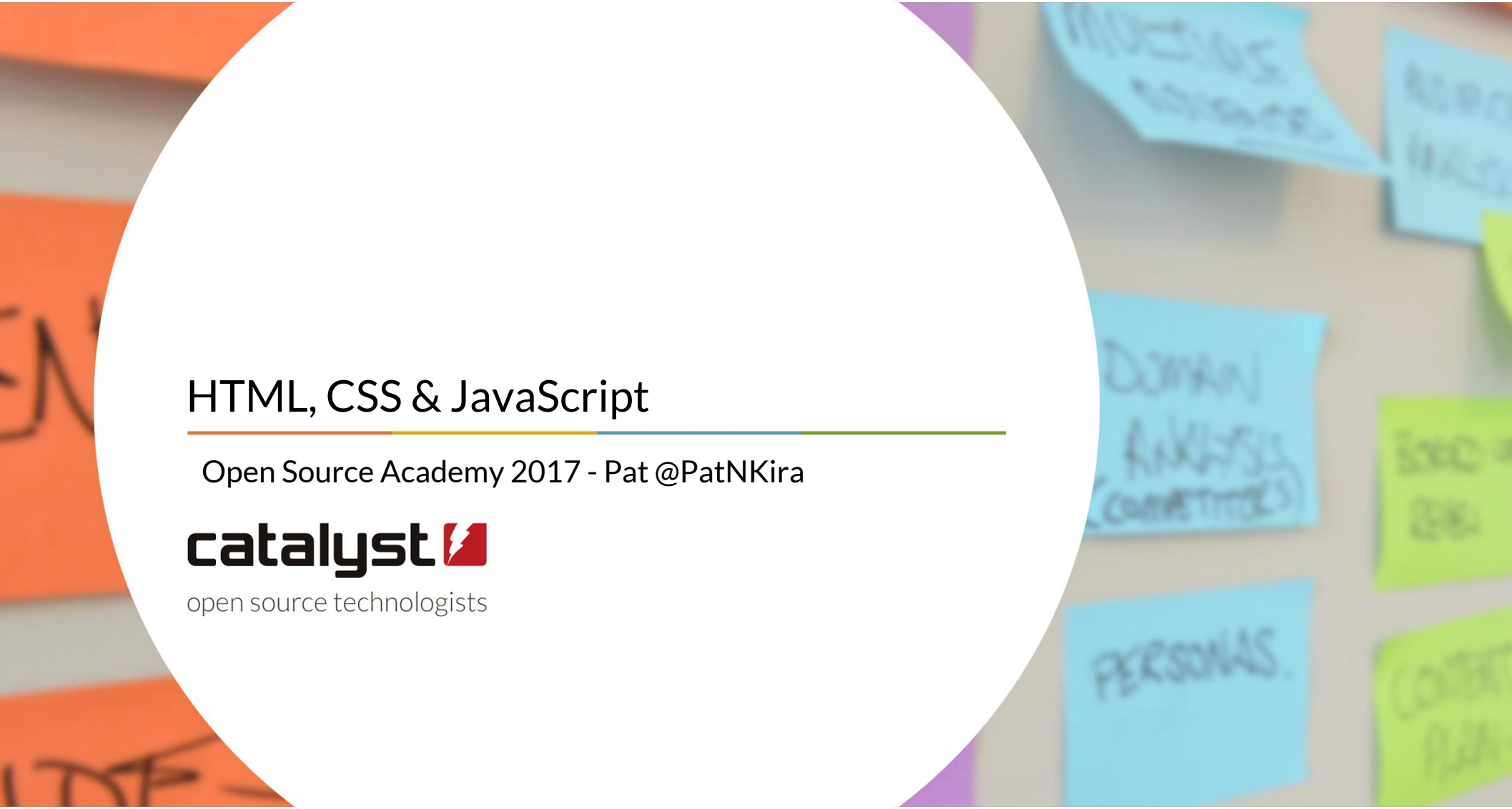

HTML, CSS & JavaScript

Open Source Academy 2017 - Pat @PatNKira

catalyst

open source technologists

Getting started

- Who here has done some HTML before?
- CSS?
- JavaScript? jQuery?

What are HTML & CSS?

HTML - HyperText Markup Language

Use for structuring content such as headings, paragraphs, images

CSS - Cascading Style Sheets

Use for styling content such as font colour, positioning content

HTML elements

Heading - <h1>, <h2>

Text - <p>,

Link - <a href="http://www.oswd.org/">

Image - <img src="mdn-logo-sm.png" alt="">

List - , <ol> with

Layout - <header>, <footer>

Content layout - <aside>, <nav>

Container - ,

Basic HTML page structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Cats are awesome!</title>
  </head>
  <body>
    <h1>Welcome to awesome cats</h1>
    <p>Rub face on everything purr for no reason toy.</p>
  </body>
</html>
```

CSS - Cascading Style Sheets

We're using it to **style** the page.

It's **separate** from the **HTML** (it is on its own **sheet**).

It is **cascading**.

Why do we need it?

Content
(what it is)

Presentation
(how it looks)

`Always use external styles`

By adding a link to a separate stylesheet in the <head> block at the top of the HTML file

A simple CSS rule

```
/* Selecting element */
p { ... }

/* CSS properties */
p {
  color: ...;
  font-weight: ...;
}

/* Assigning properties some values */
p {
  color: red;
  font-weight: bold;
}
```

CSS selectors: element selectors

```
p {
  color: green;
}

div {
  background-color: blue;
}

ul, ol {
  margin: 0;
}
```

Notice also that the last example uses two selectors, comma separated!

CSS selectors: ID selectors

```
<!--HTML -->
<p id="my-unique-name">Unique is my name.</p>
```

```
/* CSS */
#my-unique-name {
  color: red;
}
```

CSS selectors: class selectors

```
<!--HTML -->
<p class="example">Text</p>
<p class="example">Text</p>
```

```
/* CSS */
.example {
  color: green;
  font-style: italic;
}
```

CSS selectors: mixing selectors

```
<!--HTML -->
<p><span class="example">Text</span></p>
<p class="example"><span>Text</span></p>
```

```
/* CSS */
p.example {
  color: #ffffff;
  font-style: italic;
}
.example span {
  color: #ffffff;
  font-style: italic;
}
```

CSS specificity

```
p {
  color: red;
}

p {
  color: blue;
}
```

Will the paragraphs be red or blue?

CSS specificity

- The **last** rule wins!
- Unless the higher rule is **more specific**
- **Class selectors** are more specific than element selectors.
- **ID selectors** are more specific than class selectors.

Padding and margin

Padding and margin

```
p {
  margin: 10px 5px 20px 15px;
  /* top right bottom left */
}
p {
  margin: 10px 20px;
  /* top,bottom right,left */
}
p {
  margin: 10px 5px 20px;
  /* top right,left bottom */
}
```

Border

```
div {
  border: 1px solid #ffffff;
  /* width type colour */
}
div {
  border-radius: 3px;
}
```

Display

Inline

Hodor, hodor hodor hodor - hodor hodor hodor hodor
hodor. Hodor. Hodor, hodor... Hodor hodor hodor
hodor?! Hodor. Hodor hodor... Hodor hodor hodor;
hodor HODOR hodor, hodor hodor hodor! Hodor hodor
HODOR! **Hold the door** Hodor HODOR
hodor, hodor hodor hodor! Hodor. Hodor hodor...
Hodor hodor hodor, hodor. Hodor hodor, hodor. Hodor
hodor?! Hodor. Hodor hodor; hodor hodor; hodor
hodor. Hodor. Hodor, hodor hodor hodor hodor? Hodor
hodor... Hodor hodor hodor; hodor hodor; hodor
hodor; hodor hodor?!

Inline-block

Hodor, hodor hodor hodor - hodor hodor hodor hodor
hodor. Hodor. Hodor, hodor... Hodor hodor hodor
hodor?! Hodor. Hodor hodor... Hodor hodor hodor;
hodor HODOR hodor, hodor hodor hodor! Hodor hodor

HODOR! **Hold the door** Hodor HODOR hodor,

hodor hodor hodor! Hodor. Hodor hodor... Hodor
hodor hodor, hodor. Hodor hodor, hodor. Hodor
hodor?! Hodor. Hodor hodor; hodor hodor; hodor
hodor. Hodor. Hodor, hodor hodor hodor hodor? Hodor
hodor... Hodor hodor hodor; hodor hodor; hodor
hodor; hodor hodor?!

http://codepen.io/anon/pen/ygeRvJ

Display

Block

Hodor, hodor hodor hodor - hodor hodor hodor hodor hodor. Hodor. Hodor, hodor... Hodor hodor hodor hodor?! Hodor. Hodor hodor... Hodor hodor hodor; hodor HODOR hodor, hodor hodor hodor! Hodor hodor HODOR!

Hold the door

Hodor HODOR hodor, hodor hodor hodor! Hodor. Hodor hodor... Hodor hodor hodor, hodor. Hodor hodor, hodor. Hodor hodor?! Hodor. Hodor hodor; hodor hodor; hodor hodor. Hodor. Hodor, hodor hodor hodor hodor? Hodor hodor... Hodor hodor hodor; hodor hodor; hodor hodor; hodor hodor?!

None

Position

Static - default value.

Relative - positioned relative to its normal position.

Absolute - positioned at specified position relative to its closest positioned ancestor.

Fixed - positioned relative to the browser window

```
p {
  position: relative;
  top: -10px
}
/* This would move it up 10 pixels from its natural position. */
```

Float

For even fancier layouts, we can use 'float: left' or 'float: right' to allow text to wrap:

We want to 'float' this image so the around it. We can float things either to the left or the right. We need to be careful though, floated elements don't sit in the page properly any more. Read about 'clearing' to learn about this.

Pseudo Selectors

We can style the **hover state** for an element such as a link.

```
a:hover {
  color: green;
}
```

We can style the **first or last** of a type of thing:

```
p:first-child,
p:last-child {
  color: red;
}
```

How to debug

- Modern browsers give us great tools for checking how our CSS is being interpreted.
- Right click on any element at choose the 'inspect element' option, or press F12 to bring up the developer tools.

JavaScript

- **Client side** programming language
- Rich interactivity
- Event driven
- Use to dynamically change HTML and/or CSS

Including JavaScript

- Like CSS, we put it in a separate file which is linked from the .html file
- At the bottom of the page, just before the </body>, we put in the link:

```
<script src="js/script.js" type="text/javascript"></script>
```

A simple bit of JavaScript

```
var a = 1;
var b = 1;
if (a === b) {
  alert('a is equal to b');
}
```

In the code above, we set up up to **variables** to store values. Every line ends in a **semicolon**. Then we use an **if statement** to compare the values, which triggers an **alert box**.

Interacting with the page

That simple JS example just has the variables set to be 1.
In a real example we probably want to get information out of the .html page – directly from the page's DOM (document object model)

We can use a **JavaScript library** to help us with this. We will use a popular library called **jQuery.**

Including jQuery

We just need to link to the library code at the bottom of the .html page, **above** the link to our **script.js** file:

https://code.jquery.com/

Writing some jQuery

In our script.js, to start using jQuery we have to put all of our code inside a special wrapper, to make sure the code runs after the .html has finished loading.

```
$(document).ready(function() {
  // our code goes in here
});
```

Selection elements

jQuery uses **CSS style selectors** and each selector has to be wrapped up so jQuery understands it.

```
$('#our-id-name')
$('.our-class-name')
```

Catching events

We can can trigger JavaScript code when certain **events** happen, like **click** or **mouseover** or **keypress**

Here is an example of putting a movie's title in a alert when the user clicks on its poster image:

```
$('.poster').click(function () {
  /* everything inside here only
  happens after the click */
  var title = $(this).next().text();
  alert(title + ' was picked!');
});
```

Let's write some JS

- There are some instructions on your instruction sheet.
- We haven't covered all of the jQuery functions you will need to use, you will have to refer to the **online documentation** at http://api.jquery.com/

Debugging

- Like with CSS, we can use built-in browser tools to help us figure out problems in our code
- Press F12 or right click on the page and inspect element. Navigate to the 'console' tab. Any errors will show up here.
- We can also use special **console.log()** statements in our code to output information here.

Some further JS concepts

Operators

- Assignment
- Arithmetic operators like + (addition) and * (multiply)

http://eloquentjavascript.net/01_values.html

- Arithmetic operators like **+** (addition) and * (multiply)
- Comparison operators like **===** (equal to) **!==** (not equal to) and **>** (greater than)
- Booleans like **&&** (and) and **||** (or)

Read more at http://eloquentjavascript.net/01_values.html

Data structures

- Strings are in '' or ""
- Numbers are either **integers** (whole) or **floats** (with decimals)
- Booleans are either **true** or **false**
- Arrays are unlabelled collections of values

```
["value", "other val", 5, 7]
```

- Objects are labelled collections of values:

Read more at http://eloquentjavascript.net/04_data.html

```
{
  name: "Sarah",
  age: 21
}
```

Functions

- A function is a way of wrapping up a bit of reusable code

Read more at http://eloquentjavascript.net/03_functions.html

```
function myFunction(parameter) {
  parameter = parameter + " done!" ;
  return parameter;
}
```