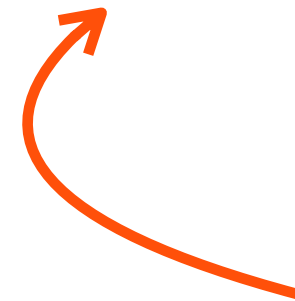
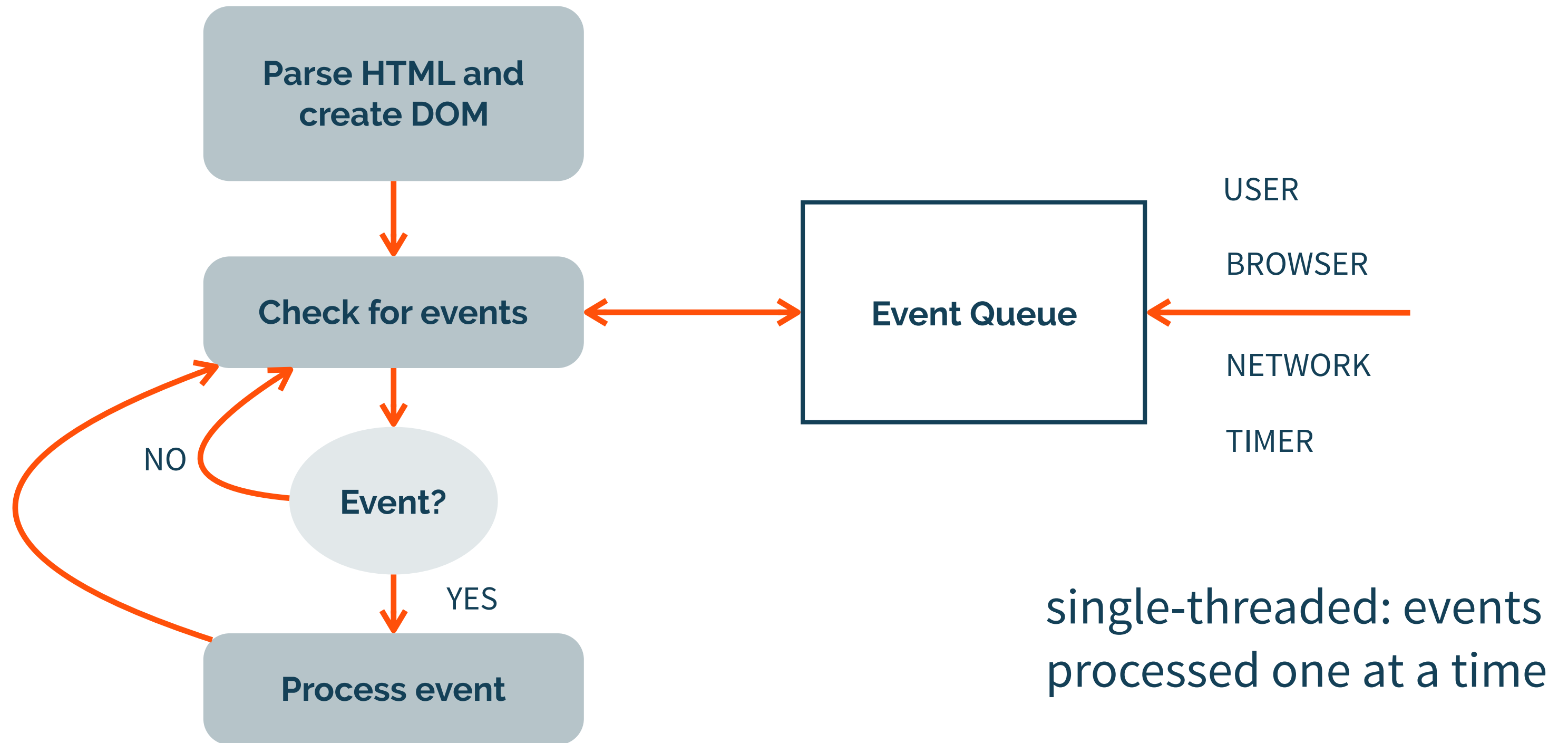


# JAVASCRIPT



*and the Web!*

# THE BROWSER EVENT LOOP



# EVENT OBJECT

contains the information about the event

HTML `<div onclick="mouseClick(event) ;">`

DOM `element.onclick = mouseClick;`

`function mouseClick(event) {...} ;`

DOM (IE) `function mouseClick() {...`  
`x = window.event.clientX;`  
`...} ;`

# EVENT PROCESSING

events propagate in two phases

*capture phase*: root to innermost element

*bubble phase*: innermost element to root

DOM standard: *capture* then *bubble*

# EVENT PROCESSING

```
element.addEventListener(event,  
function, useCapture)
```

 *set capture or bubble phase*

```
event.stopPropagation()
```

CODEPEN

# JAVASCRIPT



*the language*

# RECOMMENDED JS REFERENCE



free PDF available online

# BASIC DATA TYPES

Booleans: **true**, **false**

Number: no integers, 64-bit floating point

String: no char, variable-length

Special Types: **null**, **undefined**



*get familiar with String methods*



# VARIABLES

Dynamically typed: types associated with values, not with variables

Use `var` to define local variables

variables defined implicitly through assignment have global scope

scopes are declared through  
functions and not blocks { }

# CHECKING EQUALITY

```
console.log(false == 0)
```

Things can be complicated when the values on both sides of the `==` operator are of different types

Use `===` or `!==` to check for precise equality

# ARRAYS

```
var classes = new Array();
```

```
classes[3] = 'cs199rk';
```


```
var numbers = [5, 3, 2, 6];
```

```
numbers.length;
```

```
other methods: push, pop, sort, ...
```

# CREATING FUNCTIONS

*name*



```
function eat() {...}
```

```
var sleep = function() {...}
```

*anonymous function*



# OBJECTS

collection of properties: name-value pairs

```
llama = {color: 'brown', age: 7,  
        hasFur: true};
```

add new properties on the fly

```
llama.family = 'camelid';
```

# jQuery

ONE FRAMEWORK TO RULE THEM ALL



CS498RK SUMMER 2017  
UNIVERSITY OF ILLINOIS @ URBANA-CHAMPAIGN

# jQuery

Extremely powerful

Most widely used

Cross browser

Open source



# WHAT CAN IT DO?

DOM Manipulation & Traversal

Event handling

Styling & Animation

AJAX\*

.... many more

# WHY USE IT?

Much easier syntax than  
native JS

Much better cross browser  
support

# SELECTORS

## **Syntax:**

`$(selector).action()`

- \$ sign to define jQuery object
- 'selector' or query to find the HTML element
- action to be performed

# SELECTORS

## Examples:

```
$('p').hide()
```

```
$('.class').click(function(){...})
```

```
$(this).hide()
```

```
$('#id').on('event', function () {...})
```

# DOM MANIPULATION

html() - <p> hi </p>

text() - hi

val() - value of the input

attr() - href etc..

# DOM MANIPULATION

## Get:

`<p> This is some <i> italicized</i> text </p>`

`$('p').html()` - This is some *italicized* text

`$('p').text()` - This is some italicized text

# DOM MANIPULATION

## Set:

`<p></p>`

`$('#p').html("This is some <i> italicized</i> text")`

`$('#p').text(This is some italicized text)`

# DOM TRAVERSAL

Ancestors

Children

Siblings



# DOM TRAVERSAL

## **Ancestors:**

- `parent()`
- `parents()`
- `parentsUntil()`

# DOM TRAVERSAL

## **Children:**

- children()
- find()

# DOM TRAVERSAL

## **Siblings:**

- siblings()
- next()
- prev()
- nextUntil, prevUntil, ...

# EVENTS

Mouse

Keyboard

Forms

Document

# EVENTS

## **Mouse:**

- hover
- click
- mouseenter
- mouseleave

# EVENTS

## **Keyboard:**

- keypress
- keyup
- keydown

# EVENTS

## Document:

- load
- unload
- **scroll**

# EVENTS

**General syntax:**

```
$(‘selector’).event(callback (){...})
```



# EVENTS

## On method

- attaches one or more event handlers for the selected elements
- `$(‘selector’).on(‘event’, fucntion(){...})`
- `$(‘selector’).on({ event1: callback(){}, event2: callback(){}})`

# CHAINING

chain actions/functions together

```
$(selector).css("color","red").slideUp(2000).slideDown(2000);
```

# STYLING

`addClass()`

`removeClass()`

`toggleClass()`

`css()`

EXAMPLE

# WHY YOU SHOULDN'T USE IT

Relic of the past

- Native JS can do most things now. You probably don't need jQuery.
- Browser compatibility not that big an issue
- native methods 'faster', file size is 'smaller'

# WHY YOU SHOULDN'T USE IT

*“They don’t know what they need jQuery for, so they just use jQuery anyway..” - Lea Verou*

<http://youmightnotneedjquery.com/>

# WHY YOU SHOULDN'T USE IT

*Real reason:*

jQuery uses its own wrapper objects!

You end up with code that mixes jQuery objects, native elements and NodeLists.

And this is where the hell begins

- *Lea Verou*

EXAMPLE