CS1116/CS5018
Web Development 2
Dr Derek Bridge
School of Computer Science & Information Technology
University College Cork
(Acknowledgment: This way of introducing JavaScript is inspired by the methods of Seb Lee-Delisle.)

A client-side JavaScript program

- An HTML Web page, page.html:

```html
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<title>Greetings!</title>
<script src="greetings.js" type="module"></script>
</head>
<body>
</body>
</html>
```

NB These days we write `type="module"`, not `type="text/javascript"`

- A JavaScript program, greetings.js:

```javascript
let now = new Date();
window.alert('Hello world. It is ' + now + ', right now.');
```

Check your understanding

- What will the server do with this JavaScript program?
- What will the browser do with this JavaScript program?
- What goes between the `<script>` start tag and `<script>` end tag?
- Are those semi-colons needed?
- In general, your users would prefer you to avoid writing programs that use the `window.alert()` method. Why?
- Suppose this program is on our server in Cork. Someone in Australia requests it. Whose time/date do they see?

JavaScript

- A programming language in which we write programs designed to be embedded in other software applications
- The core language (sometimes called ECMAScript) has:
  - typical operators, expressions and statements; and
  - core objects, such as Array, Date and Math
- The core is standardized
- Client-side JavaScript extends the core with objects to:
  - control the browser
  - interact with the user
  - communicate with the server and
  - alter the document content
- These extensions are less standardized — can differ from browser to browser
- Other extensions of the core allow JavaScript programs to be used in servers, in PDF documents, ...
- JavaScript and Java are both partly inspired by C but otherwise unrelated
**HTML canvas**

- An HTML Web page, particles.html:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Particles</title>
    <link rel="stylesheet" href="particles.css" />
    <script src="particles.js" type="module"></script>
  </head>
  <body>
    <canvas width="500" height="300">
    </canvas>
  </body>
</html>
```

- A CSS stylesheet, particles.css:

```css
body {
  background-color: black;
}
canvas {
  display: block;
  margin-left: auto;
  margin-right: auto;
  border: 1px solid white;
}
```

**Variables**

JavaScript variables should be explicitly declared (using `let` or `const`)

**Python**

```python
hourly_pay = 9.5
hours_worked = 35
total_pay = hourly_pay * hours_worked
print(total_pay)
```

```python
# Hurray! A pay rise:
hourly_pay = 10.5
total_pay = hourly_pay * hours_worked
print(total_pay)
```

**JavaScript**

```javascript
let hourly_pay;
let hours_worked;
let total_pay;
hourly_pay = 9.5;
hours_worked = 35;
total_pay = hourly_pay * hours_worked;
console.log(total_pay);
```

```javascript
// Hurray! A pay rise:
hourly_pay = 10.5;
total_pay = hourly_pay * hours_worked;
console.log(total_pay);
```

**Variables, again**

But JavaScript does allow you to combine variable declaration with initialization

**Python**

```python
hourly_pay = 9.5
hours_worked = 35
total_pay = hourly_pay * hours_worked
print(total_pay)
```

```python
# Hurray! A pay rise:
hourly_pay = 10.5
total_pay = hourly_pay * hours_worked
print(total_pay)
```

**JavaScript**

```javascript
let hourly_pay;
let hours_worked;
let total_pay;
hourly_pay = 9.5;
hours_worked = 35;
total_pay = hourly_pay * hours_worked;
console.log(total_pay);
```

```javascript
// Hurray! A pay rise:
hourly_pay = 10.5;
total_pay = hourly_pay * hours_worked;
console.log(total_pay);
```

**The beginnings of particles.js**

```javascript
let canvas;
let context;
let width;
let height;
document.addEventListener('DOMContentLoaded', init, false);

function init() {
  canvas = document.querySelector('canvas');
  context = canvas.getContext('2d');
  width = canvas.width;
  height = canvas.height;
}
```
Comments

- Single-line comments
  - Python
    ```python
    # This is a comment
    x = 3
    # This is also a comment
    ```
  - JavaScript
    ```javascript
    // This is a comment
    let x = 3; // This is also a comment
    ```

- Multiline comments
  - Python
    ```python
    ""
    This is another comment.
    It extends over more than one line.
    ""
    ```
  - JavaScript
    ```javascript
    ```
    This is another comment.
    It extends over more than one line.
    ```

Q: How do you make comments in HTML? CSS? SQL?

Function definitions

- Python
  ```python
  def print_1_to_n(n):
    for i in range(1, n+1):
      print(i)
  print_1_to_n(10)
  ```

- JavaScript
  ```javascript
  function print_1_to_n(n) {
    for (let i = 1; i <= n; i += 1) {
      console.log(i);
    }
  }
  print_1_to_n(10);
  ```

NB: Python uses indentation to denote blocks of code; JavaScript uses curly braces

Drawing a coloured rectangle

- `context.fillRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle
  - y: The y-coordinate of the upper-left corner of the rectangle
  - width: The width of the rectangle, in pixels
  - height: The height of the rectangle, in pixels

Clearing a rectangle

- `context.clearRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle to clear
  - y: The y-coordinate of the upper-left corner of the rectangle to clear
  - width: The width of the rectangle to clear, in pixels
  - height: The height of the rectangle to clear, in pixels

Drawing a coloured rectangle

- `context.fillRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle
  - y: The y-coordinate of the upper-left corner of the rectangle
  - width: The width of the rectangle, in pixels
  - height: The height of the rectangle, in pixels

Clearing a rectangle

- `context.clearRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle to clear
  - y: The y-coordinate of the upper-left corner of the rectangle to clear
  - width: The width of the rectangle to clear, in pixels
  - height: The height of the rectangle to clear, in pixels

Drawing a coloured rectangle

- `context.fillRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle
  - y: The y-coordinate of the upper-left corner of the rectangle
  - width: The width of the rectangle, in pixels
  - height: The height of the rectangle, in pixels

Clearing a rectangle

- `context.clearRect(x, y, width, height)`
  - x: The x-coordinate of the upper-left corner of the rectangle to clear
  - y: The y-coordinate of the upper-left corner of the rectangle to clear
  - width: The width of the rectangle to clear, in pixels
  - height: The height of the rectangle to clear, in pixels
Calling a function at a fixed interval

- `window.setInterval(function, milliseconds)`

<table>
<thead>
<tr>
<th>function</th>
<th>The function that will be executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliseconds</td>
<td>The interval between calls to the function</td>
</tr>
</tbody>
</table>