JavaScript

Class 20

User Experience

- the purpose of JavaScript is to enhance the user experience
- JS makes the browser interface more convenient
  - features not available in HTML
  - features available in PHP, without consulting the server
- JS cannot be used for the server's security

Events

- JS is event-driven
- many events exist, each of which can trigger JS

Events

onclick	mouse click on element
ondblclick	mouse double click on element
ondrag	element is being dragged
ondragend	at end of drag operation
ondragenter	element has dragged to valid drop target
ondragleave	element leaves valid drop target
ondragover	element is dragged over valid drop target
ondragstart	at start of drag operation
ondrop	when dragged element is dropped
onmousedown	mouse button is pressed down on element
onmousemove	mouse pointer moves over an element
onmouseout	mouse pointer moves out an element
onmouseover	mouse pointer moves over on element
onmouseup	mouse button is released over an element

Events

onmousewheel	mouse wheel being rotated
onscroll	element scrollbar being scrolled
onkeydown	while pressing a key
onkeypress	after press and release key
onkeyup	after release a key
onblur	element loses focus
onchange	element value is changed
onfocus	element gets focus
onformchange	form changes
onforminput	form get input field
oninput	element get input field
onselect	after select text in an element
onsubmit	form is submitted

Events

- example only-one-box.html

Display and Visibility

- a block element has the visible property:

visible	the default, the box is visible
hidden	the box is invisible, not drawn, but still takes up room as though it were there; the element cannot receive focus
collapse	only for tables; we won't use it

- the display property, which is in major transition
  here, we are only considering two values

none	the box is as though it does not exist; it is not visible and does not occupy any space in the layout
block	the default for a block element

Disabled

- a form element may have the Boolean attribute disabled
- a disabled element is not mutable or focusable
- a disabled element is not submitted with the form (so it doesn't appear in $_POST)
- a disabled element does not generate key or mouse events

- using visibility and disabled, we can control what a user can see and do

visibility.html

Fat Arrow

- the $ function

```
function $(id)
{
  return document.getElementById(id);
}
```

- JS6 and above has several abbreviated syntaxes for functions

- first, we can get rid of the "function" keyword

```
const $ = (id) =>
{
  return document.getElementById(id);
};
```

- note that this is now an assignment statement, and so needs a semicolon at the end.

Fat Arrow

```
function $(id)
{
  return document.getElementById(id);
}
```

vs

```
const $ = (id) =>
{
  return document.getElementById(id);
};
```

- Douglas Crockford is a big name in JS
- he invented JSON and did lots of other amazing stuff
- he calls the JS fat arrow operator the fart operator
- not everyone does, but it's a cute name

Step Two

- just losing the function keyword didn't buy us much
- the function body is a single statement, and so doesn't need curly braces or two semicolons
- and since it's understood that a function returns a value, we can lose the "return" keyword to give

```
const $ = (id) => document.getElementById(id);
```

vs

```
function $(id)
{
  return document.getElementById(id);
}
```

Short Form

```
const $ = (id) => document.getElementById(id);
```

- this short form can be used when the function body is a single return statement

Anonymous Function Expression

- when a function consists of multiple statements, or has no return, we can still use an anonymous function expression
- this is typically seen as the function to be registered as a callback

```
window.onload = function()
{
  $("serif").onclick = fontClick;
  $("sans-serif").onclick = fontClick;
  $("monospace").onclick = fontClick;
};
```

- here, the onload event triggers a callback function
- the function doesn't need a name, it just needs to work

Anonymous Function Expression

- this can be carried to subsequent levels
- in the first example, only-one-box.js, we had:

```
window.onload = function ()
{
  $("text1").onkeypress = clear2;
  $("text2").onkeypress = clear1;
};

function clear2()
{
  $("text2").value = "";
}

function clear1()
{
  $("text1").value = "";
}
```

- but clear1 and clear2 can also be anonymous functions

Anonymous Function Expressions

```
window.onload = function ()
{
  $("text1").onkeypress = function () {$("text2").value = "";};
  $("text2").onkeypress = function () {$("text1").value = "";};
};
```

for Loop

- earlier in the style example we saw a for loop

```
function fontClick()
{
  const lyriclist = document.getElementsByClassName("lyricline");
  for (let i = 0; i < lyriclist.length; i++)
  {
    lyriclist[i].classList.remove("familynormal");
    lyriclist[i].classList.remove("familysans-serif");
    lyriclist[i].classList.remove("familymono");
    lyriclist[i].classList.add(this.value);
  }
}
```

- this works, but is not JS-esque

forEach Loop

- the main use of a for loop is to iterate over the elements of an array
- for this purpose, the forEach loop combined with an anonymous function expression is much better suited

```
function fontClick()
{
  document.getElementsByClassName("lyricline").forEach(function (item)
  {
    item.classList.remove("familynormal");
    item.classList.remove("familysans-serif");
    item.classList.remove("familymono");
    item.classList.add(this.value);
  });
}
```

this is style2.html and style2.js

Clean Code

- like the C family of languages, JS has evolved over the years
- early mistakes are still in the language
  - == type coercion vs === strict equality
  - terminal semicolons are not required, except when they are
  - curly braces are not required, except when they are

Lint

- in 1978 Stephen Johnson at Bell Labs was working to port Unix to a new architecture
- the C compiler back then was crude compared to today's compilers
- Johnson wrote lint, a tool to catch small errors in C code that the compiler let slip through
- named for lint, the tiny bits of fiber and fluff shed by fabric in a clothes dryer
- tiny bits that add up to a big problem

JSlint

- in 2002 Douglas Crockford wrote JSlint
- a static program analysis tool for JS, to catch suspect constructions and style mistakes
- Crockford has incredibly strong opinions about code quality and style, just like a professor you may know
- tabs not allowed, line length, strict indenting, braces required, semicolons required
- check your JS style at `https://www.jslint.com`