow threads on this:

- http://stackoverflow.com/questions/2360655/jquery-event-handlers-always-execute-in-order-they-were-bound-any-way-around-t
- http://stackoverflow.com/questions/290254/how-to-order-events-bound-with-jquery

My solution in the end was to do something really hacky:

```
  // Put this in the parent element that contains a checkbox. By adding this, clicking
  // the checkbox will still cause this code to be hit, but the 'if' check will force an early return.
  var $eventTarget = $(event.target);
  if ($eventTarget.hasClass('ui-button-text') || $eventTarget.attr('type') === 'checkbox') {
    return;
  }
```

==========

3/16/2015
This is a great resource on float (reference).

You should also read on clearing floats though (reference).

Related:
When making two divs next to each other, you should really just use inline-block on both (reference) instead of trying to float things weirdly. UPDATE/NOTE: inline-block elements should technically be converted into spans (reference).

HTML:

```
<div class="left-div">
```

```
        <textarea type='text'>This is a text area</textarea>
</div>
<div class="right-div">
        <textarea type='text'>This is a text area</textarea>
</div>
```

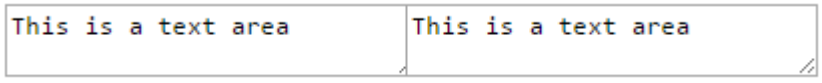
CSS:

```
.left-div {
   display: inline-block;
   width: 200px;
}

textarea {
   width: 100%;
}

.right-div {
   display: inline-block;
   width: 200px;
}
```

Output:



To make something go beneath it (e.g. a set of buttons), just put the buttons in their own div as a sibling to the two existing divs.

If the contents of the divs get pushed around weirdly, then add vertical-align:top to them (reference).

Also, you need to make sure you have specific widths assigned to each.

==========